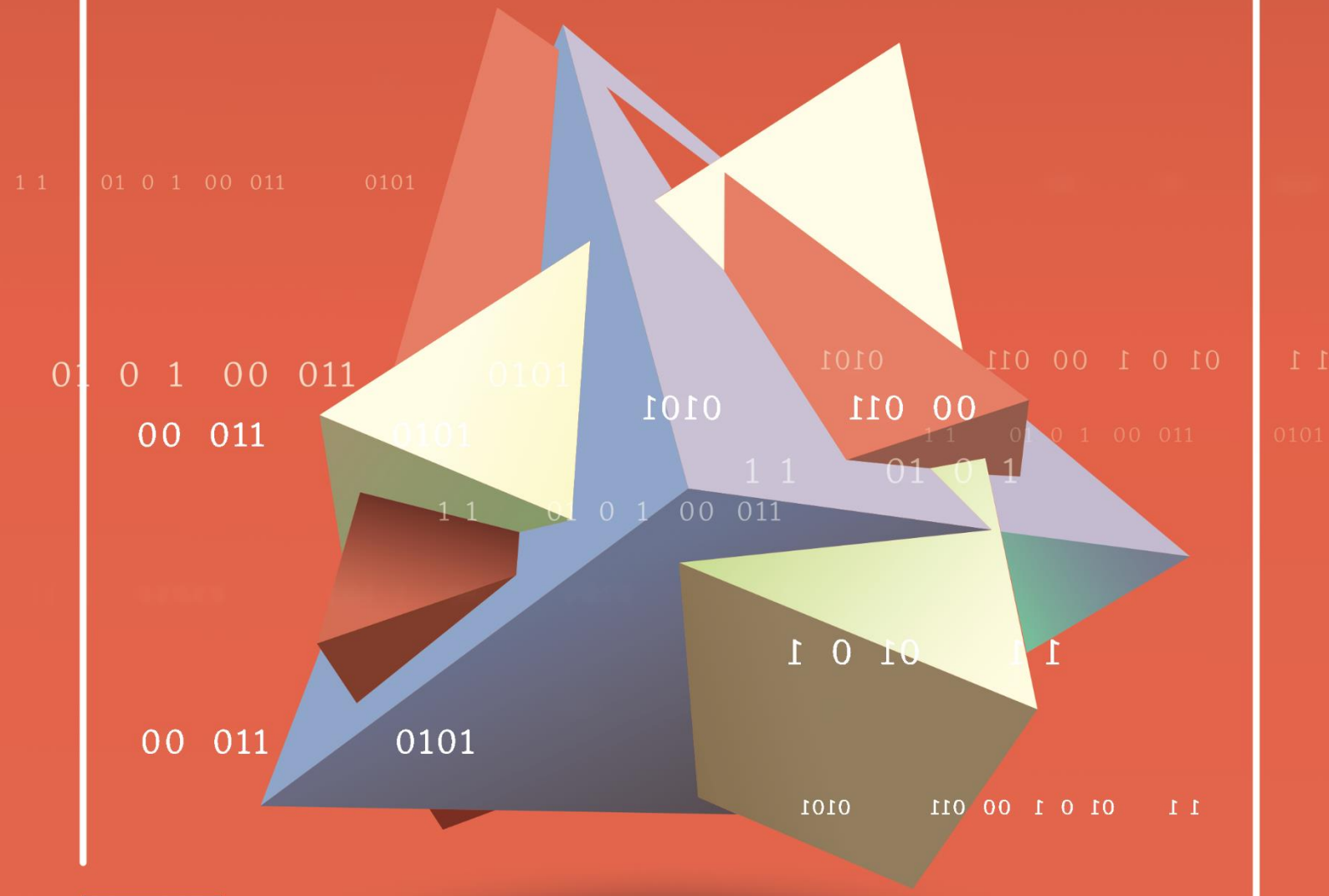
Συστήματα Βάσεων Δεδομένων

Χρήστος Σκουρλάς
Ομότιμος καθηγητής ΠΑΔΑ



ΚΑΛΛΙΠΟΣ
ανοικτές ακαδημαϊκές εκδόσεις

ΑΘΗΝΑ 2022

ΧΡΗΣΤΟΣ ΣΚΟΥΡΛΑΣ
Ομότιμος Καθηγητής Πανεπιστημίου Δυτικής Αττικής

Συστήματα βάσεων δεδομένων

Συστήματα βάσεων δεδομένων
Τόμος Α

Συγγραφή
Χρήστος Σκουρλάς

Συντελεστές έκδοσης
Γλωσσική Επιμέλεια: Όνομα (style: Sintelestes)
Γραφιστική Επιμέλεια: Όνομα (style: Sintelestes)

ΚΑΛΛΙΠΟΣ

Εθνικό Μετσόβιο Πολυτεχνείο

Ηρώων Πολυτεχνείου 9, 15780 Ζωγράφου

www.kallipos.gr

ISBN:

Βιβλιογραφική Αναφορά: Σκουρλάς, Χ. (2022). Συστήματα βάσεων δεδομένων Τόμος Α

[Προπτυχιακό Εγχειρίδιο]. Αθήνα: Κάλλιπος, Ανοικτές Ακαδημαϊκές Εκδόσεις.

Στη Ρούλα, στη Μελίνα

Πίνακας Περιεχομένων

Ευρετήριο Εικόνων

Ευρετήριο Πινάκων

Πρόλογος

Ο τομέας των βάσεων δεδομένων παρουσιάζει μεγάλο ερευνητικό ενδιαφέρον και ταυτόχρονα έχει πάρα πολλές εφαρμογές εδώ και πολλά χρόνια. Η οικονομία και οι επιχειρήσεις, η επιστημονική έρευνα, οι επιστημονικές-τεχνολογικές εφαρμογές βασίζονται σε βάσεις δεδομένων. Τα συστήματα βάσεων δεδομένων είναι καλά εδραιωμένα και απαραίτητα σε κοινωνία, οικονομία, έρευνα, εκπαίδευση αλλά και στην καθημερινή ζωή. Κάθε καινοτομία στον τομέα των υπολογιστών και της πληροφορικής συνεπάγεται μια θεαματική επίδραση στον τομέα των εφαρμογών των βάσεων δεδομένων. Είναι γνωστό, τέλος, ότι η σχετική βιβλιογραφία είναι εκτενέστατη και πλουτίζεται καθημερινά με εντυπωσιακό ρυθμό. Υπάρχουν πολλά άρθρα επιστημονικών περιοδικών, ανακοινώσεις σε πρακτικά συνεδρίων και βιβλία, μονογραφίες, συγγράμματα κ.λπ. που προσφέρουν πολλά στοιχεία και διαφορετικές οπτικές γωνίες σε θέματα που συμπεριλάβαμε στο παρόν σύγγραμμα μας και σε άλλα. Επιπλέον, υπάρχει εκτενέστατη βιβλιογραφία σε θέματα προγραμματισμού προϊόντων γνωστών προμηθευτών ΣΔΒΔ.

Το βιβλίο απευθύνεται σε φοιτητές και φοιτήτριες τμημάτων Πληροφορικής και σε μηχανικούς πληροφορικής που θέλουν να σχεδιάσουν και να υλοποιήσουν συστήματα βάσεων δεδομένων και σύνθετες εφαρμογές βάσεων δεδομένων. Η συλλογή και η επεξεργασία του υλικού που περιλαμβάνεται στο σύγγραμμα αντανακλά επαγγελματικές και εκπαιδευτικές δραστηριότητές μας που καλύπτουν περισσότερα από σαράντα έτη. Κατά κύριο λόγο η κατεύθυνση που υιοθετείται συνδυάζει επισκόπηση και συζήτηση βασικών εννοιών, πολλά ενδιαφέροντα παραδείγματα και εφαρμογές που "οικοδομούνται" στα δημοφιλή προϊόντα Oracle, MySQL. Το εκπαιδευτικό υλικό δοκιμάστηκε με επιτυχία σε εκπαιδευτική διαδικασία, από φοιτητές και φοιτήτριες αλλά και προγραμματιστές και ελπίζουμε ότι στο μέλλον θα συνεχίσει να εμπλουτίζεται σε διάφορες κατευθύνσεις.

Θα θέλαμε να εκφράσουμε τις ευχαριστίες μας:

- Στη Ρούλα Χριστοπούλου και στη Μελίνα Σκουρλά για την πολυετή συμπαράσταση, την κατανόηση και την υπομονή τους
- Στον Θόδωρο Αλεβίζο, ομότιμο καθηγητή στο Διεθνές Πανεπιστήμιο, για τις πολλές συζητήσεις μας, τις παρατηρήσεις και υποδείξεις του και τόσα άλλα. Ευχαριστίες και για την ευγενή παραχώρηση του υλικού του για κανονικοποίηση με χρήση συναρτησιακών εξαρτήσεων.
- Στον Βασίλη Τσουκαλά, MSc, ΕΤΕΠ στο Διεθνές Πανεπιστήμιο, για τις πολλές συζητήσεις μας και τις παρατηρήσεις του. Ευχαριστίες και για την ευγενή παραχώρηση του υλικού του για τη μελέτη περίπτωσης του κεφαλαίου 10 και τις οδηγίες εγκατάστασης του προϊόντος της Oracle.
- Στον δρ. Τάσο Τσολακίδη για τη συνεργασία μας, τις πολλές συζητήσεις μας και τις παρατηρήσεις του. Ευχαριστίες και για τη βοήθειά του στη συγγραφή των κεφαλαίων 11 και 12.
- Στον υποψήφιο διδάκτορα Κώστα Χύτα, MSc, και στον Νίκο Λαζαρίδη, Msc, ΕΔΙΠ στο Πανεπιστήμιο Δυτικής Αττικής, για τη συνεργασία μας, τις πολλές συζητήσεις μας και την πολύτιμη βοήθεια τους στην ολοκλήρωση του παρόντος.
- Στον δρ. Πέτρο Μπέλση για τη συνεργασία μας, τις πολλές συζητήσεις μας, τη βοήθειά του στη συγγραφή του κεφαλαίου 2 και την ευγενή παραχώρηση υλικού του για οντολογία ασφάλειας συστημάτων.
- Στην Αθηνά Μουντζούρη για τη δημιουργική εργασία της στο εξώφυλλο και το οπισθόφυλλο του βιβλίου.

- Στους συναδέλφους Νικήτα Καρανικόλα, Αικατερίνη Μαρινάγη, Ελένη Γαλιώτου, Δημήτρη Μάγο, Γιάννη Χάλαρη, Δημήτρη Δέρβο, Martti Leiho, Βασίλη Μάμαλη και Νίκο Βασιλά για τη συνεργασία μας σε θέματα του παρόντος συγγράμματος.
- Στους φοιτητές μου και στις φοιτήτριες μου, για τις ερωτήσεις τους, τις διορθώσεις και υποδείξεις τους. Ευχαριστίες ειδικότερα στους Μάρκο Λινάρδο για τη συμβολή του στην ενότητα 12.2.4 και στη Μαρία Χριστοδουλάκη για τη συμβολή της στην παρουσίαση των NoSQL βάσεων δεδομένων.

Αθήνα 2022

Εισαγωγή

Η διαχείριση βάσης δεδομένων, ως κλάδος, γνωρίζει μεγάλη δημοτικότητα μεταξύ επαγγελματιών και ακαδημαϊκών εδώ και πολλά χρόνια. Κάθε καινοτομία στην πληροφορική και τις εφαρμογές της στηρίχτηκε και στηρίζεται στις εφαρμογές των βάσεων δεδομένων και τα συστήματα βάσεων δεδομένων αξιοποιούν κάθε νέα καινοτομία για να εξελιχθούν και να συνεισφέρουν με νέες εφαρμογές στον κόσμο των επιχειρήσεων αλλά και στη διευκόλυνση της καθημερινής ζωής και στη βελτίωση της. Η γενικευμένη χρήση του «εργαλείου» που ονομάζουμε τεχνολογία βάσεων δεδομένων βοηθά ιδιαίτερα και στην κατανόηση των πιθανών επιχειρηματικών προκλήσεων που εισάγει κάθε νέα τεχνολογική εξέλιξη (και καινοτομία) και αποτελεί το υπόβαθρο στο οποίο θα βασιστεί η εφαρμογή της. Στο πλαίσιο της οικονομίας της γνώσης, έννοιες και όροι όπως «διαχείριση δεδομένων μεγάλης κλίμακας» αλλάζουν ριζικά την οπτική μας σε πάρα πολλά ζητήματα. Η ανάγκη αξιοποίησης των «δεδομένων μεγάλης κλίμακας» προβάλλει επιτακτικά για τη σύγχρονη επιχείρηση και όχι μόνο.

Το ερέθισμα για αυτό το σύγγραμμα προήλθε από το γεγονός ότι υπάρχει ανάγκη, «έν τινι μέτρω», για ένα ηλεκτρονικό εγχειρίδιο το οποίο θα περιλαμβάνει: 1) μία παρουσίαση και συζήτηση των «θεωρητικών» ζητημάτων, 2) μια μελέτη του προγραμματισμού βάσεων δεδομένων με ενδιαφέροντα παραδείγματα και 3) μια παράθεση μελετών περιπτώσεων η οποία να χαρακτηρίζεται από σφαιρική και όσο γίνεται διαθεματική προσέγγιση στα συστήματα βάσεων δεδομένων. Απευθύνεται σε προπτυχιακούς φοιτητές και φοιτήτριες και επαγγελματίες μηχανικούς πληροφορικής που θέλουν να αναπτύξουν τις δεξιότητές τους σε θέματα σχεδιασμού και προγραμματισμού εφαρμογών βάσεων δεδομένων αλλά και σε μεταπτυχιακούς φοιτητές και φοιτήτριες διοίκησης επιχειρήσεων, βιβλιοθηκονομίας και αρχειονομίας και πληροφόρησης, αλλά και άλλων τομέων οι οποίοι ενδιαφέρονται για τις εφαρμογές των συστημάτων βάσεων δεδομένων. Στοχεύει επίσης στην επίλυση προβλημάτων διαχείρισης δεδομένων και στην κάλυψη αναγκών των επαγγελματιών του τομέα.

Στον τομέα των συστημάτων βάσεων δεδομένων υπάρχουν δύο τουλάχιστον κυρίαρχες διαστάσεις:

1) Βασική και τεχνολογική έρευνα, με πάρα πολλά ευρήματα και γνώσεις ως αποτέλεσμα της συνεχούς πολυετούς δραστηριότητας του ακαδημαϊκού χώρου, των ερευνητικών κέντρων αλλά και των τμημάτων έρευνας μεγάλων προμηθευτών (vendors) συστημάτων διαχείρισης βάσης δεδομένων. Ο τομέας των βάσεων δεδομένων δεν θα υπήρχε στην παρούσα μορφή χωρίς την έρευνα και το συγγραφικό έργο πρωτοπόρων όπως οι βραβευμένοι με βραβείο Turing, Edgar F. Codd, Jim Gray, Michael Stonebraker, Charles Bachman, Jeffrey Ullman, την έρευνα και το συγγραφικό έργο των Peter Chen, Chris Date, James Martin και τόσων άλλων που σφράγισαν τον τομέα ως οραματιστές, ερευνητές, σύμβουλοι κ.λπ. Επισημαίνουμε ότι μεγάλες επιτεύξεις της τεχνολογίας των βάσεων δεδομένων είχαν ως αφετηρία τα ερευνητικά κέντρα προμηθευτών συστημάτων διαχείρισης βάσης δεδομένων, π.χ. οι σχεσιακές βάσεις δεδομένων.

2) Αξιοποίηση και αποτύπωση (codification) της τεχνογνωσίας, της εμπειρίας αλλά και του πειραματισμού με τα συστήματα και τα εργαλεία των βάσεων δεδομένων σπουδαίων επιστημόνων και επαγγελματιών (practitioner) όπως καταγράφεται σε πρότυπα (standard), διπλώματα ευρεσιτεχνίας, εγχειρίδια προμηθευτών (manual), αναφορές-εκτυπωτικά (reports), white papers, βιβλία σχεδίασης και προγραμματισμού εφαρμογών βάσεων δεδομένων, υλικό σεμιναρίων (συνήθως πια σε βίντεο) και blogs. Είναι ενδιαφέρον ότι εκατομμύρια επιστήμονες και επαγγελματίες προγραμματιστές εφαρμογών βάσεων δεδομένων συλλειτουργούν εθελοντικά σε πάρα πολλά φόρουμ (forum), σε ένα είδος παγκόσμιας κοινότητας πρακτικής (community of practice), για ανταλλαγή απόψεων και επίλυση προβλημάτων.

Υπάρχουν πάρα πολλά σημαντικά συγγράμματα και «απειράριθμα» αξιόλογα εγχειρίδια και βιβλία που εξετάζουν συνήθως μία από τις παραπάνω κατευθύνσεις, αλλά κατά κανόνα στα βιβλία αυτά υπάρχει σχετικά μικρή διασταύρωση μεταξύ των δύο διαστάσεων-κατευθύνσεων στις οποίες αναφερθήκαμε και παρά το γεγονός ότι πάρα πολλοί επιστήμονες/ερευνητές ασχολούνται με ποικίλα θέματα και στις δύο κατευθύνσεις. Σκοπός του εγχειριδίου αυτού είναι η συνδυασμένη προσέγγιση, έτσι ώστε η δυναμική της διαχείρισης βάσης δεδομένων να αναδειχθεί σε σωστές διαστάσεις και να καλυφθεί ένα μέρος του εύρους του σχετικού υλικού. Αντικείμενο του βιβλίου αποτελούν όχι μόνο η μελέτη και αξιοποίηση εδραιωμένων κατευθύνσεων, εργαλείων και τεχνολογιών τα οποία είναι δημοφιλή στο πλαίσιο των συστημάτων βάσεων δεδομένων αλλά και οι νέες πολλά υποσχόμενες τεχνολογίες, όπως η διαχείριση μεγάλων δεδομένων

Στο ηλεκτρονικό βιβλίο περιλαμβάνονται δώδεκα κεφάλαια τα οποία γράφτηκαν έτσι ώστε να μπορούν να διαβαστούν κάθε ένα ξεχωριστά, ανεξάρτητα από τα υπόλοιπα. Καταβλήθηκε προσπάθεια έτσι ώστε ακόμη και οι ενότητες των κεφαλαίων, τα παραδείγματα και τα θέματα να μπορούν να μελετηθούν (και να δοκιμαστούν σε υπολογιστή) αυτόνομα.

Κεφάλαιο 1. Εισαγωγή στις έννοιες της τεχνολογίας βάσεων δεδομένων»

Παρατίθεται μία εισαγωγή στις έννοιες της τεχνολογίας βάσεων δεδομένων, παρουσιάζονται εισαγωγικά θέματα σχεδίασης και υλοποίησης σχεσιακών βάσεων δεδομένων, γίνεται μία εισαγωγή στη χρήση της δομημένης γλώσσας ερωτημάτων SQL-Structured Query Language, και σκιαγραφείται ο σύγχρονος τομέας των συστημάτων βάσεων δεδομένων και των εφαρμογών τους. Όλα τα εισαγωγικά θέματα του κεφαλαίου παρουσιάζονται σε βάθος σε επόμενα κεφάλαια του βιβλίου. Το κεφάλαιο περιλαμβάνει τρεις κύριες ενότητες:

1) Εισαγωγή σε έννοιες (κυρίως) των σχεσιακών βάσεων δεδομένων, χωρίς μαθηματικό φορμαλισμό.
2) Στοιχεία της μοντελοποίησης βάσης δεδομένων και
3) Στοιχεία της κανονικοποίησης βάσης δεδομένων.

Κεφάλαιο 2. Τύποι συστημάτων βάσεων δεδομένων, σύγχρονες τεχνολογίες και εφαρμογές. Διαχείριση δεδομένων μεγάλης κλίμακας

Σκιαγραφείται ο σύγχρονος τομέας των συστημάτων βάσεων δεδομένων και των εφαρμογών τους. Παρουσιάζονται θέματα διαχείρισης και αξιοποίησης δομημένων και μη δομημένων δεδομένων και γίνεται αναφορά στα ανοικτά δεδομένα, στα συνδεδεμένα δεδομένα και κυρίως στα δεδομένα μεγάλης κλίμακας (big data). Παρουσιάζονται τα συστήματα διαχείρισης δεδομένων, «παραδοσιακά» και νεότερα, και το πως επηρεάστηκαν και επηρεάζονται από τις τεχνολογικές εξελίξεις. Στο κεφάλαιο περιλαμβάνονται τέσσερις ενότητες:

1) Αρχιτεκτονικές εφαρμογών βάσης δεδομένων και συστήματα διαχείρισης συναλλαγών (online transaction processing) και η σημασία τους για τη λειτουργία της σύγχρονης επιχείρησης. Παρουσιάζονται προβλήματα ταυτοχρονισμού (concurrency problems), οι ιδιότητες ACID (ACID principle) και τα επίπεδα απομόνωσης (isolation levels) συναλλαγών σύμφωνα με το πρότυπο ISO ANSI SQL.
2) Ο ρόλος του Σημασιολογικού ιστού στο παρόν και το μέλλον των συστημάτων βάσεων δεδομένων, αναφορά σε μεταδεδομένα, πρότυπα, γλώσσες σήμανσης, XML, JSON, οντολογίες κ.λπ., οι βάσεις δεδομένων πολυμέσων και συστήματα ανάκτησης βασισμένα στο περιεχόμενο (multimodal database, content based information retrieval system), τα συστήματα ανάκτησης πληροφοριών (information retrieval system) και τα συστήματα ανάκτησης κειμένου (text retrieval) και οι μηχανές αναζήτησης,
3) Τεχνολογίες που αξιοποιούνται και στο πλαίσιο της διαχείρισης δεδομένων μεγάλης κλίμακας σε δύο κατευθύνσεις: (α) τεχνολογίες με εφαρμογές στην επιχειρηματική λειτουργία, όπως, κατανεμημένες

βάσεις δεδομένων (distributed databases), βάσεις στο υπολογιστικό νέφος (cloud databases), ενεργές βάσεις δεδομένων (active databases) και προγραμματισμός με χρήση εναυσμάτων (trigger), και (β) τεχνολογίες με εφαρμογές στην ανάλυση δεδομένων και τη στήριξη επιχειρηματικών αποφάσεων, όπως, Data Mining, Data Warehouse, Datamart, Business Intelligence, Knowledge Management

4) Τεχνολογίες δεδομένων μεγάλης κλίμακας, βάσεις στο νέφος (cloud databases), εργαλεία και αρχιτεκτονικές διαχείρισης δεδομένων. Ενσωμάτωση δομημένων και μη δομημένων δεδομένων (integration of structured and unstructured data) και ο ρόλος των Map Reduce, Hadoop. Τεχνολογίες NoSQL βάσεων δεδομένων. Σκιαγράφηση του προϊόντος MySQL που συνδυάζει τεχνολογίες σχεσιακής βάσης δεδομένων και αποθήκης εγγράφων (document store.

Κεφάλαιο 3. Υλοποίηση σχεσιακών βάσεων δεδομένων. Η γλώσσα SQL (Structured Query Language)

Παρουσιάζονται θέματα υλοποίησης σχεσιακών βάσεων δεδομένων με χρήση της δομημένης γλώσσας ερωτημάτων SQL (Structured Query Language). Περιγράφονται η γλώσσα SQL σύμφωνα με τα πρότυπα ANSI και οι τρεις υπογλώσσες της. Στο κεφάλαιο περιλαμβάνονται οι παρακάτω ενότητες:

1) Σύνταξη δηλώσεων ορισμού δεδομένων, δηλώσεων επεξεργασίας δεδομένων, απλών δηλώσεων αναζήτησης δεδομένων, και δηλώσεων ελέγχου δεδομένων με χρήση των προϊόντων Oracle SQL, MySQL.

2) Σύνθετες αναζητήσεις δεδομένων (SELECT). Συναρτήσεις και η συνηθισμένη σύνταξή τους. Αναζήτηση με τους γνωστούς από τη θεωρία συνόλων τελεστές UNION, INTERSECT, MINUS. Εισαγωγή στη δημιουργία, χρήση και ενημερωσιμότητα όψεων (view).

Κεφάλαιο 4. Περιηγήσεις στην υλοποίηση σχεσιακών βάσεων δεδομένων με γλώσσα SQL (Tours on Structured Query Language)

Παρουσιάζονται θέματα υλοποίησης σχεσιακών βάσεων δεδομένων με χρήση γλώσσας SQL (Structured Query Language). Υιοθετείται η προσέγγιση της «περιήγησης» με την έννοια του διαλόγου του προγραμματιστή με το ΣΔΒΔ. Ο προγραμματιστής θέτει ερωτήματα (queries) προς εκτέλεση και το ΣΔΒΔ εκτελεί και απαντά. Οι περιηγήσεις αποσκοπούν στην εξοικείωση των αναγνωστών με τη γλώσσα SQL. Περιλαμβάνονται οι παρακάτω περιηγήσεις:

1) Διαχείριση βάσης δεδομένων με τη γλώσσα Oracle SQL.

2) Αναλυτική παρουσίαση συνδέσεων (join) πινάκων, φωλιασμένων-εμφωλευμένων αναζητήσεων (embedded query), χρήσης συναρτήσεων (function) και ομαδοποίησης δεδομένων (GROUP BY) σε Oracle και MySQL.

3) Περαιτέρω συζήτηση και εμβάθυνση στη χρήση σημαντικών για τις εφαρμογές δηλώσεων SELECT σε Oracle και MySQL.

Κεφάλαιο 5. Μελέτη περίπτωσης. Πληροφοριακό σύστημα νοσοκομείων. Υλοποίηση σχεσιακής βάσης δεδομένων με τη γλώσσα SQL (Structured Query Language). Σχεδίαση της διεπαφής χρήστη.

Στη Μελέτη Περίπτωσης προσεγγίζεται σφαιρικά ένα σύστημα βάσης δεδομένων μεγαλύτερης κλίμακας και περιγράφεται ο σχεδιασμός και η υλοποίησή του. Παρατίθεται περιγραφή του συστήματος, ανάλυση δεδομένων (data analysis), μοντελοποίηση και κανονικοποίηση. Στη συνέχεια παρατίθενται δηλώσεις για τη δημιουργία και την αξιοποίηση του συστήματος. Τέλος, παρατίθεται σχεδίαση της διεπαφής του χρήστη.

Κεφάλαιο 6: Σχεδίαση βάσεων δεδομένων και συστημάτων βάσεων δεδομένων. Μοντελοποίηση. Κανονικοποίηση. Χρήση τεχνολογίας πληροφοριακών συστημάτων

Παρουσιάζονται εκτενέστερα η σχεδίαση βάσεων δεδομένων. Παρατίθεται μια περιεκτική συζήτηση των θεμάτων της μοντελοποίησης βάσης δεδομένων και στη συνέχεια παρουσιάζονται αναλυτικά οι σχεσιακές βάσεις δεδομένων και χρήσιμα θέματα κανονικοποίησης της βάσης δεδομένων. Στο κεφάλαιο περιλαμβάνονται οι παρακάτω ενότητες:

1) Εννοιολογική μοντελοποίηση (conceptual modeling). Σημασιολογικά μοντέλα. Μοντέλο Οντοτήτων–Συσχετίσεων. Επεκτάσεις μοντέλου οντοτήτων-συσχετίσεων.

2) Σχεσιακές βάσεις δεδομένων. Κανονικοποίηση. Μέθοδος Συναρτησιακών Εξαρτήσεων. Πρώτη Κανονική Μορφή. Δεύτερη Κανονική Μορφή. Τρίτη Κανονική Μορφή. Μέθοδος Συναρτησιακών Εξαρτήσεων και Κανονική Μορφή Boyce-Codd. Άλλες Κανονικές Μορφές. Μετασχηματισμός μοντέλου οντοτήτων συσχετίσεων σε σχεσιακή βάση δεδομένων. Εμβάθυνση στη μοντελοποίηση και την κανονικοποίηση. Ενοποίηση διαφορετικών συστημάτων βάσεων δεδομένων. Ενιαίο παράδειγμα σχεδίασης σχεσιακής βάσης δεδομένων και υλοποίησης με γλώσσα SQL.

3) Μελέτη Περίπτωσης εταιρείας οργάνωσης και διεξαγωγής σεμιναρίων. Ανάλυση και σχεδιασμός συστήματος βάσης δεδομένων με χρήση τεχνολογίας πληροφοριακών συστημάτων. Παράλληλα με τη διεκπεραίωση του θέματος, οι αναγνώστες μπορούν να μελετήσουν τον τρόπο εφαρμογής της τεχνολογίας πληροφοριακών συστημάτων, μεθοδολογίες και εργαλεία. Καλύπτονται θέματα όπως, προμελέτη, μελέτη σκοπιμότητας, προδιαγραφές διαγωνισμού, δομημένες συνεντεύξεις και ερωτηματολόγια, καταγραφή αποτελεσμάτων συνεντεύξεων, συλλογή εντύπων και σχεδιασμός μοντέλων δεδομένων, σχεδιασμός βάσης δεδομένων.

Κεφάλαιο 7: Ασφάλεια Βάσης Δεδομένων. Διαχειριστής Βάσης Δεδομένων (Data Base Administrator), Γλώσσα Ελέγχου Δεδομένων (Data Control Language). Όψεις (Views). Παραδείγματα στα προϊόντα Oracle και MySQL

Περιγράφονται σημαντικές έννοιες ελέγχου δεδομένων, η χρήση του μηχανισμού της όψης, η σημασία των δηλώσεων της γλώσσας ελέγχου δεδομένων (Data Control Language), και ο ρόλος του διαχειριστή βάσης δεδομένων στη διασφάλιση της συνέπειας (consistency), της ασφάλειας (security) και της ακεραιότητας (integrity) των δεδομένων. Στο κεφάλαιο περιλαμβάνονται οι παρακάτω ενότητες:

1) Ο Ρόλος και τα καθήκοντα του Διαχειριστή Βάσεων Δεδομένων. Γλώσσα Ελέγχου Δεδομένων. Συναλλαγές. Δηλώσεις COMMIT, ROLLBACK της Γλώσσας SQL. Όψεις (Views). Ενημερωσιμότητα όψεων.

2) Περιήγηση στη διαχείριση της ασφάλειας ενός συστήματος βάσης δεδομένων προσωπικού, Χρήση ORACLE SQL και MySQL

Κεφάλαιο 8: Προγραμματισμός εφαρμογών βάσεων δεδομένων. Εναύσματα (trigger)

Περιγράφονται αναλυτικά σημαντικές έννοιες και εργαλεία του προγραμματισμού εφαρμογών βάσεων δεδομένων. Περιορισμοί (constraints) και όψεις (views) ως δομικά στοιχεία για μια στρατηγική υλοποίησης που επιτρέπει να διασφαλίσουμε τη συνέπεια (consistency), την ασφάλεια (security) και την ακεραιότητα (integrity) της βάσης δεδομένων. Ο προγραμματισμός εναυσμάτων (trigger) και οι ενεργές βάσεις δεδομένων (Active databases) σε περιβάλλον MySQL και Oracle PL/SQL.

Το κεφάλαιο περιλαμβάνει τις παρακάτω ενότητες που είναι οργανωμένες σαν «περιηγήσεις» (tours):

1) Περιήγηση στους περιορισμούς (a guided tour on SQL Constraints)

2) Περιήγηση στις όψεις (a guided tour on SQL View)

3) Triggers by example. Χρήση τεχνολογίας PL/SQL. Προγραμματισμός εναυσμάτων (trigger) στο προϊόν MySQL. Διαχείριση παραγγελιών με χρήση εναυσμάτων (trigger) σε περιβάλλον Oracle PL/SQL. Διαχείριση παραγγελιών στο προϊόν MySQL

Κεφάλαιο 9: Προγραμματισμός εφαρμογών βάσεων δεδομένων. Stored procedures. Procedures, functions, triggers, cursors σύμφωνα με το πρότυπο SQL PSM

Γίνεται αναφορά στο ρόλο των αποθηκευμένων διαδικασιών (stored procedures) και ειδικότερα στη συγγραφή και χρήση εναυσμάτων (triggers), διαδικασιών (procedures), συναρτήσεων (functions) και cursors. Η παρουσίαση ακολουθεί το πρότυπο Persistent Stored Modules (SQL PSM standard) αλλά, επιπλέον, παρουσιάζεται και η τεχνολογία Oracle PL/SQL. Το κεφάλαιο περιλαμβάνει τις παρακάτω ενότητες που είναι οργανωμένες σαν «περιηγήσεις» (tours):

1) Προγραμματισμός και χρήση εναυσμάτων (Trigger) για την υλοποίηση περιορισμών-επιχειρησιακών κανόνων (constraints-business rules).

2) Προγραμματισμός trigger, function, procedure, cursor σε διάφορα προϊόντα διαχείρισης βάσεων δεδομένων

3) Διαχείριση προβλημάτων από πλεονάζοντα δεδομένα, διαχείριση ακεραιότητας και συνέπειας δεδομένων και καταγραφή ενεργειών χρήστη με εναύσματα (trigger).

Κεφάλαιο 10: Μελέτη του προϊόντος διαχείρισης βάσης δεδομένων της Oracle. Τεχνολογία Oracle PL/SQL. Μελέτη περίπτωσης ανάπτυξης εφαρμογής λογισμικού με χρήση τεχνολογίας Oracle PL/SQL και γλώσσας προγραμματισμού JAVA

Παρατίθεται περιεκτική σκιαγράφηση συνιστωσών του προϊόντος της Oracle και επισκόπηση της φιλοσοφίας κατασκευής εφαρμογών που το συνοδεύει. Σκιαγραφούνται το περιβάλλον (συνιστώσα) SQL*PLUS, δηλαδή το περιβάλλον χρησιμοποίησης της SQL, και η γλώσσα (τεχνολογία) ORACLE PL/SQL. Παρατίθεται η υλοποίηση εφαρμογής διαχείρισης βάσης δεδομένων που χρησιμοποιεί τη γλώσσα προγραμματισμού JAVA, το λογισμικό ανάπτυξης εφαρμογών Apache Netbeans IDE και το σύστημα διαχείρισης βάσεων δεδομένων της εταιρείας ORACLE.

Κεφάλαιο 11: Βάσεις δεδομένων στο διαδίκτυο. Προγραμματισμός Web εφαρμογών με χρήση τεχνολογιών HTML, PHP και MySQL

Περιγράφονται έννοιες και εργαλεία του προγραμματισμού Web εφαρμογών βάσεων δεδομένων για χρήση σε δυναμικούς ιστότοπους στο διαδίκτυο. Επεξηγούνται οι τεχνολογίες HTML και PHP. Παρατίθενται εφαρμογές βάσεων δεδομένων με χρήση PHP και διεπαφής προγραμματισμού εφαρμογών API. Το κεφάλαιο περιλαμβάνει τις παρακάτω ενότητες:

1) Εισαγωγή στις τεχνολογίες HTML και δημιουργία ιστοτόπου Edgar Frank "Ted" Codd Fun Club με χρήση HTML και του εργαλείου Netbeans.

2) Εφαρμογές βάσεων δεδομένων με χρήση PHP και διεπαφής προγραμματισμού εφαρμογών (Application Programming Interface) API.

3) Παρουσιάζεται η έννοια των διεπαφών προγραμματισμού εφαρμογών API (Application Programming

Κεφάλαιο 12: Βάσεις δεδομένων στο διαδίκτυο. Προγραμματισμός Web εφαρμογών με χρήση τεχνολογιών JSP pages, JDBC API και MySQL

Παρατίθενται και επεξηγούνται εφαρμογές δυναμικών σελίδων JSP Pages με χρήση JDBC API. Περιλαμβάνονται οι παρακάτω ενότητες:

1) Εισαγωγή στις δυναμικές σελίδες Java Server Pages (JSP) και στη σύνδεση δυναμικών σελίδων JSP και βάσης δεδομένων. Κατασκευή απλοποιημένου Calculator και Login.
2) Εισαγωγή στις εφαρμογές Δυναμικών Ιστοσελίδων (JSP pages, mySQL). Υλοποίηση συστήματος πωλήσεων της εταιρείας Mythical Car.
3) Επισκόπηση της χρήσης JDBC API. Μελέτη Περίπτωσης. Διαχείριση προσωπικού με εφαρμογή Δυναμικών Ιστοσελίδων και χρήση τεχνολογίας JSP pages, Netbeans IDE και mySQL. Χρήση εξυπηρετητή ιστοσελίδων Glassfish/Apache.

Προτάσεις πλοήγησης στα κεφάλαια του βιβλίου

Υπάρχουν πολλοί τρόποι διδασκαλίας μαθημάτων βάσεων δεδομένων.

Το κεφάλαιο 1, ενότητες από το κεφάλαιο 2 και τα κεφάλαια 3-6, με τη σειρά που παρατίθενται ή σύμφωνα με την σειρά προτίμησης των διδασκόντων, μπορούν να χρησιμοποιηθούν στο πλαίσιο ενός προπτυχιακού εξαμηνιαίου εισαγωγικού μαθήματος στα συστήματα βάσεων δεδομένων. Τα κεφάλαια 7-12 και ενότητες από το κεφάλαιο 2 μπορούν να χρησιμοποιηθούν στο πλαίσιο ενός δευτέρου εξαμηνιαίου μαθήματος το οποίο εστιάζει σε τεχνικές προγραμματισμού συστημάτων βάσεων δεδομένων. Σημειώνεται ότι τα περιλαμβανόμενα στο βιβλίο γράφτηκαν έτσι ώστε κάθε κεφάλαιο (και σε πολλές περιπτώσεις και κάθε ενότητα κεφαλαίου) να μπορεί να μελετηθεί ανεξάρτητα από τα υπόλοιπα, επομένως η σειρά διδασκαλίας εξαρτάται από τις προτεραιότητες-προτιμήσεις των διδασκόντων. Επιπλέον, τα κεφάλαια του βιβλίου δοκιμάστηκαν στην πολυετή θεωρητική διδασκαλία και την πρακτική άσκηση στο πλαίσιο δύο εξαμηνιαίων μαθημάτων, του εισαγωγικού μαθήματος και του μαθήματος που δίνει έμφαση στη σχεδίαση και τον προγραμματισμό συστημάτων βάσεων δεδομένων.

Στην περίπτωση διδασκαλίας ενός μόνο εξαμηνιαίου προπτυχιακού μαθήματος, το κεφάλαιο 1, ενότητες από το κεφάλαιο 2, το κεφάλαιο 3, και ενότητες από τα κεφάλαια 4-7 μπορούν να αποτελέσουν το αντικείμενο διδασκαλίας. Τα κεφάλαια αυτά δοκιμάστηκαν στη διδασκαλία ενός εξαμηνιαίου μαθήματος στο πλαίσιο μεταπτυχιακών προγραμμάτων σπουδών τα οποία παρακολουθούν πτυχιούχοι θετικών επιστημών.

Τα κεφάλαια 1-2 μπορούν να χρησιμοποιηθούν ως τμήμα της διδασκαλίας μαθήματος για τη διαχείριση δεδομένων μεγάλης κλίμακας. Τα κεφάλαια αυτά δοκιμάστηκαν στη διδασκαλία εξαμηνιαίου μαθήματος στο πλαίσιο μεταπτυχιακών προγραμμάτων σπουδών και συνοδεύτηκαν με την ανάθεση βιβλιογραφικής εργασίας σε θέματα διαχείρισης δεδομένων μεγάλης κλίμακας και την ανάθεση έργου (project) εξόρυξης δεδομένων και υλοποιήσεων με χρήση Hadoop και βάσεων NoSQL.

Κεφάλαιο 1
Εισαγωγή στις έννοιες της τεχνολογίας βάσεων δεδομένων»

Σύνοψη
Στο κεφάλαιο αυτό γίνεται μία εισαγωγή στις έννοιες της τεχνολογίας βάσεων δεδομένων, παρουσιάζονται εισαγωγικά θέματα σχεδίασης και υλοποίησης σχεσιακών βάσεων δεδομένων, γίνεται μία εισαγωγή στη χρήση της Δομημένης Γλώσσας Ερωτημάτων SQL-Structured Query Language, και σκιαγραφείται ο σύγχρονος τομέας των Συστημάτων Βάσεων Δεδομένων και των εφαρμογών τους. Όλα τα εισαγωγικά θέματα του κεφαλαίου παρουσιάζονται σε βάθος σε επόμενα κεφάλαια.

Το κεφάλαιο περιλαμβάνει τις παρακάτω ενότητες:

1. *Εισαγωγή. Πληροφοριακό Σύστημα. Δεδομένα, Πληροφορία και Γνώση. Γνώση και Διαχείριση γνώσης. Επιχείρηση και Οργανισμός και Σύστημα Βάσεων Δεδομένων. Η γλώσσα SQL-Structured Query Language. Συνοπτική ιστορία των βάσεων δεδομένων. Το παρόν και το μέλλον των βάσεων δεδομένων*
2. *Τι είναι Βάση Δεδομένων (Data Base) και Σύστημα Βάσης Δεδομένων (Data Base System) σύμφωνα με τη βιβλιογραφία.*
3. *Σχεσιακό μοντέλο και Σχεσιακές βάσεις δεδομένων. Σύνδεση της έννοιας της μοντελοποίησης και της έννοιας του συστήματος βάσης δεδομένων*
4. *Μία απόπειρα σύνθεσης των ορισμών βάσεων δεδομένων, χρήστες της Βάσης Δεδομένων, και βασικές έννοιες, ακεραιότητα δεδομένων, κ.λπ.*
5. *Πρώτη αναφορά σε μία συστηματική προσέγγιση στη σχεδίαση της σχεσιακής βάσης δεδομένων. Εισαγωγή στη χρήση συναρτησιακών εξαρτήσεων (functional dependencies).*
6. *Κανονικοποίηση*
7. *Μοντελοποίηση – Μοντέλο οντοτήτων συσχετίσεων και προϊόν MySQL Workbench*
8. *Σύστημα Διαχείρισης Βάσεως Δεδομένων. Διαχειριστής συστήματος. Συναλλαγές (transactions) και Ταυτοχρονισμός (concurrency). Ρόλος και καθήκοντα του ΔΒΔ. Ανάκαμψη (recovery).*
9. *Αρχιτεκτονική ANSI/SPARC και υλοποίηση Συστημάτων Βάσης Δεδομένων.*
10. *Γιατί χρησιμοποιούμε τελικά βάση δεδομένων. Η σημασία της μοντελοποίησης των δεδομένων.*
11. *Μοντέλο Οντοτήτων Συσχετίσεων (ΜΟΣ) και Σχεσιακή Βάση Δεδομένων. Πρώτοι κανόνες μετασχηματισμού ΜΟΣ σε βάση δεδομένων.*
12. *Εννοιολογική σχεδίαση και ΜΟΣ.*
13. *Κατασκευή σχεσιακού μοντέλου από ΜΟΣ. Περισσότεροι κανόνες μετασχηματισμού ΜΟΣ σε βάση δεδομένων και παραδείγματα.*
14. *Σχεδίαση Σχεσιακής Βάσης Δεδομένων. Ευρετικοί κανόνες.*
15. *Κανονικοποίηση δεδομένων (Data normalization) και η σημασία της για τις εφαρμογές. Σχεδιασμός Βάσης Δεδομένων και Κανονικοποίηση. Πρώτη, Δεύτερη και Τρίτη Κανονική Μορφή. Κανονική Μορφή Boyce-Codd, Τέταρτη Κανονική Μορφή, Πέμπτη Κανονική Μορφή.*
16. *Κύρια παραδοτέα της Σχεδίασης Βάσης Δεδομένων.*

Στόχοι του κεφαλαίου είναι:

- *Να κατανοήσουμε τις έννοιες βάση δεδομένων (database), σύστημα (και προϊόν) διαχείρισης βάσης δεδομένων (ΣΔΒΔ) (database management systems, DBMS), σύστημα βάσης δεδομένων (database systems).*

- *Να παρουσιάσουμε την αρχιτεκτονική ANSI-SPARC συστημάτων βάσεων δεδομένων.*
- *Να μελετήσουμε μοντέλα δεδομένων. ιστορικά μοντέλα (ιεραρχικό, δικτυωτό). σχεσιακό μοντέλο (relational model). Σχεσιακό-αντικειμενοστραφές μοντέλο (Object-Relational model).*
- *Να παρουσιάσουμε την αρχιτεκτονική εφαρμογών βάσεων δεδομένων και το μοντέλο πελάτη-εξυπηρετητή (client-server).*
- *Να κατανοήσουμε το πως σχεδιάζουμε τη βάση δεδομένων και πως καθορίζουμε ποιες πληροφορίες θα αποθηκεύονται στη βάση.*
- *Να κατανοήσουμε το πως σχεδιάζουμε και υλοποιούμε ένα πληροφοριακό σύστημα (ΠΣ) που βασίζεται σε βάση δεδομένων χρησιμοποιώντας προϊόντα διαχείρισης βάσης δεδομένων και άλλα λογισμικά.*

Προαπαιτούμενη γνώση
Δεν απαιτείται.

1.1 Εισαγωγή

Εδώ και πάρα πολλά χρόνια στο κόσμο της εκπαίδευσης και της έρευνας αλλά και στον κόσμο των επιχειρήσεων και των οργανισμών οι βάσεις δεδομένων (databases) κατέχουν μία σημαντική θέση. μία εισαγωγή στις βάσεις δεδομένων πρέπει να μας βοηθήσει να κατανοήσουμε τη σημασία τους για τη διαχείριση των δεδομένων (data management) και τις επιστημονικές και επιχειρηματικές εφαρμογές. ταυτόχρονα πρέπει να περιγράψει τη σχέση των βάσεων δεδομένων με τα πληροφοριακά συστήματα (information systems) και το λογισμικό εφαρμογών (application software). Στις μέρες μας όλα τα πληροφοριακά συστήματα και όλες οι εφαρμογές λογισμικού χρησιμοποιούν βάση δεδομένων. Επιπλέον, το εισαγωγικό κεφάλαιο πρέπει να μας βοηθήσει να κατανοήσουμε τις έννοιες της τεχνολογίας των βάσεων δεδομένων και τον τρόπο χρησιμοποίησης των προϊόντων διαχείρισης βάσης δεδομένων (database management products), για να σχεδιάσουμε, να υλοποιήσουμε και να λειτουργήσουμε συστήματα βάσης δεδομένων (database systems).

1.1.1 Πληροφοριακό Σύστημα, Δεδομένα, Πληροφορία και Γνώση

Εδώ και πολλά χρόνια παράγονται μεγάλες ποσότητες δεδομένων, ενώ στο παρόν και πιθανότατα και στο μέλλον οι επιστημονικές και οι επιχειρηματικές εφαρμογές εστιάζουν όλο και περισσότερο στη διαχείριση δεδομένων μεγάλης κλίμακας (big data). Παραδείγματα πιθανών πηγών δεδομένων (sources) και πόρων ιδιαίτερης αξίας (valuable resources) συναντώνται σε όλους τους τομείς της ανθρώπινης δραστηριότητας: επιχείρηση, επιστήμη, ιατρική, οικονομία, περιβάλλον, σπορ, τουρισμό, ψυχαγωγία κ.λπ.

Υπάρχουν πολλοί ορισμοί της έννοιας των δεδομένων (στοιχείων, data). Ένας απλός ορισμός των δεδομένων (Hirscheim, Klein, Lattinen, 1995) είναι ότι αποτελούν ‘‘Αναλλοίωτες σταθερές (invariants) με πιθανό νόημα για κάποιον που μπορεί να τις ερμηνεύσει’’.

Η έννοια της πληροφορίας θα μπορούσε να οριστεί ως τα δεδομένα συν το νόημα (την ερμηνεία) που τους αποδίδεται (Κιουντούζης 1997). Η πληροφορία προκύπτει ως αποτέλεσμα λογικής επεξεργασίας των δεδομένων από τον ανθρώπινο παράγοντα. Διάφοροι επιστημονικοί τομείς «φωτίζουν» από ειδικότερη οπτική γωνία τις έννοιες αυτές. Για παράδειγμα, σύμφωνα με την εξόρυξη δεδομένων (data mining) τα ακατέργαστα δεδομένα (Raw data), δηλαδή τα δεδομένα χωρίς επεξεργασία, είναι περίπου άχρηστα. Κατά συνέπεια, απαιτούνται τεχνικές για την αυτόματη εξαγωγή πληροφορίας από αυτά. Πιο συγκεκριμένα, η πληροφορία (Information) ταυτίζεται με πρότυπα (patterns) υποκρυπτόμενα στα δεδομένα (underlying the data).

Σύμφωνα με τον Engles τα Λειτουργικά Στοιχεία (Operational data) ενός οργανισμού είναι τα δεδομένα (στοιχεία) τα οποία χρειάζεται ο οργανισμός για να διαχειριστεί τις καθημερινές του λειτουργίες (“for handling its day-to-day operations”) (Engles, 1974).

Σε μία περιγραφική προσέγγιση τα δεδομένα (στοιχεία) είναι αποθηκευμένα γεγονότα (recorded facts). Για παράδειγμα, στον πίνακα TEACH της βάσης δεδομένων αποθηκεύεται το γεγονός (εικόνα 1.1):

Ο (διδάσκων) "Ulman" διδάσκει (το μάθημα) "Database" στο (μεταπτυχιακό πρόγραμμα) "Informatics"

Πίνακας TEACH

LECTURER	PROGRAMME	COURSE	κ.λπ.
διδάσκων	μεταπτυχιακό πρόγραμμα σπουδών (ΜΠΣ)	Μάθημα	κ.λπ.
ULLMAN	INFORMATICS	DATABASE	κ.λπ.
DATE	INFORMATION SYSTEM	DATABASE DESIGN	κ.λπ.

Ο Ulman διδάσκει στο ΜΠΣ Informatics το μάθημα Database

Ο Date διδάσκει στο ΜΠΣ Information System το μάθημα Database Design

Εικόνα 1.1 Αναπαράσταση γεγονότων ως γραμμών του πίνακα TEACH της βάσης δεδομένων

Τα προϊόντα (συστήματα) διαχείρισης βάσεων δεδομένων (database management systems) είναι λογισμικά ανάπτυξης και διαχείρισης εφαρμογών βάσεων δεδομένων. Αποτελούν την «καρδιά» των πληροφοριακών συστημάτων και έχουν κατά κύριο λόγο σχεδιαστεί για να διαχειρίζονται δεδομένα (data) και μεγάλο αριθμό συναλλαγών (high transaction throughput). Οι συναλλαγές συχνά κάνουν μικρές αλλαγές στα λειτουργικά στοιχεία του οργανισμού. Για παράδειγμα, οι τράπεζες διεκπεραιώνουν πάρα πολλές συναλλαγές μεταφοράς χρημάτων αλλά μια συγκεκριμένη μεταφορά κάνει μικρές μεταβολές στα δεδομένα.

Τα πληροφοριακά συστήματα αποτελούν συστήματα ανθρώπινης δραστηριότητας σύμφωνα με τον Checkland (1981), διακρίνονται δε, από δύο βασικά χαρακτηριστικά: (α) την αναζήτηση ενός σκοπού και (β) το δυναμικό τους χαρακτήρα. Ακολουθεί ένας κάπως «περιγραφικός» αλλά σχετικά πλήρης ορισμός του πληροφοριακού συστήματος.

Πληροφοριακό σύστημα είναι ένα σύνολο (ή κοινωνικό σύστημα) από αλληλοεπιδρώντα στοιχεία:

- Οργανωτική δομή (οργανόγραμμα κ.λπ.) του οργανισμού ή της επιχείρησης
- Προσωπικό (εργαζόμενοι, ρόλοι κ.λπ.)
- Μέσα (πόροι, resources), στα οποία περιλαμβάνονται: Δεδομένα (data), συστήματα (προϊόντα) διαχείρισης βάσης δεδομένων, εξοπλισμός (hardware), λογισμικό (software) συστήματος και λογισμικό εφαρμογών, υποδομή δικτύωσης (intranet, extranet), υπηρεσίες νέφους κ.λπ.
- Στην καρδιά του πληροφοριακού συστήματος βρίσκεται η βάση δεδομένων. το εργαλείο για την υλοποίηση της βάσης δεδομένων και τη διαχείρισή της είναι το σύστημα διαχείρισης βάσης δεδομένων.
- Στο λογισμικό ανάπτυξης και λειτουργίας εφαρμογών, εκτός από τα σχεσιακά προϊόντα (συστήματα) διαχείρισης βάσεων δεδομένων (relational dbms) και τα προϊόντα NoSQL, συμπεριλαμβάνονται συστήματα ανάκτησης πληροφορίας (information retrieval systems), λογισμικό συστήματος, εγκάρσια συστήματα επιχειρήσεων (Enterprise Resource Planning) για την «οικοδόμηση» ολοκληρωμένων πληροφοριακών συστημάτων (integrated Information Systems), εργαλεία ανάπτυξης εφαρμογών (application generators, report generators, CASE tools), εφαρμογές ειδικού σκοπού και προγράμματα-εφαρμογές σε τομείς όπως επιχειρηματική ευφυία (Business Intelligence), αποθήκες δεδομένων (Data Warehouse), θεματικές αποθήκες (Datamart), εξόρυξη δεδομένων (Data Mining), διαχείριση γνώσης (Knowledge Management).
- Λειτουργίες, διαδικασίες (operations, functions, procedures, methods).

1.1.2 Γνώση, διαχείριση γνώσης και οργανωσιακή μνήμη

Ο όρος γνώση είναι ένας όρος με μεγάλη ιστορία και ερμηνευτικούς κινδύνους. Οι Nonaka και Takeuchi (1995) θεωρώντας ως πηγή τους τον Polanyi (1966) κάνουν διάκριση μεταξύ σαφούς, ρητής (explicit) και μη προφανούς, σιωπηρής, άρρητης (tacit/implicit) γνώσης. Με τον όρο ‘‘γνώση’’ πολλές εταιρίες/οργανισμοί γενικά εννοούν κωδικοποιημένη πληροφορία με προστιθέμενη - χάρη στον ανθρώπινο παράγοντα - αξία, μέσω ερμηνείας, πείρας, σοφίας κοκ. (Davenport, Volpel, 2001). Σύμφωνα με το Αμερικανικό κέντρο παραγωγικότητας και ποιότητας (APQC), η διαχείριση γνώσης ορίζεται ως ‘‘οι στρατηγικές και οι διαδικασίες προσδιορισμού, σύλληψης και αναμόχλευσης γνώσης, με στόχο την ενδυνάμωση της ανταγωνιστικότητας’’ (Mc Campbell, Mordhead, Clare, Gitters, 1999). Στόχο, επομένως, της διαχείρισης αποτελεί η διευκόλυνση της αλληλεπίδρασης μεταξύ των εργαζομένων, ώστε να τους ευαισθητοποιήσει απέναντι στα ερεθίσματα του περιβάλλοντος και η εξατομικευμένη γνώση να ισχυροποιηθεί και να διανεμηθεί στο εσωτερικό του οργανισμού συνεισφέροντας στη βάση γνώσης που διαθέτει ο οργανισμός. (Nonaka, 1994). Για τη συνέχεια, η γνώση είναι η ‘‘πληροφορία που μετασχηματίζεται σε δυνατότητα για αποτελεσματική δράση’’ (Nonaka and Taceuchi, 1995).

1.1.3 Επιχείρηση/Οργανισμός ως Σύστημα

Με τον όρο επιχείρηση ή οργανισμό εννοούμε ένα σύνολο από ανθρώπους οι οποίοι, με ορθολογιστικά συντονισμένες ενέργειες, επιδιώκουν την επίτευξη κάποιου στόχου. Ένας οργανισμός, μπορεί να θεωρηθεί ότι αποτελείται από τρία υποσυστήματα, καθένα από τα οποία επεξεργάζεται, παράγει ή ελέγχει διαφορετικά πράγματα. Συγκεκριμένα:

1. Το φυσικό σύστημα παραγωγής, που μετασχηματίζει την πρώτη ύλη σε προϊόν σύμφωνα με τις εντολές που παίρνει από το (υπό) σύστημα διοίκησης.
2. Το σύστημα διοίκησης / λήψης αποφάσεων, που επεξεργάζεται τις πληροφορίες και τα δεδομένα που παίρνει από το σύστημα παραγωγής πληροφοριών και δίνει εντολές στο φυσικό σύστημα παραγωγής και
3. Το πληροφοριακό σύστημα, που αποτελεί το συνδετικό κρίκο των δύο άλλων υποσυστημάτων, φροντίζοντας για τη ροή των πληροφοριών ανάμεσά τους (Κιουντούζης, 1997).

Βασικός ρόλος της διοίκησης ενός οργανισμού είναι η αξιολόγηση των πληροφοριών που μέσω του πληροφοριακού συστήματος φτάνουν σ’ αυτή, με στόχο τη λήψη αποφάσεων σύμφωνων με τη στρατηγική του οργανισμού. Η στρατηγική αποτελεί αναπόσπαστο κομμάτι ενός οργανισμού, συνδεδεμένου στενά με τη χρήση της πληροφορικής.

Η οργανωσιακή μνήμη (organizational memory), σχετίζεται με τη δυνατότητα να μεταδίδονται και να διατηρούνται πληροφορίες, από παλαιότερα σε νεότερα μέλη του κοινωνικού συστήματος (Stein E.W., 1995). Στο πλαίσιο του οργανισμού, παρατηρούμε ότι η οργανωσιακή μνήμη είναι ‘‘το μέσο με το οποίο η γνώση του παρελθόντος συνδέεται με δραστηριότητες του παρόντος, συμβάλλοντας κατ’ αυτό τον τρόπο σε υψηλότερα επίπεδα αποδοτικότητας’’ (Stein E.W., 1995). Δραστηριότητες που σχετίζονται με τον παραπάνω ορισμό είναι η λήψη αποφάσεων, η σχεδίαση, η καθοδήγηση, η επικοινωνία κοκ.

Η γνώση ενός οργανισμού φυλάσσεται σε μια ειδική βάση, η οποία μπορεί να κατηγοριοποιηθεί ως εξής (O’ Leary, 1998):

- γνώση σχετική με προτάσεις που ο οργανισμός έκανε στο παρελθόν (proposals knowledge repository),
- γνώση που αφορά νέα σχετικά με συγκεκριμένους τομείς του οργανισμού (news knowledge repository), γνώση που αφορά στον τρόπο με τον οποίο μπορούν να γίνουν ορισμένα πράγματα με τον καλύτερο τρόπο (best practices repository) και

- γνώση σχετική με τις δεξιότητες των ατόμων που καθορίζουν και το ποιος είναι ο ειδικός σε κάθε αντικείμενο (experts knowledge repository).

Βασική μέριμνα παραμένει η ελαχιστοποίηση των αναγκών για δημιουργία λογισμικού (Abecker, Bernardi, Hinkelmann, Kuhn and M. Sintek, 1998). Οι οργανισμοί δεν είναι διατεθειμένοι να επενδύσουν μεγάλα ποσά σε καινοτομικές τεχνολογίες που δεν παρέχουν σαφή οφέλη σε σύντομο χρονικό διάστημα. Αυτό οδήγησε πολλούς ερευνητές να προτείνουν ότι οι οργανωσιακές μνήμες θα πρέπει να εξερευνούν την ήδη διαθέσιμη πληροφορία (κυρίως από βάσεις δεδομένων και ηλεκτρονικά κείμενα ή έγγραφα), παρέχοντας οφέλη με ελάχιστο κόστος.

1.1.3.1 Βιβλιογραφία και αναφορές

Abecker, A., Bernardi, A., Hinkelmann, K., Kühn, O., & Sintek, M. (1998). Techniques for organizational memory information systems. DFKI Document D-98-02, 2.

Avizon D., Fitzgerald G. (2006) Information systems development: Methodologies, Techniques and Tools, McGraw-Hill

Checkland P. (1981). Systems thinking, systems practice. John Wiley and Sons, Chichester, UK.

Davenport, T. H., & Völpel, S. C. (2001). The rise of knowledge towards attention management. Journal of knowledge management, vol. 5, No 3, pp 212-221.

Engles R.W. (1974) A Tutorial on Data-Base Organization, Computer Science and Technology

and their applications.

Hirscheim R., Klein H., and Lattinen K. (1995). Information systems development and data modeling: Conceptual and philosophical foundations. Cambridge University Press, UK.

McCampbell A., Mordhead Clare L., Howard Gittters S. (1999). Knowledge management: the new challenge for the 21st century, Journal of Knowledge Management, vol. 3, No 3, 1999, pp 172-179.

Nonaka I. (1994). A Dynamic Theory of Organizational Knowledge Creation, Organization Science, vol. 5, No. 1, pp. 14-37.

Nonaka I., Takeuchi H. (1995). The knowledge Creating Company, Oxford University Press, Oxford.

O' Leary D. (1998). Using AI in knowledge management: Knowledge Bases and Ontologies. IEEE Intelligent Systems, May/June 1998, pp.34-39.

Polanyi, M. (1966). The Tacit Dimension, Routledge & Kegan Paul, London.

Shalton G., McGill M. (1983). Introduction to Modern Information Retrieval, McGraw-Hill

Stein E. W. (1995). Organizational Memory: Review of concepts and Recommendations for Management, International Journal of Information Management, Vol. 15, No 2 pp 17-32.

Αλεβίζος Θ., Γαλιώτου Ε., Δουλκερίδης Χ., Μαρινάγη Α., Μπέλσης Π., Παλιούρας Γ., Σκουρλάς Χ. (2005) Διαγλωσσική Ανάκτηση Πληροφοριών, Εξατομίκευση και Μοντέλο Χρήστη, 1ο συνέδριο «Αρχιμήδης, Καινοτόμος ανάπτυξη και Τεχνολογία: Ποσοτική και Ποιοτική Αντιμετώπιση».

Κιουντούζης Ε. (1997). Μεθοδολογίες Ανάλυσης και σχεδιασμού Πληροφοριακών Συστημάτων, εκδόσεις Ε. Μπένου, Αθήνα.

Μαρινάγη Α., Σκουρλάς Χ. (2022) Διαχείριση Γνώσης, Κάλλιπος

1.2 Βάσεις δεδομένων και προϊόντα διαχείρισης

Παλαιότερα η αποθήκευση των δεδομένων ενός πληροφοριακού συστήματος (ΠΣ) γινόταν σε παραδοσιακά αρχεία, σειριακά ή τυχαίας προσπέλασης, και οι διαχειριστικές εφαρμογές (business application software) γραφόταν σε γλώσσες προγραμματισμού υψηλού επιπέδου, π.χ., COBOL, BASIC.

Από τη δεκαετία του 80, σχεδόν αποκλειστικά όλα τα δεδομένα των εφαρμογών ΠΣ αποθηκεύονται σε βάσεις δεδομένων. Υπάρχουν πολλά προϊόντα διαχείρισης βάσεων δεδομένων όπως τα προϊόντα Microsoft SQL Server, ORACLE, IBM DB2, INFORMIX, MySQL, κ.λπ.

Η γλώσσα SQL-Structured Query Language χρησιμοποιείται ευρύτατα και αποτελεί de facto πρότυπο, έναν κοινό παρονομαστή, για τα προϊόντα αλλά και για τις εφαρμογές διαχείρισης βάσεων δεδομένων. Σήμερα σχεδόν όλα τα προϊόντα διαχείρισης βάσης υποστηρίζουν τη χρήση της γλώσσας αυτής για τη δημιουργία των εφαρμογών. Επίσης, είναι συνηθισμένη πρακτική να χρησιμοποιούνται για την ανάπτυξη εφαρμογών διαχείρισης συστημάτων βάσης δεδομένων γλώσσες προγραμματισμού μαζί με δηλώσεις της γλώσσας SQL (SQL statements), π.χ., η γλώσσα C# χρησιμοποιείται με τα προϊόντα της Microsoft, η γλώσσα PL/SQL με το προϊόν της Oracle. Οι γλώσσες προγραμματισμού για τη σύνδεση με βάσεις δεδομένων και την εκτέλεση δηλώσεων της γλώσσας SQL χρησιμοποιούν συναρτήσεις/μεθόδους που παρέχονται από διεπαφή API (Application Programming Interface). Δημοφιλής διεπαφή API για τη γλώσσα JAVA είναι η διεπαφή JDBC API (Java Database Connectivity Application Programming Interface).

Έτσι, με τη γλώσσα προγραμματισμού και τη γλώσσα SQL, τη χρήση διεπαφής API, με καλό σχεδιασμό της βάσης δεδομένων μπορούμε να δημιουργήσουμε χρήσιμα προγράμματα-εφαρμογές διαχείρισης συστημάτων βάσης δεδομένων.

Το πιο σημαντικό μέρος του σχεδιασμού μας αναφέρεται στον τρόπο που τα δεδομένα μας θα δομηθούν. Μια πρόχειρα σχεδιασμένη Βάση Δεδομένων μπορεί να δημιουργήσει πολλά προβλήματα στην κατασκευή και τη συντήρηση της εφαρμογής ενώ αντίθετα η καλή σχεδίαση διευκολύνει τη ζωή του προγραμματιστή και του διαχειριστή της βάσης.

Ακολουθεί συνοπτική ιστορία της τεχνολογίας βάσεων δεδομένων.

1.2.1 Συνοπτική ιστορία των βάσεων δεδομένων

Ακολουθεί σύντομη ιστορική επισκόπηση των βάσεων δεδομένων.

1) **απαρχές:** Οι ρίζες πάνε πίσω σε βιβλιοθήκες, κυβερνητικά, επιχειρηματικά και ιατρικά αρχεία. Υπάρχει μια πολύ μακρά ιστορία αποθήκευσης πληροφοριών, ευρετηρίασης και ανάκτησης.

2) **δεκαετία 1960**: Οι υπολογιστές γίνονται οικονομικά αποδοτικοί. Δύο βασικά μοντέλα δεδομένων αναπτύχθηκαν: το μοντέλο του δικτύου (network model) CODASYL και το ιεραρχικό (hierarchical) IMS. Η πρόσβαση στη βάση δεδομένων είναι μέσω λειτουργιών χαμηλού επιπέδου δεικτών που συνδέουν τις εγγραφές ("Access to database is through low-level pointer operations linking records").

3) **δεκαετία 1970**: Ο E.F. Codd προτείνει το Σχεσιακό Μοντέλο - Relational Model στο άρθρο:

   Codd, E.F., (1970) A Relational Model of Data for Large Shared Data Banks, Comm. ACM, 13:6, pp. 377-387

   "It provides a means of describing data with its natural structure only--that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation on the other."

4) **δεκαετία 1970:** Πώς η θεωρία οδηγεί σε βέλτιστες πρακτικές.

   Ingres: Αναπτύχθηκε στην UCB. Το σύστημα χρησιμοποιεί QUEL ως γλώσσα ερωτημάτων.

   Σύστημα R: Αναπτύχθηκε στην IBM. Το σύστημα χρησιμοποιεί SEQUEL ως γλώσσα ερωτημάτων.

5) **1976:** Ο P. Chen προτείνει το μοντέλο οντοτήτων συσχετίσεων, Entity-Relationship (ER) model, για σχεδίαση βάσεων (database design) στο άρθρο: Peter Chen (March 1976), The Entity-Relationship Model – Toward a Unified View of Data, ACM Transactions on Database Systems, 1:1, pp. 9–36.

6) **αρχές δεκαετίας 1980**: Σχεσιακά προϊόντα διαχείρισης βάσεων

7) **μέσα δεκαετίας 1980**: SQL (Structured Query Language). Το πρότυπο ("standard").

8) **αρχές δεκαετίας 1990**: PowerBuilder (Sybase), Oracle Developer, VB (Microsoft), …

   Client-server model. Excel/Access (MS). ODBC

   Object Database Management Systems (ODBMS) prototypes.

9) **μέσα δεκαετίας 1990**: Internet/WWW. «Εκτίναξη» του Client-server. Web/DB εκθετική ανάπτυξη.

10) **τέλη δεκαετίας 1990**: Active Server Pages, Front Page, Java Servlets, JDBC, Enterprise Java Beans, ColdFusion, Dream Weaver, Oracle Developer 2000, ….

    Open source: Apache, MySQL, …

    Online Transaction processing (OLTP), online analytic processing (OLAP).

11) **αρχές 21ου αιώνα**: Συνεχής αύξηση εφαρμογών των βάσεων. Χρήση PDAs, POS transactions, …

    Μεγάλοι παίκτες στην αγορά βάσεων (DB market): IBM, Microsoft, Oracle.

12) **Μελλοντικές τάσεις:** Βλέπε 1.2.2

1.2.1.1 Βιβλιογραφία και αναφορές

Codd, E. F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13(6), 377-387.

Chen, P. P. S. (1976). The entity-relationship model—toward a unified view of data. ACM transactions on database systems (TODS), 1(1), 9-36.

Berg, K. L., Seymour, T., & Goel, R. (2013). History of databases. International Journal of Management & Information Systems (IJMIS), 17(1), 29-36.

Mintz. J. (2019) A Brief Overview of the Database Landscape, 2019

https://www.ibm.com/cloud/blog/brief-overview-database-landscape

1.2.2 Το παρόν και το μέλλον των βάσεων δεδομένων

Οι ενδιαφερόμενοι για το παρόν και το μέλλον των βάσεων δεδομένων, είναι απαραίτητο να ανατρέξουν στις επισκοπήσεις και τις προβλέψεις που συντάσσονται περιοδικά από κορυφαίους ειδικούς. Ενδεικτικά παραθέτουμε κάποιες γνωστές αναφορές.

Bernstein, P. A., Dayal, U., DeWitt, D. J., Gawlick, D., Gray, J., Jarke, M., Lindsay, B.G., Lockemann, P.C., Maier, D., Neuhold, E.J., Reuter, A., Rowe, L.A., Schek, H.J., Schmidt, J.W., Schrefl, M. & Stonebraker, M. (1989). Future directions in DBMS research-the Laguna beach participants. ACM SIGMOD Record, 18(1), 17-26.

Bernstein, P., Brodie, M., Ceri, S., DeWitt, D., Franklin, M., Garcia-Molina, H., Gray, J., Held, G., Hellerstein, J.M., Jagadish, H.V., Lesk, M., Maier, J., Naughton, J.F., Pirahesh, H., Stonebraker M. & Ullman, J. (1998). The Asilomar Report on Database Research. ACM SIGMOD Record, 27(4), 74-80.

Abiteboul, S., Agrawal, R., Bernstein, P., Carey, M., Ceri, S., Croft, B., DeWitt, D.J., Franklin, M.J., Garcia-Molina, H., Gawlick, D., Gray, J., Haas, L.M., Halevy, A.Y., Hellerstein, J.M., Ioannidis, Y.E., Kersten, M.L., Pazzani, M.J., Lesk, M., Maier, D., Naughton, J.F., Schek, H.-J., Sellis, T.K., Silberschatz, A., Stonebraker, M., Snodgrass, R.T., Ullman, J.D., Weikum, G., Widom, J. & Zdonik, S. (2005). The Lowell database research self-assessment. Communications of the ACM, 48(5), 111-118.

R. Agrawal, A. Ailamaki, P. A. Bernstein, E. A. Brewer, M. J. Carey, S. Chaudhuri, A. Doan, D. Florescu, M. J. Franklin, H. Garcia-Molina, J. Gehrke, L. Gruenwald, L. M. Haas, A. Y. Halevy, J. M. Hellerstein, Y. E. Ioannidis, H. F. Korth, D. Kossmann, S. Madden, R. Magoulas, B. C. Ooi, T. O'Reilly, R. Ramakrishnan, S. Sarawagi, M. Stonebraker, A. S. Szalay, and G. Weikum. "The claremont report on database research". Communications of the ACM, 52(6):56–65, 2009.

Abadi, D., Agrawal, R., Ailamaki, A., Balazinska, M., Bernstein, P. A., Carey, M. J., Chaudhuri, S., Dean, J., Doan, A., Franklin, M.J., Gehrke, J., Haas, L.M. Halevy, A.Y., Hellerstein, J.M., Ioannidis, Y.E., Jagadish, H.V., Kossmann, D., Madden, S., Mehrotra, S., Milo, T., Naughton, J. F., Ramakrishnan, R., Markl, V., Olston, C., Ooi, B.C., Ré,, C., Suciu, D., Stonebraker, M., Walter, T. & Widom, J. (2016). The Beckman report on database research. Communications of the ACM, 59(2), 92-99.

Abadi, D., Ailamaki, A., Andersen, D., Bailis, P., Balazinska, M., Bernstein, P., Boncz, P., Chaudhuri, S., Cheung, A., Doan, A., Dong, L., Franklin, M.J., Freire, J., Halevy, A., Hellerstein, J.M., Idreos, S., Kossmann, S., Kraska, T., Krishnamurthy, S., Markl, V., Melnik, S., Milo, T., Mohan, C., Neumann, T., Ooi, B.C., Ozcan, F., Patel, J., Pavlo, A., Popa, R., Ramakrishnan, R., Ré, C., Stonebraker, M. & Suciu, D. (2020). The Seattle report on database research. ACM SIGMOD Record, 48(4), 44-53.

1.3 Τι είναι βάση δεδομένων (Data Base) και σύστημα βάσης δεδομένων (Data Base System) σύμφωνα με τη βιβλιογραφία

1.3.1 Πρώτη προσέγγιση: Ορισμός της βάσης σύμφωνα με τα κοινώς αποδεκτά ή τι είναι βάση δεδομένων σύμφωνα με την οπτική γωνία που υιοθετείται σε γνωστά προϊόντα διαχείρισης βάσεως δεδομένων

Θα εξετάσουμε αρχικά την έννοια της βάσης δεδομένων, κάπως περιγραφικά, σύμφωνα με την οπτική γωνία των προϊόντων που κυκλοφορούν στο εμπόριο, π.χ., Oracle, MySQL.

- Μία βάση δεδομένων μπορούμε να την περιγράψουμε πολύ απλά σαν μια οργάνωση των δεδομένων που ενδιαφέρουν μια επιχείρηση, ένα οργανισμό ή και ένα φυσικό πρόσωπο.
- Η αποθήκευση και η ανάκτηση των δεδομένων γίνεται με ενιαίο τρόπο με τη βοήθεια ενός προγράμματος διαχείρισης βάσης δεδομένων, όπως το προϊόν Microsoft SQL Server, το προϊόν της ORACLE κ.λπ.
- Κύριο χαρακτηριστικό της Σχεσιακής βάσης δεδομένων, που αποτελεί de facto πρότυπο για τους μεγάλους κατασκευαστές Συστημάτων Διαχείρισης Βάσης Δεδομένων (ΣΔΒΔ), είναι ότι τα δεδομένα της βάσης δεδομένων είναι οργανωμένα/αποθηκευμένα σε πίνακες χωρίς περιττές επαναλήψεις (redundancy), έχουν οριστεί ευρετήρια και κατασκευάστηκαν ερωτήσεις (queries) που διευκολύνουν τη γρήγορη αναζήτηση των δεδομένων.

Σε «επιτραπέζια» ΣΔΒΔ η βάση δεδομένων είναι ένα αρχείο που περιέχει πίνακες (tables), ευρετήρια (index) και ερωτήσεις (queries) αλλά και φόρμες (e-forms), αναφορές-εκτυπώσεις (reports) κ.λπ. Για παράδειγμα, στο γνωστό προϊόν της Microsoft Access η βάση δεδομένων είναι ένα αρχείο (.mdb) με όλα όσα προαναφέραμε.

1.3.2 Δεύτερη προσέγγιση: Ορισμοί Engles, Martin, Date, Elmasri-Navathe, Ullman-Widow

Για τις ανάγκες των παρακάτω ορισμών θεωρούμε έναν οποιοδήποτε οργανισμό ή επιχείρηση.

Ο Οργανισμός θα πρέπει να αποθηκεύει στοιχεία (δεδομένα, data) για τις λειτουργίες του. Τα στοιχεία αυτά είναι τα λειτουργικά στοιχεία (operational data) του Οργανισμού. Είναι προφανές ότι τα στοιχεία είναι διαφορετικά για κάθε έναν από τους οργανισμούς ή/και για τα τμήματα των οργανισμών ή ακόμα και για διάφορες κατηγορίες υπαλλήλων. Χωρίς να υπεισέλθουμε στις διαφορές των στοιχείων-δεδομένων θα προσπαθήσουμε να εστιάσουμε σε γνωστούς ορισμούς που "ομαδοποιούν" τις επιμέρους θεωρήσεις των στοιχείων του Οργανισμού χρησιμοποιώντας τον όρο Βάση Δεδομένων (data base). Παραθέτουμε στη συνέχεια τον ορισμό της βάσης δεδομένων κατά Engles:

Ορισμός (Engles, 1974)

«Μια βάση δεδομένων ενός οργανισμού ή μιας επιχείρησης είναι μια συλλογή αποθηκευμένων λειτουργικών στοιχείων που χρησιμοποιούνται από τις μηχανογραφικές εφαρμογές του πληροφοριακού συστήματος. Τα λειτουργικά δεδομένα αντιπροσωπεύουν πληροφορίες σχετικές με οντότητες που ενδιαφέρουν την επιχείρηση».(“A data base is defined as the total collection of stored, operational data used in the application systems of a particular enterprise. Operational data represents certain information about entities of concern to the enterprise”).

Ο Date, ενώ αρχικά χρησιμοποιούσε τον παραπάνω ορισμό, στις νεότερες εκδόσεις του βιβλίου του (βλέπε για παράδειγμα στο βιβλίο Date, An introduction to database systems, vol.1, vol.2 Addison-Wesley) προτείνει τη χρήση του όρου persistent data αντί του όρου operational data στον παραπάνω ορισμό του Engles.

Σημείωση

Θα αποδίδαμε τον όρο persistent data ως στατικά-σταθερά δεδομένα σε αντιδιαστολή με τον όρο δυναμικά δεδομένα, dynamic data-transactional data.

Ο Date, δίνει έμφαση στο γεγονός ότι οι βάσεις χρησιμοποιούνται όλο και περισσότερο σε συστήματα στήριξης αποφάσεων (Decision Support Systems) πέρα από τις παραδοσιακές εφαρμογές που κάλυπτε ο ορισμός του Engles. Ο όρος persistent data, επίσης, αντιπαραβάλλεται/ αντιδιαστέλεται, στο βιβλίο του, με τον όρο transient data (θα το αποδίδαμε ως παροδικά ή μη μόνιμα δεδομένα) που καλύπτει δεδομένα εισόδου/εξόδου, εντολές ελέγχου κ.λπ..

Στον περιγραφικό ορισμό του Martin που ακολουθεί γίνεται μια προσπάθεια να ορισθεί η Βάση Δεδομένων σαν αντίποδας της συνηθισμένης οργάνωσης όπου μια συλλογή εγγραφών (records) αρχείων (files) σχεδιάζεται για μία μόνο εφαρμογή.

Ορισμός (Martin)

Μια βάση δεδομένων είναι μια συλλογή αλληλοσχετιζόμενων δεδομένων (data) που αποθηκεύονται μαζί χωρίς άχρηστους πλεονασμούς (redundancies) για την εξυπηρέτηση πολλών εφαρμογών. Η αποθήκευση των δεδομένων είναι τέτοια ώστε τα δεδομένα είναι ανεξάρτητα των προγραμμάτων που τα διαχειρίζονται. Η εισαγωγή νέων δεδομένων, η τροποποίηση και η ανάκτηση δεδομένων από τη βάση δεδομένων ακολουθεί

κοινή και ελεγχόμενη προσέγγιση για όλες τις εφαρμογές. Η δομή των δεδομένων παρέχει το θεμέλιο για ανάπτυξη μελλοντικών εφαρμογών. Δηλαδή, η οργάνωση των δεδομένων είναι τέτοια ώστε μπορούμε να υλοποιήσουμε νέες εφαρμογές χωρίς ιδιαίτερο κόπο και χωρίς να αλλάξει οτιδήποτε στις παλιές. Τέλος, λέμε ότι ένα σύστημα περιέχει μια συλλογή από Βάσεις Δεδομένων αν οι Βάσεις αυτές είναι τελείως ξεχωριστές σε δομή.

Ακολουθούν δύο ορισμοί από γνωστά και δημοφιλή συγγράμματα.

Ορισμός Elmasri-Navathe

Βάση δεδομένων είναι μία συλλογή από σχετιζόμενα δεδομένα, όπου δεδομένα σημαίνει καταγεγραμμένα γεγονότα (η λέξη γεγονότα αποδίδει τους όρους events, facts). Μία τυπική βάση δεδομένων παριστάνει κάποια άποψη του πραγματικού κόσμου (Miniworld, Universe of Discource) και χρησιμοποιείται για συγκεκριμένους λόγους (δηλαδή, για να καλύψει συγκεκριμένες ανάγκες) από μία ή περισσότερες ομάδες χρηστών (σελ. 20, 39-40)

Ορισμός Ullman-Widow

Βάση δεδομένων είναι μία «συλλογή πληροφοριών (information) που υπάρχει για ένα μεγάλο χρονικό διάστημα, συχνά για πολλά χρόνια στην καθομιλουμένη ο όρος βάση δεδομένων αναφέρεται σε μια συλλογή δεδομένων που τη διαχειρίζεται ένα Σύστημα Διαχείρισης Βάσης Δεδομένων (DBMS - ΣΔΒΔ)».

Θα πρέπει να υπογραμμίσουμε ότι στην πράξη δεν αρκεί να αγοράσουμε και να χρησιμοποιήσουμε ένα προϊόν (ΣΔΒΔ) για να έχουμε μία βάση δεδομένων. Συχνά και ως αποτέλεσμα κακής σχεδίασης έχουμε πλεονασμούς στη βάση δεδομένων και επιπλέον πολλές βάσεις δεδομένων στον ίδιο οργανισμό με περιορισμένη ολοκλήρωση-ενοποίηση δεδομένων (data integration), ασυνεπή δεδομένα (inconsistent data) κ.λπ.

Οι Ullman-Widow συμπληρώνουν ότι ένα ΣΔΒΔ πρέπει να:

- επιτρέπει στους χρήστες να δημιουργήσουν νέες βάσεις δεδομένων και να καθορίσουν τη λογική δομή των δεδομένων τους (logical structure of data) χρησιμοποιώντας μία Γλώσσα Ορισμού Δεδομένων (DDL)
- επιτρέπει στους χρήστες να αναζητούν δεδομένα (to query the data) και να τα τροποποιούν χρησιμοποιώντας μία γλώσσα ερωτήσεων ή Γλώσσα Διαχείρισης Δεδομένων (a query language or DML)
- υποστηρίζουν την αποθήκευση μεγάλων ποσοτήτων δεδομένων και για μεγάλα χρονικά διαστήματα κατά τρόπο που να εξασφαλίζει την ασφάλεια των δεδομένων αυτών
- ελέγχει την πρόσβαση στα δεδομένα αυτά για διάφορες κατηγορίες χρηστών

1.3.3 Τρίτη προσέγγιση: Εξειδίκευση ορισμού για τα μοντέλα δεδομένων σχεσιακό, ιεραρχικό και δικτυωτό

Σε κάθε μοντέλο δεδομένων (σχεσιακό, ιεραρχικό και δικτυωτό (γνωστό και ως CODASYL) μπορεί να εξειδικευθεί ο ορισμός της βάσης δεδομένων.

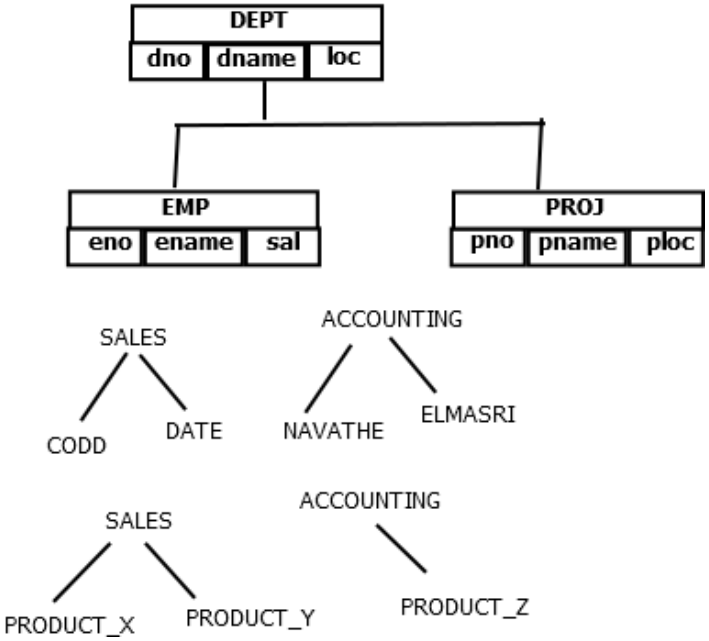
Θα ξεκινήσουμε με μία σύντομη αναφορά στα ιστορικά μοντέλα δεδομένων, το ιεραρχικό και το δικτυωτό, βασισμένη κυρίως στο συνοπτικό και περιεκτικό βιβλίο F.D. Rolland, The essence of databases, Prentice-Hall, 1998.

Μία ιεραρχική βάση δεδομένων είναι μία βάση που έχει δομηθεί σαν ένα δέντρο συσχετισμένων τύπων εγγραφών (record types). Όλες οι ιεραρχικές βάσεις δεδομένων έχουν μία και μόνο μία "ρίζα" (root). Αυτή η

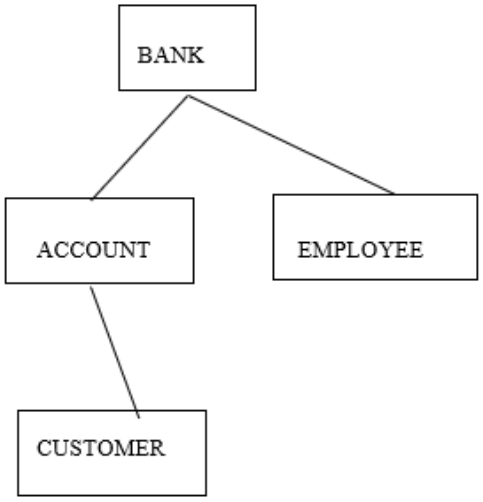
"ρίζα"-τύπος δεδομένων μπορεί να συμμετέχει σε 1:M συσχετίσεις με οποιοδήποτε αριθμό τύπων δεδομένων - "παιδιών". Κάθε "παιδί" μπορεί επίσης να έχει "παιδιά" - τύπους δεδομένων. Στην Εικόνα 1.2 δίδεται ιεραρχικό σχήμα δύο επιπέδων της βάσης δεδομένων εταιρίας που περιλαμβάνει τους τύπους εγγραφών (record types) DEPT, EMP, PROJ. Αυτοί οι τύποι εγγραφών αντιστοιχούν στα τμήματα, τους υπαλλήλους και τα έργα μιας εταιρίας. Στην Εικόνα 1.2 βλέπουμε γεγονότα, όπως τα παρακάτω:

- Στο τμήμα πωλήσεων (SALES) εργάζονται οι CODD και DATE
- Το τμήμα πωλήσεων (SALES) προωθεί τα προϊόντα PRODUCT_X και PRODUCT_Y

Στην Εικόνα 1.3 δίδεται παράδειγμα Ιεραρχικής βάσης δεδομένων τριών επιπέδων που περιλαμβάνει τους τύπους εγγραφών (record types) BANK, ACCOUNT, EMPLOYEE, CUSTOMER.



Εικόνα 1.2 Ιεραρχικό σχήμα βάσης δεδομένων εταιρίας



Εικόνα 1.3 Παράδειγμα Ιεραρχικής βάσης δεδομένων με τύπους εγγραφών (record types) BANK, ACCOUNT, EMPLOYEE, CUSTOMER

Μια δικτυωτή βάση δεδομένων (network database) αποτελείται κατά κύριο λόγο από μία συλλογή τύπων εγγραφών. Οι τύποι εγγραφών συνδέονται μεταξύ τους με ειδικές συνδέσεις (links). Μία σύνδεση

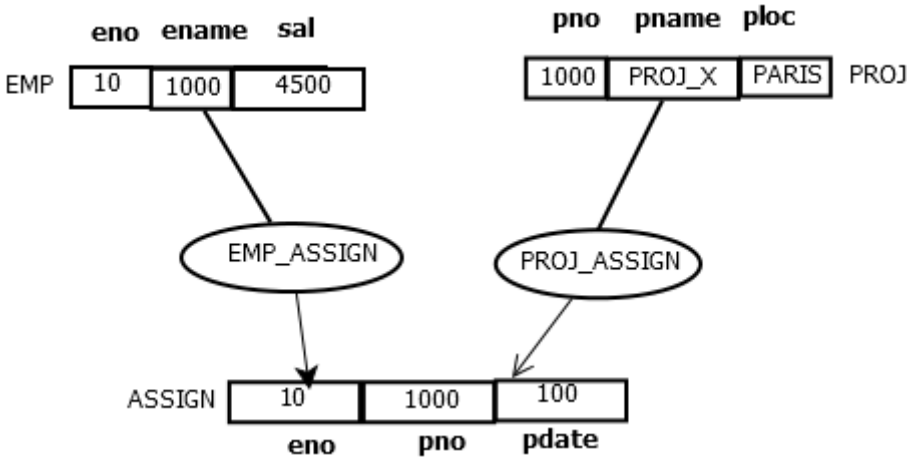
αναπαριστά μία συσχέτιση 1:M ανάμεσα σε μία εγγραφή ενός τύπου (που ονομάζεται ιδιοκτήτης, owner) και σε ένα σύνολο εγγραφών (set of records) ενός άλλου τύπου (που ονομάζονται μέλη, members).

Η κύρια διαφορά της δικτυωτής βάσης δεδομένων από την ιεραρχική βάση συνίσταται στο γεγονός ότι δεν υπάρχει περιορισμός στον αριθμό και την κατεύθυνση των συνδέσεων που μπορούμε να ορίσουμε ανάμεσα σε τύπους εγγραφών. Επίσης, δεν υπάρχει ανάγκη για τύπο εγγραφής "ρίζα" .

Το Data Base Task Group - DBTG της CODASYL στα 1971, διατύπωσε τον παρακάτω ορισμό για το δικτυωτό μοντέλο:

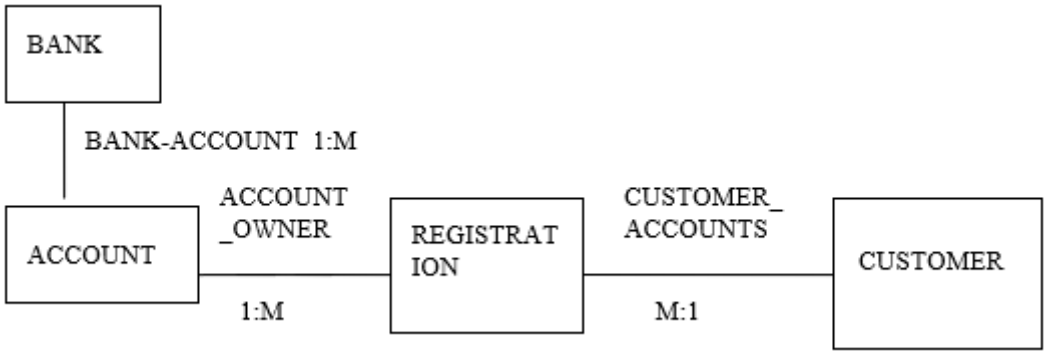
Ορισμός (CODASYL network model, 1971)

Μια βάση δεδομένων αποτελείται από όλα τα στιγμιότυπα εγγραφής, τις συλλογές (τα σύνολα, set) στιγμιοτύπων και τις περιοχές που ελέγχονται από ένα συγκεκριμένο σχήμα. Εάν μια εγκατάσταση έχει πολλές βάσεις δεδομένων, πρέπει να υπάρχει ξεχωριστό σχήμα για κάθε βάση. Επιπλέον, τα περιεχόμενο των διαφορετικών βάσεων δεδομένων θεωρούνται μη συνδεδεμένα. ("A data base consists of all the record occurrences, set occurrences and areas which are controlled by a specific schema. If an installation has multiple data bases, there must be a separate schema for each data base. Furthermore, the content of different data bases is assumed to be disjoint."). Δηλαδή, στο δικτυωτό μοντέλο (network data model) η βάση δεδομένων αποτελείται από μία συλλογή στιγμιότυπων (set-type occurrences). Κάθε συλλογή στιγμιοτύπων (set-type occurrences) έχει ένα στιγμιότυπο (occurrence) της εγγραφής ιδιοκτήτη (owner record) και 0 ή πολλά στιγμιότυπα (zero or more occurrences) των εγγραφών μελών (member records). Οι εγγραφές μέλη (member records) που ανήκουν σε διαφορετική εγγραφή ιδιοκτήτη (owner record) είναι ξένες μεταξύ τους. Για τον ορισμό μιας δικτυωτής βάσης δεδομένων (network database) απαιτείται ο ορισμός των τύπων των εγγραφών (database record types) που αποτελούνται από data items και ο ορισμός των set-types. Δείτε και το παράδειγμα της εικόνας 1.4.



Εικόνα 1.4 Δικτυωτό σχήμα βάσης δεδομένων εταιρίας. Περιλαμβάνει record types (EMP, PROJ, ASSIGN), data *items ανά record type (π.χ. eno, ename, sal για record type EMP), set-types (EMP_ASSIGN, PROJ_ASSIGN).*

Στην Εικόνα 1.5 δίδεται παράδειγμα CODASYL τραπεζικής βάσης δεδομένων που περιλαμβάνει τους τύπους εγγραφών (record types) BANK, ACCOUNT, REGISTRATION, CUSTOMER.



Εικόνα 1.5 Παράδειγμα CODASYL τραπεζικής βάσης δεδομένων

Η εξειδίκευση του ορισμού της βάσης δεδομένων για το σχεσιακό μοντέλο θα γίνει στην ενότητα 1.4.

1.4 Σχεσιακό μοντέλο και Σχεσιακές βάσεις δεδομένων. Σύνδεση της έννοιας της Μοντελοποίησης και της έννοιας του Συστήματος Βάσης Δεδομένων

Παραθέτουμε απόσπασμα από το σύγγραμμα των Ullman-Widow (p. 4): "Following a famous paper (πρόκειται για το πασίγνωστο άρθρο Codd, E. F., A relational model for large shared data banks, Comm. ACM, 13:6, pp. 377-387) written by Ted Codd in 1970 database system changed significantly. Codd proposed the Database systems should present the user with a view of data organised as tables called relations".

1.4.1 Τι είναι σχεσιακή βάση δεδομένων;

Αρχικά θα παρατεθεί ένα πρώτο παράδειγμα σχεσιακής βάσης δεδομένων (Εικόνα 1.6). Στη σχεσιακή βάση δεδομένων (relational data base) όλα τα δεδομένα-στοιχεία (data) είναι οργανωμένα σε πίνακες (tables).

Για τα δεδομένα ισχύουν κάποιοι περιορισμοί (constraint, business rules):

1. Κάθε υπάλληλος ανήκει σε ένα τμήμα,
2. Σε κάθε τμήμα ανήκουν πολλοί υπάλληλοι,
3. Ο μισθός του υπαλλήλου εξαρτάται από την προσωπική συμφωνία.

Πίνακες σχεσιακής βάσης δεδομένων

Dept (πίνακας με στοιχεία τμημάτων)

DeptNo	Dname
Κωδικός	Όνομα
10	ΠΩΛΗΣΕΙΣ
20	ΛΟΓΙΣΤΗΡΙΟ
30	ΜΙΣΘΟΔΟΣΙΑ

Κύριο κλειδί (Primary Key)=DeptNo (το κύριο κλειδί έχει μοναδική τιμή για κάθε τμήμα)

Emp (πίνακας με στοιχεία υπαλλήλων)

Empno	Surname	Name	Job	DeptNo	Sal
Κωδικός	Επώνυμο	Όνομα	Θέση	Κωδικός τμήματος	Μισθός

10	ΜΑΡΚΟΥ	ΜΑΡΚΟΣ	ΠΩΛΗΤΗΣ	10	4200
20	ΣΠΥΡΟΥ	ΣΠΥΡΟΣ	ΠΩΛΗΤΗΣ	10	4000
30	ΧΡΗΣΤΟΥ	ΧΡΗΣΤΟΣ	ΑΝΑΛΥΤΗΣ	20	3000
40	ΝΙΚΟΥ	ΝΙΚΟΣ	ΧΕΙΡΙΣΤΗΣ	30	1500

Κύριο κλειδί= Empno (έχει μοναδική τιμή για κάθε υπάλληλο),
Ξένο κλειδί (Foreign key)=DeptNo

(το ξένο κλειδί είναι κύριο κλειδί στον άλλο πίνακα)

Εικόνα 1.6 Σχεδίαση βάσης δεδομένων όταν ο μισθός εξαρτάται από προσωπική συμφωνία του υπαλλήλου. Στους πίνακες ορίζονται κύρια (πρωτεύοντα) κλειδιά και ξένα κλειδιά.

Σε περιγραφική προσέγγιση, το κύριο (ή πρωτεύον κλειδί, primary key) έχει μοναδική τιμή για κάθε γραμμή του πίνακα. Στο παράδειγμά μας, κάθε τμήμα έχει μοναδική τιμή για τον κωδικό (DeptNo) του και κάθε υπάλληλος έχει μοναδική τιμή για τον κωδικό (Empno) του.

Η στήλη ενός πίνακα είναι ξένο κλειδί (foreign key) αν είναι κύριο κλειδί σε άλλον πίνακα. Μπορούμε να έχουμε και σύνθετα ξένα κλειδιά. Στην περίπτωση αυτή, οι στήλες που αποτελούν το ξένο κλειδί πρέπει να είναι σύνθετο κύριο κλειδί σε άλλον πίνακα.

Στην Εικόνα 1.7 βλέπουμε σχεσιακή βάση δεδομένων που περιλαμβάνει τους πίνακες Employee, Department, Job και τα κύρια κλειδιά τους. Ο πίνακας Employee έχει δύο ξένα κλειδιά. Η σχεδίαση της βάσης δεδομένων προκύπτει από τους περιορισμούς:

1. Κάθε υπάλληλος ανήκει σε ένα τμήμα και έχει μία θέση.
2. Σε ένα τμήμα ανήκουν πολλοί υπάλληλοι.
3. Πολλοί υπάλληλοι έχουν την ίδια θέση.
4. Ο μισθός εξαρτάται από τη θέση, π.χ., όλοι οι πωλητές έχουν μισθό 4200, όλοι οι αναλυτές δεδομένων μισθό 3000, όλοι οι χειριστές μισθό 1500.

Employee (πίνακας με στοιχεία υπαλλήλων)

Empno	Name	JobNo	DeptNo
PK		FK	FK

Department

DeptNo	Dname
PK	

Job

JobNo	Job	Sal
PK		

Εικόνα 1.7 Σχεδίαση βάσης δεδομένων όταν ο μισθός εξαρτάται από τη θέση του υπαλλήλου. Κύρια και ξένα κλειδιά πινάκων της βάσης.

Αρχικά θα δοθούν κάπως «περιγραφικοί» ορισμοί της σχεσιακής βάσης δεδομένων. Στη συνέχεια, στην εργασία μας θα δοθεί και αυστηρότερος ορισμός της έννοιας της σχεσιακής βάσης.

Η οργάνωση της βάσης με κύριο στοιχείο τους πίνακες είναι γνωστή και ως σχεσιακή (relational) οργάνωση βάσης δεδομένων:

- Οι πίνακες αποτελούνται από γραμμές και στήλες. Εκτός από τους πίνακες η βάση δεδομένων περιλαμβάνει ευρετήρια, όψεις, υλοποίηση περιορισμών και κανόνων ακεραιότητας δεδομένων κ.λπ.
- Ένα ευρετήριο είναι μια λίστα δεικτών που συνδέονται με τις γραμμές κάποιου πίνακα της βάσης. Το ευρετήριο επιταχύνει την ανάκτηση των δεδομένων.
- Ένα Ερώτημα (query) είναι ένας αποθηκευμένος τρόπος αναζήτησης στοιχείων της βάσης.
- Οι φόρμες είναι ηλεκτρονικές οθόνες που υποστηρίζονται από προγράμματα που κατασκεύασαν προγραμματιστές και επιτρέπουν σε έναν τελικό χρήστη να διαχειριστεί τα δεδομένα της βάσης δεδομένων.

Αυτή η "περιγραφική" προσέγγιση στον ορισμό της σχεσιακής βάσης αντανακλά την οπτική γωνία που υιοθετούν γνωστά προϊόντα. Για παράδειγμα, διαβάζουμε στο "MySQL 8.0 Reference Manual" του προϊόντος MySQL:

«Μια σχεσιακή βάση δεδομένων αποθηκεύει δεδομένα σε ξεχωριστούς πίνακες αντί να τοποθετεί όλα τα δεδομένα σε μια μεγάλη περιοχή αποθήκευσης. Οι δομές της βάσης δεδομένων οργανώνονται σε φυσικά αρχεία βελτιστοποιημένα για την ταχύτερη ανάκτηση των δεδομένων. Το λογικό μοντέλο, με αντικείμενα όπως βάσεις δεδομένων, πίνακες, όψεις, γραμμές και στήλες, προσφέρει ένα ευέλικτο περιβάλλον προγραμματισμού. Επιτρέπει τη ρύθμιση των κανόνων που διέπουν τις συσχετίσεις μεταξύ διαφορετικών πεδίων δεδομένων, όπως συσχετίσεις ένα προς ένα, συσχετίσεις ένα προς πολλά, πεδία δεδομένων μοναδικά, υποχρεωτικά ή προαιρετικά και "δείκτες" μεταξύ διαφορετικών πινάκων. Η βάση δεδομένων εγγυάται αυτούς τους κανόνες, έτσι ώστε με μια καλά σχεδιασμένη βάση δεδομένων, τα προγράμματα εφαρμογών δεν βλέπουν ασυνεπή, διπλά, ορφανά, μη ενημερωμένα ή ελλιπή δεδομένα.»

"A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and "pointers" between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data."

Ο ορισμός αυτός θα γίνει πιο κατανοητός όταν μελετήσετε το μοντέλο οντοτήτων συσχετίσεων (βλέπε, π.χ., 1.3.4)

Σύμφωνα με τη Microsoft:

«Μια βάση δεδομένων στο προϊόν διαχείρισης βάσης δεδομένων SQL Server αποτελείται από μια συλλογή πινάκων που αποθηκεύει ένα συγκεκριμένο σύνολο δομημένων δεδομένων. Ένας πίνακας περιέχει μια συλλογή γραμμών, που αναφέρονται επίσης ως εγγραφές ή πλειάδες, και στηλών, που αναφέρονται επίσης ως χαρακτηριστικά. Κάθε στήλη στον πίνακα έχει σχεδιαστεί για να αποθηκεύει έναν συγκεκριμένο τύπο πληροφοριών, για παράδειγμα, ημερομηνίες, ονόματα, ποσά και αριθμούς. Ένας υπολογιστής μπορεί να έχει εγκατεστημένο ένα ή περισσότερα στιγμιότυπα (αντίγραφα) του SQL Server. Κάθε στιγμιότυπο του SQL Server μπορεί να περιέχει μία ή πολλές βάσεις δεδομένων. Μέσα σε μια βάση δεδομένων, υπάρχουν μία ή πολλές ομάδες ιδιοκτησίας αντικειμένων της βάσης που ονομάζονται σχήματα. Μέσα σε κάθε σχήμα υπάρχουν αντικείμενα βάσης δεδομένων όπως πίνακες, όψεις και αποθηκευμένες διαδικασίες."

"A database in SQL Server is made up of a collection of tables that stores a specific set of structured data. A table contains a collection of rows, also referred to as records or tuples, and columns, also referred to as attributes. Each column in the table is designed to store a certain type of information, for example, dates, names, dollar amounts, and numbers. A computer can have one or more than one instance of SQL Server installed. Each instance of SQL Server can contain one or many databases. Within a database, there are one or

many object ownership groups called schemas. Within each schema there are database objects such as tables, views, and stored procedures."

(https://docs.microsoft.com/en-us/sql/relational-databases/databases/databases?view=sql-server-2017)

1.4.2 Σχεσιακές και αντικειμενοστρεφείς βάσεις δεδομένων

Οι αντικειμενοστρεφείς βάσεις ξεκίνησαν σαν μία νέα δυναμική κατεύθυνση που ελέγετο, από πολλούς, ότι θα αντικαταστήσει τις σχεσιακές βάσεις. Ήδη, στις μέρες μας, η ανάλυση της κατάστασης μάλλον οδηγεί στο συμπέρασμα ότι η σχεσιακή προσέγγιση "καλά κρατεί" αλλά σίγουρα η κατασκευή εφαρμογών με χρήση των σχεσιακών βάσεων επηρεάστηκε καθοριστικά από το αντικειμενοστρεφές παράδειγμα. Όπως θα δούμε και στο κεφάλαιο για την ανάπτυξη εφαρμογών (με χρήση εργαλείων όπως JDBC API και βάση δεδομένων, PHP και βάση δεδομένων) η ανάπτυξη γίνεται όλο και περισσότερο σύμφωνα με τις αρχές του αντικειμενοστρεφούς παραδείγματος. Βέβαια εδώ και πολλά χρόνια μιλάμε για προϊόντα με μία ενοποιημένη προσέγγιση ΣΔΒΔ και αντικειμενοστρεφούς ΣΔΒΔ (βλέπε πχ. βιβλίο Stonebraker). Είναι σημαντικό να αναφέρουμε ότι τα γνωστά προϊόντα διαχείρισης σχεσιακής βάσης δεδομένων συμπληρώνονται με ενδιαφέροντα αντικειμενοστρεφή χαρακτηριστικά και επομένως μιλάμε πλέον για αντικειμενοσχεσιακό σύστημα βάσης δεδομένων (object-relational database system).

Σύμφωνα με το προϊόν PostgreSQL:

"Το προϊόν PostgreSQL είναι ένα ισχυρό, ανοιχτού κώδικα αντικειμενοσχεσιακό σύστημα βάσης δεδομένων που χρησιμοποιεί και επεκτείνει τη γλώσσα SQL σε συνδυασμό με πολλές δυνατότητες αποθήκευσης, επεξεργασίας και επέκτασης (κλιμάκωσης) με ασφάλεια σύνθετων φόρτων δεδομένων."

"PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads."

Σύμφωνα με το προϊόν Oracle:

«Η Oracle υλοποιεί το αντικειμενοστραφές σύστημα ως επέκταση του σχεσιακού μοντέλου. Η ανικειμενοστρεφής διεπαφή συνεχίζει να υποστηρίζει τις τυπικές λειτουργίες της σχεσιακής βάσης δεδομένων, όπως ερωτήματα (SELECT…FROM…WHERE), γρήγορες επικυρώσεις-καταχωρήσεις δεδομένων συναλλαγών, δημιουργία αντιγράφων και ανάκαμψη μετά από βλάβη, εύκολα κλιμακούμενη συνδεσιμότητα, κλείδωμα σε επίπεδο γραμμής, συνέπεια κατά την ανάγνωση δεδομένων, διαμέριση πινάκων, παράλληλα ερωτήματα, συσταδοποίηση βάσης δεδομένων, εξαγωγή και εισαγωγή, και φόρτωση δεδομένων. Η γλώσσα SQL και διάφορες προγραμματιστικές διεπαφές για την Oracle. συμπεριλαμβανομένων των PL/SQL, Java, Oracle Call Interface, Pro*C/C++ και OO4O. έχουν βελτιωθεί με νέες επεκτάσεις για την υποστήριξη αντικειμένων. Το αποτέλεσμα είναι ένα αντικειμενοσχεσιακό μοντέλο, το οποίο προσφέρει τη διαισθητική προσέγγιση και την οικονομία μιας αντικειμενοστρεφούς διεπαφής, διατηρώντας τον υψηλό ταυτοχρονισμό και την απόδοση μιας σχεσιακής βάσης δεδομένων.»

(βλέπε Oracle Database Application Developer's Guide - Object-Relational Features)

"Oracle implements the object-type system as an extension of the relational model. The object-type interface continues to support standard relational database functionality such as queries (SELECT…FROM…WHERE), fast commits, backup and recovery, scalable connectivity, row-level locking, read consistency, partitioned tables, parallel queries, cluster database, export and import, and loader. Plus SQL and various programmatic interfaces to Oracle; including PL/SQL, Java, Oracle Call Interface, Pro*C/C++, and OO4O; have been enhanced with new extensions to support objects. The result is an object-relational model, which offers the

intuitiveness and economy of an object interface while preserving the high concurrency and throughput of a relational database." (βλέπε Oracle Database Application Developer's Guide - Object-Relational Features)

1.4.3 Παραδείγματα Σχεσιακής βάσης δεδομένων

Στην Εικόνα 1.8 φαίνεται μία απλή βάση που περιλαμβάνει στοιχεία βιβλίων μιας προσωπικής βιβλιοθήκης. Όπως βλέπετε όλη η βάση είναι ένας πίνακας. Υπάρχουν προφανείς περιορισμοί στην προσέγγιση αυτή. Για παράδειγμα, λόγω του "απλοϊκού" σχεδιασμού καταχωρίζεται στη βάση δεδομένων μόνο το όνομα του πρώτου συγγραφέα των βιβλίων. Αν υποτεθεί ότι αποφασίζουμε να εισάγουμε σαν τιμή στο πεδίο "ΣΥΓΓΡΑΦΕΊΣ" όλους τους συγγραφείς χωρίζοντας τους με κάποιο ειδικό χαρακτήρα (delimiter), π.χ., Ξανθάκης%Σκουρλάς, θα έχουμε δυσκολίες στην αναζήτηση και την ανάκτηση βιβλίων με χρήση του ονόματος συγγραφέα. Επομένως, η δημιουργία προγραμμάτων αυτοματοποίησης των εργασιών μιας πραγματικής βιβλιοθήκης, η δημιουργία Βάσης Δεδομένων για τους Καταλόγους βιβλίων της βιβλιοθήκης κ.λπ. είναι ιδιαίτερα πολύπλοκη, τα προγράμματα εφαρμογής συνδέονται με πολλούς πίνακες δεδομένων και απαιτείται προσεκτική σχεδίαση.

Συγγραφείς	Τίτλος	Εκδότης
Ξανθάκης, Σκουρλάς	Έλεγχος προγραμμάτων	Νέες Τεχνολογίες
Σκουρλάς	Σχεσιακές βάσεις δεδομένων	Νέες Τεχνολογίες
Παναγιωτόπουλος, Δραγώνας, Σκουρλάς	Τηλεπληροφορική	Νέες Τεχνολογίες
Σκουρλάς	Βάσεις Δεδομένων	σε σι ελάσσονα

Εικόνα 1.8 Βάση δεδομένων βιβλίων προσωπικής βιβλιοθήκης

Έστω ότι η πολυεθνική εταιρεία ανταλλακτικών "Mythical Car" έχει μια βάση δεδομένων για τη διαχείριση των πελατών της, των ανταλλακτικών της κ.λπ. Στην Εικόνα 1.9 δίνεται ένα δείγμα δεδομένων (sample of data) για μία απλουστευμένη βάση "ΠΡΟΜΗΘΕΥΤΕΣ-ΑΝΤΑΛΛΑΚΤΙΚΑ". Η βάση αυτή μπορεί να χρησιμοποιείται από διάφορες εφαρμογές και τμήματα της εταιρείας. Το τμήμα Παραγγελιών χρησιμοποιεί τον πίνακα "Προμηθευτές". Το τμήμα Ανταλλακτικών χρησιμοποιεί τον πίνακα "Ανταλλακτικά" κ.λπ. Όπως φαίνεται στην Εικόνα 1.9 για τις διάφορες στήλες των πινάκων μπορούμε να χρησιμοποιήσουμε κατα τη δημιουργία της βάσης ονόματα με λατινικούς χαρακτήρες. Υπάρχουν κάποια προϊόντα που επιτρέπουν και ελληνικά ονόματα.

SUPPLIERS (Προμηθευτές)

SNO	SNAME	SCITY
S010	FIAT	ΛΟΝΔΙΝΟ
S020	OPEL	ΠΑΡΙΣΙ
S030	FORD	ΠΑΡΙΣΙ
S040	PORCHE	ΑΘΗΝΑ

PARTS (Ανταλλακτικά)

PNO	PNAME	PCOLOR	PCITY
P0100	ΒΙΔΑ Α	ΚΟΚΚΙΝΗ	ΛΟΝΔΙΝΟ
P0200	ΒΙΔΑ Β	ΚΟΚΚΙΝΗ	ΠΑΡΙΣΙ
P0300	ΒΙΔΑ C	ΚΟΚΚΙΝΗ	ΡΩΜΗ
P0400	ΒΙΔΑ C	ΚΙΤΡΙΝΗ	ΛΟΝΔΙΝΟ

SHIPMENT (Εφοδιασμός)

SNO	PNO	QTY
S010	P0100	300
S010	P0200	200
S010	P0300	400
S020	P0100	300
S020	P0200	400
S030	P0200	200

Εικόνα 1.9 Απλουστευμένη βάση δεδομένων "ΠΡΟΜΗΘΕΥΤΕΣ-ΑΝΤΑΛΛΑΚΤΙΚΑ"

Αν ήδη έχετε χρησιμοποιήσει τη γλώσσα SQL γνωρίζετε καλύτερα τι είναι ένας πίνακας, ένα ευρετήριο, μία ερώτηση. Πράγματι, η γλώσσα αυτή δίδει ένα απλό πλαίσιο για τη δημιουργία και διαχείριση της βάσης δεδομένων και των αντικειμένων της.

1.4.4 Βασικές έννοιες μοντελοποίησης σχεσιακών δεδομένων

Όταν θέλουμε να σχεδιάσουμε ένα σύστημα σχεσιακής βάσης δεδομένων αρχικά μοντελοποιούμε το σύστημα σχεδιάζοντας ένα εννοιολογικό μοντέλο (conceptual model), π.χ. το Μοντέλο Οντοτήτων Συσχετίσεων (ΜΟΣ), για όλες τις εφαρμογές που μας ενδιαφέρουν. Το μοντέλο αναπαριστά όλες τις οντότητες (entities) και τις μεταξύ τους συσχετίσεις (relationships). Στη συνέχεια εφαρμόζοντας κάποιους κανόνες που θα αναφέρουμε κατασκευάζουμε τη σχεσιακή βάση δεδομένων.

Ακολουθεί συζήτηση βασικών εννοιών:

1) Μοντελοποίηση δεδομένων και Μοντέλο Οντοτήτων Συσχετίσεων (ΜΟΣ).

   Εννοιολογική μοντελοποίηση είναι ένας τρόπος (μία μεθοδολογία ή τμήμα μεθοδολογίας) να περιγράψουμε τη βάση δεδομένων πριν σχεδιάσουμε τους πίνακες. Το αποτέλεσμα της μοντελοποίησης είναι το Μοντέλο Οντοτήτων Συσχετίσεων. Στο μοντέλο ορίζουμε οντότητες για τις οποίες πρέπει να καταχωρίζουμε στοιχεία, π.χ. Οντότητα Σπουδαστής, Οντότητα Μάθημα. Ορίζουμε και ποια στοιχεία (χαρακτηριστικά, attributes) θα καταχωρίζουμε στη βάση για κάθε οντότητα. Επιπλέον, ορίζουμε συσχετίσεις ανάμεσα σε οντότητες, π.χ. Ο Νίκος γράφτηκε στα μαθήματα Προγραμματισμός και Βάσεις Δεδομένων.

2) Στο ΜΟΣ, στην πραγματικότητα, δεν ορίζουμε οντότητες και συσχετίσεις αλλά τύπους οντοτήτων και τύπους συσχετίσεων. Για παράδειγμα, οντότητες είναι οι διάφοροι σπουδαστές, ο Κώστας, ο Γιάννης, η Βίλμα, ο Βαγγέλης. Χρησιμοποιούμε τον τύπο οντότητας Σπουδαστής ώστε να περιγράφουμε όλους τους σπουδαστές μας με τον ίδιο τρόπο. Τον τύπο οντότητας τον μετατρέπουμε σε πίνακα στη σχεσιακή βάση δεδομένων. Ο τύπος συσχέτισης συσχετίζει τύπους οντοτήτων. Ένας τύπος συσχέτισης δεν αντιστοιχεί πάντα σε πίνακα, π.χ. ο τύπος συσχέτισης 1:N (ένα-προς-πολλά)

3) Οι περιορισμοί (constraints, business rules) στα δεδομένα περιγράφουν τους περιορισμούς που ισχύουν για τα χαρακτηριστικά των οντοτήτων. Για παράδειγμα κάθε σπουδαστής έχει ένα μοναδικό κωδικό (τον αριθμό μητρώου του) που θα είναι το κλειδί της οντότητας. Επίσης, περιγράφουν τις συσχετίσεις οντοτήτων. Για παράδειγμα, κάθε σπουδαστής γράφεται σε πολλά μαθήματα και σε κάθε μάθημα γράφονται πολλοί σπουδαστές.

4) Σχέση ΜΟΣ με Σχεσιακό μοντέλο δεδομένων. Στη συνέχεια του κεφαλαίου θα μιλήσουμε για μια συνταγή που θα εφαρμόζουμε στο ΜΟΣ για να κατασκευάσουμε τη σχεσιακή βάση δεδομένων. Η συνταγή αυτή θα εμπλουτιστεί στο κεφάλαιο 6.

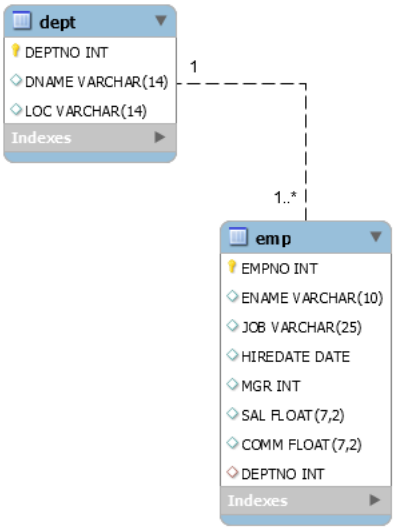
1.4.5 Σύνδεση της έννοιας της μοντελοποίησης και της έννοιας του συστήματος βάσης δεδομένων

Όταν σχεδιάζουμε ένα σύστημα βάσεων δεδομένων λέμε ότι πρέπει να αναπαριστά τις οντότητες (entities) της εφαρμογής αλλά και τις μεταξύ τους συσχετίσεις (ή σχέσεις, relationships). Συνήθως κατασκευάζουμε ένα μοντέλο οντοτήτων συσχετίσεων πριν την κατασκευή της εφαρμογής.

Σαν οντότητες θεωρούμε πρόσωπα, αντικείμενα, γεγονότα, πράξεις ή/και αφηρημένες έννοιες γύρω από τις οποίες θα συγκεντρωθεί και θα αποθηκευτεί πληροφορία. Κάθε οντότητα μπορεί να παρασταθεί συνήθως με ένα πίνακα.

Επειδή οι οντότητες έχουν κάποιες συσχετίσεις (relationships) μεταξύ τους πολλές φορές αυτές τις συσχετίσεις τις παριστάνουμε είτε με σχέσεις ανάμεσα στους πίνακες (που αναπαριστούν τις οντότητες) ή και με πίνακες.

Στην επόμενη Εικόνα 1.10 φαίνεται απόσπασμα μοντέλου οντοτήτων συσχετίσεων μίας βάσης δεδομένων προσωπικού. Το μοντέλο έχει σχεδιαστεί στο προϊόν MySQL workbench. Επιπλέον, παρατίθεται το αποτέλεσμα της λειτουργίας “MySQL Workbench-Forward Engineering”. Η διαδικασία που ακολουθήθηκε είναι ότι σχεδιάσαμε το μοντέλο και το προϊόν παρήγαγε τον κώδικα δημιουργίας της βάσης δεδομένων και των πινάκων της. Τέλος, στην Εικόνα παρατίθεται και δείγμα δεδομένων στους πίνακες dept, emp.




```
-- MySQL Workbench Forward Engineering
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,
ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
-- -----------------------------------------------------
-- Schema personnel
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `personnel` DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_0900_ai_ci ;
USE `personnel` ;
-- -----------------------------------------------------
```

```
-- Table `personnel`.`dept`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `personnel`.`dept` (
  `DEPTNO` INT NOT NULL,
  `DNAME` VARCHAR(14) NULL DEFAULT NULL,
  `LOC` VARCHAR(14) NULL DEFAULT NULL,
  PRIMARY KEY (`DEPTNO`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
-- -----------------------------------------------------
-- Table `personnel`.`emp`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `personnel`.`emp` (
  `EMPNO` INT NOT NULL,
  `ENAME` VARCHAR(10) NULL DEFAULT NULL,
  `JOB` VARCHAR(25) NULL DEFAULT NULL,
  `HIREDATE` DATE NULL DEFAULT NULL,
  `MGR` INT NULL DEFAULT NULL,
  `SAL` FLOAT(7,2) NULL DEFAULT NULL,
  `COMM` FLOAT(7,2) NULL DEFAULT NULL,
  `DEPTNO` INT NULL DEFAULT NULL,
  PRIMARY KEY (`EMPNO`),
  INDEX `DEPTNO` (`DEPTNO` ASC) VISIBLE,
  CONSTRAINT `emp_ibfk_1`
FOREIGN KEY (`DEPTNO`)
REFERENCES `newpersonnel`.`dept` (`DEPTNO`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

```
mysql> SELECT * FROM EMP;
+-------+---------+------------+------------+------+---------+---------+--------+
| EMPNO | ENAME   | JOB        | HIREDATE   | MGR  | SAL     | COMM    | DEPTNO |
+-------+---------+------------+------------+------+---------+---------+--------+
|    10 | CODD    | ANALYST    | 1989-01-01 |   15 | 3000.00 |    NULL |     10 |
|    15 | ELMASRI | ANALYST    | 1995-05-02 |   15 | 1200.00 |  150.00 |     10 |
|    20 | NAVATHE | SALESMAN   | 1977-07-07 |   20 | 2000.00 |    NULL |     20 |
|    30 | DATE    | PROGRAMMER | 2004-05-04 |   15 | 1800.00 |  200.00 |     10 |
+-------+---------+------------+------------+------+---------+---------+--------+
4 rows in set (0.00 sec)

mysql>            SELECT * FROM DEPT;
+--------+------------+----------+
| DEPTNO | DNAME      | LOC      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
+--------+------------+----------+
4 rows in set (0.00 sec)

mysql> _
```

Εικόνα 1.10 Απόσπασμα μοντέλου οντοτήτων συσχετίσεων μίας βάσης δεδοένων προσωπικού, δημιουργία κώδικα (forward engineering) και δείγμα δεδομένων πινάκων της βάσης

1.5 Μία απόπειρα σύνθεσης των ορισμών βάσεων δεδομένων Βασικές έννοιες, ακεραιότητα δεδομένων, κ.λπ.

Ορισμός βάσης δεδομένων

Μια Βάση Δεδομένων ενός Οργανισμού/Επιχείρησης είναι μια συλλογή αλληλοσχετιζόμενων, διαμοιραζόμενων (shared) λειτουργικών στοιχείων (operational data) και μόνιμων δεδομένων που αποθηκεύονται μαζί (για μεγάλο χρονικό διάστημα) χωρίς άχρηστους πλεονασμούς (redundancies) για την ταυτόχρονη εξυπηρέτηση πολλών εφαρμογών. Η βάση χρησιμοποιείται για συγκεκριμένους λόγους από μια ή περισσότερες ομάδες χρηστών. Ειδική μνεία πρέπει να γίνει για τη χρήση της στη στήριξη αποφάσεων (decision support) επικεφαλής και υψηλόβαθμων στελεχών.

Ακολουθεί μία λίστα από κρίσιμα χαρακτηριστικά μιας βάσης δεδομένων:

- Αναπαριστά κάποια όψη του πραγματικού κόσμου και τα δεδομένα της είναι γνωστά, "αναγνωρίσιμα" γεγονότα (events, facts) που μπορούν να καταγραφούν και έχουν κάποια σημασία για τους χρήστες.
- Η αποθήκευση των δεδομένων είναι τέτοια ώστε τα δεδομένα να είναι ανεξάρτητα των προγραμμάτων που τα διαχειρίζονται.
- Η εισαγωγή νέων δεδομένων, η τροποποίηση και η ανάκτηση δεδομένων από τη Βάση Δεδομένων ακολουθεί κοινή και ελεγχόμενη προσέγγιση για όλες τις εφαρμογές.
- Η οργάνωση των δεδομένων είναι τέτοια ώστε μπορούμε να υλοποιήσουμε νέες εφαρμογές χωρίς ιδιαίτερο κόπο και χωρίς να αλλάξει οτιδήποτε στις παλιές.
- Η οργάνωση των δεδομένων διευκολύνει, επίσης, την υλοποίηση νέων εφαρμογών στήριξης αποφάσεων.

Παρατήρηση

Ήδη νεότερες έννοιες, καινοτομίες, νεότερα προϊόντα και η έρευνα ετών άλλαξε αρκετά την παραπάνω εικόνα. Η συζήτηση για βάσεις πολλαπλών μέσων, αντικειμενοστρεφείς βάσεις, αποθήκες δεδομένων (data warehousing), βάσεις στο διαδίκτυο, ενοποίηση Information Retrieval και DBMS κ.λπ. καθιστούν στην πράξη και αυτόν τον ορισμό κάπως ξεπερασμένο.

Στη συνέχεια θα συζητήσουμε την έννοια του Συστήματος Βάσης Δεδομένων (ΣΒΔ) βασιζόμενοι σε γνωστά συγγράμματα αλλά και την εμπειρία από τη χρήση γνωστών προϊόντων.

1.5.1 Σύστημα βάσης δεδομένων (database system)

Ένα σύστημα βάσης δεδομένων (ΣΒΔ), πέρα από τους διάφορους ορισμούς που μπορούν να διατυπωθούν, δεν παύει να είναι ένα πληροφοριακό σύστημα καταχώρησης, ενημέρωσης και αναζήτησης πληροφοριών βασιζόμενο σε υπολογιστή. Επομένως, οποιοδήποτε ΣΒΔ αποτελείται από συνιστώσες:

- οργανωτική δομή (οργανόγραμμα) για την υποστήριξη της λειτουργίας του
- μέσα ή πόρους όπως υλικό (hardware) και λοιπό εξοπλισμό, δεδομένα (data) οργανωμένα σε βάση δεδομένων και λογισμικό (software), με κυριότερο στοιχείο του το αντικειμενοστρεφές ΣΔΒΔ
- χρήστες (users), π.χ., προσωπικό (και ρόλοι), πελάτες κ.λπ.
- διαδικασίες και λειτουργίες, π.χ., λήψη εφεδρικών αντιγράφων (back-up), ανάνηψη από βλάβες (recovery), ρύθμιση (tuning), βελτιστοποίηση ερωτημάτων (query optimization).

Για να συνδεθούμε με τη διαδικασία μοντελοποίησης πρέπει να επισημάνουμε ότι ένα σύστημα βάσεων δεδομένων, επίσης, πρέπει να αναπαριστά τις οντότητες (entities) και τις μεταξύ τους συσχετίσεις (relationships).

Ακολουθεί σύντομη συζήτηση των βασικών στοιχείων ενός συστήματος βάσεως δεδομένων:

1. **Δεδομένα (data).** Τα δεδομένα της βάσεως θεωρούνται ενοποιημένα – ολοκληρωμένα (integrated) και διαμοιραζόμενα ή κοινόχρηστα (shared):

   Ενοποίηση δεδομένων (data integration) (Date)

   Η βάση δεδομένων μπορεί να θεωρείται ως μία συνένωση πολλών αρχείων (πινάκων) δεδομένων, που από κάθε άποψη είναι ξεχωριστά το ένα από το άλλο, ενώ κάθε πλεονασμός εξαιτίας της επανάληψης δεδομένων μεταξύ αυτών των αρχείων έχει εξαλειφθεί εντελώς ή τουλάχιστον κατά ένα μέρος.

   Μερισμός δεδομένων (data sharing) (Date)

   Τα μεμονωμένα στοιχεία δεδομένων της βάσης δεδομένων μπορούν να τα μοιράζονται πολλοί διαφορετικοί χρήστες, με την έννοια ότι κάθε ένας από αυτούς μπορεί να έχει πρόσβαση στο ίδιο στοιχείο δεδομένων την ίδια στιγμή ("ταυτόχρονη προσπέλαση", concurrency) και οι διάφοροι χρήστες μπορούν να το χρησιμοποιούν για διαφορετικό σκοπό.

   Οι Elmasri-Navathe συζητώντας για το περιβάλλον ενός συστήματος βάσης δεδομένων εστιάζουν και σε:

   Μεταδεδομένα (metadata)

   Αποθηκεύονται σε κατάλογο (catalog) ή λεξικό δεδομένων (data dictionary) του συστήματος ο ορισμός (η περιγραφή) της βάσης και των αντικειμένων της, π.χ., πίνακες, ευρετήρια, όψεις

   Αφαίρεση δεδομένων (data abstraction)

   Πρέπει να διαχωρίζεται η εννοιολογική αναπαράσταση των δεδομένων από τις λεπτομέρειες της αναπαράστασης και της αποθήκευσης τους που δεν ενδιαφέρουν τους χρήστες.

2. **Υλικό (Hardware)**

   Σε επόμενη ενότητα γίνεται αναφορά στο υλικό που υποστηρίζει συστήματα βάσεων δεδομένων σήμερα και τις προβλέψεις για το μέλλον. Οι κατανεμημένες βάσεις δεδομένων, η διαχείριση δεδομένων μεγάλης κλίμακας (big data) και η υπολογιστική νέφους (cloud computing) αλλάζουν καθημερινά το τοπίο.

3. **Λογισμικό (Software)**

   Σημαντικότερο στοιχείο του συστήματος βάσεως δεδομένων είναι το σύστημα διαχείρισης βάσεως δεδομένων. Άλλα στοιχεία λογισμικού είναι τα προγράμματα εφαρμογών, τα βοηθητικά προγράμματα, γεννήτριες εφαρμογών (application generator), γεννήτριες εκτυπώσεων (report generator) κ.λπ.

   Στο σύγγραμμά τους οι Elmasri-Navathe υπογραμμίζουν ότι τα προγράμματα εφαρμογών (προσπέλασης της βάσης κ.λπ.) δεν ενσωματώνουν τη δομή (των αρχείων) δεδομένων που αποθηκεύονται ξεχωριστά στον κατάλογο του ΣΔΒΔ. Έτσι υποστηρίζεται η ανεξαρτησία προγραμμάτων δεδομένων (program - data independence) και επομένως οι αλλαγές στα αρχεία δεν απαιτούν αλλαγές και στα προγράμματα. Ειδικά για τα αντικειμενοστρεφή ΣΔΒΔ ο όρος διευρύνεται και όχι μόνο η δομή δεδομένων αλλά και πράξεις επί των δεδομένων διαχωρίζονται από τις

λεπτομέρειες (της «εσωτερικής» υλοποίησης κ.λπ.) που δεν ενδιαφέρουν άμεσα τα προγράμματα των χρηστών. Δηλαδή έχουμε και την ανεξαρτησία προγραμμάτων-πράξεων (program-operation independence).

4. **Χρήστες (Users)**

   Υπάρχουν τέσσερις κύριες κατηγορίες χρηστών:

   - Προγραμματιστές εφαρμογών
   - Απλοί (τελικοί) χρήστες (end-users)
   - Υπεύθυνος Διαχείρισης βάσεων (Data Base Administrator - DBA)
   - Υπεύθυνος Διαχείρισης Δεδομένων (Data Administrator)

Data Administrator

Παίρνει αποφάσεις στρατηγικής/πολιτικής αναφορικά με τα δεδομένα της επιχείρησης, αποφασίζει δηλαδή ποια δεδομένα θα αποθηκευτούν, ποια άτομα μπορούν να διαχειρίζονται δεδομένα, ποια θα είναι η πολιτική για την ασφάλεια των δεδομένων κ.λπ.

Κατά τον Date, ο Data Administrator πρέπει να βρίσκεται σε ανώτερο διοικητικό επίπεδο. Δηλαδή σύμφωνα με την άποψή του δεν είναι απαραίτητο να είναι τεχνικός.

Data Base Administrator (DBA)

Στην πράξη πρόκειται για φυσικό πρόσωπό ή συνήθως για ομάδα η οποία παρέχει την τεχνική υποστήριξη για την υλοποίηση των αποφάσεων του DA και έχει την ευθύνη της αποδοτικής λειτουργίας του συστήματος. Πιο συγκεκριμένα αναλαμβάνει τα παρακάτω:

α. Αφού αποφασιστεί το περιεχόμενο της βάσης δημιουργεί το αντίστοιχο εννοιολογικό (ή καθολικό) σχήμα (conceptual schema)

β. Αποφασίζει πως θα αποθηκευτούν τα δεδομένα στη βάση και δημιουργεί τους αντίστοιχους ορισμούς αποθηκευτικής δομής, δηλαδή το εσωτερικό σχήμα (internal schema) .

γ. Υποβοήθηση χρηστών και δημιουργία εξωτερικών σχημάτων (external schema) που επιτρέπουν πρόσβαση σε και διαχείριση ενός υποσυνόλου της βάσης.

δ. Ορισμός κανόνων ασφαλείας και ακεραιότητας δεδομένων

ε. Ορισμός διαδικασιών λήψης περιοδικά αντιγράφων της βάσης σε μέσα εφεδρικής αποτύπωσης (back-up), ορισμός διαδικασιών επαναφόρτωσης (ανάκαμψη, recovery) της βάσης από εφεδρικά αντίγραφα (back up) κ.λπ.

στ. Μελέτη της απόδοσης (performance evaluation), περιοδική αναδιοργάνωση της βάσης, ικανοποίηση νέων απαιτήσεων κ.λπ.

Οι Elmasri-Navathe διαχωρίζουν τις ομάδες χρηστών σε :

- DBA - Data Base Administrator
- Σχεδιαστές
- Τελικοί χρήστες. Υψηλόβαθμα ή μέσα διοικητικά στελέχη ή άλλοι περιστασιακοί χρήστες, «απλοϊκοί» τελικοί χρήστες, εξειδικευμένοι τελικοί χρήστες (power users)
- αναλυτές συστημάτων
- προγραμματιστές (developer)

Επίσης, υπογραμμίζουν ότι πρέπει να υποστηρίζονται πολλαπλές όψεις των δεδομένων (views), δηλαδή στο ΣΒΔ πρέπει να εξασφαλίζεται ότι κάθε χρήστης μπορεί να βλέπει και να χειρίζεται το υποσύνολο της βάσης δεδομένων που τον ενδιαφέρει.

1.6 Πρώτη αναφορά σε μία συστηματική προσέγγιση στη σχεδίαση της σχεσιακής βάσης δεδομένων. Εισαγωγή στη χρήση συναρτησιακών εξαρτήσεων (functional dependencies)

Η βάση δεδομένων αναπαριστά κάποια στοιχεία-δεδομένα (data) του «πραγματικού» κόσμου. Τα δεδομένα ικανοποιούν περιορισμούς (constraints, business rules). Ακολουθούν παραδείγματα περιγραφής των περιορισμών:

1. Κάθε τμήμα έχει έναν μοναδικό κωδικό αριθμό (deptNumber) και μοναδικό όνομα (deptName).
2. *deptNumber → deptName* (μαθηματική αναπαράσταση του περιορισμού ως συναρτησιακής εξάρτησης). Το όνομα του τμήματος (deptName) εξαρτάται συναρτησιακά από τον κωδικό του τμήματος (deptNumber).
3. *deptName → deptNumber* (μαθηματική αναπαράσταση του περιορισμού ως συναρτησιακής εξάρτησης). Ο κωδικός του τμήματος (deptNumber) εξαρτάται συναρτησιακά από το όνομα του τμήματος (deptName).
4. Στην Εικόνα 1.11 βλέπουμε πίνακα (σχέση σε ορολογία Codd) στον οποίο η στήλη deptNumber μπορεί να είναι πρωτεύον κλειδί. Ο πίνακας (σχέση) «οπτικοποιεί» τη συναρτησιακή εξάρτηση *deptNumber → deptName*
5. Στην ίδια Εικόνα βλέπουμε πίνακα στον οποίο η στήλη deptName μπορεί να είναι πρωτεύον κλειδί. Ο πίνακας (σχέση) «οπτικοποιεί» τη συναρτησιακή εξάρτηση *deptName → deptNumber*

deptNumber	deptName

Ο πίνακας «οπτικοποιεί» τη συναρτησιακή εξάρτηση (αντιστοιχεί στη ΣΕ) deptNumber → deptName

deptName	deptNumber

Ο πίνακας «οπτικοποιεί» τη συναρτησιακή εξάρτηση (αντιστοιχεί στη ΣΕ)
deptName → deptNumber

Εικόνα 1.11 Η στήλη deptName μπορεί να είναι πρωτεύον κλειδί.1

Σε επόμενο κεφάλαιο θα μελετήσουμε τη διαδικασία κανονικοποίησης, ακριβέστερα της κατασκευής της κανονικής μορφής BCNF, ως εξής:

1. Καταγραφή όλων των περιορισμών υπό μορφή ΣΕ
2. Απαλοιφή «περιττών» («αποδείξιμων») ΣΕ
3. Σύμπτυξη των ΣΕ της μορφής a -->b, a -->c, ... σε μία a -->b,c, ...
4. Οι προκύπτουσες ΣΕ αποτελούν τις σχέσεις-πίνακες της βάσης δεδομένων.

Επομένως πριν σχεδιάσουμε τη βάση δεδομένων μελετούμε μία λεκτική περιγραφή, όπως η παρακάτω για να καθορίσουμε περιορισμούς-κανόνες της επιχείρησης. Οι περιορισμοί μπορούν να περιγραφούν με ΣΕ.

1.6.1 Εξαγωγή των κανόνων–περιορισμών από την περιγραφή της επιχείρησης

Μια εταιρεία είναι οργανωμένη σε τμήματα (departments). Κάθε τμήμα έχει ένα μοναδικό όνομα, έναν μοναδικό αριθμό, έναν εργαζόμενο (employee) που το διευθύνει (manages) και έναν αριθμό εργαζομένων που εργάζεται σε αυτό. Ένα τμήμα ελέγχει (controls) αποκλειστικώς έναν αριθμό έργων (projects) καθένα από τα οποία έχει ένα μοναδικό όνομα, έναν μοναδικό αριθμό και εκτελείται σε μια τοποθεσία. Για κάθε εργαζόμενο κρατούμε: αριθμό ταυτότητας, το πλήρες όνομα (επώνυμο, όνομα, όνομα πατέρα), διεύθυνση, φύλο, μισθό. Κάθε εργαζόμενος ανήκει σε ένα τμήμα αλλά εργάζεται σε διάφορα έργα που δεν ελέγχονται κατ' ανάγκη από το τμήμα του. Για κάθε εργαζόμενο κρατούμε τις ώρες που εργάζεται για κάθε έργο. Για ασφαλιστικούς λόγους κρατούμε τα στοιχεία των μελών της οικογένειας κάθε εργαζόμενου που είναι εξαρτώμενα από αυτόν: όνομα, φύλο, ημερομηνία γέννησης και σχέση με τον εργαζόμενο.

Κανόνας της Επιχείρησης

Κάθε τμήμα έχει ένα μοναδικό όνομα (deptName), έναν μοναδικό αριθμό (deptNumber), έναν εργαζόμενο (mngrIdNum)που το διευθύνει. Να αναλυτικά οι περιορισμοί που ισχύουν.

deptName → deptNumber

deptNumber → deptName

deptNumber → mngrIdNum

deptName → mngrIdNum

Κανόνας της Επιχείρησης

Κρατούμε πάντοτε την ημερομηνία (StartDate) που ανέλαβε τη διεύθυνση του τμήματος ο σημερινός διευθυντής, ο οποίος δεν μπορεί να διευθύνει δεύτερο Τμήμα. Μήπως μας ενδιαφέρει πότε ο σημερινός διευθυντής ανέλαβε το τμήμα για πρώτη φορά; Αν μας ενδιαφέρει κάτι τέτοιο (είναι η περίπτωση που ενδιαφέρουν τα «ιστορικά» στοιχεία) τότε μας ενδιαφέρει η σχέση (ο πίνακας):

deptNumber, mngrIdNum, startDate → θ,

όπου θ είναι χαρακτηριστικό (attribute) σχέσης (πίνακα) ή τίποτα.

Κανόνας της Επιχείρησης

Οι δραστηριότητες του τμήματος απλώνονται σε πολλές τοποθεσίες (deptLocation). Αυτός ο κανόνας δεν μας επιτρέπει να γράψουμε τη σχέση deptName → deptLocation αφού αυτό θα σήμαινε ότι το τμήμα είναι εγκατεστημένο σε μία και μόνο τοποθεσία. Η σωστή σχέση (πίνακας) είναι:

deptNumber, deptLocation → θ ή εναλλακτικά η σχέση

deptName, deptLocation → θ

Ερώτηση

Τι θα σήμαινε η συναρτησιακή εξάρτηση deptLocation → deptName;

Θα σήμαινε ότι σε κάθε τοποθεσία υπάρχει ένα μόνο τμήμα της εταιρίας, π.χ. στην Πανεπιστημίου έχουμε μόνον το Λογιστήριο, στην Ακαδημίας έχουμε τις Πωλήσεις, στην Καλλιθέα τη Διοίκηση, στο Πικέρμι την Παραγωγή (εργοστάσιο), στα Οινόφυτα έχουμε επίσης Παραγωγή (δεύτερο εργοστάσιο) κ.λπ.

1.6.2 Κανόνες ακεραιότητας. Πρώτη (περιγραφική) προσέγγιση

Στη βάση δεδομένων της εικόνας 1.12 παραβιάζονται οι δύο κανόνες ακεραιότητας.

Dept (στοιχεία τμήματος)

DeptNo	Dname
10	ΠΩΛΗΣΕΙΣ
20	ΛΟΓΙΣΤΗΡΙΟ
30	ΜΙΣΘΟΔΟΣΙΑ
NULL	MARKETING
NULL	ΕΡΕΥΝΑ

Emp (στοιχεία υπαλλήλου)

Empno	Surname	Name	Job	DeptNo	Sal
Κωδικός	Επώνυμο	Όνομα	Θέση	Κωδικός τμήματος	Μισθός
10	ΜΑΡΚΟΥ	ΜΑΡΚΟΣ	ΠΩΛΗΤΗΣ	10	4200
20	ΣΠΥΡΟΥ	ΣΠΥΡΟΣ	ΠΩΛΗΤΗΣ	10	4000
30	ΧΡΗΣΤΟΥ	ΧΡΗΣΤΟΣ	ΑΝΑΛΥΤΗΣ	20	3000
30	ΣΠΥΡΟΥ	ΝΙΚΟΣ	ΑΝΑΛΥΤΗΣ	10	3000
40	ΝΙΚΟΥ	ΝΙΚΟΣ	ΧΕΙΡΙΣΤΗΣ	50	1500
NULL	ΝΙΚΟΥ	ΝΙΚΟΣ	ΧΕΙΡΙΣΤΗΣ	30	1500

Εικόνα 1.12 Σχεδίαση πινάκων που παραβιάζει τους κανόνες ακεραιότητας

Στον πίνακα dept παραβιάζεται ο πρώτος Κανόνας Ακεραιότητας (Integrity Rule I) επειδή ενώ θεωρούμε ότι κύριο κλειδί (primary key) είναι η στήλη DeptNo επιτρέψαμε να μην έχει τιμή. Ένα κύριο κλειδί πρέπει να έχει μοναδική τιμή για κάθε τμήμα. Στον δεύτερο πίνακα παραβιάζεται, επίσης, ο πρώτος Κανόνας Ακεραιότητας. Δείτε ότι το κύριο κλειδί (Primary Key) Empno δεν έχει μοναδική τιμή για κάθε υπάλληλο. Επιπλέον, παραβιάζεται ο δεύτερος κανόνας ακεραιότητας ή κανόνας ακεραιότητας αναφοράς (Integrity Rule II ή Referential Integrity Rule). Στη στήλη deptno που θεωρούμε ότι είναι ξένο κλειδί (Foreign key) (δηλαδή είναι primary key στον άλλο πίνακα) έχουμε τιμή 50 που δεν αντιστοιχεί σε τμήμα της εταιρείας (δεν υπάρχει στον πίνακα dept). Επομένως, αν ορίζουμε κύρια και ξένα κλειδιά τότε διασφαλίζουμε ότι ισχύουν ο πρώτος και ο δεύτερος κανόνας ακεραιότητας. Ακολουθεί ορισμός βάσης δεδομένων σε MySQL που διασφαλίζει τους κανόνες ακεραιότητας:

```
CREATE DATABASE personnel;
USE personnel;
CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL,
 DNAME VARCHAR(14), LOC VARCHAR(14),
 PRIMARY KEY(DEPTNO));
CREATE TABLE EMP(EMPNO INT(4) NOT NULL,
  ENAME VARCHAR(10), JOB VARCHAR(25),
  HIREDATE DATE, MGR INT(4),  SAL FLOAT(7,2), COMM FLOAT(7,2),
  DNO INT(2),
  PRIMARY KEY(EMPNO),
  FOREIGN KEY(DNO) REFERENCES DEPT(DEPTNO));
```

1.7 Κανονικοποίηση.

Στα παραδείγματα που ακολουθούν αναφερόμαστε στη σχεδίαση της σχεσιακής βάσης δεδομένων διαδοχικά στην πρώτη κανονική μορφή, στη δεύτερη κανονική μορφή και στην τρίτη κανονική μορφή.

1.7.1 Αρχή με παράδειγμα. Κανονικοποίηση της βάσης δεδομένων Συλλόγου Γονέων και Κηδεμόνων.

Έστω ότι ο σύλλογος γονέων και κηδεμόνων της εταιρείας Strategy for Data Modelling έχει μία βάση δεδομένων των υπαλλήλων της εταιρείας που είναι γονείς και κηδεμόνες ανήλικων τέκνων. Σε έναν πίνακα (ή εναλλακτικά σε περισσότερους) αναγράφουμε όλα τα στοιχεία των υπαλλήλων και των τέκνων τους. Δηλαδή στον πίνακα Employee όλοι οι υπάλληλοι έχουν παιδί ή παιδιά.

Περιορισμοί

Υποτίθεται ότι κάθε υπάλληλος έχει μία θέση, ανήκει σε ένα τμήμα, ο μισθός του εξαρτάται από τη θέση και έχει ανήλικο παιδί ή παιδιά.

Πρώτη κανονική μορφή (First Normal Form, 1NF)

Στην πρώτη κανονική μορφή της εικόνας 1.13 έχουμε σε έναν πίνακα όλες τις στήλες της βάσης δεδομένων. Στον πίνακα δεν υπάρχουν σύνθετες στήλες και έχουμε απλές τιμές για κάθε στήλη. Οι στήλες του είναι οι εξής: Empno=Κωδικός υπαλλήλου, Name=όνομα, JobNo=κωδικός θέσης, Job=θέση, Deptno=κωδικός τμήματος, Dname=τμήμα Sal=μισθός, C_No=αριθμός παιδιών υπαλλήλου, C_Name=όνομα παιδιού, B_Date= ημερομηνία γέννησης παιδιού.

Περιορισμός: Κάθε υπάλληλος έχει ένα ή περισσότερα παιδιά.

Employee

Empno	Name	JobNo	Job	DeptNo	Dname	Sal	C_no	C_Name	B_date
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2	ΜΑΡΙΑ	10/01/2009
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2	ΙΩΑΝΝΗΣ	20/03/2010
30	ΝΙΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΜΙΣΘΟΔΟΣΙΑ	1000	1	ΘΩΜΑΣ	10/06/2009

Σύνθετο κύριο κλειδί: (empno, c_name)

Εναλλακτική σχεδίαση της πρώτης κανονικής μορφής

Empno	Name	JobNo	Job	DeptNo	Dname	Sal	C_no
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2
30	ΝΙΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΜΙΣΘΟΔΟΣΙΑ	1000	1

κύριο κλειδί: empno

Empno	C_Name	B_date
10	ΜΑΡΙΑ	10/01/2009
10	ΙΩΑΝΝΗΣ	20/03/2010
30	ΘΩΜΑΣ	10/06/2009

Σύνθετο κύριο κλειδί: (empno, c_name)

Εικόνα 1.13 Πρώτη κανονική μορφή. Όλα τα στοιχεία της βάσης δεδομένων σε έναν πίνακα και εναλλακτική σχεδίαση της βάσης δεδομένων με όλα τα στοιχεία σε δύο πίνακες

Εναλλακτικά η 1NF θα μπορούσε να περιλαμβάνει περισσότερους πίνακες, π.χ., emp(empno, name, jobno, job, deptno, dname, sal, C_no), child(empno, c_name, b_date) (Εικόνα 1.13).

Στην Εικόνα 1.14 βλέπουμε μη κανονικοποιημένους πίνακες, δηλαδή πίνακες που παραβιάζουν την πρώτη κανονική μορφή. Παρατηρήστε ότι στον πρώτο πίνακα υπάρχουν σύνθετες στήλες και στον δεύτερο υπάρχουν στήλες που έχουν πολλαπλές τιμές.

Empno	Name	(JobNo, Job)	DeptNo	Dname	Sal	C_no	C_Name	B_date
10	ΣΠΥΡΟΥ	(100, ΠΩΛΗΤΗΣ)	50	ΠΩΛΗΣΕΙΣ	2200	2	ΜΑΡΙΑ	10/01/2009
10	ΣΠΥΡΟΥ	(100, ΠΩΛΗΤΗΣ)	50	ΠΩΛΗΣΕΙΣ	2200	2	ΙΩΑΝΝΗΣ	20/03/2010
30	ΝΙΚΟΥ	(300, ΧΕΙΡΙΣΤΗΣ)	70	ΜΙΣΘΟΔΟΣΙΑ	1000	1	ΘΩΜΑΣ	10/06/2009

Στον πίνακα υπάρχουν σύνθετες στήλες

Empno	Name	JobNo	Job	DeptNo	Dname	Sal	C_no	C_Name	B_date
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2	ΜΑΡΙΑ ΙΩΑΝΝΗΣ	10/01/2009 20/03/2010
30	ΝΙΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΜΙΣΘΟΔΟΣΙΑ	1000	1	ΘΩΜΑΣ	10/06/2009

Στον πίνακα υπάρχουν στήλες που έχουν πολλαπλές τιμές

Εικόνα 1.14 Παραδείγματα μη κανονικοποιημένων πινάκων

Δεύτερη κανονική μορφή (2NF)

Θα εξετάσουμε για όλα τα τμήματα του σύνθετου κύριου κλειδιού αν μπορούν να αποτελέσουν κύριο κλειδί για κάποιες στήλες του πίνακα (ή των πινάκων) της πρώτης κανονικής μορφής. Με τον τρόπο αυτό διασπάται ο πίνακα (ή οι πίνακες) της πρώτης κανονικής μορφής. Κάθε τμήμα του κύριου κλειδιού γίνεται κύριο κλειδί για κάποιες στήλες σε ξεχωριστό πίνακα. Στη συνέχεια εξετάζουμε τις περιπτώσεις:

empno →

c_name →

(empno, c_name) →

Στην Εικόνα 1.15 βλέπουμε τη δεύτερη κανονική μορφή!

Employee

Empno	Name	JobNo	Job	DeptNo	Dname	Sal	C_no
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2
20	ΧΡΗΣΤΟΥ	200	ΑΝΑΛΥΤΗΣ	60	ΛΟΓΙΣΤΗΡΙΟ	2000	
30	ΝΙΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΜΙΣΘΟΔΟΣΙΑ	1000	1

Κύριο κλειδί: (empno)

Child

Empno	C_Name	B_date
10	ΜΑΡΙΑ	10/01/2009
10	ΙΩΑΝΝΗΣ	20/03/2010

30	ΘΩΜΑΣ	10/06/2009

Κύριο κλειδί: (empno, c_name)

Προαιρετικά και ο πίνακας Names

C_Name
ΘΩΜΑΣ
ΙΩΑΝΝΗΣ
ΜΑΡΙΑ

Κύριο κλειδί: (c_name)

Εικόνα 1.15 Δεύτερη κανονική μορφή

Παρατηρήσεις

1) Αν στην 1NF είχαμε τους πίνακες emp, child τότε είμασταν ήδη στη 2NF.

2) Αν θέλουμε, μπορούμε να κρατήσουμε στη 2NF και τον πίνακα Names με κύριο κλειδί c_names. Ορίζοντας τη στήλη c_names ως ξένο κλειδί στον πίνακα child έχουμε περιορισμό ότι στον πίνακα αυτόν μπορούμε να εισάγουμε μόνο ονόματα παιδιών που ήδη υπάρχουν στον πίνακα child.

Τρίτη κανονική μορφή 3NF

Εξετάζουμε όλους τους πίνακες της δεύτερης κανονικής μορφής. Για κάθε πίνακα εξετάζουμε αν υπάρχει στήλη ή σύνθετες στήλες εκτός κύριου κλειδιού που μπορούν να αποτελέσουν κύριο κλειδί για κάποιες άλλες στήλες του πίνακα (για στήλες που δεν περιλαμβάνονται, επίσης, στο κύριο κλειδί). Στην περίπτωση αυτή μιλάμε για το φαινόμενο μεταβατικής εξάρτησης στον πίνακα. Τότε διασπάται ο πίνακας της δεύτερης κανονικής μορφής ώστε να εξαλείψουμε το φαινόμενο. Στην Εικόνα 1.16 βλέπουμε δύο περιπτώσεις μεταβατικής εξάρτησης.

Employee

Empno	Name	JobNo	Job	DeptNo	Dname	Sal	C_no
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2
20	ΧΡΗΣΤΟΥ	200	ΑΝΑΛΥΤΗΣ	60	ΛΟΓΙΣΤΗΡΙΟ	2000	
30	ΝΙΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΜΙΣΘΟΔΟΣΙΑ	1000	1

Κύριο κλειδί=Empno

Στον πίνακα βλέπουμε δύο περιπτώσεις μεταβατικής εξάρτησης.

Jobno -- > Job, Sal

Deptno -- > Dname

Άρα κατασκευάζουμε τους αντίστοιχους πίνακες

Job

JobNo	Job	Sal
100	ΠΩΛΗΤΗΣ	2200
200	ΑΝΑΛΥΤΗΣ	2000
300	ΧΕΙΡΙΣΤΗΣ	1000

Κύριο κλειδί =jobno

Dept

DeptNo	Dname

50	ΠΩΛΗΣΕΙΣ
60	ΛΟΓΙΣΤΗΡΙΟ
70	ΜΙΣΘΟΔΟΣΙΑ

Κύριο κλειδί =deptno

Εικόνα 1.16 Στον πίνακα Employee βλέπουμε δύο περιπτώσεις μεταβατικής εξάρτησης. Οι αντίστοιχοι πίνακες Job, Dept μαζί με ένα τμήμα του πίνακα Employee περιλαμβάνονται στην τρίτη κανονική μορφή.

Στην Εικόνα 1.17 βλέπουμε την τρίτη κανονική μορφή. Αν θέλουμε μπορούμε να συμπεριλάβουμε στην 3NF και τον πίνακα Names.

Τρίτη Κανονική Μορφή της βάσης δεδομένων γονέων και κηδεμόνων

Employee

Empno	Name	JobNo	DeptNo	C_no
10	ΣΠΥΡΟΥ	100	50	2
20	ΧΡΗΣΤΟΥ	200	60	
30	ΝΙΚΟΥ	300	70	1

Κύριο κλειδί: empno

Jobs

JobNo	Job	Sal
100	ΠΩΛΗΤΗΣ	2200
200	ΑΝΑΛΥΤΗΣ	2000
300	ΧΕΙΡΙΣΤΗΣ	1000

Κύριο κλειδί: JobNoΚύριο κλειδί: deptno

Dept

DeptNo	Dname
50	ΠΩΛΗΣΕΙΣ
60	ΛΟΓΙΣΤΗΡΙΟ
70	ΜΙΣΘΟΔΟΣΙΑ

Child

Empno	C_Name	B_date
10	ΜΑΡΙΑ	10/01/2009
10	ΙΩΑΝΝΗΣ	20/03/2010
30	ΘΩΜΑΣ	10/06/2009

Κύριο κλειδί: (empno, c_name)

Αν θέλουμε κρατάμε και τον πίνακα Names στην 3NF.

Names

C_Name
ΘΩΜΑΣ
ΙΩΑΝΝΗΣ
ΜΑΡΙΑ

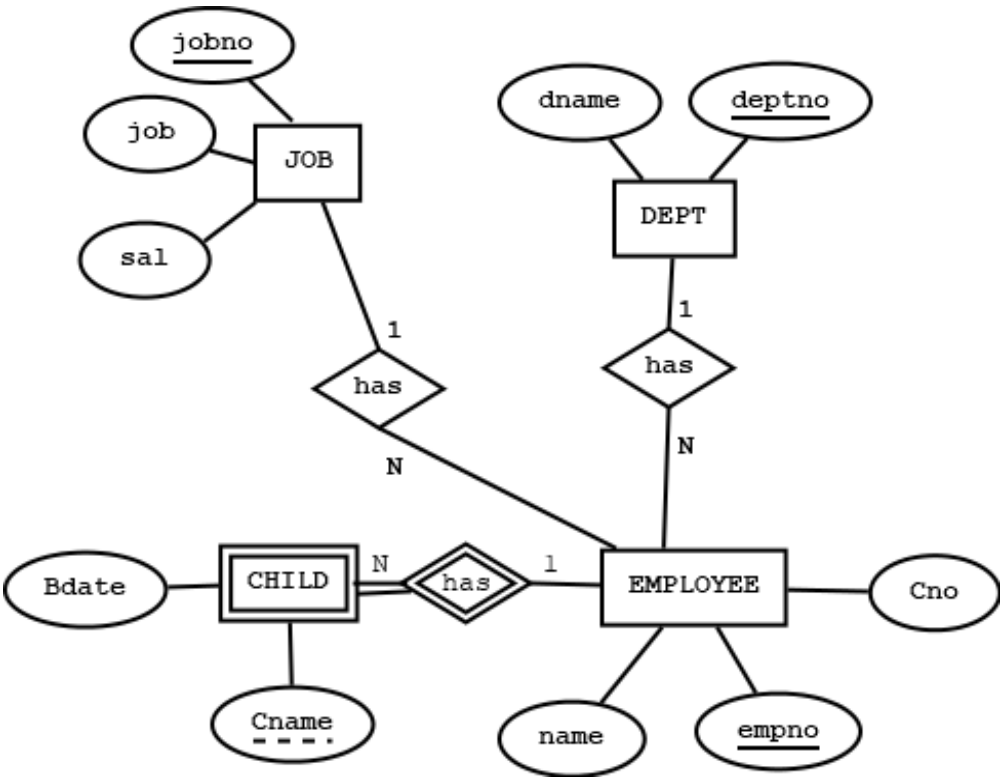
Κύριο κλειδί: (c_name)

Εικόνα 1.17 Τρίτη κανονική μορφή

1.7.2 Πρώτη αναφορά στο Μοντέλο Οντοτήτων Συσχετίσεων

Ακολουθεί παράδειγμα μοντέλου οντοτήτων συσχετίσεων (Entity Relationship Model) (Εικόνα 1.18). Στο μοντέλο βλέπουμε τις οντότητες EMPLOYEE (υπάλληλοι), JOB (θέσεις), DEPT (τμήματα) γραμμένες σε ορθογώνια παραλληλόγραμμα. Κάθε χαρακτηριστικό (attribute) οντότητας γράφεται σε έλλειψη και επιπλέον αν είναι κλειδί της υπογραμμίζεται. Η οντότητα Employee έχει χαρακτηριστικά empno, name, c_no και το κλειδί της είναι το χαρακτηριστικό empno. Η οντότητα Dept έχει χαρακτηριστικά depno, dname και το κλειδί της είναι το χαρακτηριστικό deptno. Η οντότητα Job έχει χαρακτηριστικά jobno, job, sal και το κλειδί της είναι το χαρακτηριστικό jobno. Επιπλέον, η συσχέτιση has συνδέει τις οντότητες jobs, employee και είναι τύπου 1:N. Δηλαδή, μία θέση «έχει» πολλούς υπαλλήλους («πολλοί υπάλληλοι έχουν το ίδιο πόστο» ή κάνουν την ίδια εργασία). Επιπλέον η συσχέτιση has η οποία συνδέει τις οντότητες dept, employee σημαίνει ότι ένα τμήμα έχει πολλούς υπαλλήλους (ή πολλοί υπάλληλοι ανήκουν στο ίδιο τμήμα). Παρατηρήστε ότι οι οντότητες του μοντέλου μεταγράφονται σε πίνακες EMPLOYEE, JOBS, DEPT. Η οντότητα του υπαλλήλου μεταγράφεται στον πίνακα EMPLOYEE και επιπλέον στον πίνακα αναγράφονται και οι στήλες jobno, deptno επειδή στο μοντέλο έχουμε τις συσχετίσεις τύπου 1:N.

Ένα άλλο στοιχείο στο μοντέλο είναι η ασθενής-εξαρτώμενη οντότητα (weak entity) CHILD. Είναι η οντότητα η οποία εξαρτάται από τη γονική οντότητα EMPLOYEE. Δείτε ότι η οντότητα CHILD μεταγράφεται σε πίνακα με σύνθετο κύριο κλειδί (δείτε Εικόνα 1.17 και Εικόνα 2.18).



Εικόνα 1.18 Το ΜΟΣ του Συλλόγου Γονέων και Κηδεμόνων. Η διπλή γραμμή σημαίνει ότι κάθε παιδί (CHILD) στη βάση δεδομένων έχει γονέα υπάλληλο (EMPLOYEE). Η απλή γραμμή σημαίνει ότι δεν έχει κατ' ανάγκη παιδί κάθε υπάλληλος.

Περισσότερα μπορείτε να μελετήσετε στα κεφάλαια 5, 6.

1.7.3 Σενάριο (use case) βάσης δεδομένων προσωπικού της εταιρείας Strategy for Data Modelling

Στη γενική περίπτωση, ένας υπάλληλος της εταιρείας μπορεί να έχει ή να μην έχει παιδιά (Εικόνα 1.19). Δείτε με τη βοήθεια του δείγματος δεδομένων και τις αλλαγές από τον προηγούμενο πίνακα Employee της εικόνας 1.13. Ποιο είναι το κύριο κλειδί; Προσοχή! Κύριο κλειδί δεν θα μπορούσε να είναι ο συνδυασμός στηλών (empno, c_name). Θυμηθείτε τους κανόνες ακεραιότητας για να αιτιολογήσετε την απάντηση.

Περιορισμός: Κάθε υπάλληλος μπορεί να έχει ή να μην έχει παιδιά.

Πίνακας Employee

Empno	Name	JobNo	Job	DeptNo	Dname	Sal	C_no	C_Name	B_date
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2	ΜΑΡΙΑ	10/01/2009
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2	ΙΩΑΝΝΗΣ	20/03/2010
20	ΧΡΗΣΤΟΥ	200	ΑΝΑΛΥΤΗΣ	60	ΛΟΓΙΣΤΗΡΙΟ	2000			
30	ΝΙΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΜΙΣΘΟΔΟΣΙΑ	1000	1	ΘΩΜΑΣ	10/06/2009

Ο συνδυασμός στηλών (empno, c_name) δεν είναι κύριο κλειδί

Εικόνα 1.19 Όλα τα στοιχεία της βάσης δεδομένων σε έναν πίνακα

Αν θέλουμε να σχεδιάσουμε την 1NF έτσι ώστε να έχει έναν μόνο πίνακα τότε μπορούμε να προσθέσουμε μία στήλη Accno (είναι ένας αύξων αριθμός) που θα είναι και κύριο κλειδί του πίνακα.

Περαιτέρω διερεύνηση

Έστω ότι η στήλη Ch_No συμβολίζει το μοναδικό κωδικό κάθε παιδιού. Δηλαδή, όλα τα παιδιά στη βάση δεδομένων έχουν μοναδικό κωδικό. Είναι η στήλη (ch_no) το κύριο κλειδί του πίνακα της εικόνας 1.20; Όχι δεν είναι γιατί δεν έχει τιμή (ή έχει τιμή NULL) για τον υπάλληλο 20. Ποιο είναι το κύριο κλειδί; Δείτε την παραπάνω απάντηση.

η στήλη Ch_No συμβολίζει το μοναδικό κωδικό κάθε παιδιού
η στήλη C_No συμβολίζει τον αριθμό παιδιών

Employee

Empno	Name	JobNo	Job	DeptNo	Dname	Sal	C_no	Ch_no	C_Name	B_date
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2	1	ΜΑΡΙΑ	10/01/2009
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2	2	ΙΩΑΝΝΗΣ	20/03/2010
20	ΧΡΗΣΤΟΥ	200	ΑΝΑΛΥΤΗΣ	60	ΛΟΓΙΣΤΗΡΙΟ	2000				
30	ΝΙΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΜΙΣΘΟΔΟΣΙΑ	1000	1	3	ΘΩΜΑΣ	10/06/2009

η στήλη (ch_no) δεν είναι το κύριο κλειδί του πίνακα

Εικόνα 1.20 Όλα τα στοιχεία της βάσης δεδομένων σε έναν πίνακα. Η στήλη Ch_No συμβολίζει το μοναδικό κωδικό κάθε παιδιού

Αν θεωρήσουμε άλλη εναλλακτική σχεδίαση για την πρώτη κανονική μορφή θα έχουμε διαφορετική προσέγγιση. Για παράδειγμα, αν στην πρώτη κανονική μορφή έχουμε τους πίνακες emp, child τότε η πρώτη και η δεύτερη κανονική μορφή συμπίπτουν.

Ακολουθεί η Τρίτη Κανονική Μορφή 3NF που είναι τελείως παρόμοια με αυτήν που κατασκευάσαμε στη βάση γονέων και κηδεμόνων (Εικόνα 1.21).

Τρίτη Κανονική Μορφή της βάσης δεδομένων της εταιρείας Strategy for Data Modelling

Ένας υπάλληλος μπορεί να έχει ή να μην έχει παιδιά.

Employee

Empno	Name	JobNo	DeptNo	C_no
Κωδικός	Όνομα	Κωδικός θέσης	Κωδικός τμήματος	Αριθμός παιδιών
10	ΣΠΥΡΟΥ	100	50	2
20	ΧΡΗΣΤΟΥ	200	60	
30	ΝΙΚΟΥ	300	70	1

Κύριο κλειδί: empno

Jobs

JobNo	Job	Sal
Κωδικός	Θέση	μισθός
100	ΠΩΛΗΤΗΣ	2200
200	ΑΝΑΛΥΤΗΣ	2000
300	ΧΕΙΡΙΣΤΗΣ	1000

Κύριο κλειδί: JobNo

Dept

DeptNo	Dname
Κωδικός	Τμήμα
50	ΠΩΛΗΣΕΙΣ
60	ΛΟΓΙΣΤΗΡΙΟ
70	ΜΙΣΘΟΔΟΣΙΑ

Κύριο κλειδί: deptno

Child

Empno	Ch_no	C_Name	B_date
Κωδικός υπαλλήλου	Κωδικός παιδιού	Όνομα παιδιού	Ημερομηνία γέννησης
10	1	ΜΑΡΙΑ	10/01/2009
10	2	ΙΩΑΝΝΗΣ	20/03/2010
30	3	ΘΩΜΑΣ	10/06/2009

Κύριο κλειδί: (ch_no), foreign key: (empno)

Names

C_Name
ΘΩΜΑΣ
ΙΩΑΝΝΗΣ
ΜΑΡΙΑ

Κύριο κλειδί: (c_name)

Εικόνα 1.21 Τρίτη Κανονική Μορφή της βάσης δεδομένων της εταιρείας Strategy for Data Modelling Ένας υπάλληλος μπορεί να έχει ή να μην έχει παιδιά.

1.7.4 Σενάριο (use case) βάσης δεδομένων προσωπικού και έργων εταιρείας

Η ανάλυση δεδομένων (data analysis) οδηγεί στις εξής στήλες:

- Empno κωδικός υπαλλήλου,
- Fname όνομα,
- Surname επώνυμο,

- JobNo κωδικός της θέσης του υπαλλήλου,
- Jobname θέση υπαλλήλου, π.χ., ΠΩΛΗΤΗΣ,
- DeptNo κωδικός τμήματος υπαλλήλου,
- Dname τμήμα υπαλλήλου,
- Dloc έδρα τμήματος υπαλλήλου,
- Sal μισθός υπαλλήλου,
- Comm Προμήθεια υπαλλήλου,
- Projno κωδικός έργου στο οποίο εργάζεται ο υπάλληλος,
- Pdescr έργο,
- Ploc έδρα έργου,
- Ptime ποσοστό χρόνου που ο υπάλληλος απασχολείται σε ένα έργο

Περιορισμοί

Ένας υπάλληλος έχει μία θέση (jobno), εργάζεται σε ένα τμήμα (deptno), μπορεί να έχει προμήθεια (comm) ή να μην έχει, και απασχολείται σε ένα ή περισσότερα έργα (projno) για κάποιο ποσοστό του χρόνου του (ptime). Ένα τμήμα έχει μοναδικό κωδικό (deptno), μία επωνυμία, μία έδρα και μπορεί να έχει πολλούς υπαλλήλους. Σε κάποια θέση π.χ., ΠΩΛΗΤΗΣ μπορεί να απασχολούνται πολλοί υπάλληλοι. Σε κάθε έργο απασχολούνται πολλοί υπάλληλοι. Ο μισθός (sal) εξαρτάται από τη θέση.

Βήμα 1 :

Γράψτε όλα τα στοιχεία υπαλλήλου σε έναν πίνακα (Εικόνα 1.22)

Θυμηθείτε ότι μπορούμε να έχουμε περισσότερους πίνακες στην Πρώτη Κανονική Μορφή. Για παράδειγμα οι παρακάτω τρεις πίνακες είναι στην Πρώτη Κανονική Μορφή.

Empno	Fname	Surname	JobNo	DeptNo	Comm	Projno	Pdescr	Ploc	ptime

JobNo	Jobname	Sal

DeptNo	Dname	dloc

Βήμα 1 : όλα σε έναν πίνακα

Employee (πίνακας στοιχείων υπαλλήλων)

Empno	Fname	Surname	JobNo	Jobname	DeptNo	Dname	dloc	Sal	Comm	Projno	Pdescr	ploc	ptime
10	ΣΠΥΡΟΣ	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	ΒΟΛΟΣ	2200	250	1000	PAYROLL	ΑΘΗΝΑ	60
10	ΣΠΥΡΟΣ	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	ΒΟΛΟΣ	2200	250	2000	MEDIA	ΒΟΛΟΣ	40
20	ΝΙΚΟΣ	ΝΙΚΟΥ	200	ΑΝΑΛΥΤΗΣ	60	ΠΡΟΣΩΠΙΚΟ	ΑΘΗΝΑ	2000	100	1000	PAYROLL	ΑΘΗΝΑ	100
30	ΝΙΚΟΣ	ΜΑΡΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΛΟΓΙΣΤΗΡΙΟ	ΑΘΗΝΑ	1000		2000	MEDIA	ΒΟΛΟΣ	100
40	ΜΑΡΚΟΣ	ΜΑΡΚΟΥ	100	ΠΩΛΗΤΗΣ	80	MARKETING	ΑΘΗΝΑ	2200	500	2000	MEDIA	ΒΟΛΟΣ	100

Πρώτη Κανονική μορφή

Σύνθετο Κύριο κλειδί=(empno, projno)

Εικόνα 1.22 Πρώτη κανονική μορφή. Όλα τα δεδομένα στον ίδιο πίνακα.

Βήμα 2 (Εικόνα 1.23):

Εξετάζουμε ποιες στήλες εκτός κλειδιού «καθορίζει» από μόνο του κάθε τμήμα του κύριου κλειδιού της πρώτης κανονικής μορφής. «Χωρίζουμε» ανάλογα σε πίνακες τον πίνακα της πρώτης κανονικής μορφής. Παραθέτουμε τα τμήματα του κύριου κλειδιού της πρώτης κανονικής μορφής.

Empno -- >

Projno -- >

empno,

projno -- >

Δεύτερη κανονική μορφή

Employ

Empno	Fname	Surname	JobNo	Jobname	DeptNo	Dname	dloc	Sal	comm
10	ΣΠΥΡΟΣ	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	ΒΟΛΟΣ	2200	250
20	ΝΙΚΟΣ	ΝΙΚΟΥ	200	ΑΝΑΛΥΤΗΣ	60	ΠΡΟΣΩΠΙΚΟ	ΑΘΗΝΑ	2000	100
30	ΝΙΚΟΣ	ΜΑΡΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΛΟΓΙΣΤΗΡΙΟ	ΑΘΗΝΑ	1000	
40	ΜΑΡΚΟΣ	ΜΑΡΚΟΥ	100	ΠΩΛΗΤΗΣ	80	MARKETING	ΑΘΗΝΑ	2200	500

Κύριο κλειδί=(empno)

Proj

Projno	Pdescr	ploc
1000	PAYROLL	ΑΘΗΝΑ
2000	MEDIA	ΒΟΛΟΣ

Κύριο κλειδί=(projno)

Works

Empno	Projno	ptime
10	1000	60
10	2000	40
20	1000	100
30	2000	100
40	2000	100

Κύριο κλειδί=(empno, projno)

Εικόνα 1.23 Δεύτερη κανονική μορφή

Βήμα 3:

Σε κάθε πίνακα εξετάζουμε μήπως κάποια στήλη ή κάποιος συνδυασμός στηλών του που δεν ανήκει στο κύριο κλειδί του πίνακα μπορεί να «καθορίζει» από μόνο του άλλες στήλες του πίνακα.

«Χωρίζουμε» ανάλογα τον πίνακα αυτόν. Επομένως, ο πίνακας emp χωρίζεται σε τρεις πίνακες.

Οι υπόλοιποι πίνακες της δεύτερης κανονικής μορφής παραμένουν όπως είναι (Εικόνα 1.24).

Τρίτη κανονική μορφή

Emp

Empno	Fname	Surname	JobNo	DeptNo	Comm
10	ΣΠΥΡΟΣ	ΣΠΥΡΟΥ	100	50	250
20	ΝΙΚΟΣ	ΝΙΚΟΥ	200	60	100
30	ΝΙΚΟΣ	ΜΑΡΚΟΥ	300	70	
40	ΜΑΡΚΟΣ	ΜΑΡΚΟΥ	100	80	500

Κύριο κλειδί=(empno)

Job

JobNo	Jobname	Sal
100	ΠΩΛΗΤΗΣ	2200
200	ΑΝΑΛΥΤΗΣ	2000
300	ΧΕΙΡΙΣΤΗΣ	1000

Κύριο κλειδί=(jobno)

Dept

DeptNo	Dname	dloc
50	ΠΩΛΗΣΕΙΣ	ΒΟΛΟΣ
60	ΠΡΟΣΩΠΙΚΟ	ΑΘΗΝΑ
70	ΛΟΓΙΣΤΗΡΙΟ	ΑΘΗΝΑ
50	MARKETING	ΑΘΗΝΑ

Κύριο κλειδί=(deptno)

Proj

Projno	Pdescr	ploc
1000	PAYROLL	ΑΘΗΝΑ
2000	MEDIA	ΒΟΛΟΣ

Κύριο κλειδί=(projno)

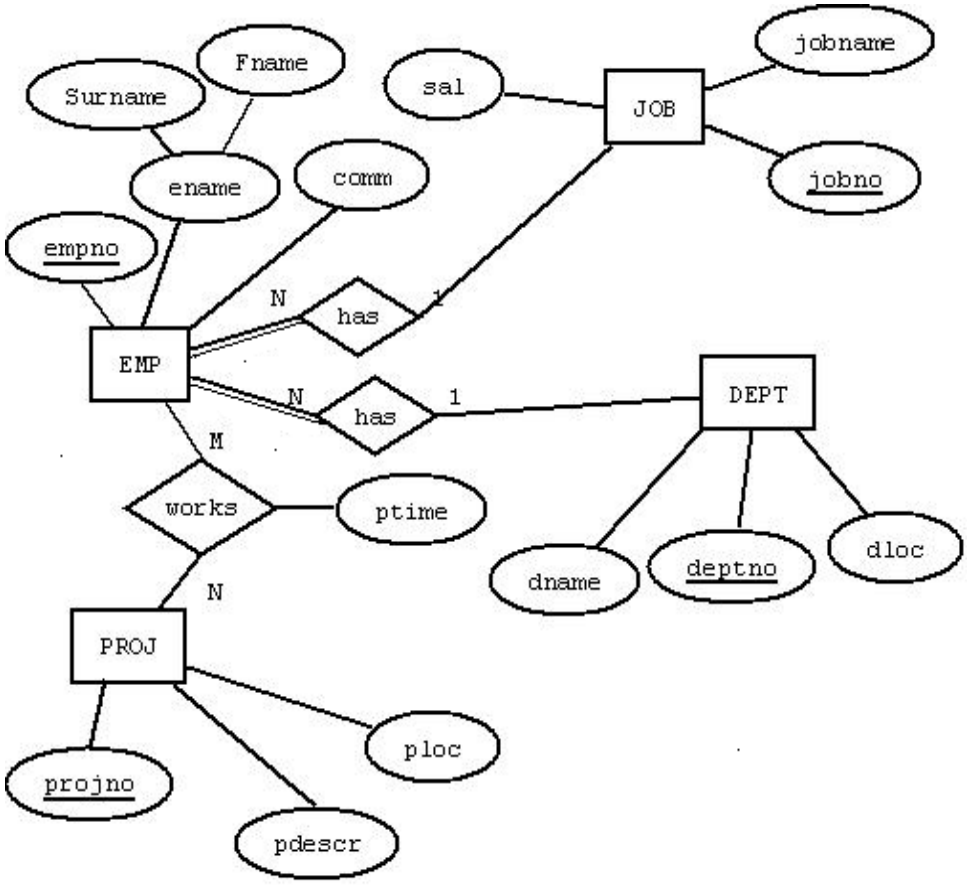
Works

Empno	Projno	Ptime
10	1000	60
10	2000	40
20	1000	100
30	2000	100
40	2000	100

Κύριο κλειδί=(empno, projno)

Εικόνα 1.24 Τρίτη κανονική μορφή

Ακολουθεί το Μοντέλο Οντοτήτων Συσχετίσεων (Εικόνα 1.25).



Εικόνα 1.25 Μοντέλο Οντοτήτων συσχετίσεων

1.8 Μοντελοποίηση εταιρίας. Μοντέλο Οντοτήτων – Συσχετίσεων και MySQL Workbench

Στο παράδειγμα θεωρούμε ότι ο μισθός εξαρτάται από προσωπική συμφωνία. Επομένως, η βάση δεδομένων προσωπικού που θα υλοποιήσουμε φαίνεται στην εικόνα 1.26!

assign (πίνακας που αποθηκεύει ποιοί υπάλληλοι απασχολούνται σε ποιά έργα)

Empno	Proj_code	A_Time
10	100	40
10	200	60
15	100	100
20	200	100
30	100	100

project (πίνακας έργων)

Proj_code	Description
100	PAYROLL
200	PERSONNEL

300	SALES

emp (πίνακας υπαλλήλων)

Empno	Ename	Job	Hiredate	Mgr	Sal	Comm	Deptno
10	Codd	ANALYST	1/1/89	15	3000		10
15	Elmasri	ANALYST	2/5/95	15	1200	150	10
20	Navathe	SALESMAN	7/7/77	20	2000		20
30	Date	PROGRAMMER	4/5/04	15	1800	200	10

dept (πίνακας τμημάτων)

Deptno	Dname	Loc
10	ACCOUNTING	ATHENS
20	SALES	LONDON
30	RESEARCH	ATHENS
40	PAYROLL	LONDON

Εικόνα 1.26 Τρίτη κανονική μορφή

Δημιουργία βάσης δεδομένων στο προϊόν MySQL

```
CREATE DATABASE my_db;
```

Χρήση βάσης

```
USE my_db;
```

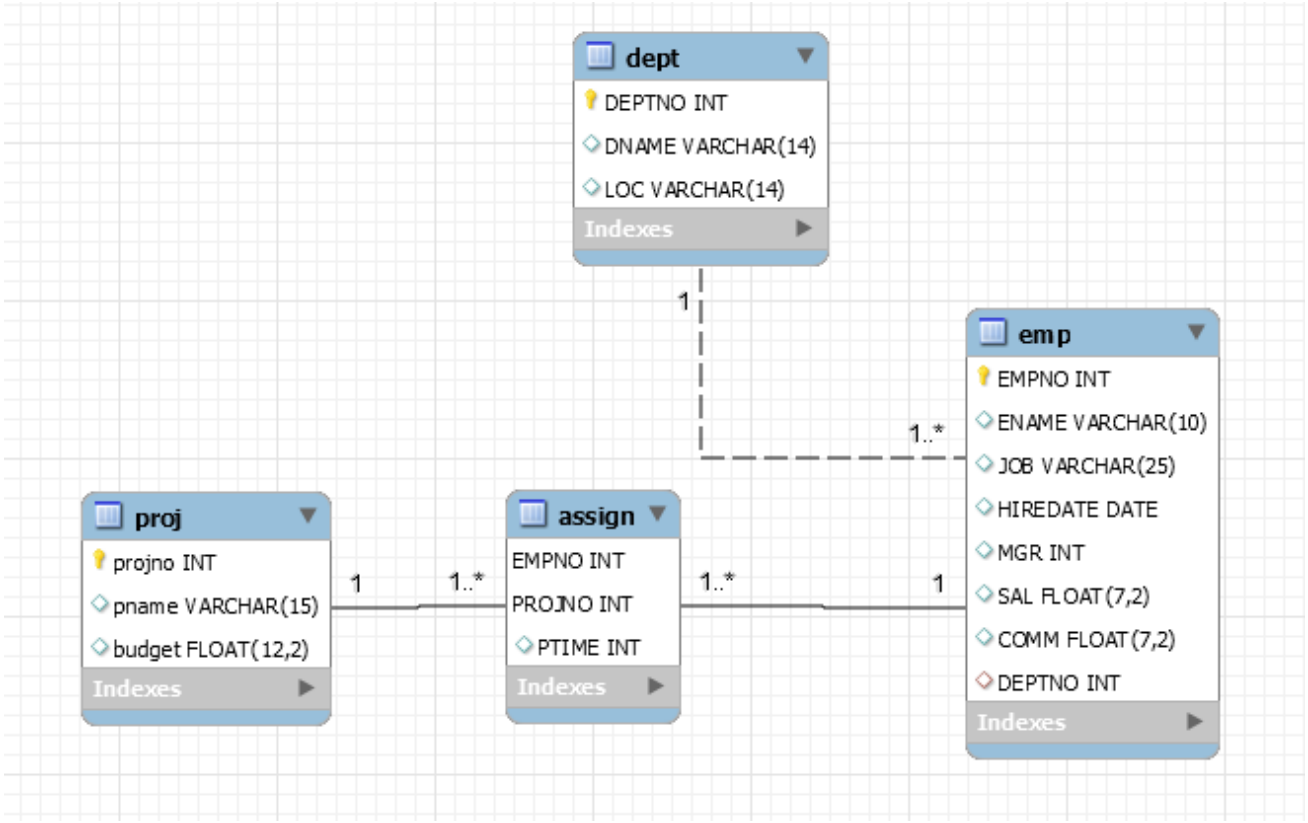
Δημιουργία πινάκων

```
CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL,
 DNAME VARCHAR(14), LOC VARCHAR(14),
 PRIMARY KEY(DEPTNO));

CREATE TABLE EMP(EMPNO INT(4) NOT NULL,
  ENAME VARCHAR(10), JOB VARCHAR(25),
  HIREDATE DATE, MGR INT(4),  SAL FLOAT(7,2), COMM FLOAT(7,2),
  DEPTNO INT(2),
  PRIMARY KEY(EMPNO),
  FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO));

CREATE TABLE PROJ (projno INT(3) NOT NULL,
   pname VARCHAR(15),
   budget FLOAT(12,2),
 PRIMARY KEY(projno));

CREATE TABLE ASSIGN(
 EMPNO INT(4) NOT NULL, PROJNO INT(3) NOT NULL,
 PTIME INT(3),
PRIMARY KEY(EMPNO,PROJNO),
FOREIGN KEY(EMPNO) REFERENCES EMP(EMPNO),
FOREIGN KEY(PROJNO) REFERENCES PROJ(PROJNO));
```

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (40, 'OPERATIONS', 'BOSTON');

INSERT INTO EMP
  VALUES (10, 'CODD', 'ANALYST', '1989/01/01', 15, 3000, NULL, 10);
INSERT INTO EMP
  VALUES (15, 'ELMASRI', 'ANALYST', '1995/05/02', 15, 1200, 150, 10);
INSERT INTO EMP
  VALUES (20, 'NAVATHE', 'SALESMAN', '1977/07/07', 20, 2000, NULL, 20);
INSERT INTO EMP
  VALUES (30, 'DATE', 'PROGRAMMER', '2004/05/04', 15, 1800, 200, 10);

INSERT INTO proj(projno, pname, budget)
VALUES(100, 'PAYROLL', 100000);
INSERT INTO proj(projno, pname, budget)
VALUES(200, 'PERSONNEL',200000 );
INSERT INTO proj(projno, pname, budget)
VALUES(300, 'SALES', 150000);

INSERT INTO assign(empno, projno, ptime)
 VALUES(10,100, 40);
INSERT INTO assign(empno, projno, ptime)
 VALUES(10, 200, 60);
INSERT INTO assign(empno, projno, ptime)
 VALUES(15, 100, 100);
INSERT INTO assign(empno, projno, ptime)
 VALUES(20, 200, 100);
INSERT INTO assign(empno, projno, ptime)
 VALUES(30, 100, 100);
```

Ακολουθούν παραδείγματα μοντέλου στο προϊόν MySQL Workbench.

Στην Εικόνα 1.27 χρησιμοποιείται ο συμβολισμός του μοντέλου UML.



Εικόνα 1.27 Μοντέλο UML

Για τις συσχετίσεις (relationship) 1: 1 και 1: n, έχουμε δύο διαφορετικούς τύπους συμβόλων:

- η καθορίζουσα (identifying) συσχέτιση συμβολίζεται με βέλος με συμπαγή γραμμή (→)
- η μη καθορίζουσα (non identifying) συσχέτιση συμβολίζεται με βέλος με διακεκομμένη γραμμή (--->).

Ακολουθεί παράδειγμα για μη καθορίζουσα συσχέτιση.

Teacher -------> Classes συσχέτιση τύπου 1:N

Παρατηρήστε ότι ένας καθηγητής μπορεί να μην έχει τοποθετηθεί σε τάξη και σε κάποια τάξη μπορεί να εκκρεμεί η τοποθέτηση καθηγητή.

Ένα άλλο παράδειγμα είναι η συσχέτιση τμημάτων και υπαλλήλων: DEPT ---> Emp.

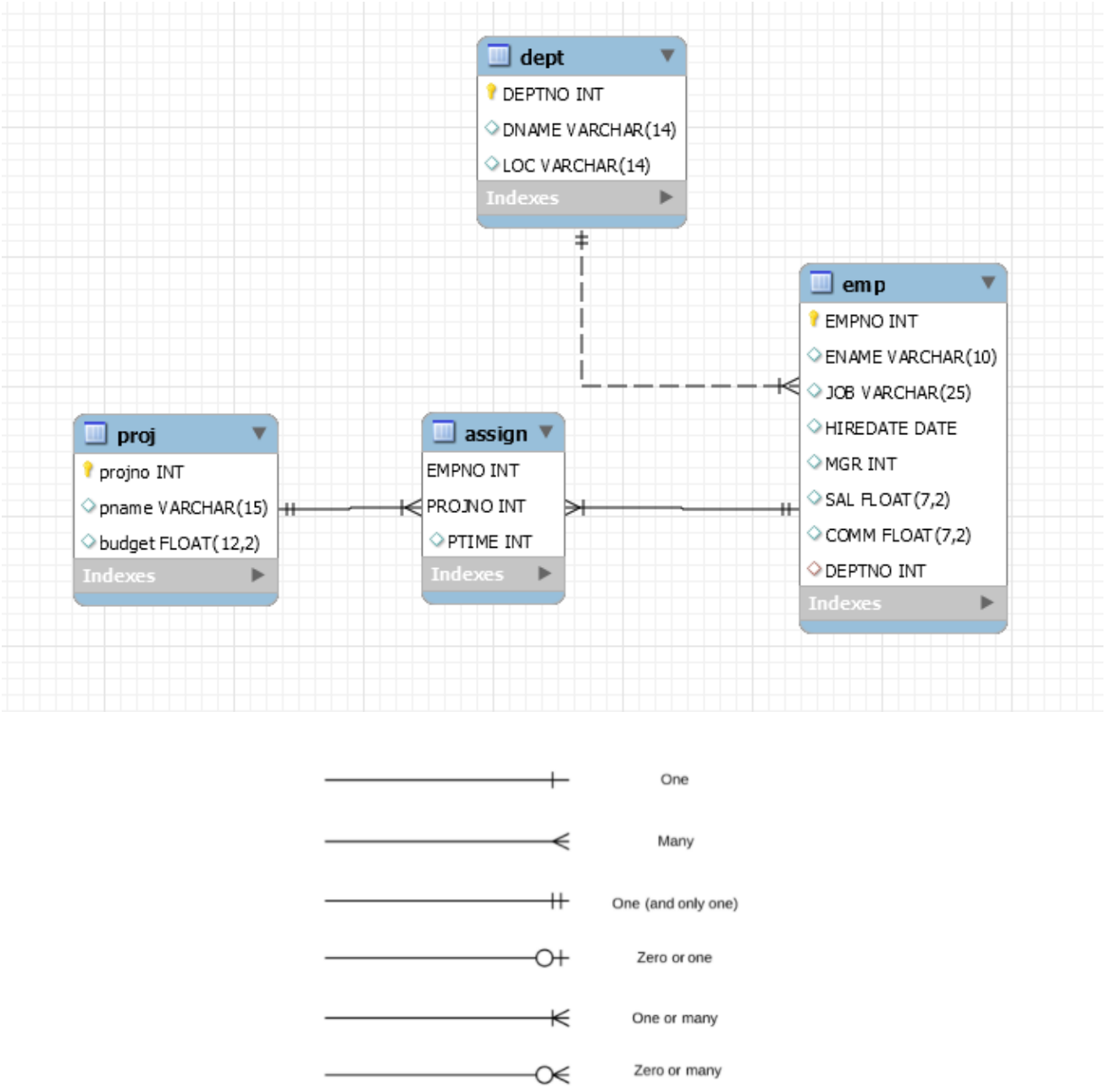
Μια συσχέτιση μεταξύ πινάκων θεωρείται καθορίζουσα όταν ο ένας πίνακας εξαρτάται πλήρως από τον άλλο για να υπάρχει. Μια γραμμή σε αυτόν τον (εξαρτημένο) πίνακα εξαρτάται από μια γραμμή στον άλλο πίνακα. Ένα κοινό παράδειγμα είναι να υπάρχει ένας ξεχωριστός πίνακας για την αποθήκευση τηλεφώνων για τους χρήστες επειδή μπορεί να υπάρχουν πολλά τηλέφωνα για έναν χρήστη. Κάθε γραμμή στον πίνακα με τα τηλέφωνα εξαρτάται εξ ολοκλήρου από τον χρήστη («ανήκει στον χρήστη»).

Identifying → (μη διακεκομμένη γραμμή) Example: Employee → Phones 1:N

Στην περίπτωση αυτή των συσχετίσεων 1: 1 και 1: n, το ξένο κλειδί που εισάγεται θα είναι μέρος του κύριου κλειδιού για τον «καθοριζόμενο» πίνακα σχηματίζοντας ένα σύνθετο πρωτεύον κλειδί, π.χ. ο πίνακας τηλεφώνων έχει κλειδί (empno, phone).

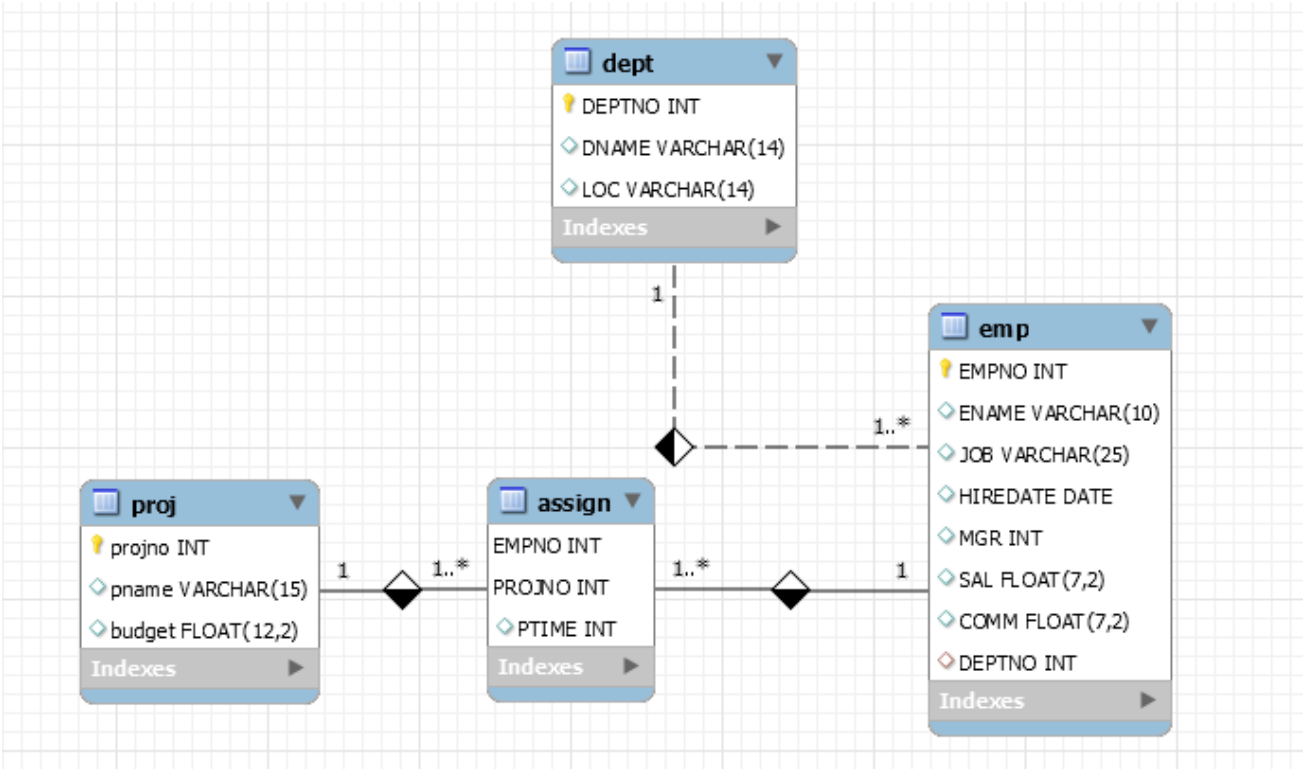
Οι καθορίζουσες συσχετίσεις χρησιμοποιούνται για τους πίνακες σύνδεσης (join tables) που δημιουργούνται από μια συσχέτιση πολλά προς πολλά. Αυτοί οι ενδιάμεσοι πίνακες εξαρτώνται πλήρως από τους δύο αρχικούς πίνακες.

Στην Εικόνα 1.28 χρησιμοποιείται ο συμβολισμός του μοντέλου (Martin) Crow's foot.



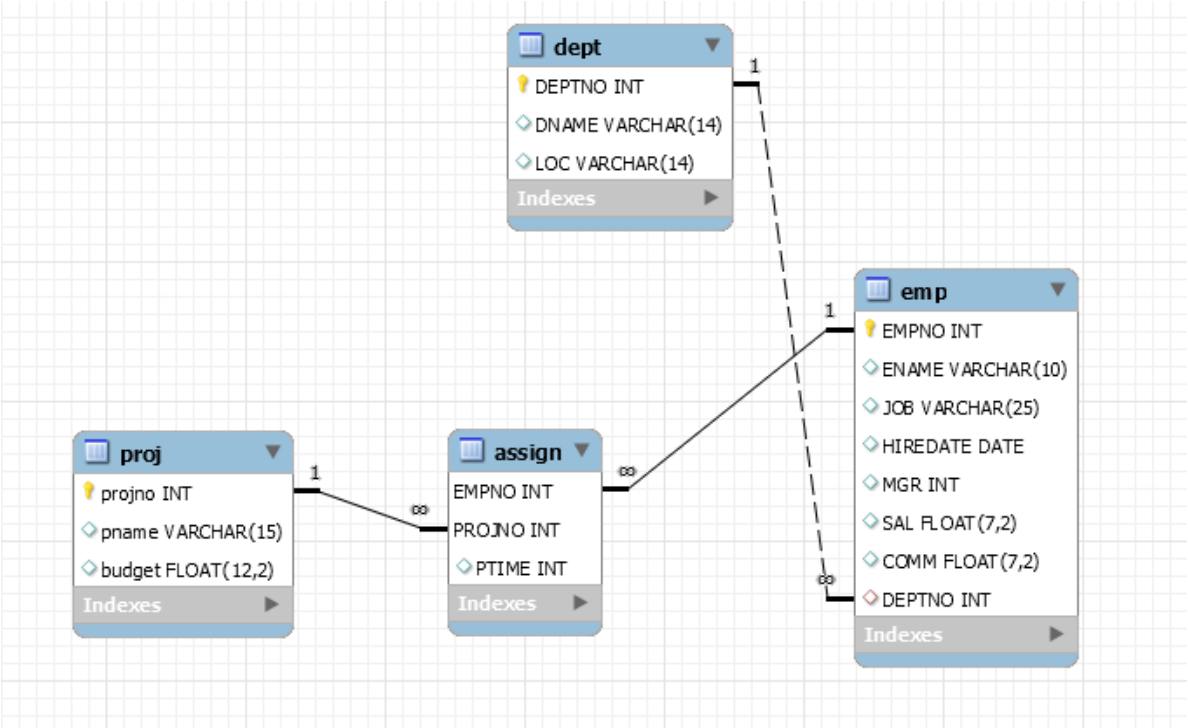
Εικόνα 1.28 Μοντέλο (Martin) Crow's foot.

Στην Εικόνα 1.29 χρησιμοποιείται ο συμβολισμός του μοντέλου Classic.



Εικόνα 1.29 Μοντέλο Classic

Στην Εικόνα 1.30 χρησιμοποιείται ο συμβολισμός του μοντέλου Connect to columns



Εικόνα 1.30 Μοντέλο Connect to columns

1.8.1 Βιβλιογραφία και αναφορές

Chen P.P.S. (1976), The entity-relationship model—toward a unified view of data, ACM Transactions on Database Systems (TODS), 1(1), 9-36, dl.acm.org (**The entity-relationship model—toward a unified view of data**)++

Chen, P.P.S. (1983), English sentence structure and entity-relationship diagrams, Information Sciences, 29(2-3), 127-149, Elsevier (**English sentence structure and entity-relationship diagrams**)

Teorey, T.J., Yang, D., & Fry, J.P. (1986), A logical design methodology for relational databases using the extended entity-relationship model, ACM Computing Surveys (CSUR), 18(2), 197-222, dl.acm.org

(**A logical design methodology for relational databases using the extended entity-relationship model**)

Alhajj R. (2003), Extracting the extended entity-relationship model from a legacy relational database, Information Systems, 28(6), 597-618.

Ordonez, C., Maabout, S., Matusevich, D. S., & Cabrera, W. (2014). Extending ER models to capture database transformations to build data sets for data mining. Data & Knowledge Engineering, 89, 38-54.

Lovison, A., Rigoni, E. (2010), Extracting optimal datasets for meta modelling and perspectives for incremental samplings, Mathematics and Computers in Simulation, 81(3), 681-692

Cleve, A., Gobert, M., Meurice, L., Maes, J., & Weber, J. (2015). Understanding database schema evolution: A case study. Science of Computer Programming, 97, 113-121.

Tseng, F. S., & Chen, C. L. (2008). Enriching the class diagram concepts to capture natural language semantics for database access. Data & Knowledge Engineering, 67(1), 1-29.

Belkadi, F., Notin, A., Drémont, N., & Troussier, N. (2012). A meta-model for knowledge representation in engineering design. IFAC Proceedings Volumes, 45(6), 1641-1646.

Morawski, R. Z. (2013). An application-oriented mathematical meta-model of measurement. Measurement, 46(9), 3753-3765.

1.8.2 Βιβλία βάσεων δεδομένων στα ελληνικά που χρησιμοποιήθηκαν στο παρόν σύγγραμμα

1. Elmasri Ramez, Navathe Shamkant B., Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων, Δίαυλος
2. Jeffrey D. Ullman, Jennifer Widom, Βασικές Αρχές για τα Συστήματα Βάσεων Δεδομένων, Κλειδάριθμος
3. Ramakrishnan Raghu, Gehrke Joahannes, Συστήματα Διαχείρισης Βάσεων Δεδομένων, Εκδόσεις Τζιόλα
4. Chris J. Date, Εισαγωγή στα συστήματα βάσεων δεδομένων (πρώτος τόμος), Κλειδάριθμος
5. Korth Henry, Silberschatz Abraham, Sudarshan S, Συστήματα Βάσεων Δεδομένων. Η Πλήρης Θεωρία των Βάσεων Δεδομένων, Γκιούρδας
6. Χρήστος Σκουρλάς Σχεσιακές Βάσεις Δεδομένων, Εκδόσεις Νέων Τεχνολογιών
7. Χρήστος Σκουρλάς, Υλοποίηση Εφαρμογών με Γλώσσα SQL, Εκδόσεις Νέων Τεχνολογιών

1.9 Σύστημα διαχείρισης βάσης δεδομένων. Διαχειριστής συστήματος. Συναλλαγές (transaction) και ταυτοχρονισμός (concurrency). Ρόλος και καθήκοντα του ΔΒΔ. Ανάκαμψη (recovery).

Ορισμός:

- Ένα σύστημα διαχείρισης βάσεων δεδομένων-ΣΔΒΔ (Data Base Management System-DBMS) είναι ένα λογισμικό που επιτρέπει στο χρήστη (επιχείρηση) την υλοποίηση και τη συντήρηση βάσεως δεδομένων. Με άλλα λόγια, το ΣΔΒΔ επιτρέπει τον ορισμό δεδομένων και τη διαχείριση δεδομένων.
- Το ΣΔΒΔ αναλαμβάνει τη διαχείριση των δεδομένων (data) όλων των εφαρμογών και συνεργαζόμενο με το λειτουργικό σύστημα τα αποθηκεύει χωρίς άχρηστους πλεονασμούς (redundancy).
- Η μεσολάβηση του ΣΔΒΔ εξασφαλίζει ότι τα δεδομένα είναι ανεξάρτητα των προγραμμάτων που τα διαχειρίζονται.
- Η εισαγωγή νέων δεδομένων, η τροποποίηση και η ανάκτηση δεδομένων από τη Βάση Δεδομένων γίνεται πάντα με τη μεσολάβηση του ΣΔΒΔ για όλες τις εφαρμογές.
- Οι Elmasri-Navathe υπογραμμίζουν ότι το ΣΔΒΔ υποστηρίζει ταυτόχρονη προσπέλαση (concurrency control) των χρηστών στα δεδομένα και εφαρμογές άμεσης επεξεργασίας συναλλαγών (δοσοληψιών) (OLTP-Online Transaction Processing). Δηλαδή οι εφαρμογές των χρηστών πρέπει στο ΣΒΔ να μπορούν να βλέπουν τα ίδια δεδομένα και να τα διαχειρίζονται κατά ελεγχόμενο τρόπο που να εξασφαλίζει την ορθότητά τους. Για παράδειγμα δύο ταξιδιωτικοί πράκτορες πρέπει να μπορούν να βλέπουν ταυτόχρονα τις ελεύθερες θέσεις μιας πτήσης αλλά φυσικά δε θα μπορούν να κάνουν κράτηση για την ίδια θέση

1.9.1 Εννοιολογική προσέγγιση στον τρόπο εργασίας του συστήματος διαχείρισης βάσης δεδομένων (Database Management System - DBMS)

Το DBMS είναι το λογισμικό που διαχειρίζεται κάθε πρόσβαση (access) στη βάση δεδομένων και είναι υπεύθυνο για την εφαρμογή των ελέγχων εξουσιοδοτημένης πρόσβασης (authorisation checks) και των διαδικασιών ορθότητας (validation procedures).

Εννοιολογικά:

1) Ο χρήστης αιτείται πρόσβαση (access request) χρησιμοποιώντας Γλώσσα Διαχείρισης Δεδομένων (DML)
2) Το DBMS δέχεται την αίτηση και τη διερμηνεύει.
3) Το DBMS ελέγχει το εξωτερικό σχήμα, την αντιστοιχία με το εννοιολογικό σχήμα (the external/conceptual mapping) και
4) Το DBMS ελέγχει την αντιστοιχία εννοιολογικού / εσωτερικού σχήματος (the conceptual internal mapping) και τον ορισμό της δομής αποθήκευσης (storage structure).
5) Τέλος, το DBMS εκτελεί τις αναγκαίες εργασίες στην αποθηκευμένη (σε αρχεία) βάση δεδομένων.

1.9.2 Συναλλαγές (transactions) και ΣΔΒΔ

Τα συστήματα βάσεων δεδομένων υποστηρίζουν συναλλαγές (transaction), δηλαδή σύνολα ενεργειών (SQL δηλώσεων) στη βάση δεδομένων που εκτελούνται όλες μαζί. Παράδειγμα συναλλαγής είναι η μεταφορά χρημάτων ανάμεσα σε τραπεζικούς λογαριασμούς. Ένα ποσό αφαιρείται από ένα λογαριασμό και πιστώνεται

σε έναν άλλο. Εκτελούνται δύο δηλώσεις UPDATE. Αν αποτύχει μια δήλωση UPDATE για οποιοδήποτε λόγο τότε η συναλλαγή αναιρείται στο σύνολό της.

Η υπογλώσσα ελέγχου της γλώσσας SQL (Data Control Language, DCL) υποστηρίζει τις δηλώσεις COMMIT, ROLLBACK. Η δήλωση COMMIT συνεπάγεται την επιτυχή ολοκλήρωση της συναλλαγής και η δήλωση ROLLBACK την αναίρεση των ενεργειών που έγιναν στη βάση στην περίπτωση που παρουσιάστηκε κάποιο πρόβλημα. Δηλαδή, αν μία συναλλαγή ολοκληρώνεται επιτυχώς τότε όλες οι ενέργειες στη βάση δεδομένων επικυρώνονται, δηλαδή οριστικοποιούνται (commit). Αν κάποιο βήμα της συναλλαγής αποτύχει, π.χ., αποτυγχάνει μία δήλωση UPDATE, τότε όλες οι ενέργειες που περιλαμβάνονται σε αυτήν ακυρώνονται (rollback). Η αναίρεση των ενεργειών επιτυγχάνεται επειδή όλες αυτές οι μεταβολές της βάσης που ζήτησε να γίνουν μία συναλλαγή στην πραγματικότητα και σε ένα πρώτο στάδιο αποθηκεύονται όχι στη βάση αλλά σε ειδικά αρχεία του συστήματος (logfile, journal) μαζί με όλες τις απαραίτητες πληροφορίες. Αργότερα και αν όλα πάνε καλά οριστικοποιούνται οι μεταβολές στη βάση. Προσοχή! Το ΣΔΒΔ κάθε στιγμή δείχνει στους χρήστες των εφαρμογών όλες τις μεταβολές στοιχείων που έχουν γίνει από τη συναλλαγή τους. Επομένως, πρέπει στις εφαρμογές να χρησιμοποιηθεί το σωστό επίπεδο απομόνωσης (isolation level). Διαφορετικά κατά την ταυτόχρονη εκτέλεση των συναλλαγών ο χρήστης μπορεί να βλέπει, ενδιαμέσως, δεδομένα τα οποία δεν έχουν οριστικοποιηθεί στην πραγματικότητα (δείτε και ενότητες 2.8.2 και 2.8.3 στο δεύτερο κεφάλαιο).

1.9.3 Παράδειγμα συναλλαγής σε SQL και ψευδοκώδικα

```
START TRANSACTION
INSERT INTO shipment (sno,pno,qty) VALUES ('S100','P1000',250);
IF error THEN GOTO undo;

UPDATE parts
SET total_quantity := total_quantity  - 250
WHERE pno = 'P1000';
IF error THEN GOTO undo;

COMMIT TRANSACTION;
GOTO finish;
undo: ROLLBACK TRANSACTION;
finish: RETURN;
```

1.9.4 Ταυτοχρονισμός (concurrency) και συναλλαγές (transaction)

Το ΣΔΒΔ επιτρέπει σε συναλλαγές την ταυτόχρονη προσπέλαση των ιδίων δεδομένων και διαθέτει ένα μηχανισμό ελέγχου που μπορεί να εξασφαλίσει ότι οι ταυτόχρονες συναλλαγές δεν παρεμβάλλονται η μία στη λειτουργία της άλλης. Το πρότυπο για τη γλώσσα SQL απαιτεί οι ενημερώσεις (insert, update, delete) που γίνονται από μία συναλλαγή να μη γίνονται ορατές στις άλλες συναλλαγές παρά μόνο όταν η συναλλαγή τερματιστεί (commit).

Στο κεφάλαιο 2 θα εξετάσουμε τα προβλήματα ταυτοχρονισμού που ανακύπτουν στην εκτέλεση συναλλαγών: Χαμένη ενημέρωση-Lost updates, πρόχειρη ανάγνωση-Dirty reads, Μη επαναληπτική ανάγνωση-Non-repeatable reads, ανάγνωση «φαντασμάτων» - Phantom reads. Θα συζητήσουμε, επίσης, τα επίπεδα απομόνωσης (ISO SQL isolation levels) στην ταυτόχρονη εκτέλεση συναλλαγών: Read Uncommitted, Read Committed, Repeatable read, Serializable.

Η γλώσσα SQL δίδει τη δυνατότητα στον προγραμματιστή για κλειδώματα (locks) εγγραφών κ.λπ.

Τα μεγάλα ΣΔΒΔ συνοδεύονται από ειδικό λογισμικό που αναλαμβάνει τη διαχείριση των συναλλαγών (Online Transaction Processing, OLTP).

1.9.5 Ο ρόλος του Διαχειριστή Βάσεως Δεδομένων (ΔΒΔ)

Μεγάλη σημασία σε ένα σύστημα Β.Δ. έχει η δραστηριότητα του διαχειριστή-ΔΒΔ (Data Base Administrator-DBA), δηλαδή του προσώπου ή της ομάδας προσώπων που έχουν την ευθύνη και το συνολικό έλεγχο του συστήματος. Για να κατανοήσουμε καλύτερα το ρόλο του ΔΒΔ πρέπει να προσθέσουμε ότι:

- Ο ΔΒΔ «φροντίζει» να διασφαλίζεται η ακεραιότητα της βάσης δεδομένων. Βέβαια το ΣΔΒΔ (πρέπει να) εμποδίζει την παραβίαση των κανόνων ακεραιότητας που ορίζει ο ΔΒΔ.
- Ο ΔΒΔ παίρνει εφεδρικά αντίγραφα της βάσης δεδομένων και διεκπεραιώνει την ανάκαμψη της βάσης μετά από αστοχία υλικού (system crash) ή λογισμικού (soft crash). Το ΣΔΒΔ (πρέπει να) εξασφαλίζει τη δυνατότητα ανάκαμψης (recovery) της βάσης κ.λπ.

Παραδείγματα κανόνων ακεραιότητας (Date)

1) Οι κωδικοί προμηθευτών έχουν τη μορφή S9999
2) Οι πόλεις (π.χ., στήλη City), δηλαδή οι έδρες προμηθευτών, καταχωρούνται σε και ανακτώνται από πίνακα.
3) Οι ποσότητες προϊόντων για αποστολή παίρνουν τιμές που είναι πολλαπλάσια του 5
4) Η στήλη (πεδίο) Status παίρνει τιμές στο διάστημα 1-1000

Όλοι αυτοί οι περιορισμοί-κανόνες είναι εξειδικευμένοι (data base specific) για συγκεκριμένη βάση. Υπάρχουν κάποιοι πολλοί σημαντικοί κανόνες γενικοί για όλες τις βάσεις, γνωστοί ως κανόνες ακεραιότητας (1st integrity rule, 2nd integrity rule or referential integrity rule) που θα αναλυθούν σε επόμενη ενότητα.

1.9.6 Ανάκαμψη

Σύμφωνα με τον Date Είναι η επαναφορά από το ΔΒΔ της βάσης σε μία κατάσταση που θεωρείται σωστή (correct) όταν κάποια αστοχία έχει κάνει τη τρέχουσα κατάσταση εσφαλμένη (incorrect) ή τουλάχιστον ύποπτη. Ο τρόπος για να εξασφαλιστεί ότι η βάση δεδομένων μπορεί να επαναφερθεί μετά από πρόβλημα είναι να εξασφαλιστεί ότι μπορούν να ανακατασκευαστούν τα δεδομένα της από κάποιες άλλες πλεονάζουσες πληροφορίες αποθηκευμένες αλλού στο σύστημα κατά τρόπο κρυφό από τον χρήστη.

Ο βασικός μηχανισμός εξασφάλισης της ανάκαμψης σε ένα ΣΔΒΔ συνήθως είναι ο διαχειριστής συναλλαγών (transaction manager).

Στο σημείο αυτό πρέπει να αναφερθεί και πάλι ένα από τα πιο σημαντικά εργαλεία του ΔΒΔ το λεξικό δεδομένων (data dictionary). Το Λεξικό Δεδομένων είναι τμήμα του ΣΔΒΔ και συνήθως είναι από μόνο του μια βάση δεδομένων όπου αποθηκεύονται όλα τα σχήματα (schemas) των εφαρμογών και άλλες πληροφορίες. Στο προϊόν MySQL η βάση δεδομένων ονομάζεται Information Schema.

1.9.7 Παρατηρήσεις και μία συζήτηση για επιτραπέζιες (ή μη) σχεσιακές βάσεις δεδομένων και εργαλεία ανάπτυξης εφαρμογών

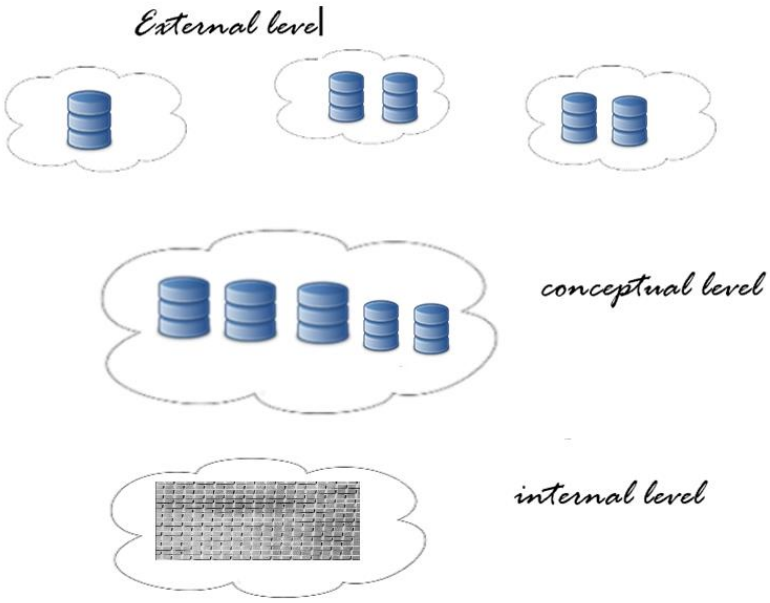
Ενδεχομένως, ο αναγνώστης θα αναρωτηθεί αν κάποια γνωστά και δημοφιλή προϊόντα (όπως το προϊόν MS Access), τα οποία συνήθως χρησιμοποιούνται σε περιβάλλον επιτραπέζιου Η/Υ, είναι ΣΔΒΔ και κάτι ουσιαστικότερο, αν μπορούν να αποτελέσουν την «καρδιά» ενός Συστήματος Βάσης Δεδομένων επιχείρησης. Σύμφωνα με κάποια σημεία από την παραπάνω συζήτηση απουσιάζουν από πολλά δημοφιλή προϊόντα

κρίσιμες συνιστώσες (OLTP κ.λπ.) που είναι απαραίτητες σε "επαγγελματικές" εφαρμογές. Επομένως, δεν είναι εύκολο να τα κατατάξει κανείς στα επαγγελματικά εργαλεία για τη "φιλόδοξη" διαχείριση πολύ μεγάλων βάσεων. Συνήθως ο όρος επιτραπέζιες Βάσεις Δεδομένων (Desktop databases) χαρακτηρίζει αυτά τα προϊόντα. Ακόμη και αυτή η διάκριση, όμως, δεν οριοθετεί τελείως τα πράγματα:

> Ένα σύνθετο προϊόν διαχείρισης βάσης δεδομένων, όπως αυτό της Oracle, μπορεί να αποτελέσει την καρδιά μιας πολύ μεγάλης βάσης δεδομένων και σε έκδοση για προσωπικούς υπολογιστές (Oracle Express) να χρησιμοποιηθεί για την οργάνωση επιτραπέζιας βάσης. Ένας developer μπορεί να εργαστεί με τις γεννήτριες εφαρμογών (application generators) και εκτυπωτικών (αναφορών, report generators) σε προσωπικό υπολογιστή και στη συνέχεια να εκτελεί τις ίδιες ηλεκτρονικές φόρμες σε περιβάλλον client server .

1.10 Αρχιτεκτονική ANSI/SPARC και υλοποίηση συστημάτων βάσης δεδομένων

Η Αρχιτεκτονική ANSI/SPARC περιλαμβάνει τρία επίπεδα (βλέπε Εικόνα 1.31):



Εικόνα 1.31 Η Αρχιτεκτονική ANSI/SPARC και τα τρία επίπεδά της

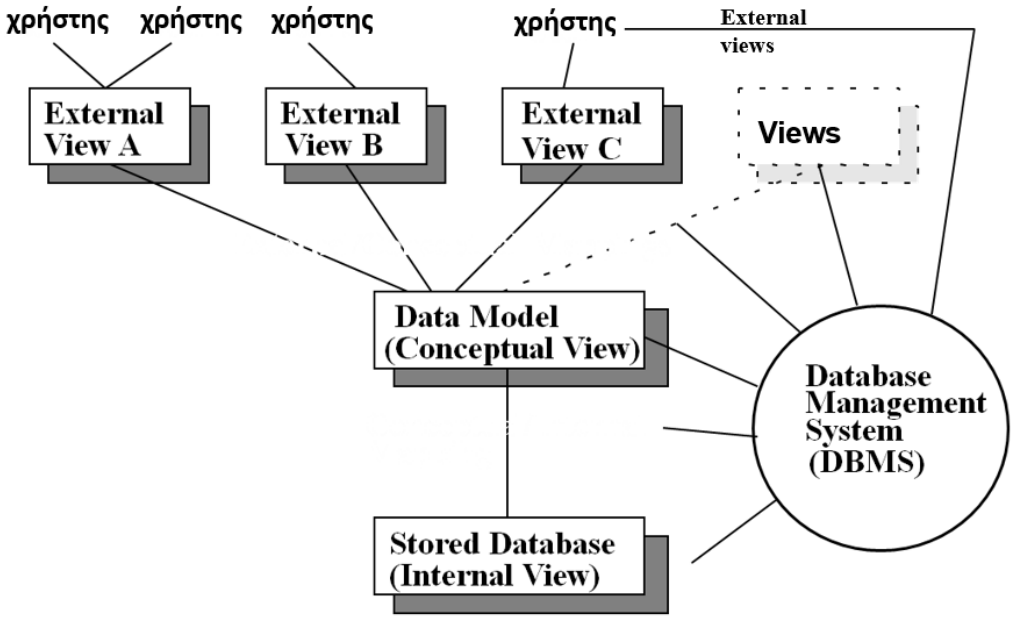
1. **Εσωτερικό επίπεδο** (internal level), δηλαδή η βάση δεδομένων σύμφωνα με την οπτική γωνία αυτού που ασχολείται με την αποθήκευση (οργάνωση) των δεδομένων (data).
2. Αυτό δε σημαίνει ότι η εσωτερική άποψη προσεγγίζει το επίπεδο των περιορισμών του υλικού (hardware)!
3. **Καθολικό ή εννοιολογικό επίπεδο** (conceptual level), δηλαδή η βάση δεδομένων σύμφωνα με την οπτική γωνία αυτού που έχει καθολική (συνολική) εικόνα της Βάσης. Ο Διαχειριστής Βάσεως Δεδομένων (Data Base Administrator-DBA) έχει την ευθύνη της ακεραιότητας της βάσης δεδομένων άρα και της δημιουργίας του πλαισίου (framework) όπου οι χρήστες (προγραμματιστές και τελικοί χρήστες) επεξεργάζονται δεδομένα, "χτίζει " τη βάση δεδομένων (database) της επιχείρησης, ορίζει και τροποποιεί τους τύπους δεδομένων (data types) των στηλών κ.λπ.
4. **Εξωτερικό επίπεδο** (external level) δηλαδή η βάση δεδομένων σύμφωνα με την οπτική γωνία των χρηστών. Οι χρήστες ανήκουν σε κατηγορίες:

- προγραμματιστές που έχουν γλώσσες προγραμματισμού στη διάθεσή τους (π.χ. Java, C++) και το Σ.Δ.Β.Δ. για να δημιουργήσουν και να υποστηρίξουν τη λειτουργία της Β.Δ.
- τελικοί χρήστες (end-users) που έχουν εξειδικευμένες γνώσεις και μπορούν να δημιουργήσουν κάποιες απλές εφαρμογές. Αυτοί χρησιμοποιούν ειδικές γλώσσες φτιαγμένες στα μέτρα τους ή και προγράμματα κατασκευής εφαρμογών που αποτελούν τμήμα του ΣΔΒΔή είναι αυτοτελείς συνιστώσες και συνεργάζονται με αυτό. Συνήθως οι γεννήτριες εφαρμογών και εκτυπώσεων (application generators, report generators) υπάγονται σε αυτήν την κατηγορία.
- τελικοί χρήστες που χρησιμοποιούν προγράμματα εφαρμογής φτιαγμένα από τους προγραμματιστές για εισαγωγή και αναζήτηση στοιχείων από τη βάση δδομένων.

Όλοι αυτοί οι χρήστες δεν ενδιαφέρονται για ολόκληρη τη βάση δεδομένων αλλά για τμήματά της. Δηλαδή, βλέπουν τη βάση σαν το σύνολο των δεδομένων που τους ενδιαφέρει.

Συχνά η βάση δεδομένων σε εξωτερικό επίπεδο (external level) λέγεται λογική (logical data base) και σε Καθολικό επίπεδο (conceptual level) φυσική (physical or stored data base).

Στην Εικόνα 1.32 βλέπουμε την αρχιτεκτονική ANSI-SPARC και το ρόλο του ΣΔΒΔ στη διαχείριση του συστήματος βάσης δεδομένων.



Εικόνα 1.32 Η Αρχιτεκτονική ANSI/SPARC και ο ρόλος του ΣΔΒΔ

1.10.1 Η Υπογλώσσα δεδομένων του ΣΔΒΔ

Το ΣΔΒΔ περιλαμβάνει την υπογλώσσα δεδομένων (ΥΔ). Στα σχεσιακά προϊόντα η ΥΔ είναι η γλώσσα SQL. Η ΥΔ είναι μία γλώσσα ορισμού και διαχείρισης βάσης δεδομένων και αποτελείται από τρεις συνιστώσες:

Γλώσσα ορισμού δεδομένων-ΓΟΔ (DDL-Data Definition Language) που μας επιτρέπει να ορίζουμε μία βάση δεδομένων, τους πίνακές της κ.λπ.

Γλώσσα χειρισμού (ή επεξεργασίας) δεδομένων-ΓΧΔ (Data Manipulation Language-DML) που μας επιτρέπει να διαχειριζόμαστε τα στοιχεία των πινάκων κ.λπ. (να εισάγουμε στοιχεία, να ενημερώσουμε και να διαγράψουμε στοιχεία, να αναζητήσουμε στοιχεία) της βάσης δεδομένων που δημιουργήσαμε με τη ΓΟΔ.

Γλώσσα ελέγχου δεδομένων - ΓΕΔ (DCL-Data Control Language) που μας επιτρέπει να ελέγχουμε και να εξασφαλίζουμε την ασφάλεια ενός συστήματος βάσεων δεδομένων.

1.10.2 Μια απλή προσέγγιση στην αρχιτεκτονική ANSI/SPARC και στην υλοποίηση βάσης δεδομένων με χρήση γλώσσας SQL

Για να έχουμε μια απλή προσέγγιση στο θέμα αρχιτεκτονική ANSI/SPARC θα εστιάσουμε στη χρήση της γλώσσας SQL και θα δοθούν κάποια απλά παραδείγματα:

1.10.2.1 Εννοιολογικό ή καθολικό σχήμα. Δημιουργία πινάκων της βάσης δεδομένων

```
CREATE TABLE DEPT (DEPTNO NUMBER(2), DNAME CHAR(14),
LOC CHAR(13));

CREATE TABLE EMP  (EMPNO NUMBER(4) NOT NULL,
ENAME CHAR(10), JOB CHAR(9), MGR NUMBER(4),
HIREDATE DATE, SAL NUMBER(7,2), COMM NUMBER(7,2),
DEPTNO NUMBER(2), PROJNO NUMBER(3));

CREATE TABLE PROJ (PROJNO NUMBER(3) NOT NULL,
PNAME CHAR(5), BUDGET NUMBER (7,2));
```

Οι ορισμοί αυτοί έγιναν στο προϊόν της Oracle.

Αν θελήσουμε να χρησιμοποιήσουμε το προϊόν MySQL ακολουθεί παράδειγμα δημιουργίας πίνακα με τα βασικά στοιχεία παραγγελίας:

```
CREATE TABLE Orders (Orderno INT(5), Custno INT(4),
SalesID CHAR (6), OrderDate DATE, Totcost FLOAT(7,2)
```

1.10.2.2 . Εξωτερικά σχήματα

Εξωτερικό σχήμα υπαλλήλων με επικεφαλής τον Ανδρέου

```
CREATE VIEW PROJMGR (EMPCODE, EMPLOYEE, SALARY)
AS SELECT  EMPNO, ENAME, SAL
FROM   EMP
WHERE  EMP.MGR = 'ΑΝΔΡΕΟΥ Ν.';
```

Εξωτερικό σχήμα των υπαλλήλων του έργου 100.

```
CREATE VIEW PROJSTAFF
(EMPCODE, EMPLOYEE,  PROJECT, PROJECT_NUMBER)
AS  SELECT EMPNO, ENAME, PNAME, EMP.PROJNO
FROM   EMP, PROJ
WHERE  EMP.PROJNO = PROJ.PROJNO
ANDPROJ.PROJNO = 100;
```

Εξωτερικό σχήμα των υπαλλήλων του Λονδίνου

```
CREATE VIEW LONDON_PROJECTS
   (PROJECT, EMPLOYEE, EMP_NUMBER, LOCATION)
AS  SELECT PNAME, ENAME, EMPNO, LOC
```

```
FROM   PROJ, EMP, DEPT
WHERE  EMP.DEPTNO = DEPT.DEPTNO
ANDEMP.PROJNO = PROJ.PROJNO
ANDDEPT.LOC = 'ΛΟΝΔΙΝΟ';
```

1.10.2.3 Εσωτερικό σχήμα

Ο Διαχειριστής (DBA) ορίζει τους απαραίτητους δείκτες, συστάδες κτλ ώστε να βελτιωθεί ο χρόνος αναζήτησης στοιχείων από τη βάση για συγκεκριμένες αναζητήσεις που ενδιαφέρουν ιδιαίτερα.

Ορισμός δεικτών (ευρετηρίων)

Η δήλωση CREATE INDEX INAME ON EMP (ENAME);

επιταχύνει την αναζήτηση (SELECT) με κριτήριο το όνομα του υπαλλήλου.

Η δήλωση CREATE INDEX ISAL ON EMP (SAL);

επιταχύνει αναζητήσεις με κριτήριο το μισθό.

Ακολουθεί δημιουργία ευρετηρίου που βασίζεται σε δύο στήλες και ταξινομεί κατά αύξουσα αλφαβητική σειρά επωνύμου (Lastname) και φθίνουσα σειρά ονόματος (Firstname), δηλαδή ταξινομεί τα επώνυμα κατά αύξουσα αλφαβητική σειρά και σε περίπτωση συνωνυμίας ταξινομεί όλα τα συνώνυμα κατά φθίνουσα σειρά των ονομάτων τους.

```
CREATE INDEX CustName ON Customers (Lastname ASC,
Firstname DESC)
```

1.10.2.4 Ερωτήσεις (ή ερωτήματα ή αναζητήσεις, queries)

Αναζήτηση όλων των στοιχείων του πίνακα των παραγγελιών

```
SELECT * FROM Orders
```

Αναζήτηση παραγγελιών πελατών από Αγγλία και Ιταλία

```
SELECT * FROM Customers WHERE Country IN (`EN', `IT')
```

Αναζήτηση και εμφάνιση πελατών με αλφαβητική σειρά επωνύμου

```
SELECT * FROM Customers ORDER BY Lastname
```

Εισαγωγή - Ενημέρωση - Διαγραφή στοιχείων

Εισαγωγή στοιχείων από τον πίνακα Customers στον πίνακα Mailings

```
INSERT INTO Mailings
 SELECT Firstname, Lastname, Address, Country FROM Customers;
```

Διαγραφή των πελατών από Γαλλία

```
DELETE FROM Customers WHERE Country='FR'
```

Αύξηση στις τιμές κατά 5%

```
UPDATE Retail Items
SET Retail=Retail*1.05
```

Σημείωση

Στην περίπτωση πολλών προϊόντων, π.χ. MySQL, ORACLE, Microsoft Access υπάρχει η δυνατότητα να κατασκευάσετε τη βάση χρησιμοποιώντας ένα γραφικό εργαλείο. Επίσης, με διάφορα εργαλεία-λογισμικά, π.χ. CASE tools, Rational μπορείτε να κατασκευάσετε και τη βάση δεδομένων και τις ηλεκτρονικές φόρμες χρησιμοποίησης της βάσης αλλά και εκτυπωτικά (αναφορές, report).

1.10.3 Περιορισμοί και επικύρωση δεδομένων σε γλώσσα SQL.

Ακολουθούν παραδείγματα που θα μας βοηθήσουν να κατανοήσουμε τις έννοιες.

Παράδειγμα Περιορισμών

```
create table customer (custid number(5) not null,
   name varchar2(40),
   address varchar2(40),
   CONSTRAINT CUSTOMET-PK PRIMARY KEY (CUSTID),
  /*   Primary Key constraint */
   CONSTRAINT CUSTID_ZERO CHECK (CUSTID > 0);
 /*check constraint */
create table contact(
      empno number(5) CONSTRAINT CONTACT_NN_EMPNO NOT NULL,
      /* NN = Not  Null */
      custid number(5) CONSTRAINT CONTACT_NN_CUSTID NOT NULL,
      subject varchar2(40),
      CONSTRAINT CONTACT_FK_EMP_EMPNO
       FOREIGN KEY (EMPNO) REFERENCES E
      MP(EMPNO),
      CONSTRAINT CONTACT_FK_CUSTOMER_CUSTID
       FOREIGN KEY (CUSTID) REFERENCES CUSTOMER(CUSTID),
      CONSTRAINT CONTACT_PK_EMPNO_CUSTID
       PRIMARY KEY (EMPNO,CUSTID));
```

Πως βλέπουμε τους περιορισμούς

```
SELECT CONSTRAINT_TYPE, TABLE_NAME, CONSTRAINT_NAME
FROM USER_CONSTRAINTS;
```

Είδη περιορισμών

Στην Εικόνα 1.33 δίδονται τα είδη περιορισμών και παραδείγματα ορισμού περιορισμών.

Είδος περιορισμών	Περιγραφή
PK	PRIMARY KEY
FK	FOREIFN KEY
NN	NOT NULL
CC	CHECK CONDITION

```
CREATE TABLE student (
student_id int NOT NULL,
FirstName char(25) NOT NULL,
LastName char(25),
```

```
Age int);
ALTER TABLE STUDENT
ADD PRIMARY KEY (student_id);

CREATE TABLE Orders(
OrderID int PRIMARY KEY,
OrderNumber int NOT NULL,
student_id int);
ALTER TABLE Orders
ADD FOREIGN KEY (student_id) REFERENCES student(student_id);

ALTER TABLE student ADD CHECK (Age>=18);
```

Εικόνα 1.33 Είδη περιορισμών και παραδείγματα

1.10.4 Παράδειγμα για την εμπέδωση της έννοια του μηχανισμού κύριου-ξένου κλειδιού.

Εστιάζουμε στις έννοιες των Σχεσιακών Βάσεων Δεδομένων και στα Κύρια και ξένα κλειδιά που διασφαλίζουν την τήρηση των κανόνων ακεραιότητα. Έστω η βάση δεδομένων προσωπικού εταιρείας my_personnel της εικόνας 1.34.

βάση δεδομένων προσωπικού της εταιρείας my_personnel.

emp (πίνακας υπαλλήλων)

Empno	Ename	Job	Hiredate	Mgr	Sal	Comm	Deptno
10	Codd	ANALYST	1/1/89	15	3000		10
15	Elmasri	ANALYST	2/5/95	15	1200	150	10
20	Navathe	SALESMAN	7/7/77	20	2000		20
30	Date	PROGRAMMER	4/5/04	15	1800	200	10

dept (πίνακας τμημάτων)

Deptno	Dname	Loc
10	ACCOUNTING	ATHENS
20	SALES	LONDON
30	RESEARCH	ATHENS
40	PAYROLL	LONDON

Εικόνα 1.34 Δείγμα δεδομένων (sample of data) για το οποίο ισχύουν οι κανόνες ακεραιότητας.

Μέσα από τον παρακάτω «διάλογο» (SQL session) με το σύστημα (προϊόν MySQL) μπορείτε να κατανοήσετε περισσότερο τον μηχανισμό κύριου-ξένου κλειδιού και τους κανόνες ακεραιότητας.

Δημιουργία πινάκων και εισαγωγή των στοιχείων.

Δοκιμάστε τις παρακάτω ενέργειες:

Διαγράψτε τη βάση δεδομένων my_personnel, αν υπάρχει, δημιουργήστε την και χρησιμοποιήστε την.

```
DROP DATABASE IF EXISTS my_personnel;
CREATE DATABASE my_personnel;
USE my_personnel;
```

Προσπαθήστε να δημιουργήσετε τον πίνακα EMP:

```
CREATE TABLE EMP(EMPNO INT(4) NOT NULL,
ENAME VARCHAR(10), JOB VARCHAR(25),
HIREDATE DATE, MGR INT(4),  SAL FLOAT(7,2), COMM FLOAT(7,2),
DEPTNO INT(2), PRIMARY KEY(EMPNO),
 FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO));
```

Γιατί δεν δημιουργείται;

Δημιουργήστε τον πίνακα DEPT.

```
CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL,
 DNAME VARCHAR(14), LOC VARCHAR(14),
 PRIMARY KEY(DEPTNO));
```

Δείτε ότι δημιουργήθηκε.

```
SHOW TABLES;
```

Προσπαθήστε και πάλι να δημιουργήσετε τον πίνακα EMP.

Τι παρατηρείτε τώρα; Μήπως δημιουργήθηκε;

```
SHOW TABLES;
```

Γράψτε την παρακάτω δήλωση:

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (40, 'OPERATIONS', 'BOSTON');
```

Δείτε ότι εκτελείται κανονικά.

Γράψτε την παρακάτω δήλωση:

```
INSERT INTO EMP
VALUES (10, 'CODD', 'ANALYST', '1989/01/01', 15, 3000, NULL, 10);
```

Γιατί δεν εκτελείται;

Αμέσως μετά δοκιμάστε πρώτα να εκτελέσετε τη δήλωση:

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
```

Δοκιμάστε τώρα αν εκτελείται η δήλωση:

```
INSERT INTO EMP
VALUES (10, 'CODD', 'ANALYST', '1989/01/01', 15, 3000, NULL, 10);
```

Εκτελέστε με σωστή σειρά τις παρακάτω δηλώσεις:

```
INSERT INTO EMP
VALUES (15, 'ELMASRI', 'ANALYST', '1995/05/02', 15, 1200, 150, 10);
INSERT INTO EMP
VALUES (20, 'NAVATHE', 'SALESMAN', '1977/07/07', 20, 2000, NULL, 20);
INSERT INTO EMP
```

```
VALUES (30, 'DATE', 'PROGRAMMER', '2004/05/04', 15, 1800, 200, 10);
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (30, 'SALES', 'CHICAGO');
```

Αν όλα πήγαν καλά θα δείτε όλα τα δεδομένα των πινάκων:

```
SELECT * FROM EMP;
SELECT * FROM DEPT;
```

Άσκηση.

Διορθώσετε το παρακάτω script:

```
# re-organize the following statements
DROP DATABASE IF EXISTS PERSONNEL;
CREATE DATABASE personnel;
USE my_personnel;

CREATE TABLE EMP(EMPNO INT(4) NOT NULL,
ENAME VARCHAR(10), JOB VARCHAR(25),
HIREDATE DATE, MGR INT(4),  SAL FLOAT(7,2), COMM FLOAT(7,2),
DEPTNO INT(2), PRIMARY KEY(EMPNO),
 FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO));

CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL,
 DNAME VARCHAR(14), LOC VARCHAR(14),
 PRIMARY KEY(DEPTNO));

SHOW TABLES;

INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (40, 'OPERATIONS', 'BOSTON');
INSERT INTO EMP
 VALUES (10, 'CODD', 'ANALYST', '1989/01/01', 15, 3000, NULL, 10);
INSERT INTO EMP
VALUES (15, 'ELMASRI', 'ANALYST', '1995/05/02', 15, 1200, 150, 10);
INSERT INTO EMP
VALUES (20, 'NAVATHE', 'SALESMAN', '1977/07/07', 20, 2000, NULL, 20);
INSERT INTO EMP
VALUES (30, 'DATE', 'PROGRAMMER', '2004/05/04', 15, 1800, 200, 10);
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (30, 'SALES', 'CHICAGO');

SELECT * FROM EMP;
SELECT * FROM DEPT;
```

Απάντηση

```
DROP DATABASE IF EXISTS newpersonnel;
CREATE DATABASE newpersonnel;
USE newpersonnel;

CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL,
 DNAME VARCHAR(14), LOC VARCHAR(14),
 PRIMARY KEY(DEPTNO));

CREATE TABLE EMP(EMPNO INT(4) NOT NULL,
ENAME VARCHAR(10), JOB VARCHAR(25),
HIREDATE DATE, MGR INT(4),  SAL FLOAT(7,2), COMM FLOAT(7,2),
DEPTNO INT(2), PRIMARY KEY(EMPNO),
 FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO));

SHOW TABLES;

INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (40, 'OPERATIONS', 'BOSTON');

INSERT INTO EMP
VALUES (10, 'CODD', 'ANALYST', '1989/01/01', 15, 3000, NULL, 10);
INSERT INTO EMP
VALUES (15, 'ELMASRI', 'ANALYST', '1995/05/02', 15, 1200, 150, 10);
INSERT INTO EMP
VALUES (20, 'NAVATHE', 'SALESMAN', '1977/07/07', 20, 2000, NULL, 20);
INSERT INTO EMP
VALUES (30, 'DATE', 'PROGRAMMER', '2004/05/04', 15, 1800, 200, 10);
SELECT * FROM EMP;
SELECT * FROM DEPT;
```

1.11 Γιατί χρησιμοποιούμε τελικά βάση δεδομένων. Η σημασία της μοντελοποίησης δεδομένων

Για εκπαιδευτικούς λόγους αλλά και για μία εμβάθυνση σε κάποια θέματα θα αναφέρουμε πλεονεκτήματα της χρήσης βάσεων (αν και ίσως στις μέρες μας δεν έχει πλέον νόημα να συζητάμε γιατί πρέπει να χρησιμοποιούμε βάση δεδομένων). Στην παρακάτω συζήτηση κυρίως χρησιμοποιήσαμε ως αφετηρία τα συγγράμματα των Elmasri-Navathe (σελ. 30-48) και Date.

Ο βασικός λόγος χρησιμοποίησης συστημάτων βάσεων δεδομένων σε μία επιχείρηση ή ένα οργανισμό είναι το γεγονός ότι ένα τέτοιο σύστημα οργάνωσης και ανάκτησης δεδομένων, "οικοδομείται" με την εποπτεία και την καθοδήγηση του οργανισμού ώστε να καλύψει τις ανάγκες του συνόλου των τμημάτων και εφοδιάζει τον οργανισμό ή την επιχείρηση με κεντρικό έλεγχο (centralized control) των λειτουργικών στοιχείων (operational data) δηλαδή:

- ο σχεδιασμός των εφαρμογών γίνεται κεντρικά από ειδικούς
- υιοθετούνται πρότυπα
- αποφεύγονται πλεονασμοί κατά τη φύλαξη στοιχείων
- κάθε νέα εφαρμογή αποτελεί αντικείμενο μελέτης και πρέπει να είναι συμβατή με τις παλαιότερες εφαρμογές
- οι εφαρμογές κατασκευάζονται με τη χρήση των ίδιων εργαλείων διαχείρισης βάσεων δεδομένων κ.λπ.
- Πρόσθετα μπορούμε να αναφέρουμε ότι :
- μειώνεται ο χρόνος υλοποίησης και συντήρησης εφαρμογών
- υπάρχει μεγαλύτερη ευελιξία στην περίπτωση αλλαγής απαιτήσεων των χρηστών
- υποστηρίζεται η άμεση διαθεσιμότητα ενημερωμένων πληροφοριών
- υποστηρίζονται πολλαπλές όψεις (views) των δεδομένων
- υποστηρίζεται η αποθήκευση μεταδεδομένων περιγραφής της βάσης
- υποστηρίζεται η ανεξαρτησία προγραμμάτων και δεδομένων και η αφαίρεση δεδομένων
- υποστηρίζεται η διαμοίραση δεδομένων ανάμεσα σε ταυτόχρονους χρήστες
- υποστηρίζεται η άμεση επεξεργασία συναλλαγών (transactions)
- περιορίζεται η μη εξουσιοδοτημένη προσπέλαση
- παρέχονται πολλές διεπαφές χρηστών (user interfaces)
- υποστηρίζεται η επιβολή περιορισμών ορθότητας που πρέπει να ισχύουν για τα δεδομένα
- υποστηρίζεται η λήψη και χρήση εφεδρικών αντιγράφων (back-up)
- υποστηρίζονται διαδικασίες ανάκαμψης από βλάβες (recovery)
- υποστηρίζεται η αναπαράσταση στη βάση δεδομένων πολύπλοκων συσχετίσεων μεταξύ των δεδομένων

Οι Elmasri-Navathe αναφέρουν σαν πλεονέκτημα:

- Για τα επαγωγικά ΣΔΒΔ τις δυνατότητες ορισμού κανόνων επαγωγής για την εξαγωγή νέων πληροφοριών (συμπερασμάτων) από αποθηκευμένα στη βάση γεγονότα.
- Για τα αντικειμενοστρεφή ΣΔΒΔ τη δυνατότητα μόνιμης αποθήκευσης για αντικείμενα προγραμμάτων και δομές δεδομένων.

Τέλος υπογραμμίζουν ότι η προσέγγιση των ΣΒΔ/ΣΔΒΔ επιτρέπει σε όλο τον οργανισμό να επενδύσει σε πιο ισχυρούς επεξεργαστές, μονάδες αποθήκευσης ή εξοπλισμό επικοινωνιών αντί να αγοράζει κάθε τμήμα ανεξάρτητα το δικό του. Έτσι ελαττώνεται το συνολικό κόστος λειτουργίας και διαχείρισης (οικονομία κλίμακος)

Στη συνέχεια θα δείξουμε κάποια από τα πλεονεκτήματα της χρησιμοποίησης βάσης δεδομένων με ένα παράδειγμα.

1.11.1 Παράδειγμα (οφείλεται στον Peter Chen)

Έστω ότι μία εταιρεία χρειάζεται ένα σύστημα βάσης δεδομένων που να κρατά στοιχεία για τους υπαλλήλους της και στοιχεία για τα τμήματα (department) της. Μετά από τη σχετική ανάλυση που γίνεται σε συνεργασία με τη «Διεύθυνση Προσωπικού» που θα έχει την ευθύνη της λειτουργίας του συστήματος επιλέγεται η γραμμογράφηση πίνακα της εικόνας 1.35.

στήλη	επεξήγηση	τύπος δεδομένων
soc_sec_no	αριθμός ασφάλισης	char(11)
Name	όνομα υπαλλήλου	char(20)
Age	Ηλικία	int(2)
Deptno	κωδικός τμήματος	int(3)
Budget	προϋπολογισμός	int(10)

Στήλη	επεξήγηση	τύπος δεδομένων
Ssno	κωδικός ασφάλισης	char(11)
emp_name	όνομα υπαλλήλου	char(20)
Age	Ηλικία	real(3,2)
Projno	Έργο	int(4)
Name	ονομασία έργου	char(30)
%time	χρόνος συμμετοχής	int(2)

Εικόνα 1.35 Δύο βάσεις δεδομένων της ίδιας εταιρίας.

Παράλληλα μια ομάδα υπαλλήλων στην επιχείρηση που εργάζεται στη «Διεύθυνση Παρακολούθησης Έργων» υλοποιεί ένα σύστημα βάσεως δεδομένων παρακολούθησης των έργων και καταχώρησης των στοιχείων των υπαλλήλων που εργάζονται σε διάφορα έργα (projects) (Εικόνα 1.35). Δηλαδή, στις δύο εφαρμογές που δημιουργήθηκαν χωρίς κανένα συντονισμό ανάμεσα στις δύο διευθύνσεις χρησιμοποιήθηκαν συνώνυμα για την ίδια στήλη, ίδιο όνομα για διαφορετικές στήλες, διαφορετικοί τύποι δεδομένων για την ίδια στήλη κ.λπ. Η αλλαγή της τιμής μιας κοινής στήλης των δύο συστημάτων μόνο στο ένα σύστημα συνεπάγεται ασυνέπεια (incosistent data) των στοιχείων των δύο συστημάτων. Το φαινόμενο αυτό ονομάζεται ανωμαλία κατά την ενημέρωση ή την διαγραφή (update-delete anomalies). Φάρμακο για τις δυσκολίες και τα προβλήματα που αναφέρθηκαν είναι η χρήση ΒΔ.

Βέβαια όπως έδειξε η καθημερινή πρακτική ποτέ δεν αρκεί η αγορά ενός ΣΔΒΔ και εξοπλισμού από μόνη της για τη δημιουργία ΒΔ. Για παράδειγμα, σε μία τράπεζα θα μπορούσε να χρησιμοποιείται το προϊόν διαχείρισης βάσης IMS (ιεραρχικό) για τις καταθέσεις και το (σχεσιακό) DB2 για καταθέσεις σε συνάλλαγμα και οι δύο εφαρμογές να είναι ανεξάρτητες (να "οικοδομήθηκαν" οι εφαρμογές ερήμην η μία της άλλης, σε διαφορετικά αρχεία κ.λπ.).

1.12 Μοντέλο οντοτήτων συσχετίσεων και σχεσιακή βάση δεδομένων. Πρώτοι κανόνες μετασχηματισμού ΜΟΣ σε βάση δεδομένων.

1.12.1 Αρχή με παράδειγμα. Εννοιολογική μοντελοποίηση και κανονικοποίηση βάσεως δεδομένων για τις αμερικανικές προεδρικές εκλογές

Πρέπει να ακολουθήσετε κάποια βήματα τυπικά για την ανάλυση δεδομένων:

1) Να κατασκευάσετε ένα αντιπροσωπευτικό δείγμα δεδομένων σε έναν πίνακα με όλες τις στήλες (όλα τα χαρακτηριστικά των οντοτήτων) που να περιγράφει τις εκλογές. Θυμίζουμε ότι μπορείτε να έχετε και περισσότερους από έναν πίνακες στην 1NF.

2) Να καταγράψετε τους περιορισμούς που ισχύουν για τα δεδομένα

3) Να σχεδιάσετε το ΜΟΣ

4) Να γράψετε τους πίνακες της βάσης δεδομένων που αντιστοιχούν στο μοντέλο.

Στο δείγμα της εικόνας 1.36 παρατίθενται εκλογικά αποτελέσματα για τις αναμετρήσεις από το 1952 έως το 2020. Η βάση δεδομένων είναι στην πρώτη κανονική μορφή. Σημαντικό είναι να ορίσουμε το Κύριο κλειδί. Στην περίπτωσή μας είναι σύνθετο, ίσο με (year, loser).

Πρώτη κανονική μορφή

YEAR	WINNER	W_ VOTES	W_PARTY	W_STATE	LOSER	L_VOTES	L_PARTY
1952	EISENHOWER	442	REPUBLICAN	TEXAS	STEVENSON	89	DEMOCRAT
1956	EISENHOWER	457	REPUBLICAN	TEXAS	STEVENSON	73	DEMOCRAT
1960	KENNEDY	303	DEMOCRAT	MASS	NIXON	219	REPUBLICAN
1964	JOHNSON	486	DEMOCRAT	TEXAS	GOLDWATER	52	REPUBLICAN
1968	NIXON	301	REPUBLICAN	CALIFORNIA	HUMPHREY	191	DEMOCRAT
1968	NIXON	301	REPUBLICAN	CALIFORNIA	WALLACE	46	INDEPENDENT
1972	NIXON	520	REPUBLICAN	CALIFORNIA	McGOVERN	17	DEMOCRAT
1976	CARTER	297	DEMOCRAT	NULL	FORD	240	REPUBLICAN
1980	REAGAN	489	REPUBLICAN	NULL	CARTER	49	DEMOCRAT
1980	REAGAN	489	REPUBLICAN	NULL	ANDERSON	0	INDEPENDENT
1984	REAGAN	525	REPUBLICAN	NULL	MONDALE	13	DEMOCRAT
1988	G. BUSH	426	REPUBLICAN	NULL	DOUKAKIS	111	DEMOCRAT
1992	B. CLINTON	370	DEMOCRAT	ARKANSAS	G. BUSH	168	REPUBLICAN
1992	B. CLINTON	370	DEMOCRAT	ARKANSAS	PERAULT	0	INDEPENDENT
1996	B. CLINTON	379	DEMOCRAT	ARKANSAS	DOLE	159	REPUBLICAN
2000	G.W. BUSH	271	REPUBLICAN	NULL	GORE	266	DEMOCRAT
2004	G.W. BUSH	286	REPUBLICAN	NULL	KERRY	251	DEMOCRAT
2008	OBAMA	365	DEMOCRAT	NULL	McCAIN	173	REPUBLICAN
2012	OBAMA	332	DEMOCRAT	NULL	ROMNEY	206	REPUBLICAN
2016	TRUMP	304	REPUBLICAN	NULL	H.R. CLINTON	227	DEMOCRAT
2020	BIDEN	306	DEMOCRAT	NULL	TRUMP	232	REPUBLICAN

Κύριο κλειδί (Primary key)=(year, loser)

Εικόνα 1.36 Δείγμα δεδομένων των εκλογικών αναμετρήσεων (1952-2020) των αμερικανικών προεδρικών αλλαγών. Πρώτη κανονική μορφή

Περιορισμοί

- Το έτος (Year) χαρακτηρίζει μοναδικά την εκλογική αναμέτρηση. Πιο συγκεκριμένα, το έτος χαρακτηρίζει μοναδικά κάποιες στήλες που περιγράφουν την εκλογική αναμέτρηση. Δηλαδή, αν σκεφτούμε το έτος μίας εκλογικής αναμέτρησης τότε αυτομάτως έρχεται στο μυαλό μας ακριβώς ένας νικητής, ο Πρόεδρος, ακριβώς ένα κόμμα, αυτό που νίκησε στις εκλογές κ.λπ.
- year - -> winner, w_votes, w_party, w_state
- Ο νικητής, ανήκει ισόβια ως υποψήφιος στο ίδιο κόμμα και ξεκινά από την ίδια πολιτεία
- winner - -> w_party, w_state
- Η εκλογική αναμέτρηση και ο ηττημένος ορίζουν τις ψήφους εκλεκτόρων που έλαβε ο ηττημένος

- year, loser - - > l_votes

Προσοχή! Αν είχαμε και τις πρώτες εκλογικές αναμετρήσεις τότε οι περιορισμοί και η βάση δεδομένων πρέπει να αλλάξουν. Δείτε για παράδειγμα τα δεδομένα της εικόνας 1.37.

Έτος	Υποψήφιοι	Κόμμα - Ψήφοι εκλεκτόρων
1789	George Washington John Adams	(no party)69 (no party)34
1792	George Washington John Adams George Clinton Thomas Jefferson Aaron Burr	Federalist 132 Federalist 77 Anti-Federalist 50 Anti-Federalist 4 Anti-Federalist 1
1796	John Adams Thomas Jefferson Thomas Pinckney Aaron Burr	Federalist71 Dem.-Rep.68 Federalist59 Dem.-Rep.30
1800	Thomas Jefferson Aaron Burr John Adams Charles C. Pinckney John Jay	Dem.-Rep. 73 Dem.-Rep. 73 Federalist 65 Federalist 64 Federalist 1

Εικόνα 1.37 Δείγμα δεδομένων των εκλογικών αναμετρήσεων (1789-1800) των αμερικανικών προεδρικών αλλαγών

Μοντέλο οντοτήτων συσχετίσεων

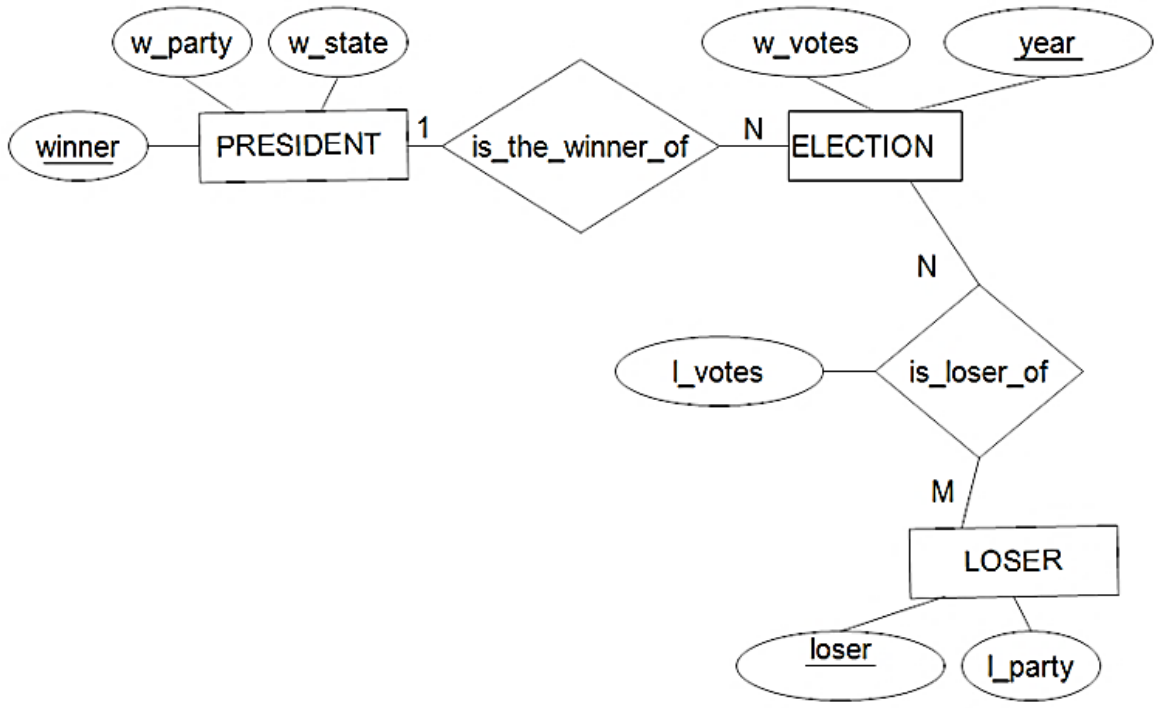
Υπάρχουν διάφοροι συμβολισμοί για να σχεδιάσουμε το Μοντέλο Οντοτήτων Συσχετίσεων (ΜΟΣ)

- Ο συμβολισμός που προτάθηκε από τον Peter Chen. Ο συμβολισμός με κάποιες επεκτάσεις του (Enhanced Entity Relationship Model) περιγράφεται αναλυτικά σε όλα τα γνωστά συγγράμματα (βλέπε π.χ., Navathe και Elmasri, Ulman και Widow, κ.λπ.)
- Ο συμβολισμός που ακολουθείται από Case Tools, όπως το Oracle Designer Case Tool
- Η μεθοδολογία UML περιλαμβάνει εργαλεία εννοιολογικής/λογικής μοντελοποίησης, όπως Class Diagram, Logical Data Model
- Το εργαλείο MySQL Workbench επιτρέπει να σχεδιάσουμε διάφορα εννοιολογικά/λογικά μοντέλα.

Εργαλεία σχεδίασης Μοντέλου οντοτήτων συσχετίσεων

1) mySQL Workbench https://dev.mysql.com/downloads/workbench/
2) Dia https://sourceforge.net/projects/dia-installer/
3) Rational http://www-01.ibm.com/software/awdtools/developer/rose/
4) Microsoft Visio http://office.microsoft.com/en-us/visio/

Ακολουθεί το κλασσικό Μοντέλο Οντοτήτων Συσχετίσεων για τις Αμερικανικές Προεδρικές εκλογές. Σχεδιάστηκε στο εργαλείο Dia (Εικόνα 1.38).



Εικόνα.1.38 Μοντέλο οντοτήτων συσχετίσεων των εκλογικών αναμετρήσεων (1952-) των αμερικανικών προεδρικών αλλαγών

Στη συνέχεια στην Εικόνα 1.39 παραθέτουμε τους τέσσερις πίνακες της τρίτης κανονικής μορφής στους οποίους επιμερίζονται τα στοιχεία των εκλογών. Το κύριο κλειδί αναγράφεται κάτω από κάθε πίνακα.

Πίνακας PRESIDENTS

WINNER	W_PARTY	W_STATE
EISENHOWER	REPUBLICAN	TEXAS
KENNEDY	DEMOCRAT	MASS
JOHNSON	DEMOCRAT	TEXAS
NIXON	REPUBLICAN	CALIFORNIA
CARTER	DEMOCRAT	NULL
REAGAN	REPUBLICAN	NULL
G. BUSH	REPUBLICAN	NULL
B. CLINTON	DEMOCRAT	ARKANSAS
G.W. BUSH	REPUBLICAN	NULL
OBAMA	DEMOCRAT	NULL
TRUMP	REPUBLICAN	NULL
BIDEN	DEMOCRAT	NULL

Κύριο κλειδί (Primary key)=Winner

Πίνακας ELECTION_WINNER

YEAR	WINNER	W_VOTES
1952	EISENHOWER	442

1956	EISENHOWER	457
1960	KENNEDY	303
1964	JOHNSON	486
1968	NIXON	301
1972	NIXON	520
1976	CARTER	297
1980	REAGAN	489
1984	REAGAN	525
1988	G. BUSH	426
1992	B. CLINTON	370
1996	B. CLINTON	379
2000	G.W. BUSH	271
2004	G.W. BUSH	286
2008	OBAMA	365
2012	OBAMA	332
2016	TRUMP	304
2020	BIDEN	306

Κύριο κλειδί=year

Πίνακας ELECTION_LOSER

YEAR	LOSER	L_VOTES
1952	STEVENSON	89
1956	STEVENSON	73
1960	NIXON	219
1964	GOLDWATER	52
1968	HUMPHREY	191
1968	WALLACE	46
1972	McGOVERN	17
1976	FORD	240
1980	CARTER	49
1980	ANDERSON	0
1984	MONDALE	13
1988	DOUKAKIS	111
1992	G. BUSH	168
1992	PERAULT	0
1996	DOLE	159
2000	GORE	266
2004	KERRY	251
2008	McCAIN	173

2012	ROMNEY	206
2016	H.R. CLINTON	227
2020	TRUMP	232

Κύριο κλειδί=(year, loser)

Πίνακας LOSERS

LOSER	L_PARTY
STEVENSON	DEMOCRAT
NIXON	REPUBLICAN
GOLDWATER	REPUBLICAN
HUMPHREY	DEMOCRAT
WALLACE	INDEPENDENT
McGOVERN	DEMOCRAT
FORD	REPUBLICAN
CARTER	DEMOCRAT
ANDERSON	INDEPENDENT
MONDALE	DEMOCRAT
DOUKAKIS	DEMOCRAT
G. BUSH	REPUBLICAN
PERAULT	INDEPENDENT
DOLE	REPUBLICAN
GORE	DEMOCRAT
KERRY	DEMOCRAT
McCAIN	REPUBLICAN
ROMNEY	REPUBLICAN
H.R. CLINTON	DEMOCRAT
TRUMP	REPUBLICAN

Κύριο κλειδί=loser

Εικόνα 1.39 Τρίτη κανονική μορφή της βάσης δεδομένων των εκλογικών αναμετρήσεων (1952-) των αμερικανικών προεδρικών αλλαγών. Το κύριο κλειδί αναγράφεται κάτω από κάθε πίνακα.

1.13 Εννοιολογική σχεδίαση και Μοντέλο Οντοτήτων Συσχετίσεων

Έως τώρα μελετήσαμε κάποια παραδείγματα μοντέλων και αναφερθήκαμε στις σχεσιακές βάσεις δεδομένων. Στη συνέχεια θα μελετήσουμε λεπτομερέστερα τη σχεδίαση του Μοντέλου Οντοτήτων Συσχετίσεων (ΜΟΣ).

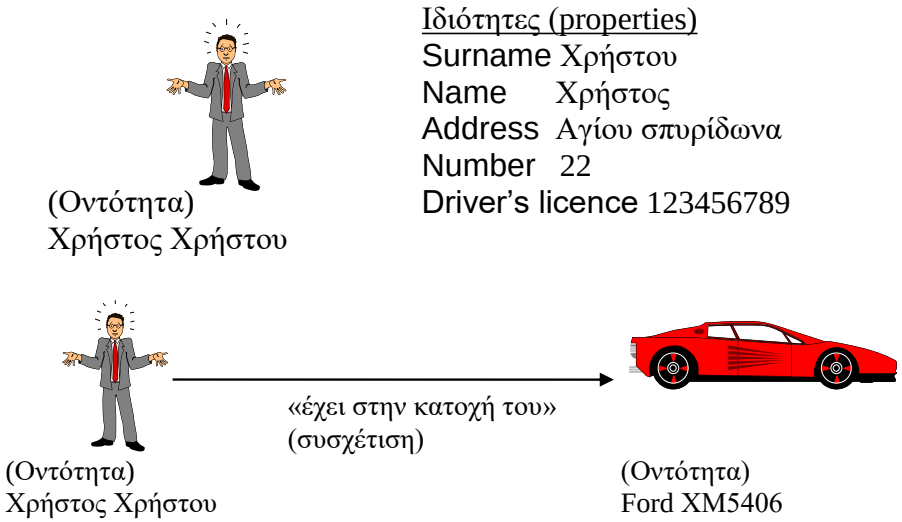
Δομικά στοιχεία Μοντέλου Οντοτήτων Συσχετίσεων (ΜΟΣ)

Διακρίνουμε δύο επίπεδα αφαίρεσης (abstraction):

1. Απλό επίπεδο αφαίρεσης (Simple abstraction level). Περιλαμβάνει οντότητες (Entities), ιδιότητες (Properties), γεγονότα (Facts), συσχετίσεις (relationships)

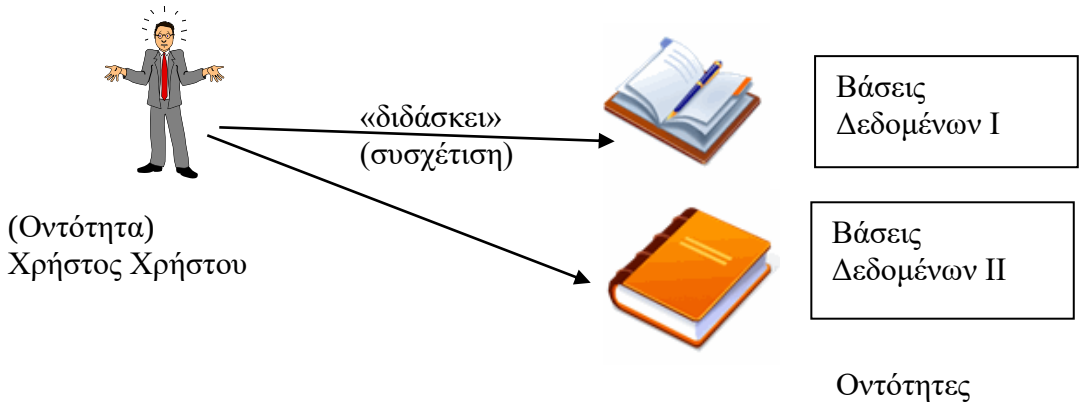
2. Σύνθετο επίπεδο αφαίρεσης (Complex abstraction level). Περιλαμβάνει τύπους-σύνολα οντοτήτων (Entity sets), πεδία ορισμού τιμών (Domains), χαρακτηριστικά τύπου οντότητας (Entity attributes), τύπους-σύνολα συσχετίσεων (Relationship sets), χαρακτηριστικά τύπου συσχέτισης (Relationship attributes).

Στο πρώτο επίπεδο εξετάζουμε οντότητες π.χ. μεμονωμένους προγραμματιστές, όπως ο Χρήστος Χρήστου, βλέπουμε τις ιδιότητές τους, όπως surname, name, address, κ.λπ., καταγράφουμε γεγονότα, όπως ο Χρήστος Χρήστου έχει address=Αγίου Σπυρίδωνα, number=22 και εξετάζουμε συσχετίσεις με άλλες οντότητες. Σχετικά μπορείτε να δείτε τα παραδείγματα που ακολουθούν (Εικόνα 1.40).



Εικόνα 1.40 Παράδειγμα οντότητας (entity) και των ιδιοτήτων (properties) της. Παράδειγμα συσχέτισης

Στην Εικόνα 1.41 παρατίθεται και άλλο παράδειγμα συσχέτισης



Εικόνα 1.41. Παράδειγμα συσχέτισης

Στο δεύτερο επίπεδο αφαίρεσης εξετάζουμε συλλογές οντοτήτων ή τύπους οντοτήτων. Οι αγγλικοί όροι που χρησιμοποιούνται στη βιβλιογραφία είναι entity sets, entity types. Για παράδειγμα, ο τύπος οντότητας ΠΡΟΓΡΑΜΜΑΤΙΣΤΗΣ περιλαμβάνει τους μεμονωμένους προγραμματιστές, όπως ο Χρήστος Χρήστου. Στη συνέχεια, καθορίζουμε τα χαρακτηριστικά, των τύπων οντοτήτων, που ενδιαφέρουν, όπως surname, name, address, κ.λπ., και καταγράφουμε τα αντίστοιχα πεδία ορισμού τιμών. Για παράδειγμα, καθορίζουμε τις νόμιμες τιμές για το χαρακτηριστικό address. Τέλος, εξετάζουμε σύνολα συσχετίσεων και τα χαρακτηριστικά τους. Σχετικά μπορείτε να δείτε τα παραδείγματα που ακολουθούν στο κεφάλαιο αυτό.

Χαρακτηριστικά –attributes- οντότητας

Για τα ονόματα των χαρακτηριστικών χρησιμοποιούμε αγγλικές λέξεις ή λέξεις με λατινικούς χαρακτήρες ή greeklish: Surname (επώνυμο), Name (όνομα), Address (διεύθυνση), Number (αριθμός), Driver's licence (αριθμός άδειας οδήγησης)

1.13.1 Κατανόηση της διαφοράς οντότητας και τύπου οντότητας με παραδείγματα

Όταν μοντελοποιούμε, δηλαδή όταν σχεδιάζουμε το ΜΟΣ, αντί να μιλάμε για μεμονωμένες οντότητες μιλάμε για τύπους οντοτήτων. Πιο συγκεκριμένα, ο Αναλυτής Δεδομένων αρχικά μελετά τις οντότητες που παρουσιάζουν ενδιαφέρουν. Για παράδειγμα, μελετά οντότητες σπουδαστών και προσπαθεί να καταλάβει ποιά στοιχεία (δεδομένα) των σπουδαστών πρέπει να αποθηκευτούν στη βάση δεδομένων που σχεδιάζει. Στη συνέχεια, αποφασίζει ποιά χαρακτηριστικά ενδιαφέρουν το σύνολο (τύπο) οντοτήτων. Για παράδειγμα, αποφασίζει ότι για κάθε τύπο οντότητας ΣΠΟΥΔΑΣΤΗΣ, άρα και για κάθε σπουδαστή, θα αποθηκεύονται: Επώνυμο, Όνομα, Έτος γέννησης, Εξάμηνο σπουδών. Επομένως, ο τύπος οντότητας «Σπουδαστής» θα έχει τελικά τα χαρακτηριστικά Επώνυμο, Όνομα, Έτος γέννησης, Εξάμηνο σπουδών. Επιπλέον, ο τύπος οντότητας «Σπουδαστής» θα αντιπροσωπεύει όλες τις οντότητες που εξετάσθηκαν.

1.13.1.1 Οπτικοποίηση της διαφοράς οντότητας, τύπου οντότητας

Στην Εικόνα 1.42 βλέπουμε έργο του Γαϊτη. Θα μπορούσαμε να ξεχωρίσουμε δύο (2) τύπους οντοτήτων. Αριστερά θα μπορούσαμε να έχουμε οντότητες σπουδαστών και δεξιά οντότητες καθηγητών. Αυτό που πρέπει να αποφασισθεί είναι ποιά χαρακτηριστικά θα έχει ο τύπος οντότητας σπουδαστής και ποιά ο τύπος οντότητας καθηγητής. Κάτι ανάλογο συμβαίνει στις περισσότερες εφαρμογές λογισμικού.

Εικόνα 1.42 Οπτικοποίηση οντοτήτων καθηγητών και φοτητών

Στην Εικόνα 1.43 εστιάζουμε στο γνωστό δίλημμα: Τι είναι οντότητα και τι χαρακτηριστικό;

Στο έργο του Γαΐτη θα μπορούσαμε να έχουμε δύο (2) τύπους οντοτήτων: κυνηγός, θήραμα. Ένα χαρακτηριστικό του κυνηγού μπορεί να είναι και το όπλο του. Κάποιος άλλος σχεδιαστής εφαρμογών λογισμικού ίσως θα έβλεπε σαν οντότητα και το όπλο και θα μπορούσε να γράψει χαρακτηριστικά και για την οντότητα αυτή.

Εικόνα 1.43 Οπτικοποίηση του διλήμματος: Το όπλο είναι οντότητα ή χαρακτηριστικό;

Ακολουθούν παραδείγματα Χαρακτηριστικών (attributes) τύπων Οντοτήτων και Συσχετίσεων οντοτήτων.

1) Ο Σπουδαστής Κυριακόπουλος Νικηφόρος έχει ένα επώνυμο (= Κυριακόπουλος), ένα όνομα (=Νικηφόρος), ένα μοναδικό αριθμό μητρώου (=213).
2) Ο Σπουδαστής Παπαπέτρου Νικόλαος ανήκει σε ένα ακριβώς Εξάμηνο (= Β)
3) Η Σπουδάστρια Αποστόλου Ζωή έχει μία διεύθυνση (= Μάρκου Μπότσαρη 14)
4) Ο Καθηγητής Ullman Jeffrey διδάσκει τα μαθήματα «Βάσεις Ι», «Βάσεις ΙΙ».

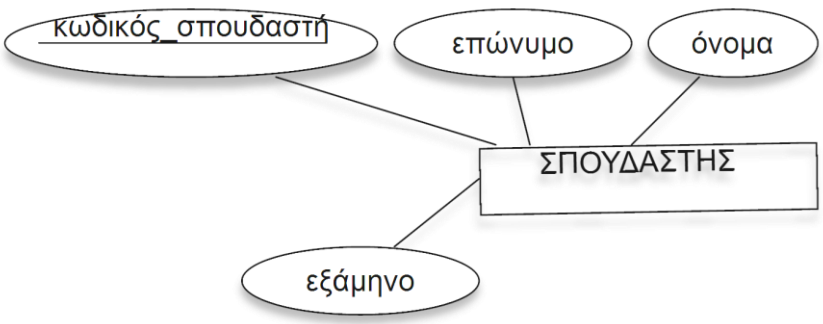
Να και τα συμπεράσματά μας:

Οντότητες είναι ο Σπουδαστής Κυριακόπουλος, ο Σπουδαστής Παπαπέτρου, η σπουδάστρια Αποστόλου, ο καθηγητής Ullman, τα μαθήματα «Βάσεις Ι», «Βάσεις ΙΙ». Τύποι οντότητας είναι: ΣΠΟΥΔΑΣΤΗΣ, ΚΑΘΗΓΗΤΗΣ, ΜΑΘΗΜΑ. Χαρακτηριστικά του τύπου οντότητας «ΣΠΟΥΔΑΣΤΗΣ» μπορούν να είναι το επώνυμο, το όνομα, ο αριθμός μητρώου, το Εξάμηνο, η διεύθυνση.

Βλέπουμε ότι όταν γράφουμε τις Οντότητες και τα Χαρακτηριστικά χρησιμοποιούμε ουσιαστικά. Το μοναδικό Χαρακτηριστικό «αριθμός μητρώου» είναι κλειδί της οντότητας Σπουδαστής. Για τη Συσχέτιση «διδάσκει» χρησιμοποιούμε ρήμα.

1.13.2 Πως σχεδιάζουμε ΜΟΣ. Μία περιγραφική προσέγγιση βήμα προς βήμα

Θα αρχίσουμε τη σχεδίαση από τον τύπο οντότητας «Σπουδαστής». Στην Εικόνα 1.44 σχεδιάσαμε τον τύπο οντότητας «ΣΠΟΥΔΑΣΤΗΣ» που έχει Χαρακτηριστικά «επώνυμο», «όνομα», «εξάμηνο». Το Κλειδί Οντότητας είναι το χαρακτηριστικό «κωδικός_σπουδαστή». Παρατηρήστε ότι το κλειδί της οντότητας είναι υπογραμμισμένο.



STUDENTS(SURNAME, NAME, STUDENT_NO, SEMESTER)

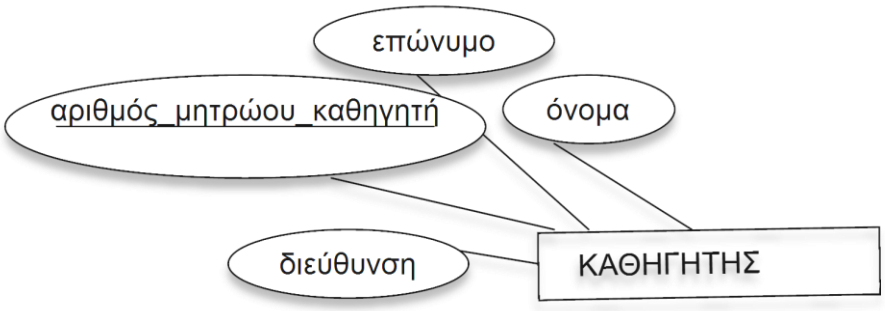
SURNAME	NAME	STUDENT_NO	SEMESTER
ΚΥΡΙΑΚΟΠΟΥΛΟΣ	ΝΙΚΗΦΟΡΟΣ	213	Δ
ΑΠΟΣΤΟΛΟΥ	ΖΩΗ	816	Α
ΠΑΠΑΠΕΤΡΟΥ	ΝΙΚΟΛΑΟΣ	450	Β
ΖΕΥΓΑΡΙΔΗΣ	ΟΡΕΣΤΗΣ	346	Γ
ΚΟΤΑΜΑΝΙΔΟΥ	ΕΙΡΗΝΗ	610	Α

Πίνακας Student Primary key=student_no

Εικόνα 1.44 Τύπος οντότητας σπουδαστή και αντίστοιχος πίνακας με οντότητες σπουδαστών

Όταν ολοκληρωθεί η σχεδίαση του ΜΟΣ θα κατασκευάσουμε έναν πίνακα της βάσης που αντιστοιχεί στον τύπο οντότητας. Ο πίνακας θα αποθηκεύει όλες τις οντότητες σπουδαστών. Ο πίνακας της βάσης δεδομένων Student, που αντιστοιχεί στον τύπο οντότητας «ΣΠΟΥΔΑΣΤΗΣ» που σχεδιάσαμε στο ΜΟΣ, αποθηκεύει όλες τις οντότητες σπουδαστών. Μπορούμε να γράψουμε το όνομα του πίνακα και τα ονόματα των στηλών του στα αγγλικά ή με λατινικούς χαρακτήρες, ακριβέστερα.

Στην Εικόνα 1.45 σχεδιάζουμε τον τύπο οντότητας «ΚΑΘΗΓΗΤΗΣ» που έχει χαρακτηριστικά «επώνυμο», «όνομα», «διεύθυνση». Το Κλειδί Οντότητας (entity key) είναι το χαρακτηριστικό «αριθμός_μητρώου_καθηγητή». Παρατηρήστε ότι το κλειδί της οντότητας είναι υπογραμμισμένο. Από τον τύπο οντότητας «Καθηγητής» προκύπτει ο Πίνακας Teacher



Από τον τύπο οντότητας «Καθηγητής» προκύπτει ο Πίνακας Teacher

SURNAME	NAME	ADDRESS	TEACHER_NO
CODD	TED	MASS	10
ULLMAN	JEFFREY	CALIF	20
WIDOM	JENNIFER	CALIF	30
ELMASRI	RAMEZ	MASS	40
NAVATHE	SHAMKANT	MASS	50

Πίνακας Teacher Primary key=teacher_no

Εικόνα 1.45 Τύπος οντότητας καθηγητή και αντίστοιχος πίνακας με οντότητες καθηγητών

Προσοχή! Στις περιπτώσεις των σχημάτων 1.44 και 1.45, σε κάθε τύπο οντότητας αντιστοιχούμε έναν πίνακα. Ανάλογα με τους τύπους συσχετίσεων στους οποίους εμπλέκεται ο τύπος οντότητας ο πίνακας της οντότητας μπορεί να εμπλουτιστεί.

1.13.3 Παράδειγμα βήμα-προς-βήμα σχεδίασης μοντέλου οντοτήτων συσχετίσεων εκπαιδευτικής βάσης δεδομένων

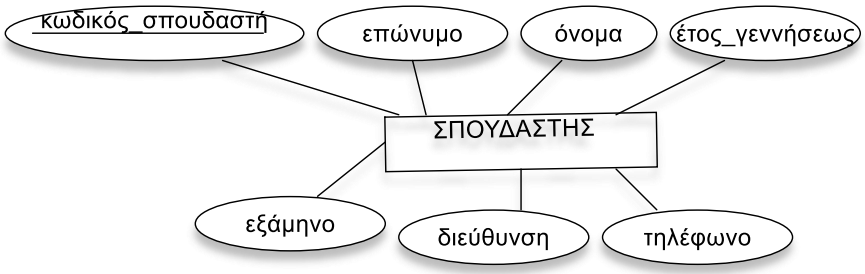
Αρχίζουμε με σύντομη περιγραφή του παραδείγματος. Μας αναθέτουν να κατασκευάσουμε ένα σύστημα σχεσιακής βάσης δεδομένων για τη γραμματεία του τμήματος Μηχανικών Πληροφορικής και Υπολογιστών. Στους πίνακες που θα κατασκευάσουμε θα μπορεί το προσωπικό της γραμματείας να καταγράφει τα στοιχεία των σπουδαστών, των καθηγητών και των μαθημάτων. Επιπλέον, το προσωπικό θα μπορεί να διαχειρίζεται τις αναθέσεις διδασκαλίας, δηλαδή να καταγράφει ποια μαθήματα διδάσκει κάθε καθηγητής του τμήματος κ.λπ. Επιπλέον, θα πρέπει να διαχειρίζεται τις εγγραφές των φοιτητών στα μαθήματα. Τέλος, οι υπάλληλοι της Γραμματείας πρέπει να μπορούν να βλέπουν τα στοιχεία που υπάρχουν στις πίνακες της βάσης δεδομένων.

Στην ενότητα αυτή θα σχεδιαστεί βήμα-βήμα το μοντέλο.

Σχεδίαση των οντοτήτων

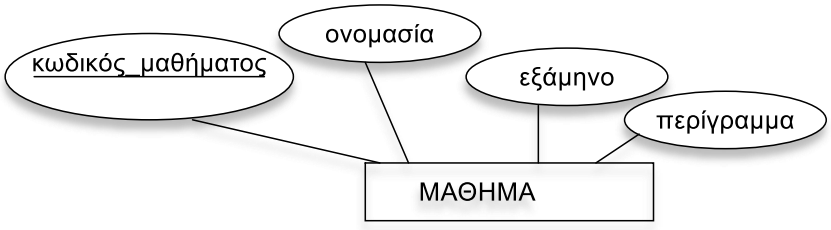
Στην Εικόνα 1.46 βλέπουμε όλους τους τύπους οντοτήτων.

Τύπος οντότητας **Σπουδαστής** και τα χαρακτηριστικά του!



Τύπος οντότητας **Καθηγητής** και τα χαρακτηριστικά του!

Τύπος οντότητας **Μάθημα** και τα χαρακτηριστικά του!



Εικόνα 1.46 Τύποι οντοτήτων φοιτητή, καθηγητή κσι μαθήματος

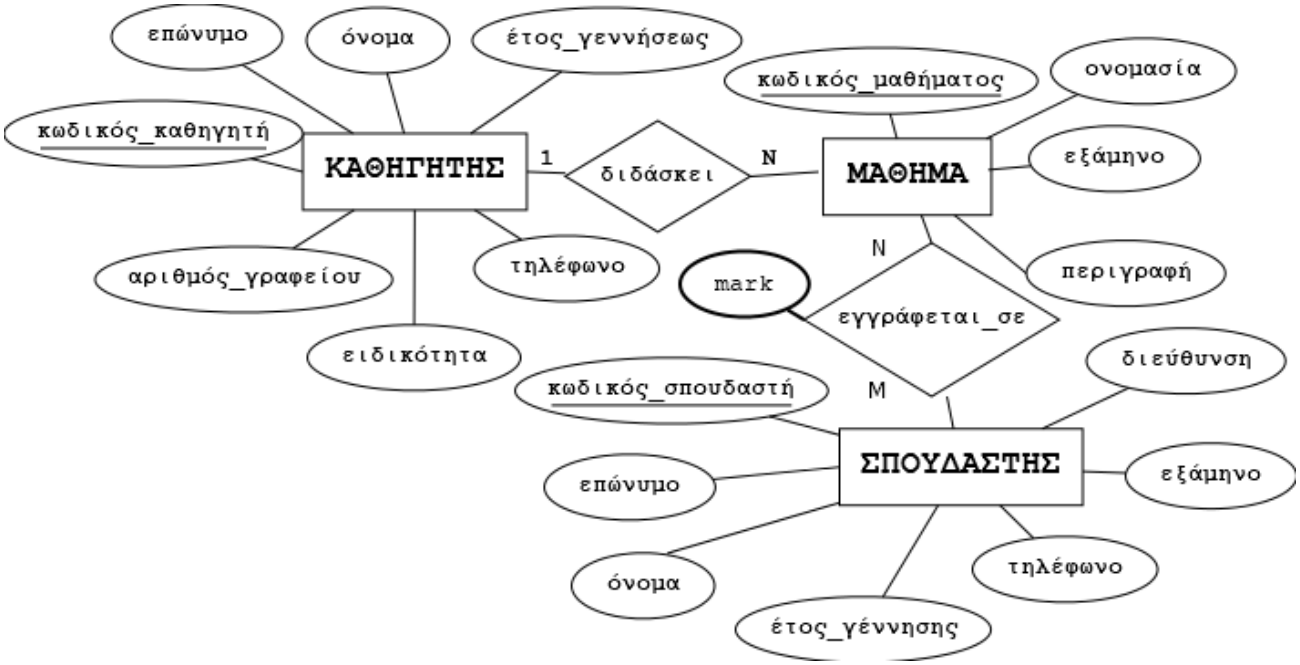
Σχεδίαση των συσχετίσεων

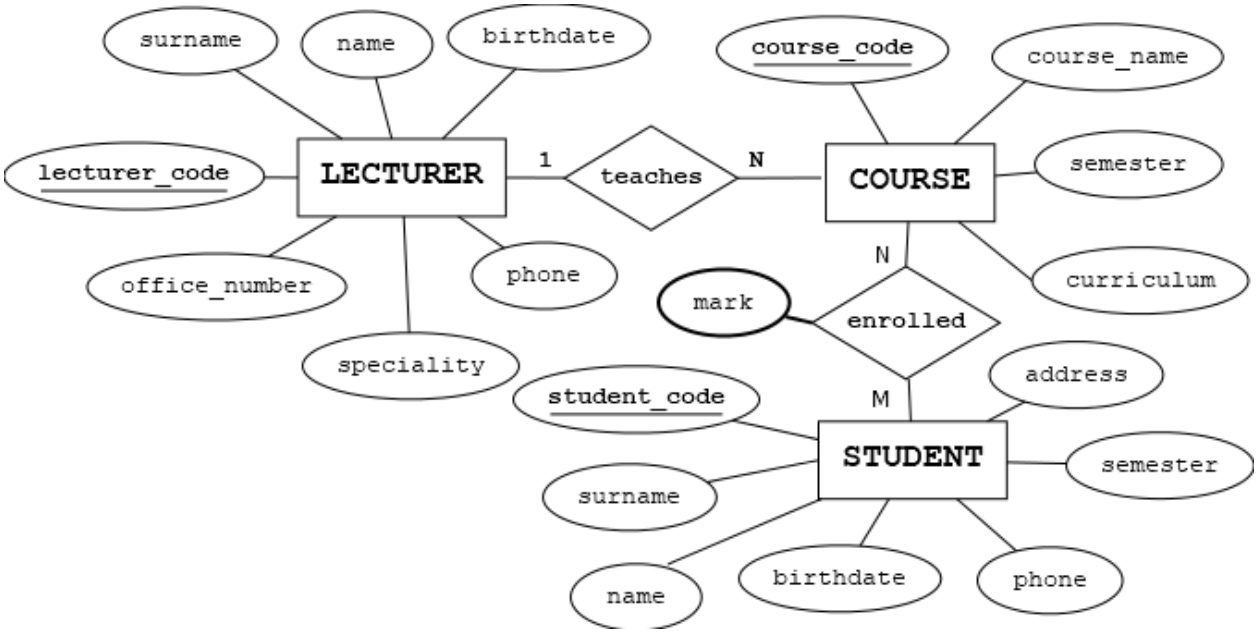
Για να ολοκληρώσουμε το ΜΟΣ πρέπει να σχεδιάσουμε και τις συσχετίσεις ανάμεσα στους τύπους οντοτήτων. Κάθε συσχέτιση έχει ένα βαθμό ανάλογο του αριθμού οντοτήτων που συνδέει (συσχετίζει)! Για παράδειγμα η συσχέτιση «διδάσκει» είναι δυαδική. Για κάθε συσχέτιση στο ΜΟΣ πρέπει να γράφουμε και τον βαθμό της συσχέτισης. Για παράδειγμα η «διδάσκει» θα μπορούσε να είναι «1-προς-Πολλά» (γράφεται και «1:N») ή «Πολλά-προς-Πολλά» (γράφεται και «M:N»).

Βαθμός Συσχέτισης

Βαθμός μιας συσχέτισης ονομάζεται ο αριθμός των οντοτήτων που συνδέει η συσχέτιση. Συνήθως οι συσχετίσεις μεταξύ δύο οντοτήτων (δυαδικές συσχετίσεις) επαρκούν για τις ανάγκες μεγάλου μέρους της εφαρμογής. Υπάρχουν περιπτώσεις, όμως, όπου τρεις ή περισσότερες οντότητες πρέπει να συνδεθούν με μια συσχέτιση ή μία συσχέτιση να οριστεί πάνω σε οντότητες και συσχετίσεις.

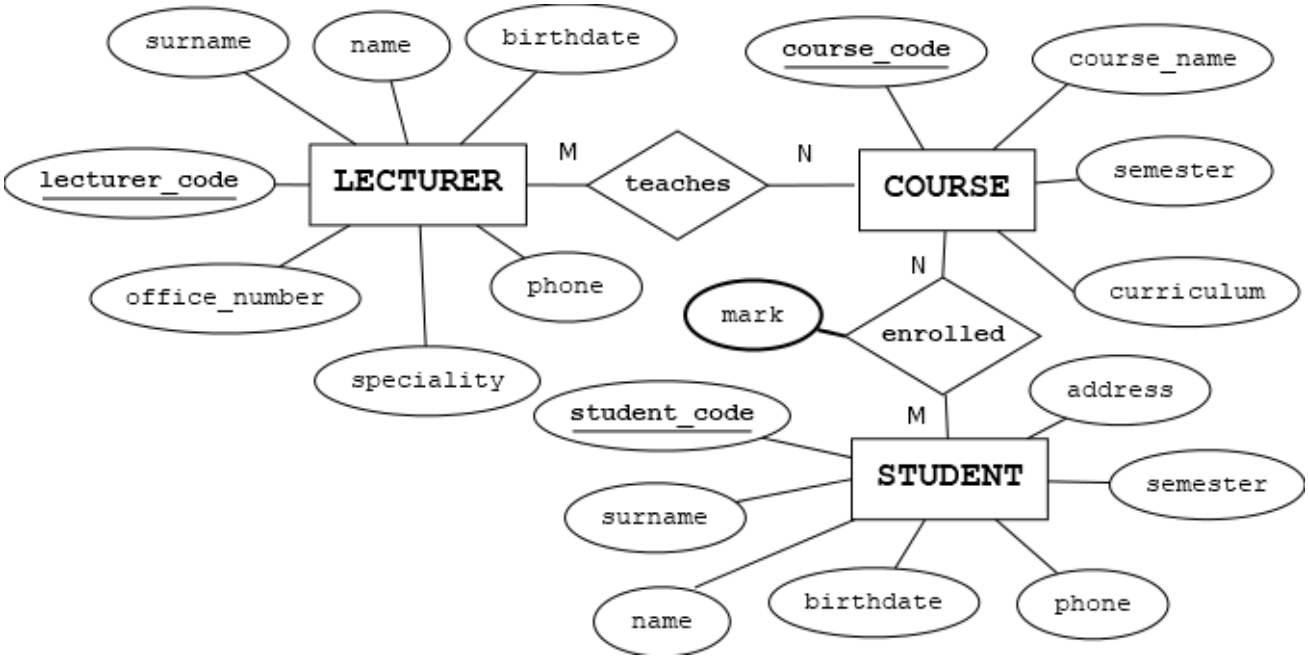
Ολοκληρώνουμε, στη συνέχεια, τη σχεδίαση του ΜΟΣ. Αρχικά σχεδιάζουμε τη συσχέτιση «διδάσκει». Στην Εικόνα 1.47 θεωρούμε ότι η συσχέτιση έχει τύπο ένα-προς-πολλά (1:N).





Εικόνα 1.47 ΜΟΣ σε ελληνικά και αγγλικά της εκπαιδευτικής βάσης δεδομένων. Η συσχέτιση «διδάσκει» είναι τύπου ένα-προς-πολλά

(Στη συνέχεια θεωρούμε ότι η συσχέτιση είναι πολλά-προς-πολλά (M:N) και ανασχεδιάζουμε το μοντέλο (Εικόνα 1.48).



Εικόνα 1.48 ΜΟΣ σε ελληνικά και αγγλικά της εκπαιδευτικής βάσης δεδομένων. Η συσχέτιση «διδάσκει» είναι τύπου πολλά-προς-πολλά

1.14 Πως από το μοντέλο (διάγραμμα) οντοτήτων συσχετίσεων (ΜΟΣ) κατασκευάζουμε το σχεσιακό μοντέλο (relational model)–πρώτη προσέγγιση

Σύμβαση

Στη βιβλιογραφία είναι συνηθισμένο όταν μιλάμε για τύπο οντότητας να χρησιμοποιούμε τον όρο οντότητα και όταν μιλάμε για τον τύπο συσχέτισης να χρησιμοποιούμε τον όρο συσχέτιση. Έτσι μιλάμε για Μοντέλο Οντοτήτων Συσχετίσεων. Στη συνέχεια χρησιμοποιούμε αυτήν τη σύμβαση.

Για τη μετάβαση από το μοντέλο οντοτήτων - συσχετίσεων σε πίνακες ακολουθήστε τους εξής κανόνες:

Κανόνας 1: Για κάθε οντότητα (τύπο οντότητας) Alpha(K1), όπου K1 το κλειδί της οντότητας, θα κατασκευάσετε έναν πίνακα που θα περιλαμβάνει σαν στήλες τουλάχιστον όλα τα χαρακτηριστικά (attributes) της. Το κύριο κλειδί της οντότητας, απλό ή σύνθετο, θα είναι και κύριο κλειδί του πίνακα που θα αναπαριστά την οντότητα.

Alpha

K1	Column11	…	Column1N
Primary key			

Κανόνας 2: Έστω μια συσχέτιση σ:Alpha(K1)-->Beta(K2), όπου Alpha(K1), Beta(K2) οντότητες και K1, K2 τα κύρια κλειδιά των οντοτήτων. Αν η συσχέτιση σ είναι τύπου 1:N τότε για τη συσχέτιση σ δεν κατασκευάζετε ξεχωριστό πίνακα. Απλά προσθέτετε στις στήλες του πίνακα Beta (που αντιστοιχούν στα χαρακτηριστικά της οντότητας Beta(K2)) το K1 σαν ξένο κλειδί. Αν η συσχέτιση σ έχει χαρακτηριστικό τότε και τα χαρακτηριστικά αυτά γίνονται στήλες του πίνακα Beta.

Ακολουθεί επεξήγηση

Για τις οντότητες Alpha και Beta, λόγω του Κανόνα 1, κατασκευάζουμε δύο πίνακες.

Alpha

K1	Column11	…	Column1M
Primary key			

Beta

K2	Column21	…	Column2N
Primary key			

Η συσχέτιση σ:Alpha(K1)-->Beta(K2) είναι τύπου 1:N. Επομένως, ο πίνακας Beta αλλάζει.

Beta

K2	Column21	…	Column2N	K1
Primary key				Foreign Key

Κανόνας 3: Αν η συσχέτιση σ είναι τύπου 1:1 τότε (για τη συσχέτιση σ) δεν κατασκευάζουμε ξεχωριστό πίνακα. Να τι κάνουμε:

Προσθέτουμε στις στήλες του πίνακα Beta (που αντιστοιχούν στα χαρακτηριστικά της οντότητας Beta(K2)) το K1 σαν ξένο κλειδί ή προσθέτουμε στις στήλες του πίνακα Alpha (που αντιστοιχούν στα χαρακτηριστικά της οντότητας Alpha(K1)) το K2 σαν ξένο κλειδί. **Ποτέ και τα δύο!**

Συμπέρασμα

Θεωρούμε ότι η συσχέτιση σ:Alpha(K1)-->Beta(K2) είναι τύπου 1:N, άρα:

Alpha

K1	Column11	…	Column1M
Primary key			

Beta

K2	Column21	…	Column2N	K1
Primary key				Foreign Key

Εναλλακτικά, θεωρούμε ότι η συσχέτιση σ:Alpha(K1)-->Beta(K2) είναι τύπου N:1, άρα:

Alpha

K1	Column11	…	Column1M	K2
Primary key				Foreign Key

Beta

K2	Column21	…	Column2N
Primary key			

Κανόνας 4: Αν η συσχέτιση είναι M:N τότε κατασκευάζεις ξεχωριστό πίνακα που περιλαμβάνει τα K1, K2, ως ξένα κλειδιά, το (K1, K2) ως σύνθετο κύριο κλειδί και επιπλέον τα χαρακτηριστικά της συσχέτισης, άν υπάρχουν τέτοια χαρακτηριστικά.

Alpha

K1	Column11	…	Column1M
Primary key			

Beta

K2	Column21	…	Column2N
Primary key			

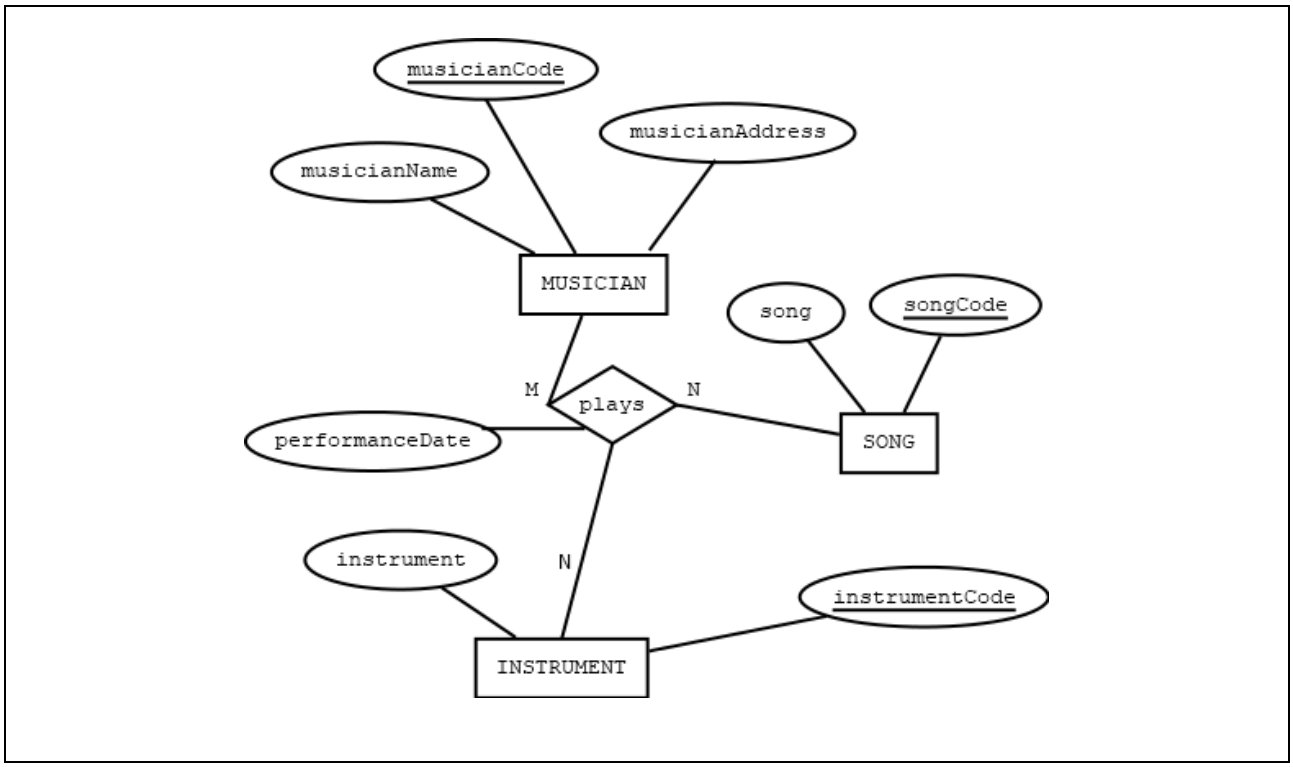
Alpha_Beta

K1	K2	Column1of the relationship	…
Foreign key	Foreign key		

Primary key=(K1, K2)

Κανόνας 5: Αν μια συσχέτιση συνδέει παραπάνω από δύο οντότητες π.χ. τις οντότητες A(K1), A(K2), A(K3) με K1, K2, K3 κύρια κλειδιά οντοτήτων αντίστοιχα, τότε για τη συσχέτιση αυτή, **αν είναι M:N:N (πολλά-προς-πολλά-προς-πολλά)**, κατασκευάζουμε ξεχωριστό πίνακα με κύριο κλειδί, συνήθως, (K1, K2, K3).

Ακολουθεί παράδειγμα εφαρμογής των κανόνων σε ΜΟΣ. Στην Εικόνα 1.49 βλέπουμε τις οντότητες μουσικού, μουσικού οργάνου και τραγουδιού και τη συσχέτιση plays που έχει τύπο πολλά-προς-πολλά-προς-πολλά («ο μουσικός Α παίζει το όργανο Β στο τραγούδι Γ»). Επιπλέον, βλέπουμε και τους αντίστοιχους πίνακες.



Musician

musicianCode	musicianName	MusicianAddress
Primary Key		

Instrument

instrumentCode	instrument
Primary Key	

Song

SongCode	Song
Primary Key	

Plays

musicianCode	instrumentCode	SongCode	performanceDate
Foreign Key	Foreign Key	Foreign Key	

Primary Key=(musicianCode, instrumentCode, SongCode).

Εικόνα 1.49 ΜΟΣ που περιλαμβάνει οντότητες μουσικού, μουσικού οργάνου και τραγουδιού και συσχέτιση plays με τύπο πολλά-προς-πολλά-προς-πολλα. Μεταγραφή σε σχεσιακή βάση δεδομένων

Εναλλακτικά, θα μπορούσαμε να προσθέσουμε ως κύριο κλειδί έναν αύξοντα αριθμό (στήλη accno). Στην περίπτωση αυτή θα πρέπει να προστεθεί στη συσχέτιση plays το χαρακτηριστικό accno υπογραμμισμένο!

1.14.1 Παράδειγμα εφαρμογής των κανόνων μεταγραφής του ΜΟΣ της εκπαιδευτικής βάσης δεδομένων σε σχεσιακή βάση δεδομένων.

Κάθε τύπος οντότητας γίνεται πίνακας με κλειδί το κλειδί της οντότητας. Παραθέτουμε τους πίνακες τους αντίστοιχους των τύπων οντοτήτων:

- lecturer(lecturer_code, surname, name, birthdate, speciality, office_number, phone)
- course(course_code, course_name, semester, content)
- student(student_code, surname, name, birthdate, semester, address, phone)

Στη συνέχεια θα εξετάσουμε τις συσχετίσεις μεταξύ οντοτήτων και τότε οι πίνακες αυτοί μπορεί να αλλάξουν.

Μια συσχέτιση «Ένα-προς-Πολλά» ή «Πολλά-προς-Ένα» δεν γίνεται πίνακας αλλά υλοποιείται προσθέτοντας το κλειδί της οντότητας «Ένα» στον πίνακα της οντότητας που συμμετέχει με τα «Πολλά».

- Η συσχέτιση “teaches” έχει τύπο «1:N». Άρα ο πίνακας lecturer δε θα αλλάξει αλλά θα πρέπει να αλλάξει ο πίνακας course.
- Επειδή στον πίνακα course θα προσθέσω το κλειδί του lecturer τότε ο πίνακας από
  course(course_code, course_name, semester, content) γίνεται
  course(course_code, course_name, semester, content, lecturer_code)
- Να και οι υπόλοιποι πίνακες.
  lecturer(lecturer_code, surname, name, birthdate, speciality, office_number, phone)
  student(student_code, surname, name, birthdate, semester, address, phone)

Μια συσχέτιση «Πολλά-προς-Πολλά» γίνεται πίνακας με σύνθετο κλειδί τα κλειδιά των οντοτήτων που συνδέει.

- Η συσχέτιση “enrolls” έχει τύπο «M:N». Άρα θα χρειαστεί να κατασκευάσουμε έναν πίνακα enrollments.
- Να οι πίνακες της βάσης δεδομένων.

  course(course_code, course_name, semester, content, lecturer_code)

  lecturer(lecturer_code, surname, name, birthdate, speciality, office_number, phone)

  student(student_code, surname, name, birthdate, semester, address, phone)

  enrollments(student_code, course_code)

1.15 Εισαγωγή στη σχεδίαση μιας βάσης δεδομένων. Ευρετικοί κανόνες.

Η σχεδίαση Βάσης Δεδομένων θα μας απασχολήσει αρκετά στο βιβλίο αυτό. Κατά συνέπεια, στην ενότητα αυτή θα προσπαθήσουμε να διεκπεραιώσουμε μία πρώτη συζήτηση του θέματος περιορίζοντας λίγο και το αντικείμενό μας. Πιο συγκεκριμένα, θα θεωρήσουμε ότι μας ενδιαφέρει η ανάπτυξη σχετικά απλών εφαρμογών διαχείρισης μικρών βάσεων και με χρήση προϊόντος σε περιβάλλον προσωπικών υπολογιστών (π.χ., χρήση MySQL, ORACLE Express) έτσι ώστε να μην ασχοληθούμε με όλες τις πλευρές και τις λεπτομέρειες της σχεδίασης. Παρ’ όλα αυτά η όλη συζήτηση και τα συμπεράσματά μας θα έχουν μία γενικότερη ισχύ.

Η δημιουργία της Βάσης Δεδομένων αρχίζει με την καλή σχεδίαση. Αυτό αποτελεί μια βασική αρχή σε όλες τις περιπτώσεις τεχνολογικών εφαρμογών. Ακόμη και από την καθημερινή μας εμπειρία γνωρίζουμε οτι δεν μπορεί να ξεκινήσει μια οικοδομή χωρίς σχέδιο. Επομένως, θα πρέπει να περιγράψουμε τι σημαίνει καλή σχεδίαση.

1.15.1 Κριτήρια καλής σχεδίασης

Τα κριτήρια της καλής σχεδίασης μιας Βάσης μπορούν να συνοψιστούν ως εξής:

- Ο χρόνος ανάκτησης όταν αναζητούμε γραμμές (εγγραφές) δεδομένων είναι μικρός
- Η αποθήκευση των δεδομένων γίνεται, κατά το δυνατόν, με τον πιο αποδοτικό τρόπο ώστε να κρατήσουμε μικρή τη βάση δεδομένων
- Οι τροποποιήσεις (μεταβολές) των στοιχείων διευκολύνονται.
- Η σχεδίαση είναι αρκετά ευέλικτη ώστε να επιτρέπει εύκολη επέκταση ή προσθήκη νέων προγραμμάτων εφαρμογών. Επιπλέον, διευκολύνεται η υποστήριξη νέων λειτουργιών που θα απαιτηθούν στο μέλλον.

1.15.2 Αντικειμενικοί στόχοι της σχεδίασης

Κατά τη σχεδίαση της Βάσης πρέπει να έχουμε κατά νουν μια σειρά από στόχους, οι οποίοι είναι επιθυμητό να ικανοποιηθούν. Βέβαια κάποιοι από αυτούς δεν είναι τελείως συμβατοί μεταξύ τους ή και αποκλείονται αμοιβαία.

- Ελαχιστοποίηση των δεδομένων που επαναλαμβάνονται και κατά συνέπεια πλεονάζουν.
- Ετοιμότητα γρήγορης ανάκτησης κάποιων δεδομένων (εγγραφών)
- Διευκόλυνση της προγραμματιστικής δουλειάς για προσθήκη και αλλαγές στη βάση δεδομένων
- Εύκολη συντήρηση της βάσης.

1.15.3 Κύριες ενέργειες της σχεδίασης

Η καλή σχεδίαση μιας εφαρμογής λογισμικού βάσης δεδομένων περιλαμβάνει τις παρακάτω κύριες δραστηριότητες:

- Ανάλυση και καθορισμός των δεδομένων που είναι απαραίτητα στην εφαρμογή και μοντελοποίηση
- Οργάνωση των δεδομένων σε πίνακες και ο καθορισμός των συσχετίσεων ανάμεσα στους πίνακες
- Καθορισμός ευρετηρίων και απαιτήσεων επικύρωσης (validation requirements) δεδομένων
- Δημιουργία και αποθήκευση των αναγκαίων ερωτήσεων (queries) που ενδιαφέρουν τους χρήστες. Αυτή η δραστηριότητα θα μας απασχολήσει στο κεφάλαιο 3 (γλώσσα SQL)

1.15.4 Μοντελοποίηση της εφαρμογής βάσης δεδομένων

Όταν θέλετε να μοντελοποιήσετε μια εφαρμογή το πρώτο πράγμα που θα πρέπει να κάνετε είναι να ορίσετε όλες τις εργασίες που θα υποστηρίξει η εφαρμογή σας. Εργαζόμενοι για τον καθορισμό των εργασιών που θα πρέπει να διεκπεραιώνει η εφαρμογή σας δημιουργείτε ένα έγγραφο "λειτουργικών προδιαγραφών" (functional specifications) της εφαρμογής. Το έγγραφο αυτό είναι χρήσιμο σε σας για να εστιάζετε κάθε στιγμή στο τι θέλετε να κάνει το πρόγραμμά σας αλλά είναι σίγουρα και απαραίτητο σαν ένα έγγραφο συμφωνίας από τη στιγμή που δημιουργείτε το πρόγραμμα για κάποιο φυσικό πρόσωπο ή εταιρεία. Επειδή δημιουργείτε το πρόγραμμα για κάποιους άλλους ο καλύτερος τρόπος για να καταλάβετε ποιες εργασίες θα πρέπει να υλοποιήσετε είναι να συζητήσετε μαζί τους, να εξετάσετε αν υπάρχει κάποιο σύστημα που πρέπει να αντικατασταθεί, αν υπάρχουν κάποια εκτυπωτικά (reports) που πρέπει να κατασκευαστούν. Υποβάλετε πολλές ερωτήσεις μέχρι να κατανοήσετε ποιοι είναι οι στόχοι και το αντικείμενο της εφαρμογής για τους ανθρώπους που θα την χρησιμοποιήσουν.

1.15.5 Καθορισμός των απαιτούμενων δεδομένων της εφαρμογής σας

Μετά τον καθορισμό των λειτουργικών προδιαγραφών μπορείτε να αρχίσετε να προσδιορίζετε με περισσότερη λεπτομέρεια τα δεδομένα που χρειάζονται τα προγράμματα σας.

Παράδειγμα

Έστω ότι θέλετε να κάνετε μοντελοποίηση μιας λίστας των μελών του συλλόγου γονέων και κηδεμόνων του σχολείου από το οποίο αποφοιτήσατε. Είναι προφανές οτι η μοντελοποίηση θα πρέπει να συμπεριλάβει τη δημιουργία τηλεφωνικών ευρετηρίων, ταχυδρομικής λίστας για τα μέλη του συλλόγου κ.λπ. Η διαπίστωση ότι θα εργαστείτε με τηλεφωνικά ευρετήρια και ταχυδρομικές λίστες σας οδηγεί στο συμπέρασμα ότι η βάση δεδομένων θα περιλαμβάνει στοιχεία όπως αριθμός τηλεφώνου και διεύθυνση κάθε μέλους. Το μοντέλο μπορεί να περιλαμβάνει και κάποια άλλα αντικείμενα της βάσης. Για παράδειγμα, είναι λογικό να συμπεριληφθεί ένα ευρετήριο για τον ταχυδρομικό κώδικα ή και να αποθηκευτεί μια ερώτηση (query) που θα ταξινομεί τα στοιχεία των μελών ως προς ταχυδρομικό κώδικα.

1.15.6 Οργάνωση των δεδομένων

Στην οργάνωση δεδομένων θα πρέπει να αποφασίσετε και στη συνέχεια να περιγράψετε πως τα δεδομένα θα αποθηκευτούν σε πολλούς πίνακες της βάσης δεδομένων. Στην επόμενη δραστηριότητα θα συμπληρώσετε την εργασία αυτή αποφασίζοντας πως θα συσχετίσετε τους πίνακες αυτούς ώστε να διευκολύνετε την αναζήτηση στοιχείων και την ενημέρωση της βάσης.

1.15.6.1 Πίνακες σαν συλλογές πληροφορίας σε συγκεκριμένο θέμα (Topics)

Στις βάσεις δεδομένων ένας πίνακας είναι συνήθως μια συλλογή πληροφοριών που είναι σχετικές με ή περιγράφουν ένα πρόσωπο (π.χ., υπάλληλο, προμηθευτή) ή ένα αντικείμενο (π.χ., ένα προϊόν) ή ένα θέμα (π.χ.,. την εγγραφή ενός σπουδαστή). Στην περίπτωση ενός πίνακα θα πρέπει να έχουμε κατα νούν ένα κριτήριο που να καθορίζει αν ένα δεδομένο ταιριάζει ή όχι με τα υπόλοιπα δεδομένα του πίνακα ώστε να μπορούμε να αποφασίσουμε αν θα το συμπεριλάβουμε ή όχι.

Παράδειγμα

Θα εξετάσουμε την περίπτωση μιας σχολικής Βάσης Δεδομένων. Υποθέτουμε ότι αποφασίζουμε αρχικά να οργανώσουμε τα δεδομένα των καθηγητών και των μαθητών στον ίδιο πίνακα επειδή κάθε πρόσωπο, καθηγητής ή μαθητής, έχει επώνυμο, όνομα, διεύθυνση, τηλέφωνο κ.λπ.. Στη συνέχεια μελετώντας τα στοιχεία καταλήγουμε στο συμπέρασμα ότι υπάρχουν στοιχεία όπως η ειδικότητα, ο μισθός και η παρακράτηση φόρου για τους καθηγητές που δεν ταιριάζουν στους μαθητές. Ισχύει βέβαια και το αντίστροφο, κάποια στοιχεία των μαθητών που δεν έχει έννοια να τα αποθηκεύουμε για τους καθηγητές. Επομένως, αν θα γράφουμε όλα τα παραπάνω δεδομένα σε έναν πίνακα μοιραία θα οδηγηθούμε σε μια σπατάλη χώρου επειδή πολλά στοιχεία δε θα συμπληρώνονται για διάφορες γραμμές (δείτε και Εικόνα 1.50).

Βάση δεδομένων γονέων και κηδεμόνων

ΑΔΑΜΟΣ	ΑΛΕΞΑΝΔΡΟΣ	ΘΗΣΕΩΣ 10	ΦΥΣΙΚΟΣ		
ΒΑΣΙΛΕΙΟΥ	ΠΑΡΑΣΚΕΥΗ	ΑΡΙΑΔΝΗΣ 20		ΑΝΤΩΝΙΟΣ	ΒΑΣΙΛΕΙΟΥ
ΒΑΦΕΙΑΔΗΣ	ΓΕΩΡΓΙΟΣ	ΙΑΣΟΝΟΣ 8		ΠΑΝΑΓΙΩΤΗΣ	ΒΑΦΕΙΑΔΗΣ
ΝΙΚΟΥ	ΗΛΙΑΣ	ΗΡΑΚΛΕΟΥΣ 5	ΦΙΛΟΛΟΓΟΣ		
ΠΑΠΑΔΟΥΚΑ	ΜΑΡΙΑ	ΤΡΟΙΑΣ 70		ΠΑΝΑΓΙΩΤΗΣ	ΠΑΠΑΔΟΥΚΑ
ΣΦΑΚΑΚΗΣ	ΜΙΧΑΗΛ	ΚΡΗΤΗΣ 50		ΔΗΜΗΤΡΙΟΣ	ΣΦΑΚΑΚΗΣ

Εικόνα 1.50 Η αποθήκευση στοιχείων μαθητών και καθηγητών στον ίδιο πίνακα οδηγεί σε σπατάλη χώρου (δείτε κενές στήλες) επειδή τα στοιχεία αυτά διαφέρουν

Η αναζήτηση στοιχείων με τα προγράμματα μας θα καθυστερεί, επίσης, επειδή όταν αναζητούμε στοιχεία καθηγητών θα πρέπει το πρόγραμμα να «υπερπηδά» τα στοιχεία των μαθητών που δεν ενδιαφέρουν. Επομένως, στο παράδειγμά μας, η σχεδίαση της εικόνας 1.51 φαίνεται να είναι καλύτερη.

Πίνακας Καθηγητή

Επώνυμο	Όνομα	Διεύθυνση	Ειδικότητα
ΑΔΑΜΟΣ	ΑΛΕΞΑΝΔΡΟΣ	ΘΗΣΕΩΣ 10	ΦΥΣΙΚΟΣ
ΝΙΚΟΥ	ΗΛΙΑΣ	ΗΡΑΚΛΕΟΥΣ 5	ΦΙΛΟΛΟΓΟΣ

Πίνακας Μαθητή

Επώνυμο	Όνομα	Διεύθυνση	Όνομα κηδεμόνα	Επώνυμο κηδεμόνα
ΒΑΣΙΛΕΙΟΥ	ΠΑΡΑΣΚΕΥΗ	ΑΡΙΑΔΝΗΣ 20	ΑΝΤΩΝΙΟΣ	ΒΑΣΙΛΕΙΟΥ
ΒΑΦΕΙΑΔΗΣ	ΓΕΩΡΓΙΟΣ	ΙΑΣΟΝΟΣ 8	ΠΑΝΑΓΙΩΤΗΣ	ΒΑΦΕΙΑΔΗΣ
ΠΑΠΑΔΟΥΚΑ	ΜΑΡΙΑ	ΤΡΟΙΑΣ 70	ΠΑΝΑΓΙΩΤΗΣ	ΠΑΠΑΔΟΥΚΑ
ΣΦΑΚΑΚΗΣ	ΜΙΧΑΗΛ	ΚΡΗΤΗΣ 50	ΔΗΜΗΤΡΙΟΣ	ΣΦΑΚΑΚΗΣ

Εικόνα 1.51 Δύο διαφορετικοί πίνακες για τους καθηγητές και τους μαθητές περιλαμβάνουν μόνο τις κατάλληλες στήλες και είναι αποδοτικότεροι

1.15.7 Καθορισμός των συσχετίσεων ανάμεσα στους πίνακες

Ο κύριος λόγος που επιβάλει τη συσχέτιση των πινάκων είναι η ανάγκη που έχουμε στις εφαρμογές να βλέπουμε όσο γίνεται πιο γρήγορα τα στοιχεία διαφόρων πινάκων που είναι λογικά συνδεδεμένα. Αυτό μας χρειάζεται ιδιαίτερα όταν θέλουμε να επεξεργαζόμαστε και να δείχνουμε ταυτόχρονα στοιχεία από δύο ή περισσότερους πίνακες. Ένας ενδιαφέρων μηχανισμός συσχετίσεων των πινάκων μπορεί να βασίζεται στα κύρια και τα ξένα κλειδιά.

1.15.7.1 Παράδειγμα

Ο πίνακας "Προμηθευτής" έχει μία στήλη Κωδικός_Προμηθευτή. Ο πίνακας "Εφοδιασμός" έχει επίσης στήλη Κωδικός_Προμηθευτή. Αυτοί οι δύο πίνακες λέμε ότι είναι συνδεδεμένοι ως προς αυτές τις στήλες. Αν ένα πρόγραμμα την ώρα που επεξεργάζεται τον πίνακα "Εφοδιασμός" χρειάζεται τα στοιχεία προμηθευτή που έκανε κάποιο συγκεκριμένο εφοδιασμό τότε η γραμμή του προμηθευτή ανακτάται ταχύτατα από τον άλλο πίνακα επειδή γνωρίζουμε την τιμή της στήλης Κωδικός_Προμηθευτή.

1.15.7.2 Παράδειγμα

Ένας άλλος τρόπος να αποφύγουμε πλεονασμούς και να αυξήσουμε την εγκυρότητα των δεδομένων είναι να χρησιμοποιήσουμε όπου είναι δυνατόν πίνακες αναζήτησης τιμών (Lookup Tables). Τυπικά ένας πίνακας αυτού του τύπου χρησιμοποιείται για την αποθήκευση έγκυρων τιμών.

Ένας πίνακας Ξένη_Γλώσσα περιλαμβάνει τις στήλες συντομογραφίας της γλώσσας και τη γλώσσα. Όταν θέλουμε να εισάγουμε σε κάποιο πίνακα με στοιχεία καθηγητή τις ξένες γλώσσες που γνωρίζει ο καθηγητής χρησιμοποιούμε τον πίνακα επιλογής τιμών (βλέπε και Εικόνα 1.52).

Πίνακας Γλωσσών

Abbreviation	Lang_greek	Lang_english
Συντομογραφία	Λεκτικό (ελληνικά)	Λεκτικό (αγγλικά)
ENG	ΑΓΓΛΙΚΑ	ENGLISH
GRE	ΕΛΛΗΝΙΚΑ	GREEK
FRE	ΓΑΛΛΙΚΑ	FRENCH

Εικόνα 1.52 Συντομογραφίες γλωσσών

1.15.8 Κανόνες οργάνωσης των πινάκων

Αν και είναι δύσκολο να διατυπωθούν απόλυτοι και «απαραβίαστοι» κανόνες θα δοθεί στη συνέχεια μια σειρά γενικών οδηγιών καλής σχεδίασης βάσεων δεδομένων:

- Ορίστε ένα συγκεκριμένο θέμα (a topic) για κάθε πίνακα και βεβαιωθείτε οτι όλα τα στοιχεία του πίνακα σχετίζονται με το θέμα αυτό
- Αν οι γραμμές του πίνακα αφήνουν κάποιες στήλες χωρίς τιμή χωρίστε τον πίνακα σε δύο πίνακες. Θυμηθείτε και το παράδειγμα Καθηγητής, Μαθητής (εικόνες 1.50, 1.51).
- Αν κάποια στοιχεία επαναλαμβάνονται σε έναν αριθμό γραμμών τότε μετακινήστε τα στοιχεία αυτά σε ένα νέο πίνακα και φροντίστε να υπάρχει μια συσχέτιση μεταξύ των δύο πινάκων που προέκυψαν
- Αν έχουμε στήλες με επαναλαμβανόμενες τιμές (Repeated fields) τότε είναι αναγκαία η δημιουργία ενός πίνακα παιδί στο πλαίσιο σχήματος πατέρας-παιδί. Για παράδειγμα σε έναν πίνακα ΕΤΑΙΡΙΑ(κωδικός, επωνυμία, έδρες) έχουμε για κάθε εταιρία τόσες γραμμές όσες και οι έδρες της.

Τότε αντί του πίνακα αυτού πρέπει να δημιουργήσουμε δύο πίνακες: ΕΤΑΙΡΙΑ(κωδικός, επωνυμία) με κύριο κλειδί τη στήλη κωδικός και ΕΤΑΙΡΙΑ_ΕΔΡΑ(κωδικός, έδρα) με κύριο κλειδί σύνθετο (κωδικός, έδρα).

- Χρησιμοποιήστε Πίνακες Αναζήτησης Τιμών (lookup tables) για να μειώσετε τον όγκο των αποθηκευμένων δεδομένων και να αυξήσετε την εγκυρότητα των δεδομένων κατά την εισαγωγή
- Μην αποθηκεύετε στοιχεία σε ένα πίνακα που μπορούν να υπολογιστούν από δεδομένα σε άλλους πίνακες, εκτός αν υπάρχει άλλος λόγος. Για παράδειγμα αν πρέπει η ανάκτηση ή η εκτύπωση στοιχείων να γίνεται πολύ γρήγορα.

Μια σοβαρή αιτία για την παραβίαση των παραπάνω οδηγιών είναι η απαίτηση να αποφεύγεται το άνοιγμα μεγάλου αριθμού πινάκων την ίδια χρονική στιγμή επειδή κατά κανόνα συνεπάγεται δέσμευση πόρων του συστήματος, μνήμης κ.λπ.. και συνήθως απολήγει στην επιβράδυνση εκτέλεση των εφαρμογών λογισμικού βάσεων δεδομένων. Βέβαια η παραβίαση των οδηγιών έχει σαν κύριες συνέπειες την αύξηση του μεγέθους της βάσης επειδή υπάρχουν πλεονάζοντα στοιχεία (redundancies) και την αυξημένη πιθανότητα λανθασμένων στοιχείων σε κάποιες γραμμές επειδή κάποια δεδομένα άλλαξαν χωρίς ταυτόχρονα να ενημερωθούν όλες οι σχετικές γραμμές σε όλους τους πίνακες.

1.15.9 Καθορισμός και Χρήση ευρετηρίων (Indexes)

Κατά την εισαγωγή στοιχείων σε ένα πίνακα της βάσης οι γραμμές αποθηκεύονται με τη σειρά εισαγωγής τους. Συνήθως, όμως, απαιτείται για τις εφαρμογές να βρίσκουμε μια συγκεκριμένη γραμμή γρήγορα και όχι ψάχνοντας ολόκληρο τον πίνακα. Ένα ευρετήριο (index) παρέχει έναν τρόπο γρήγορης πρόσβασης στα δεδομένα του πίνακα.

1.15.9.1 Περιγραφή του προβλήματος με παράδειγμα

Στην Εικόνα 1.53 βλέπουμε έναν πίνακα EMP με στοιχεία υπαλλήλων. Υποθέτουμε ότι θέλουμε να εκτελέσουμε τη δήλωση SELECT * FROM emp WHERE loc=”ATHENS”;

Επιπλέον, υποθέτουμε ότι για την ανάγνωση μιας γραμμής απαιτείται μία λειτουργία “read” (read operation) και ότι κάθε γραμμή βρίσκεται σε διαφορετική σελίδα της αποθηκευμένης βάσης δεδομένων. Τότε για την απάντηση θα χρειαστούν επτά “read” σε επτά διαφορετικές σελίδες της αποθηκευμένης βάσης δεδομένων.

Πίνακας EMP

Empno	Ename	Loc	Database Page
1	SMITH	CAIRO	1
6	PAGE	DUBLIN	2
4	CHEN	CAIRO	3
3	JONES	CAIRO	4
7	ELMASRI	DUBLIN	5
12	MARTIN	ATHENS	6
11	STONE	DUBLIN	7

Ευρετήριο BRANCH_INDEX

ATHENS	6
CAIRO	1, 3, 4

DUBLIN	2, 5, 7

Εικόνα 1.53 Πίνακας υπαλλήλων με στοιχεία αποθηκευμένα σε σελίδες της βάσης δεδομένων και ευρετήριο το οποίο διευκολύνει την αναζήτηση στοιχείων σχετικών με την έδρα της εργασίας τους

Ένα ευρετήριο της βάσης δεδομένων περιλαμβάνει μια λίστα διατεταγμένων τιμών για μία στήλη (ή συνδυασμό στηλών) ενός πίνακα της βάσης. Περιλαμβάνει, επίσης, δείκτες (pointers) στις σελίδες της βάσης (database pages) που περιέχουν τις γραμμές στις οποίες υπάρχει κάθε μία τιμή. Σε φυσικό επίπεδο, το ευρετήριο περιλαμβάνει δείκτες (pointers) που λένε στη μηχανή της βάσης δεδομένων (database engine) που βρίσκεται αποθηκευμένη η πραγματική εγγραφή με τα δεδομένα. Στο παράδειγμά μας το ευρετήριο θα μπορούσε να έχει τη μορφή που βλέπουμε στην Εικόνα 1.53.

Αντιστοιχία με ευρετήριο βιβλίων

Το ευρετήριο αυτό είναι παρόμοιο με το ευρετήριο, Index, που υπάρχει στο τέλος πολλών βιβλίων. Χρησιμοποιώντας ένα ευρετήριο βιβλίου βρίσκετε πρώτα τις λέξεις ή και τα θέματα (terms, keywords) που σας ενδιαφέρουν και στη συνέχεια υπάρχουν οι δείκτες (αριθμός σελίδας στην περίπτωση του βιβλίου) που σας επιτρέπουν να πάτε και να διαβάσετε αυτό που σας ενδιαφέρει. Ακολουθεί παράδειγμα αποσπάσματος ευρετηρίου.

Inconsistent analysis, 192

Indexes, 176-80, 183-4, 199

B-tree, 177-9

In SQL, 180

Sparse, 183-4

Information hiding, 152

Παραδείγματα ευρετηρίων

Η δομή του ευρετηρίου επιτρέπει τη γρήγορη αναζήτηση και ανάκτηση των στοιχείων. Για παράδειγμα, ένα ευρετήριο πίνακα πελατών επιτρέπει να βλέπουμε και να ψάχνουμε με αλφαβητική σειρά, άρα γρήγορα, τα στοιχεία των πελατών μας.

Για τον ίδιο πίνακα μπορούμε να ορίσουμε πολλά ευρετήρια. Για παράδειγμα, μπορούμε να ορίσουμε ευρετήρια του πίνακα των Υπαλλήλων βασιζόμενα στις στήλες ημερομηνία γέννησης, ημερομηνία πρόσληψης αλλά και στο συνδυασμό στηλών Επώνυμο και Όνομα. Κάθε ένα από τα ευρετήρια επιτρέπει γρήγορη πρόσβαση στις γραμμές του πίνακα με διαφορετική σειρά και με διαφορετικό τρόπο ή και διαφορετικό στόχο.

Σημείωση

Μπορείτε να δείτε τα στοιχεία των πινάκων σας με διάφορους τρόπους (διαφορετική σειρά) χρησιμοποιώντας την υποπρόταση ταξινόμησης ORDER BY σε μία δήλωση SELECT της γλώσσας Structured Query Language (SQL). Η ύπαρξη ευρετηρίου μειώνει το χρόνο εκτέλεσης της δήλωσης SELECT.

Παραδείγματα ευρετηρίων βασισμένων σε απλές στήλες

Μπορούμε να ορίσουμε Ευρετήρια του πίνακα "Υπάλληλος" βασιζόμενα αντίστοιχα στις απλές στήλες:

Κωδικός_Υπαλλήλου, Αριθμός_Αστυνομικής_ταυτότητας, ΑΜΚΑ.

Παραδείγματα ευρετηρίων βασισμένων σε πολλές στήλες

Για τον πίνακα του υπαλλήλου ορίζουμε δύο ευρετήρια του πίνακα "Υπάλληλος" βασιζόμενα αντίστοιχα στα σύνθετες στήλες (Επώνυμο, Όνομα υπαλλήλου), (Όνομα, Επώνυμο υπαλλήλου). Επισημαίνουμε ότι τα δύο ευρετήρια είναι διαφορετικά! Μάλιστα το δεύτερο έχει ένα σοβαρό μειονέκτημα αν μας ενδιαφέρει ανάκτηση υπαλλήλων με το επώνυμό τους. Μπορείτε να το διαπιστώσετε στο παράδειγμα της εικόνας 1.54.

Ευρετήριο βασιζόμενο σε (επώνυμο, όνομα)

Επώνυμο	Όνομα	Δείκτες
ΠΑΠΑΔΟΠΟΥΛΟΣ	ΙΩΑΝΝΗΣ	……..
ΠΑΠΑΔΟΠΟΥΛΟΣ	ΝΙΚΟΛΑΟΣ	……..
ΠΑΠΑΔΟΠΟΥΛΟΣ	ΧΡΗΣΤΟΣ	……..
ΡΑΠΤΗΣ	ΙΩΑΝΝΗΣ	……..
ΡΑΠΤΗΣ	ΝΙΚΟΛΑΟΣ	……..
ΣΠΥΡΟΥ	ΙΩΑΝΝΗΣ	……..
ΣΠΥΡΟΥ	ΝΙΚΟΛΑΟΣ	……..

Ευρετήριο βασιζόμενο σε (όνομα, επώνυμο)

Όνομα	Επώνυμο	Δείκτες
ΙΩΑΝΝΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	……..
ΙΩΑΝΝΗΣ	ΡΑΠΤΗΣ	……..
ΙΩΑΝΝΗΣ	ΣΠΥΡΟΥ	……..
ΝΙΚΟΛΑΟΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	……..
ΝΙΚΟΛΑΟΣ	ΡΑΠΤΗΣ	……..
ΝΙΚΟΛΑΟΣ	ΣΠΥΡΟΥ	……..
ΧΡΗΣΤΟΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	……..

Εικόνα 1.54 Τα δύο ευρετήρια βασίζονται σε διαφορετική σειρά των στηλών όνομα, επώνυμο και μειώνουν το χρόνο ανάκτησης δεδομένων διαφορετικών ερωτημάτων (queries)

1.16 Κανονικοποίηση δεδομένων (Data normalization) και η σημασία της για τις εφαρμογές

Κανονικοποίηση είναι μια διαδικασία (process) που διασφαλίζει ότι οι σχεσιακές δομές (οι πίνακες) μιας σχεσιακής βάσης δεδομένων είναι σε μία «επαρκή» (efficient) μορφή. Τι σημαίνει επαρκής μορφή:

1) **Απαλοιφή (εξάλειψη) των πλεονασμών (redundant data)** στην αποθήκευση των δεδομένων στη βάση δεδομένων. Η πλήρης εφαρμογή της διαδικασίας κανονικοποίησης οδηγεί στην αποθήκευση κάθε δεδομένου (πληροφορίας) ακριβώς μια φορά στη βάση.
2) **Ελαχιστοποίηση της χρήσης τιμών NULL** στους πίνακες. Χωρίς την κανονικοποίηση μπορεί να έχουμε στοιχεία πελατών και στοιχεία υπαλλήλων στον ίδιο πίνακα. Τότε, για όλα τα ειδικά στοιχεία των πελατών οι πίνακες έχουν τιμή NULL στην περίπτωση των γραμμών με στοιχεία των υπαλλήλων και βέβαια ισχύει και το αντίστροφο.
3) **Αποφυγή απώλειας της πληροφορίας**. Για παράδειγμα, αν δεν εφαρμοστεί σωστά η διαδικασία κανονικοποίησης σε ένα σύστημα παραγγελιών όταν διαγράψουμε τα στοιχεία κάποιας παραγγελίας

μπορεί να απωλέσουμε και τα στοιχεία του πελάτη. Αν ο πελάτης δώσει νέα παραγγελία τότε θα πρέπει να ξαναπεράσουμε τα στοιχεία του.

4) **Εύκολη και γρήγορη σύνδεση (join)** στοιχείων από πολλούς πίνακες.

Στις εμπορικές εφαρμογές η διαδικασία της κανονικοποίησης πρέπει να οδηγήσει στην Τρίτη Κανονική Μορφή.

Η διαδικασία της κανονικοποίησης θα μας απασχολήσει αρκετά και σε επόμενο κεφάλαιο. Στη συνέχεια θα συζητήσουμε την πρώτη, τη δεύτερη και την τρίτη κανονική μορφή. Επίσης, σε επόμενο κεφάλαιο θα παρουσιάσουμε την τέταρτη και την πέμπτη κανονική μορφή και την κανονική μορφή Boyce Codd.

Παράδειγμα

Έστω ότι θέλουμε να σχεδιάσουμε ένα απλό σύστημα διαχείρισης παραγγελιών ηλεκτρονικών ειδών. Για κάθε είδος που περιλαμβάνεται στην παραγγελία είναι απαραίτητο να κρατάμε κωδικό, περιγραφή, τιμή καταλόγου, κωδικό παραγγελίας, ημερομηνία αλλά και κωδικό πελάτη, όνομα, διεύθυνση και τηλέφωνο. Στην Εικόνα 1.55 βλέπουμε ένα αντιπροσωπευτικό δείγμα δεδομένων. Ακολουθεί επεξήγηση των στηλών:

Orderno Κωδικός παραγγελίας, Custno Κωδικός Πελάτη παραγγελίας, Name Επωνυμία πελάτη παραγγελίας, Address Διεύθυνση πελάτη, Phone τηλέφωνο, Odate Ημερομηνία παραγγελίας, Itemno Κωδικός Είδους παραγγελίας, Item Περιγραφή είδους, Price τιμή, Qty ποσότητα, Sale Έκπτωση σε κάποιο είδος (%)

Πίνακας Orders (Παραγγελίας)

Orderno	CustNo	Name	Address	Phone	Odate	Itemno	Item	Price	Qty	Sale (%)
1	123	JIM	THISSEUS	2101234567	7/1/2021	10	BOLT A	1	100	10
1	123	JIM	THISSEUS	2101234567	7/1/2021	15	BOLT B	2	200	

(α) Η πρώτη κανονική μορφή της βάσης είναι ένας μεγάλος μη αποδοτικός (μη «επαρκής» για τις εφαρμογές) πίνακας δεδομένων

Orderno	CustNo	Odate	Itemno	Item	Price	Qty	Sale (%)
1	123	7/1/2021	10	BOLT A	1	100	10
1	123	7/1/2021	15	BOLT B	2	200	

Custno	Name	Address	phone
123	JIM	THISSEUS	2101234567

(β) Εναλλακτική σχεδίαση της πρώτης κανονικής μορφής της βάσης κάπως πιο αποδοτική για τις εφαρμογές

Εικόνα 1.55 Σχεδίαση πρώτης κανονικής μορφής

Στην Εικόνα 1.55(α) βλέπουμε ότι πολλά από τα δεδομένα στον πίνακα επαναλαμβάνονται. Αυτή η επανάληψη οδηγεί σε δύο σημαντικά προβλήματα. Το πρώτο είναι η σπατάλη χώρου επειδή επαναλαμβάνουμε πληροφορία. Το δεύτερο πρόβλημα είναι πρόβλημα συμβατότητας και εγκυρότητας δεδομένων. Αν για παράδειγμα ένας πελάτης αλλάξει τηλέφωνο ή διεύθυνση πρέπει το στοιχείο αυτό να αλλάξει σε όλες τις γραμμές του πελάτη και βέβαια υπάρχει πιθανότητα να μην αλλαχθεί σε κάποιες από

αυτές. Έτσι κάποια στιγμή αναζητώντας το τηλέφωνο ή τη διεύθυνση πελάτη είναι πιθανόν να ανακτήσουμε λάθος στοιχεία.

Μια καλύτερη λύση είναι να τοποθετήσετε τα στοιχεία του πελάτη σε ένα πίνακα και τα στοιχεία της παραγγελίας σε άλλον. Για τη σύνδεση των πινάκων που θα φτιάξετε πρέπει να συμπεριλάβετε τον κωδικό του πελάτη και στον πίνακα των παραγγελιών, όπως φαίνεται στην Εικόνα 1.55(β).

Τα ίδια ισχύουν και για τα στοιχεία του είδους. Οπότε μπορούμε να σχεδιάσουμε τη Δεύτερη Κανονική μορφή (Εικόνα 1.56)

Δεύτερη Κανονική μορφή (Εικόνα 14)

Πίνακας Orders (Παραγγελίας)

Orderno	Custno	Odate	Itemno	Qty
1	123	7/1/2021	10	100
1	123	7/1/2021	15	200

Πίνακας Customer (Πελάτη)

Custno	Name	Address	Phone
123	JIM	THISSEUS	2101234567

Πίνακας Item (Είδους)

Itemno	Item	Price	Sale (%)
10	BOLT A	1	10
15	BOLT B	2	

Εικόνα 1.56 Δεύτερη Κανονική Μορφή. Οι πίνακες Πελάτη, Είδους και Παραγγελίας απαλείφουν κάποιους από τους πλεονασμούς στην αποθήκευση δεδομένων

Με τη διευθέτηση αυτή των στοιχείων το τηλέφωνο και η διεύθυνση πελάτη εμφανίζονται μόνο σε μία γραμμή και σε περίπτωση μεταβολής των στοιχείων αυτών απλά αλλάζετε μόνο μία γραμμή.

Η ίδια παρατήρηση με την παραπάνω ισχύει και για τα είδη της παραγγελίας και τα βασικά στοιχεία της παραγγελίας. Επομένως, οδηγούμαστε σε καλή σχεδίαση όπου η βάση δεδομένων περιλαμβάνει τέσσερις πίνακες, όπως φαίνεται στην Εικόνα 1.57. Στην περίπτωση αυτή είστε σίγουροι ότι όταν πρέπει να αλλάξει κάποιο στοιχείο θα αλλάξει μόνο σε μία θέση στη βάση δεδομένων. Από τους πίνακες της σχεδίασης που φαίνονται και στην Εικόνα 1.57, ο πίνακας Βασικά_Στοιχεία_Παραγγελίας και ο πίνακας Είδη_Παραγγελίας συνδέονται αντίστοιχα με τους πίνακες Πελάτης και Είδος. Ο πίνακας Είδη_Παραγγελίας περιλαμβάνει μία γραμμή για κάθε είδος που περιλαμβάνεται στην παραγγελία. Ο πίνακας Παραγγελία συσχετίζει τα είδη με την ημερομηνία παραγγελίας και τον πελάτη .

Πίνακας orders (Βασικών Στοιχείων Παραγγελίας)

Orderno	Custno	Odate
1	123	7/1/2021

Πίνακας orderlines (Λεπτομερειών-Γραμμών Παραγγελίας)

Orderno	Itemno	Qty
1	10	100
1	15	200

Πίνακας items (Προϊόντων)

Itemno	Item	Price	Sale
10	BOLT A	1	10
15	BOLT B	2	

Πίνακας customers (Πελάτη)

Custno	Name	Address	Phone
123	JIM	THISSEUS	2101234567

Εικόνα 1.57 Τρίτη Κανονική Μορφή. Η πλήρης κανονικοποίηση των πινάκων παρέχει τη βέλτιστη σχεδίαση και διασφαλίζει οικονομική αποθήκευση και γρήγορη ανάκτηση δεδομένων

Συμπέρασμα

Η οργάνωση των δεδομένων σε βάση πρέπει να διευκολύνει την αναζήτηση-εύρεση αλλά και την τροποποίηση της αποθηκευμένης πληροφορίας. Επομένως, είναι απαραίτητη η καλή σχεδίαση της βάσης δεδομένων. Δηλαδή, απαιτούνται: (1) ορθές αποφάσεις κατά την ανάλυση δεδομένων ώστε να καταγράφουμε και να διαχειριζόμαστε όλα τα απαραίτητα στοιχεία-δεδομένα και (2) οργάνωση της βάσης δεδομένων στην τρίτη κανονική μορφή.

1.16.1 Σχεδιασμός βάσης δεδομένων και κανονικοποίηση. Πρώτη, δεύτερη και τρίτη κανονική μορφή

Στη συνέχεια θα μελετηθεί, χωρίς μαθηματικό φορμαλισμό, και με τη βοήθεια παραδειγμάτων μία συστηματικότερη προσέγγιση στο σχεδιασμό βάσεων δεδομένων. Πιο συγκεκριμένα, στην ενότητα αυτή θα ασχοληθούμε με τη σχεσιακή προσέγγιση της οργάνωσης των δεδομένων παρουσιάζοντας τη στρατηγική σχεδιασμού των πινάκων που είναι γνωστή στη βιβλιογραφία σαν κανονικοποίηση (normalisation).

Μελέτη περίπτωσης

Θεωρούμε μια απλουστευμένη βάση δεδομένων προσωπικού. Στην Εικόνα 1.58 βλέπουμε την παγκόσμια σχέση ή πίνακα (universal relation) περιγραφής της βάσης, τις ιδιότητες της σχέσης (attributes) και κάποια στιγμιότυπα (occurrences) του τύπου γραμμής-εγγραφής (row-record type).

Προσοχή! Η λέξη σχέση ή πίνακας που χρησιμοποιούμαι για τον όρο relation ταυτίζεται, ανάλογα με την περίπτωση με τον όρο entity (οντότητα) ή relationship (συσχέτιση) που χρησιμοποιήσαμε στη μοντελοποίηση. Η διαφορά θα φανεί καλύτερα στο κεφάλαιο 6.

ΠΙΝΑΚΑΣ EMPLOYEE

ENAME	(LANG, USE)	JOB	EXPR	SAL	(DEPTNO, MGR)
Βαλάκος	(C++, 3), (JAVA, 2)	Προγραμματιστής	4	1200	(020, Μπομπότης)
Καμαρινός	(C++, 2), (PHP, 1))	Προγραμματιστής	3	1150	(010, Χολογκούνης)
Μπομπότης	(C++, 4)	Επικεφαλής	4	1450	(020, Μπομπότης)
Πλακίδης	(C++, 5), (JAVA, 2)	Προγραμματιστής	7	1300	(020, Μπομπότης)
Χολογκούνης	(C++, 4), (PYTHON, 1)	Επικεφαλής	5	1500	(020, Χολογκούνης)

ENAME κύριο κλειδί

Εικόνα 1.58 Μη κανονικοποιημένη μορφή βάσης δεδομένων προσωπικού

Στη περίπτωση αυτή χρησιμοποιούμε ένα δείγμα δεδομένων (sample of data).Το δείγμα αυτό θα μας χρησιμεύσει για να διαπιστώσουμε κάποια μειονεκτήματα που έχει ο σχεδιασμός «όλα σε έναν πίνακα» και θα μας επιτρέψει να δούμε τα αποτελέσματα κάποιων τροποποιήσεων στο σχεδιασμό.

1.16.2 Ιδιότητες της σχέσης (πίνακα)

Στη σχέση EMPLOYEE χρησιμοποιούμε τις παρακάτω ιδιότητες (στήλες):

1) ENAME όνομα υπαλλήλου
2) LANG γλώσσα προγραμματισμού που γνωρίζει
3) USE πόσα χρόνια τη χρησιμοποιεί
4) JOB θέση
5) EXPR συνολική εμπειρία
6) SAL μισθός
7) DEPTNO τμήμα στο οποίο ανήκει
8) MGR επικεφαλής τμήματος

Κύριο κλειδί στη σχέση είναι η ιδιότητα ENAME. Αυτό δεν είναι και πολύ σωστό σε πραγματικές συνθήκες και ειδικότερα στην περίπτωση που έχουμε συνωνυμίες. Ορθότερα, θα χρησιμοποιούσαμε έναν κωδικό μοναδικό για κάθε υπάλληλο ως κύριο κλειδί.

Τα πεδία ορισμού των χαρακτηριστικών είναι τα εξής:

- DOM(ENAME)= σύνολο αλφαβητικών συμβολοσειρών (strings)
- DOM(MGR)={Βαλάκος, ..., Χολογκούνης}=
  σύνολο αλφαβητικών συμβολοσειρών (strings)
- DOM(LANG) = {C++, JAVA, PYTHON, PHP}
- DOM(USE) =D(EXPR) = {n | n διψήφιος ακέραιος θετικός αριθμός}
- DOM(JOB) = {ΑΝΑΛΥΤΗΣ, ΠΡΟΓΡΑΜΜΑΤΙΣΤΗΣ, DBA, ΕΠΙΚΕΦΑΛΗΣ ΕΡΓΟΥ}
- DOM(SAL) = {x |x είναι 7-ψήφιος θετικός πραγματικός αριθμός με 2 δεκαδικά ψηφία το πολύ }
- DOM(DEPTNO) = {010-ΑΠΟΘΗΚΕΣ, 020-ΜΙΣΘΟΔΟΣΙΑ}

Επειδή ο υπάλληλος μπορεί να γνωρίζει πολλές γλώσσες και επειδή για κάθε γλώσσα έχει συγκεκριμένη εμπειρία στη σχέση εμφανίζεται η σύνθετη ιδιότητα: (LANG, USE). Η ιδιότητα αυτή ορίζεται πάνω σε σύνθετο πεδίο ορισμού (domain): DOM(LANG) x DOM(USE)

Μάλιστα η ιδιότητα μπορεί να πάρει πολλά ζεύγη τιμών για ένα υπάλληλο, π.χ.,{ (C++,3) , (JAVA,2) }

Στη περίπτωση αυτή μιλάμε για σύνθετη επαναλαμβανόμενη ιδιότητα.

Επίσης, εμφανίζεται η σύνθετη ιδιότητα (DEPTNO, MGR).

Με τη χρησιμοποίηση σύνθετων ιδιοτήτων η σχέση EMPLOYEE γίνεται πιο οικονομική σε απαιτήσεις μνήμης αλλά έχουμε προβλήματα κατά την αναζήτηση.

Συνέπειες σχεδιασμού

Η σχεδίαση αυτή της βάσης επιτρέπει εύκολα απάντηση σε ερωτήσεις της μορφής

- Ποιες γλώσσες γνωρίζει ο υπάλληλος Βαλάκος;
  αλλά πολύ δύσκολα σε ερωτήσεις της μορφής:

- Ποιοι υπάλληλοι χρησιμοποιούν γλώσσα JAVA;
- Ποιος ο επικεφαλής του τμήματος "Αποθήκες" ;
- Ποιοι υπάλληλοι ανήκουν στο τμήμα "Αποθήκες" ;
  Δύσκολη είναι και η ενημέρωση της βάσης δεδομένων για αλλαγές όπως
- Αντικατάσταση του επικεφαλής του τμήματος "Αποθήκες" με τον υπάλληλο Βαλάκο.

Από τα παραπάνω είναι φανερή η ανάγκη για μια διαδικασία, τη διαδικασία κανονικοποίησης, που θα οδηγήσει σε ένα επανασχεδιασμό της βάσης ώστε να μπορούμε να τη χρησιμοποιήσουμε πιο αποδοτικά. Ο εντοπισμός της "ρίζας του κακού" είναι εύκολος:

"Φταίνε" οι σύνθετες επαναλαμβανόμενες ιδιότητες (attributes) δηλαδή οι ιδιότητες που έχουν σύνθετο πεδίο ορισμού (domain) και δέχονται πολλές τιμές. Άρα:

Κανόνας 1

Σε μια σχέση/πίνακα δεν πρέπει να εμφανίζονται σύνθετες στήλες / πεδία ορισμού ανά ιδιότητα, δηλαδή, σε περιγραφική προσέγγιση για κάθε γραμμή του πίνακα, κάθε στήλη είναι απλή ιδιότητα και πρέπει να περιέχει ακριβώς μια τιμή.

Η εφαρμογή του κανόνα 1 σε μια μη κανονικοποιημένη σχέση όπως αυτή της εικόνας 1.58 οδηγεί σε σχέση πρώτης κανονικής μορφής (first normal form) (βλέπε και Εικόνα 1.59) .

ΠΙΝΑΚΑΣ EMPLOY

ENAME	LANG	USE	JOB	EXPR	SAL	DEPTNO	MGR
Βαλάκος	C++	3	Προγραμματιστής	4	1200	020	Μπομπότης
Βαλάκος	JAVA	2	Προγραμματιστής	4	1200	020	Μπομπότης
Καμαρινός	C++	2	Προγραμματιστής	3	1150	010	Χολογκούνης
Καμαρινός	PHP	1	Προγραμματιστής	3	1150	010	Χολογκούνης
Μπομπότης	C++	4	Επικεφαλής	4	1450	020	Μπομπότης
Πλακίδης	C++	5	Προγραμματιστής	7	1300	020	Μπομπότης
Πλακίδης	JAVA	2	Προγραμματιστής	7	1300	020	Μπομπότης
Χολογκούνης	C++	4	Επικεφαλής	5	1500	020	Χολογκούνης
Χολογκούνης	PYTHON	1	Επικεφαλής		1500	020	Χολογκούνης

(ENAME, LANG) σύνθετο κύριο κλειδί

Εναλλακτική σχεδίαση πρώτης κανονικής μορφής

ENAME	LANG	USE

ENAME	JOB	EXPR	SAL	DEPTNO	MGR

Εικόνα 1.59 Πρώτη κανονική μορφή βάσης δεδομένων προσωπικού

Σημείωση

Εναλλακτικά η πρώτη κανονική μορφή θα μπορούσε να αποτελείται από περισσότερους πίνακες, Για παράδειγμα από τους πίνακες της εικόνας 1.59.

Αξιοσημείωτο είναι ότι στη σχέση πρώτης κανονικής μορφής αναγκασθήκαμε να αλλάξουμε το κύριο κλειδί και αυξήθηκαν οι απαιτήσεις της βάσης σε χώρο για αποθήκευση. Κύριο κλειδί είναι το σύνθετο κλειδί (ENAME, LANG).

Δυστυχώς, υπάρχουν πολλά προβλήματα ακόμη στην ενημέρωση της βάσης. Για παράδειγμα, είναι δύσκολο να ικανοποιηθεί το παρακάτω αίτημα:

Ο υπάλληλος Βαλάκος να γίνει επικεφαλής του τμήματος "Αποθήκες"

Βέβαια, υπάρχει τρόπος να λυθούν αυτά τα προβλήματα. Πράγματι, το σύνθετο κύριο κλειδί έχει ένα τμήμα (στήλη ENAME) που μπορεί να ορίζει μονοσήμαντα άλλες ιδιότητες (στήλες). Έτσι, αν η βάση "χωρισθεί" όπως στην Εικόνα 1.60 είναι φανερό ότι η ικανοποίηση των προηγουμένων αιτημάτων είναι πιο εύκολη στην προκύπτουσα βάση. Δηλαδή, βλέπουμε ότι η εναλλακτική σχεδίαση της πρώτης κανονικής μορφής στην Εικόνα 1.59 είναι και στη δεύτερη κανονική μορφή. Πρώτη και δεύτερη κανονικές μορφές ταυτίζονται στο παράδειγμά μας.

ΠΙΝΑΚΑΣ QUALIF

ENAME	LANG	USE
Βαλάκος	C++	3
Βαλάκος	JAVA	2
Καμαρινός	C++	2
Καμαρινός	PHP	1
Μπομπότης	C++	4
Πλακίδης	C++	5
Πλακίδης	JAVA	2
Χολογκούνης	C++	4
Χολογκούνης	PYTHON	1

(ENAME, LANG) κύριο κλειδί

ΠΙΝΑΚΑΣ EMPL

ENAME	JOB	EXPR	SAL	DEPTNO	MGR
Βαλάκος	Προγραμματιστής	4	1200	020	Μπομπότης
Καμαρινός	Προγραμματιστής	3	1150	010	Χολογκούνης
Μπομπότης	Επικεφαλής	4	1450	020	Μπομπότης
Πλακίδης	Προγραμματιστής	7	1300	020	Μπομπότης
Χολογκούνης	Επικεφαλής	5	1500	020	Χολογκούνης

ENAME κύριο κλειδί

Εικόνα 1.60 Δεύτερη κανονική μορφή βάσης δεδομένων προσωπικού

Στη συνέχεια διατυπώνεται κανόνας τον οποίο μπορούμε να εφαρμόζουμε για να κατασκευάσουμε τη δεύτερη κανονική μορφή.

Κανόνας 2

Αν μία σχέση (πίνακας) είναι στην πρώτη κανονική μορφή και το κύριο κλειδί της (δηλαδή το κλειδί που ορίζει μονοσήμαντα όλες τις στήλες της) είναι σύνθετο (αποτελείται από περισσότερες από μια ιδιότητες) και

ένα τμήμα του ορίζει μονοσήμαντα ιδιότητες (στήλες) πρέπει το τμήμα αυτό και οι αντίστοιχες ιδιότητες (στήλες) να αποτελέσουν μια ξεχωριστή σχέση (πίνακα).

Οι σχέσεις EMPL, QUALIF που προκύπτουν από την εφαρμογή του κανόνα 2 (Εικόνα 1.60) λέμε ότι είναι στη δεύτερη κανονική μορφή (second normal form). Κύριο κλειδί στη σχέση EMPL είναι η ιδιότητα ENAME και στη σχέση QUALIF η σύνθετη ιδιότητα (ENAME, LANG).

Υπάρχουν περιπτώσεις που είναι δύσκολη η ενημέρωση της βάσης δεδομένων ή η απάντηση σε ερωτήματα και στην περίπτωση που η βάση δεδομένων σε δεύτερη κανονική μορφή . Για παράδειγμα:

Ποιά η εμπειρία σε C++ των προγραμματιστών οι οποίοι εργάζονται στο τμήμα με επικεφαλής τον Μπομπότη;

Στην περίπτωση αυτή αν η βάση "χωρισθεί" όπως στην Εικόνα 1.61 η απάντηση στην ερώτηση είναι πιο εύκολη. Η μορφή αυτή της βάσης, είναι η τρίτη κανονική μορφή και προέκυψε όταν οι ιδιότητες (στήλες) του τμήματος DEPTNO και MGR αποτέλεσαν ξεχωριστή σχέση.

ΠΙΝΑΚΑΣ QUALIF

ENAME	LANG	USE
Βαλάκος	C++	3
Βαλάκος	JAVA	2
Καμαρινός	C++	2
Καμαρινός	PHP	1
Μπομπότης	C++	4
Πλακίδης	C++	5
Πλακίδης	JAVA	2
Χολογκούνης	C++	4
Χολογκούνης	PYTHON	1

(ENAME, LANG) κύριο κλειδί

ΠΙΝΑΚΑΣ EMP

ENAME	JOB	EXPR	SAL	DEPTNO
Βαλάκος	Προγραμματιστής	4	1200	020
Καμαρινός	Προγραμματιστής	3	1150	010
Μπομπότης	Επικεφαλής	4	1450	020
Πλακίδης	Προγραμματιστής	7	1300	020
Χολογκούνης	Επικεφαλής	5	1500	020

ENAME κύριο κλειδί

ΠΙΝΑΚΑΣ DEPTMGR

DEPTNO	MGR
010	Χολογκούνης
020	Μπομπότης

(DEPTNO, MGR) κύριο κλειδί

Εικόνα 1.61 Τρίτη κανονική μορφή βάσης δεδομένων προσωπικού

Το κρίσιμο σημείο είναι ότι η βάση δεδομένων είναι στη δεύτερη κανονική μορφή και ενώ το κύριο κλειδί ορίζει μονοσήμαντα τις ιδιότητες (στήλες) DEPTNO και MGR η ιδιότητα (στήλη) DEPTNO ορίζει επίσης μονοσήμαντα την ιδιότητα (στήλη) MGR και αντίστροφα.

Το φαινόμενο αυτό ονομάζεται μεταβατική εξάρτηση (transitive dependency) μεταξύ των ιδιοτήτων (στηλών).

Στη συνέχεια διατυπώνεται ο κανόνας 3 τον οποίο εφαρμόζουμε για να κατασκευάσουμε την τρίτη κανονική μορφή.

Κανόνας 3

Σε κάθε σχέση (πίνακα) της δεύτερης κανονικής μορφής όλες οι ιδιότητες (στήλες) της πρέπει να αντιστοιχούν απευθείας στο κύριο κλειδί χωρίς μεταβατικές εξαρτήσεις διαμέσου των άλλων στηλών.

Οι σχέσεις (πίνακες) EMP, QUALIF, DEPTMGR οι οποίες προκύπτουν από την εφαρμογή (στη δεύτερη κανονική μορφή) του κανόνα 3 αποτελούν την τρίτη κανονική μορφή (third normal form). Κύρια κλειδιά των σχέσεων αυτών είναι αντίστοιχα τα παρακάτω:

ENAME, (ENAME, LANG), DEPTNO

1.16.3 Περιπτώσεις παραβίασης της τρίτης κανονικής μορφής (αποκανονικοποίηση)

Μετά το σχεδιασμό της 3NF και για την εξυπηρέτηση ειδικών αναγκών της εφαρμογής λογισμικού βάσης δεδομένων υπάρχουν περιπτώσεις που παραβιάζουμε την τρίτη κανονική μορφή. Τέτοιες περιπτώσεις είναι:

- η φύλαξη των ίδιων στοιχείων δύο ή περισσότερες φορές ώστε να διευκολυνθεί και επιταχυνθεί η ανάκτηση και η εμφάνιση των αποτελεσμάτων.
- η φύλαξη λιγότερων στοιχείων για την ελάττωση του χώρου που θα δεσμευθεί για τη βάση δεδομένων.
- Η ύπαρξη δεδομένων που αλλάζουν ελάχιστα και δεδομένων που υφίστανται πολλές μεταβολές. Για παράδειγμα, κάποια βασικά στοιχεία μίας εφαρμογής παραμένουν στην ίδια σχέση ή τις ίδιες σχέσεις περίπου αναλλοίωτα. Όσα στοιχεία μεταβάλλονται κατά τη μηνιαία ή ημερήσια απογραφή/επικαιροποίηση στοιχείων, ενδεχομένως, πρέπει να υπάρχουν σε διαφορετικές σχέσεις.

1.16.4 Παράδειγμα σχεσιακής βάσης δεδομένων διαχείρισης παραγγελιών και αποκανονικοποίηση

Έστω απλοποιημένη βάση δεδομένων διαχείρισης παραγγελιών.

Η τρίτη κανονική μορφή είναι η εξής:

SUPPLIERS(SNO, SNAME, STATUS, SCITY)

PARTS(PNO, PNAME, PCOLOR, PCITY)

ORDER(ORDERNO, SNO, TOTAL, ORDERDATE)

ORDERLINE(ORDERNO, LINENO, PNO, QUANTITY)

όπου ORDERNO-κωδικός παραγγελίας, TOTAL-σύνολο, DATE-ημερομηνία, LINENO-γραμμή, QUANTITY-ποσότητα είδους.

Για να εμφανίζουμε ταχύτερα τα αποτελέσματα της ανάκτησης στοιχείων μπορούμε να φυλάξουμε στοιχεία με τον εξής τρόπο:

SUPPLIERS(SNO, SNAME, STATUS, SCITY)

PARTS(PNO, PNAME, PCOLOR, PCITY)

ORDER(ORDERNO, SNO, SNAME, TOTAL, ORDERDATE)

ORDERLINE(ORDERNO, LINENO, PNO, PNAME, QUANTITY)

Για να ελαττώσουμε το δεσμευμένο χώρο για τη βάση δεδομένων μπορούμε να φυλάξουμε στοιχεία με τον εξής τρόπο:

ORDER(ORDERNO, SNO ή SNAME,TOTAL, ORDERDATE)

ORDERLINE(ORDERNO, LINENO, PNO ή PNAME, QUANTITY)

Σημείωση

Η παραπάνω μεθοδολογία σχεδιασμού αφορά όχι μόνο τις βάσεις δεδομένων αλλά και το σχεδιασμό συστημάτων που θα υλοποιηθούν με παραδοσιακά αρχεία. Στο σχεδιασμό των βάσεων, συνήθως στην καθημερινή πρακτική, αρκεί να κατασκευαστεί η τρίτη κανονική μορφή ή η ισχυρότερη κανονική μορφή Boyce-Codd στην οποία θα αναφερθούμε στη συνέχεια. Επίσης, θα αναφερθούμε για λόγους πληρότητας και στις 4NF, 5NF.

1.17 Κανονική Μορφή Boyce-Codd, Τέταρτη Κανονική Μορφή, Πέμπτη Κανονική Μορφή.

1.17.1 Τέταρτη κανονική μορφή

Έστω ότι ο πίνακας EMP_LANG_PROD ανήκει στο Σύστημα Βάσης Δεδομένων μίας κατασκευαστικής εταιρείας. Για τον πίνακα αυτό μπορούμε να πούμε ότι βρίσκεται στην τρίτη κανονική μορφή και το κύριο κλειδί του είναι το σύνθετο κλειδί (PersonNo, Language, ProductNo).

Παρατήρηση

Σε κάθε τιμή της στήλης PersonNo μπορεί να αντιστοιχούν μία ή περισσότερες τιμές του ProductNo. Στην περίπτωση αυτή λέμε ότι η στήλη ProductNo έχει "εξάρτηση πολλαπλών τιμών" (multi value dependent) από τη στήλη PersonNo. Στον ίδιο πίνακα η στήλη Language, επίσης, έχει "εξάρτηση πολλαπλών τιμών" από τη στήλη PersonNo.

Επισήμανση κάποιων μειονεκτημάτων της 3NF

Έστω ότι το προϊόν «010» σταματά να παράγεται. Άρα, πρέπει να διαγράψουμε από τον πίνακα τις γραμμές. Σε περίπτωση διαγραφής μόνο της μίας γραμμής, λόγω λάθους, υπάρχει πρόβλημα ακεραιότητας της βάσης. Ανάλογο πρόβλημα μπορούμε να δούμε στην περίπτωση δήλωσης Insert. Η λύση στο πρόβλημα μας είναι η διάσπαση του πίνακα σε δύο πίνακες: Speaks, Produces (Εικόνα 1.62).

Πίνακας στην τρίτη κανονική μορφή

ΠΙΝΑΚΑΣ EMP_LANG_PROD

PersonNo	Language	ProductNo

100	ΓΕΡΜΑΝΙΚΑ	010
100	ΓΕΡΜΑΝΙΚΑ	020
100	ΓΑΛΛΙΚΑ	010
100	ΓΑΛΛΙΚΑ	020
200	ΑΓΓΛΙΚΑ	010
200	ΑΓΓΛΙΚΑ	030
200	ΓΑΛΛΙΚΑ	010
200	ΓΑΛΛΙΚΑ	030

(PersonNo, Language, ProductNo) κύριο κλειδί

Γραμμές των οποίων η διαγραφή δημιουργεί προβλήματα

100	ΓΕΡΜΑΝΙΚΑ	010
100	ΓΑΛΛΙΚΑ	010

Πίνακες τέταρτης κανονικής μορφής

ΠΙΝΑΚΑΣ SPEAKS

PersonNo	Language
100	ΓΕΡΜΑΝΙΚΑ
100	ΓΑΛΛΙΚΑ
200	ΑΓΓΛΙΚΑ
200	ΓΑΛΛΙΚΑ

(PersonNo, Language)κύριο κλειδί

ΠΙΝΑΚΑΣ PRODUCES

PersonNo	ProductNo
100	010
100	020
200	010
200	030

(PersonNo, ProductNo) κύριο κλειδί

Εικόνα 1.62 Η διαγραφή κάποιων προϊόντων, που δεν παράγονται πλέον, από τον πίνακα της τρίτης κανονικής είναι δυνατόν να οδηγήσει σε προβλήματα τα οποία επιλύονται αν χρησιμοποιήσουμε την τέταρτη κανονική μορφή

Κανόνας

Μια σχέση είναι στην τέταρτη κανονική μορφή (4NF) αν είναι στην τρίτη (3NF) και δεν περιλαμβάνει "εξαρτήσεις πολλαπλών τιμών".

Σημαντική παρατήρηση

Αν ένα πρόσωπο πρέπει να γνωρίζει μία συγκεκριμένη γλώσσα για να παράγει ένα συγκεκριμένο προϊόν τότε δεν πρέπει να διαχωρίσουμε τον πίνακα EMP_LANG_PROD στους δύο πίνακες Speaks, Produces γιατί κάθε γραμμή του περιγράφει ένα τέτοιο γεγονός (Εικόνα 1.63).

PersonNo	Language	ProductNo
100	ΓΕΡΜΑΝΙΚΑ	010
100	ΓΕΡΜΑΝΙΚΑ	020

Εικόνα 1.63 Αν πρέπει να γνωρίζουμε για κάθε πρόσωπο ποια συγκεκριμένη γλώσσα πρέπει να γνωρίζει για να παράγει ένα συγκεκριμένο προϊόν τότε διατηρούμε τον πίνακα

1.17.2 Πέμπτη κανονική μορφή

Έστω ότι ο πίνακας EMP_LANG_PROD (Εικόνα 1.64) ανήκει στο Σύστημα Βάσης Δεδομένων μίας κατασκευαστικής εταιρείας.

Τέταρτη κανονική μορφή

Περιορισμός; κάθε πρόσωπο ποια συγκεκριμένη γλώσσα πρέπει να γνωρίζει για να παράγει ένα συγκεκριμένο προϊόν

ΠΙΝΑΚΑΣ EMP_LANG_PROD

PersonNo	Language	ProductNo
100	ΓΕΡΜΑΝΙΚΑ	010
100	ΓΕΡΜΑΝΙΚΑ	020
100	ΓΑΛΛΙΚΑ	010
100	ΓΑΛΛΙΚΑ	020
200	ΑΓΓΛΙΚΑ	010
200	ΑΓΓΛΙΚΑ	030
200	ΓΑΛΛΙΚΑ	010

(PersonNo, Language, ProductNo) κύριο κλειδί

Πέμπτη κανονική μορφή

ΠΙΝΑΚΑΣ SPEAKS

PersonNo	Language
100	ΓΕΡΜΑΝΙΚΑ
100	ΓΑΛΛΙΚΑ
200	ΑΓΓΛΙΚΑ
200	ΓΑΛΛΙΚΑ

(PersonNo, Language) κύριο κλειδί

ΠΙΝΑΚΑΣ PRODUCES

PersonNo	ProductNo
100	010
100	020
200	010
200	030

(PersonNo, ProductNo) κύριο κλειδί

ΠΙΝΑΚΑΣ LANG_PROD

Language	ProductNo

ΓΕΡΜΑΝΙΚΑ	010
ΓΕΡΜΑΝΙΚΑ	020
ΓΑΛΛΙΚΑ	010
ΓΑΛΛΙΚΑ	020
ΑΓΓΛΙΚΑ	010
ΑΓΓΛΙΚΑ	030

(Language, ProductNo) κύριο κλειδί

Εικόνα 1.64 Μετασχηματισμός τέταρτης κανονικής σε πέμπτη κανονική μορφή

Προσοχή!

Σε σχέση με τον πίνακα του παραδείγματος της εικόνας 1.62 θεωρούμε ότι παραλείπεται η τελευταία γραμμή του πίνακα.

Για τον πίνακα αυτό μπορούμε να πούμε ότι βρίσκεται στην τέταρτη κανονική μορφή, δηλαδή ισχύει η σημαντική παρατήρηση της προηγούμενης ενότητας, και το κύριο κλειδί του είναι το σύνθετο κλειδί (PersonNo, Language, ProductNo).

Επισήμανση κάποιων μειονεκτημάτων της 4NF

Η αποσύνθεση (decomposition) της σχέσης (πίνακα) EMP_LANG_PROD στις σχέσεις (πίνακες) Speaks, Products δεν συνεπάγεται ότι μπορούμε να ανασυνθέσουμε την αρχική σχέση με συνδέσεις (join) των δύο νέων σχέσεων. Η ανασύνθεση μπορεί να γίνει μόνο αν προσθέσουμε και τη σχέση Lang_Prod(Language, ProductNo) στις σχέσεις Speaks, Produces.

Η πέμπτη κανονική μορφή έχει τρεις πίνακες (Εικόνα 1.64).

Μετά από την παρατήρηση μπορούμε να δώσουμε ένα κανόνα που αντιμετωπίζει την εξάρτηση σύνδεσης (join dependency) .

Κανόνας

Μια σχέση είναι στην πέμπτη κανονική μορφή (5NF) αν δεν είναι δυνατόν να ανασυνθέσουμε τη σχέση με συνδέσεις (join) απλούστερων σχέσεων (με διαφορετικά κλειδιά).

Σημαντική παρατήρηση

Δυστυχώς στις πραγματικές εφαρμογές δεν είναι πάντοτε ξεκάθαρο αν η τέταρτη κανονική μορφή παραβιάζει την πέμπτη.

1.17.3 Κανονική μορφή BOYCE-CODD

Έστω ότι ο πίνακας STUD_COUR_TEACH (Εικόνα 1.65) ανήκει στο Σύστημα Βάσης Δεδομένων ενός Πανεπιστημίου.

Για τον πίνακα αυτό μπορούμε να πούμε ότι βρίσκεται στην τρίτη κανονική μορφή και το κύριο κλειδί του είναι το σύνθετο κλειδί (Student, Course).

Παρατήρηση

Αν ένας διδάσκων διδάσκει ακριβώς ένα μάθημα θα μπορούσε το χαρακτηριστικό Teacher να καθορίζει το χαρακτηριστικό Course. Δηλαδή, στο παράδειγμα μας έχουμε μία (σχετικά ασυνήθιστη) περίπτωση όπου χαρακτηριστικό εκτός κλειδιού ορίζει τμήμα του σύνθετου κλειδιού ενός πίνακα της Τρίτης Κανονικής Μορφής.

Επισήμανση κάποιων μειονεκτημάτων της 3NF

Η διαγραφή ενός σπουδαστή π.χ.. του Μαυρουδάκη οδηγεί στην απώλεια της πληροφορίας ότι ο καθηγητής Καστανός διδάσκει Prolog. Το πρόβλημα λύνεται αν θεωρήσουμε την αποσύνθεση (decomposition) της σχέσης (πίνακα) σε δύο σχέσεις (πίνακες) Student (Student, Teacher), Cour_Teach (Teacher, Course). Η κανονική μορφή Boyce-Codd έχει δύο πίνακες:

Τρίτη κανονική μορφή

ΠΙΝΑΚΑΣ STUD_COUR_TEACH

Student	Course	Teacher
ΔΟΥΜΑ	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ	ΣΚΟΥΡΛΑΣ
ΜΑΥΡΟΥΔΑΚΗΣ	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ	ΜΑΥΡΟΣ
ΠΑΠΟΥΤΣΗΣ	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ	ΜΑΥΡΟΣ
ΔΟΥΜΑ	PROLOG	ΞΑΝΘΟΣ
ΜΑΥΡΟΥΔΑΚΗΣ	PROLOG	ΚΑΣΤΑΝΟΣ

(Student, Course) κύριο κλειδί

Η διαγραφή γραμμών οδηγεί σε προβλήματα

Student	Course	Teacher
ΔΟΥΜΑ	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ	ΣΚΟΥΡΛΑΣ
ΠΑΠΟΥΤΣΗΣ	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ	ΜΑΥΡΟΣ

Κανονική μορφή Boyce-Codd

ΠΙΝΑΚΑΣ STUDENT

Student	Teacher
ΔΟΥΜΑ	ΣΚΟΥΡΛΑΣ
ΜΑΥΡΟΥΔΑΚΗΣ	ΜΑΥΡΟΣ
ΠΑΠΟΥΤΣΗΣ	ΜΑΥΡΟΣ
ΔΟΥΜΑ	ΞΑΝΘΟΣ
ΜΑΥΡΟΥΔΑΚΗΣ	ΚΑΣΤΑΝΟΣ

(Student, Teacher) κύριο κλειδί

ΠΙΝΑΚΑΣ COUR_TEACH

Teacher	Course
ΣΚΟΥΡΛΑΣ	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ
ΜΑΥΡΟΣ	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ
ΞΑΝΘΟΣ	PROLOG
ΚΑΣΤΑΝΟΣ	PROLOG

Teacher κύριο κλειδί

Εικόνα 1.65 Τρίτη κανονική μορφή και κανονική μορφή Boyce-Codd

Μετά από την παρατήρηση μπορούμε να δώσουμε τους ορισμούς και τους κανόνες που οδηγούν σε σχεδίαση που αντιμετωπίζει το πρόβλημα.

Ορισμός

Ένα συναρτησιακό συστατικό (functional determinant) είναι ένα χαρακτηριστικό από το οποίο ένα άλλο χαρακτηριστικό είναι (συναρτησιακά) εξαρτώμενο.

Παράδειγμα

Έστω η σχέση FIRST(SupplierNo, Status, Scity, Partno, Quantity).

Τα χαρακτηριστικά της SupplierNo, (SupplierNo, Partno) είναι συναρτησιακά χαρακτηριστικά επειδή τα χαρακτηριστικά Status, Scity, Quantity είναι (συναρτησιακά) εξαρτώμενα από αυτά.

Κανόνας

Μία σχέση R είναι στην κανονική μορφή Boyce-Codd (BCNF) αν και μόνο αν κάθε συναρτησιακό συστατικό της είναι ένα υποψήφιο κύριο κλειδί.

Σημαντική παρατήρηση

Αν μία σχέση είναι στη κανονική μορφή (BCNF) τότε είναι και στην τρίτη κανονική μορφή.

Για μία πληρέστερη αναφορά στο θέμα της BCNF δείτε και το κεφάλαιο 3.

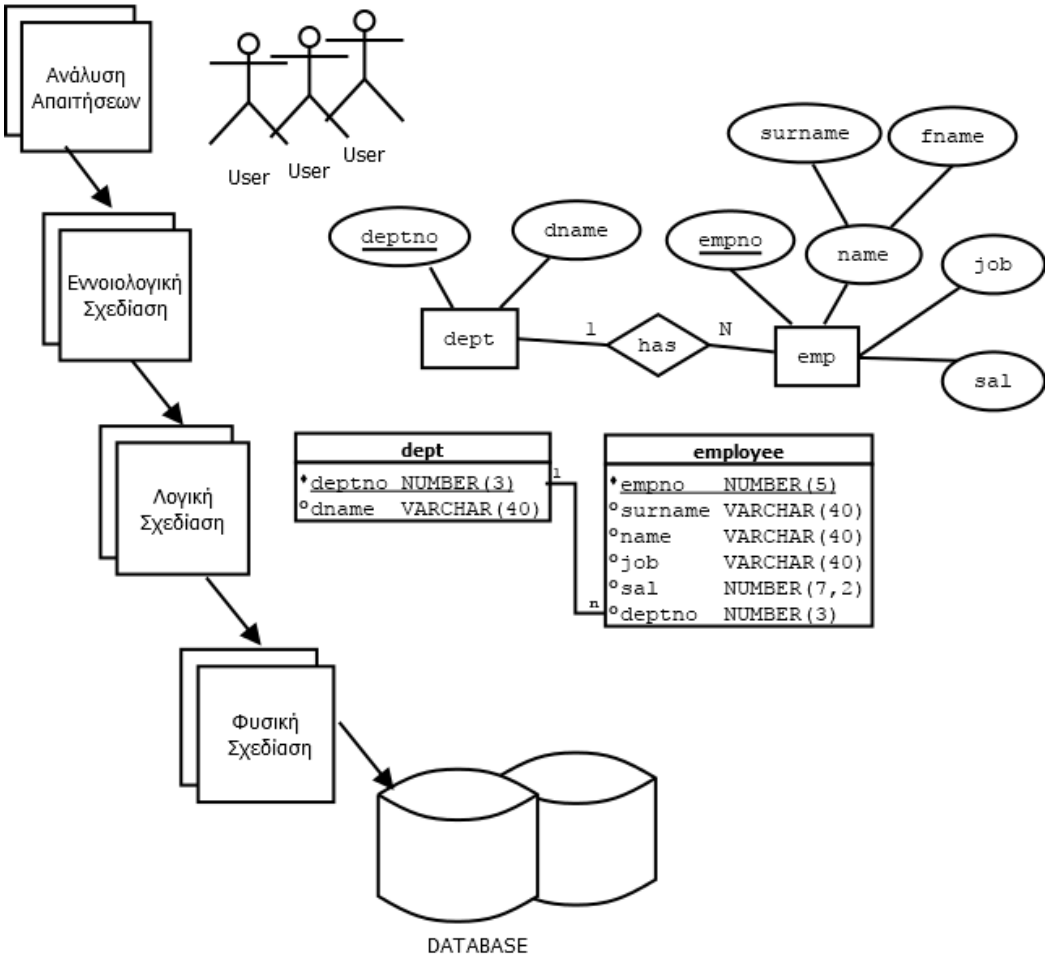
Στο κεφάλαιο 6 αφού μελετήσουμε λεπτομερέστερα το σχεσιακό μοντέλο θα προχωρήσουμε και σε μία πιο "αυστηρή" εξέταση της συναρτησιακής εξάρτησης αλλά και σε μία μεθοδική αντιμετώπιση του θέματος της κανονικοποίησης.

1.18 Κύρια παραδοτέα της Σχεδίασης Βάσης Δεδομένων

Παραθέτουμε ένα χαρακτηριστικό απόσπασμα από άρθρο του Alhajj R. (2003) για το ΜΟΣ:

"In traditional systems analysis and design, the result of the data analysis phase is the entity-relationship (ER) diagram, which is transformed into relations (tables) in the relational model. As the conventional relational model is concerned, entities and some relationships generate relations. These relationships include those with attributes, many-to-many (M:M) and n-ary (n≥3) relationships (those that connect three or more entities). The corresponding relation includes the primary key of each of the participating entities and the additional attributes, if any. The remaining two types of relationships, one-to-one (1:1) and many-to-one (M:1) without attributes, result in adding attribute(s) that constitute the primary key of one relation to the other."

(Alhajj R. (2003), Extracting the extended entity-relationship model from a legacy relational database, Information Systems, 28(6), 597–618)



Εικόνα 1.66 Τα κύρια παραδοτέα της σχεδίασης βάσης δεδομένων

Κύρια παραδοτέα της Σχεδίασης Βάσης Δεδομένων, ακριβέστερα της ανάλυσης δεδομένων (data analysis), είναι (Εικόνα 1.66):

1. **Ανάλυση απαιτήσεων (requirement analysis).** Περιγράφουμε τι θέλουμε να κάνει το σύστημα. Η περιγραφή αυτή θα μας χρειαστεί στη συνέχεια. Πιο συγκεκριμένα, καταγράφουμε τις απαιτήσεις της Διοίκησης και του προσωπικού του οργανισμού που αφορούν τη σχεδίαση της βάσης δεδομένων και τις εφαρμογές της επιχείρησης και τους περιορισμούς των εφαρμογών (constrains, business rules)
2. **Εννοιολογική σχεδίαση (Conceptual Data Design).** Βασιζόμαστε στην περιγραφή που κάναμε και προχωρούμε στην κατασκευή εννοιολογικού μοντέλου, π.χ., Μοντέλου Οντοτήτων-Συσχετίσεων (Entity-Relationship Model)
3. **Λογική σχεδίαση (Logical Data Design).** Σχεδιάζουμε τη Σχεσιακή βάση δεδομένων (Relational database). Καταλήγουμε συνήθως στην Τρίτη κανονική μορφή (Third Normal Form) ή την Κανονική Μορφή Boyce-Codd. Πιο περιγραφικά, ξεκινώντας από το ΜΟΣ εφαρμόζουμε μια συνταγή και κατασκευάζουμε τους πίνακες της Σχεσιακής βάσης δεδομένων.
4. **Φυσική σχεδίαση (Physical Data Design).** Καταγράφουμε την οργάνωση του χώρου των πινάκων (tablespaces), των ευρετηρίων (indexes), των συστάδων πινάκων (clusters) κ.λπ. Δηλαδή, οργανώνουμε καλύτερα τα στοιχεία των πινάκων ώστε να τα ανακτούμε εύκολα και γρήγορα.

Κεφάλαιο 2
Τύποι Συστημάτων Βάσεων Δεδομένων, σύγχρονες τεχνολογίες και εφαρμογές. Διαχείριση δεδομένων μεγάλης κλίμακας

Σύνοψη
*Στο κεφάλαιο αυτό σκιαγραφείται ο σύγχρονος τομέας των συστημάτων βάσεων δεδομένων και των εφαρμογών τους. Εστιάζει στα δεδομένα και στα δεδομένα μεγάλης κλίμακας (*big data*) και τα συστήματα διαχείρισης δεδομένων, «παραδοσιακά» και νεότερα. Παρουσιάζονται μοντέλα οργάνωσης δεδομένων, επεξηγούνται έννοιες της ανάκτησης δεδομένων και αναλύεται η έννοια των μεταδεδομένων, τα σχήματα μεταδεδομένων και η σημασία τους. Γίνεται αναφορά στα ανοικτά και συνδεδεμένα δεδομένα, σε θέματα ανακάλυψης νέας σημασιολογικής γνώσης και σημασιολογικού ιστού (*semantic web*) και σε έννοιες όπως τα ενιαία αναγνωριστικά ονόματα (URIs), οι γλώσσες σήμανσης, π.χ. γλώσσες XML, HTML, η οικογένεια προτύπων RDF, οι οντολογίες (*Ontology*), το πρότυπο OWL, τα ελεγχόμενα λεξιλόγια.*

*Ο όρος δεδομένα μεγάλης κλίμακας περιγράφει μεγάλους όγκους δεδομένων οι οποίοι προκύπτουν με υψηλή ταχύτητα, και αποτελούν σύνθετα και μεταβλητά ετερογενή δεδομένα, δομημένα (*structured*) και αδόμητα (*unsturctured*).*

Οι παραδοσιακές τεχνολογίες οργάνωσης και διαχείρισης δεδομένων είναι καλά εδραιωμένες αλλά τα μεγάλα δεδομένα απαιτούν επιπλέον προηγμένες τεχνικές και τεχνολογίες για να καταστεί δυνατή η συλλογή, αποθήκευση, διανομή, διαχείριση και ανάλυση των πληροφοριών.

*Στο κεφάλαιο περιλαμβάνονται, τα συστήματα διαχείρισης συναλλαγών (*transaction*), βάσεις δεδομένων στο διαδίκτυο και πρότυπα σημασιολογικού ιστού, πολυμέσα και πολυτροπικές βάσεις δεδομένων, συστήματα ανάκτησης πληροφοριών-κειμένου και μηχανές αναζήτησης, συστήματα ανάκτησης εικόνων βασιζόμενα στο περιεχόμενό τους, κατανεμημένες βάσεις δεδομένων και βάσεις στο νέφος, ενεργές βάσεις δεδομένων. Περιγράφονται, επίσης, σύγχρονες εξελίξεις του τομέα των βάσεων δεδομένων και μελλοντικές τάσεις. Τέλος, σκιαγραφείται η σχέση βάσεων δεδομένων και επιχειρήσεων με αναφορές σε τομείς, όπως, εξόρυξη δεδομένων, αποθήκες δεδομένων, επιχειρηματική ευφυΐα, διαχείριση γνώσης.*

Το κεφάλαιο περιλαμβάνει τις παρακάτω ενότητες:

1) *Αρχιτεκτονική εφαρμογών βάσης δεδομένων. Συναλλαγές (*transactions*) και η σημασία τους για τις εφαρμογές.*
2) *Το παρόν και το μέλλον των εφαρμογών βάσεων δεδομένων. Ο ρόλος των δεδομένων μεγάλης κλίμακας. Διαχείριση δεδομένων μεγάλης κλίμακας.*
3) *Ο ρόλος του Σημασιολογικού ιστού. Πρότυπα, γλώσσες σήμανσης, XML, JSON, οντολογίες κ.λπ.*
4) *Μορφές πολυμέσων και Πολυτροπικές βάσεις δεδομένων (*multimodal databases*)*
5) *Συστήματα Ανάκτησης Πληροφοριών-ΣΑΠ (*Information Retrieval systems*), Συστήματα Ανάκτησης Κειμένου (*Text Retrieval*) και Μηχανές Αναζήτησης*
6) *Συστήματα Ανάκτησης Εικόνων Βασιζόμενα στο Περιεχόμενό τους. Δεικτοδότηση και Ανάκτηση Εικόνας (Content Based Information Retrieval systems)*
7) *Κατανεμημένες βάσεις δεδομένων (*distributed databases*). Βάσεις στο νέφος (*cloud databases*)*
8) *Ενεργές βάσεις δεδομένων (*active databases*) και τεχνολογία εναυσμάτων (*triggers*)*
9) *Βάσεις Δεδομένων και Επιχειρήσεις. Data Mining, Data Warehouse, Datamart, Business Intelligence, Knowledge Management*

10) Βάσεις Δεδομένων NoSQL

Προαπαιτούμενη γνώση
Προτείνεται η μελέτη του κεφαλαίου 1 του παρόντος συγγράμματος.

2.1 Δεδομένα πληροφορία και γνώση και συστήματα οργάνωσης δεδομένων και γνώσεων. Αρχή με μελέτη περίπτωσης

Η παγκοσμιοποίηση και η εξέλιξη της Τεχνολογίας Πληροφορικής (Τ.Π). έχει ως συνέπεια ο όγκος των επιχειρηματικών δεδομένων, καθώς και η γεωγραφική κατανομή τους, να προσεγγίζει μια κλίμακα πρωτόγνωρη στη βιομηχανία λογισμικού. Οι παρακάτω τεχνολογίες γνωρίζουν ευρύτατη αποδοχή και συνδέονται με τα συστήματα διαχείρισης δεδομένων στις σύγχρονες επιχειρήσεις (Μαρινάγη και Σκουρλάς, 2022):

Βάσεις δεδομένων (Databases): Η βάση δεδομένων είναι η κεντρική τεχνολογική υποδομή για κάθε επιχείρηση και σχεδόν για κάθε είδος επιχειρηματικό λογισμικό. Γενικευμένη είναι η χρήση των σχεσιακών συστημάτων βάσης δεδομένων και ανερχόμενη η χρήση προϊόντων noSQL (not only SQL)

Αποθήκες δεδομένων (Data warehouses): Οι μεγάλες επιχειρήσεις δημιουργούν αποθήκες δεδομένων τις οποίες τροφοδοτούν από τις βάσεις δεδομένων τους και εξωτερικά δεδομένα . Οι αποθήκες περιλαμβάνουν πολλά δεδομένα και βελτιστοποιούνται για πολύ γρήγορους χρόνους ανάγνωσης. Κύριος στόχος της λειτουργίας τους είναι η υποβοήθηση των επιχειρηματικών αποφάσεων. Τεχνολογίες όπως η εικονικοποίηση (virtualization) και η υπολογιστική νέφους (cloud computing) παρέχουν νέες δυνατότητες στη διαχείριση αποθηκών στη σύγχρονη επιχείρηση.

Επιχειρηματική ευφυΐα (Business intelligence): Η επιχειρηματική ευφυΐα (ή επιχειρηματική νοημοσύνη) αναφέρεται στις εφαρμογές λογισμικού οι οποίες χειρίζονται δεδομένα μέσα σε αποθήκες δεδομένων και στηρίζουν τις αποφάσεις (decision support) με αναφορές (reports) για θέματα χρηματοδότησης, πωλήσεων και γενικότερα για τις δραστηριότητες οποιασδήποτε μεγάλης επιχείρησης.

Διαχείριση κύκλου ζωής πληροφοριών (Information Lifecycle Management): Οι περισσότερες ώριμες επιχειρήσεις χρησιμοποιούν τα εργαλεία αυτά για την αρχειοθέτηση παλαιότερων δεδομένων και με τον τρόπο αυτό διασφαλίζουν ότι τα δεδομένα είναι προσβάσιμα σε περίπτωση ελέγχου κ.λπ.

Συστήματα Διαχείρισης Περιεχομένου (Content Management Systems − CMS): Αυτά τα συστήματα είναι τα κεντρικά αποθετήρια για επιχειρηματικά έγγραφα και υποστηρίζουν παρακολούθηση και έλεγχο των εκδόσεων των εγγράφων (versioning) και επιλεκτική διάδοση πληροφορίας (selective dissemination of information) από το κοινό σύστημα αποθήκευσης.

Enterprise Resource Planning (ERP): Κεντρικές επιχειρηματικές λειτουργίες όπως οικονομική διαχείριση, η διαχείριση ανθρωπίνων πόρων και τα συστήματα διαχείρισης εφοδιαστικής αλυσίδας, logistics κ.λπ. συνήθως κατηγοριοποιούνται ως συστήματα διαχείρισης επιχειρησιακών πόρων (ERP).

Ενσωμάτωση (Integration): Η ενσωμάτωση δεδομένων, η ολοκλήρωση διαδικασιών και οι τεχνολογίες ενσωμάτωσης μηνυμάτων χρησιμοποιούνται συνήθως για να βοηθήσουν τα υπόλοιπα συστήματα λογισμικού να συνεργαστούν.

Επιχειρηματική αναζήτηση (business retrieval). Μια μηχανή αναζήτησης μπορεί να δημιουργήσει ένα μεγάλο ευρετήριο περιεχομένου και λέξεων κλειδιών που μπορούν εύκολα να αναζητηθούν επιτρέποντας έτσι στους ανθρώπους να αναζητούν αντιστοιχίσεις κειμένου στα δεδομένα (text matches in the data).

Για την καλύτερη κατανόηση της μελέτης περίπτωσης παραθέτουμε κάποια πρώτη παρουσίαση εννοιών και όρων οι οποίοι θα μας απασχολήσουν όχι μόνο στη μελέτη περίπτωσης αλλά και σε όλο το κεφάλαιο.

2.1.1 Εισαγωγή στα είδη δεδομένων, βασικές έννοιες και όροι (Glossary)

Τα δεδομένα διακρίνονται σε δύο κύριες κατηγορίες, δομημένα (structured) και μη δεδομένα (unstructured). (Σημείωση. Στη βιβλιογραφία υπάρχει και μία τρίτη κατηγορία δεδομένων τα ημι-δομημένα-semi-structured).

Παράδειγμα βάσης δεδομένων η οποία περιλαμβάνει δομημένα και μη δομημένα δεδομένα αποτελεί η βάση δεδομένων MIMIC-III (Medical Information Mart for Intensive Care, https://mmic.physionet.org/). Είναι μια δωρεάν διαθέσιμη σχεσιακή βάση δεδομένων που αναπτύχθηκε από το MIT Lab for Computational Physiology. Τα δεδομένα καταγράφηκαν από περισσότερες από 90 ΜΕΘ (ICU-Intensive Care Unit) και η βάση περιλαμβάνει δεδομένα υγείας που σχετίζονται με περισσότερους από 50.000 ασθενείς εντατικής θεραπείας. Η βάση περιλαμβάνει δημογραφικά στοιχεία, εργαστηριακές εξετάσεις, φάρμακα κ.λπ. και χρησιμοποιείται ευρέως σε όλο τον κόσμο στην ακαδημαϊκή έρευνα, την εκπαίδευση και τη βιομηχανία. Η ανάκτηση στοιχείων γίνεται με χρήση γλώσσας SQL:

```
SELECT *
FROM patients
INNER JOIN admissions
ON patients.subject_id = admissions.subject_id
WHERE gender = 'F'
AND patients.subject_id = 40080;
```

Είναι ενδιαφέρον να παραθέσουμε στοιχεία για τα δεδομένα αυτά.

«Τα δεδομένα για κάθε περιστατικό (ασθενή) περιλαμβάνουν σήματα και περιοδικές μετρήσεις που λαμβάνονται από ένα μόνιτορ δίπλα στο κρεβάτι καθώς και κλινικά δεδομένα που λαμβάνονται από τον ιατρικό φάκελο του ασθενούς. Οι μαγνητοσκοπήσεις ποικίλλουν σε διάρκεια από 20 έως 40+ ώρες. Συνολικά, η βάση δεδομένων περιέχει σχεδόν 200 ημέρες ασθενούς με σήματα σε πραγματικό χρόνο και συνοδευτικά δεδομένα.

Κάθε εγγραφή ασθενούς αποτελείται από εκατοντάδες μεμονωμένα αρχεία. Τα δεδομένα που λαμβάνονται από μόνιτορ χωρίζονται σε αρχεία κάθε ένα των οποίων περιέχει 10 λεπτά ηχογραφημένων σημάτων. Τα σήματα μπορούν να συνενωθούν χωρίς κενά για να σχηματίσουν συνεχή εγγραφή.»

("The data in each case include signals and periodic measurements obtained from a bedside monitor as well as clinical data obtained from the patient's medical record. The recordings vary in length; almost all of them are at least 20 hours, and many are 40 hours or more. In all, the database contains nearly 200 patient-days of real-time signals and accompanying data.

Each record typically consists of several hundred individual files. The data obtained from the bedside monitors are divided into files each containing 10 minutes of recorded signals, which can then be assembled without gaps to form a continuous recording. For this reason, each record is kept in a separate directory, named after the record it contains.)"

https://mmic.physionet.org/

Περισσότερα στοιχεία μπορείτε να μελετήσετε στις παρακάτω πηγές:

Moody, G.B., Mark, R.G. (1996) A Database to Support Development and Evaluation of Intelligent Intensive Care Monitoring. Computers in Cardiology 23:657–660 (περιγραφή της βάσης δεδομένων)

Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation 101(23): e215-e220 [Circulation Electronic Pages; http://circ.ahajournals.org/content/101/23/e215.full]; 2000 (June 13) (τυπική αναφορά για PhysioNet)

Τα δεδομένα κειμένου (textual data) αποτελούν τη συντριπτική πλειοψηφία του όγκου των δεδομένων σήμερα. Σύμφωνα με μια ενδιαφέρουσα αναλογία «αν όλα τα δεδομένα στον κόσμο ήταν ισοδύναμα με το νερό στη γη, τότε τα δεδομένα κειμένου είναι σαν το νερό των ωκεανών, δηλαδή αποτελούν την πλειοψηφία του όγκου» ("if all the data in the world was equivalent to the water on earth, then textual data is like the ocean, making up a majority of the volume") (Siegel, 2016). Δείτε σχετικά και το παρακάτω βιβλίο.

Eric Siegel (2016) Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die, Revised and Updated, ISBN: 978-1-119-14567-7, Wiley

Η ανάλυση κειμένου (Text analytics) καθοδηγείται από την ανάγκη επεξεργασίας της φυσικής ανθρώπινης γλώσσας (Natural Language Processing- NLP)). Η φυσική γλώσσα δεν υπάρχει σε «δομημένη» μορφή, δηλαδή τα δεδομένα (το «ελεύθερο» κείμενο) δεν αποτελούνται από γραμμές και στήλες (σύμφωνα με την ορολογία των σχεσιακών ΣΔΒΔ) ή παραδείγματα (examples) και χαρακτηριστικά (attributes) (σύμφωνα με την ορολογία της εξόρυξης δεδομένων). Η εξόρυξη κειμένου αξιοποιεί την επεξεργασία φυσικής γλώσσας (NLP) για να μετατρέψει το «ελεύθερο» (μη δομημένο) κείμενο σε έγγραφα και βάσεις δεδομένων. Τα δομημένα δεδομένα επιτρέπουν την ανάλυση και τη χρήση αλγορίθμων μηχανικής μάθησης (ML).

Η ανάλυση δεδομένων (Analytics) είναι η διαδικασία με την οποία παρέχονται με τον απαιτούμενο ασφαλή και αποδοτικό τρόπο (security and performance) στους ανθρώπους της επιχείρησης έγκυρα και περιεκτικά δεδομένα (accurate and comprehensive data), όταν τα χρειάζονται, γεγονός που τους επιτρέπει να επικεντρωθούν στις προκλήσεις του τομέα και τη λήψη ορθών επιχειρηματικών αποφάσεων.

Τα τελευταία χρόνια υπάρχει αυξανόμενο ενδιαφέρον για την ανάλυση μεγάλων δεδομένων (Big Data Analysis, Big Data Analytics, BDA) και τις εφαρμογές της. Η τεχνητή νοημοσύνη (Artificial intelligence) και ιδιαίτερα η μηχανική μάθηση (Machine Learning-ML), η επεξεργασία φυσικής γλώσσας (Natural Language Processing), και η βαθιά μάθηση (Deep Learning) έχουν σημαντικό ρόλο στην ανάπτυξη του τομέα BDA.

Η μηχανική μάθηση (Machine Learning) εστιάζει σε εργαλεία και τεχνικές για τη δημιουργία μοντέλων τα οποία εφαρμόζονται στα δεδομένα. Περιλαμβάνει αλγόριθμους που μπορούν να εκπαιδευτούν και να κάνουν πρόβλεψη, εποπτευόμενη μάθηση (supervised learning), και αλγόριθμους που διαβάζουν δεδομένα και προσπαθούν να τα ταξινομήσουν κ.λπ. μη εποπτευόμενη μάθηση (unsupervised learning)

Η επιστήμη δεδομένων (data science) μελετά τα δεδομένα και τρόπους εξαγωγής νοήματος από αυτά και χρησιμοποιεί μεθόδους μηχανικής μάθησης και στατιστικής ανάλυσης. Οι επαγγελματίες της επιστήμης δεδομένων (data scientist) επικεντρώνονται συχνά στη συλλογή και επεξεργασία μεγάλων δεδομένων χρησιμοποιώντας διάφορα εργαλεία και τεχνικές.

2.1.2 Μονάδα Εντατικής Θεραπείας και διαχείριση μεγάλων δεδομένων

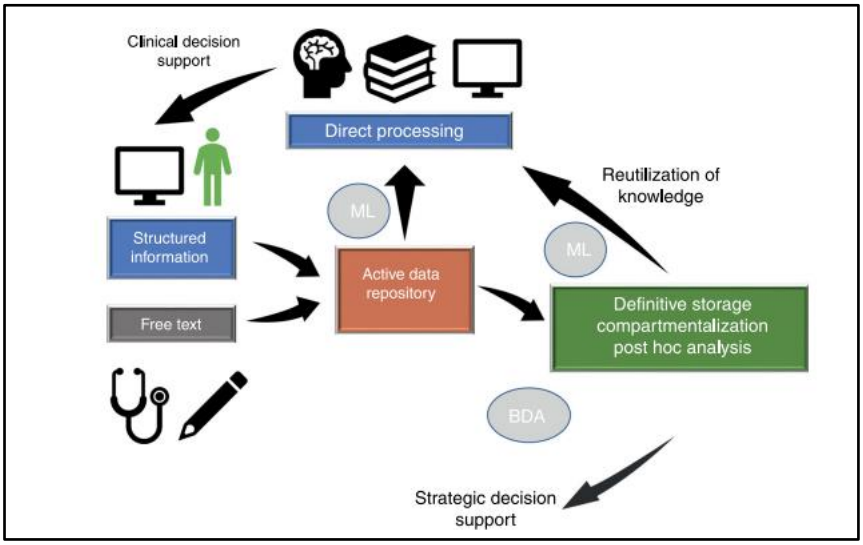
Η σύγχρονη ΜΕΘ-Μονάδα Εντατικής Θεραπείας συνδέεται με την επεξεργασία και αποθήκευση σε ηλεκτρονική μορφή μεγάλου όγκου δεδομένων και πληροφοριών, που αποτελούν αντικείμενο εκτεταμένης έρευνας και εφαρμογών σε πολλούς τομείς, όπως η υποστήριξη κλινικών αποφάσεων, η κλινική έρευνα κ.λπ. Οι Reiz, de la Hoz και Garcia, (Reiz, 2019) εξετάζουν και αξιολογούν διαφορετικές ερευνητικές τεχνικές της ανάλυσης μεγάλων δεδομένων και της μηχανικής μάθησης και διερευνούν πώς αυτές οι τεχνικές θα μπορούσαν να χρησιμοποιηθούν στον τομέα της ΜΕΘ. Προτείνουν επίσης τη συνεργασία των κλινικών γιατρών με έναν «επαγγελματία της επιστήμης των δεδομένων» (data scientist) ειδικευμένο στην ανάλυση δεδομένων του τομέα της υγείας που θα συμβάλει στην ανάπτυξη τεχνικών ανάλυσης μεγάλων δεδομένων και μηχανικής μάθησης. Χρησιμοποιούν ένα διάγραμμα ροής δεδομένων (data flow diagram) (εικόνα 2.1) για την απεικόνιση της ροής δεδομένων στην εντατική θεραπεία με στόχο την υποστήριξη κλινικών και στρατηγικών αποφάσεων, περιγράφουν δομημένες και μη δομημένες πληροφορίες ως εισροές και προτείνουν τη χρήση μηχανικής μάθησης (ML), ανάλυση μεγάλων δεδομένων (BDA) και, επιπλέον,

εξαγωγής/επαναχρησιμοποίησης γνώσης. Στην προσέγγισή τους, τα δεδομένα αποθηκεύονται σε ένα αποθετήριο (repository) χρησιμοποιώντας δεδομένα από τοπικές και κοινόχρηστες εξωτερικές βάσεις δεδομένων. Το συμπέρασμα είναι ότι τα μεγάλα δεδομένα μας επιτρέπουν να βελτιώσουμε τις διαδικασίες φροντίδας και να εξάγουμε γνώση από τα δεδομένα.

Η εικόνα 2.1 συνοδεύεται από την παρατήρηση ότι η ποιότητα παροχής φροντίδας και η στρατηγική απόφαση της διαχείρισης του περιστατικού εξαρτώνται από δύο παράγοντες:

- την επεξεργασία της πληροφορίας η οποία υπάρχει σε «ενεργό αποθετήριο δεδομένων (active data repository), το οποίο είναι μία (ενεργή) βάση δεδομένων στην οποία αποτίθεται «η πληροφορία που δημιουργείται κάθε στιγμή και η οποία είναι διαθέσιμη για λήψη αποφάσεων» (“information that is being generated in each moment and which is available for decision making”)
- την επαναχρησιμοποίηση της γνώσης η οποία είναι αποθηκευμένη σε τοπικές ή κοινές βάσεις δεδομένων.

Η αυτοματοποιημένη επεξεργασία πληροφοριών μπορεί να βασιστεί στη μηχανική μάθηση (ML) και η δυνατότητα ταχείας πρόσβασης σε μεγάλους όγκους δεδομένων ετερογενούς δομής μπορεί να βασιστεί στην ανάλυση μεγάλων δεδομένων (BDA).



Εικόνα 2.1 Ροή δεδομένων στη ΜΕΘ (Dataflow in intensive care medicine) (Reiz et.al) (Celi et.al, 2013)

Οι Reiz et al. βάσισαν την εικόνα 2.1 στο άρθρο:

Celi, L.A., Mark, R.G., Stone, D.J., Montgomery, R.A. (2013) Big Data in the intensive care unit. Closing the data loop. Am J Respir Crit Care Med. 187(11), 1157-60.

Είναι ενδιαφέρον να δούμε τη σχέση μη δομημένου κειμένου και δομημένων/ημιδομημένων δεδομένων. Οι Riez et al., παρατηρούν ότι ο ιατρός κατανοεί εύκολα μία παράγραφο γραμμένη σε ελεύθερο κείμενο (textual data) όπως η παρακάτω:

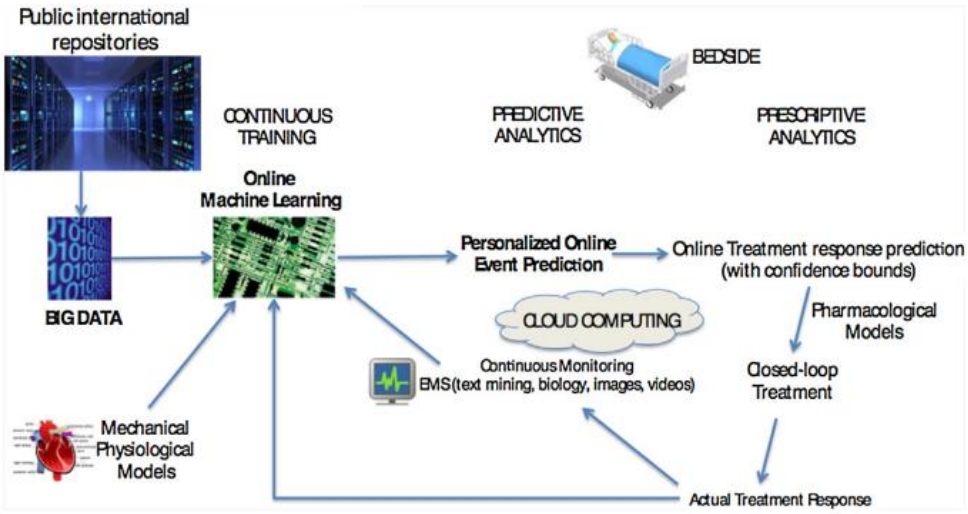
‘‘A 58-year-old male reporting one hour ago to the emergency room due to dyspnea and central chest pain irradiating to the left arm. . .’’.

Αναφέρουν ότι στην παράγραφο των 23 λέξεων υπάρχουν τουλάχιστον 8 έννοιες που μπορούν να εκφραστούν με δομημένο τρόπο και παραθέτουν έγγραφο xml (Εικόνα 2.2).

```
<Episode id=xxxx>
  <Patient>
    <ID>xxxx</ID>
    <Sex>Male</Sex>
    <Age unit = ''years''>58</Age>
  </Patient>
  <Place>Emergencies</Place>
  <Moment>2018-08-12 15:30:00</Moment>
  <Registry date<2018-08-12 16:30:00</Registry date >
  <Symptoms>
    <Dyspnea/>
    <Pain>
      <Location>Central chest</Location>
      <Irradiation>Left arm</Irradiation>
    </Pain>
  </Symptoms>
</Episode>
```

Εικόνα 2.2 Έγγραφο XML (Riez et al., 2019)

Οι Pirracchio et al. (2019) πιστεύουν ότι η τεχνολογία μεγάλων δεδομένων, η μηχανική μάθηση και η επιστήμη των δεδομένων προσφέρουν νέες ευκαιρίες για τη θεραπεία ασθενών και τις ιατρικές αποφάσεις στη μονάδα εντατικής θεραπείας. Υπογραμμίζουν επίσης πόσο σημαντικό είναι να υπάρχει ένα πλαίσιο βασισμένο σε λεπτομερή ολοκληρωμένα δεδομένα ασθενών, το οποίο να επιτρέπει εξατομικευμένες αναλύσεις και θα ενσωματώνει και θα αξιοποιεί τεχνικές ML. Προτείνουν επίσης μια πρώτη (preliminary) αρχιτεκτονική για τη ΜΕΘ του μέλλοντος (εικόνα 2.3). η οποία θα υποστηρίζει εξατομικευμένη ανάλυση των στοιχείων με στόχο την πρόβλεψη της εξέλιξης του περιστατικού και τη θεραπεία (συνταγογράφηση, κ.λπ.). Αυτή η αρχιτεκτονική βασίζεται 1) σε ενσωματωμένα-ενοποιημένα (integrated) μεγάλα δεδομένα, στα οποία περιλαμβάνονται και (μεγάλα) δεδομένα από μόνιτορ δίπλα στον ασθενή (bedside monitoring), και 2) λαμβάνει υπόψη έργα και βάσεις δεδομένων τελευταίας τεχνολογίας, π.χ. MIMIC (Medical Information Mart for Intensive Care), οι οποίες περιλαμβάνουν δεδομένα επιμελημένα από ειδικούς.



Εικόνα 2.3 Μελλοντική ΜΕΘ και η λειτουργία της (Pirracchio et al. 2019)

Η διείσδυση της τεχνητής νοημοσύνης σε κάθε πτυχή της καθημερινότητάς μας, οι υψηλές προσδοκίες σχετικά με τις δυνατότητες της ML και τα μεγάλα δεδομένα είναι θετικοί παράγοντες που μπορούν να αξιοποιηθούν για τη βελτίωση της φροντίδας των ασθενών και στη ΜΕΘ. Υπάρχουν όμως προς επίλυση προβλήματα διαλειτουργικότητας μεταξύ πλατφορμών, νομικοί φραγμοί και ζητήματα αξιοπιστίας δεδομένων που αποτελούν αρνητικούς παράγοντες για την αξιοποίηση και την προσβασιμότητα των gigabyte δεδομένων που παράγονται στις ΜΕΘ.

Το κρίσιμο σημείο για την αξιοποίηση της ανάλυσης δεδομένων σε ασθενείς ΜΕΘ είναι η επεξεργασία των μεγάλων δεδομένων που παράγονται από μόνιτορ δίπλα στο κρεβάτι και από ηλεκτρονικά ιατρικά αρχεία (medical records) και η αξιοπιστία των εφαρμοζόμενων αλγορίθμων ML οι οποίοι πρέπει να παράγουν αξιόπιστες εκτιμήσεις. Η υπολογιστική νέφους (cloud computing) και η εξέλιξη της κατανεμημένης επεξεργασίας (distributed computing) αποτελούν θετικούς παράγοντες στην επίλυση προβλημάτων και δεν απαιτούν από τα νοσοκομεία να έχουν πρόσβαση σε δική τους αποκλειστικά τεχνολογία. Στην αρχιτεκτονική που απεικονίζεται στη εικόνα 2.3 οι ερευνητές πιστεύουν ότι η σχετική τεχνολογία έχει ωριμότητα η οποία επιτρέπει την ανάπτυξη ευφυών διαδικτυακών «μαθητών» (online learners), δηλαδή προγραμμάτων τα οποία θα είναι πιθανώς ενσωματωμένα στα μόνιτορ δίπλα στο κρεβάτι και θα πληροφορούν το εξειδικευμένο ιατρικό προσωπικό της ΜΕΘ ώστε να παρέχεται εξατομικευμένη ιατρική φροντίδα.

2.1.3 Περιγραφή του προβλήματος (Problem's outline)

Σε μια ΜΕΘ είναι σημαντικό να έχουμε πρόσβαση σε πραγματικό χρόνο σε όλα τα δεδομένα που παράγονται μέσα στη ΜΕΘ, δεδομένα από μόνιτορ (bedside monitoring), και σε δεδομένα που παράγονται και εκτός ΜΕΘ, π.χ. δεδομένα ιατρικού φακέλου. Επειδή αυτά τα δεδομένα προέρχονται από διαφορετικές πηγές και πλατφόρμες, θα πρέπει να υποβληθούν σε προεπεξεργασία (data preparation) και να ενσωματωθούν σε ενιαία βάση δεδομένων (data integration) προκειμένου να χρησιμοποιηθούν για την πρόγνωση, τη στήριξη της λήψης αποφάσεων και τη φροντίδα των ασθενών στη ΜΕΘ. Επιπλέον, υπάρχει ανάγκη για τη «χαρτογράφηση» των δεδομένων. Πράγματι, είναι απαραίτητο για την ενοποίηση των δεδομένων να γνωρίζουμε τις διαφορετικές μορφές των δεδομένων που θα ενσωματώσουμε, τα διάφορα πρότυπα για τους κλινικούς όρους, την κωδικοποίηση και την ανταλλαγή πληροφοριών, κ.λπ. Επομένως, είναι απαραίτητο να περιγράψουμε ένα εννοιολογικό πλαίσιο (conceptual framework) του πληροφοριακού συστήματος που θα λύνει τα προβλήματα της ολοκλήρωσης μεγάλων δεδομένων σε πραγματικό χρόνο και θα διασφαλίζει την αποδοτική χρήση τους στις ΜΕΘ. Το εννοιολογικό πλαίσιο που θα παρουσιαστεί στη συνέχεια υιοθετεί και χρησιμοποιεί τεχνικές ML και NLP (κυρίως τεχνολογία επεξεργασίας κειμένου, text processing) και βασίζεται στην ενσωμάτωση Big Data. Αποτελείται από πέντε συνιστώσες-ενότητες (componnets):

- «Ολοκλήρωση μεγάλων δεδομένων» για χρήση στη ΜΕΘ,
- «Υπηρεσίες εντατικής θεραπείας» στη ΜΕΘ,
- «Χρήση προτύπων» στη ΜΕΘ,
- «Εφαρμογή μηχανικής μάθησης» στη ΜΕΘ,
- «Εφαρμογή NLP (επεξεργασίας κειμένου κ.λπ.)» στη ΜΕΘ

Πριν από την παρουσίαση του εννοιολογικού μοντέλου παρουσιάζουμε δύο σημαντικά ζητήματα:

1) Ωριμότητα της τεχνολογίας μεγάλων δεδομένων για εφαρμογή σε ΜΕΘ, και
2) Μεθοδολογία για το σχεδιασμό του εννοιολογικού πλαισίου του πληροφοριακού συστήματος.

2.1.3.1 Ωριμότητα τεχνολογίας μεγάλων δεδομένων για ΜΕΘ

Παρά την ανάπτυξη της τεχνολογίας Big Data, σε ένα τμήμα της βιβλιογραφίας υπάρχει κάποιος σκεπτικισμός σχετικά με τις τρέχουσες δυνατότητες των εφαρμογών της και τη χρήση της στη ΜΕΘ. Ενδεικτικά, οι Adibuzzaman et al. (2017) υποστηρίζουν ότι τα μεγάλα δεδομένα φάνηκε να υπόσχονται πολλά στον τομέα της υγειονομικής περίθαλψης γενικά, αλλά τα υπάρχοντα συστήματα δεν είναι ακόμη πλήρως λειτουργικά («είναι ακόμη σε αρχικό στάδιο», "are still in their infancy"). Αυτή την εκτίμησή τους την δικαιολογούν με αναφορά στην ποιότητα των δεδομένων, το πρόβλημα του ιδιωτικού απόρρητου αλλά και σε ρυθμιστικές πολιτικές που επηρεάζουν την κλινική πρακτική και επιβάλλουν πρόσθετους περιορισμούς.

Οι Gonçalves et al. (2017) πιστεύουν ότι ο τομέας της υγειονομικής περίθαλψης είναι ένας από τους πολλά υποσχόμενους τομείς για την εφαρμογή λύσεων μεγάλων δεδομένων. Αλλά και αυτοί τονίζουν ότι παρά τη συνεχή ανάπτυξη της τεχνολογίας Big Data, υπάρχουν λίγοι οργανισμοί στον τομέα της υγειονομικής περίθαλψης που επωφελούνται από τις «πραγματικές δυνατότητες» των Big Data. Επιπλέον, υπογραμμίζουν ότι υπάρχει έλλειψη συγκεκριμένης μεθοδολογίας για την υλοποίηση αρχιτεκτονικών και έργων Big Data. Στην εργασία τους παρουσιάζουν μια αρχιτεκτονική διαχείρισης δεδομένων σε πραγματικό χρόνο για τις ΜΕΘ. Αναφέρονται, επίσης, στην αξιολόγηση ενός συστήματος που βασίζεται σε αυτήν την αρχιτεκτονική στο Centro Hospitalar do Porto. Γενικότερα στη βιβλιογραφία προτείνονται διάφορες προσεγγίσεις οι οποίες μπορούν να συμβάλουν και στον σχεδιασμό εννοιολογικού πλαισίου για την αξιοποίηση μεγάλων δεδομένων σε ΜΕΘ, π.χ., εμπειρία από έργα διαχείρισης, οπτικοποίηση της ροής των δεδομένων, αρχιτεκτονικές μεγάλων δεδομένων, λίστες ελέγχου (checklists) για τις υπηρεσίες εντατικής θεραπείας, κ.λπ.

2.1.3.2 Η υιοθέτηση μεθοδολογίας για ένα πληροφοριακό σύστημα σε ΜΕΘ

Υπάρχουν πολλές μεθοδολογίες σχεδιασμού πληροφοριακών συστημάτων οι οποίες μπορούν να χρησιμοποιηθούν για τη σχεδίαση ενός «παραδοσιακού» πληροφοριακού συστήματος (ΠΣ) για ΜΕΘ. Με τον όρο «παραδοσιακό» ΠΣ εννοούμε ένα σύστημα το οποίο θα διαχειρίζεται δομημένα δεδομένα ασθενών, τα στοιχεία προγραμματισμού των εισαγωγών, τα στατιστικά στοιχεία κ.λπ. Η αξιοποίηση δεδομένων μεγάλης κλίμακας προσφέρει νέες ευκαιρίες για βελτιωμένες υπηρεσίες εντατικής θεραπείας, καλύτερες αποφάσεις, δυνατότητες πρόβλεψη της πορείας των ασθενών κ.λπ. Από τη μελέτη της βιβλιογραφίας προκύπτει η ανάγκη χρήσης μεθοδολογιών σχεδιασμού πληροφοριακών συστημάτων οι οποίες μπορούν να οδηγήσουν ευκολότερα στην υιοθέτηση καινοτομίας σχετικής με την τεχνολογία των μεγάλων δεδομένων. Παραθέτουμε δύο παραδείγματα μεθοδολογιών σχεδίασης και ανάπτυξης πληροφοριακών συστημάτων, και εννοιολογικών πλαισίων (conceptual framework) για ΠΣ, Βήματα των μεθοδολογιών αυτών θα συζητήσουμε στη μελέτη περίπτωσης.

Μεθοδολογία Design Science Research Methodology (DSRM)

Η μεθοδολογία “Design Science Research Methodology ” είναι μια σχετικά νέα προσέγγιση στην έρευνα αναφορικά με πληροφοριακά συστήματα η οποία αποσκοπεί στην επίλυση προβλημάτων μέσα από μια νέα οπτική. Κύριος στόχος της είναι να εμβαθύνουμε πέρα από την κλασσική κατανόηση των λειτουργιών ενός παραδοσιακού πληροφοριακού συστήματος το οποίο θα ικανοποιήσει τις τρέχουσες ανάγκες της επιχείρησης και να αξιοποιήσουμε καινοτομία και αναδυόμενες τεχνολογίες. Δηλαδή, η μεθοδολογία DSR εστιάζει στην «οικοδόμηση» μιας νέας «πραγματικότητας» για τη λειτουργία της επιχείρησης ώστε να αποκτήσει συγκριτικό πλεονέκτημα στον τομέα δραστηριότητάς της. Η μεθοδολογία DSRM για ΠΣ είναι μια επαναληπτική διαδικασία (iteration process) έξι (6) βημάτων:

1) καθορισμός προβλήματος και κίνητρο επίλυσης (problem identification/statement and motivation),
2) μελέτη συναφούς υλικού (review of background material) και ορισμός στόχων μίας λύσης (definition of the objectives for a solution),
3) σχεδίαση και ανάπτυξη (design and development),
4) επίδειξη (demonstration) και περαιτέρω επεξεργασία του τελικού σχεδιασμού (refinement of the final design)
5) αποτίμηση (evaluation),
6) διάδοση-διάχυση (communication) της λύσης.

Η εφαρμογή της μεθοδολογίας περιγράφεται με απλό και εύληπτο τρόπο στην ακόλουθη διατριβή:

Reubens, R. (2016). To craft, by design, for sustainability: Towards holistic sustainability design for developing-country enterprises. PhD. (online). Delft University of Technology.

http://resolver.tudelft.nl/uuid:0c2c14c8-9550-449d-b1ff-7e0588ccd6c2 (17.05.2022).

Μεθοδολογία Soft System Methodology (SSM)

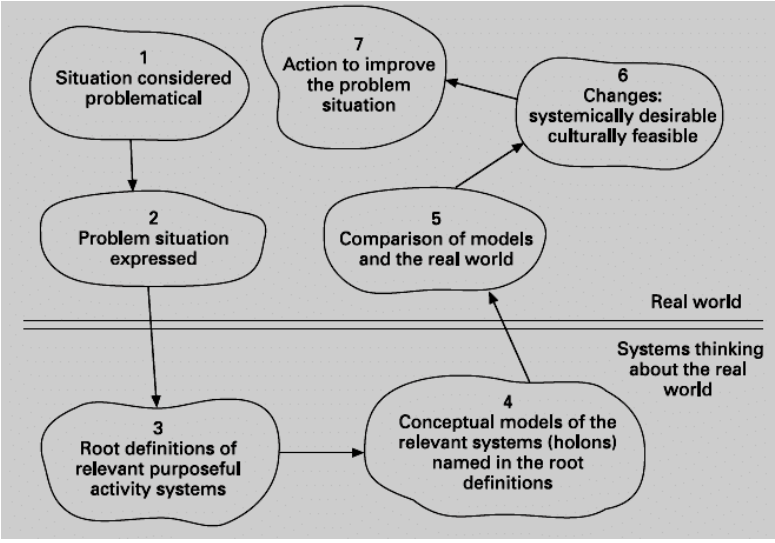
Η μεθοδολογία SSM αποτελεί δημοφιλή προσέγγιση στο Ηνωμένο Βασίλειο, στην Σκανδιναβία και όχι μόνο, και είναι κατάλληλη για να κατανοήσουμε σύνθετες, «χαοτικές», καταστάσεις που αφορούν την οργάνωση και την πολιτική της επιχειρήσεις, να ορίσουμε και να κατανοήσουμε τα «δύσκολα» (όχι καλά ορισμένα) προβλήματα που αντιμετωπίζει και να οδηγηθούμε σε δομημένες λύσεις. Δηλαδή, επιτρέπει στους χρήστες να αντιμετωπίσουν τα δύσκολα προβλήματα με οργανωμένο τρόπο και να αναζητήσουν ολοκληρωμένες λύσεις και όχι απλά τεχνικές λύσεις. Η μεθοδολογία SSM περιλαμβάνει επτά βήματα όπως φαίνεται στην εικόνα 2.4. Παρόμοια εικόνα υπάρχει στα ακόλουθα άρθρα:

Susan Gasson, The Process of Soft Systems Methodology

https://cci.drexel.edu/faculty/sgasson/SSM/Process.html

Dana Indra Sensuse and Arief Ramadhan (2012) Enriching Soft Systems Methodology (SSM) With Hermeneutic in e-Government Systems Development Process, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 2, ISSN (Online): 1694-0814 2012_Enriching_Soft_Systems_Methodology_SSM_With_Hermen.pdf

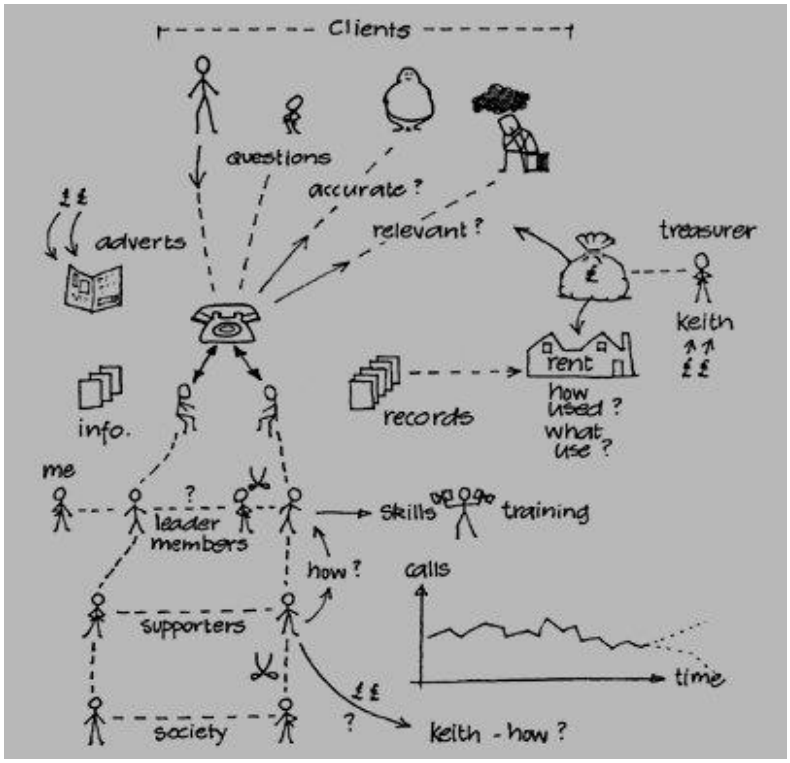
Η εφαρμογή των επτά (7) βημάτων της μεθοδολογίας Soft System (SSM) προτείνεται προκειμένου να οριστούν εννοιολογικά μοντέλα (εννοιολογικά πλαίσια) για πληροφοριακά συστήματα (Checkland and Poulter, 2006). Το έργο του Checkland είναι αυτό που σφραγίζει τη μεθοδολογία αυτή.



Εικόνα 2.4 Βήματα μεθοδολογίας SSM (Sensuse and Ramadhan, 2012).

Ένα κρίσιμο σημείο στη μεθοδολογία SSM είναι η συνεργασία με τα ενδιαφερόμενα μέρη. Συνήθης πρακτική αποτελούν οι συνεντεύξεις και οι συζητήσεις με κύριο υλικό «πλούσιες» ενημερωτικές εικόνες (Rich pictures), μοντέλα δραστηριοτήτων, κ.λπ. Υποθέτουμε, στη μεθοδολογία αυτή, ότι η διεξαγωγή των συζητήσεων θα βελτίωση την κατανόηση έτσι ώστε να μπορέσουμε να απεικονίσουμε καλύτερα τα αποτελέσματα της ανάλυσή μας. Στην εικόνα 2.5 βλέπουμε μέρος μιας πλούσιας εικόνας μιας κατάστασης τηλεφωνικής γραμμής βοήθειας (a telephone helpline situation) (Sensuse and Ramadhan, 2012).

http://systems.open.ac.uk/materials/T552/pages/rich/richAp pendix.html



Εικόνα 2.5 Πλούσια εικόνα μιας κατάστασης τηλεφωνικής γραμμής (Sensuse and Ramadhan, 2012).

Στη βιβλιογραφία υπάρχουν πολλές μεθοδολογίες για τον σχεδιασμό αρχιτεκτονικών και εφαρμογών Big Data σε ΜΕΘ, ανάλογα με τους στόχους της συγκεκριμένης έρευνας. Για παράδειγμα, οι Gonçalves et al (2019) συζητούν το πώς θα μπορούσαμε να προσαρμόσουμε τη μεθοδολογία DSRM και προτείνουν μια αρχιτεκτονική Big Data για επεξεργασία και αποθήκευση δεδομένων σε πραγματικό χρόνο που βασίζεται σε τεχνολογίες ανοιχτού κώδικα.. Οι Peffers et al (2008) προτείνουν, επίσης, τη χρήση της μεθοδολογίας DSRM. Αξίζει να αναφερθεί ότι οι συγγραφείς παρουσίασαν επίσης το σχεδιασμό μιας βάσης δεδομένων για την υποστήριξη μεθόδων αξιολόγησης της υγείας.

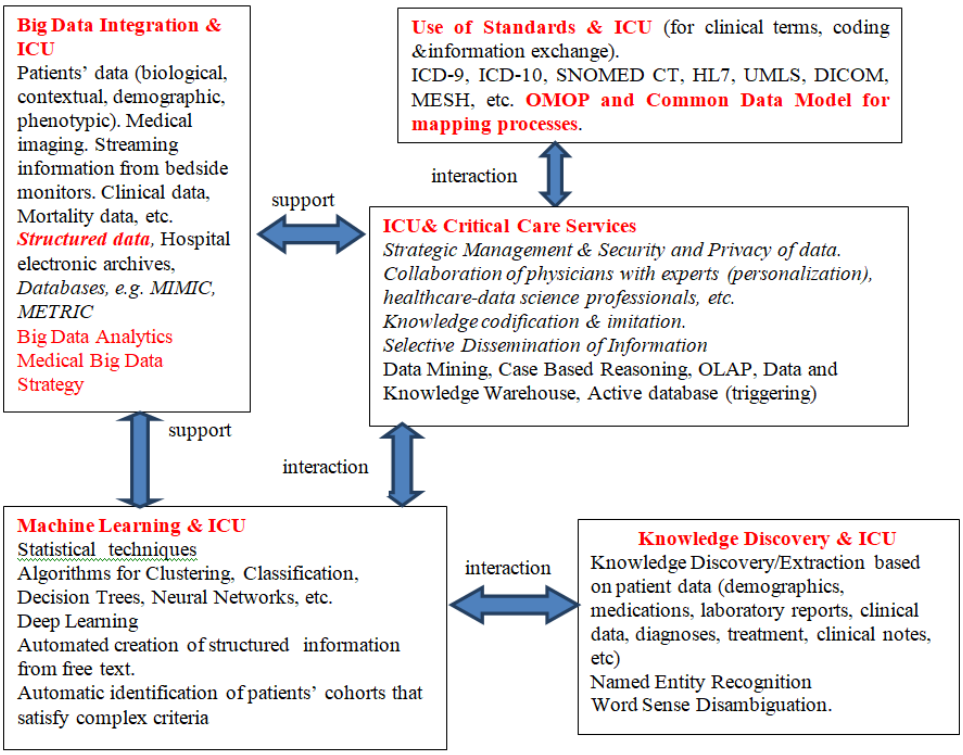
Fillion et al. (2015) χρησιμοποιεί βήματα προσαρμοσμένα από τη μεθοδολογία Soft System (SSM) για να σχεδιάσουν ένα ολοκληρωμένο μοντέλο διαχείρισης γνώσης (Knowledge Management model) το οποίο μπορεί να εφαρμοστεί σε νοσοκομειακό και κλινικό πλαίσιο.

Στη συνέχεια θα εξετάσουμε πως χρησιμοποιούνται τα βήματα 1, 2 και 3 της μεθοδολογίας DSRM. Θα μπορούσαμε εναλλακτικά/παράλληλα να χρησιμοποιήσουμε διαγράμματα εμπλουτισμένων εικόνων (rich pictures diagram) κατά τις συναντήσεις με τους γιατρούς, με τον κίνδυνο όμως η προσέγγιση με εμπλουτισμένες εικόνες να θεωρηθεί από τους γιατρούς ως λιγότερο επιστημονική. Η προσέγγιση DSRM μπορεί να έχει ως στόχο να απαντηθούν οι ακόλουθες ερωτήσεις:

1) Ποια είναι τα κύρια προβλήματα που αντιμετωπίζουμε, όχι μόνο στην παροχή υπηρεσιών εντατικής θεραπείας, αλλά και στην αξιοποίηση της τεχνολογίας big data στη ΜΕΘ σε πραγματικό χρόνο; Πόσο σημαντική είναι η επίλυση αυτών των προβλημάτων;
2) Ποιοι είναι οι στόχοι της επίλυσης προβλημάτων; Τι περιορισμοί υπάρχουν; Τι είναι δυνατόν να γίνει;
3) Ποια είναι τα αποτελέσματα της παραγωγής λύσεων; Ποια είναι τα αποτελέσματα που οδηγούν σε νέες έννοιες ή/και αλλαγή των αρχών-πεποιθήσεων μας, νέα μοντέλα και μεθόδους, σενάρια διαχείρισης και παραδείγματα υποσχόμενα τη βελτίωση των υπηρεσιών στη ΜΕΘ;

Η εικόνα 2.6 απεικονίζει ένα εννοιολογικό πλαίσιο για τις Μονάδες Εντατικής Θεραπείας και τις Υπηρεσίες Εντατικής Θεραπείας. Το πλαίσιο περιλαμβάνει την κεντρική ενότητα και τέσσερα υποσυστήματα, και βασίζεται στην εφαρμογή της μεθοδολογίας SSM και σε επιλεκτική βιβλιογραφία (βλέπε και Markopoulos

et.al., 2021). Το εννοιολογικό πλαίσιο προϋποθέτει τη χρήση τεχνικών και εργαλείων ML και NLP και βασίζεται στην ενσωμάτωση Big Data (BD). Τα μεγάλα δεδομένα περιλαμβάνουν πολλές και διαφορετικές μορφές δεδομένων και πρέπει να χρησιμοποιούν τα διάφορα πρότυπα για τους κλινικούς όρους και την κωδικοποίηση και την ανταλλαγή πληροφοριών. Επομένως, υπάρχει ανάγκη για διαδικασίες χαρτογράφησης (mapping processes) για την ενοποίηση δεδομένων που υιοθετούν διαφορετικά πρότυπα. Στην εικόνα 2.6 υπάρχει κάποια επικάλυψη μεταξύ ML και NLP. Στη συνέχεια παρουσιάζουμε και συζητάμε την κεντρική και τις άλλες ενότητες του πλαισίου.



Εικόνα 2.6. Πλαίσιο για Μονάδες Εντατικής Θεραπείας και τις Υπηρεσίες Εντατικής Θεραπείας (Intensive Care Unit and Critical Care Services) (Markopoulos et al., 2021)

Η ενότητα «ολοκλήρωσης μεγάλων δεδομένων» (The "Big Data Integration and ICU" module")

Εδώ και πολλά χρόνια, η βιβλιογραφία έχει σημειώσει την τεράστια επίδραση των ηλεκτρονικών συστημάτων πληροφοριών στη συλλογή διαφόρων ιατρικών δεδομένων. Τα δεδομένα που παράγονται στη μονάδα εντατικής θεραπείας χαρακτηρίζονται ως big data γιατί συνδυάζουν τεράστια ποσότητα (όγκος, volume), υψηλή ταχύτητα συσσώρευσης (Velocity) και μεγάλη ποικιλία (variety). Τα μεγάλα δεδομένα μπορεί να περιλαμβάνουν (μετά από επεξεργασία) ηλεκτρονικούς φακέλους υγείας (Electronic Health Record-EHR), δημογραφικά στοιχεία, κλινικά δεδομένα, αποτελέσματα εργαστηριακών αναλύσεων, δεδομένα ιατρικών συσκευών, π.χ. σήματα από συστήματα παρακολούθησης-μόνιτορ- δίπλα στο κρεβάτι, διαγνωστικά δεδομένα, σχεδιασμό θεραπείας, βιολογικά και πειραματικά δεδομένα, κ.λπ. Τα δεδομένα είναι δομημένα ή μη και αποθηκεύονται σε διάφορες μορφές, π.χ. ελεύθερο κείμενο, εικόνες, ήχος, βίντεο κ.λπ. Σήμερα, διοικητικές βάσεις δεδομένων (administrative database), κοινωνικά δίκτυα, τηλεϊατρική και πολλές άλλες πηγές χρησιμοποιούνται για την εξαγωγή και τη χρήση μεγάλων δεδομένων. Είναι ενδιαφέρον να σημειωθεί ότι σύμφωνα με τη βιβλιογραφία, π.χ. (Sevenster, 2015), γιατροί, νοσηλευτές και τεχνολόγοι «δαπανούν» μεγάλο μέρος του χρόνου τους καταγράφοντας σωστά το περιστατικό και τη θεραπεία του ασθενούς. Η τεκμηρίωση ενός περιστατικού, στις περισσότερες περιπτώσεις, περιλαμβάνει ελεύθερο κείμενο (σημειώσεις σε ελεύθερο κείμενο) σε συνδυασμό με ελεγχόμενους όρους (controlled terms) που χρησιμοποιούνται για τον χαρακτηρισμό της διάγνωσης, της θεραπείας κ.λπ. δοκιμές και αποτελέσματα, δεδομένα παρακολούθησης

κ.λπ. Επειδή τα δεδομένα και οι πληροφορίες αποθηκεύονται σε διάφορες μορφές, κείμενο, εικόνες, ήχος, βίντεο, κ.λπ., έχουμε μια τεράστια πολυεπίπεδη βάση δεδομένων, η οποία όμως μπορεί να δομηθεί. Βλέπε και το παράδειγμα της βάσης δεδομένων MIMEX.

Η ενότητα «ΜΕΘ & Υπηρεσίες Εντατικής Θεραπείας» (The "ICU & Critical Care Services" module)

Η κεντρική ενότητα «ΜΕΘ & Υπηρεσίες Εντατικής Θεραπείας» θα μπορούσε να θεωρηθεί ως ένα ευφυές σύστημα πληροφοριών (Intelligent Information system) ειδικού σκοπού για την υποστήριξη κλινικών συνεργατών (clinical partners) στις πραγματικές συνθήκες των ΜΕΘ. Ένα τέτοιο σύστημα έχει οργανωτική δομή και χρησιμοποιεί συγκεκριμένους πόρους για τη διεξαγωγή διαδικασιών και οι κλινικοί εταίροι έχουν συγκεκριμένους ρόλους. Είναι σημαντικό να υπάρχει ένα λειτουργικό περιβάλλον που θα μπορεί να υποστηρίζει τους γιατρούς να έχουν πρόσβαση σε πόρους μεγάλων δεδομένων σε πραγματικό χρόνο, οποτεδήποτε και οπουδήποτε, και να υποστηρίζει τις αποφάσεις τους. Επομένως, αυτή η ενότητα θα υποστηρίξει όλες τις δραστηριότητες που σχετίζονται με τη στρατηγική διαχείριση και την ασφάλεια και το απόρρητο των δεδομένων.

Η κεντρική μονάδα υποστηρίζει επίσης υπηρεσίες και διαδικασίες προς δύο κατευθύνσεις:

1) Υπηρεσίες Διαχείρισης Πληροφοριών. Τέτοιες υπηρεσίες περιλαμβάνουν εξόρυξη δεδομένων, οπτικοποίηση, συλλογιστική βάσει περιπτώσεων (Case Based Reasoning-CBR), OLAP, αποθήκες δεδομένων και πρατήρια δεδομένων, ενεργή βάση δεδομένων η οποία απαιτείται για την ενεργοποίηση ενεργειών (triggering of actions) σε συγκεκριμένα συμβάντα, Επιλεκτική διάδοση πληροφοριών (Selective Dissemination of Information).
2) Υπηρεσίες διαχείρισης γνώσης (knowledge management services). Τέτοιες υπηρεσίες (services) και διαδικασίες σχετίζονται με την εξαγωγή γνώσης και την επαναχρησιμοποίηση για την υποστήριξη της λήψης αποφάσεων στις ΜΕΘ.

Οι κύριες διαδικασίες είναι οι εξής:

1) Κωδικοποίηση γνώσης (knowledge codification) και «μίμηση» (αντιγραφή-αξιοποίηση της γνώσης άλλων) (knowledge imitation). Οι ΜΕΘ ως μέρος ενός περιβάλλοντος ιατρικής περίθαλψης πρέπει να έχουν μια στρατηγική διαχείρισης γνώσης για την κωδικοποίηση και αποθήκευση της γνώσης σε βάσεις γνώσεων. Στόχος είναι η ανταλλαγή γνώσεων και καλών πρακτικών (good practices) θεραπείας και ιατρικής περίθαλψης. Στην περίπτωση συνεργασίας μεταξύ ΜΕΘ, μια διαδικασία «μίμησης» της γνώσης ή η λειτουργία κοινοτήτων ενιαίας διαχείρισης της (ιατρικής) πρακτικής (Communities of Practices) θα μπορούσε να είναι χρήσιμη.
2) Συνεργασία ιατρών με ειδικούς, επαγγελματίες υγείας-επιστήμης δεδομένων κ.λπ. Είναι απαραίτητο οι γιατροί να επικοινωνούν και να συζητούν περίπλοκες καταστάσεις και προβλήματα με τους εμπειρογνώμονες. Ο όρος εξατομίκευση (personalization) χρησιμοποιείται στο πλαίσιο της διαχείρισης γνώσης για να περιγράψει τη διαδικασία συνεχούς προσωπικής επικοινωνίας μεταξύ ειδικών και μελών ενός οργανισμού.

Η ενότητα «Χρήση προτύπων & ΜΕΘ» (The "Use of Standards & ICU" module)

Η εντατική φροντίδα (Critical care), η τεκμηρίωση της διάγνωσης και η ανταλλαγή πληροφοριών υγειονομικής περίθαλψης πρέπει να βασίζονται σε δημοφιλή-ευρεία αποδοχής- πρότυπα. Στη ΜΕΘ μπορούμε να χρησιμοποιήσουμε πρότυπα για κλινικούς όρους, κωδικοποίηση, τεκμηρίωση, ανταλλαγή πληροφοριών κ.λπ. Η ενότητα πρέπει να βασίζεται σε πρότυπα προκειμένου να επωφεληθεί από όλα τα οφέλη των νέων τεχνολογιών, όπως ο σημασιολογικός ιστός, και η υπολογιστική νέφους (cloud computing).

Όσον αφορά τα δεδομένα, στην πράξη υπάρχει μια ποικιλία σχημάτων που υιοθετούνται σε δύο κατευθύνσεις: 1) πρότυπα για την (ημι)αυτόματη επεξεργασία και ταξινόμηση ιατρικών αρχείων (files) ή εγγράφων (document), π.χ., ICD-10 και 2) για την ανταλλαγή δεδομένων (information exchange) μεταξύ συστημάτων ιατρικής πληροφόρησης.

Το υπάρχον πρόβλημα στην ενσωμάτωση και αξιοποίηση των δεδομένων σχετίζεται ακριβώς με τη Βαβέλ των σχετικών προτύπων. Για παράδειγμα, εάν το ICD-9 χρησιμοποιείται για διάγνωση, υπάρχει κόστος για την έναρξη χρήσης του ICD-10. Οι ίδιες δυσκολίες εμφανίζονται αν η διάγνωση βασίζεται στα πρότυπα MESH, UMLS, SNOMED CT κ.λπ. οπότε πρέπει να εφαρμόσουμε κάποια «χαρτογράφηση» μεταξύ προτύπων εάν θέλουμε επικοινωνία μεταξύ διαφορετικών πληροφοριακών συστημάτων. Το πρότυπο OMOP (2019) προτείνει ένα κοινό μοντέλο δεδομένων για τη χαρτογράφηση διαδικασιών μεταξύ διαφορετικών κωδικοποιήσεων δεδομένων και είναι το αποτέλεσμα της συνεχιζόμενης συνεργατικής δράσης της κοινότητας Observational Health Data Sciences and Informatics (OHDSI).

Η ενότητα «Μηχανική Μάθηση & ΜΕΘ» (The "Machine Learning & ICU" module)

Οι αλγόριθμοι πρόβλεψης και προγνωστικής μάθησης (Predictive and prognostic learning algorithms) χρησιμοποιούνται ευρέως στα δεδομένα που ενδιαφέρουν στη ΜΕΘ. Η δημιουργία τέτοιων μοντέλων ανήκει σε δευτερογενή χρήση των αρχείων υγείας (health records). Τα μοντέλα πρόβλεψης (Prediction models) συνήθως εκπαιδεύονται για να προβλέψουν την πιθανότητα μιας κατάστασης, ενός γεγονότος ή της απόκρισης, (σε κάποια ενέργεια), π.χ., της αντίδρασης σε ένα φάρμακο ή ένα σχέδιο θεραπείας. Από την άλλη πλευρά, τα προγνωστικά μοντέλα (prognostic models) εκπαιδεύονται για να προβλέψουν την πιθανότητα ενός «καταληκτικού σημείου» που σχετίζεται με την πάθηση, όπως η θνησιμότητα. Η μηχανική μάθηση είναι επίσης χρήσιμη για την πρόβλεψη του κινδύνου ασθενειών και τον προσδιορισμό των επιδημιών. Επιπλέον, η αναγνώριση φυσικής γλώσσας χρησιμοποιείται για τον εντοπισμό παραγόντων υψηλού κινδύνου, την εξαγωγή γνώσεων από κλινικές σημειώσεις, την αποδοτικότητα κόστους και τη βελτίωση της φροντίδας και τη μείωση της πιθανότητας νοσηρότητας και θνησιμότητας. Βλέπε, ενδεικτικά (Markopoulos et al., 2021) και επισυναπτόμενη βιβλιογραφία.

Ωστόσο, τα αρχεία υγείας περιλαμβάνουν μεγάλο μέρος σημαντικών δεδομένων που είναι αναπόφευκτα αδόμητα και κάποτε κακής ποιότητας. Αυτά τα δεδομένα μπορεί να περιλαμβάνουν σημειώσεις γιατρών για τη θεραπεία ή την παρακολούθηση ασθενούς. Με τη συμβολή της μηχανικής μάθησης, η επεξεργασία φυσικής γλώσσας (NLP) θα μπορούσε να βοηθήσει στην ανάλυση, εξαγωγή και επεξεργασία αυτών των δεδομένων προκειμένου να ενισχυθεί η ακρίβεια των ηλεκτρονικών φακέλων των ασθενών (EHR) μετατρέποντας το ελεύθερο κείμενο σε τυποποιημένα-δομημένα δεδομένα. Χρησιμοποιώντας τεχνολογία NLP, οι γιατροί μπορούν να έχουν πρόσβαση σε αυτές τις πληροφορίες για λήψη αποφάσεων και αναλύσεις, καθώς έχουν μια πιο ακριβή άποψη σχετικά με το αρχείο υγείας των ασθενών (patient health record).

Η ενότητα «ανακάλυψη γνώσης και ΜΕΘ» (The "knowledge discovery and ICU" module)

Έχοντας συγκεντρώσει τις πληροφορίες που απαιτούνται από όλες τις διαθέσιμες πηγές, όπως δημογραφικά στοιχεία, θεραπείες, εργαστηριακές εξετάσεις, διαγνώσεις, κλινικές σημειώσεις και ιστορικό ασθενών, είναι δυνατό να δημιουργηθεί γνώση. Η σωστή χρήση των ιατρικών δεδομένων περιλαμβάνει πολλές προκλήσεις επειδή είναι δύσκολο να οργανωθούν, να προσπελαστούν και να χρησιμοποιηθούν με δομημένο και ομοιογενή τρόπο. Επομένως, εκτός από τη σωστή χρήση των τεχνολογιών που προαναφέρθηκαν, πρέπει να γίνει σωστή επιλογή δεδομένων για να είναι επιτυχής η εξαγωγή γνώσης.

Παραθέτουμε τα συμπεράσματα της μελέτης περίπτωσης.

Μεθοδολογικά είναι σκόπιμο όταν σχεδιάζουμε ένα έργο διαχείρισης (μεγάλων) δεδομένων να υιοθετήσουμε στοιχεία μιας μεθοδολογίας, π.χ., DSRM, SSM, και να συνθέσουμε ένα εννοιολογικό μοντέλο του πληροφοριακού συστήματος, ως αφετηρία όλων των επόμενων ενεργειών μας.

Στη συγκεκριμένη μελέτη περίπτωσης εστιάσαμε στην αξιοποίηση τεχνικών μηχανικής μάθησης, επεξεργασίας φυσικής γλώσσας και τεχνικών διαχείρισης γνώσης για το σχεδιασμό του εννοιολογικού πλαισίου του ΠΣ και της υποστήριξης αποφάσεων σε ΜΕΘ. Διερευνήσαμε επίσης τον ρόλο της μεθοδολογίας στην ανάπτυξη ενός εννοιολογικού πλαισίου για τη ΜΕΘ, και παρουσιάσαμε ένα πλαίσιο και τις ενότητές του (εικόνα 2.6).

Η ανάπτυξη ενός πραγματικού εννοιολογικού πλαισίου περιλαμβάνει συνεντεύξεις με συνεργάτες στη ΜΕΘ και τα συμπεράσματα που θα μπορούσαμε να αντλήσουμε από αυτούς, μελέτη επιλεγμένης βιβλιογραφίας που εστιάζει στις δυνατότητες εφαρμογής της τεχνολογίας μεγάλων δεδομένων και στην παροχή υπηρεσιών στη ΜΕΘ.

Ο σχεδιασμός επαρκούς επιχειρηματικής αρχιτεκτονικής της διαχείρισης των δεδομένων ενός οργανισμού και των λειτουργιών του και η σύγκριση μεταξύ διαφορετικών αρχιτεκτονικών αποτελούν ένα πολύ γνωστό πρόβλημα στο πλαίσιο της έρευνας και της πρακτικής (δηλαδή της οικοδόμησης συστημάτων) στα πληροφοριακά συστήματα. Σημαντικά εμπόδια είναι η συχνά διφορούμενη ορολογία και η έλλειψη σαφήνειας στη χρήση όρων, εννοιών, μοντέλων κ.λπ. Η ανάπτυξη ενός εννοιολογικού πλαισίου φαίνεται να αποτελεί λύση για τη βελτίωση ενός προβληματικού σχεδιασμού της επιχειρηματικής αρχιτεκτονικής.

Το εννοιολογικό πλαίσιο που θα προτείνουμε σε ένα έργο θα πρέπει να βασίζεται στην έρευνα και αξιολόγηση της σχετικής βιβλιογραφίας, και θα πρέπει να αναφέρει με σαφήνεια τις διάφορες έννοιες και όρους, να τις οργανώνει, και να επισημαίνει «την υποκείμενη σύλληψη» (the underlying conceptualization) στη διαδικασία σχεδιασμού.

2.1.3.3 Ενδεικτική βιβλιογραφία και περαιτέρω αναφορές

Núnez Reiz, M.A. Armengol de la Hoz, M., Sánchez García (2019), Big Data Analysis and Machine Learning in Intensive Care Units, Med Intensiva, 43(7), pp.416-426

Al-jumeily, D., Hussain, A., Mallucci, C. , Oliver, C. *Applied Computing in Medicine and Health.* 2015

Gonçalves, A., Portela, F., Santos, M. F., & Rua, F. (2017). Towards of a real-time big data architecture to intensive care. International Workshop on Healthcare Interoperability and Pervasive Intelligent Systems (HiPIS 2017), Procedia Computer Science, 113, 585–590.

Carayon, P., Wetterneck, T.B., Alyousef, B., Brown, R. L., Cartmill R. S., McGuire K., & Wood, K. (2015) Impact of electronic health record technology on the work and workflow of physicians in the intensive care unit. *International Journal of Medical Informatics*, 84(8), 578-594. doi:10.1016/j.ijmedinf.2015.04.002

Celi, L. A., Mark, R. G., Stone, D. J., & Montgomery, R. A. (2013) "Big data" in the intensive care unit. Closing the data loop. *American journal of respiratory and critical care medicine*, 187(11), 1157–1160. doi:10.1164/rccm.201212-2311ED.

Checkland P and Poulter J (2006) Learning for Action: A Short Definitive Account of Soft Systems Methodology and Its Use, for Practitioners, Teachers and Students. John Wiley and Sons Ltd, Chichester.

Chen, M., Hao, Y. , Hwang, K., Wang, L. [Lu], Wang, L.[Lin]. (2017) Disease prediction by machine learning over big data from healthcare communities. *IEEE Access*, 8869–8879. doi:10.1109/access.2017.2694446..

Chytas K., Tsolakidis A., Skourlas C. (2020) Towards a Framework for Learning Systems in Smart Universities. In: Kumar V., Troussas C. (eds) Intelligent Tutoring Systems. ITS 2020. Lecture Notes in Computer Science, vol 12149. Springer, Cham. https://doi.org/10.1007/978-3-030-49663-0_32</bib>

Fillion, G., BootoEkionea, J. P., & Plaisent, M. (2015, October). Using the soft system methodology for designing an integrated and inter-firm knowledge management capabilities maturity model in health care organization. In The first International Conference on Multidisciplinary in Management.

Herasevich, V., Pickering, B.W., Dong, Y., Peters, S.G. & Gajic, O. (2010). Informatics infrastructure for syndrome surveillance, decision support, reporting, and modelling of critical illness. Mayo Clin Proc, 85(3), 247–54. http://dx.doi.org/10.4065/mcp.2009.0479.

Intensive Care Society (2019), Guidelines for the provision of intensive care services (GPICS), 2nd edition, Available at https://www.ficm.ac.uk/sites/default/files/gpics-v2.pdf

Lee, J., Scott, D.J., Villarroel, M., Clifford, G.D., Saeed, M. & Mark, R.G. (2011). Open-access MIMIC-II database for intensive care research. Annu Int Conf IEEE, Eng Med Biol Soc 8315–8, http://dx.doi.org/10.1109/IEMBS.2011.6092050.

Adibuzzaman, M., DeLaurentis, P., Hill, J., & Benneyworth, B. D. (2017). Big data in healthcare–the promises, challenges and opportunities from a research perspective: A case study with a model database. In AMIA Annual Symposium Proceedings, 384–392. American Medical Informatics Association.

Observational Health Data Sciences and Informatics (2019) The Book of OHDSI, ISBN 978-1088855-195, p.455

Peffers, K., Tuunanen, T., Rothenberger, M.A.., & Chatterjee, S. (2008). A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems, 24(3), 45–78.

Pirracchio, R., Cohen, M. J., Malenica, I., Cohen, J., Chambaz, A., Lee, C., Resche-Rigon, M. & Hubbard, A. (2019) Big data and targeted machine learning in action to assist medical decision in the ICU, Anaesthesia Critical Care & Pain Medicine, 38(4), 377-384

Buckl, S., Matthes, F., Roth, S., Schulz, C., & Schweda, C. M. (2010, November). A conceptual framework for enterprise architecture design. In International Workshop on Trends in Enterprise Architecture Research. Springer, Berlin, Heidelberg, Lecture Notes in Business Information Processing 70, 44-56

Sanchez-Pinto, L.N., Luo, Y., Churpek, M.M. (2018) Big data and data science in critical care. *Chest*, 1239-1248. doi:10.1016/j.chest.2018.04.037.

Bailly, S., Meyfroidt, G., & Timsit, J. F. (2018). What's new in ICU in 2050: big data and machine learning. Intensive care medicine, 44(9), 1524-1527.

Sevenster, M., Buurman, J., Liu, P., Peters, J. F., & Chang, P. J. (2015). Natural language processing techniques for extracting and categorizing finding measurements in narrative radiology reports. Applied clinical informatics, 6(03), 600-610.

van de Klundert, N., Holman, R., Dongelmans, D. A., & de Keizer, N. F. (2015). Data resource profile: the Dutch National Intensive Care Evaluation (NICE) registry of admissions to adult intensive care units. International journal of epidemiology, 44(6), 1850-1850h.

Markopoulos, D., Tsolakidis, A., N. Karanikolas, N., & Skourlas, C. (2020, November). Towards the design of a Conceptual Framework for the operation of Intensive Care Units based on Big Data Analysis. In 24th Pan-Hellenic Conference on Informatics, 411-415 ACM

https://doi.org/10.1145/3437120.3437352

2.2 Δεδομένα (data) και δεδομένα μεγάλης κλίμακας (big data)

Αν και με την πρόοδο της τεχνολογίας αλλά και τις εξελίξεις στο διαδίκτυο είχαμε πάρα πολλά δεδομένα, η αναζήτηση και η διαχείριση κάθε πηγής δεδομένων μπορούσε να γίνει ανεξάρτητα. Για παράδειγμα, σε ένα παραδοσιακό ολοκληρωμένο σύστημα βιβλιοθήκης έχουμε δομημένα δεδομένα διαχειρίσιμα με σχεσιακό ΣΔΒΔ, π.χ., Oracle DBMS, αλλά και (ημι)δεδομένα σε MARC XML μορφότυπο για φόρτωση στον κατάλογο της βιβλιοθήκης.

Η διαφορά σήμερα είναι ότι υπάρχουν σημαντικά περισσότερα δεδομένα τα οποία ποικίλλουν σε τύπο και μορφότυπο αλλά και σε τρόπους ανάκτησης και διαχείρισης. Για παράδειγμα, έχουμε μη δομημένα δεδομένα υπό τη μορφή σχολίων στην ομάδα στο Facebook της βιβλιοθήκης.

Τα ολοκληρωμένα ή ενοποιημένα ή ενσωματωμένα δεδομένα (integrated data), τα οποία περιλαμβάνουν δομημένα-ημιδομημένα-μη δομημένα δεδομένα, αν πληρούν ορισμένες προϋποθέσεις ονομάζονται δεδομένα μεγάλης κλίμακας (ή μεγάλα δεδομένα) και η έννοια αυτή συνδέεται άμεσα με αυτό που ονομάζουμε «η ευκαιρία και η πρόκληση των μεγάλων δεδομένων» (“the opportunity and challenge of big data”).

Ως αφετηρία για έναν πρώτο ορισμό, μπορούμε να θεωρήσουμε ότι τα μεγάλα δεδομένα ορίζονται ως κάθε είδους πηγή δεδομένων που έχει τουλάχιστον τρία κοινά χαρακτηριστικά:

1) Εξαιρετικά μεγάλος όγκος δεδομένων (Volumes)
2) Εξαιρετικά υψηλή ταχύτητα παραγωγής δεδομένων (Velocity)
3) Εξαιρετικά μεγάλη ποικιλία δεδομένων σε τύπο-data type και μορφότυπο-format (Variety)

Η παρακάτω υποενότητα αναφέρεται στα δεδομένα μεγάλης κλίμακας και βασίζεται στο σύγγραμμα: Μαρινάγη, Α., Σκουρλάς, Χ. (2022), Διαχείριση γνώσης, Κάλλιπος

2.2.1 Οργάνωση δεδομένων και δεδομένα μεγάλης κλίμακας (Data organization and big data)

Οι Gandomi & Haider (2015) υπογραμμίζουν ότι συχνά το μέγεθος είναι η πρώτη, και μερικές φορές, η μόνη διάσταση που έρχεται στο μυαλό μας όταν γίνεται αναφορά στα δεδομένα μεγάλης κλίμακας, ενώ υπάρχουν βέβαια και άλλα μοναδικά και καθοριστικά χαρακτηριστικά των δεδομένων μεγάλης κλίμακας. Επομένως, χρειαζόμαστε έναν «ευρύ» ορισμό για την έννοια. Πιστεύουν, επιπλέον, ότι ακαδημαϊκή έρευνα και η σχετική συζήτηση των δεδομένων μεγάλης κλίμακας, δεν έχουν ακόμη καλύψει ικανοποιητικά το θέμα. Υπολογίζουν ότι τα μη δομημένα δεδομένα αποτελούν το 95% των δεδομένων μεγάλης κλίμακας! (Σημείωση. Γενικά στη βιβλιογραφία η συνήθης εκτίμηση είναι ότι τα μη δομημένα δεδομένα αποτελούν το 80% των δεδομένων). Επομένως, υπάρχει μεγάλη ανάγκη ανάπτυξης κατάλληλων και αποτελεσματικών αναλυτικών μεθόδων για την αξιοποίηση μαζικών όγκων ετερογενών δεδομένων σε μη δομημένες μορφές κειμένου, ήχου και βίντεο. Τέλος, πιστεύουν ότι υπάρχει ανάγκη επινόησης «νέων εργαλείων ανάλυσης-πρόβλεψης και για δομημένα δεδομένα μεγάλης κλίμακας» (new tools for predictive analytics for structured big data). Δείτε σχετικά το άρθρο,

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. International journal of information management, 35(2), 137-144. ISSN 0268-4012, https://doi.org/10.1016/j.ijinfomgt.2014.10.007.

Η γρήγορη εξέλιξη των τεχνολογιών δεδομένων μεγάλης κλίμακας και η άμεση αποδοχή της έννοιας από τον δημόσιο και τον ιδιωτικό τομέα άφησε λίγο χρόνο για να αναπτυχθεί και να ωριμάσει ο (δια)λόγος στον ακαδημαϊκό τομέα. Η Gartner, Inc. ορίζει τα δεδομένα μεγάλης κλίμακας ως εξής:

Σύμφωνα με το Gartner IT Glossary, «Τα δεδομένα μεγάλης κλίμακας αποτελούν περιουσιακά στοιχεία της επιχείρησης και είναι πληροφορίες μεγάλου όγκου, υψηλής ταχύτητας και μεγάλης ποικιλίας που απαιτούν οικονομικά αποδοτικές, καινοτόμες μορφές επεξεργασίας με στόχους την επαύξηση-βελτίωση κατανόησης (διορατικότητας) και λήψης αποφάσεων και τον αυτοματισμό διαδικασιών» ("Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making, and process automation.") (Gartner IT Glossary, 2021)

https://www.gartner.com/it-glossary/big-data/

Το Foundation TechAmerica ορίζει τα δεδομένα μεγάλης κλίμακας ως εξής:

«Τα δεδομένα μεγάλης κλίμακας είναι ένας όρος που περιγράφει μεγάλους όγκους υψηλής ταχύτητας, σύνθετα και μεταβλητά δεδομένα που απαιτούν προηγμένες τεχνικές και τεχνολογίες για να καταστεί δυνατή η συλλογή, αποθήκευση, διανομή, διαχείριση και ανάλυση των πληροφοριών. (Ομοσπονδιακή Επιτροπή Μεγάλων Δεδομένων του Ιδρύματος TechAmerica, 2012)»

(Big data is a term that describes large volumes of high velocity, complex and variable data that require advanced techniques and technologies to enable the capture, storage, distribution, management, and analysis of the information.) (TechAmerica Foundation's Federal Big Data Commission, 2012)

Δείτε και το πλήρες κείμενο στον οδηγό,

Federal Big Data Commission. (2012). Demystifying big data: A practical guide to transforming the business of government. Technical report, TechAmerica Foundation.

Δεδομένα μεγάλης κλίμακας και ορισμοί 3Vs, 6Vs …

Σύμφωνα με τους Gandomi & Haider (2015), οι αδόμητες πληροφορίες, όπως το ηλεκτρονικό ταχυδρομείο, το βίντεο, τα ιστολόγια, οι συνομιλίες στο τηλεφωνικό κέντρο και τα μέσα κοινωνικής δικτύωσης, αποτελούν περίπου το 85% των δεδομένων που δημιουργούνται σήμερα. Τα δεδομένα αυτά «παρουσιάζουν προκλήσεις για την απόκτηση νοήματος με τα συμβατικά εργαλεία επιχειρησιακής νοημοσύνης» (presents challenges in deriving meaning with conventional business intelligence tools). Οι συσκευές παραγωγής πληροφοριών, όπως αισθητήρες, tablet και κινητά τηλέφωνα συνεχίζουν να πολλαπλασιάζονται. Η κοινωνική δικτύωση αυξάνεται επίσης με επιταχυνόμενο ρυθμό καθώς ο κόσμος γίνεται πιο συνδεδεμένος. «Τέτοιες επιλογές ανταλλαγής πληροφοριών αντιπροσωπεύουν μια θεμελιώδη αλλαγή στον τρόπο που οι άνθρωποι, η κυβέρνηση και οι επιχειρήσεις αλληλοεπιδρούν μεταξύ τους» (Such information sharing options represents a fundamental shift in the way people, government and businesses interact with each other) (Gandomi & Haider, 2015).

Πίνακας 2.1 Τι θεωρείται μεγάλα δεδομένα: Petabyte, Exabyte, Zettabyte, Yottabyte

Μονάδα	Πλήθος byte	Προσέγγιση	Ισοδύναμο με
Byte	8 bits		1 χαρακτήρα
Kilobyte	2^{10} (1024) bytes	10^3 bytes	Μισή σελίδα
Megabyte	2^{20} (1048576) bytes	10^6 bytes	1 ψηφιακή εικόνα
Gigabyte	2^{30} (1073741824) bytes	10^9 bytes	45 ώρες μουσικής
Terabyte	2^{40} bytes	10^{12} bytes	500.000.000 σελίδες κειμένου
Petabyte	2^{50} bytes	10^{15} bytes	1.000 σκληρούς δίσκους 1TB
Exabyte	2^{60} bytes	10^{18} bytes	1.000.000 σκληρούς δίσκους 1TB
Zettabyte	2^{70} bytes	10^{21} bytes	Αριθμό ατόμων σε 1 σταγόνα νερού (0,03ml)
Yottabyte	2^{80} bytes	10^{24} bytes	Αριθμό ατόμων σε 2 κουταλιές νερού (30ml)

Τα δεδομένα μεγάλης κλίμακας (Big Data) συχνά χαρακτηρίζονται από τρεις παράγοντες: τον όγκο, την ταχύτητα και την ποικιλία (3Vs, volume, velocity, and variety). Η έρευνα και οι εφαρμογές προσέθεσαν και άλλα Vs. Στον Πίνακα 2.1 βλέπουμε τι θεωρείται μεγάλο.

Ορισμοί 3Vs – 6Vs

Όπως έχει ήδη αναφέραμε στην αρχή της υποενότητας, τα δεδομένα μεγάλης κλίμακας είναι δεδομένα που έχουν τις ακόλουθες τρεις βασικές διαστάσεις, γνωστές ως 3Vs:

- 1V (Volume): πολύ μεγάλος όγκος.
- 2V (Velocity): μεγάλη ταχύτητα παραγωγής και διακίνησης.
- 3V (Variety): μεγάλη ποικιλία.

Επεκτείνοντας τις βασικές διαστάσεις, προτείνονται τρεις νέες διαστάσεις (6 Vs):

- 4V (Value): Υπάρχει ανάγκη (απαίτηση) μετατροπής των δεδομένων μεγάλης κλίμακας σε αξία για την επιχείρηση, μέσω επεξεργασίας και ανάλυσης.
- 5V (Veracity και Validity): ορθότητα και αξιοπιστία των δεδομένων.

Τα δεδομένα για τις εφαρμογές πρέπει να ικανοποιούν τη συνθήκη:

Data = quantity + quality

Επομένως, απαιτούνται μέθοδοι «καθαρισμού δεδομένων» οι οποίες αφαιρούν τα αναξιόπιστα δεδομένα (dirty data: ελλιπή, με θόρυβο, αντιφατικά-ασυνεπή). Τα αναξιόπιστα δεδομένα μπορεί να οδηγήσουν, για παράδειγμα, σε παραπλανητικές οικονομικές εκθέσεις και λανθασμένες στρατηγικές αποφάσεις και επομένως σε απώλεια κερδών, αξιοπιστίας και πελατών, και σε λάθη δεδομένων ιατρικών συνταγών και επομένως σε απώλεια ζωής.

- 6V (Visibility και Visualization): ορατότητα και απεικόνιση δεδομένων.

Μετά την ολοκλήρωση της επεξεργασίας των δεδομένων χρειαζόμαστε έναν τρόπο παρουσίασής τους, ώστε να είναι προσβάσιμα και αναγνώσιμα, π.χ., δεδομένα σε πίνακες, γραφήματα, διαγράμματα κ.λπ. Υπάρχουν πολλά εργαλεία οπτικοποίησης (Big Data Visualization tools), π.χ., Microstrategy, tableau, Cognos, sas, pentaho, QlikView, ClearStory, looker, BusinessObjects.

Τέλος, επισημαίνουμε την κρισιμότητα του ζητήματος με τα ανοιχτά δεδομένα. Η ορατότητα των δεδομένων από όλους εγείρει θέματα ιδιωτικότητας, ασφάλειας και προέλευσης

Ορισμός περισσότερων Vs

Η έρευνα και οι εφαρμογές οδηγούν σε περισσότερες διαστάσεις Vs για τα δεδομένα μεγάλης κλίμακας:

- Volatility: μεταβλητότητα δεδομένων που αφορά το χρονικό διάστημα εγκυρότητας και διατήρησης των δεδομένων.
- Viscosity: ο χρόνος καθυστέρησης (lag time) των δεδομένων τα οποία σχετίζονται με το συμβάν που περιγράφεται (μπορεί να σχετιστεί με το Velocity).
- Virality: ο ρυθμός διάδοσης των δεδομένων, δηλαδή πόσο συχνά επιλέγονται και αναπαράγονται από άλλους χρήστες ή συμβάντα.
- Variability: αναφέρεται στα δεδομένα των οποίων η σημασία σταθερά αλλάζει, π.χ. όταν η συλλογή των δεδομένων εξαρτάται από τη γλωσσική επεξεργασία.

Οι έννοιες μεγάλα δεδομένα και εξόρυξη δεδομένων είναι διαφορετικές έννοιες. Τα μεγάλα δεδομένα είναι ένας όρος ο οποίος *αναφέρεται σε δεδομένα μεγάλου όγκου, τα οποία ανακτώνται ταχύτητα (συχνά σε πραγματικό χρόνο) και έχουν ποικίλες μορφές. Ο όρος εξόρυξη δεδομένων αναφέρεται στις διαδικασίες εξαγωγής γνώσης (knowledge), μοτίβων δεδομένων (patterns), πληροφορίας (information) από έναν μικρό ή μεγάλο όγκο δεδομένων. Στον πίνακα 2.2. επισημαίνουμε διαφορές.*

Πίνακας 2.2 Διαφορές εξόρυξης δεδομένων και της ανάλυσης μεγάλων δεδομένων αναφορικά με τον όγκο, την ταχύτητα και την ποικιλία των δεδομένων

Διάσταση	Εξόρυξη δεδομένων - διαχείριση	Big data analytics
Volume (Όγκος)	GB, TB	Διαχείριση TB, PB, EB, ZB
Velocity (Ταχύτητα)	Κεντρικές βάσεις δεδομένων	Κατανεμημένες βάσεις δεδομένων
Variety (Ποικιλία)	Δομημένα δεδομένα	Ημιδομημένα, αδόμητα δεδομένα

Κεντρικός σε κάθε είδους διαχείριση (και των μεγάλων) δεδομένων είναι ο ρόλος των μεταδεδομένων.

2.2.2 Μεταδεδομένα (Metadata)

Σύμφωνα με έναν ορισμό της Oracle για τα μεταδεδομένα στις αποθήκες δεδομένων,

«Τα μεταδεδομένα είναι δεδομένα σχετικά με τα δεδομένα και, ως εκ τούτου, είναι μια σημαντική πτυχή του περιβάλλοντος της αποθήκης δεδομένων. Τα μεταδεδομένα επιτρέπουν στον τελικό χρήστη και στον επιχειρηματικό αναλυτή να περιηγηθούν στις δυνατότητες χωρίς να γνωρίζουν το περιεχόμενο-πλαίσιο των δεδομένων ή τι ακριβώς αντιπροσωπεύουν τα δεδομένα. Η διαχείριση μεταδεδομένων είναι μια ολοκληρωμένη, συνεχής διαδικασία επίβλεψης και ενεργούς διαχείρισης μεταδεδομένων σε ένα κεντρικό περιβάλλον η οποία βοηθά μια επιχείρηση να κατανοήσει-προσδιορίσει πώς δημιουργούνται τα δεδομένα, ποια δεδομένα υπάρχουν και τι σημαίνουν τα δεδομένα …»

(“Metadata is any data about data and, as such, is an important aspect of the data warehouse environment. Metadata allows the end user and the business analyst to navigate through the possibilities without knowing the context of the data or what the data represents. Metadata management is a comprehensive, ongoing process of overseeing and actively managing metadata in a central environment which helps an enterprise to identify how data is constructed, what data exists, and what the data means”.)

Σύμφωνα με την Oracle τα μεταδεδομένα μπορούν να οργανωθούν σε τρεις κατηγορίες:

- Τα μεταδεδομένα στο επιχειρηματικό επίπεδο (Business metadata), δηλαδή τα δεδομένα τα οποία περιγράφουν την επιχειρηματική ερμηνεία των στοιχείων δεδομένων στην αποθήκη-βάση δεδομένων. Δηλαδή, επιτρέπουν την αντιστοίχιση (mapping) των αντικειμένων της βάσης δεδομένων (ορισμούς και περιορισμούς-κανόνες για τις πληροφορίες-δεδομένα (όπως ονόματα πινάκων, στηλών κ.λπ.) στα μεταδεδομένα της επιχείρησης (σε επιχειρηματικούς ορισμούς και επιχειρηματικούς κανόνες),
- Τα τεχνικά μεταδεδομένα (Technical metadata), τα οποία αντιπροσωπεύουν τις τεχνικές πτυχές των δεδομένων, και περιλαμβάνουν χαρακτηριστικά, όπως είναι οι τύποι δεδομένων, τα μήκη κ.λπ.,
- Τα μεταδεδομένα εκτέλεσης διαδικασιών (Process execution metadata), τα οποία παρουσιάζουν στατιστικά στοιχεία σχετικά με τα αποτελέσματα της εκτέλεσης διαδικασιών, π.χ. διαδικασίας ETL (Extraction, Transformation and Loading) για τη φόρτωση δεδομένων σε αποθήκη δεδομένων, συμπεριλαμβανομένων μέτρων όπως γραμμών πινάκων οι οποίες φορτώθηκαν επιτυχώς, γραμμές που απορρίφθηκαν, ο χρόνος φόρτωσης κ.λπ.

Introduction to Oracle Retail Data Model Customization

Επειδή τα μεταδεδομένα είναι ιδιαίτερα σημαντικά στη διαχείριση πληροφοριών, υπάρχουν ενδιαφέροντα πρότυπα, όπως:

- Πρότυπο κωδικοποίησης και μετάδοσης μεταδεδομένων (METS-Metadata Encoding and Transmission Standard). «Πρότυπο για την κωδικοποίηση περιγραφικών, διοικητικών και δομικών μεταδεδομένων σχετικά με αντικείμενα μέσα σε μια ψηφιακή βιβλιοθήκη» ("A standard for encoding descriptive, administrative, and structural metadata regarding objects within a digital library").
- Object Management Group, Common Warehouse Metamodel (CWM-Common Warehouse Metamodel). Προδιαγραφές μοντελοποίησης μεταδεδομένων για σχεσιακά, μη σχεσιακά, πολυδιάστατα (multi-dimensional) (και άλλα) αντικείμενα σε περιβάλλον αποθήκευσης δεδομένων.

Πολλοί εθνικοί και διεθνείς οργανισμοί προσπαθούν να τυποποιήσουν τα μεταδεδομένα:

- Αμερικανικό Εθνικό Ινστιτούτο Προτύπων (ANSI-American National Standards Institute). Ο οργανισμός που συντονίζει τα εθελοντικά συστήματα τυποποίησης και αξιολόγησης της συμμόρφωσης των ΗΠΑ.
- Διεθνής Οργανισμός Τυποποίησης (ISO-International Organization for Standardization). Ο φορέας που θεσπίζει, αναπτύσσει και προωθεί πρότυπα για τις διεθνείς ανταλλαγές.

Σε επόμενες ενότητες θα συζητηθούν περαιτέρω τα μεταδεδομένα και θα δοθούν παραδείγματα.

2.2.3 «Παραδοσιακή» διαχείριση δεδομένων και η πρόκληση των μεγάλων δεδομένων

Όπως ήδη αναφέραμε υπάρχει σήμερα η ανάγκη διαχείρισης δεδομένων (1) εξαιρετικά μεγάλου όγκου (Volumes), (2) τα οποία παράγονται με εξαιρετικά υψηλή ταχύτητα (Velocity) και (3) τα οποία χαρακτηρίζονται από εξαιρετικά μεγάλη ποικιλία τύπων και μορφοτύπων (Variety).

Οι αρχιτέκτονες λογισμικού (software architects) και οι έμπειροι προγραμματιστές (developers) γνωρίζουν πως να αντιμετωπίσουν σχετικά εύκολα μία ή και δύο από τις παραπάνω καταστάσεις.

Στην περίπτωση δεδομένων συναλλαγών μεγάλου όγκου (high-volume transactional data), και δεδομένου ότι η διαχείριση συναλλαγών συνοδεύεται συχνά από απαίτηση για ανοχή σφαλμάτων (fault tolerance) άρα και από αποθήκευση των δεδομένων περισσότερες από μία φορές (πλεονασμός, redundancy), οι αρχιτέκτονες και οι προγραμματιστές μπορούν να υλοποιήσουν συστάδες-συμπλέγματα (clusters) σχεσιακών βάσεων δεδομένων με πλεονάζοντα δεδομένα (redundant relational database). Το σύστημα βάσης δεδομένων εγκαθίσταται για χρήση-εκμετάλλευση σε ένα κέντρο δεδομένων (Data Center) με πολύ γρήγορη υποδομή δικτύου. Μία λύση, βέβαια, αυτού του τύπου είναι αρκετά ακριβή.

Στην περίπτωση απαίτησης για ολοκλήρωση-ενσωμάτωση πολλών διαφορετικών τύπων δεδομένων (integration of different data types) από διάφορες πηγές θα μπορούσαν να επιλέξουν να υλοποιήσουν ένα επεκτάσιμο μετα-μοντέλο (meta-model) που θα απλοποιεί τη λειτουργία μιας εταιρικής αποθήκης δεδομένων προσαρμοσμένης στις ανάγκες του οργανισμού. Παράδειγμα ενός τέτοιου μοντέλου είναι το Oracle Retail Data Model.

Σύμφωνα με την Oracle,

«Το μοντέλο Oracle Retail Data Model είναι μια προκατασκευασμένη (προκαθορισμένη) προσέγγιση για την αποθήκευση δεδομένων λιανικής, βασισμένη σε πρότυπα, η οποία επιτρέπει σε μια εταιρεία λιανικής να αποκτήσει-κατανοήσει γρηγορότερα πληροφορίες από τα δεδομένα της. Το μοντέλο μειώνει το κόστος … αξιοποιώντας out-of-box λύσεις αποθήκης δεδομένων και επιχειρηματικής ευφυΐας που βασίζονται στην

Oracle Το μοντέλο μπορεί να χρησιμοποιηθεί σε οποιοδήποτε περιβάλλον εφαρμογής και είναι εύκολα επεκτάσιμο»

("Oracle Retail Data Model is a standards-based, pre-built approach to retail data warehousing enabling a retail company to gain insight from their data more quickly. Oracle Retail Data Model reduces costs for both immediate and on-going operations by leveraging out-of-box Oracle based data warehouse and business intelligence solutions …You can use Oracle Retail Data Model in any application environment. Also, you can easily extend the model").

Το μοντέλο περιλαμβάνει τα ακόλουθα στοιχεία:

- Λογικό μοντέλο το οποίο βασίζεται στα πρότυπα τρίτης κανονικής μορφή (3NF) και μοντέλου οντότητας-αντικειμένου ("a third normal form (3NF) entity-object standards-based model"),
- Φυσικό μοντέλο που ορίζεται ως σχήμα βάσης δεδομένων Oracle (Oracle Database schema),
- Πακέτα εξαγωγής, μετασχηματισμού και φόρτωσης δεδομένων (Intra-ETL database packages) και δέσμες ενεργειών SQL (SQL scripts) οι οποίες εκτελούνται στους φυσικούς πίνακες της τρίτης κανονικής μορφής (3NF) και παράγουν αντικείμενα και ομαδοποιήσεις αντικειμένων (derived and aggregate objects),
- Δείγματα αναφορών (reports) και πινάκων εργαλείων (dashboards) που αναπτύχθηκαν χρησιμοποιώντας Oracle Business Intelligence Suite Enterprise Edition,
- Μοντέλα μοντέλων εξόρυξης δεδομένων (Data Mining Models models) για αναλύσεις υπαλλήλων, πελατών, προϊόντων, αποθήκευσης (store analysis), ποικίλων αντικειμένων (item analysis) και
- DDL και δέσμες ενεργειών εγκατάστασης (installation scripts).

Βλέπε περαιτέρω στοιχεία στο εγχειρίδιο,

Van Raalte, T. (2013) Oracle®Retail Data Model Implementation and Operations Guide, Release 11.3. 2 E20363-03.

Introduction to Oracle Retail Data Model Customization

Η πρόκληση σήμερα έγκειται όχι μόνο στην κατανόησης και τη «διασταύρωση» (τομή, "intersection") όλων αυτών των διαφορετικών τύπων δεδομένων αλλά και στη διαχείριση των δεδομένων όχι μόνον με τους παραδοσιακούς τρόπους αλλά και με νέους. Οι οργανισμοί επινοούν όλο και περισσότερους τρόπους χρήσης αυτών των πληροφοριών και επομένως, πρέπει να εξετάσουμε κάπως διαφορετικά τη διαχείριση των δεδομένων.

Σε αυτήν την κατεύθυνση περιγράφουμε ένα πλαίσιο για τη διαχείριση δεδομένων μεγάλης κλίμακας και επιπλέον παρουσιάζουμε την εξέλιξη της κίνησης των δεδομένων αυτών ("the movement to big data ") και συζητούμε τι μπορεί να σημαίνει η κίνηση αυτή για έναν οργανισμό, σήμερα και στο μέλλον.

Η σύγκλιση των νέων αναδυόμενων τεχνολογιών (emerging technologies) σε συνδυασμό με τη μείωση του κόστους εφαρμογής των νέων τεχνολογιών μπορούν να αλλάξει τον τρόπο με τον οποίο διαχειριζόμαστε και αξιοποιούμε τα δεδομένα. Η διαχείριση δεδομένων μπορεί να αξιοποιεί τα τεχνολογικά επιτεύγματα στο υλικό, στην αποθήκευση δεδομένων, στη δικτύωση και στα μοντέλα υπολογισμού (computing models). Οι τεχνολογίες εικονικοποίησης (vitualization) και η υπολογιστική νέφους (cloud computing) μπορούν άμεσα να συνδυαστούν με την τεχνολογία διαχείρισης και μεγάλων δεδομένων.

Οι τεχνολογίες εικονικοποίησης και η υπολογιστική νέφους παρουσιάζονται σε επόμενη ενότητα του παρόντος κεφαλαίου.

Γενικότερα, οι νέες τεχνολογικές προκλήσεις και η αξιοποίησή τους θα μπορούσαν να συνδεθούν με την απάντηση σε κάποια ερωτήματα, ως εξής:

«Πώς αντιλαμβάνεστε τα μεγάλα δεδομένα; Ποια η σημασία τους για την επιχείρηση και τους ανταγωνιστές σας; Πως θα μπορούσατε εύκολα να αναγνωρίσετε τα μοτίβα (patterns) που έχουν τη μεγαλύτερη σημασία για τις επιχειρηματικές σας αποφάσεις; Πώς θα μπορούσε να αντιμετωπίζει ο οργανισμός σας τις τεράστιες ποσότητες δεδομένων με ουσιαστικό τρόπο;

Στη συνέχεια, παρουσιάζουμε τους τρεις σταθμούς της εξέλιξης της διαχείρισης δεδομένων.

2.3 Τεχνολογίες σχεσιακών συστημάτων βάσεων δεδομένων και η εξέλιξή τους

Από τη δεκαετία του 1970 το σχεσιακό μοντέλο δεδομένων και το σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (RDBMS) κυριαρχούν στη διαχείριση δεδομένων. Το σχεσιακό μοντέλο παρέχει ένα επίπεδο αφαίρεσης (τη γλώσσα δομημένης αναζήτησης (SQL), γεννήτριες αναφορών και άλλα εργαλεία διαχείρισης δεδομένων τα οποία επιτρέπουν ακόμη και σε όχι και τόσο έμπειρους προγραμματιστές να είναι παραγωγικοί και να ικανοποιούν τις συνεχώς αυξανόμενες επιχειρηματικές απαιτήσεις και να εξάγουν αξία από δεδομένα.

Με την ορολογία των τελευταίων ετών, το σχεσιακό μοντέλο προσφέρει ένα οικοσύστημα εργαλείων (ecosystem of tools). Τα εργαλεία παράγονται συνηθέστατα από μεγάλο αριθμό αναδυόμενων εταιρειών λογισμικού και υιοθετούνται, επίσης, από τις μεγάλες εταιρείες-προμηθευτές ΣΔΒΔ. Με τη χρήση των ΣΔΒΔ και των εργαλείων καλύπτεται αφενός μεν η αυξανόμενη ανάγκη καλύτερης οργάνωσης των δεδομένων και των συναλλαγών (transactions) των οργανισμών, αφετέρου δε οι ανάγκες της διοίκησης για την υποστήριξη της λήψης αποφάσεων.

Σε πρώτη φάση, η υιοθέτηση της τρίτης κανονικής μορφής και του μοντέλου οντοτήτων συσχετίσεων, και σε δεύτερη φάση η τεχνολογία αποθηκών δεδομένων και η επιχειρηματική ευφυΐα (τεχνολογίες οι οποίες αξιοποιούνται και στη διαχείριση δεδομένων μεγάλης κλίμακας) προέκυψαν ως απάντηση σε ένα διαχρονικό, «αιώνιο», πρόβλημα, το οποίο συνδέεται με τη ζήτηση για οργάνωση ολοένα και περισσότερων δεδομένων και τη συνεχώς αυξανόμενη ανάγκη για απαντήσεις σε ποικίλα επιχειρηματικά προβλήματα.

Το διαχρονικό πρόβλημα θα μπορούσε να διατυπωθεί ως εξής:

Η αποθήκευση και η εν γένει διαχείριση του συνεχώς αυξανόμενου όγκου των δεδομένων είναι δαπανηρή, η πρόσβαση στα δεδομένα δεν είναι αρκετά γρήγορη και η αξιοποίηση των δεδομένων επιχειρηματικά υπολείπεται των αναγκών και των προσδοκιών της επιχείρησης.

Για παράδειγμα, το σύστημα συνταγογράφησης φαρμάκων στο επίπεδο μιας χώρας απαιτεί ολοένα και μεγαλύτερες επενδύσεις σε υλικό αλλά κυρίως η απάντηση σε κάποια κρίσιμα ερωτήματα των χρηστών-αναλυτών είναι χρονοβόρα. Επιπλέον, απαιτείται πλεονασμός (redundancy) στην αποθήκευση, δηλαδή υιοθετείται η ύπαρξη πολλών αντιγράφων των δεδομένων (data duplication) έτσι ώστε να εξασφαλιστεί η διαρκής πρόσβαση στο σύνολο των δεδομένων ανεξάρτητα από αστοχίες υλικού ή λογισμικού. Τέλος, απαιτείται βελτιστοποίηση των απαντήσεων στα ερωτήματα (query optimization) και η καλύτερη ρύθμιση της βάσης δεδομένων (database tuning).

Όλες αυτές οι απαιτήσεις αυξάνουν το κόστος και γενικότερα επιδεινώνουν τα προβλήματα. Τέλος, η πραγματική επιχειρηματική αξία των δεδομένων είναι συχνά δύσκολο να μετρηθεί. Επιπλέον, τα προβλήματα ασφάλειας και παραβίασης της ιδιωτικότητας είναι μεγάλα (security and privacy).

2.3.1 Τρίτη κανονική μορφή και μοντέλο οντοτήτων-συσχετίσεων

Το πρόβλημα της μεγέθυνσης των δεδομένων και της πολυπλοκότητας της διαχείρισής τους λύθηκε σε πρώτη φάση με τη χρήση του μοντέλου Entity-Relationship (ER) το οποίο παρέχει ένα σύνολο τεχνολογιών για την

υποστήριξη του σχεσιακού μοντέλου. Το μοντέλο πρόσθεσε επιπλέον αφαίρεση (abstraction) στο σχεσιακό μοντέλο η οποία συνεπάγεται την αύξηση της χρηστικότητας των δεδομένων. Σε αυτό το μοντέλο, κάθε στοιχείο ορίζεται ανεξάρτητα από τη χρήση του και οι προγραμματιστές μπορούν εύκολα να δημιουργήσουν νέες συσχετίσεις μεταξύ πηγών δεδομένων (new relationships between data sources) χωρίς αυτό να συνεπάγεται πολύπλοκο προγραμματισμό εφαρμογών.

Για παράδειγμα, θα μπορούσαμε να θεωρήσουμε ότι αρχικά η συσχέτιση μεταξύ διδασκόντων και μαθημάτων σε μία εκπαιδευτική βάση ήταν η παρακάτω:

Lecturer – (1:N) -- > Course

Δηλαδή ένας διδάσκων διδάσκει πολλά μαθήματα και ένα μάθημα διδάσκεται μόνο από έναν διδάσκοντα.

Η μετεγγραφή του ΜΟΣ σε σχεσιακή βάση δεδομένων περιλαμβάνει τους παρακάτω πίνακες:

Πίνακας διδασκόντων Lecturer

LectID	Surname	Name	Specility	……..

Πίνακας μαθημάτων Course

CourseID	Cname	Semester	LectID	……..

Αν αλλάξουν οι απαιτήσεις (οι περιορισμοί, οι επιχειρηματικοί κανόνες) και ένα μάθημα μπορεί να διδάσκεται ταυτόχρονα από περισσότερους διδάσκοντες έτσι ώστε να δημιουργηθούν μικρότερες ομάδες διδασκομένων, τότε η συσχέτιση μεταξύ διδασκόντων και μαθημάτων είναι η παρακάτω:

Lecturer – (M:N) -- > Course

Η μετεγγραφή του νέου ΜΟΣ σε σχεσιακή βάση δεδομένων είναι η παρακάτω:

Πίνακας διδασκόντων Lecturer

LectID	Surname	Name	Specility	……..

Πίνακας μαθημάτων Course

CourseID	Cname	Semester	……..

Πίνακας διδασκαλίας Teach

CourseID	LectID	……..

Το συμπέρασμα που προκύπτει είναι ότι η μετάπτωση (database migration) στο νέο σύστημα δεν είναι πάρα πολύ δύσκολη.

Η εννοιολογική σχεδίαση και η μετεγγραφή του ΜΟΣ σε σχεσιακή βάση συνετέλεσε αποφασιστικά στην τεράστια πρόοδο, επέτρεψε στους προγραμματιστές να ξεπεράσουν τα όρια της τεχνολογίας στο παρελθόν και επιπλέον η αγορά των σχεσιακών βάσεων δεδομένων εκτοξεύθηκε και παραμένει ενεργή και ιδιαίτερα σημαντική όσο ποτέ για τη διαχείριση δομημένων δεδομένων συναλλαγών (transactional data management).

2.3.2 Online Transaction Processing

Αν και τα περισσότερα διαθέσιμα δεδομένα σήμερα δεν είναι δομημένα οι επιχειρήσεις παραδοσιακά επενδύουν σε συστήματα διαχείρισης δομημένων δεδομένων επειδή τα δεδομένα αυτά συνδέονται στενά με τα συστήματα διαχείρισης συναλλαγών στα οποία βασίζεται η καθημερινή λειτουργία της επιχείρησης. Η

ορολογία για τα συστήματα αυτά περιλαμβάνει τον όρο Online Transaction Processing System αλλά και όρους όπως line-of-business transactional systems ή line-of-business transaction systems.

Σύμφωνα με την Oracle, «Η άμεση ηλεκτρονική επεξεργασία συναλλαγών, OLTP, είναι ένας τύπος επεξεργασίας δεδομένων που συνίσταται στην εκτέλεση ενός αριθμού συναλλαγών που πραγματοποιούνται ταυτόχρονα—για παράδειγμα, τραπεζικές συναλλαγές μέσω διαδικτύου, αγορές, εισαγωγή παραγγελιών ή αποστολή μηνυμάτων κειμένου. Αυτές οι συναλλαγές αναφέρονται παραδοσιακά ως οικονομικές ή χρηματοοικονομικές συναλλαγές, καταγράφονται και διασφαλίζονται έτσι ώστε μια επιχείρηση να μπορεί να έχει πρόσβαση στις πληροφορίες ανά πάσα στιγμή για λογιστικούς σκοπούς ή σκοπούς αναφοράς» ("OLTP or Online Transaction Processing is a type of data processing that consists of executing a number of transactions occurring concurrently—online banking, shopping, order entry, or sending text messages, for example. These transactions traditionally are referred to as economic or financial transactions, recorded and secured so that an enterprise can access the information anytime for accounting or reporting purposes").

Είναι ενδιαφέρον να δούμε την άποψη της Oracle για την εξέλιξη του ορισμού:

«Στο παρελθόν, το OLTP περιοριζόταν σε πραγματικές αλληλεπιδράσεις στις οποίες ανταλλάσσονταν κάτι – χρήματα, προϊόντα, πληροφορίες, αιτήματα για υπηρεσίες κ.λπ. Ωστόσο, ο ορισμός της συναλλαγής σε αυτό το πλαίσιο έχει επεκταθεί με την πάροδο των ετών, ειδικά μετά την εμφάνιση του διαδικτύου, για να συμπεριλάβει κάθε είδους ψηφιακή αλληλεπίδραση ή δέσμευση με μια επιχείρηση που μπορεί να ενεργοποιηθεί από οπουδήποτε στον κόσμο και μέσω οποιουδήποτε αισθητήρα συνδεδεμένου στο διαδίκτυο . Περιλαμβάνει επίσης κάθε είδους αλληλεπίδραση ή ενέργεια, όπως λήψη pdf σε μια ιστοσελίδα, προβολή συγκεκριμένου βίντεο ή αυτόματη συντήρηση ή σχόλια σε κανάλια κοινωνικής δικτύωσης που ίσως είναι κρίσιμα για μια επιχείρηση να καταγράψει για να εξυπηρετήσει καλύτερα τους πελάτες της».

("In the past, OLTP was limited to real-world interactions in which something was exchanged–money, products, information, request for services, and so on. But the definition of transaction in this context has expanded over the years, especially since the advent of the internet, to encompass any kind of digital interaction or engagement with a business that can be triggered from anywhere in the world and via any web-connected sensor. It also includes any kind of interaction or action such as downloading pdfs on a web page, viewing a specific video, or automatic maintenance triggers or comments on social channels that maybe critical for a business to record to serve their customers better".)

What is Online Transaction Processing (OLTP) | Oracle

Στα συστήματα line-of-business transactional systems ή line-of-business transaction systems ο όρος γραμμή επιχείρησης ή επιχειρηματική γραμμή (Line of business, LOB) είναι ένας γενικός όρος που αναφέρεται σε ένα προϊόν ή ένα σύνολο σχετικών προϊόντων που εξυπηρετούν συγκεκριμένες συναλλαγές πελατών (transactions). ή μια επιχειρηματική ανάγκη.

Σε ένα γενικότερο πλαίσιο, ο όρος "Line of business" αναφέρεται σε μια εσωτερική εταιρική επιχειρηματική μονάδα (σημείωση, ενώ ο όρος "industry" ("βιομηχανία") αναφέρεται στο εξωτερικό περιβάλλον, δηλαδή στους ανταγωνιστές της επιχείρησης). Στην περίπτωση αυτή, ο όρος συνδέεται με/αφορά μια επιχειρηματική ανάγκη και επομένως περιλαμβάνει και την εξέταση της θέσης της μονάδας σε έναν κλάδο χρησιμοποιώντας αναλύσεις όπως PESTLE, SWOT, Porter five forces analysis κ.λπ.

Τα συστήματα OLTP παρουσιάζονται σε επόμενη ενότητα του κεφαλαίου.

2.3.3 Αποθήκη δεδομένων (data warehouse) και πρατήρια δεδομένων (data marts)

Όταν ο όγκος των δομημένων δεδομένων που διαχειρίζονταν οι οργανισμοί αυξήθηκε, συχνά εκτός ελέγχου, η τεχνολογία της αποθήκης δεδομένων, η οποία εφαρμόζεται από τη δεκαετία του 1990 μέχρι και σήμερα, επέτρεψε στον οργανισμό να επιλέξει ένα υποσύνολο των δεδομένων πιο εστιασμένο σε συγκεκριμένους τομείς της επιχείρησης. Τα απαραίτητα δεδομένα αποθηκεύονται έτσι ώστε να είναι ευκολότερο για την επιχείρηση να τα αναλύσει και να πάρει αποφάσεις για αύξηση της απόδοσης (performance).

Με τις αποθήκες δεδομένων εισάγεται η έννοια του διαχωρισμού των δομημένων δεδομένων τα οποία συνδέονται με την καθημερινή λειτουργίας της επιχείρησης, από τα δεδομένα της υποστήριξης αποφάσεων (decision support). Οι αποθήκες δεδομένων αποθηκεύουν δεδομένα και από προηγούμενα έτη και επιπλέον παρέχουν μια ολοκληρωμένη πηγή πληροφοριών (integrated source of information) από διάφορες πηγές δεδομένων που μπορούν να χρησιμοποιηθούν για την ανάλυση και την κατανόηση της απόδοσης του οργανισμού.

Η πρόοδος σε νεότερες τεχνολογίες όπως η τεχνολογία επεκτασιμότητας του υλικού (scalability), η τεχνολογία εικονικοποίησης (virtualization), η τεχνολογία δημιουργίας ολοκληρωμένων συστημάτων υλικού και λογισμικού για τη διαχείριση δεδομένων (είναι γνωστά ως appliances) έδωσαν μεγάλη ώθηση και στην τεχνολογία των αποθηκών δεδομένων.

Συχνά οι αποθήκες δεδομένων είναι περίπλοκες και μεγάλες και έτσι δημιουργούνται θεματικά πρατήρια δεδομένων (data marts) τα οποία επικεντρώνονται σε συγκεκριμένα επιχειρηματικά ζητήματα και προσφέρουν την ταχύτητα και την ευελιξία που απαιτεί η επιχείρηση.

2.3.3.1 Η τεχνολογία των αποθηκών σήμερα και τα μεγάλα δεδομένα. Data appliances

Η τεχνολογία των αποθηκών και των πρατηρίων δεδομένων έχει εξελιχθεί και υποστηρίζει νέες αναδυόμενες τεχνολογίες, όπως ολοκληρωμένα συστήματα (integrated systems) και τις αρχιτεκτονικές data appliances. Στην εικόνα 2.7 βλέπουμε το εννοιολογικό πλαίσιο για την αρχιτεκτονική επεξεργασίας του φόρτου εργασίας (workloads) στη διαχείριση δεδομένων μεγάλης κλίμακας. Η εικόνα είναι διασκευή της εικόνας (Figure 10.8) από το άρθρο,

Krishnan, K. (2013) Integration of Big Data and Data Warehousing, Data Warehousing in the Age of Big Data.

Στην εικόνα 2.7 διακρίνουμε το σύστημα διαχείρισης συναλλαγών OLTP το οποίο τροφοδοτεί το σύστημα ETL ή ELT ή CDC. Με τον όρο ETL-Extract Transform Load εννοούμε τη διαδικασία και τα εργαλεία τα οποία εξάγουν δεδομένα από ένα σύστημα πηγής, μετατρέπουν τα δεδομένα κατά τη μεταφορά και φορτώνουν τα δεδομένα σε βάση δεδομένων-στόχο. Σύμφωνα με την IBM,

«Οι διαδικασίες ELT (εξαγωγή, φόρτωση, μετασχηματισμός) και ETL (εξαγωγή, μετασχηματισμός, φόρτωση) είναι και οι δύο διαδικασίες ενοποίησης δεδομένων που μεταφέρουν ακατέργαστα δεδομένα από ένα σύστημα πηγής σε μια βάση δεδομένων-στόχο, π.χ., μια λίμνη δεδομένων ή μια αποθήκη δεδομένων. Αυτές οι πηγές δεδομένων μπορεί να βρίσκονται σε πολλαπλά, διαφορετικά αποθετήρια ή σε συστήματα παλαιού τύπου. Τα δεδομένα από τις πηγές μεταφέρονται με χρήση ELT ή ETL σε μια τοποθεσία δεδομένων-στόχου».

("ELT (extract, load, transform) and ETL (extract, transform, load) are both data integration processes that move raw data from a source system to a target database, such as a data lake or data warehouse. These data sources can be in multiple, different repositories or in legacy systems that are then transferred using ELT or ETL to a target data location".)

ELT vs. ETL: What's the Difference? | IBM

Ο όρος Change data capture (CDC) αναφέρεται σε διαδικασία η οποία ασχολείται μόνο με αλλαγές δεδομένων. Πιο συγκεκριμένα, είναι η διαδικασία 1) «σύλληψης» (capturing) και καταγραφής των αλλαγών που έγιναν στην πηγή δεδομένων και 2) της εφαρμογής των αλλαγών στην επιχείρηση. Η διαδικασία CDC ελαχιστοποιεί τους πόρους που απαιτούνται για τις διαδικασίες ETL (ή ELT) επειδή ασχολείται μόνο με αλλαγές δεδομένων.

Το σύστημα Acquire αναλαμβάνει την παραγωγή/τροφοδοσία από πηγές μεγάλων δεδομένων, π.χ., αισθητήρες, εξωτερικές πηγές.

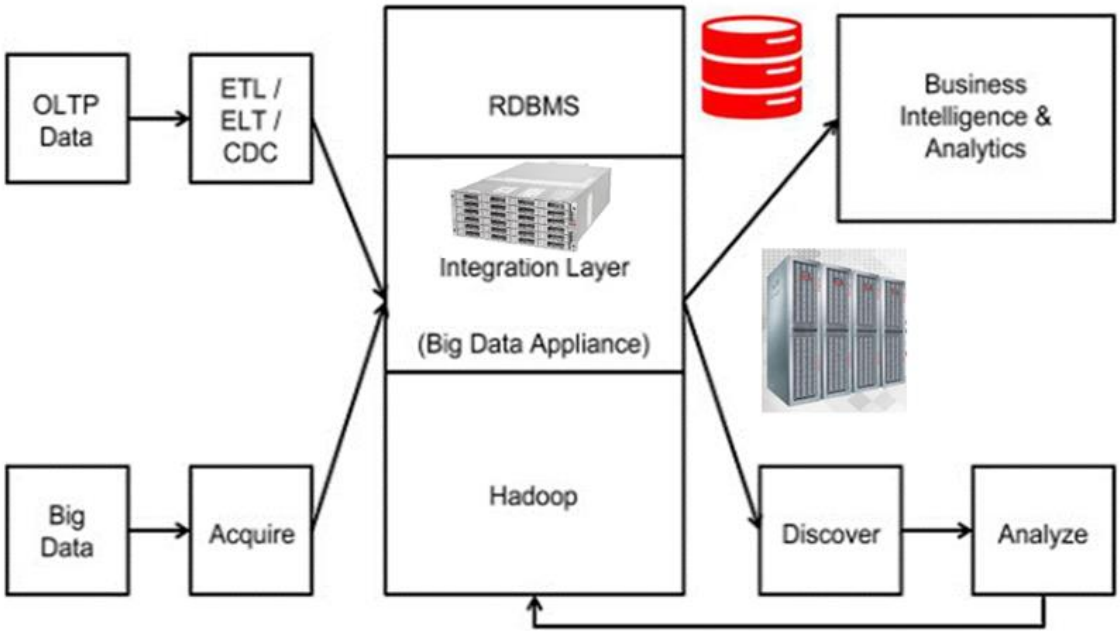
Το επίπεδο-σύστημα ολοκλήρωσης περιλαμβάνει σχεσιακό ΣΔΒΔ, σύστημα Hadoop για τη διαχείριση μεγάλων δεδομένων και Big Data appliances.

Το σύστημα BI &Analytics αναλαμβάνει 1) τη διαχείριση δεδομένων, δηλαδή επιλεγμένων λειτουργικών δεδομένων (operational data) αλλά και ιστορικών δεδομένων (επιλογή από τα λειτουργικά δεδομένα του παρελθόντος) της επιχείρησης, 2) χρησιμοποιεί στατιστικά στοιχεία, οπτικοποίηση δεδομένων, ανάλυση ακατέργαστων πληροφοριών και 3) αναλαμβάνει την παροχή πληροφοριών στη διοίκηση για την υποστήριξη αποφάσεων.

Τα συστήματα Discover, Analyse αναλαμβάνουν, αντίστοιχα, τις διαδικασίες οι οποίες θα διασφαλίσουν ότι τα δεδομένα μπορούν να εντοπιστούν από τους ενδιαφερόμενους και την εφαρμογή μεθόδων ανάλυσης.

Στη συνέχεια επεξηγούμε σύντομα τις έννοιες Data warehouse appliances, Big Data appliances.

Οι «συσκευές αποθήκευσης και διαχείρισης δεδομένων» (βλέπε και https://www.sciencedirect.com/topics/computer-science/data-warehouse-appliance) εμφανίστηκαν ως μια ισχυρή αρχιτεκτονική μαύρου κουτιού για την επεξεργασία φόρτου εργασίας (workloads) ειδικά για δεδομένα μεγάλης κλίμακας. Μία από τις επεκτάσεις αυτής της αρχιτεκτονικής είναι η εμφάνιση των «συσκευών Big Data» (Big Data appliances). Αυτές οι συσκευές έχουν διαμορφωθεί για να διαχειρίζονται ταυτόχρονα την «ακαμψία» του φόρτου εργασίας (workloads), την πολυπλοκότητα των Big Data και την τρέχουσα αρχιτεκτονική RDBMS. Η εικόνα 2.7 είναι διασκευή της εικόνας από το άρθρο Integration of Big Data and Data Warehousing (Krish Krishnan (2013) Data Warehousing in the Age of Big Data).



Εικόνα 2.7 Διασκευή της εικόνας Conceptual Big Data appliance από το άρθρο Integration of Big Data and Data Warehousing του Krish Krishnan (Figure 10.8)

Δείτε σχετικά και την παρακάτω βιβλιογραφία για Data Appliance:

Krishnan, K. (2013) Integration of Big Data and Data Warehousing, In *Data Warehousing in the Age of Big Data*

Curtis, B., Arshad, F., Benner, E., Elsins, M., Gallagher, M., Sharman, P., & Velikanov, Y. (2014). Business Values for the ODA. In Practical Oracle Database Appliance (pp. 159-172). Apress, Berkeley, CA. Practical Oracle Database Appliance. Expert's voice in Oracle. Apress ISBN 9781430262657

Oracle (2013) Oracle Database Appliance-An Oracle White Paper "Oracle Database Appliance White Paper"

Peters, Mark (2020) White Paper: Oracle Database Appliance X8 Replaces the Pains of Do-it-yourself Infrastructure with Operational Gains esg-white-paper-oracle-aug-2020.pdf

Curtis, B., Arshad, F., Benner, E., Elsins, M., Gallagher, M., Sharman, P., & Velikanov, Y. (2014). Oracle Database Appliance. In Practical Oracle Database Appliance (pp. 1-12). Apress, Berkeley, CA.

Ακολουθούν κάποιες παρατηρήσεις και συμπεράσματα.

Οι αποθήκες δεδομένων και τα πρατήρια δεδομένων (Data warehouses and data marts) λύνουν πολλά προβλήματα των επιχειρήσεων οι οποίες χρειάζονται έναν συνεπή τρόπο διαχείρισης μαζικών δεδομένων συναλλαγών (massive transactional data). Αλλά όταν πρόκειται για τη διαχείριση τεράστιων όγκων αδόμητων ή ημιδομημένων δεδομένων, η αποθήκη δεν μπορεί να εξελιχθεί αρκετά για να ανταποκριθεί στις μεταβαλλόμενες απαιτήσεις. Επιπλέον, οι αποθήκες δεδομένων συνήθως τροφοδοτούνται σε τακτά χρονικά διαστήματα, συνήθως ανά εβδομάδα ή καθημερινά. Επομένως μπορούν να καλύψουν τις ανάγκες προγραμματισμού (planning), τις αναφορές οικονομικών στοιχείων (financial reporting) και τις συνήθεις καμπάνιες μάρκετινγκ, αλλά συχνά είναι αργές και δεν καλύπτουν τις ανάγκες των επιχειρηματικών εφαρμογών πραγματικού χρόνου (real-time business).

Επειδή οι τεχνολογίες «λίμνες δεδομένων» (Data lakes) και «αποθήκες δεδομένων» (data warehouses) θα μπορούσαν να χρησιμοποιηθούν για την αποθήκευση μεγάλων δεδομένων, σύμφωνα με τις ανάγκες μας είναι σκόπιμο να ξεκαθαρίσουμε τη διαφορά ανάμεσα στις δύο έννοιες.

Η αποθήκη δεδομένων αποθηκεύει δομημένα, φιλτραρισμένα δεδομένα μετά από επεξεργασία, π.χ., διαδικασία ETL, για συγκεκριμένο σκοπό.

Μια λίμνη δεδομένων είναι μια τεράστια δεξαμενή ακατέργαστων δεδομένων, η οποία περιλαμβάνει δομημένα, ημι-δομημένα και μη δομημένα δεδομένα, και επιπλέον, συχνά ο σκοπός της δεν έχει ακόμη καθοριστεί τελείως.

Τέλος, η τεχνολογία επιχειρηματικής ευφυίας συνδέεται άμεσα με τις αποθήκες δεδομένων και η χρήση της δίνει συγκριτικό πλεονέκτημα στις εταιρίες επειδή είναι προσαρμοσμένη στη διαδικασία υποστήριξης αποφάσεων.

Σε ξεχωριστή ενότητα του κεφαλαίου 2, την ενότητα 2.14, θα γίνει σύντομη συγκριτική παρουσίαση της σχέσης βάσεων δεδομένων, επιχείρησης, data mining, data warehouse, Datamart, business intelligence, knowledge management.

2.3.4 Object Oriented Database Systems

Οι προμηθευτές ΣΔΒΔ σε μία διαρκή προσπάθεια να μετασχηματίσουν-βελτιώσουν τις παραδοσιακές προσεγγίσεις διαχείρισης δεδομένων για να χειριστούν αποτελεσματικά τον αυξανόμενο όγκο των μη δομημένων δεδομένων χρησιμοποίησαν διάφορες λύσεις, όπως η προσθήκη της δυνατότητας διαχείρισης δυαδικών μεγάλων αντικειμένων (BLOB-binary large objects). Για να κατανοήσουμε καλύτερα τα BLOB

μπορούμε να θυμηθούμε την ορολογία chunk of data, μια ορολογία γνωστή από τα πολυμέσα, τα συστήματα βάσης δεδομένων, τον κατανεμημένο υπολογισμό κ.λπ.

Για παράδειγμα, ο όρος chunk αναφέρεται σε ένα «κομμάτι-τμήμα πληροφοριών» το οποίο χρησιμοποιείται σε πολλές μορφές αρχείων πολυμέσων, π.χ., PNG, TIFF, MP3, AVI. Κάθε «κομμάτι» περιέχει μια κεφαλίδα (header) η οποία ορίζει παραμέτρους, π.χ., τύπο του «κομματιού», σχόλια, μέγεθος «κομματιού», και ακολουθείται από μια περιοχή μεταβλητού μήκους η οποία περιλαμβάνει τα δεδομένα. Το πρόγραμμα διαχείρισης αποκωδικοποιεί τα δεδομένα χρησιμοποιώντας τις παραμέτρους που ορίζονται στην κεφαλίδα.

Σε μια σχεσιακή βάση δεδομένων ένα μη δομημένο στοιχείο δεδομένων μπορεί να αποθηκευτεί ως ένα συνεχόμενο κομμάτι δεδομένων (contiguous chunk of data) και δεν θα μπορούσατε να δείτε το περιεχόμενο (τι υπάρχει μέσα σε αυτό το αντικείμενο) και να το επεξεργαστείτε. Δηλαδή, σε μια παραδοσιακή σχεσιακή βάση δεδομένων το BLOB είναι απλά μια προσαρτημένη, ως ανεξάρτητη, μονάδα. Σαφώς, το BLOB είναι πολύ χρήσιμο αλλά δεν μπορεί να επιλύσει τις μεταβαλλόμενες επιχειρηματικές ανάγκες.

Η εισαγωγή του αντικειμενοστρεφούς συστήματος διαχείρισης βάσης δεδομένων αντικειμένων, object database management system (ODBMS), προσπάθησε να επιλύσει το πρόβλημα. Η αντικειμενοστρεφής βάση δεδομένων αποθηκεύει το BLOB ως «ένα διευθυνσιοδοτούμενο σύνολο κομματιών» (addressable set of pieces) στο οποίο μπορούμε να έχουμε πρόσβαση («να δούμε τι υπάρχει»), δηλαδή το αντικειμενοστρεφές μοντέλο βάσης δεδομένων παρέχει μια ενοποιημένη προσέγγιση για τη διαχείριση δομημένων και μη δομημένων δεδομένων. Το αντικειμενοστρεφές σύστημα διαχείρισης βάσης δεδομένων περιλαμβάνει μία γλώσσα προγραμματισμού και μια δομή για τα στοιχεία δεδομένων, έτσι ώστε να είναι ευκολότερος ο χειρισμός διαφόρων αντικειμένων δεδομένων. Επομένως, με τα συστήματα αυτά εισάγεται ένα καινοτομικό στοιχείο το οποίο βελτιώνει τη διαχείριση δεδομένων. Τα παραδοσιακά ΣΔΒΔ υιοθετούν, σήμερα, πολλά χαρακτηριστικά των αντικειμενοστρεφών. Στη συνέχεια παραθέτουμε παράδειγμα αντικειμενοστρεφούς μοντέλου σε Oracle. Το παράδειγμα περιλαμβάνεται στο εγχειρίδιο χρήσης (reference manual) της Oracle 8i. Introduction to Oracle Objects

Στο παράδειγμα βλέπουμε πως μπορούμε να ορίσουμε τους τύπους αντικειμένων (object types) PERSON, LINEITEM, LINEITEM_TABLE, and PURCHASE_ORDER. Τα χαρακτηριστικά (attributes) των τύπων είναι NAME, PHONE, ITEM_NAME, κ.λπ. Το χαρακτηριστικό (attribute) CONTACT είναι ένα αντικείμενο (object), και το χαρακτηριστικό (attribute) LINEITEMS είναι ένας ένθετος πίνακας (nested table). Ο πίνακας LINEITEM_TABLE είναι ένας πίνακας στον οποίο κάθε γραμμή είναι ένα αντικείμενο τύπου LINEITEM.

```
CREATE TYPE person AS OBJECT (
  nameVARCHAR2(30),
  phone   VARCHAR2(20) );

CREATE TYPE lineitem AS OBJECT (
  item_name   VARCHAR2(30),
  quantityNUMBER,
  unit_price  NUMBER(12,2) );

CREATE TYPE lineitem_table AS TABLE OF lineitem;

CREATE TYPE purchase_order AS OBJECT (
  id  NUMBER,
  contact person,
  lineitems   lineitem_table,

  MEMBER FUNCTION
  get_value   RETURN NUMBER );
```

Το παράδειγμα είναι απλοποιημένο και δεν περιλαμβάνει το σώμα (body) της μεθόδου GET_VALUE. Για να ορίσουμε το σώμα χρησιμοποιούμε τη δήλωση CREATE OR REPLACE TYPE BODY. Μπορούμε να χρησιμοποιήσουμε LINEITEM, PERSON, PURCHASE_ORDER σε δηλώσεις SQL κατά το πλείστον όπως χρησιμοποιούμε τους τύπους NUMBER ή VARCHAR2. Για παράδειγμα, θα μπορούσαμε να ορίσουμε τον πίνακα ιχνηλάτησης των επαφών μας:

```
CREATE TABLE contacts (
  contact person
  dateDATE );
```

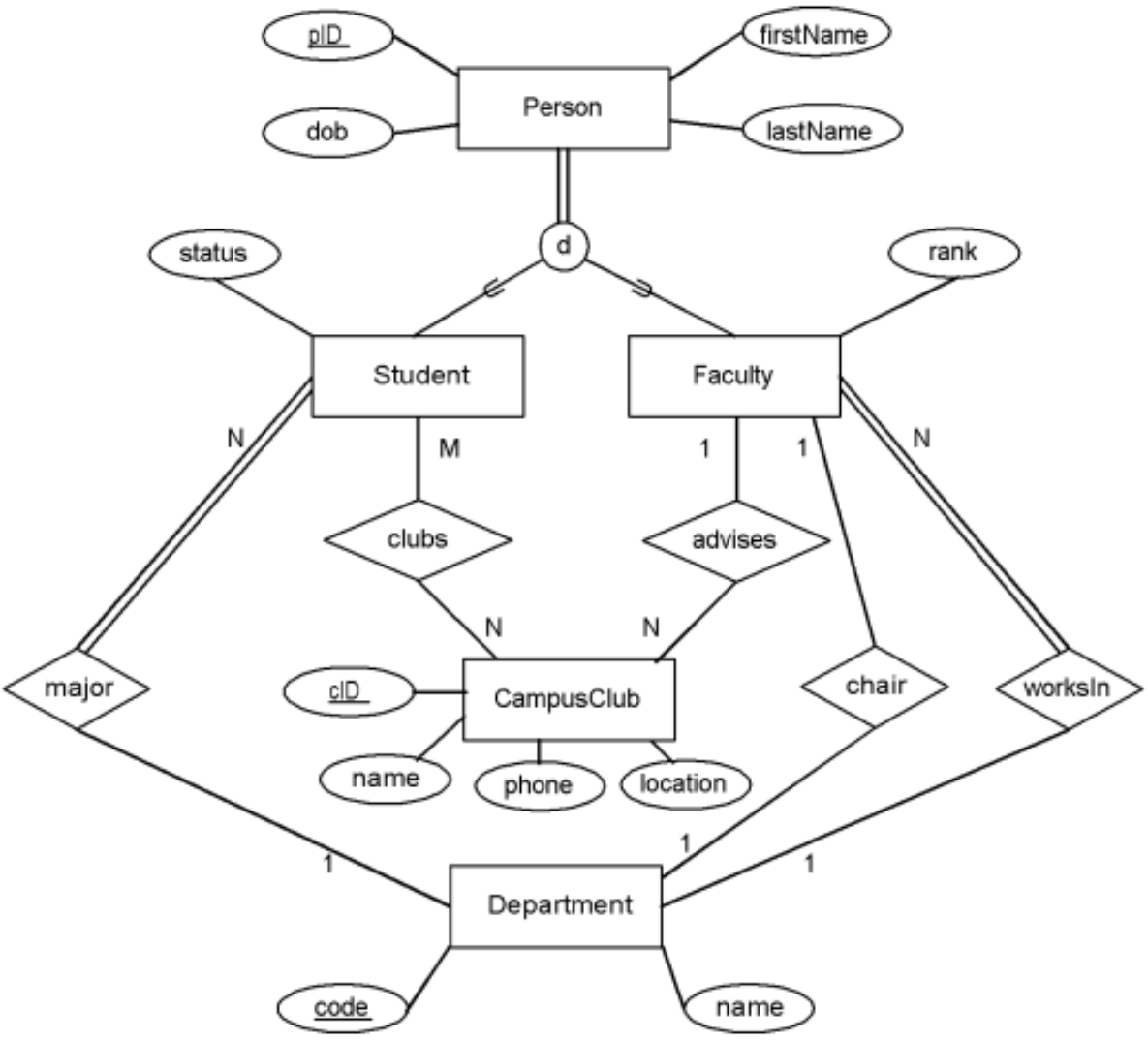
Οι Urban and Dietrich (2004) σκιαγραφούν τον τρόπο με τον οποίο η μεθοδολογία UML μπορεί να χρησιμοποιηθεί για να μελετηθεί η αντιστοίχιση εννοιολογικών μοντέλων σε διαφορετικούς τύπους συστημάτων βάσεων δεδομένων. Παραθέτουν το μοντέλο EER μίας βάσης δεδομένων που περιλαμβάνει φοιτητικό πληθυσμό, εκπαιδευτικό προσωπικό, τμήματα και λέσχες (club) που λειτουργούν στο πανεπιστήμιο (Εικόνα 2.8). Χρησιμοποιούν διαγράμματα UML για την αντιστοίχιση εννοιολογικών σχεδίων στο σχεσιακό μοντέλο, στο αντικειμενοστραφές μοντέλο και στο αντικειμενοσχεσιακό μοντέλο (object-relational model). Για αντικειμενοστραφή και αντικειμενοσχεσιακά συστήματα βάσεων δεδομένων, περιγράφουν τις αντιστοιχίσεις στα πρότυπα ODMG και SQL99.

Παραθέτουμε κάποια από τα αντίστοιχα διαγράμματα από το άρθρο,

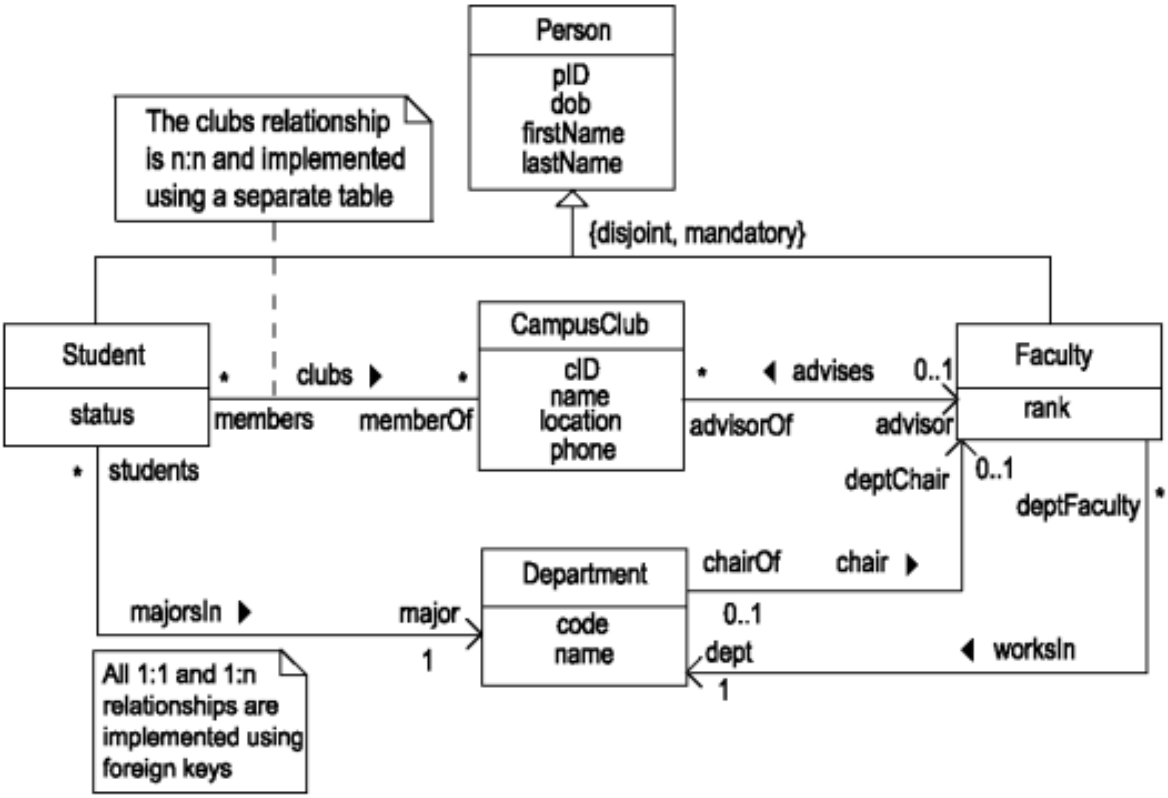
Susan D. Urban and Suzanne W. Dietrich (2004) Using UML Class Diagrams for a Comparative Analysis of Relational, Object-Oriented, and Object-Relational Database Mappings, SIGCSE'03, February 19-23, 2003, Reno, Nevada, USA.

Microsoft Word - p144Urban.doc (acm.org)

Στην εικόνα 2.8 βλέπουμε το επεκτεταμένο ΜΟΣ (Enhanced Enity Relationship Diagram) βάσης δεδομένων Σχολής. Στην εικόνα 2.9 βλέπουμε το διάγραμμα UML για τη σχεσιακή υλοποίηση, Στην εικόνα 2.10 βλέπουμε το διάγραμμα UML για αντικειμενοστρεφή υλοποίηση.



Εικόνα 2.8 Επεκτεταμένο ΜΟΣ (Enhanced Enity Relationship Diagram) βάσης δεδομένων Σχολής (School Database, Urban & Dietrich)



Εικόνα 2.9 Διάγραμμα UML για σχεσιακή υλοποίηση (School Database, Relational Implementation, Urban & Dietrich)

Παραθέτουμε το αντίστοιχο σχεσιακό σχήμα.

Person (pID, dob, firstName, lastName)

Student (pID, status, major), foreign key (pID) references Person(pID),

foreign key(major) references Department(code)

Faculty (pID, rank, dept), foreign key(pID) references Person(pID),

foreign key(dept) references Department(code)

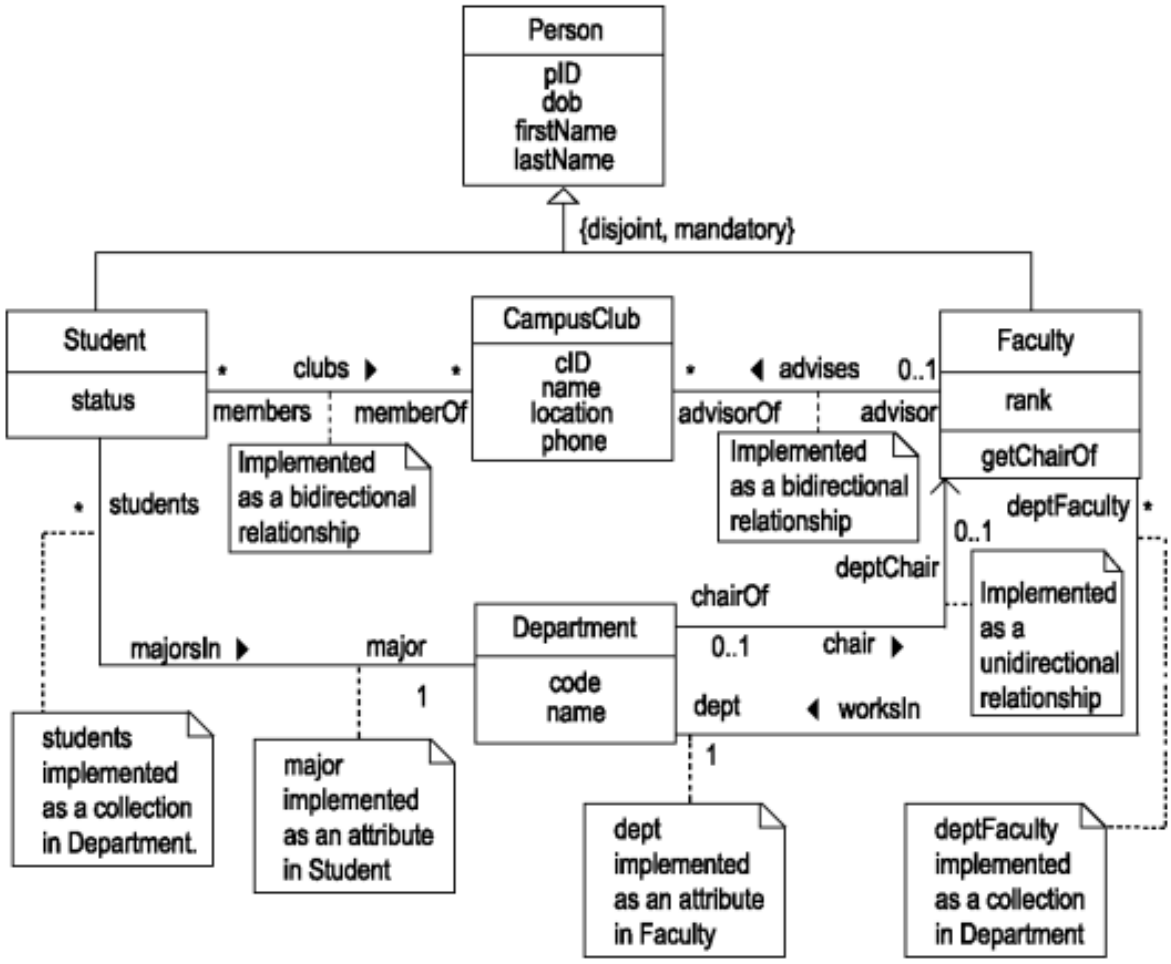
Department (code, name, chair), foreign key (chair) references Faculty(pID)

CampusClub (cID, name, phone, location, advisor),

foreign key(advisor) references Faculty(pID)

Clubs(pID, cID), foreign key(pID) references Student(pID),

foreign key(cID) references CampusClub(cID)



Εικόνα 2.10 Διάγραμμα UML για αντικειμενοστρεφή υλοποίηση (School Database, Object-Oriented Implementation, Urban & Dietrich)

2.4 Παγκόσμιος ιστός Web, σημασιολογικός ιστός και τα συστήματα διαχείρισης περιεχομένου

Ο παγκόσμιος ιστός Web αποτελεί μία επανάσταση στη διαχείριση δεδομένων και όχι μόνο. Οι τεχνολογίες του σημασιολογικού ιστού (semantic WEB) αποτελούν μία δεύτερη επανάσταση η οποία σφραγίζει και το παρόν και το μέλλον της διαχείρισης δεδομένων. Οι ενδιαφερόμενοι αναγνώστες για εμβάθυνση στις τεχνολογίες σημασιολογικού ιστού και στα συνδεδεμένα δεδομένα παραπέμπονται στα παρακάτω αναλυτικά συγγράμματα των εκδόσεων «Κάλλιπος»:

Πούλος, Μ., (2015). *Σημασιολογική επεξεργασία της πληροφορίας*. [ηλεκτρ. βιβλ.] Αθήνα: Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών. Το σύγγραμμα εστιάζει σε ανάκτηση πληροφορίας, συντακτική και στατιστική επεξεργασία της πληροφορίας, υπολογιστική και ποσοτική γλωσσολογία, τεχνολογίες διαδικτύου, οντολογίες και εφαρμογές.

Καπιδάκης, Σ., Λαζαρίνης, Φ., Τοράκη, Κ., (2015). *Θέματα βιβλιοθηκονομίας και επιστήμης των πληροφοριών*. [ηλεκτρ. βιβλ.] Αθήνα: Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών. Το σύγγραμμα περιγράφει με αναλυτικό τρόπο πάρα πολλά θέματα που αναφέρονται σύντομα στο Κεφάλαιο 2. Ειδικότερα προτείνεται η μελέτη θεμάτων μεταδεδομένων, γλώσσας XML, RDF, Dublin Core, ελεγχόμενων λεξιλογίων κ.λπ.

Στάμου, Γ., (2015). *Αναπαράσταση οντολογικής γνώσης και συλλογιστική. Βασική αναπαράσταση γνώσης και συλλογιστική, ευφυή συστήματα.* [ηλεκτρ. βιβλ.] Αθήνα: Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών. Το σύγγραμμα εστιάζει σε αναπαράσταση γνώσης και συλλογιστικής με χρήση περιγραφικών λογικών.

Στεφανιδάκης, Μ., Ανδρόνικος, Θ., Παπαδάκης, Ι. (2015). *Ανοικτά συνδεδεμένα δεδομένα και εφαρμογές*. [ηλεκτρ. βιβλ.] Αθήνα: Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών. Το σύγγραμμα περιγράφει με αναλυτικό τρόπο πάρα πολλά θέματα συνδεδεμένων δεδομένων, οντολογιών κ.λπ.

Ακολουθούν και άλλες προτάσεις για περαιτέρω μελέτη:

Ένα ενδιαφέρον σύγγραμμα για συνδεδεμένα δεδομένα είναι το σύγγραμμα:

Wood, D., Zaidman, M., Ruth, L. and Hausenblas, M. (2013). *Linked Data: Structured data on the Web*. Manning Publications, https://www.manning.com/books/linked-data

Ένα ενδιαφέρον σύγγραμμα για δομημένα δεδομένα στον σημασιολογικό ιστό με πολλά παραδείγματα είναι το σύγγραμμα:

Sikos, L.F. (2015). *Mastering Structured Data on the Semantic Web*. Apress. ISBN: 9781484210505

Οι ενδιαφερόμενοι αναγνώστες για ψηφιακές βιβλιοθήκες και οργάνωση γνώσης (Digital Libraries and Knowledge Organization) παραπέμπονται στο σύγγραμμα: Kruk, S.R., McDaniel, B. (eds.) (2009). *Semantic Digital Libraries*. Springer.

Περιλαμβάνονται και άρθρα για Semantic Web, Ontologies, Social Semantic Information Spaces, κ.λπ.

Οι αναγνώστες χωρίς προηγούμενη γνώση στο θέμα θα βρουν χρήσιμη τη μελέτη των αντίστοιχων κεφαλαίων του συγγράμματος:

Μαρινάγη, Α., Σκουρλάς, Χ. (2022), Διαχείριση γνώσης, Κάλλιπος

Οι αναγνώστες χωρίς τεχνικό υπόβαθρο σε θέματα σημασιολογικού ιστού θα βρουν χρήσιμη μελέτη βιβλίων όπως,

Pollock, J.T. (2009). Semantic Web For Dummies. Wiley Publishing, Inc.

Ο σημασιολογικός ιστός θα μας απασχολήσει εκτενέστερα στην ενότητα 2.9 του παρόντος συγγράμματος.

2.4.1 Συστήματα διαχείρισης εταιρικού περιεχομένου (Enterprise Content Management systems)

Τα συστήματα διαχείρισης εταιρικού περιεχομένου, Enterprise Content Management systems, άρχισαν να εξελίσσονται από τη δεκαετία του 1980, παρέχοντας στις επιχειρήσεις τη δυνατότητα να διαχειρίζονται εύκολα τα έγγραφα (documents). Στη δεκαετία του 1990 με την άνοδο του Ιστού οδήγησε στην αναγκαιότητα διαχείρισης, πέραν των εγγράφων, περιεχομένου του ιστού, εικόνων, ήχου και βίντεο.

Υιοθετείται, σήμερα, το ενοποιημένο μοντέλο, το οποίο και παραμένει εξαιρετικά δημοφιλές μέχρι σήμερα, και σε μια πλατφόρμα ενσωματώνεται η διαχείριση των επιχειρηματικών διαδικασιών (business process management), ο έλεγχος των εκδόσεων (version control), η αναγνώριση της πληροφορίας (information recognition), η διαχείριση κειμένου (text management) και η συνεργασία (collaboration). Η νέα γενιά συστημάτων προσθέτει τα μεταδεδομένα (δηλαδή τις πληροφορίες σχετικά με την οργάνωση και τα χαρακτηριστικά των αποθηκευμένων πληροφοριών).

Η σύγκλιση τεχνολογιών-παραγόντων, όπως ο σημασιολογικός ιστός, η εικονικοποίηση (virualization) και η υπολογιστική νέφους (cloud computing) οδηγεί σήμερα στην ανάδυση μιας νέας γενιάς απαιτήσεων για το επερχόμενο «κύμα» (“the next wave”) και οι οργανισμοί αρχίζουν να κατανοούν την αναγκαιότητα διαχείρισης δεδομένων μεγάλης κλίμακας

Η τεχνολογία των συστημάτων διαχείρισης περιεχομένου (content management systems) προσέφερε ένα νέο εργαλείο στους προγραμματιστές και διευκόλυνε την ανάπτυξη εφαρμογών και τη διαχείριση περιεχομένου.

Σύμφωνα με την Adobe,

«Ενα σύστημα διαχείρισης περιεχομένου είναι μια εφαρμογή λογισμικού ή ένα σύνολο εργαλείων και δυνατοτήτων, που σας επιτρέπει να δημιουργείτε, να διαχειρίζεστε και να παραδίδετε περιεχόμενο μέσω ψηφιακών καναλιών… Ένα CMS κάνει τη διαδικασία δημιουργίας, επεξεργασίας και δημοσίευσης περιεχομένου πιο αποτελεσματική και επιτρέπει στους επαγγελματίες του μάρκετινγκ να έχουν περισσότερο έλεγχο. Το καλύτερο CMS δεν παρέχει απλώς περιεχόμενο, αλλά διαθέτει επίσης δυνατότητες διαχείρισης ροής εργασιών, δυνατότητα εύκολης αποθήκευσης ή ανάκτησης στοιχείων, δυνατότητα εύκολης ενσωμάτωσης με άλλα συστήματα και δυνατότητα παροχής εξατομικευμένων εμπειριών. Παραδοσιακά, οι πλατφόρμες CMS δημιουργήθηκαν για να προσφέρουν εμπειρίες ιστού που βασίζονται σε HTML, αλλά εξελίσσονται για να προσαρμόζονται στον πολλαπλασιασμό των καναλιών. Οι οργανισμοί μπορούν να επιλέξουν ένα δωρεάν σύστημα CMS ανοιχτού κώδικα όπως το Wordpress, το Joomla ή το Drupal ή να επενδύσουν σε λογισμικό επί πληρωμή.»

("A content management system is a software application or a set of tools and capabilities, that allows you to create, manage and deliver content via digital channels … A CMS makes the process of creating, editing and publishing content more efficient and allows marketers to have more control. The best CMS doesn't just deliver content, but also has workflow management capabilities, the ability to easily store or retrieve assets, the ability to easily integrate with other systems and the ability to deliver personalised experiences. Traditionally, CMS platforms were created to deliver HTML-based web experiences, but they are evolving to adapt to the proliferation of channels. Organisations can choose a free open-source CMS system like Wordpress, Joomla or Drupal or invest in paid software").

(What is a content management system? | Adobe Glossary)

Η πρόοδος σε τεχνολογίες όπως η τεχνολογία επεκτασιμότητας του υλικού (scalability), η τεχνολογία εικονικοποίησης (virtualization), η τεχνολογία δημιουργίας ολοκληρωμένων συστημάτων υλικού και λογισμικού (είναι γνωστά ως appliances) έδωσαν μεγάλη ώθηση και στις τεχνολογία των αποθηκών δεδομένων και των συστημάτων διαχείρισης περιεχομένου (content management systems).

2.4.2 Σημασιολογικός ιστός

Ο σημασιολογικός ιστός, η οργάνωση των δεδομένων, της πληροφορίας και της γνώσης −στο πλαίσιο αντίστοιχα της διαχείρισης δεδομένων (Database Management) της διαχείρισης πληροφορίας (Information Management) και της διαχείρισης γνώσης (Knowledge Management), η ανάκτηση στοιχείων από πολυμεσικές βάσεις δεδομένων, η εξόρυξη δεδομένων, τα δεδομένα μεγάλης κλίμακας (big data) κ.λπ. έχουν ένα κεντρικό σημείο:

Ένα λογισμικό εφαρμογής σημασιολογικού ιστού (Semantic Web) πρέπει να εστιάζει στο πώς η σημασιολογία της πληροφορίας μπορεί να είναι κατανοητή από άνθρωπο και Η/Υ και να περιλαμβάνει και να αξιοποιεί σχετικές τεχνολογίες.

Στο πλαίσιο του σημασιολογικού ιστού οι εφαρμογές μας διαχειρίζονται δεδομένα, πληροφορία και γνώση λαμβάνοντας υπόψη το περιεχόμενο και την οργάνωσή του. Τεχνολογίες σχετικές με τον σημασιολογικό ιστό είναι οι γλώσσες σήμανσης, π.χ. XML, η οικογένεια προδιαγραφών του W3C Resource Description Framework (RDF), για την εννοιολογική περιγραφή πόρων του διαδικτύου (web resources), οι οντολογίες και τα εργαλεία συγγραφής τους, π.χ. Web Ontology Language (OWL), οικογένεια γλωσσών (knowledge representation languages) για συγγραφή (authoring) οντολογιών.

Λέξεις-κλειδιά σχετικές με τις τεχνολογίες (και έννοιες) του σημασιολογικού ιστού είναι: XML, RDF, RDFS, Dublin Core, JSON, Controlled vocabularies, οντολογίες κ.λπ. Επισημαίνουμε ότι κάποιες από αυτές τις τεχνολογίες είναι γνωστές και δημοφιλείς από τη δεκαετία του 1990.

Σε ξεχωριστή ενότητα, την ενότητα 9.2, θα αναφερθούμε εκτενέστερα στις τεχνολογίες αυτές.

2.5 Αρχιτεκτονική διαχείρισης μεγάλων δεδομένων

Η σχεδίαση της αρχιτεκτονικής διαχείρισης μεγάλων δεδομένων πρέπει να λάβει υπόψη μια σειρά από παράγοντες:

- Κατανόηση των δεδομένων και των αναγκών μας. Πρέπει να καταγράψουμε:
  a. ποια δεδομένα μας ενδιαφέρουν, ποιος είναι ο σημερινός όγκος των δεδομένων και ποια είναι η πρόβλεψή μας για το μέλλον.
  b. πόσο συχνά χρειάζεται ο οργανισμός σήμερα τη διαχείριση δεδομένων σε πραγματικό χρόνο ή σχεδόν σε πραγματικό χρόνο, και ποια είναι η πρόβλεψή μας για το μέλλον.
  c. πόσο σημαντική είναι η απαιτούμενη ταχύτητα διαχείρισης δεδομένων
  d. πόσο συγκεκριμένα-ακριβή πρέπει να είναι τα δεδομένα για να μπορέσουμε να τα διαχειριστούμε.
- Εξωτερικό περιβάλλον. Πρέπει να καταγράψουμε:
  e. αν υπάρχουν κίνδυνοι από τον ανταγωνισμό και πόσο μπορεί να αντέξει ο οργανισμός μου
  f. αν υπόκειται ο κλάδος μας σε αυστηρές απαιτήσεις ασφαλείας, συμμόρφωσης (compliance) και κυβερνητικές ρυθμίσεις (governance requirements)

2.5.1 Κύκλος ζωής Διαχείρισης Μεγάλων Δεδομένων

Στην ενότητα αυτή θα παρατεθούν παραδείγματα του κύκλου ζωής και της διαχείρισης μεγάλων δεδομένων.

Παράδειγμα κύκλου ζωής της διαχείρισης μεγάλων δεδομένων

Στην εικόνα 2.11 (Pouchard, 2015) βλέπουμε ένα παράδειγμα κύκλου ζωής της διαχείρισης μεγάλων δεδομένων σε ένα πανεπιστήμιο/ερευνητικό κέντρο. Οι δραστηριότητες Describe & Assure εφαρμόζονται σε όλα τα βήματα. Η δραστηριότητα Describe αναλαμβάνει και διεκπεραιώνει την περιγραφή των δεδομένων και των διαδικασιών. Η δραστηριότητα Assure αναλαμβάνει όλες τις απαιτούμενες ενέργειες διασφάλισης της ποιότητας των δεδομένων. Τα εσωτερικά γρανάζια περιγράφουν τις τρεις αναγκαίες υποδομές:

- Το εταιρικό αποθετήριο IR (Institutional Repository), είναι ένα θεσμοθετημένο διαδικτυακό αρχείο (archive) για τη συλλογή, τη διατήρηση και τη διάδοση ψηφιακών αντιγράφων της πνευματικής παραγωγής ενός ιδρύματος, π.χ., της ερευνητικής δραστηριότητας ενός πανεπιστημίου, ενός ερευνητικού ιδρύματος.
- Το θεματικό αποθετήριο DR (Disciplinary Repository), είναι ένα θεσμοθετημένο διαδικτυακό αρχείο (archive) το οποίο περιλαμβάνει έργα ή δεδομένα σχετιζόμενα με τα έργα, των ερευνητών-μελετητών για μια συγκεκριμένη θεματική περιοχή (subject area).

Βλέπε σχετικά,

Rob Kling (2003) The Internet and Unrefereed Scholarly Publishing, CSI Working paper,

No. WP- 03-01 The Internet and Unrefereed Scholarly Publishing (iu.edu)

- HPC (High Performance Computing data center) αναφέρεται σε ένα κέντρο δεδομένων το οποίο εξασφαλίζει τη δυνατότητα υψηλής απόδοσης υπολογισμού (High Performance Computing), δηλαδή περιλαμβάνει κατάλληλες υποδομές, υπολογιστών και δικτύων, και στοχεύει στην εκτέλεση μεγάλου όγκου υπολογισμών σε σύντομο χρονικό διάστημα.

Εικόνα 2.11 Παράδειγμα κύκλου ζωής της διαχείρισης μεγάλων δεδομένων σε ένα πανεπιστήμιο/ερευνητικό κέντρο

Ακολουθεί περιγραφή των φάσεων του κύκλου ζωής της διαχείρισης μεγάλων δεδομένων η οποία παρουσιάζεται στο άρθρο:

Pouchard L. (2015) Revising the Data Lifecycle with Big Data Curation. International Journal of Digital Curation, 10(2), 176-192.

https://www.researchgate.net/publication/305095078_Revisiting_the_Data_Lifecycle_with_Big_Data_Curation

Σχεδιασμός (planning)

Απαιτείται η προσεκτική επιλογή των δεδομένων που θα πρέπει να διατηρηθούν στο σύστημα διαχείρισης λόγω του μεγάλου όγκου των δεδομένων. Σε κάποιες περιπτώσεις πρέπει να διατηρηθούν όλα τα δεδομένα λόγω της κρισιμότητας τους, ανεξάρτητα από το κόστος. Στην περίπτωση διαχείρισης σε πραγματικό χρόνο (online data) πρέπει να εξεταστεί η περίπτωση να κρατήσουμε δείγμα δεδομένων. Σε κάποιες περιπτώσεις μπορούμε να επιλέξουμε να κρατάμε μόνο τα αποτελέσματα της ανάλυσης δεδομένων ή όποια δεδομένα είναι χρήσιμα για να εκτελούμε («τρέχουμε») μια προσομοίωση.

Απόκτηση (acquire)

Εξετάζουμε και επιλέγουμε τρόπους τροφοδότησης του συστήματος που μας ενδιαφέρει με δεδομένα. Εξετάζουμε πως παράγονται-δημιουργούνται τα δεδομένα, π.χ., λαμβάνονται από αισθητήρες, εξωτερικές πηγές κ.λπ.

Προετοιμασία (prepare)

Μορφοποίηση και καθαρισμός των δεδομένων ώστε να μπορούν να χρησιμοποιηθούν στη συνέχεια για ανάλυση και οπτικοποίηση

Ανάλυση (analysis)

Εφαρμογή μεθόδων ανάλυσης. Προαπαιτούμενο είναι, η λεπτομερής καταγραφή και η διατήρηση των παραμέτρων των πειραμάτων μας, των σεναρίων προσομοίωσης και ολόκληρου του υπολογιστικού περιβάλλοντος, για έχουμε τη δυνατότητα να ξανατρέξουμε το μοντέλο και να αναπαράγουμε τα αποτελέσματα.

Διατήρηση (preserve)

Εξετάζουμε και καθορίζουμε πως θα διατηρηθούν τα αποτελέσματα για μακροχρόνια χρήση. Δυστυχώς, η διατήρηση των διεργασιών είναι πιο δύσκολο να επιτευχθεί επειδή οι διαδικασίες και οι είσοδοι/έξοδοι αλλάζουν γρήγορα (σε ημέρες ή ώρες)

Ανακάλυψη (discover)

Αναφέρεται στο σύνολο των διαδικασιών που διασφαλίζει ότι τα σύνολα δεδομένων μπορούν να εντοπιστούν και αξιοποιηθούν από τρίτους (ανοιχτά δεδομένα). Θα πρέπει τα αποθετήρια να επιτρέπουν την κοινή χρήση δεδομένων και να βασίζονται σε πρότυπα δεδομένων και μεταδεδομένων, σε ενοποιημένες-ολοκληρωμλενες αναζητήσεις, εργαλεία λογισμικού και αναγνωριστικά που διευκολύνουν την ανακάλυψη δεδομένων

Παράδειγμα εφαρμογής δεδομένων μεγάλης κλίμακας σε βιώσιμες έξυπνες πόλεις

Στην εικόνα 2.12. προτείνεται ένα πλαίσιο ανάλυσης για τις βιώσιμες έξυπνες πόλεις. Το πλαίσιο αναλύεται στο άρθρο,

Bibri , S. E. (2018). The IoT for smart sustainable cities of the future: An analytical framework, Sustainable Cities and Society, 38, 230-25.3

Στην εικόνα 2.12 στο πλαίσιο της βιώσιμης έξυπνης πόλη βλέπουμε τις παρακάτω εφαρμογές:

- Smart Build Environment (έξυπνα κτίρια): Αναλαμβάνει παρακολούθηση και έλεγχο των μηχανικών, ηλεκτρικών και ηλεκτρονικών συστημάτων σε κτίρια δημόσια, βιομηχανικά, εμπορικά
- Smart Infrastructure Monitoring: Αναλαμβάνει παρακολούθηση και έλεγχο υποδομών, π.χ. , γέφυρες, σιδηροδρομικές γραμμές, τούνελ
- Smart Transport & Mobility: Αναλαμβάνει παρακολούθηση των οχημάτων, της κυκλοφορίας, διόδια, εντοπισμός θέσεων παρκινγκ, κλπ,
- Smart Traffic: Αναλαμβάνει ρύθμιση φωτεινών σηματοδοτών
- Smart Energy: Οι εταιρείες παροχής ηλεκτρικής ενέργειας ρυθμίζουν την παροχή ενέργειας στις περιοχές και στις ώρες όπου υπάρχει αυξημένη ανάγκη – αν όλες οι συσκευές στέλνουν δεδομένα– Επίσης οι ιδιοκτήτες-πελάτες των εταιρειών μπορούν να έχουν απομακρυσμένο έλεγχο

Ως υποδομή διαχείρισης προτείνεται Cloud/Fog Hadoop MapReduce. Αυτή η συνιστώσα σχετίζεται με τη διαδικασία ανακάλυψη γνώσης και την εξόρυξη δεδομένων.

Το μοντέλο **Fog Computing** είναι εναλλακτικό μοντέλο του υπολογιστικού νέφους (cloud computing) κατάλληλο για IoT και ανάλυση μεγάλων δεδομένων (big data analytics). Παρουσιάζει μεγαλύτερη γεωγραφική κατανομή και μεγαλύτερη εγγύτητα στους τελικούς χρήστες. Η επεξεργασία και διαχείριση δεδομένων εκτελείται από Hadoop MapReduce και προκύπτει η ανακάλυψη γνώσης.

Υπάρχουν διάφοροι τρόποι-τύποι ανάλυσης γνώσης (Data analytics) ανάλογα με το ερώτημα-πρόβλημα που ενδιαφέρει:

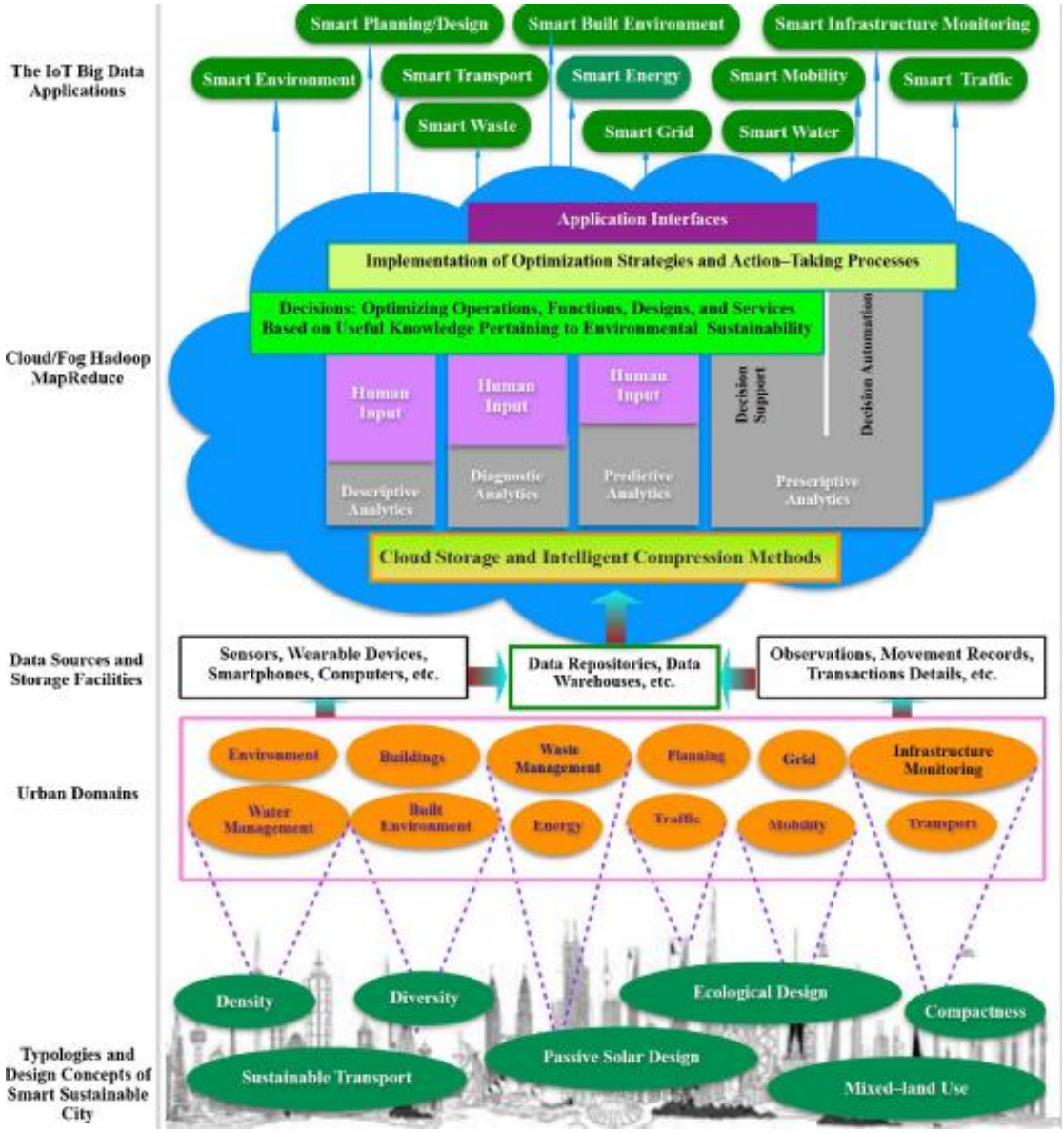
- Descriptive, “Τι έχει συμβεί;”

- Diagnostic, “Γιατί έχει συμβεί;”
- Predictive, “Τι είναι πιθανό να συμβεί στο μέλλον;”
- Prescriptive, “Τι θα πρέπει να κάνουμε στη συνέχεια;

Η συνιστώσα **Big Data sources/storage facilities & data categories** σχετίζεται με τη συλλογή την αποθήκευση και τη διαχείριση δεδομένων

Οι αστικοί τομείς (urban domains) περιλαμβάνουν: περιβάλλον, κτήρια, διαχείριση αποβλήτων, αστικός σχεδιασμός, δίκτυα (grid), παρακολούθηση υποδομών, διαχείριση υδάτων, ενέργεια, κίνηση στους δρόμους, μεταφορές

Τυπολογίες και Σχέδια για να επιτύχουμε το στόχο της βιώσιμης πόλης (Typologies and design concepts of smart sustainable cities), π.χ., βιώσιμες μεταφορές, οικολογικός σχεδιασμός, μεικτή χρήση γης, κ.λπ.



Εικόνα 2.12 Πλαίσιο για βιώσιμες έξυπνες πόλεις (Bibri, 2018)

2.5.2 Αρχιτεκτονικές μεγάλων δεδομένων (Big data architectures) και προμηθευτές ΣΔΒΔ

Παραθέτουμε έναν ορισμό των μεγάλων δεδομένων σύμφωνα με την Oracle.

«Τα μεγάλα δεδομένα είναι πέραν των συνηθισμένων δεδομένων γιατί συγκρινόμενα με αυτά περιλαμβάνουν μεγαλύτερη ποικιλία δεδομένων, φθάνουν σε αυξανόμενους όγκους και με μεγαλύτερη ταχύτητα. Τα γνωστά τρία V (variety, volume, velocity). Με απλά λόγια, τα μεγάλα δεδομένα είναι μεγαλύτερα, πιο σύνθετα σύνολα δεδομένων, και ειδικότερα προέρχονται από νέες πηγές δεδομένων. Αυτά τα σύνολα δεδομένων είναι τόσο ογκώδη που το παραδοσιακό λογισμικό επεξεργασίας δεδομένων απλά δεν μπορεί να τα διαχειριστεί. Αλλά αυτοί οι τεράστιοι όγκοι δεδομένων μπορούν να χρησιμοποιηθούν για την αντιμετώπιση επιχειρηματικών προβλημάτων που δεν θα μπορούσατε να αντιμετωπίσετε πριν»

("The definition of big data is data that contains greater variety, arriving in increasing volumes and with more velocity. This is also known as the three Vs. Put simply, big data is larger, more complex data sets, especially from new data sources. These data sets are so voluminous that traditional data processing software just can't manage them. But these massive volumes of data can be used to address business problems you wouldn't have been able to tackle before"). What Is Big Data? | Oracle

Η αρχιτεκτονική δεδομένων είναι ένα πλαίσιο (framework) περιγραφής της υποδομής πληροφορικής μιας επιχείρησης για τη διαχείριση των δεδομένων της. Αποτελεί το θεμέλιο για την υποστήριξη της στρατηγικής της επιχείρησης αναφορικά με τα δεδομένα της.

Η αρχιτεκτονική δεδομένων παρουσιάζει την υποδομή της εταιρείας, και εξειδικεύεται στην περιγραφή συνιστωσών της αρχιτεκτονικής όπως οι συνιστώσες συλλογής/απόκτησης, μεταφοράς, αποθήκευσης, αναζήτησης και διασφάλισης της ασφάλειας των δεδομένων. Δεν μπορούμε να προχωρήσουμε σε οποιασδήποτε στρατηγική δεδομένων χωρίς την αρχιτεκτονική δεδομένων. Το προφανές συμπέρασμα είναι ότι θα πρέπει στις εφαρμογές της διαχείρισης των μεγάλων δεδομένων να έχουμε ως αφετηρία μία νέα αρχιτεκτονική δεδομένων.

Σύμφωνα με την Microsoft,

«*Μια αρχιτεκτονική μεγάλων δεδομένων σχεδιάζεται για να διαχειρίζεται την απορρόφηση, την επεξεργασία και την ανάλυση δεδομένων τα οποία είναι πολύ μεγάλα ή πολύπλοκα για να διαχειριστούμε με τα παραδοσιακά συστήματα βάσεων δεδομένων. Το όριο στο οποίο οι οργανισμοί εισέρχονται στη σφαίρα των μεγάλων δεδομένων διαφέρει, ανάλογα με τις δυνατότητες των χρηστών και των εργαλείων τους. Για κάποιους, μπορεί να σημαίνει εκατοντάδες gigabyte δεδομένων, ενώ για άλλους σημαίνει εκατοντάδες terabyte. Όσο τα εργαλεία διαχείρισης μεγάλων συνόλων δεδομένων εξελίσσονται τόσο εξελίσσεται και η έννοια των μεγάλων δεδομένων. Όλο και περισσότερο, ο όρος συσχετίζεται περισσότερο με την αξία που μπορούμε να εξαγάγουμε από τα σύνολα δεδομένων, με χρήση μεθόδων ανάλυσης, και λιγότερο με το μέγεθος των δεδομένων, αν και πάντοτε αναφερόμαστε σε δεδομένα τα οποία τείνουν να είναι αρκετά μεγάλα….*»

("A big data architecture is designed to handle the ingestion, processing, and analysis of data that is too large or complex for traditional database systems. The threshold at which organizations enter into the big data realm differs, depending on the capabilities of the users and their tools. For some, it can mean hundreds of gigabytes of data, while for others it means hundreds of terabytes. As tools for working with big datasets advance, so does the meaning of big data. More and more, this term relates to the value you can extract from your data sets through advanced analytics, rather than strictly the size of the data, although in these cases they tend to be quite large …. ")

Big data architectures - Azure Architecture Center | Microsoft Docs

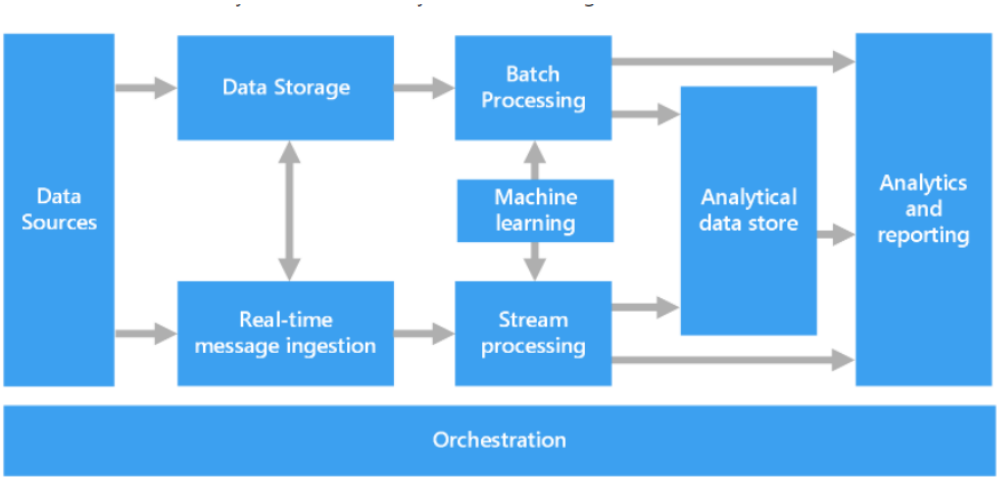
Παραθέτουμε, στη συνέχεια, κάποιες συστάσεις και παρουσιάζουμε σύντομα μία προτεινόμενη, από τη Microsoft, αρχιτεκτονική μεγάλων δεδομένων γενικού σκοπού (εικόνα 2.13) και τις συνιστώσες (Components) της.

Συστάσεις

Πρέπει να εξετάζουμε αρχιτεκτονικές μεγάλων δεδομένων όταν χρειάζεται:

- Αποθήκευση και επεξεργασία δεδομένων τα οποία χαρακτηρίζονται από μεγάλο όγκο για να τα διαχειριστούμε με μια παραδοσιακή βάση δεδομένων.
- Μετασχηματισμός μη δομημένων δεδομένων (unstructured data) για ανάλυση (analysis) και αναφορά (reporting).
- Λήψη, επεξεργασία και ανάλυση απεριόριστων ροών δεδομένων (unbounded streams of data) σε πραγματικό χρόνο ή όταν απαιτείται όσο γίνεται πιο μικρή καθυστέρηση στη διαχείριση τους (with low latency).

Στην εικόνα 2.13 το διάγραμμα αποτυπώνει τα λογικά στοιχεία (logical components) τα οποία θα «ταίριαζαν» σε μια αρχιτεκτονική μεγάλων δεδομένων. Δεν είναι απαραίτητο, κάθε εφαρμογή να περιλαμβάνει όλα τα στοιχεία του διαγράμματος.



Εικόνα 2.13 Στοιχεία αρχιτεκτονικής μεγάλων δεδομένων» (Microsoft).

Πριν προχωρήσουμε στη συζήτηση των συνιστωσών της αρχιτεκτονικής της εικόνας 2.13 πρέπει να γίνει μνεία για την υπηρεσία Azure HDInsight. Η υπηρεσία αυτή είναι μια ανοιχτού κώδικα υπηρεσία ανάλυσης στοιχείων (open-source analytics service) στο νέφος (cloud) για επιχειρήσεις η οποία επιτρέπει τη χρησιμοποίηση πλαισίων ανοικτού κώδικα (open-source frameworks) όπως Hadoop, Apache Spark, Apache Hive, LLAP, Apache Kafka, Apache Storm, R κ.λπ. σε Azure environment.

Πηγές δεδομένων (Data sources)

Σε όλες τις εφαρμογές μεγάλων δεδομένων εκκινούμε από μία ή περισσότερες πηγές, π.χ., σχεσιακές βάσεις δεδομένων, στατικά δεδομένα με τη μορφή αρχείων καταγραφής διακομιστή ιστού (web server log files), δεδομένα πραγματικού χρόνου από συσκευές IoT (IoT devices).

Αποθήκευση δεδομένων (Data storage)

Για να διαχειριστούμε μεγάλα δεδομένα, μια συχνά χρονοβόρα διαδικασία, απαιτείται η αποθήκευση των δεδομένων τα οποία χαρακτηρίζονται από μεγάλο όγκο και μεγάλη ποικιλία μορφοτύπων (formats) σε ένα κατανεμημένο χώρο αποθήκευσης αρχείων (distributed file store) επαρκούς χωρητικότητας. Η Microsoft προτείνει για την αποθήκευση τη χρήση «λίμνης» δεδομένων (data lake), έννοια η οποία αποτελεί εξέλιξη της «παραδοσιακής» αποθήκης δεδομένων. Επιλογές για την υλοποίηση αυτής της αποθήκευσης, οι οποίες προτείνονται από τη Microsoft, περιλαμβάνουν το Azure Data Lake Store ή τα κοντέινερ blob στο Azure Storage (blob containers in Azure Storage).

Ομαδική-μαζική επεξεργασία (batch processing)

Επειδή τα σύνολα δεδομένων είναι πολύ μεγάλα εφαρμόζουμε συχνά χρονοβόρες διαδικασίες ομαδικής-μαζικής επεξεργασίας (batch processing) για το φιλτράρισμά τους (filtering), την ομαδοποίησή τους (aggregation) και γενικότερα την προετοιμασία τους (data preparation) για ανάλυση. Συνήθως αυτές οι εργασίες (batch jobs) περιλαμβάνουν την ανάγνωση αρχείων πηγής, την επεξεργασία τους και την εγγραφή της εξόδου σε νέα αρχεία.

Επιλογές για την υλοποίηση αυτών των εργασιών (jobs), οι οποίες προτείνονται από τη Microsoft, περιλαμβάνουν:

- την εκτέλεση U-SQL jobs στο Azure Data Lake Analytics,
- τη χρήση Hive, Pig ή προσαρμοσμένων εργασιών (custom jobs) Map/Reduce σε HDInsight Hadoop cluster
- τη χρήση προγραμμάτων Java, Scala ή Python σε HDInsight Spark cluster.

Σε επόμενη ενότητα του κεφαλαίου αυτού θα αναφερθούμε στις έννοιες Map/Reduce, Hadoop.

«Απορρόφηση» (αξιοποίηση) μηνυμάτων σε πραγματικό χρόνο (Real-time message ingestion)

Εάν αντλούμε στοιχεία από πηγές σε πραγματικό χρόνο, η αρχιτεκτονική μας πρέπει να περιλαμβάνει έναν τρόπο λήψης και αποθήκευσης «μηνυμάτων σε πραγματικό χρόνο για επεξεργασία ροής» (capture and store of real-time messages for stream processing). Απαιτείται μια «αρχιτεκτονική ροής» (a streaming architecture) η οποία υποστηρίζει ένα είδος «ενδιάμεσης αποθήκευσης της ροής» (stream buffering). Επομένως, σε πολλές περιπτώσεις απαιτείται «χώρος απορρόφησης -προσωρινής αποθήκευσης- μηνυμάτων» (a message ingestion store) για την κλιμάκωση της επεξεργασίας(scale-out processing), την αξιόπιστη παράδοση (delivery) κ.λπ. Αυτό το τμήμα μιας αρχιτεκτονικής ροής αναφέρεται συχνά ως buffering ροής. Οι προτεινόμενες λύσεις από τη Microsoft περιλαμβάνουν Azure Event Hub, Azure IoT Hub και Kafka.

Επεξεργασία ροής (Stream processing)

Μετά τη λήψη μηνυμάτων σε πραγματικό χρόνο ακολουθεί η προετοιμασία τους (data preparation) για ανάλυση και τα επεξεργασμένα δεδομένα ροής (stream data) εγγράφονται σε μία «δεξαμενή εξόδου» (output sink). Η Microsoft προτείνει χρήση της υπηρεσίας Azure Stream Analytics, μια υπηρεσία επεξεργασίας και διαχείρισης ροής η οποία βασίζεται σε ερωτήματα SQL. Μπορούμε να χρησιμοποιήσουμε ανοικτού κώδικα τεχνολογίες ροής Apache (open source Apache streaming technologies), π.χ., Storm, Spark Streaming σε HDInsight cluster.

Αποθήκη αναλυτικών δεδομένων (Analytical data store)

Οι εφαρμογές μεγάλων δεδομένων προετοιμάζουν τα δεδομένα για την ανάλυση και αποθηκεύουν τα επεξεργασμένα δεδομένα σε μια δομημένη μορφή. Η μορφή αυτή πρέπει να επιτρέπει τη χρήση τους από αναλυτικά εργαλεία για την υποβολή ερωτημάτων.

Έχουμε διάφορες επιλογές για την αποθήκευσης και παρουσίαση των αναλυτικών δεδομένων:

1. μια αποθήκη σχεσιακών δεδομένων, λύση συνδεδεμένη με τις περισσότερες παραδοσιακές λύσεις επιχειρηματικής ευφυΐας (BI).
2. χρήση τεχνολογίας NoSQL με μικρή καθυστέρηση στην παρουσίαση αποτελεσμάτων (a low-latency), π.χ. HBase,
3. Η υπηρεσία Azure Synapse Analytics διαχειρίζεται μεγάλης κλίμακας αποθήκευση δεδομένων σε περιβάλλον νέφους (large-scale, cloud-based data warehousing)

Ανάλυση και αναφορά (Analysis and reporting)

Ο στόχος των περισσότερων εφαρμογών μεγάλων δεδομένων είναι να παρέχουν πληροφορίες για τα δεδομένα μέσω ανάλυσης και αναφοράς. Επομένως, η αρχιτεκτονική μπορεί να περιλαμβάνει και να υποστηρίζει ενδεικτικά:

- πολυδιάστατο κύβο OLAP (multidimensional OLAP cube)
- BI αυτοεξυπηρέτησης (self-service BI), μοντελοποίηση και οπτικοποίηση
- διαδραστική εξερεύνησης δεδομένων (interactive data exploration) από επιστήμονες δεδομένων ή αναλυτές δεδομένων.
- αναλυτικά σημειωματάρια (analytical notebooks), π.χ., Jupyter, και δυνατότητες χρήσης Python, R κ.λπ.

Συντονισμός λειτουργίας («ενορχήστρωση», Orchestration)

Οι περισσότερες εφαρμογές μεγάλων δεδομένων αποτελούνται από επαναλαμβανόμενες λειτουργίες επεξεργασίας δεδομένων, ενσωματωμένες σε ροές εργασίας (encapsulated in workflows), οι οποίες μετασχηματίζουν δεδομένα πηγών, μετακινούν τα δεδομένα μεταξύ πολλαπλών πηγών και «δεξαμενών δεδομένων εξόδου» (sinks), και αποθηκεύουν τα επεξεργασμένα δεδομένα σε έναν χώρο αναλυτικών δεδομένων (analytical data store) ή τα προωθούν για αναφορά (to a report) ή σε πίνακα εργαλείων (dashboard). Για να αυτοματοποιήσουμε αυτές τις ροές εργασίας χρειαζόμαστε τεχνολογία «ενορχήστρωσης». Για παράδειγμα, η Microsoft προτείνει χρήση Azure Data Factory ή Apache Oozie και Sqoop.

2.5.3 Αρχιτεκτονική μιας τυπικής τεχνολογικής στοίβας (stack) για μεγάλα δεδομένα

Σε μία τυπική στοίβα τεχνολογίας μεγάλων δεδομένων (Big Data Technology Stack) θα μπορούσαμε να αντιστοιχίσουμε μια αρχιτεκτονική πολλών επιπέδων-τεχνολογιών:

- διεπαφές και τροφοδότηση (Interfaces and feeds)
- ανάλυση δεδομένων (data analytics)
- μοντελοποίηση δεδομένων (data modelling)
- αποθήκευση δεδομένων (data warehousing)
- διοχέτευση δεδομένων (data pipeline)

Τα επίπεδα είναι αλληλεξαρτώμενα και κάθε επίπεδο διαδραματίζει έναν ρόλο απαραίτητο για την ομαλή λειτουργία.

2.5.3.1 Διεπαφές και τροφοδότηση (Interfaces and feeds)

Επειδή τα μεγάλα δεδομένα είναι συλλογές ενοποιημένων δεδομένων (data integration), από πολλά δεδομένα πολλών πηγών, οι ανοιχτές διεπαφές προγραμματισμού εφαρμογών API (open application programming interfaces) είναι σημαντικές, υπάρχουν εντός κάθε επιπέδου και μεταξύ των επιπέδων της τεχνολογικής στοίβας και διαδραματίζουν σπουδαίο ρόλο στην ενοποίηση των δεδομένων παρέχοντας πρόσβαση σε και από εφαρμογές λογισμικού. Επιπλέον, οι διεπαφές API διασφαλίζουν ότι η δημιουργία νέων εφαρμογών διευκολύνεται ή και απλουστεύεται.

Ειδική μνεία πρέπει να γίνει για την τεχνολογία REST. Επισημαίνουμε ότι και οι τεχνολογίες μεγάλων δεδομένων υποστηρίζουν την τεχνολογία REST (Representational State Transfer). Ένα RESTful API παρέχει έναν τυποποιημένο τρόπο δημιουργίας μιας προσωρινής σχέσης (ονομάζεται και χαλαρή σύζευξη, loose coupling) μεταξύ των πόρων του Παγκόσμιου Ιστού. Η χρήση της τεχνολογίας REST σημαίνει ότι ένα αίτημά σας στις εφαρμογές μεγάλων δεδομένων θα περιμένει και θα απαντηθεί μόνο όταν η υπηρεσία είναι διαθέσιμη.

Τα μεγάλα δεδομένα μπορεί να απαιτούν και εξειδικευμένη προσέγγιση στην ανάπτυξη ή την υιοθέτηση API.

Η διεπαφή API (Application Programming Interface) είναι ένα ενδιάμεσο λογισμικό (software intermediary) το οποίο επιτρέπει σε δύο εφαρμογές να συνομιλούν μεταξύ τους. Οι διεπαφές ακολουθούν καθορισμένα πρωτόκολλα, δηλαδή σύνολα κανόνων τα οποία ορίζουν τον τρόπο «συνομιλίας» μεταξύ υπολογιστών, εφαρμογών λογισμικού. Οι διεπαφές επιτρέπουν στους προγραμματιστές (developers) να δημιουργούν, να συνδέονται και να ενσωματώνουν (integrate) εφαρμογές γρήγορα, απλά και σε κλίμακα (at scale) και βέβαια χρησιμοποιούνται εδώ και χρόνια για να διευκολύνουν την εργασία τους και σε πλήθος εφαρμογών στον παγκόσμιο ιστό, στα λειτουργικά συστήματα, στα συστήματα βάσεων δεδομένων κ.λπ.

Θα μπορούσαμε να κατανοήσουμε καλύτερα την έννοια της διεπαφής με ένα απλό παράδειγμα. Έχετε εγγραφεί σε κάποια διαδικτυακή υπηρεσία και συνδέεστε στη διαδικτυακή εφαρμογή, η οποία υλοποιεί τη συγκεκριμένη υπηρεσία. Τα στοιχεία σας υπάρχουν σε κάποιους εξυπηρετητές. Η σύνδεσή σας γίνεται με τον υπολογιστή σας ή με το κινητό σας και η διαδικτυακή εφαρμογή είναι εξειδικευμένη για κάθε ένα από το μέσο με το οποίο συνδέεστε. Ας υποθέσουμε ότι συνδέστε με το κινητό σας. Ουσιαστικά ενημερώνετε τη διαδικτυακή εφαρμογή ότι θέλετε να αποκτήσετε πρόσβαση στον λογαριασμό σας στην υπηρεσία. Η εφαρμογή πραγματοποιεί κλήση σε ένα API για να ανακτήσει τον λογαριασμό (account) και τα διαπιστευτήριά σας (credentials) στην υπηρεσία. Στη συνέχεια, η εφαρμογή θα έχει πρόσβαση στα στοιχεία από έναν από τους διακομιστές και επιστρέφει τα δεδομένα στην εφαρμογή για κινητά.

Πάρα πολλά δεδομένα δεν είναι δομημένα και παράγονται εκτός της επιχείρησης. Για παράδειγμα, στα κοινωνικά δίκτυα έχουμε πάρα πολλά κείμενα σε φυσική γλώσσα. Επομένως, η επεξεργασία φυσικής γλώσσας (NLP) αναδύεται ως ενδιαφέρουσα μέθοδος «διασύνδεσης» των μεγάλων δεδομένων με τα προγράμματα εφαρμογής. Η τεχνική NLP επιτρέπει τη διατύπωση ερωτημάτων σε φυσική γλώσσα. Έτσι ένα απλό ερώτημα σε φυσική γλώσσα είναι κατά πολύ απλούστερο ενός ερωτήματος σε γλώσσα ερωτημάτων,, π.χ. για το αντίστοιχο ερώτημα σε γλώσσα SQL μπορεί να απαιτηθεί να γράψουμε 50 γραμμές κώδικα.

The role of the API in managing Big Data | DreamFactory Software- Blog

Ένας τρόπος για να εργαστείτε με τις διεπαφές είναι να χρησιμοποιήσετε τεχνικές αρχιτεκτονικής με προσανατολισμό στις υπηρεσίες (Service Oriented Architecture-SOA), π.χ., το εργοστάσιο "σύνδεσης" (“connector” factory), το οποίο προσθέτει ένα επίπεδο αφαίρεσης και προβλεψιμότητας (layer of abstraction and predictability) στη διαδικασία.

Copy and transform data from and to a REST endpoint by using Azure Data Factory - Azure Data Factory & Azure Synapse | Microsoft Docs

https://docs.microsoft.com/en-us/azure/data-factory/quickstart-create-data-factory-rest-api

Για μία εισαγωγή για αρχάριους σχετικά με την αρχιτεκτονική SOA ανατρέξτε στο:

Judith S. Hurwitz, Robin Bloor, Marcia Kaufman, Fern Halper (2009) Service Oriented Architecture (SOA) For Dummies, 2nd Edition, John Wiley & Sons, Inc.

2.5.3.2 Η φυσική υποδομή υποστήριξης της ασφαλούς διαχείρισης μεγάλων δεδομένων (Redundant physical secure infrastructure) από διαφορετικές πήγες και με πλεονασμούς

Η υποστήριξη του όγκου μεγάλων δεδομένων απαιτεί φυσική υποδομή η οποία βασίζεται σε ένα κατανεμημένο υπολογιστικό μοντέλο (distributed computing model). Τα δεδομένα αποθηκεύονται με πλεονασμό (Redundancy) σε πολλές τοποθεσίες και συνδέονται μεταξύ τους μέσω δικτύων. Ο πλεονασμός στην αποθήκευση των μεγάλων δεδομένων είναι σημαντικός-απαραίτητος επειδή θέλουμε συνεχή διαθεσιμότητα και γρήγορη επεξεργασία πολλών δεδομένων από πολλές διαφορετικές πηγές.

Ως υποδομή μπορούμε να θεωρήσουμε ένα δίκτυο παράδοσης περιεχομένου (Content Delivery Network-CDN) όρος που αναφέρεται σε μια γεωγραφικά κατανεμημένη ομάδα διακομιστών που συνεργάζονται για να παρέχουν γρήγορη παράδοση περιεχομένου στο Διαδίκτυο. Για παράδειγμα το Netflix εξασφαλίζει την απόδοση της προβολής των video με χρήση αυτής της υποδομής. Δείτε σχετικά το άρθρο,

A cooperative approach to content delivery-A Netflix briefing paper, 2021

Open-Connect-Briefing-Paper.pdf (netflix.com)

Μπορούμε να χρησιμοποιήσουμε ιδιωτικό νέφος (private cloud) επεκτάσιμο (κλιμακούμενο, ο αγγλικός όρος είναι scale out) ώστε να μπορεί να υποστηρίξει την αλλαγή του φόρτου εργασίας (workloads). Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε εξωτερικές υπηρεσίες νέφους (cloud). Στην περίπτωση μίσθωσης υπηρεσιών νέφους, αν η ανάλυση μεγάλων δεδομένων είναι σημαντική για την επιχείρηση μπορούμε να επιλέξουμε υπηρεσία Software as a Service (SaaS), η οποία διασφαλίζει «εξελιγμένη» ανάλυση δεδομένων (sophisticated data analysis) και έχει χαμηλότερο κόστος.

Όσο η σημασία της ανάλυσης μεγάλων δεδομένων αυξάνεται για την επιχείρηση τόσο γίνεται ακόμη πιο σημαντική η διασφάλιση-ασφάλεια (security) των δεδομένων. Σε κάποιες περιπτώσεις, π.χ., υγειονομικής περίθαλψης, είναι ιδιαίτερα κρίσιμο το ζήτημα της αποφυγής παραβίασης της ιδιωτικότητας (privacy). Γενικά, η διαχείριση πρέπει να χαρακτηρίζεται από:

- τη δυνατότητα επαλήθευσης της ταυτότητας των χρηστών
- την προστασία της ταυτότητα των πελατών, ασθενών κ.λπ.

2.5.3.3 Πηγές λειτουργικών δεδομένων (Operational data sources)

Συνήθως με τον όρο λειτουργικά δεδομένα (Operational data) της επιχείρησης αναφερόμασταν μόνο στα δομημένα δεδομένα τα οποία είναι απαραίτητα για την καθημερινή λειτουργία της επιχείρησής. Τα σχεσιακά ΣΔΒΔ (RDBMS), π.χ., Oracle, DB2, περιλαμβάνουν συνιστώσες-συστήματα OLAP και OLTP. Η συνιστώσα OLTP αναλαμβάνει τη διαχείριση των συναλλαγών και η συνιστώσα OLAP αναλαμβάνει την ανάλυση δεδομένων και την ανακάλυψη γνώσης (από ιστορικά δεδομένα) η οποία είναι χρήσιμη για επιχειρηματικές αποφάσεις.

Στην περίπτωση του όρου μεγάλα δεδομένα, είναι σημαντικό να κατανοήσουμε ότι τα επιχειρησιακά λειτουργικά δεδομένα πρέπει να περιλαμβάνουν-ενσωματώνουν ένα ευρύτερο σύνολο δομημένων και μη δομημένων πηγών δεδομένων, π.χ., «παραδοσιακά στοιχεία», δεδομένα πελατών και κοινωνικών μέσων.

Γενικά, ο στόχος των εφαρμογών μεγάλων δεδομένων είναι η διαχείρισή τους να παρέχει μια πληρέστερη εικόνα της επιχείρησης και της λειτουργίας της. Με άλλη διατύπωση, στόχος είναι να προσεγγίσουμε τη λεγόμενη «επιχειρηματική προβολή 360 μοιρών» (360-degree business view).

Υπάρχουν διάφορες αναδυόμενες προσεγγίσεις για τη διαχείριση μεγάλων δεδομένων οι οποίες περιλαμβάνουν αρχιτεκτονικές βάσεων δεδομένων εγγράφων (document database), γραφημάτων (graph database), στηλών (columnar database) και γεωχωρικών δεδομένων (geospatial database). Συλλογικά, αναφερόμαστε σε αυτές τις νέες βάσεις δεδομένων με το όρο βάσεις δεδομένων NoSQL ή βάσεις δεδομένων όχι μόνο-not only-SQL. Ουσιαστικά, πρέπει να αντιστοιχίσουμε αρχιτεκτονικές δεδομένων με τύπους συναλλαγών (types of transactions) κατά τρόπο που να διασφαλίζει ότι τα απαραίτητα δεδομένα είναι διαθέσιμα όταν τα χρειαζόμαστε. Μία αρχιτεκτονική πρέπει να υποστηρίζει και πολύπλοκο μη δομημένο περιεχόμενο (complex unstructured content) και να περιλαμβάνει:

- σχεσιακές βάσεις δεδομένων,
- μη σχεσιακές βάσεις δεδομένων,
- μη δομημένες πηγές δεδομένων, όπως συστήματα διαχείρισης περιεχομένου,

Η Gartner ορίζει την έννοια «συστήματα καταγραφής» (“system of record”) ως τα συστήματα τα οποία υποστηρίζουν την επεξεργασία βασικών συναλλαγών και διαχειρίζονται τα κρίσιμα κύρια δεδομένα του οργανισμού (“systems that support core transaction processing and manage the organization’s critical master data.”). Διακρίνουν δύο είδη συστημάτων: «πακέτο-package- από καθιερωμένες εφαρμογές ή αναπτυχθέντα εντός της επιχείρησης συστήματα παλαιού τύπου» (“established packaged applications or legacy homegrown”).

Gartner’s PACE Layered Application Strategy - CIO Wiki (cio-wiki.org)

Οι πηγές επιχειρησιακών λειτουργικών δεδομένων (operational data sources) αντιπροσωπεύουν «συστήματα καταγραφής» τα οποία παρακολουθούν (keep track) τα κρίσιμα δεδομένα που απαιτούνται για την καθημερινή λειτουργία της επιχείρησης σε πραγματικό χρόνο. Η συνεχής ενημέρωση τους βασίζεται στις συναλλαγές που πραγματοποιούνται εντός των επιχειρηματικών μονάδων (business units) και από δεδομένα από τον Παγκόσμιο Ιστό. Στοχεύοντας στην έγκυρη αναπαράσταση της επιχείρησης συνδυάζουν δομημένα και μη δομημένα δεδομένα. Παρέχουν δυνατότητα κλιμάκωσης (scale) για την υποστήριξη χιλιάδων χρηστών. Μπορεί να περιλαμβάνουν συστήματα ηλεκτρονικού εμπορίου βασιζόμενα σε συναλλαγές (transactional e-commerce systems), συστήματα διαχείρισης πελατειακών σχέσεων (customer relationship management systems), συστήματα τηλεφωνικής εξυπηρέτησης (call center applications) κ.λπ.

2.5.3.4 Οργάνωση υπηρεσιών και εργαλείων δεδομένων (Organizing data services and tools) MapReduce, Hadoop και Big Table

Δεν είναι όλα τα δεδομένα που χρησιμοποιούν οι οργανισμοί λειτουργικά δεδομένα. Για παράδειγμα, μεγάλος όγκος δεδομένων μπορεί να προέρχεται από αισθητήρες, από μαζικές δημόσιες-ιδιωτικές πηγές δεδομένων (massive public-private data sources).

Παλαιότερα η διαχείρισή μεγάλου όγκου δεδομένων απαιτούσε χρήση υπερυπολογιστών και είχε μεγάλο κόστος. Η εξέλιξη της υπολογιστικής τεχνολογίας, οι κατανεμημένοι υπολογιστές, η εικονικοποίηση (virtualization), το υπολογιστικό νέφος, καθιστούν εφικτή τη διαχείριση μεγάλου όγκου δεδομένων με μικρότερο κόστος. Η εξέλιξη αυτή δρομολογήθηκε σε μεγάλο βαθμό από εταιρίες όπως Yahoo!, Google,

Facebook, οι οποίες κατανόησαν την ανάγκη-ευκαιρία δημιουργίας εσόδων από τις τεράστιες ποσότητες δεδομένων που διαχειρίζονταν.

Οι νέες τεχνολογίες αποθήκευσης, πρόσβασης και ανάλυσης τεράστιας ποσότητας δεδομένων σε (σχεδόν) πραγματικό χρόνο συνδέονται με καινοτομικές λύσεις, όπως οι τεχνολογίες MapReduce, Hadoop και Big Table, οι οποίες οδήγησαν σε μια νέα γενιά διαχείρισης δεδομένων (new generation of data management).

Τεχνολογία MapReduce

Η τεχνολογία MapReduce σχεδιάστηκε από την Google για να υποστηρίξει την αποτελεσματική εκτέλεση, σε λειτουργία δέσμης (batch mode), ενός συνόλου λειτουργιών διαχείρισης μεγάλου όγκου δεδομένων. Η συνιστώσα ή συνάρτηση «χάρτης αντιστοίχισης» (“map” component-function) ουσιαστικά διαχειρίζεται τον κατανεμημένο υπολογισμό, δηλαδή κατανέμει τις προγραμματιστικές εργασίες επίλυσης ενός προβλήματος σε έναν πολύ μεγάλο αριθμό συστημάτων. Διαχειρίζεται την ανάθεση των στοιχειωδών εργασιών (tasks) με τρόπο που εξισορροπεί το φόρτο (workload) και διαχειρίζεται και την ανάνηψη (recovery) από αστοχίες. Η συνιστώσα ή συνάρτηση «αναγωγή και ομαδοποίηση αποτελεσμάτων» (“reduce” component- function) συγκεντρώνει όλα τα αποτελέσματα της συνάρτησης “map” κάνει αναγωγή (reduce, «μείωση») ώστε να απαλείψει τις επαναλήψεις και τελικά δίδει ένα αποτέλεσμα. Ένα παράδειγμα χρήσης της τεχνολογίας (του αλγόριθμου) MapReduce θα ήταν να προσδιορίσουμε πόσες σελίδες όλων των βιβλίων μιας ψηφιακής βιβλιοθήκης είναι γραμμένες σε καθεμία από πέντε διαφορετικές γλώσσες.

Τεχνολογία Big Table

Η τεχνολογία Big Table σχεδιάστηκε και υλοποιήθηκε από την Google ως ένα κατανεμημένο σύστημα αποθήκευσης (distributed storage system) για τη διαχείριση δομημένων δεδομένων υψηλής κλιμάκωσης (highly scalable structured data). Η οργάνωση των δεδομένων γίνεται γραμμές και στήλες πινάκων, αλλά διαφέρει από το σχεσιακό μοντέλο γιατί η έννοια Big Table συνδέεται με έναν αραιό, κατανεμημένος, επίμονο πολυδιάστατο ταξινομημένο χάρτη (“a sparse, distributed, persistent multidimensional sorted map”) ο οποίος χρησιμοποιείται για την αποθήκευση τεράστιου όγκου δεδομένων σε διακομιστές-εξυπηρετητές (servers).

Τεχνολογία Hadoop

Η τεχνολογία Hadoop είναι ένα πλαίσιο λογισμικού (software framework), το οποίο προέρχεται από και βασίζεται σε MapReduce και Big Table. Είναι ένα έργο το οποίο διαχειρίζεται η Apache. Η τεχνολογία Hadoop επιτρέπει σε εφαρμογές που βασίζονται στο MapReduce να εκτελούνται σε μεγάλα συμπλέγματα (large clusters) υλικού (σε “Commodity hardware”). Ο όρος “Commodity hardware” χρησιμοποιείται για φθηνούς, τυποποιημένους διακομιστές, που είναι εύκολο να αγοραστούν «από το ράφι» (off the shelf) και από οποιονδήποτε προμηθευτή (vendor), και είναι δικτυωμένοι για να παρέχουν περισσότερη επεξεργαστική ισχύ. Αποτελούν εξαιρετική λύση όταν δεν θέλουμε να χρησιμοποιήσουμε ακριβούς και πολύπλοκους υπερυπολογιστές. Το έργο Hadoop είναι το θεμέλιο της αρχιτεκτονικής υπολογισμού (computing architecture) που υποστηρίζει τη Yahoo!. Το Hadoop έχει σχεδιαστεί για να παραλληλίζει την επεξεργασία δεδομένων σε υπολογιστικούς κόμβους (computing nodes), να επιταχύνει τους υπολογισμούς και να διασφαλίζει μικρή καθυστέρηση (to hide latency). Αποτελείται από δύο κύριες συνιστώσες (components):

- ένα μαζικά επεκτάσιμο κατανεμημένο σύστημα αρχείων (massively scalable distributed file system) που μπορεί να υποστηρίξει petabyte δεδομένων και
- μία μαζικά επεκτάσιμη μηχανή MapReduce (massively scalable MapReduce engine) που υπολογίζει με μαζική επεξεργασία (batch processing) τα αποτελέσματα.

Ο ενδιαφερόμενος αναγνώστης για Hadoop και MapReduce μπορεί να μελετήσει και το «Κεφάλαιο 8: Υπολογιστικές Μέθοδοι για Ανάλυση Μεγάλων Δεδομένων (Hadoop και MapReduce)»,

Βερύκιος, Β., Καγκλής, Β., & Σταυρόπουλος, Η. (2015). Η επιστήμη των δεδομένων μέσα από τη γλώσσα R [Προπτυχιακό εγχειρίδιο]. Κάλλιπος, Ανοικτές Ακαδημαϊκές Εκδόσεις. http://hdl.handle.net/11419/2965

2.5.3.5 Αναλύσεις μεγάλων δεδομένων (big data analytics)

Η επιχείρηση στην προσπάθεια να κατανοήσει και να αξιοποιήσει όλα τα δεδομένα, σε όλες τις μορφές τους, χρειάζεται πολλές διαφορετικές προσεγγίσεις στην ανάλυση τους, ανάλογα με το πρόβλημα που πρέπει να επιλυθεί. Ορισμένες αναλύσεις χρησιμοποιούν παραδοσιακή αποθήκη δεδομένων, ενώ άλλες αναλύσεις επωφελούνται περισσότερο από προηγμένες αναλύσεις πρόβλεψης (predictive analytics).

Ανάλυση δεδομένων, αποθήκη δεδομένων και πρατήρια δεδομένων (Analytical data warehouses and data marts)

Μετά την ταξινόμηση τεράστιου όγκου δεδομένων μπορούμε να επιλέξουμε ένα υποσύνολο δεδομένων, να ανακαλύψουμε μοτίβα (patterns) τα οποία ενδιαφέρουν την επιχείρηση και να τα αποθηκεύσουμε σε αποθήκες και πρατήρια δεδομένων. Η λύση αυτή παρέχει συμπίεση πληροφορίας (compression), πολυεπίπεδη κατάτμηση δεδομένων (multilevel partitioning) και αποτελεί θεμέλιο για μια αρχιτεκτονική μαζικά παράλληλης επεξεργασίας (massively parallel processing).

Αναλύσεις μεγάλων δεδομένων (big data analytics)

Η διαχείριση και ανάλυση δεδομένων μεγάλης κλίμακας, petabyte και άνω, για να οδηγεί στην επίλυση ενός επιχειρηματικού προβλήματος ώστε να έχει κάποιο αντίκτυπο στην επιχείρηση, συχνά απαιτεί περίπλοκες αναλύσεις μεγάλων δεδομένων, διαχείριση ευρέως κατανεμημένων δεδομένων και βελτιστοποίηση αποτελεσμάτων. Για παράδειγμα, ορισμένοι οργανισμοί χρησιμοποιούν περίπλοκα μοντέλα πρόβλεψης που συνδυάζουν δομημένα και μη δομημένα δεδομένα για να προβλέψουν απάτες (to predict fraud). Οι περίπλοκες αναλύσεις μέσων κοινωνικής δικτύωσης (Social media analytics), η «δύσκολη» ανάλυση κειμένου (text analytics) και πολλές άλλες αναλύσεις ενδιαφέρουν και χρησιμοποιούνται από όλο και περισσότερους οργανισμούς που θέλουν να κατανοήσουν και να αξιοποιήσουν πληροφορίες «υποκρυπτόμενες στα μεγάλα δεδομένα.

2.5.3.6 Αναφορές και οπτικοποίηση (Reporting and visualization)

Οι οργανισμοί βασίζονταν πάντα στη δημιουργίας αναφορών για να κατανοήσουν τα δεδομένα, π.χ., οι αναφορές με τα στοιχεία μηνιαίων πωλήσεων αποτελούν εργαλείο για τις προβλέψεις ανάπτυξης του οργανισμού. Τα μεγάλα δεδομένα αλλάζουν τον τρόπο διαχείρισης και χρήσης των δεδομένων και μια επιχείρηση μπορεί να χρησιμοποιήσει μια νέα γενιά εργαλείων. Η αναφορά και η οπτικοποίηση δεδομένων γίνονται εργαλεία για την εξέταση του πλαισίου (context) λειτουργίας της επιχείρησης, του τρόπου συσχέτισης των δεδομένων και του αντίκτυπου αυτών των συσχετίσεων στο μέλλον.

Οι ενδιαφερόμενοι αναγνώστες για θέματα οπτικοποίησης αλλά και διερευνητικής ανάλυσης δεδομένων μπορούν να μελετήσουν το κεφάλαιο:

Κύρκος, Ε. (2015). Οπτική και Διερευνητική Ανάλυση Δεδομένων [Κεφάλαιο]. Στο Κύρκος, Ε. 2015. Επιχειρηματική ευφυΐα και εξόρυξη δεδομένων [Προπτυχιακό εγχειρίδιο]. Κάλλιπος, Ανοικτές Ακαδημαϊκές Εκδόσεις. κεφ 6. http://hdl.handle.net/11419/1232

2.5.3.7 Εφαρμογές μεγάλων δεδομένων και μεταμόρφωση της καθημερινής ζωής και της συμπεριφοράς των αγορών (transformation of the behavior of a market)

Παραδοσιακά, η επιχείρηση περιμένει ότι η διαχείριση δεδομένων θα βελτιώσει την καθημερινή λειτουργία της και θα συνεισφέρει στην καλύτερη διαχείριση των πελατών, των παραγγελιών, των τιμολογίων, των πόρων κ.λπ. Επιπλέον, προσδοκά ότι η ανάλυση των δεδομένων θα βοηθήσει να απαντηθούν ερωτήματα σχετικά με τις ενέργειες που πρέπει να γίνουν και πότε ακριβώς είναι σκόπιμο να γίνουν. Με την έλευση των μεγάλων δεδομένων, οι προσδοκίες αλλάζουν και ως αποτέλεσμα αναδύονται εφαρμογές οι οποίες έχουν σχεδιαστεί ειδικά για να εκμεταλλεύονται τα μοναδικά χαρακτηριστικά των μεγάλων δεδομένων. Οι εφαρμογές μεγάλων δεδομένων μπορούν να έχουν μεγάλες επιπτώσεις και να οδηγήσουν ακόμη και σε «μεταμόρφωση της συμπεριφοράς των αγορών» (transformation of the behavior of a market)**.** Ακολουθούν παραδείγματα.

Στον τομέα της υγειονομικής περίθαλψης, οι εφαρμογές μεγάλων δεδομένων μπορούν παρακολουθούν και να διαχειρίζονται αποτελεσματικότερα τις παρεμβάσεις σε ΜΕΘ, επιδημίες, πρόωρα βρέφη κ.λπ.

Στον κατασκευαστικό τομέα, οι εφαρμογές μπορούν να αναλάβουν τη διαχείριση παραγωγής, να αποτρέψουν τη διακοπή λειτουργίας ενός μηχανήματος, κ.λπ.

Στον τομέα της διαχείρισης της κυκλοφορίας, οι εφαρμογές μπορούν να μειώσουν τον αριθμό των κυκλοφοριακών συμφορήσεων, την κίνηση σε πολυσύχναστους αυτοκινητόδρομους, τον αριθμό των ατυχημάτων, μπορούν να εξοικονομήσουν καύσιμα και να μειώσουν την περιβαλλοντική ρύπανση.

Στον επιχειρηματικό τομέα, οι εφαρμογές μπορούν να βοηθήσουν στην πρόβλεψη και την επίλυση προβλημάτων και στην αξιοποίηση ευκαιριών, στη βελτίωση της ικανοποίησης των πελατών, στη βελτίωση της ποιότητας των προϊόντων κ.λπ.

2.6 Τύποι μεγάλων δεδομένων (Big Data Types)

Στην ενότητα αυτή θα περιγραφούν τύποι δεδομένων και θα μελετηθεί η προέλευσή τους και θα δοθούν παραδείγματα.

2.6.1 Δομημένα Δεδομένα

Ο όρος δομημένα δεδομένα αναφέρεται γενικά σε δεδομένα που έχουν καθορισμένο μήκος και μορφότυπο (format). Παραδείγματα δομημένων δεδομένων περιλαμβάνουν τα λειτουργικά δεδομένα συναλλαγών, κωδικούς (number), ημερομηνίες παραγγελιών (date) ονόματα (συμβολοσειρές, π.χ., char(70)) και διευθύνσεις πελατών (συμβολοσειρές) κ.λπ.

Οι περισσότεροι ειδικοί συμφωνούν ότι αυτού του είδους τα δεδομένα αντιπροσωπεύουν περίπου το 20 τοις εκατό του συνόλου των δεδομένων. Τα δομημένα δεδομένα συνήθως αποθηκεύεται σε σχεσιακή βάση δεδομένων την οποία διαχειριζόμαστε με τη γλώσσα δομημένης αναζήτησης (SQL). Στα δομημένα δεδομένα της επιχείρησης περιλαμβάνονται και τα δεδομένα διαχείρισης σχέσεων πελατών (customer relationship management-CRM), τα δεδομένα προγραμματισμού επιχειρησιακών πόρων (enterprise resource planning-ERP), οικονομικά δεδομένα κ.λπ.. Συχνά τα δομημένα δεδομένα ενσωματώνονται-ολοκληρώνονται (integrated data) σε μια αποθήκη δεδομένων για ανάλυση.

Παραθέτουμε παραδείγματα δομημένων δεδομένων που δημιουργούνται από τον άνθρωπο,

Δεδομένα εισαγωγής (Input data), δηλαδή δεδομένα τα οποία εισάγουν οι χρήστες σε ένα πληροφοριακό σύστημα ή σε μία ηλεκτρονική φόρμα, όπως όνομα και επώνυμο, ημερομηνία γέννησης, εισόδημα, διάφορες απαντήσεις στις κλειστές ερωτήσεις ενός δομημένου ερωτηματολογίου έρευνας (survey) κ.λπ.

Δεδομένα ροών κλικ του χρήστη (Click-stream data), τα οποία δημιουργούνται κάθε φορά που ο χρήστης κάνει κλικ σε έναν σύνδεσμο σε έναν ιστότοπο.

Αυτά τα δύο είδη δεδομένων μπορούν να αποτελέσουν μέρος του προφίλ του χρήστη (user profile, user model) και είναι χρήσιμα για την κατανόηση της βασικής συμπεριφοράς των πελατών κ.λπ.

Παραθέτουμε παραδείγματα δομημένων δεδομένων που δημιουργούνται από μηχανή.

1) **Δεδομένα από αισθητήρα**. Ειδική αναφορά πρέπει να γίνει για:
    - **Ετικέτες ID ραδιοσυχνοτήτων (radio frequency ID)** RFID tags. Η δημοφιλής τεχνολογία RFID χρησιμοποιεί τσιπ υπολογιστή (computer chips) για να ιχνηλατεί αντικείμενα σε απόσταση και ενδιαφέρει σε εφαρμογές διαχείρισης της εφοδιαστικής αλυσίδας και εφαρμογές ελέγχου των αποθεμάτων Ένα παράδειγμα εφαρμογής της τεχνολογίας είναι η παρακολούθηση της μεταφοράς και τοποθέτησης βιβλίων από τα ράφια ενός κτηρίου βιβλιοθήκης στα ράφια ενός άλλου κτηρίου στην περίπτωση μετεγκατάστασης της βιβλιοθήκης.
    - **Δεδομένα του Παγκόσμιου Συστήματος Εντοπισμού Θέσης (Global Positioning System)** GPS. Όλα τα έξυπνα τηλέφωνα (smartphone) περιέχουν αισθητήρες GPS που μπορούν να χρησιμοποιηθούν για την κατανόηση της συμπεριφοράς των συνδρομητών.
    - Ιατρικές φορητές συσκευές
2) **Δεδομένα καταγραφής Ιστού (Web log data).** Καταγραφή δεδομένων λειτουργίας διακομιστών, εφαρμογών, κ.λπ. Η ανάλυση αυτού του τεράστιου όγκου δεδομένων είναι χρήσιμη σε πολλές περιπτώσεις, π.χ., πρόβλεψη παραβιάσεων ασφάλειας (security breaches).
3) **Δεδομένα σημείου πώλησης (Point-of-sale data).** Η χρήση γραμμωτού κώδικα (bar code) στο ταμείο επιτρέπει, με την κατάλληλη εφαρμογή, τη σύνδεση διαφόρων στοιχείων όλων των αγορών (purchasing) με όλα τα προϊόντα που αγοράζονται οδηγώντας σε ένα πάρα πολύ μεγάλο σύνολο δεδομένων.

2.6.2 Κατανόηση του ρόλου των σχεσιακών βάσεων δεδομένων στα μεγάλα δεδομένα

Από τον αντικειμενοστρφή προγραμματισμό είναι γνωστή η έννοια «διατηρητέα αντικείμενα» (“Persistent Objects”). Ο όρος αναφέρεται στα αντικείμενα που υπάρχουν στη μνήμη ακόμη και πέρα από τη διάρκεια της διαδικασίας (process) που τα δημιουργεί. Αυτά τα αντικείμενα στη συνέχεια μπορούν να αποθηκεύονται στη βάση δεδομένων. Ένα αντικείμενο βρίσκεται σε «διατηρητέα» κατάσταση (Persistent state) εάν έχει κάποια αναφορά (reference) στη βάση δεδομένων, δηλαδή αντιπροσωπεύει-αναπαριστά (reprsent) κάποια γραμμή στη βάση δεδομένων και του έχει εκχωρηθεί τιμή αναγνωριστικού (identifier value).

Στην περίπτωση process persistence οι διεργασίες δεν «σκοτώνονται» (kill) ή τερματίζονται (shut down) από άλλες διεργασίες και υπάρχουν μέχρι να τις «σκοτώσει» ο χρήστης.

Στο πλαίσιο του κεφαλαίου αυτού ο όρος **διατήρηση εκδόσεων δεδομένων (Data persistence)** αναφέρεται:

- σε ένα αντικείμενο το οποίο δεν διαγράφεται μέχρι να εμφανιστεί η ανάγκη να αφαιρεθεί από τη μνήμη. Ορισμένα μοντέλα βάσεων δεδομένων παρέχουν μηχανισμούς για την αποθήκευση μόνιμα δεδομένων με τη μορφή αντικειμένων.
- στον τρόπο με τον οποίο μια βάση δεδομένων διατηρεί τις εκδόσεις της όταν τροποποιείται.

Πρόγονος των persistent data stores θα μπορούσε να θεωρηθεί το σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (RDBMS). Το σχεσιακό μοντέλο προτάθηκε από τον Edgar Codd τη δεκαετία του 1970 και

υιοθετήθηκε από όλους τους μεγάλους προμηθευτές (vendors), την IBM, την Oracle, τη Microsoft και άλλους. Τα σχεσιακά προϊόντα διαχείρισης βάσης δεδομένων κυριαρχούν τις τελευταίες δεκαετίες και επομένως η κατανόηση της σχεσιακής βάσης δεδομένων είναι σημαντική. Επειδή, όμως, μπορεί να είναι δύσκολο να χρησιμοποιηθούν σχεσιακά προϊόντα όταν διαχειριζόμαστε πολύ μεγάλες ροές διαφορετικών τύπων δεδομένων προτάθηκαν και χρησιμοποιούνται και νέα μοντέλα βάσεων δεδομένων, όπως NoSQL βάσεις δεδομένων, βάσεις δεδομένων ροής (streaming databases) κ.λπ.

Οι προμηθευτές σχεσιακών βάσεων δεδομένων δεν παραμένουν παθητικά και «επεκτείνουν» τις σχεσιακές βάσεις δεδομένων για να διαχειρίζονται μεγάλα δεδομένα.

2.6.3 Διερεύνηση πηγών μη δομημένων δεδομένων

Τα μη δομημένα δεδομένα κυριαρχούν στην καθημερινή ζωή φυσικών προσώπων και επιχειρήσεων και επηρεάζουν βαθύτατα τη ζωή μας, τις καταναλωτικές μας συνήθειες, τις αγορές και τις επιχειρήσεις.

Ακολουθούν παραδείγματα μη δομημένων δεδομένων που δημιουργούνται από τον άνθρωπο:

- **Κείμενα-πληροφορίες (text) παραγόμενα εσωτερικά στην επιχείρηση.** Περιλαμβάνονται έγγραφα, αρχεία καταγραφής (logs), αποτελέσματα ερευνών (survey), μηνύματα ηλεκτρονικού ταχυδρομείου (e-mails) κ.λπ.
- **Δεδομένα μέσων κοινωνικής δικτύωσης (Social media data)**, δηλαδή δεδομένα που δημιουργούνται από τις δημοφιλείς πλατφόρμες κοινωνικών μέσων, Facebook, Twitter, LinkedIn, YouTube, Flickr κ.λπ.
- **Δεδομένα κινητής τηλεφωνίας (Mobile data).** Περιλαμβάνονται μηνύματα (κειμένου) και πληροφορίες τοποθεσίας.
- **Περιεχόμενο ιστότοπου (Website content).** Περιλαμβάνουν μη δομημένο περιεχόμενο από ιστοτόπους, Instagram, Flickr κ.λπ.

Η τεχνολογία ανάλυσης κειμένου και η συναισθηματική ανάλυση (emotion analysis) (βλέπε και ενότητα 2.14) μπορούν να χρησιμοποιηθούν στην ανάλυση μη δομημένου κειμένου από μέσα κοινωνικής δικτύωσης, από συνομιλίες πελατών με τηλεφωνικό κέντρο εξυπηρέτησης (help desk), από e-mail, από σχόλια σε ανοικτές ερωτήσεις έρευνας (survey) για την κατανόηση της συμπεριφοράς των πελατών κ.λπ. Η ανάλυση κειμένου μπορεί να εξάγει δεδομένα μετατρέψιμα σε δομημένες πληροφορίες οι οποίες αποθηκεύονται και μπορούν να χρησιμοποιηθούν-αξιοποιηθούν με διάφορους τρόπους στην επιχείρηση.

Ακολουθούν παραδείγματα μη δομημένων δεδομένων που δημιουργούνται από μηχανή.

- **Δορυφορικές εικόνες (Satellite image)** που περιλαμβάνουν δορυφορικές εικόνες για την πρόγνωση του καιρού, εικόνες δορυφορικής επιτήρησης (satellite surveillance imagery), Google Earth κ.λπ.
- **Επιστημονικά δεδομένα** που περιλαμβάνουν σεισμικά δεδομένα, δεδομένα της γήινης ατμόσφαιρας κ.λπ.
- **Φωτογραφίες και βίντεο** που χρησιμοποιούνται σε ποικίλες εφαρμογές, ασφάλειας και επιτήρησης (security & surveillance), διαχείρισης της κυκλοφορίας κ.λπ.
- **Δεδομένα ραντάρ ή σόναρ (Radar or sonar data)** που χρησιμοποιούνται: στη μετερεολογία, την ωκεανογραφία κ.λπ.

2.6.4 Δομημένα, ημι-δομημένα και μη δομημένα δεδομένα και ολοκλήρωση (integration) σε περιβάλλον μεγάλων δεδομένων

Στη συνέχεια συζητάμε την ολοκλήρωση των τριών τύπων δεδομένων στα μεγάλα δεδομένα.

2.6.4.1 Ημι-δομημένα δεδομένα

Τα ημιδομημένα δεδομένα (Semi-structured data) είναι ένα είδος δεδομένων που εμπίπτει μεταξύ δομημένων και μη δομημένων δεδομένων. Τα ημιδομημένα δεδομένα δεν συμμορφώνονται απαραίτητα με ένα σταθερό σχήμα, δηλαδή δεν έχουν απαραίτητα δομή (structure), αλλά μπορεί να είναι αυτοπεριγραφικά (self-describing). Επίσης, μπορεί να περιλαμβάνουν απλά ζεύγη ετικετών/τιμών (label/value pairs). Ακολουθεί παράδειγμα:

<family_name>=Adams

<father>=Jim

<mother>=Jane

<daughter>=Joan

Παραθέτουμε παράδειγμα XML εγγράφου.

```
<?xml version = "1.0"?>
<contact-info>
   <name>Jin Adams</name>
   <company>University of Attica</company>
   <phone>6974123456</phone>
</contact-info>
```

2.6.4.2 Μη δομημένα δεδομένα και σύγχρονα συστήματα διαχείρισης περιεχομένου

Οι επιχειρήσεις αποθηκεύουν κάποια μη δομημένα δεδομένα σε βάσεις δεδομένων, αλλά χρησιμοποιούν επίσης εταιρικά συστήματα διαχείρισης περιεχομένου, (enterprise) content management systems) CMS ή ECM, με τα οποία και διαχειρίζονται τον κύκλο ζωής του πολυμεσικού περιεχομένου τους, το οποίο μπορεί να περιλαμβάνει διάφορες μορφές, ιστοσελίδα, έγγραφο, εικόνα κ.λπ.

Σύμφωνα με τον μη κερδοσκοπικό οργανισμό Association for Information and Image Management (AIIM; www.aiim.org):

«ένα σύστημα Enterprise Content Management (ECM) περιλαμβάνει τις στρατηγικές, μεθόδους και εργαλεία που χρησιμοποιούνται για σύλληψη, διαχείριση, αποθήκευση, διατήρηση και παροχή περιεχομένου και εγγράφων που σχετίζονται με οργανωτικές διαδικασίες»

("ECM comprises strategies, methods, and tools used to capture, manage, store, preserve, and deliver content and documents related to organizational processes.")

Τεχνολογίες που περιλαμβάνονται στο ECM είναι η διαχείριση εγγράφων (document management), η διαχείριση εγγραφών (records management), η απεικόνιση (imaging), η διαχείριση ροής εργασιών (workflow management), η διαχείριση περιεχομένου ιστού (web content management) και η υποστήριξη της συνεργασίας (collaboration).

2.7 Θέματα ενοποίησης διαφόρων τύπων δεδομένων σε περιβάλλον δεδομένων μεγάλης κλίμακας

Στην πράξη είναι πιθανότερο να χρησιμοποιήσουμε μια υβριδική προσέγγιση για τη διαχείριση των μεγάλων δεδομένων. Για παράδειγμα, δεν είναι πάντα αναγκαίο να εγκαταστήσουμε μια πλατφόρμα Hadoop και να «φορτώσουμε» σε αυτήν όλο το περιεχόμενο ειδήσεων (news content) επειδή η πλατφόρμα αυτή εξειδικεύεται στη διαχείριση μη δομημένων δεδομένων.

Παραθέτουμε παραδείγματα περιπτώσεων στις οποίες έχει νόημα μια εταιρεία να θελήσει να αξιοποιήσει μεγάλα δεδομένα σε πραγματικό χρόνο για να αποκτήσει συγκριτικό πλεονέκτημα.

- Παρακολούθηση (Monitoring) σε πραγματικό χρόνο για την εξακρίβωση απάτης (fraud)
- Παρακολούθηση των ειδήσεων (news feeds) και μέσων κοινωνικής δικτύωσης (social media) σε πραγματικό χρόνο για τον προσδιορισμό γεγονότων που είναι πιθανό να επηρεάσουν τις χρηματοπιστωτικές αγορές,
- Παρακολούθηση ροών Twitter σε πραγματικό χρόνο ώστε να δρομολογήσουμε αποδοτικότερα τη διαφήμιση κατά τη διάρκεια των αγώνων του παγκοσμίου πρωταθλήματος ποδοσφαίρου.

Οι ροές δεδομένων (streaming data) μπορεί να διαφέρουν, π.χ., ροές που τροφοδοτούνται με μεγάλη ταχύτητα από μία μικρή ποικιλία πηγών, ροές που τροφοδοτούνται από μεγάλη ποικιλία πηγών, ροές που τροφοδοτούνται με μεγάλη ταχύτητα από μεγάλη ποικιλία πηγών. Σε όλες τις περιπτώσεις η απαίτηση για διαχείριση σε πραγματικό χρόνο, σε συνδυασμό με τεράστιο όγκο δεδομένων ή με την ταχύτητα τροφοδότησης, ενδεχομένως θα κατακλύσει τα συστήματα της επιχείρησης.

Το κρίσιμο ερώτημα είναι αν θα επιλέξουμε τη διαχείριση των ροών δεδομένων σε πραγματικό χρόνο με τις νεότερες δυνατότητες ή αν θα πρέπει να επιλύσουμε το πρόβλημα με τις παραδοσιακές δυνατότητες διαχείρισης πληροφοριών ή αν θα επιλέξουμε έναν συνδυασμό και των δύο.

Αν χρειαζόμαστε διαχείριση μεγάλων δεδομένων σε πραγματικό χρόνο τότε αυξάνονται οι απαιτήσεις υποδομής για να υποστηρίξουμε την κατανεμημένη επεξεργασία (distributed computing) που είναι απαραίτητη.

Παραθέτουμε μερικά παραδείγματα των χαρακτηριστικών της απαιτούμενης υποδομής.

- Η απαίτηση για «χαμηλή χρονική καθυστέρηση» ή «χαμηλό λανθάνοντα χρόνο» (Low latency) παροχής υπηρεσίας σημαίνει ότι πρέπει να αναβαθμίσουμε υπολογιστική ισχύ (compute power) και να αναιρέσουμε τους περιορισμούς δικτύου (network constraints).
- Η επεκτασιμότητα (Scalability) συνδέεται με τη δυνατότητα διατήρησης της σε συνθήκες αυξανόμενου φόρτου (loads)
- Η ευελιξία (Versatility) σημαίνει ότι το σύστημα πρέπει να υποστηρίζει δομημένες και μη δομημένες ροές δεδομένων.

Παρατηρήσεις

Η ανάγκη επεξεργασίας συνεχώς αυξανόμενων ποσοτήτων διαφορετικού τύπου δεδομένων συνηγορεί για την επιλογή της μίσθωσης υπηρεσιών cloud.

Οι εφαρμογές μεγάλων δεδομένων μιας επιχείρησης δεν χρειάζεται να βασίζονται μόνο σε δεδομένα τα οποία τροφοδοτούνται από εσωτερικά συστήματα της επιχείρησης αλλα μπορεί να αξιοποιούν δεδομένα από εξωτερικές πηγές (ή και να βασίζονται εξ ολοκλήρου σε εξωτερικές πηγές). Για παράδειγμα, θα μπορούσατε

να επεξεργάζεστε μεγάλα δεδομένα τροφοδοτούμενα από μέσα κοινωνικής δικτύωσης και στατιστικά στοιχεία τρίτων.

Τα δεδομένα δεν είναι απαραίτητο να τροφοδοτούν τις εφαρμογές σε πραγματικό χρόνο για να έχουμε διαχείριση μεγάλων δεδομένων. Αρκεί να έχουμε μεγάλους όγκους και τα δεδομένα να είναι διαφορετικής φύσης. Και η περίπτωση αυτή θα μπορούσε ακόμα να χαρακτηριστεί ως πρόβλημα μεγάλων δεδομένων.

Τέλος, αν είναι δυνατόν, θα ήταν σκόπιμο να χρησιμοποιούμε τα δεδομένα στη φυσική (εγγενή) τους μορφή (Native format) χωρίς χρονοβόρους μετασχηματισμούς που κοστίζουν σε χρόνο και χρήμα.

Η διαχείριση δεδομένων απαιτεί τη χρήση συνδέσεων (connectors) και μεταδεδομένων (metadata).

Συνδέσεις

Οι συνδέσεις (Connectors) επιτρέπουν την άντληση μεγάλων δεδομένων από πηγές. Παραδείγματα είναι Facebook connector, Twitter connector.

Η Oracle παρέχει τον προσαρμογέα Facebook (Facebook Adapter) ο οποίος δίνει τη δυνατότητα ενοποίησης (integration) εφαρμογών (π.χ., αποθήκης δεδομένων) με μια εφαρμογή Facebook. Παρέχει, επίσης, τον προσαρμογέα Twitter (Twitter Adapter) ο οποίος δίνει τη δυνατότητα ενοποίησης (integration) εφαρμογών με μια εφαρμογή Twitter.

Using the Facebook Adapter with Oracle Integration

Using the Twitter Adapter with Oracle Integration

Είναι ενδιαφέρον να αναφέρουμε την ενιαία εφαρμογή σύνδεσης Facebook Connect, μια ενιαία εφαρμογή σύνδεσης (single sign-on application) που επιτρέπει στους χρήστες να αλληλοεπιδρούν σε άλλους ιστότοπους μέσω του λογαριασμού τους στο Facebook.

What is Facebook Connect? - Definition from WhatIs.com (techtarget.com)

Μεταδεδομένα (Metadata)

Ο ρόλος των μεταδεδομένων είναι κρίσιμος για την ενοποίηση-ενσωμάτωση δεδομένων διαφορετικού τύπου σε μεγάλα δεδομένα. Ο «κλασσικός» ορισμός των μεταδεδομένων είναι «δεδομένα για τα δεδομένα» (Metadata is “data about data.”).

Σε μία πιο περιγραφική προσέγγιση θα μπορούσαμε να θεωρήσουμε ότι τα μεταδεδομένα περιλαμβάνουν ορισμούς, αντιστοιχίσεις και άλλα χαρακτηριστικά τα οποία χρησιμοποιούνται για να περιγράψουν τον τρόπο εύρεσης, πρόσβασης και χρήσης των δεδομένων της επιχείρησης. Η έννοια αποθετήριο μεταδεδομένων (metadata repository) αναφέρεται σε κοντέινερ (container) με συνεπείς ορισμούς επιχειρηματικών δεδομένων και κανόνων για την αντιστοίχιση δεδομένων στις πραγματικές φυσικές τους τοποθεσίες στο σύστημα.

Η ανάλυση μεγάλων δεδομένων μπορεί να βασιστεί σε δύο ή περισσότερα σύνολα δεδομένων, π.χ., σε αποθήκη δεδομένων και πηγή μεγάλων δεδομένων, αρκεί οι πηγές να περιλαμβάνουν δικά τους προσεκτικά σχεδιασμένα μεταδεδομένα. Αυτό ισχύει για πολλές πηγές δεδομένων. Οι σύνθετοι ιστότοποι ηλεκτρονικού εμπορίου (e-commerce) περιλαμβάνουν καλά καθορισμένα δεδομένα.

Ο μη κερδοσκοπικός οργανισμός The Open Data Foundation έχει ως στόχους την προώθηση και την υιοθέτηση παγκόσμιων προτύπων μεταδεδομένων (global metadata standards) και στην ανάπτυξη ανοιχτού κώδικα για τη χρήση στατιστικών δεδομένων. Εστιάζει στους τομείς της οικονομίας, της υγειονομικής περίθαλψης, της εκπαίδευσης, της εργασίας, των κοινωνικών επιστημών, της τεχνολογίας, της γεωργίας, της ανάπτυξης και του περιβάλλοντος. (www.opendatafoundation.org)

2.8 Αρχιτεκτονικές επεξεργασίας δεδομένων, βάση δεδομένων και οι συναλλαγές και η σημασία τους

Υπάρχουν δύο βασικές αρχιτεκτονικές επεξεργασίας δεδομένων (Data processing techniques):

- Η κεντρική επεξεργασία (Centralized processing) στην οποία όλα τα δεδομένα συλλέγονται σε μια ενιαία κεντρική περιοχή αποθήκευσης (centralized storage area) και υποβάλλονται σε επεξεργασία από έναν μόνο υπολογιστή με μεγάλες δυνατότητες όσον αφορά τη μνήμη, τον επεξεργαστή και τη δευτερεύουσα μνήμη (δίσκους).
- Η κατανεμημένη επεξεργασία (Distributed processing) στην οποία τα δεδομένα και η επεξεργασία τους κατανέμονται σε γεωγραφικές περιοχές ή/και κέντρα δεδομένων (Data Centers) και μετά την επεξεργασία τους όλα τα αποτελέσματα της επεξεργασίας συνενώνονται σε μια κεντρική αποθήκη (storage)

Υπάρχουν διάφορες αρχιτεκτονικές κατανεμημένης επεξεργασίας:

- Η **αρχιτεκτονική πελάτη-διακομιστή (Client–server)** στην οποία ο πελάτης αναλαμβάνει τη συλλογή και την παρουσίαση των δεδομένων και ο διακομιστής (ή εξυπηρετητής, server) αναλαμβάνει την επεξεργασία και διαχείριση των δεδομένων.

  Σημείωση 1. Στο σημείο αυτό πρέπει να αναφέρουμε τη βασική διαφορά μεταξύ thin client, ο οποίος χρησιμοποιεί έναν απομακρυσμένο υπολογιστή για την επεξεργασία μιας εφαρμογής, και fat client.(ή thick client), ο οποίος κάνει την επεξεργασία της εφαρμογής, δηλαδή εκτελεί τοπικά τις εφαρμογές. Επισημαίνουμε ότι υπάρχουν πολλές ακόμη διαφορές μεταξύ thin client και thick client.
- Η αρχιτεκτονική τριών επιπέδων (Three-tier architecture) στην οποία η αρχιτεκτονική πελάτη-διακομιστή εξελίσσεται σε συστήματα τριών επιπέδων: ο πελάτης αναλαμβάνει τη συλλογή δεδομένων και η λογική επεξεργασία μεταφέρεται σε μια ενδιάμεση βαθμίδα υπηρεσιών (a middle tier of services), και δεν απαιτείται ο πελάτης να συνδεθεί με τον διακομιστή. Η αρχιτεκτονική τριών επιπέδων επιτρέπει την επεκτασιμότητα (scalability) κάθε επιπέδου. Χρησιμοποιείται κυρίως στις εφαρμογές ανάλυσης δεδομένων (analytical applications) και επιχειρηματικής ευφυΐας (business intelligence).
- Αρχιτεκτονική n-επιπέδων (n-tier or multitier architecture) στην οποία οι πελάτες, το ενδιάμεσο λογισμικό (middleware), οι εφαρμογές και οι διακομιστές απομονώνονται σε επίπεδα (tiers). Με αυτήν την αρχιτεκτονική κάθε βαθμίδα (tier) μπορεί να κλιμακωθεί (scale) ανεξάρτητα από τις υπόλοιπες. Οι εφαρμογές Ιστού χρησιμοποιούν αυτόν τον τύπο αρχιτεκτονικής.

  Σημείωση 2. Στο σημείο αυτό πρέπει να αναφέρουμε τη βασική διαφορά μεταξύ εφαρμογών stateful και stateless. Μια αίτηση/διαδικασία Stateless δεν παραπέμπει σε πληροφορίες σχετικά με προηγούμενες λειτουργίες. Για παράδειγμα, αν μία αναζήτηση (query) διακοπεί για κάποιο λόγο θα πρέπει απλά να ξεκινήσουμε νέα. Μια εφαρμογή stateful «θυμάται» συγκεκριμένες λεπτομέρειες ενός χρήστη, όπως προφίλ και προηγούμενες ενέργειες. Ως παράδειγμα, θυμηθείτε το καλάθι αγορών.

  Σημείωση 3. Αν οι συνδέσεις είναι stateful με τη βάση δεδομένων τότε καταναλώνονται πόροι του DBMS, έχουμε περιορισμό της απόδοση και τελικά μείωση του αριθμού των εξυπηρετούμενων χρηστών. Στην περίπτωση three/N tier οι κλήσεις των πελατών (clients) είναι stateless.(τουλάχιστον όσον αφορά το πρωτόκολλο HTTP).
- Αρχιτεκτονική συμπλέγματος (Cluster architecture) στην οποία όλα τα μηχανήματα συνδέονται σε αρχιτεκτονική δικτύου με χρήση λογισμικού ή υλικού και αναλαμβάνουν την παράλληλη επεξεργασία των δεδομένων. Κάθε μηχάνημα του συμπλέγματος αναλαμβάνει μια εργασία (task) την οποία εκτελεί

τοπικά και όλα τα αποτελέσματα (result sets) συλλέγονται αρχικά σε έναν κύριο διακομιστή (master server) ο οποίος τα επιστρέφει στον χρήστη.

- Αρχιτεκτονική από μηχανή-σε-μηχανή (Peer-to-peer architecture) στην οποία όλες οι εργασίες επεξεργασίας κατανέμονται μεταξύ όλων των μηχανών (peer). Κάθε μηχάνημα μπορεί να αναλάβει το ρόλο ενός πελάτη ή ενός διακομιστή ή απλώς να επεξεργαστεί δεδομένα.

Η χρήση εφαρμογών λογισμικού, όπως CRM, ERP, Supply-chain-management (SCM), επεξεργασία συναλλαγών (transaction processing) συνήθως χρησιμοποιούν αρχιτεκτονική client server. Οι εφαρμογές Ιστού (Web applications) απαιτούν αρχιτεκτονική επεκτάσιμη (scalable) και ευέλικτη (flexible). Η αποθήκη δεδομένων απαιτεί μια πλατφόρμα υποδομής, ισχυρή (robust) και επεκτάσιμη.

Η φύση των δεδομένων και ο τύπος της επεξεργασίας καθορίζει και την αρχιτεκτονική που θα υιοθετηθεί.

Αρχικά θα συζητήσουμε το ρόλο των συναλλαγών και το εννοιολογικό πλαίσιο της αρχιτεκτονικής των εφαρμογών βάσης δεδομένων.

Συναλλαγή ή δοσοληψία (transaction) και βάσεις δεδομένων

Η Συναλλαγή (transaction) μπορεί να οριστεί σαν ένα σύνολο από λειτουργίες πελάτη (client operations) που αποτελούν σε λογικό επίπεδο, μια μονάδα εργασίας. Η μονάδα, δηλαδή όλες οι λειτουργίες που την αποτελούν, εκτελείται με τέτοιο τρόπο ώστε είτε όλες οι μεταβολές στη βάση δεδομένων να επικυρώνονται (δηλαδή να καταγράφονται μόνιμα στη βάση δεδομένων) ή όλες οι μεταβολές να αναιρούνται. Μία συναλλαγή πελάτη θα πρέπει να αφήνει τη βάση δεδομένων σε μία κατάσταση που να εξασφαλίζει ότι η συναλλαγή εκτελείται σε απομόνωση σε σχέση με τις συναλλαγές των άλλων πελατών που εκτελούνται ταυτόχρονα. Μετά την ανάκτηση από μια κατάρρευση του συστήματος, η συναλλαγή αφήνει τη βάση δεδομένων σε μια συνεπή κατάσταση, δηλαδή είτε όπως ήταν πριν από την έναρξη της εκτέλεσής της, ή όπως είναι προδιαγεγραμμένο να γίνει μετά την εκτέλεση της συναλλαγής στο σύνολό της.

Στην εικόνα 2.14 εστιάζουμε στην αρχιτεκτονική των εφαρμογών λογισμικού (application software), ενός συστήματος βάσης δεδομένων. Οι παρεχόμενες υπηρεσίες ακολουθούν το μοντέλο πελάτη εξυπηρετητή (client server model). Στην αρχιτεκτονική περιλαμβάνονται τρία επίπεδα:

1) Το επίπεδο διεπαφής του τελικού χρήστη.

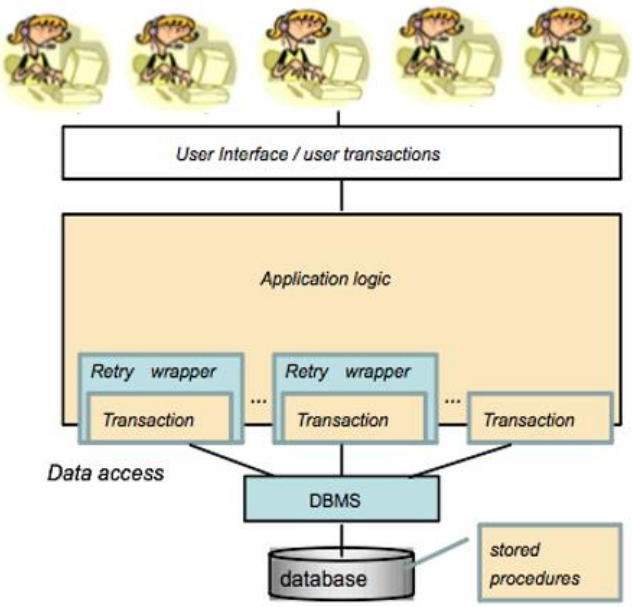
Η διεπαφή επιτρέπει στον τελικό χρήστη να κάνει με «φιλικό» τρόπο κάποια ενέργεια (σύνολο από λειτουργίες πελάτη) ή κάποιες ενέργειες. Οι επιτρεπτές ενέργειες του χρήστη υποστηρίζονται από το επόμενο επίπεδο της αρχιτεκτονικής του λογισμικού εφαρμογής, δηλαδή τη λογική της εφαρμογής (application logic). Πιο συγκεκριμένα, ο πελάτης (τελικός χρήστης) μπορεί να χρησιμοποιήσει μία σειρά συναλλαγών χρήστη (user transactions). Δηλαδή, η λογική της εφαρμογής παρέχει κάποιες περιπτώσεις-σενάρια χρήσης (use cases) που επιτρέπουν στο χρήστη να διεκπεραιώσει διάφορες εργασίες, π.χ., μεταφορά χρημάτων μεταξύ τραπεζικών λογαριασμών, κράτηση αεροπορικών εισιτηρίων.

2) Το ενδιάμεσο επίπεδο του λογισμικού εφαρμογής, δηλαδή η λογική της εφαρμογής (application logic).

Η λογική της εφαρμογής υλοποιεί τις συναλλαγές χρήστη σαν SQL συναλλαγές (SQL transactions). Για παράδειγμα, η μεταφορά χρημάτων μεταξύ λογαριασμών (ή οι πληρωμές μέσω e-banking) θα μπορούσε να περιλαμβάνει ανάκτηση (που υλοποιείται με δηλώσεις SELECT για να βλέπει ο χρήστης τα υπόλοιπα των λογαριασμών του), δηλώσεις UPDATE για να αφαιρέσει χρήματα από λογαριασμούς και δηλώσεις UPDATE για να προσθέσει χρήματα σε λογαριασμούς. Φυσικά, ο τελικός χρήστης βλέπει μόνο τη δική του περίπτωση χρήσης που είναι η μεταφορά χρημάτων και αγνοεί SELECT, UPDATE κ.λπ. Στη λογική της εφαρμογής μόνο όταν χρησιμοποιείται δήλωση COMMIT η συναλλαγή ενημερώνει τη βάση δεδομένων.

3) Το επίπεδο διαχείρισης συναλλαγών (transactions) διασφαλίζει την ασφάλεια, τη συνέπεια και την ακεραιότητα των δεδομένων. Για παράδειγμα, η δήλωση UPDATE για την αφαίρεση ποσού από ένα λογαριασμό και η δήλωση UPDATE για τη μεταφορά του ποσού σε άλλο λογαριασμό αποτελούν μία SQL συναλλαγή. Αν όλα εξελιχθούν ομαλά η συναλλαγή τερματίζεται με την καταχώρηση των μεταβολών στη βάση δεδομένων (COMMIT). Αν κάτι αποτύχει τότε η συναλλαγή τερματίζεται με ακύρωση όλων των μεταβολών (ROLLBACK).

Το λογισμικό διαχείρισης συναλλαγών αναλαμβάνει την ασφάλεια, τη συνέπεια και την ακεραιότητα των δεδομένων σε ένα περιβάλλον χρήσης του Συστήματος Βάσης Δεδομένων που επιτρέπει την ταυτόχρονη εκτέλεση πολλών συναλλαγών. Στην εικόνα 1 βλέπουμε και τις ρουτίνες επανεκτέλεσης (Retry wrappers). Αν το λογισμικό εφαρμογής διαπιστώνει κάποιο πρόβλημα ταυτοχρονισμού, δηλαδή κάποια ή κάποιες συναλλαγές αποτυγχάνουν τότε καλεί ρουτίνα (ή ρουτίνες) επανεκτέλεσης για την επανεκτέλεση της SQL συναλλαγής.



Εικόνα 2.14 SQL συναλλαγές και αρχιτεκτονική των εφαρμογών λογισμικού (application software), ενός συστήματος βάσης δεδομένων

2.8.1 Αρχιτεκτονική εφαρμογής βασιζόμενη στο μοντέλο πελάτη εξυπηρετητή (Client Server model)

Στην εικόνα 2.15 περιγράφεται ο κύκλος επεξεργασίας μιας εντολής SQL (command).

Ο πελάτης χρησιμοποιεί τις υπηρεσίες δικτύου και στέλνει ένα αίτημα εξυπηρέτησης στον εξυπηρετητή. Το λογισμικό εφαρμογής διεκπεραιώνει τον απαραίτητο διάλογο για την ικανοποίηση του αιτήματος, δηλαδή μεσολαβεί για να φτάσει το αίτημα του χρήστη στον εξυπηρετητή, ο εξυπηρετητής επεξεργάζεται το αίτημα και εξάγει τα αποτελέσματα και το λογισμικό εφαρμογής επιστρέφει τα αποτελέσματα στο χρήστη.

Κάθε εντολή (command) SQL περιλαμβάνει μία ή περισσότερες SQL δηλώσεις (statements), π.χ. δηλώσεις INSERT, UPDATE, DELETE.

Τα μεταδεδομένα και το πλάνο βελτιστοποίησης (query optimization) που εφαρμόζεται στο σύστημα βάσης δεδομένων, «παίζουν» κεντρικό ρόλο στην επεξεργασία των εντολών. Μετά τη μεταγλώττισή της στον εξυπηρετητή κάθε SQL δήλωση (που περιλαμβάνεται σε μία SQL εντολή), αφού αναλυθεί με βάση τα

μεταδεδομένα που έχουν καταγραφεί στη βάση δεδομένων, εκτελείται λαμβάνοντας υπόψη το πλάνο βελτιστοποίησης.

Ο εξυπηρετητής διατηρεί για τις SQL δηλώσεις τις σχετικές γραμμές (εγγραφές) στη μνήμη του υπολογιστή για να επιταχύνει την επεξεργασία των δεδομένων και βέβαια, όπως είπαμε, χρησιμοποιεί πλάνα «βέλτιστης εκτέλεσης» (εκτέλεσης σε ικανοποιητικό χρόνο, στην πράξη) για να βελτιστοποιήσει την εκτέλεση SQL δηλώσεων.

Υπενθυμίζεται ότι τα σχέδια εκτέλεσης αποθηκεύονται μαζί με τα άλλα μεταδεδομένα στο επίπεδο της βάσης δεδομένων που χρησιμοποιεί το ΣΔΒΔ.

Ο εξυπηρετητής χρησιμοποιεί επίσης server-side cache για να αποθηκεύσει το σύνολο των αποτελεσμάτων και χρησιμοποιεί τη cache μνήμη στη RAM ("bufferpool") για να αποθηκεύσει μόνο τις πρόσφατες γραμμές. Με αυτό τον τρόπο ο εξυπηρετητής μπορεί να μειώσει την I/O κυκλοφορία για την εξυπηρέτηση του πελάτη.

Για κάθε SQL εντολή ισχύει ότι αποτελεί «μονάδα», δηλαδή πρέπει όλες οι SQL δηλώσεις της να εκτελεστούν σωστά αλλιώς αν κάποιες δηλώσεις αποτύχουν τότε η εντολή θα πρέπει να αναιρεθεί (rollback) στο σύνολό της.

Σε απάντηση της SQL εντολής ο εξυπηρετητής ανάλογα με την επιτυχή ή όχι ολοκλήρωσή της στέλνει κάποια σχετικά διαγνωστικά μηνύματα. Στην εικόνα 2.15 βλέπουμε ότι στην περίπτωση σφάλματος η αποτυχία εκτέλεσης της εντολής του πελάτη σημειώνεται σαν εξαίρεση.

Σημείωση

Η εκτέλεση SQL δήλωσης UPDATE ή DELETE θεωρείται επιτυχής για τον προγραμματιστή εφαρμογών (εκτελείται με επιτυχία) ακόμη και αν δεν υπάρχουν γραμμές που να μεταβάλλονται από τη δήλωση. Για παράδειγμα, με δήλωση UPDATE ζητάμε όλοι οι πωλητές ενός τμήματος της επιχείρησης να πάρουν αύξηση 10%. Αν το τμήμα δεν έχει πωλητές ο πίνακας των υπαλλήλων δεν μεταβάλλεται αλλά η δήλωση εκτελείται με επιτυχία και δεν έχουμε κάποιο σφάλμα εκτέλεσης.

Από την πλευρά της εφαρμογής (application logic) αλλά και για τον τελικό χρήστη αυτές οι ενέργειες-περιπτώσεις χρήσης αποτυγχάνουν επειδή δεν ενημερώνουν/διαγράφουν στοιχεία.

Έτσι αν έχουμε σε μία εντολή (SQL command) δύο δηλώσεις UPDATE που εκτελούν μεταφορά χρημάτων ανάμεσα σε τραπεζικούς λογαριασμούς θα πρέπει το πρόγραμμά μας (το λογισμικό εφαρμογής) να κάνει έλεγχο των διαγνωστικών μηνυμάτων που επιστρέφει ο εξυπηρετητής. Ανάλογα με τα μηνύματα μπορεί να αποφασίσει αν το αίτημα μπορεί να εξυπηρετηθεί ή όχι.

Παράδειγμα

Ο χρήστης αιτείται τη μεταφορά χρημάτων. Η εντολή SQL αναλαμβάνει να διεκπεραιώσει τη μεταφορά χρημάτων. Η εντολή αποτελείται από δύο δηλώσεις και άλλη προγραμματιστική λογική (if then …) διαχείρισης μηνυμάτων του εξυπηρετητή.

SQL command

```
……..
UPDATE accounts
SET balance=balance-500
WHERE account_number=12345;
…….
UPDATE accounts
SET balance=balance+500
WHERE account_number=23456;
……..
```

Αν μία δήλωση UPDATE (μέρος της εντολής SQL μεταφοράς χρημάτων) εφαρμόζεται σε έναν ανύπαρκτο λογαριασμό (π.χ., ο λογαριασμός 23456 δεν υπάρχει) τότε το λογισμικό εφαρμογής εξετάζει το μήνυμα του εξυπηρετητή, «κατανοεί» ότι η δήλωση UPDATE εκτελέστηκε «σωστά» αλλά βέβαια πρέπει επίσης να «καταλάβει» ότι δεν έγινε μεταβολή στοιχείων. Τότε, πρέπει να ακυρώσει τη συναλλαγή μεταφοράς χρημάτων και να στείλει κατάλληλο μήνυμα στον χρήστη.

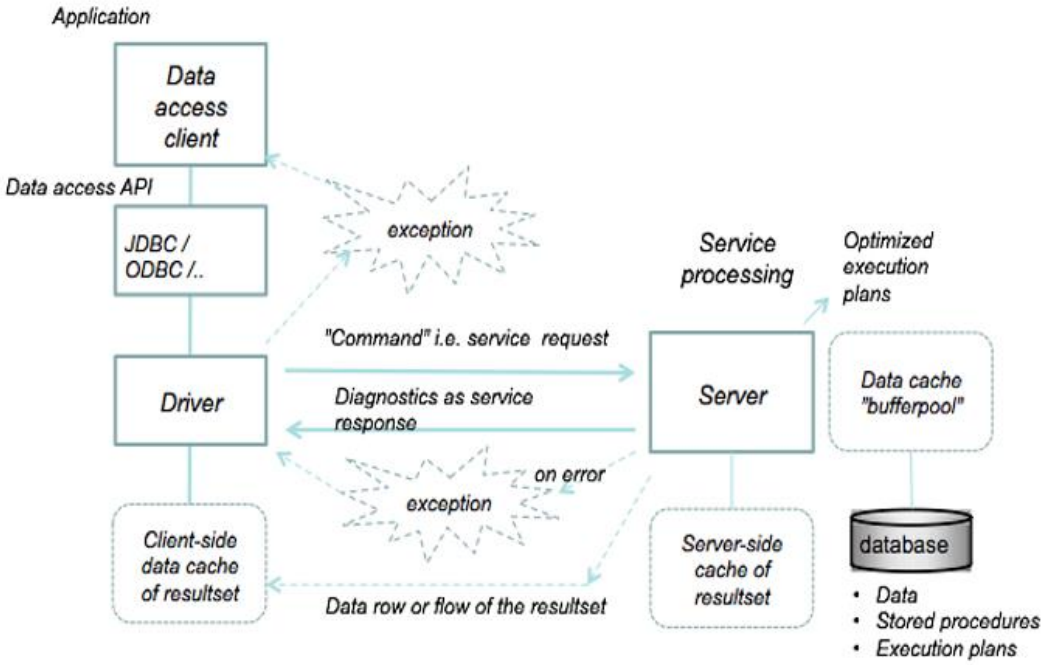
Στην περίπτωση μιας δήλωσης SELECT ο πελάτης ανακτά γραμμή-γραμμή τα αποτελέσματα (resultset).

Το ορθογώνιο παραλληλόγραμμο στο οποίο είναι γραμμένο JDBC/ODBC απεικονίζει την ιδέα της εξυπηρέτησης ενός αιτήματος πελάτη. Το αίτημα υπηρεσίας εξυπηρετείται από τις λειτουργίες/μεθόδους που περιλαμβάνει μία προγραμματιστική διεπαφή DATA ACCESS API (Application Programming Interface), π.χ. η διεπαφή JDBC. Πιο συγκεκριμένα, ο πελάτης αιτείται τη σύνδεση ή την εγγραφή του στην εφαρμογή. Η διεπαφή API παρέχει μεθόδους στο λογισμικό εφαρμογής που χρησιμοποιούνται για τη δημιουργία σύνδεσης (connection), για τη συνεδρία (session), την εκτέλεση (query execution) των δηλώσεων SQL κλπ. Όλη αυτή η λειτουργικότητα εξυπηρετείται από τον οδηγό (driver), π.χ. το πρόγραμμα οδήγησης MySQL για τη διεπαφή JDBC.

Ακολουθεί μια σύνοψη:

1) Σύνδεση στη βάση δεδομένων ή SQL-session που αρχικοποιείται από το πρόγραμμα-πελάτη
2) Υπηρεσίες DB (DB services) που χρησιμοποιούνται για να περαστούν οι SQL εντολές ως παράμετροι για τις συναρτήσεις/μεθόδους της διεπαφής API
3) Μια εντολή SQL (SQL command) μπορεί να περιλαμβάνει περισσότερες από μία δηλώσεις SQL (SQL statements)
4) Οι SQL εντολές εκτελούνται σε ατομική μορφή (σαν μονάδα) στον εξυπηρετητή της βάσης δεδομένων
5) Διάλογος client-server σε λογικό επίπεδο, χρησιμοποιώντας τη σύνταξη της γλώσσας SQL

Οι εξαιρέσεις (exceptions) που δημιουργούνται από τον εξυπηρετητή της βάσης και τα διαγνωστικά (diagnostics) διευκολύνουν την υποστήριξη των συναλλαγών των χρηστών σε επίπεδο λογισμικού εφαρμογής



Εικόνα 2.15 Κύκλος επεξεργασίας SQL εντολής (command)

Οι εικόνες 2.14 και 2.15 της ενότητας, η περιγραφή και η συζήτησή τους δημοσιεύονται στα παρακάτω άρθρα.

Dervos, D. A., Skourlas, C, Laiho, M. (2015), A DBTechNet course module on database SQL transactions for VET teachers training and higher education informatics education, AIP Conference Proceedings, Volume 1644, Issue 1, p.147-152

Laiho, M., Skourlas, C., Dervos, D. A. (2015), Zero tolerance for incorrect data: Best practices in SQL transaction programming, AIP Conference Proceedings, Volume 1644, Issue 1, p.113-119

Skourlas, C., Dervos, D. A., Laiho, M. (2015) Database SQL transactions and learning by verifying in practice: a case study, PCI '15 Proceedings of the 19th Panhellenic Conference on Informatics, ACM Digital Library, pp. 361-362

2.8.2 Προβλήματα ταυτοχρονισμού (Concurrency problems, anomalies)

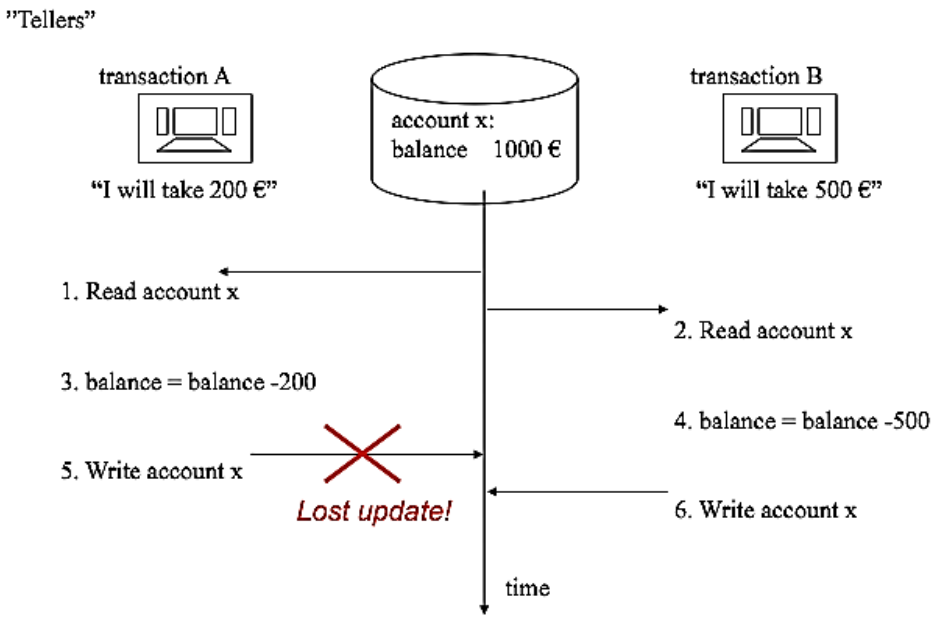
Η ταυτόχρονη εκτέλεση συναλλαγών συνδέεται με τέσσερα προβλήματα:

1) το πρόβλημα της χαμένης ενημέρωσης (lost update)

   Στην εικόνα 2.16 βλέπουμε παράδειγμα εμφάνισης του προβλήματος της χαμένης ενημέρωσης κατά την ταυτόχρονη εκτέλεση δύο τραπεζικών συναλλαγών στον ίδιο λογαριασμό (account x). Η συναλλαγή Α κάνει ανάληψη 200 ευρώ και η δεύτερη κάνει ανάληψη 500 ευρώ. Η σειρά εκτέλεσης των βημάτων αριθμείται στην εικόνα 2.16.

   Πρώτα εκτελείται η ενέργεια 1, δηλαδή το πρώτο το βήμα της συναλλαγής Α ("Read account x"). Στη συνέχεια εκτελείται η ενέργεια 2, δηλαδή το πρώτο βήμα της συναλλαγής Β ("Read account x"). Στη συνέχεια εκτελείται η ενέργεια 3 , δηλαδή το δεύτερο βήμα της συναλλαγής Α ("balance=balance-200 x")

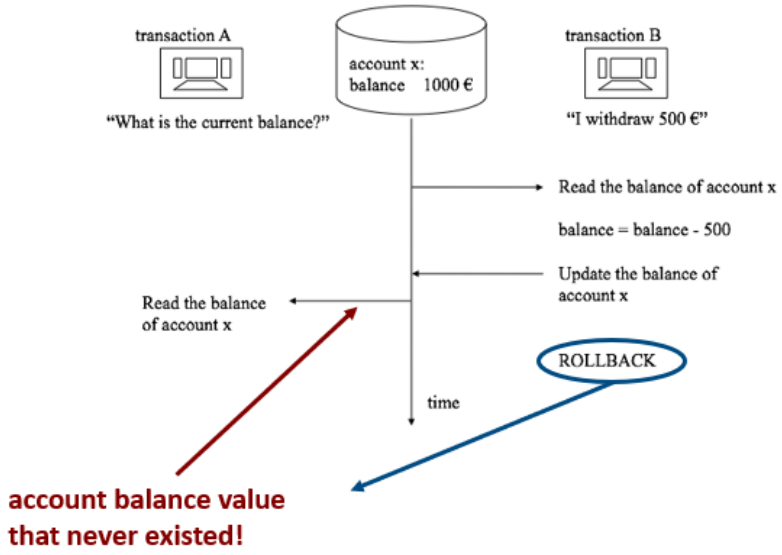
   Προσοχή! Όλα τα DBMS μας προστατεύουν από το «lost update problem».



Εικόνα 2.16 Εμφάνιση του προβλήματος της χαμένης ενημέρωσης σε τραπεζικές συναλλαγές (M Laiho 2013)

2) **το πρόβλημα της πρόχειρης ανάγνωσης (dirty read),** δηλαδή η ανάγνωση δεδομένων η εγκυρότητα των οποίων δεν έχει ακόμη επικυρωθεί από τις ταυτόχρονα εκτελούμενες συναλλαγές που τα έχουν καταχωρήσει στη βάση

Στην εικόνα 2.17 βλέπουμε παράδειγμα εμφάνισης του προβλήματος της πρόχειρης ανάγνωσης κατά την ταυτόχρονη εκτέλεση δύο τραπεζικών συναλλαγών στον ίδιο λογαριασμό (account x). Η συναλλαγή Α διαβάζει το υπόλοιπο λογαριασμού και η συναλλαγή Β κάνει ανάληψη 500. Η σειρά εκτέλεσης των βημάτων φαίνεται στην εικόνα 2.17.



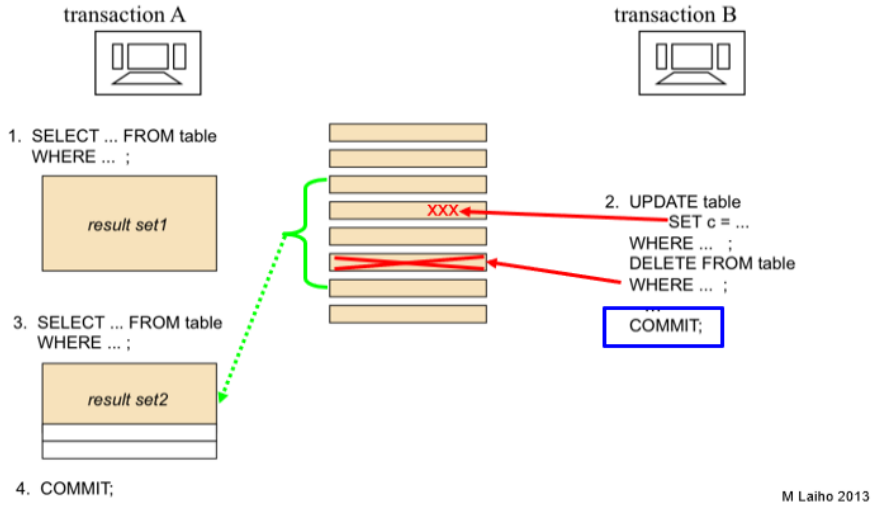
Εικόνα 2.17 Εμφάνιση του προβλήματος της πρόχειρης ανάγνωσης σε τραπεζικές συναλλαγές (M Laiho 2013)

3) **το πρόβλημα της μη-επαναλήψιμης ανάγνωσης (non-repeatable read)**, δηλαδή περιπτώσεις όπου διαδοχικές αναγνώσεις με το ίδιο κριτήριο αναζήτησης δεν επιστρέφουν τις ίδιες πλειάδες/γραμμές στο αποτέλεσμα

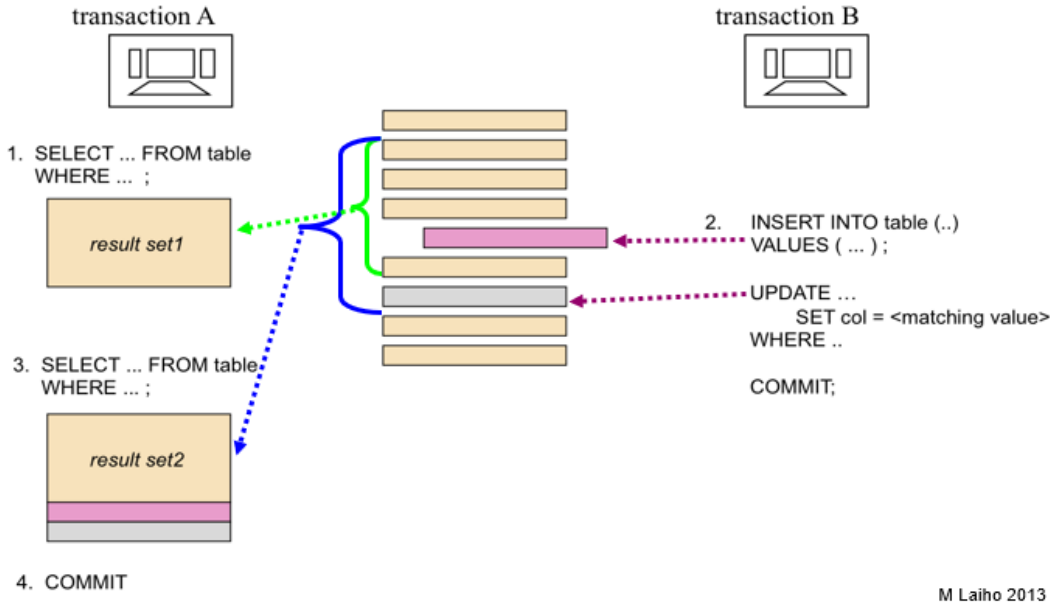
Στην εικόνα 2.18 βλέπουμε παράδειγμα εμφάνισης του προβλήματος τη μη-επαναλήψιμης ανάγνωσης

4) **το πρόβλημα ανάγνωσης φαντάσματος (phantom read problem)**, δηλαδή, κατά την εκτέλεση της συναλλαγής μερικές από τις πλειάδες οι οποίες θα έπρεπε να συνυπολογιστούν στην επεξεργασία εξαιρούνται γιατί η συναλλαγή δεν αισθάνεται την ύπαρξή τους.

Στην εικόνα 2.19 βλέπουμε παράδειγμα εμφάνισης του προβλήματος ανάγνωσης φαντάσματος.



Εικόνα 2.18 Εμφάνιση του προβλήματος της μη-επαναλήψιμης ανάγνωσης σε τραπεζικές συναλλαγές(M Laiho 2013)



Εικόνα 2.19 Εμφάνιση του προβλήματος της ανάγνωσης φαντάσματος σε τραπεζικές συναλλαγές (M Laiho 2013)

2.8.3 Τα επίπεδα απομόνωσης

Η αντιμετώπιση των προβλημάτων γίνεται με την εκτέλεση των συναλλαγών με το κατάλληλο επίπεδο απομόνωσης. Στον πίνακα 2.3 βλέπουμε πως πρέπει να οριστεί το επίπεδο απομόνωσης για να αποφύγουμε κάθε ένα από τα τέσσερα προβλήματα.

Πίνακας 2.3 Ορισμός επίπεδου απομόνωσης και αποφυγή προβλημάτων ταυτόχρονης εκτέλεσης συναλλαγών

Anomalies: / Isolation Level:	Lost Update	Dirty Read	Nonrepeatable Read	Phantoms
READ UNCOMMITTED	NOT possible	Possible !	Possible !	Possible !
READ COMMITTED	NOT possible	NOT possible	Possible !	Possible !
REPEATABLE READ	NOT possible	NOT possible	NOT possible	Possible !
SERIALIZABLE	NOT possible	NOT possible	NOT possible	NOT possible

SET TRANSACTION Syntax

```
SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL
  {
   REPEATABLE READ
 | READ COMMITTED
 | READ UNCOMMITTED
 | SERIALIZABLE
   }
```

Παραθέτουμε παράδειγμα,

```
SET TRANSACTION ISOLATION LEVEL <isolation level>
```

Η σύνταξη στο προϊόν IBM DB2 διαφέρει: SET CURRENT ISOLATION = <isolation level>

Η διεπαφή JDBC API «βλέπει» μόνο τα ονόματα των επιπέδων απομόνωσης του προτύπου ISO SQL. Ακολουθεί παράδειγμα σε JDBC:

<connection>.setTransactionIsolation(Connection.<TRANSACTION_SERIALIZABLE>);

2.8.4 Ιδανική συναλλαγή, ιδιότητες ACID (ACID principle)

Οι ιδιότητες ACID παρουσιάστηκαν για πρώτη φορά με το όνομα «η αρχή ACID» (ACID principle) από τους Theo Härder και Andreas Reuter με άρθρο τους στο περιοδικό ACM Computing Surveys το 1983. Μέσω της εν λόγω αρχής, οι συγγραφείς του άρθρου ορίζουν το ιδανικό για την αξιόπιστη εκτέλεση των συναλλαγών SQL σε πολυχρηστικό υπολογιστικό περιβάλλον. Πρόκειται για ακροστιχίδα που συνιστούν τα αρχικά των επόμενων τεσσάρων ιδιοτήτων των συναλλαγών (στα Αγγλικά).

Δείτε σχετικά το άρθρο,

Theo Härder & Andreas Reuter (1983) Principles of Transaction-Oriented Database Recovery, Computing Surveys, vol. 15, no. 4

Στον πίνακα 2.4 βλέπουμε τις ιδιότητες ACID με σύντομη επεξήγηση.

Πίνακας 2.4 Ιδιότητες ACID

Ατομικότητα (Atomicity)	Πρέπει να καταγραφούν όλες οι αλλαγές στη βάση δεδομένων που επιφέρει η συναλλαγή ή να αναιρεθούν όλες.
Συνέπεια (Consistency)	Η εκτέλεση μιας συναλλαγής «απομονωμένα» (δηλαδή, χωρίς ταυτόχρονη εκτέλεση άλλων συναλλαγών) διατηρεί τη συνέπεια της βάσης δεδομένων, δηλαδή διασφαλίζεται η ισχύς των κανόνων ακεραιότητας και των περιορισμών στη βάση δεδομένων
Απομόνωση (Isolation)	Ακόμα και αν εκτελούνται ταυτόχρονα πολλές συναλλαγές το σύστημα εγγυάται ότι για κάθε ζευγάρι από συναλλαγής Ti και Tj, θα φαίνεται στην Ti ότι η Tj τελείωσε την εκτέλεση πριν ξεκινήσει η Ti ή ότι η Tj ξεκίνησε την εκτέλεση αφού τελείωσε η Ti. Έτσι, κάθε συναλλαγής δεν ξέρει για τις άλλες συναλλαγής που εκτελούνται ταυτόχρονα στο σύστημα.
Ανθεκτικότητα (Durability)	Αφού μια συναλλαγής ολοκληρωθεί με επιτυχία παραμένουν οι αλλαγές που έχει κάνει στη βάση δεδομένων, ακόμα και αν το σύστημα έχει πρόβλημα (system/soft crash).

Στον πίνακα 2.5 βλέπουμε παραδείγματα παραβίασης των ιδιοτήτων ACID.

Πίνακας 2.5 Παραδείγματα παραβίασης των ιδιοτήτων ACID

Atomicity	Μια συναλλαγή αφαιρεί 100 ευρώ από τον λογαριασμό 101 και στη συνέχεια δεν μπορεί να μεταφέρει αυτό το ποσό στον λογαριασμό 201.
Consistency	Η συνέπεια μιας συναλλαγής απαιτεί τα δεδομένα να πληρούν όλους τους κανόνες επικύρωσης (validation rules). Ένας τραπεζικός λογαριασμός με αρνητικό υπόλοιπο αποτελεί παραβίαση. Πρόβλημα παρουσιάζεται και στην περίπτωση παραβίασης της ακεραιότητας αναφοράς (Referential integrity). Αν δεν ορίσουμε μηχανισμό πρωτεύοντος-ξένου κλειδιού μπορούμε να εισάγουμε νέο λογαριασμό ο οποίος δεν αντιστοιχεί σε πελάτη.
Isolation	Δείτε όλα τα παραδείγματα των προβλημάτων ταυτόχρονης χρήσης (Concurrency problems) της ενότητας. Τα προβλήματα προκαλούνται όταν ορίζουμε συναλλαγές με μη επαρκές επίπεδο απομόνωσης (isolation level)
Durability	Ας υποθέσουμε ότι μια συναλλαγή μεταφέρει 100 ευρώ από τον λογαριασμό 101 στον λογαριασμό 201 και στέλνεται ένα μήνυμα "επιτυχίας" στον πελάτη. Στη συνέχεια, οι αλλαγές χάνονται από εκτέλεση δήλωσης ROLLBACK η οποία προκαλείται από αποτυχία τροφοδοσίας (Power fail)

Οι βάσεις δεδομένων NoSQL δεν χρειάζονται μια σταθερή δομή πίνακα και δεν παρέχουν πλήρη υποστήριξη των ιδιοτήτων ACID. Για παράδειγμα, τελικά συνέπεια στην περίπτωση βάσης NoSQL μπορεί να σημαίνει ότι τα δεδομένα θα είναι συνεπή για μια χρονική περίοδο (Orend, 2010). Οι βάσεις δεδομένων NoSQL θα περιγραφούν σε επόμενη ενότητα του κεφαλαίου 2.

Συμπέρασμα

Η επεξεργασία δεδομένων μεγάλης κλίμακας συνήθως έχει ένα υβριδικό χαρακτήρα και περιλαμβάνει τουλάχιστον δύο συνιστώσες:

1) Πλατφόρμα διαχείρισης συναλλαγών για την εκτέλεση εφαρμογών υψηλής κλίμακας, σε πραγματικό χρόνο. Η βάση δεδομένων των λειτουργικών δεδομένων (operational data) της επιχείρησης είναι συμβατή με το πρότυπο ACID, δηλαδή υποστηρίζει τις ιδιότητες για ατομικότητα, συνέπεια, απομόνωση και ανθεκτικότητα. Στις εφαρμογές μας θα πρέπει να χρησιμοποιήσουμε τα κατάλληλα επίπεδα απομόνωσης για την ασφάλεια των συναλλαγών.
2) Περιλαμβάνει μια Hadoop πλατφόρμα η οποία χειρίζεται τον όγκο, την ποικιλία και την πολυπλοκότητα των μη δομημένων δεδομένων που ενδιαφέρουν την επιχείρηση.

2.9 Το παρόν και το μέλλον των εφαρμογών βάσεων δεδομένων. Ο ρόλος του σημασιολογικού ιστού.

Η σχεσιακή βάση (relational data base) είναι η βάση δεδομένων που έχει τα χαρακτηριστικά που αναφέρθηκαν και συζητήθηκαν στο κεφαλαίο 1: τα στοιχεία της είναι οργανωμένα υπό τη μορφή πινάκων (tables) και είναι διαχειρίσιμη με μία σειρά από πράξεις όπως Επιλογή γραμμών πίνακα, Προβολή στηλών πίνακα, Σύνδεση Πινάκων ή απλά (όπως λέγεται στο χώρο των εταιριών πληροφορικής) είναι διαχειρίσιμη με τη γλώσσα SQL.

Οι γραμμές του πίνακα καλούνται, συχνά, εγγραφές (records) και οι στήλες πεδία (fields).

- Ένα ευρετήριο (index) είναι μια λίστα δεικτών που συνδέονται με τις γραμμές-εγγραφές κάποιου πίνακα της βάσης. Το ευρετήριο επιταχύνει την ανάκτηση των δεδομένων.
- Το ερώτημα (query, δήλωση SELECT) είναι ένας τρόπος αναζήτησης («ψαξίματος») στοιχείων της βάσης. Συχνά είναι αποθηκευμένο και εκτελείται περιοδικά.
- Οι ηλεκτρονικές φόρμες (forms) είναι μια σειρά από φόρμες που διαχειριζόμαστε στον υπολογιστή. Υποστηρίζονται από προγράμματα που κατασκεύασε ένας προγραμματιστής και επιτρέπουν σε έναν τελικό χρήστη να διαχειριστεί τη βάση δεδομένων με εύκολο, φιλικό τρόπο.

Οι σχεσιακές βάσεις δεδομένων είναι ιδιαίτερα δημοφιλείς στον κόσμο των Πληροφοριακών Συστημάτων. Ειδικά, για τις εφαρμογές βάσεων δεδομένων στο διαδίκτυο πρέπει να επισημάνουμε ότι πάρα πολλές χρησιμοποιούν και ειδικού τύπου λογισμικό, τα συστήματα ανάκτησης πληροφοριών-ΣΑΠ (IR-Information Retrieval) και τα συστήματα διαχείρισης κειμένου (text retrieval):

- Οι κατάλογοι ανοικτής πρόσβασης (Open Access Catalogues) των βιβλιοθηκών, οι βιβλιογραφικές βάσεις δεδομένων, οι βάσεις πλήρους κειμένου, οι ψηφιακές βιβλιοθήκες (digital libraries) κ.λπ. βασίζουν λειτουργίες τους σε ΣΑΠ.
- Η "ιδέα" των μηχανών αναζήτησης στο διαδίκτυο (search engines) είναι πολύ κοντά στις "ιδέες" και τη λογική του ΣΑΠ. Μία τυπική μηχανή αναζήτησης καταχωρεί, για τις σελίδες που ενδιαφέρουν, εγγραφές σε ειδικά σχεδιασμένα (αντεστραμμένα) ευρετήρια έτσι ώστε πολύ γρήγορα να μπορεί να ικανοποιήσει αιτήματα χρηστών που "ψάχνουν" για πληροφορίες (σελίδες) με λέξεις κλειδιά, με ονόματα προσώπων, εταιριών κ.λπ.

Βέβαια, υπάρχει και έντονη δραστηριότητα αλληλεπίδρασης ΣΑΠ και ΣΔΒΔ και ενσωμάτωσης στοιχείων του ενός στο άλλο. Για παράδειγμα, όλα τα πακέτα αυτοματισμού βιβλιοθήκης και ΣΑΠ αναθέτουν πλέον τη διαχείριση των αρχείων σε κάποιο ΣΔΒΔ, πχ. Oracle ενώ τα γνωστά ΣΔΒΔ έχουν συνιστώσα ανάκτησης κειμένου (text retrieval) κ.λπ.

Νεότερες έννοιες, καινοτομίες, νεότερα προϊόντα και η έρευνα προηγούμενων ετών πρόσθεσε και προσθέτει πάρα πολλά νέα στοιχεία στην παραπάνω εικόνα της ενότητας:

Σημασιολογικός ιστός (Semantic Web), Συστήματα Ανάκτησης Πληροφορίας και Μηχανές Αναζήτησης, Πολυτροπικές (multimodal) βάσεις δεδομένων, Γλώσσες Σήμανσης, Οντολογίες, Βάσεις Δεδομένων στο Νέφος (cloud databases), Δεδομένα Μεγάλης Κλίμακας (Big Data) είναι τεχνολογίες σημαντικές στο παρόν και πολλά υποσχόμενες για το μέλλον. Στη συνέχεια θα αναφερθούμε συνοπτικά στις τεχνολογίες αυτές.

2.9.1 Σημασιολογικός ιστός και η επίδρασή του

Ο παγκόσμιος ιστός και ο σημασιολογικός ιστός επηρέασαν και επηρεάζουν καθοριστικά τις ζωές μας, τον κόσμο των επιχειρήσεων και των αγορών. Επιπρόσθετα, οι τεχνολογίες τους συνεχίζουν να αλλάζουν τη διαχείριση δεδομένων (data(base) management), τη διαχείριση πληροφορίας (information management) και

τη διαχείριση γνώσης (knowledge management). Τα ΣΔΒΔ, άλλαξαν και αλλάζουν για να εκμεταλλευθούν τις νέες ευκαιρίες και δυνατότητες που παρέχονται από τον σημασιολογικό ιστό.

Πέρα από τα ΣΔΒΔ και όλες οι υπόλοιπες τεχνολογίες και τα εργαλεία που χρησιμοποιούν οι επιχειρήσεις, π.χ., ΣΑΠ για την ανάκτηση πληροφορίας, συστήματα ανάκτησης εικόνας βασιζόμενα στο περιεχόμενο (CBIR-Content Based Image Retrieval), συστήματα ανάκτησης στοιχείων από πολυμεσικές βάσεις δεδομένων (multimodal database), η εξόρυξη δεδομένων (data mining), η αποθήκη δεδομένων (data warehouse), η επιχειρηματική ευφυία (business intelligence), αλλάζουν κάτω από την επίδραση της τεχνολογίας του παγκόσμιου ιστού και του σημασιολογικού ιστού. Τα δεδομένα μεγάλης κλίμακας (big data) και οι αναδυόμενες τεχνολογίες που συνδέονται με τη διαχείρισή τους, η τεχνητή νοημοσύνη (artificial intelligence) και η μηχανική μάθηση (machine learning) εισάγουν καινοτομικά στοιχεία, τα οποία φαίνεται ότι μπορούν να αλλάξουν τη ζωή των φυσικών προσώπων, τον τρόπο λειτουργίας των επιχειρήσεων αλλά και των αγορών.

Ένα κεντρικό προαπαιτούμενο για οποιοδήποτε λογισμικό εφαρμογής σημασιολογικού ιστού (Semantic Web) είναι η συμμόρφωση με μία απαίτηση: η σημασιολογία της πληροφορίας μπορεί να είναι κατανοητή από άνθρωπο και μηχανή. Στο πλαίσιο του σημασιολογικού ιστού οι εφαρμογές μας διαχειρίζονται δεδομένα, πληροφορία και γνώση λαμβάνοντας υπόψη το περιεχόμενο και την οργάνωσή τους.

Στις μέρες μας υπάρχουν δημοφιλείς μορφότυποι εξίσου κατανοητοί από άνθρωπο και μηχανή. Στη συνέχεια βλέπουμε παράδειγμα με το οποίο μπορούμε να κατανοήσουμε καλύτερα την έννοια της κατανόησης (understanding) από άνθρωπο και μηχανή.

Παράδειγμα μορφότυπου JSON κατανοητού από άνθρωπο και μηχανή

Ο μορφότυπος JSON (JavaScript Object Notation) γνωρίζει ευρύτατη εφαρμογή σήμερα. Ακολουθεί παράδειγμα JSON αναπαράστασης (representation) ενός προσώπου (Wikipedia,

https://en.wikipedia.org/wiki/JSON).

```
{
        "firstName": "John",
        "lastName": "Smith",
        "isAlive": true,
        "age": 27,
        "address": {
      "streetAddress": "21 2nd Street",
      "city": "New York",
      "state": "NY",
      "postalCode": "10021-3100"
        },
        "phoneNumbers": [
      {
        "type": "home",
        "number": "212 555-1234"
      },
      {
        "type": "office",
        "number": "646 555-4567"
      }
        ],
        "children": [],
        "spouse": null
      }
```

Δείτε και το παράδειγμα στις σελίδες 21-23 (“a purchase order”) του εγχειριδίου, Oracle® Database JSON Developer’s Guide, 21c, F30948-06, April 2022

JSON Developer’s Guide (oracle.com)

Σχετικές τεχνολογίες με το σημασιολογικό ιστό περιλαμβάνουν γλώσσες σήμανσης, π.χ. XML, την οικογένεια προδιαγραφών του W3C Resource Description Framework (RDF), για την εννοιολογική περιγραφή πόρων του διαδικτύου (web resources), τις οντολογίες (Ontology) και τα εργαλεία συγγραφής τους, π.χ., Web Ontology Language (OWL), οικογένεια γλωσσών (knowledge representation languages) για συγγραφή (authoring) οντολογιών κ.λπ.

Λέξεις κλειδιά για το αντικείμενο της ενότητας αυτής (κάποιες γνωστές από τη δεκαετία του 90), και άλλες πιο πρόσφατες, είναι: Οντολογίες, XML, RDF, RDFS, Dublin Core, JASON κ.λπ.

Ειδικότερα, στην ενότητα αυτή σκιαγραφείται ο σύγχρονος τομέας των συστημάτων βάσεων δεδομένων και των εφαρμογών τους όπως διαμορφώνεται από την επίδραση του σημασιολογικού ιστού. Αρχικά η ενότητα εστιάζει στα ανοικτά και συνδεδεμένα δεδομένα και αναλύεται η έννοια των μεταδεδομένων, τα σχήματα μεταδεδομένων και η σημασία τους. Στη συνέχεια επεξηγούνται έννοιες του σημασιολογικού ιστού (semantic web) όπως τα ενιαία αναγνωριστικά ονόματα (URIs), οι γλώσσες σήμανσης, π.χ. γλώσσες XML, HTML, η οικογένεια προτύπων RDF, Ontology, OWL, ελεγχόμενα λεξιλόγια (controlled vocabularies) κ.λπ.

Τέλος, εξετάζουμε πως η οργάνωση δεδομένων μεγάλης κλίμακας (big data) επηρεάζει το σημασιολογικό ιστό. Θυμίζουμε ότι ο όρος δεδομένα μεγάλης κλίμακας περιγράφει μεγάλους όγκους δεδομένων οι οποίοι προκύπτουν με υψηλή ταχύτητα, και αποτελούν σύνθετα και μεταβλητά ετερογενή δεδομένα, δομημένα (structured), ημιδομημένα (semi-structured) και αδόμητα (unstructured). Τα μεγάλα δεδομένα, επιπρόσθετα από τις «παραδοσιακές» τεχνικές διαχείρισης δεδομένων, απαιτούν επιπλέον προηγμένες τεχνικές και τεχνολογίες για να καταστεί δυνατή η συλλογή, αποθήκευση, διανομή, διαχείριση και ανάλυση των πληροφοριών. Γενικά υπάρχει η αντίληψη στην ερευνητική κοινότητα και στον κόσμο των επιχειρήσεων ότι τα μεγάλα δεδομένα προσφέρουν μια νέα οπτική γωνία-αντίληψη σε όλες τις έννοιες και σε όλες τις τεχνολογίες που εμπλέκονται στον κύκλο ζωής των πληροφοριακών συστημάτων και στον κύκλο ζωής του λογισμικού.

2.9.1.1 Ανοιχτά δεδομένα

Σύμφωνα με το *Open Data Handbook* (2021):

Ανοιχτά είναι τα δεδομένα που μπορούν ελεύθερα να χρησιμοποιηθούν, να επα*ναχρησιμοποιηθούν και να αναδιανεμηθούν από οποιονδήποτε – υπό τον όρο να γίνεται αναφορά στους δημιουργούς και να διατίθενται, με τη σειρά τους, υπό τους ίδιους όρους (Open data is data that can be freely used, re-used and redistributed by anyone − subject only, at most, to the requirement to attribute and share alike).*

The Open Data Handbook

Τα ανοικτά δεδομένα πρέπει να είναι (Μαρινάγη & Σκουρλάς, 2022):

- Προσβάσιμα και διαθέσιμα για λήψη με λογικό κόστος.
- Επαναχρησιμοποιήσιμα. Πρέπει να επιτρέπεται η αναδιανομή τους χωρίς περιορισμούς. Για παράδειγμα, περιορισμοί για «μη εμπορική χρήση» ή περιορισμοί για χρήση μόνο για εκπαιδευτικούς σκοπούς δεν είναι επιτρεπτοί αν αναφερόμαστε σε ανοικτά δεδομένα.
- Επιπλέον, πρέπει να διασφαλίζεται η διαλειτουργικότητα, δηλαδή τα ανοιχτά σύνολα δεδομένων διαφορετικών συστημάτων (ή συνιστωσών τους) πρέπει να μπορούν να «αναμειγνύονται» (να συνδυάζονται) και τα συστήματα να λειτουργούν μαζί (να διαλειτουργούν). Με τον τρόπο αυτό

επιτρέπεται η δόμηση μεγαλύτερων και πιο πολύπλοκων συστημάτων και υποβοηθείται η ανάπτυξη καλύτερων προϊόντων και υπηρεσιών.

Είναι προφανές ότι στα ανοιχτά δεδομένα δεν περιλαμβάνονται προσωπικά δεδομένα και κυβερνητικά δεδομένα που αφορούν την εθνική ασφάλεια.

2.9.1.2 Συνδεδεμένα δεδομένα (linked data)

Το όραμα του Tim Berners-Lee και το αντικείμενο του Linking Open Data Community Project είναι η αξιοποίηση των μορφοτύπων δεδομένων (data formats) του σημασιολογικού ιστού έτσι ώστε να καταστεί δυνατή:

1) η κοινή χρήση του περιεχομένου των βάσεων δεδομένων,
2) η σύνδεση των βάσεων μεταξύ τους και τελικά
3) η δημιουργία ενός ιστού συνδεδεμένων δεδομένων που θα εκτείνεται σε ολόκληρο τον πλανήτη.

Σε αντίθεση με τον ιστό των συνδεδεμένων εγγράφων (linked documents),

«ο ιστός των συνδεδεμένων δεδομένων θα επιτρέψει στους εκδότες να περιγράψουν μοντέλα δεδομένων, έννοιες δεδομένων και αρχεία δεδομένων με τέτοιο τρόπο ώστε τα δεδομένα να μπορούν να συνδεθούν, να περιγραφούν και να ανακτηθούν σαν να ήταν μέρος μιας ενιαίας βάσης δεδομένων»

(“the Web of linked data will allow publishers to describe data models, data concepts, and data records in such a way that they can be linked, described, and queried as if they were part of a single database”) (Pollock, 2009, σελ.13).

Το Linking Open Data project μπορεί να θεωρηθεί και ως πλαίσιο για την εφαρμογή των αρχών των συνδεδεμένων δεδομένων σε μεγάλα ανοιχτά σύνολα δεδομένων και ως ένα φόρουμ για την εκτέλεση εργασιών που αξιοποιούν την ύπαρξη πολλών συνδεδεμένων δεδομένων στον Ιστό.

2.9.1.3 Μεταδεδομένα και σχήματα μεταδεδομένων στον σημασιολογικό ιστό

Τα μεταδεδομένα συχνά ορίζονται ως «δεδομένα για τα δεδομένα» και περιλαμβάνουν πληροφορίες που επιτρέπουν στους χρήστες (ανθρώπους ή υπολογιστές) να εντοπίζουν δεδομένα, έγγραφα, αναγραφές, πόρους κ.λπ. και να διαχειρίζονται το περιεχόμενό τους.

Γνωρίζουμε την έννοια των μεταδεδομένων ήδη από τις σχεσιακές βάσεις δεδομένων. Η λογική σχεδίασή τους συνδέεται με τη διαδικασία της κατασκευής σχήματος της βάσης (database schema), πινάκων (tables) και σχέσεων-συσχετίσεων (relationships) μεταξύ των πινάκων οι οποίες περιγράφονται με περιορισμούς για τα πρωτεύοντα (κύρια) και τα ξένα κλειδιά. Μπορούν, επίσης, να οριστούν και περισσότερα είδη δομικών περιορισμών (structural assignments/constraints) του περιεχομένου, δηλαδή των δεδομένων της βάσης. Οι περιορισμοί αυτοί είναι μεταδεδομένα της βάσης δεδομένων και επομένως τα δεδομένα αποθηκεύονται στη βάση και προβάλλονται στον χρήστη στο πλαίσιο (context) αυτών των περιγραφικών μεταδεδομένων (descriptive metadata). Ακολουθεί παράδειγμα ορισμού πίνακα.

```
CREATE TABLE Students(
Student_ID int NOT NULL,
Student_Surname varchar(255) NOT NULL, Student_Name varchar(255) NOT NULL,
-συσχετίσεωνClass_Name varchar(255),
Age int CHECK(Age >18),
PRIMARY KEY (Student_ID)
FOREIGN KEY (Class_Name) References classes(Class_Name)
);
```

Σύμφωνα με τον Tim Berners-Lee: «Metadata is machine understandable information about web resources or other things».

Δηλαδή, με τον όρο μεταδεδομένα (metadata) αναφερόμαστε σε δεδομένα που χρησιμοποιούνται για προσδιορισμό, περιγραφή ή εντοπισμό πληροφορίας που υπάρχει σε ηλεκτρονικούς ή μη πόρους (resources). Τα μεταδεδομένα που μας ενδιαφέρουν, στο παρόν σύγγραμμα, είναι δομημένη πληροφορία που περιγράφει ένα ψηφιακό έγγραφο (document) ή γενικότερα έναν ψηφιακό πόρο (resource) και διευκολύνει την ανάκτησή του. Τα μεταδεδομένα υπάρχουν σε όλες τις ιστοσελίδες στον Ιστό. Το πρόγραμμα περιήγησης (browser) διαβάζει κωδικοποιημένο περιεχόμενο και μεταδεδομένα εντός της ιστοσελίδας, π.χ., ετικέτες (tag), λέξεις-κλειδιά (keyword), και εμφανίζει την ιστοσελίδα στον τελικό χρήστη.

Ο οργανισμός Dublin Core™ Metadata Initiative, DCMI, είναι η κοινότητα επιστημόνων-επαγγελματιών, οι οποίοι εργάζονται με μεταδεδομένα, και έχουν ως σκοπό την υποστήριξη προσπαθειών γεφύρωσης «καθιερωμένων πρακτικών μεταδεδομένων και λύσεων που βασίζονται σε γραφήματα για την ενσωμάτωση-ολοκλήρωση των δεδομένων (σε σιλό δεδομένων), στο πλαίσιο του οικοσυστήματος των συνδεδεμένων δεδομένων» (“established metadata practices and graph-based solutions for integration across data silos within a Linked Data ecosystem”).

Σύμφωνα με τον οργανισμό,

«Τα μεταδεδομένα, ειδικότερα τα περιγραφικά μεταδεδομένα, είναι δομημένα δεδομένα για οτιδήποτε μπορεί να ονομαστεί («περιγραφεί»), όπως ιστοσελίδες, βιβλία, άρθρα περιοδικών, εικόνες, τραγούδια, προϊόντα, διαδικασίες, άτομα και οι δραστηριότητές τους, ερευνητικά δεδομένα, έννοιες και υπηρεσίες».

(“Metadata, literally "data about data" -- specifically, descriptive metadata -- is structured data about anything that can be named, such as Web pages, books, journal articles, images, songs, products, processes, people (and their activities), research data, concepts, and services”.)

DCMI: Metadata Basics (dublincore.org)

Η χρήση μεταδεδομένων DCMI διασφαλίζει τη διαλειτουργικότητα κατά τρόπο συμβατό με τις αρχές του σημασιολογικού ιστού και των συνδεδεμένων δεδομένων επειδή τα μεταδεδομένα χρησιμοποιούν τα «ομοιόμορφα ή καθολικά αναγνωριστικά πόρων (Uniform Resource Identifiers) URI, ως καθολικά-μοναδικά-αναγνωριστικά (global identifiers), είτε περιγράφεται οποιοδήποτε αντικείμενο (πόρος) είτε αναφέρονται σε όρους (λεξιλόγια, vocabularies) που χρησιμοποιούνται για την περιγραφή του.

Η κοινότητα DCMI, από το 2000, εστιάζει στα «προφίλ εφαρμογής» (application profile), δηλαδή σε «μια προδιαγραφή η οποία περιγράφει λεπτομερώς πώς χρησιμοποιούνται, περιορίζονται ή συνδυάζονται τα εξειδικευμένα λεξιλόγια, όπως το Dublin Core, με γνωστά γενικά λεξιλόγια για την κάλυψη των απαιτήσεων συγκεκριμένων εφαρμογών» (a specification detailing how well-known generic vocabularies such as the Dublin Core are used, constrained, or combined with more specialized vocabularies to meet the requirements of specific applications)

Μία δημοφιλής μορφή («περιγραφή») των μεταδεδομένων χρησιμοποιεί την οικογένεια προτύπων Resource Description Framework (RDF). Για μία εισαγωγή στο πρότυπο μπορείτε να δείτε την αναφορά: RDF 1.1 Concepts and Abstract Syntax (w3.org)

Ακολουθεί ένα πρώτο παράδειγμα των μεταδεδομένων περιγραφής πόρου στον Παγκόσμιο Ιστό.

Παράδειγμα

Στην εικόνα 2.20 βλέπουμε μία RDF αναπαράσταση (περιγραφή) η οποία υπάρχει στο έγγραφο,

Eric Miller, Paul Miller and Dan Brickley (1999) Guidance on expressing the Dublin Core within the Resource Description Framework (RDF)-draft proposal, το οποίο αποτελεί μία απλή και ωραία εισαγωγή στο θέμα.

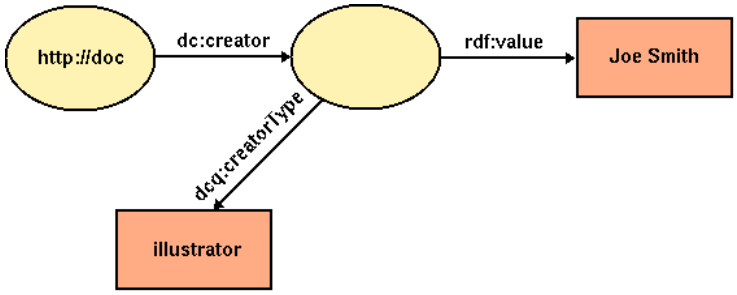
Guidance on expressing the Dublin Core within the Resource Description Framework (RDF) (ukoln.ac.uk)

Ο ενδιαφερόμενος αναγνώστης για μια επικαιροποιημένη αναφορά στο ζήτημα παραπέμπεται στον ιστότοπο του DCMI, π.χ., στο έγγραφο:

DCMI: Expressing Dublin Core™ metadata using the Resource Description Framework (RDF)

Στην εικόνα 2.20 βλέπουμε διαγραμματική RDF περιγραφή και συντακτική RDF αναπαράσταση της δήλωσης, "Joe Smith" is the illustrator of the resource http://doc.

Diagrammatic RDF Representation



Syntactic RDF Representation

```
<?xml version='1.0'?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc = "http://purl.org/dc/elements/1.0/"
 xmlns:dcq = "http://purl.org/dc/qualifiers/1.0/">
 <rdf:Description rdf:about = "http://doc">
  <dc:creator>
   <rdf:Description>
<rdf:value> Joe Smith </rdf:value>
<dcq:creatorType> Illustrator </dcq:creatorType>
   </rdf:Description>
  </dc:creator>
 </rdf:Description>
</rdf:RDF>
```

Εικόνα 2.20 Παράδειγμα RDF περιγραφών των Eric Miller, Paul Miller and Dan Brickley

Ακολουθεί αναλυτική περιγραφή του εγγράφου Guidance on expressing the Dublin Core within the Resource Description Framework (RDF) η οποία υπάρχει στο ίδιο το έγγραφο των Eric Miller, Paul Miller and Dan Brickley (1999). Στο παράδειγμα χρησιμοποιείται η γλώσσα XML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.0/"
 xmlns:dcq="http://purl.org/dc/qualifiers/1.0/">
  <rdf:Description
rdf:about="http://www.ukoln.ac.uk/metadata/resources/dc/datamodel/
WD-dc-rdf/">
<dc:title>
  <rdf:Alt>
```

```
<rdf:li xml:lang="en">Guidance on expressing the Dublin Core within the Resource
 Description Framework (RDF)</rdf:li>
<rdf:li xml:lang="no">Veiledning å uttrykke Dublin Core innenfor rammen av
 Resource Description Framework (RDF)</rdf:li>
<rdf:li xml:lang="de">Dublin Core in RDF: Eine Anleitung</rdf:li>
  </rdf:Alt>
</dc:title>
<dc:creator>
  <rdf:Bag>
<rdf:li>Eric Miller</rdf:li>
<rdf:li>Paul Miller</rdf:li>
<rdf:li>Dan Brickley</rdf:li>
  </rdf:Bag>
</dc:creator>
<dc:description>
  <rdf:Alt>
<rdf:li xml:lang="en">This document describes work carried out by the Data Model
 Working Group of the Dublin Core Metadata Initiative.
 Specifically, the document discusses means by which the fifteen elements of the
 Dublin Core (as defined in RFC 2413) may be expressed using the Resource
Description
 Framework (RDF) and encoded with the eXtensible Markup Language (XML). RDF-
based
 mechanisms by which the 15 elements may be qualified are also
introduced.</rdf:li>
<rdf:li xml:lang="no">Dette dokumentet beskriver arbeide utført av
arbeidsgruppen
 for datamodellering knyttet til Dublin Core-initiativet.
 Spesifikt diskuterer dokumentet hvordan de femten elementene i Dublin Core
(slik
 disse er definert i RFC 2413) kan uttrykkes ved hjelp av Resource Description
 Framework (RDF) og kodes ved hjelp av eXtensible Markup Language (XML). Videre
 introduseres RDF-baserte mekanismer for å kvalifisere de 15
elementene.</rdf:li>
  </rdf:Alt>
</dc:description>
<dc:subject> Dublin Core; Resource Description Framework; RDF; eXtensible
 Markup Language; XML </dc:subject>
<dc:publisher> Dublin Core Metadata Initiative </dc:publisher>
<dc:contributor> Dublin Core Data Model Working Group </dc:contributor>
<dc:date>
   <rdf:Description>
<dcq:dateScheme> WTN8601 </dcq:dateScheme>
<rdf:value> 1999-07-01 </rdf:value>
   </rdf:Description>
</dc:date>
<dc:format>
   <rdf:Description>
<dcq:formatScheme> IMT </dcq:formatScheme>
<rdf:value> text/html </rdf:value>
   </rdf:Description>
</dc:format>
<dc:language>
```

```
   <rdf:Description>
<dcq:languageScheme> RFC1766 </dcq:languageScheme>
<rdf:value> en </rdf:value>
   </rdf:Description>
</dc:language>
  </rdf:Description>
</rdf:RDF>
```

2.9.1.4 Μεταδεδομένα και μεγάλα δεδομένα

Τα μεγάλα δεδομένα συνδέονται με ενέργειες όπως "likes " στο Facebook, tweets, τηλεφωνικές κλήσεις σε helpdesk, κ.λπ. επομένως, πρέπει να επισημάνουμε ότι το ζήτημα "Big data" metadata απαιτεί ξεχωριστή διαχείριση.

Στη συνέχεια μπορείτε να μελετήσετε το παράδειγμα δημιουργίας μιας πλατφόρμας μεγάλων δεδομένων για την ψηφιακή βιβλιοθήκη της Σαγκάης.

Δείτε σχετικά το άρθρο, Cuijuan, X., Lihua, W., & Wei, L. (2021). Shanghai memory as a digital humanities platform to rebuild the history of the city. Digital Scholarship in the Humanities, 36(4), 841-857.

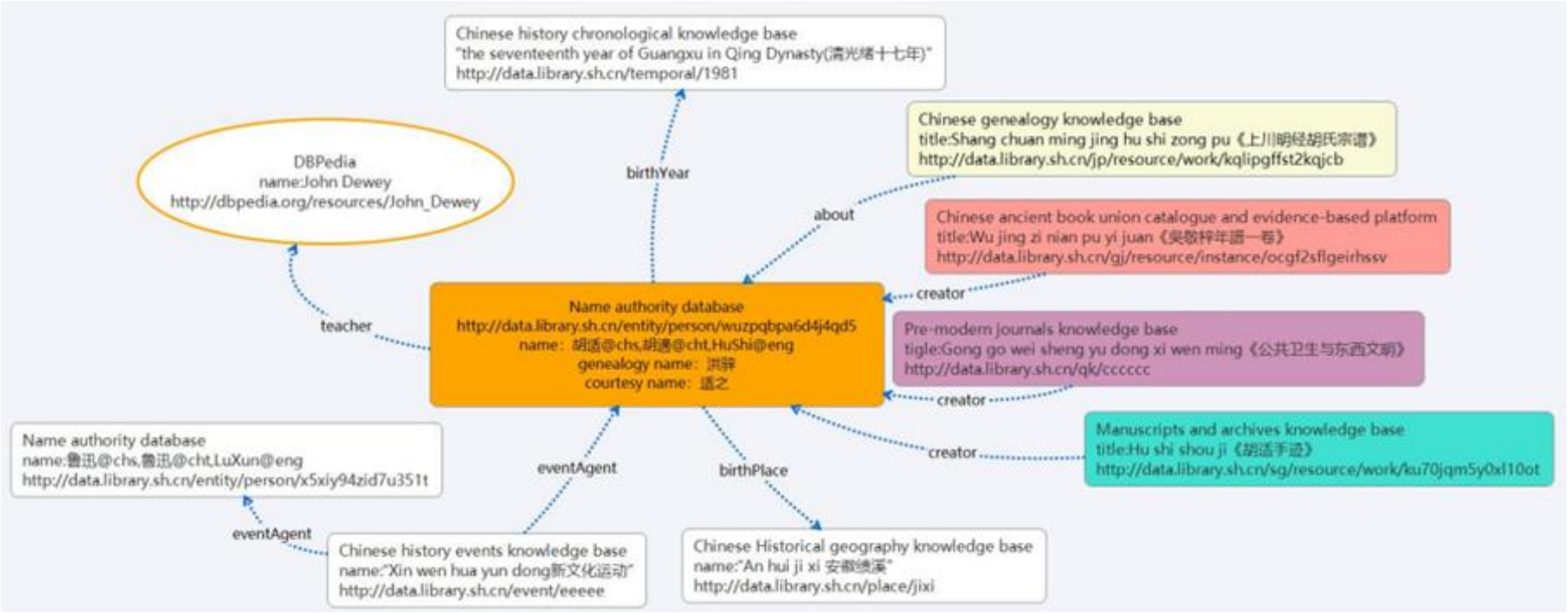
Cuijuan Xia (2020) The Implementation of Historical and Humanities Big Data Platform of Shanghai Library, DCMI Annual Conferences, DCMI Virtual 2020,

DCMI: The Implementation of Historical and Humanities Big Data Platform of Shanghai Library (dublincore.org)

Οι συγγραφείς, περιγράφουν τα βήματα της διαδικασίας:

- **Βήμα 1.** Μετατροπή όλων των εγγραφών μεταδεδομένων διαφορετικών πόρων, οι οποίοι έχουν διαφορετικούς μορφότυπους (format), σε RDF,
- **Βήμα 2.** Αντιμετώπιση των συγγραφέων, των συντελεστών, των εκδοτών κ.λπ. ως οντοτήτων (entities),
- **Βήμα 3.** Εκχώρηση σε κάθε οντότητα ενός “cool URI” (URI το οποίο θα διατηρηθεί, δεν θα αλλάξει, μελλοντικά) ως καθολικού αναγνωριστικού (identifier) και εντοπιστή (locator). Στη συνέχεια, NER (Named Entity Recognition) και αποσαφήνιση οντοτήτων (entity disambiguation). Τέλος, σύνδεση μεταξύ τους,
- **Βήμα 4.** Εμπλουτισμός των οντοτήτων με περισσότερα σημασιολογικά δεδομένα (semantic data) τα οποία είναι δομημένα και εξάγονται από το περιεχόμενο πόρων ή από ανοικτά σύνολα δεδομένων στον Παγκόσμιο Ιστό,
- **Βήμα 5.** Παροχή ανοιχτών δεδομένων στον Παγκόσμιο Ιστό, μετά από έλεγχο «καθιέρωσης» των όρων (authority control). Παροχή υπηρεσιών (αναφορικά με τα δεδομένα) και παροχή API για προγραμματιστές άλλων οργανισμών

.



Εικόνα 2.21 LOD-Linked and Open Data για μεταδεδομένα με στόχο την έκδοση (publishing) και διαμοίραση (sharing) authority data, Xia Cuijuan, Wang Lihua, Liu Wei (2021)

2.9.2 Σημασιολογικός Παγκόσμιος Ιστός, το εννοιολογικό πλαίσιο των συστημάτων ανάκτησης πληροφορίας και οι οντολογίες

Οι μηχανές αναζήτησης (search engines) στον Παγκόσμιο Ιστό βασίζονται σε έννοιες και τεχνικές των συστημάτων ανάκτησης πληροφορίας (Information Retrieval system) και επιτρέπουν αναζήτηση εγγράφων (documents) με λέξεις κλειδιά (keywords) ή και την ελεύθερη αναζήτηση στο κειμένου των εγγράφων (free text searching).

Οι Shalton, McGill στο γνωστό σύγγραμμά τους θεωρούν τα Συστήματα Ανάκτησης Πληροφορίας (ΣΑΠ) ως ένα από τους σημαντικότερους τύπους Πληροφοριακών Συστημάτων και παρατηρούν ότι:

«Η Ανάκτηση Πληροφορίας (information retrieval) γίνεται πιο κατανοητή αν έχουμε στο μυαλό μας ότι η πληροφορία που επεξεργαζόμαστε συνδέεται με (αφορά) έγγραφα (τεκμήρια, documents). Σε αυτά τα συμφραζόμενα (in that context), ένα Σύστημα Ανάκτηση Πληροφορίας ασχολείται με: Την αναπαράσταση (representation), την αποθήκευση (storage) και προσπέλαση (access) των εγγράφων (τεκμηρίων, documents) και την αντιπροσωπευτική περιγραφή τους (representatives of documents, document surrogates). Η πληροφορία εισόδου στο ΣΑΠ είναι πιθανόν να περιλαμβάνει το κείμενο ή απόσπασμα κειμένου του εγγράφου σε φυσική γλώσσα και περιλήψεις (abstracts). Η έξοδος ενός ΣΑΠ σε απάντηση κάποιου αιτήματος αναζήτησης (“in response to a search request”) αποτελείται από ένα σύνολο αναφορών (references). Οι αναφορές αυτές παρέχουν στο χρήστη πληροφορία γύρω από έγγραφα (τεκμήρια) που ενδεχομένως τον ενδιαφέρουν».

Η έννοια του ΣΑΠ ως ειδικού τύπου πληροφοριακού συστήματος διευρύνεται στις μέρες μας. Μία σειρά από νέες έννοιες κυρίως σχετιζόμενες με την ανακάλυψη, εξόρυξη και διαχείριση γνώσης (Knowledge Discovery, Data Mining, Knowledge Management), τα κατανεμημένα και ετερογενή συστήματα (Distributed and Heterogeneous Systems), την ασφάλεια συστημάτων και τις πολιτικές ασφάλειας (Information system security, policies) και η εδραίωση σειράς τεχνολογιών όπως τεχνολογίες διαδικτύου, μηχανικής μάθησης (machine learning) κ.λπ. συνεισφέρουν στην κατεύθυνση αυτή.

Στις μηχανές αναζήτησης θέλουμε πλέον η αναζήτηση να γίνεται με έννοιες αντί λέξεις κλειδιά. Αυτό γίνεται στις μέρες μας κυρίως με χρήση οντολογιών και περιλαμβάνει τη σημασιολογική εστίαση και διεύρυνση των ερωτήσεων (queries) του χρήστη.

Ο παγκόσμιος ιστός (Web) αναπτύχθηκε με εκπληκτικό ρυθμό, με ιστοσελίδες ποικίλου περιεχομένου και ενσωμάτωση και χρήση ποικίλων μορφοτύπων δεδομένων όπως κείμενο, ήχο, βίντεο. Σύμφωνα με τον Tim Berners-Lee “The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation”.

Η εξάπλωση των γλωσσών σήμανσης (XML κ.α.) στοχεύει ακριβώς στη σήμανση του περιεχομένου και των υπηρεσιών αντί της απλής παρουσίασης πληροφορίας στο διαδίκτυο.

Οντολογίες με πληροφορίες σήμανσης μπορούν να χρησιμοποιηθούν για συγκριτικές αναζητήσεις αλλά και για να «εκμεταλλευτούν» πληροφορίες γενίκευσης και ειδίκευσης ή για να υποστηρίξουν διαγλωσσική ανάκτηση (interlingua ontology).

Εάν μια μηχανή αναζήτησης «αντιλαμβάνεται» ότι το ερώτημα (query) του χρήστη δύσκολα μπορεί να απαντηθεί ή επιδέχεται πολλές διαφορετικές (εναλλακτικές) απαντήσεις τότε σε «συνεργασία» με οντολογίες μπορεί να προτείνει απαντήσεις ή νέους όρους αναζήτησης ή ακόμη μπορεί να πάρει την πρωτοβουλία να «τρέξει» και άλλα ερωτήματα με στόχο τη βελτίωση των ερωτημάτων αναζήτησης.

Το Πλαίσιο Περιγραφής Πόρων (Resource Description Framework) RDF του W3C είναι ένα μοντέλο για την αναπαράσταση μεταδεδομένων που περιγράφουν πόρους (resources) του ιστού. Παράδειγμα πόρου είναι ένας ιστότοπος ή ένα html έγγραφο. Το πλαίσιο υποστηρίζει τη διαγλωσσική περιγραφή πόρων και εφαρμογές μεταδεδομένων, όπως είναι οι βιβλιογραφίες.

Σημαντικό ρόλο για την εδραίωση του σημασιολογικού ιστού αποτελεί η πρωτοβουλία του Dublin Core.

Στη συνέχεια θα παρουσιάσουμε έννοιες και παραδείγματα οντολογιών, γλωσσών σήμανσης, RDF, Dublin Core, αλλά και βασικές έννοιες ανάκτησης πληροφορίας και ανάκτησης κειμένου κ.λπ.

2.9.2.1 Οντολογία

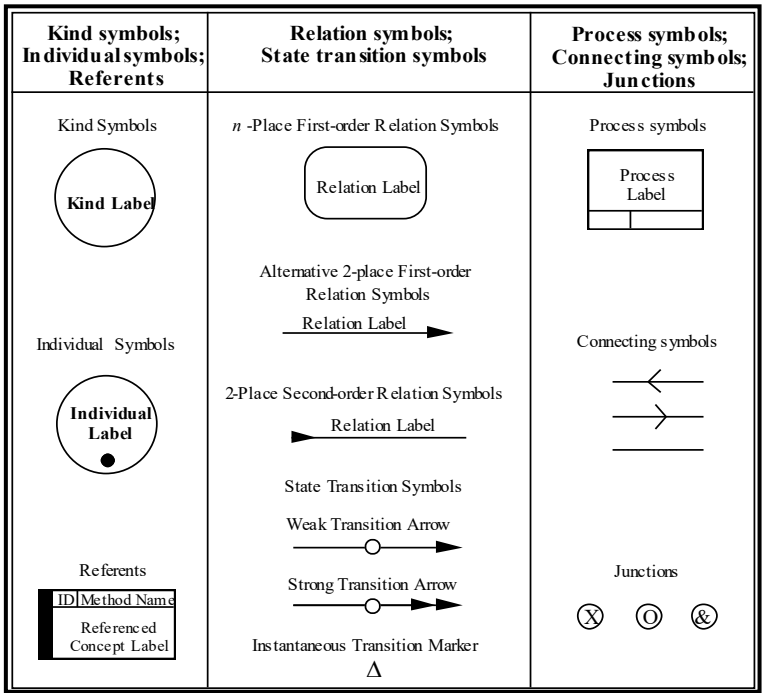
Με τον όρο οντολογία, εννοούμε ένα σύνολο όρων που χρησιμοποιούνται σε ένα (γνωστικό) πεδίο, τους κανόνες που καθορίζουν πώς αυτοί οι όροι μπορούν να συνδυαστούν για την παραγωγή προτάσεων και τα έγκυρα συμπεράσματα που μπορούν να προκύψουν από την εφαρμογή αυτών των προτάσεων στο συγκεκριμένο πεδίο. Επειδή οι άνθρωποι αντιλαμβάνονται από το πλαίσιο της συζήτησης την έννοια των όρων ενώ στον υπολογιστή αυτό δεν ισχύει οι οντολογίες χρησιμοποιούνται σε μία προσπάθεια να μεταφέρουν το εννοιολογικό μοντέλο του ανθρώπου στην αναπαράσταση στον υπολογιστή. Δηλαδή, μία οντολογία ορίζει την τυπική σημασιολογία της πληροφορίας διευκολύνοντας την επεξεργασία της πληροφορίας από τον Η/Υ. Πιο συγκεκριμένα, ορίζει τη σημασιολογία του πραγματικού κόσμου επιτρέποντας τη σύνδεση του περιεχομένου το οποίο επεξεργάζεται μηχανικά, με τη σημασία που του δίνουν οι άνθρωποι βασιζόμενοι σε κοινά αποδεκτή ορολογία.

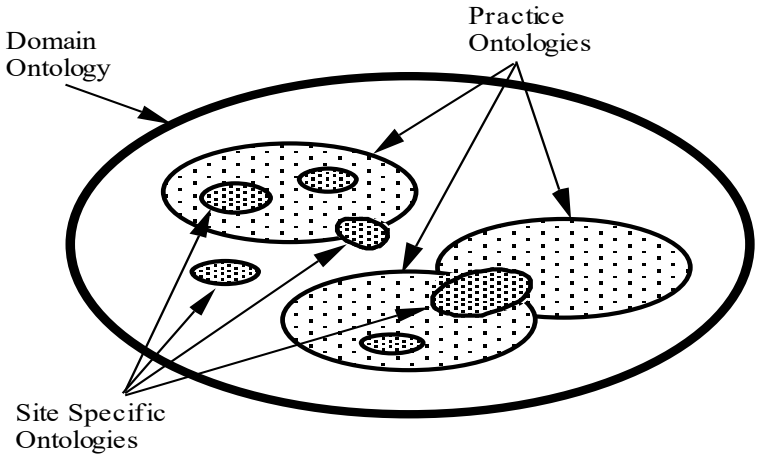
Σύμφωνα με τον Gruber οντολογία είναι “μια διαμοιρασμένη και κοινή κατανόηση κάποιου τομέα, η οποία μπορεί να ανταλλαγεί μεταξύ ανθρώπων και συστημάτων εφαρμογών”. Σε μία παραλλαγή του ορισμού, η οντολογία είναι «ένας τρόπος αναπαράστασης της γνώσης με τέτοιο τρόπο ώστε να μπορεί να αξιοποιηθεί τόσο από ανθρώπους όσο και από υπολογιστές» (Benjamins et al., 1999). Οι οντολογίες διαδραματίζουν σημαντικό ρόλο σ’ αυτή την κατεύθυνση από τη δεκαετία του 90.

Η χρήση οντολογιών (περιοχής) μαζί με γλωσσικούς πόρους, παίζουν σημαντικό ρόλο στην ανάκτηση πληροφοριών (Domingue et al., 2000) (Guarino et al., 1999), (Liao et al., 1999), (Lee et al., 1996), στην εξαγωγή πληροφορίας από έγγραφα Abecker et al., 1998a), (Erdman et al., 2001), (Staab et al., 2001), στη μηχανική ή ημι-αυτοματοποιημένη αναπαράσταση της πληροφορίας από έγγραφα (Erdman et al., 2001) (Staab et al., 2001).

Στον πίνακα 2.6 δίνεται ένα συνοπτικό λεξικό της σχηματικής γλώσσας περιγραφής οντολογιών IDEF5. Η IDEF5 επιλέγεται στο παρόν σύγγραμμα γιατί παρέχει ένα απλό και ξεκάθαρο τρόπο εισαγωγής και συζήτησης των εννοιών της οντολογίας. Ο ενδιαφερόμενος αναγνώστης για περαιτέρω ανάλυση του ζητήματος στην ενδεικτική βιβλιογραφία.

Πίνακας 2.6 Λεξικό όρων σχηματικής γλώσσας περιγραφής οντολογιών (IDEF5, 1994).





Εικόνα 2.22 Τα τρία επίπεδα οντολογιών (IDEF 5, 1994).

Στην εικόνα 2.21 βλέπουμε τα τρία επίπεδα οντολογιών;

1) **Οντολογίες τομέα (domain ontology):** Αποτελούν μια γενική κατηγορία που καλύπτει (και τις επιμέρους οντολογίες που ορίζονται στα πλαίσια ενός) τομέα π.χ. Οι Οντολογίες αυτοκινήτων ανήκουν σε οντολογία κατασκευαστικού τομέα ή από Οντολογίες αυτοκινήτων σχηματίζουμε οντολογία κατασκευαστικού τομέα.
2) **Οντολογίες πρακτικής (Practice ontologies):** Αποτελούν εξειδικευμένες οντολογίες που εστιάζουν (εξειδικεύονται) σε ειδικότερα θέματα ενός τομέα π.χ. Οντολογία εξαρτημάτων αυτοκινήτου.
3) **Οντολογίες κατά περιοχή (site-specific):** Αποτελούν εξειδικευμένες οντολογίες που καλύπτουν ειδικές ανάγκες μιάς περιοχής ή πρακτικής π.χ. Η Οντολογία εξαρτημάτων αυτοκινήτων FIAT αποτελεί εξειδίκευση οντολογίας πρακτικής

Δείτε σχετικά το εγχειρίδιο,

IDEF 5 (1994). "IDEF5 Method report", Information Integration for Concurrent Engineering, 1994, available at http://www.kbsi.com Idef5.pdf (tcd.ie) (πρόσβαση στις 9/5/2022)

Γλώσσες περιγραφής οντολογιών

Υπάρχουν περιγραφικές και σχηματικές γλώσσες για την περιγραφή οντολογιών πχ το προϊόν IDEF5 της ICAM (Integrated Computer Aided Manufacturing), γλώσσες και εργαλεία ανάπτυξης οντολογιών προσανατολισμένων στο Web πχ. RDF, SHOE, Protégé, DAML-OIL, OilEd αλλά και προγράμματα πράκτορες (software agents) που αναζητούν πόρους με τη βοήθεια της οντολογίας. Στην περίπτωση αυτή αντί της απλής αναζήτησης έχουμε σημασιολογικά εμπλουτισμένη αναζήτηση.

Παράδειγμα οντολογίας κρασιών

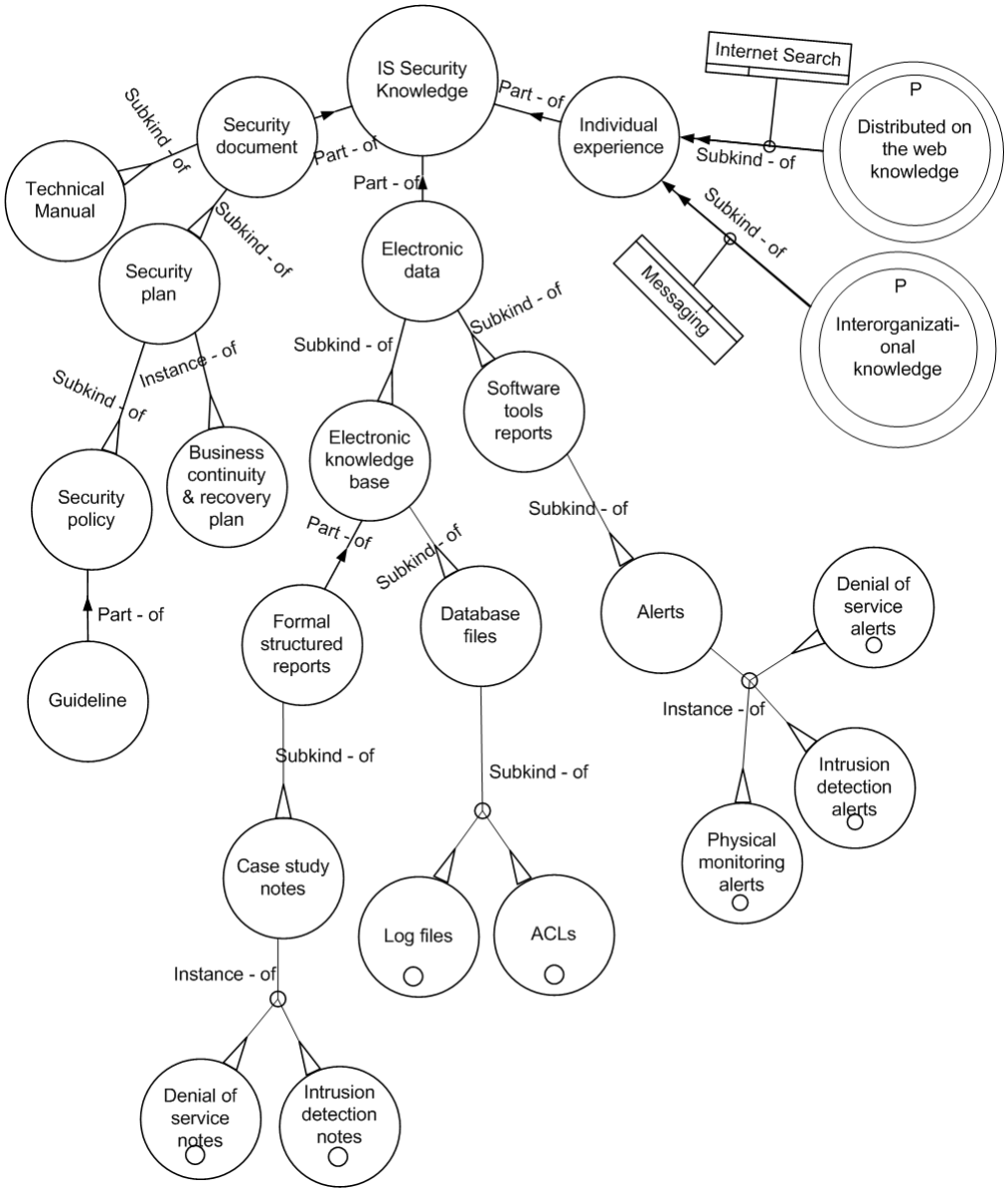
Παραθέτουμε απόσπασμα οντολογίας κρασιών γραμμένο στο γνωστό και δημοφιλές σε πανεπιστήμια εργαλείο protégé (http://protege/stanford.edu/index.html). Στο παράδειγμα, η οντολογία αποτελείται από έννοιες (όρους) και από πεδία (attributes). Ανάλογα με τις τιμές των πεδίων (attributes) μπορούμε να αποδίδουμε σημασιολογία στην έννοια.

```
([Chateau+Morgon+Beaujolais] of Beaujolais
        (body LIGHT)
                (sugar DRY)
                (tannin+level LOW)
                (flavor DELICATE)
                (grape [Gamay+grape])
                (name "Chateau Morgon Beaujolais")
                (maker [Chateau+Morgon])
)
 ([Congress+Springs+Semillon] of Semillon
                (body MEDIUM)
                (sugar DRY)
                (flavor MODERATE)
                (name "Congress Springs Semillon")
                (maker [Congress+Springs])
)
```

Παρατηρήστε ότι η λέξη congress αναφέρεται σε τύπο κρασιού! 0ι παραδοσιακές μηχανές αναζήτησης όταν δίνουμε τη λέξη congress πιθανότατα θα μας δείξουν σελίδες που περιλαμβάνουν τη λέξη με τη σημασία εκείνη που παρουσιάζει το μεγαλύτερο ποσοστό εμφάνισης και το μεγαλύτερο κύρος στον ιστό. Έτσι, αν θέλουμε να ψάξουμε για τον παραπάνω τύπο κρασιού θα πρέπει να εμπλουτίσουμε την αναζήτησή μας με αρκετούς όρους αλλά και τότε γενικά είναι δύσκολο να ικανοποιηθεί το αίτημά μας χωρίς τη χρήση οντολογίας.

Παράδειγμα οντολογίας διαχείρισης της ασφάλειας πληροφοριών

Ακολουθεί οντολογική περιγραφή της γνώσης που αφορά τη διαχείριση της πληροφορίας (και της γνώσης) που είναι σχετική με την ασφάλεια συστημάτων. Γίνεται χρήση της σχηματικής γλώσσας περιγραφής IDEF5 (βλέπε εικόνα 2.22).



Εικόνα 2.23 Οντολογική περιγραφή της γνώσης που αφορά τη διαχείριση της πληροφορίας της σχετικής με την ασφάλεια συστήματος, (ACL: Access Control Lists)

Η κεντρική έννοια που εμφανίζεται στην εικόνα 2.23, είναι αυτή της γνώσης που αφορά τη διαχείριση της πληροφορίας (και γνώσης) της σχετικής με την ασφάλεια συστήματος. Η γνώση αυτή είναι ρητή (explicit), δηλαδή είναι καταγεγραμμένη / εντοπίζεται σε διαφορετικές πηγές, όπως σε έγγραφα που σχετίζονται με την ασφάλεια ή σε ηλεκτρονικά δεδομένα. Επίσης είναι άρρητη (tacit) γνώση, δηλαδή μπορεί να εντοπίζεται για παράδειγμα στην πείρα ειδικών σε θέματα ασφάλειας, και αποτελεί πρόκληση να μπορέσουμε να την αξιοποιήσουμε.

Στη γλώσσα IDEF5 οι σχέσεις part-of και subkind-of έρχονται να αντικαταστήσουν τη σχέση is-a που χρησιμοποιούμε συνήθως σε αναπαραστάσεις της τεχνητής νοημοσύνης όπως τα σημασιολογικά δίκτυα. Στην εικόνα βλέπουμε την έννοια της γνώσης ασφάλειας να σχετίζεται με τη βοήθεια σχέσης part-of με τις διαφορετικές πηγές γνώσης που προαναφέρθηκαν. Παρατηρούμε ότι η γνώση που σχετίζεται με την ασφάλεια μπορεί να διακριθεί σε γνώση που εντοπίζεται σε έγγραφα, γνώση που εντοπίζεται σε ηλεκτρονικά αρχεία και γνώση που αποτυπώνεται στο εσωτερικό του οργανισμού ή βρίσκεται αποτυπωμένη σε πηγές, όπως είναι οι ιστότοποι του Internet, από όπου είναι δύσκολο να την ανακτήσουμε. Στην περίπτωση της άρρητης, μη καταγεγραμμένης άρα όχι άμεσα αξιοποιήσιμης γνώσης, υπάρχει μια δομή την οποία είναι δύσκολο να

αποτυπώσουμε. Για τον λόγο αυτό στο διάγραμμα (που επεξηγεί τους όρους τις οντολογίας) έχουμε απόκρυψη της εσωτερικής δομής, γεγονός που παριστάνεται από δύο ομόκεντρους κύκλους. Με τη βοήθεια τεχνικών αναζήτησης μπορούμε να ανακαλέσουμε τη γνώση αυτή, γεγονός που παριστάνεται από τη μετάβαση μέσα στο πλαίσιο όπου αναγράφεται "messaging". Επίσης, αναπαριστούμε για κάθε πηγή γνώσης και ένα αριθμό συγκεκριμένων στιγμιότυπων. Για παράδειγμα στην περίπτωση ηλεκτρονικού τύπου δεδομένων έχουμε αναφορές που προέρχονται από εργαλεία λογισμικού (OLAP κ.λπ.) και επιπλέον τα περιεχόμενα ηλεκτρονικής βάσης γνώσης. Τέλος, οι ειδικές περιπτώσεις αρχείων παριστάνονται μέσα σε κύκλο με τελεία στο κέντρο, γεγονός που υπογραμμίζει την ύπαρξη στιγμιοτύπων με ειδική σημασία.

Σύντομη αναφορά στη γλώσσα OWL και τις οντολογίες

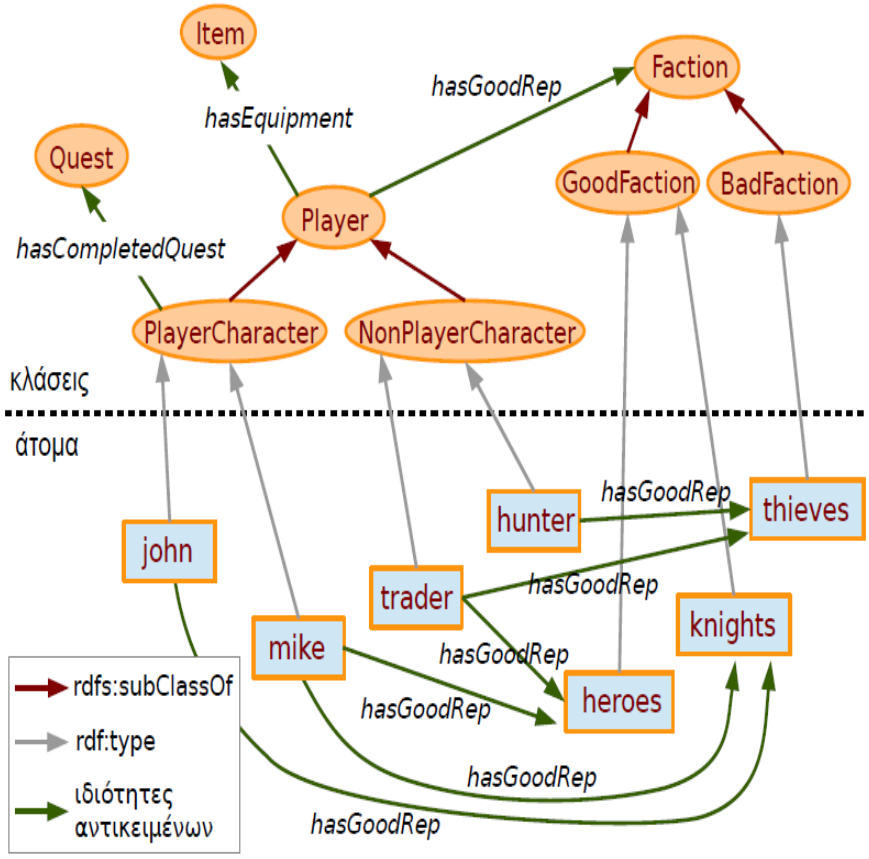
Ο ενδιαφερόμενος αναγνώστης παραπέμπεται στο σύγγραμμα των Στεφανιδάκη κ.ά. (2015) για περαιτέρω μελέτη και ειδικότερα στην ενότητα Περιγραφικές Λογικές (Descriptive Logics) που έχει άμεση σχέση με τη γλώσσα OWL. Η γλώσσα OWL περιγράφει ένα πεδίο εφαρμογής χρησιμοποιώντας τις ακόλουθες βασικές έννοιες (Στεφανιδάκης κ.ά., 2015):

1) κλάσεις, που αντιστοιχούν στις έννοιες των Περιγραφικών Λογικών
2) ιδιότητες μεταξύ δύο κλάσεων, που αντιστοιχούν στους ρόλους των Περιγραφικών Λογικών
3) άτομα που ανήκουν στο πεδίο της εφαρμογής και τα οποία αντιστοιχούν στα άτομα των Περιγραφικών Λογικών.

Η γλώσσα OWL είναι μια αντικειμενοστρεφής γλώσσα, η οποία έχει τη δική της αυστηρή σύνταξη (syntax), και χρησιμοποιείται για να: 1) ορίζει οντολογίες που αποτελούνται από κλάσεις, ιδιότητες και άτομα, και 2) περιγράφει τις περίπλοκες σχέσεις που υφίστανται μεταξύ κλάσεων, ιδιοτήτων, καθώς και μεταξύ ατόμων.

Η οντολογία OWL (OWL Ontology) είναι ένα σύνολο αξιωμάτων (axioms), τα οποία παρέχουν λογικούς ισχυρισμούς (logical assertions) για τρεις τύπους πραγμάτων: κλάσεις (classes), άτομα (individuals) και ιδιότητες (properties). Χρησιμοποιώντας λογισμικό εξαγωγής συμπερασμάτων (reasoner), μπορούμε να συμπεράνουμε άλλα γεγονότα (facts) που περιέχονται στην οντολογία. Για παράδειγμα εάν ένα άτομο (individual) Martin είναι στην κλάση (class) Student, και η κλάση Student είναι μια υποκλάση (subclass) της κλάσης Person, ο reasoner θα συμπεράνει ότι ο Μάρτιν είναι Person (Πρόσωπο).

Με βάση τα παραπάνω ορίζονται στη γλώσσα OWL οντολογίες που απαρτίζονται από αξιώματα που εκφράζουν τις σχέσεις μεταξύ των κλάσεων και των ατόμων, και τα οποία αντιστοιχούν σε αξιώματα των Περιγραφικών Λογικών. Παραθέτουμε παράδειγμα (Στεφανιδάκης κ.ά., 2015). Στην Εικόνα 2.23 βλέπουμε τα άτομα και τις ιδιότητες οντολογίας διαδικτυακού παιχνιδιού ρόλων (Role Playing Game-RPG).



Εικόνα 2.24 Οντολογία διαδικτυακού παιχνιδιού ρόλων (Στεφανιδάκης κ.ά., 2015).

2.9.2.2 Αναφορές και ενδεικτική βιβλιογραφία

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. Scientific American, 284(5), 34-43.

Belsis P., Gritzalis S., (2004) Distributed Autonomous Knowledge Acquisition and Dissemination Ontology based Framework in Proceedings of the PAKM 2004 5th International Conference on Practical Aspects of Knowledge Management - Workshop on Enterprise Modeling and Ontology: Ingredients for Interoperability, D. Karagiannis (Ed.), December 2004, Vienna, Austria, Univ. of Vienna, pp. 100-104

Belsis P., Gritzalis S., Skourlas C. (2005). Security Enhanced Distributed Knowledge Management Architecture, in Proceedings of the I-KNOW'05 5th International Conference on Knowledge Management, K. Tochtermann, H. Maurer (Eds.) July 2005, Graz, Austria, Springer

Benjamins R., Fensel D., Decker S., Perez A. (1999). KA: Building Ontologies for the Web: a mid-term report, Int. J. of Human-Computer Studies, 51, 687-712.

Gruber T. R., (1994) (1994) Towards principles for the design of ontologies used for knowledge sharing, International Journal of Human Computer Studies, 43 (3/4), 907-928.

Guarino N. (1996). Understanding, building and using ontologies, International Journal of Human – computer Studies/Knowledge Acquisition, Special issue on using Ontologies.

Domingue, J., & Motta, E. (2000). PlanetOnto: from news publishing to integrated knowledge management support. IEEE Intelligent Systems and their Applications, 15(3), 26-32.

Guarino, N., Masolo, C., & Vetere, G. (1999). Ontoseek: Content-based access to the web. IEEE Intelligent Systems and their Applications, 14(3), 70-80.

Lee, J., Gruninger, M., Jin, Y., Malone, T., Tate, A., Yost, G., & PIF Working Group. (1998). The process interchange format and framework. The knowledge engineering review, 13(1), 91-120.

Liao, M., Abecker, A., Bernardi, A., Hinkelmann, K., & Sintek, M. (1999, June). Ontologies for knowledge retrieval in organizational memories. In Proceedings of the Learning Software Organizations (LSO'99) workshop, Kaiserslauten, Germany (pp. 19-26).

Abecker, A., Bernardi, A., Hinkelmann, K., Kühn, O., & Sintek, M. (1998). Techniques for organizational memory information systems. DFKI Document D-98-02, 2.

Erdmann, M., Maedche, A., Schnurr, H. P., & Staab, S. (2000, August). From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. In Proceedings of the COLING-2000 workshop on semantic annotation and intelligent content (pp. 79-85).

Staab, S., & Maedche, A. (2001). Knowledge portals: Ontologies at work. AI magazine, 22(2), 63-63.

IDEF 5 (1994). Information Integration for Concurrent Engineering, (IDEF5 Method report) 1994, available at http://www.kbsi.com

Μπέλσης Π., Σκουρλάς Χ., Διαγλωσσική Ανάκτηση Πληροφοριών, Σημασιολογικός Ιστός και Μηχανές Αναζήτησης, 2ο συνέδριο «Αρχιμήδης, 2006.

Δανίκας Γ., Μ. Τζαμάκου, (2005) Σημασιολογικός Ιστός – Εφαρμογές και προοπτικές με χρήση οντολογιών προσανατολισμένων στην Ασφάλεια Π.Σ., Διπλωματική Εργασία, ΤΕΙ Αθήνας, 2005

2.9.3 Μεταδεδομένα και Γλώσσες σήμανσης περιεχομένου. XML

Με τον όρο μεταδεδομένα ή μεταπληροφορία (metadata) αναφερόμαστε σε δεδομένα που χρησιμοποιούνται για προσδιορισμό, περιγραφή ή εντοπισμό πληροφορίας που υπάρχει σε πηγές (ή πόρους, resources) ηλεκτρονικές ή μη. Δηλαδή, τα μεταδεδομένα είναι δομημένη πληροφορία που περιγράφει ένα ψηφιακό έγγραφο (document) και διευκολύνει την ανάκτησή του.

Σύμφωνα με τον Tim Berners-Lee "Metadata is machine understandable information about web resources or other things."

Υπάρχουν πολλά πρότυπα μεταδεδομένων ανάλογα με το είδος του εγγράφου.

Η γλώσσα σήμανσης περιεχομένου XML (Extensible Markup Language) προσφέρει ένα πρότυπο επικοινωνίας και ανταλλαγής πληροφοριών μεταξύ ετερογενών κόμβων πληροφοριών του διαδικτύου. Είναι μια γλώσσα μεταδεδομένων που αντιμετωπίζει αποτελεσματικά την ανάγκη καθορισμού ετικετών σήμανσης (tags) εξειδικευμένων για κάθε εφαρμογή και παρέχει ένα μορφότυπο δεδομένων (format) για τη δόμηση του περιεχομένου των εγγράφων. Ακολουθεί παράδειγμα.

```
<INSTITUTE>University of Macedonia</INSTITUTE>
      <DEPARTMENT>Department of Applied Informatics</DEPARTMENT>
            <ADDRESS>
                  <STREET>Egnatias</STREET>
                  <NUMBER>156</NUMBER>
            </ADDRESS>
```

Χρησιμοποιεί ένα σχήμα DTD (Document Type Definition), ή XML σχήμα, για να ορίσει το ειδικό «λεξιλόγιο» της συγκεκριμένης εφαρμογής και τον συνδυασμό των ετικετών που επιτρέπονται.

Παράδειγμα χρήσης MARC XML για την περιγραφή βιβλιογραφικών αναγραφών

Ακολουθεί απόσπασμα δίγλωσσης (ελληνικά και αγγλικά) βιβλιογραφικής αναγραφής. Χρησιμοποιείται το βιβλιοθηκονομικό πρότυπο MARC και γλώσσα XML. Στην εικόνα 2.24 βλέπουμε την αναγραφή (record).

Ταυτότητα Εγγραφής:	000066
Τίτλος:	Συνεργατική βασιζόμενη σε οντολογία ευρετηρίαση και ανάκτηση
Μεταφρασμένος τίτλος :	Collaborative ontology-based information indexing and retrieval
Συγγραφέας:	Χριστόπουλος, Βασίλειος Ν. (070) Christopoulos, Vassilios N. (070)
Σημειώσεις:	Βιογραφικό Σημείωμα …
Τμήμα του:	Διαχείριση γνώσης και ανάκτηση πληροφορίας. 2006
Θέματα:	Διαχείριση γνώσης / Knowledge Management Εξατομίκευση / Personalization Ανάκτηση πληροφορίας / Information Retrieval
Τοπική ταξινόμηση:	19.04

Εικόνα 2.24 *Βιβλιογραφική αναγραφή.*

Ακολουθεί κώδικας XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
 <bibliographic_records>
 <webgate>Βιβλιογραφική Βάση Δεδομένων</webgate>
 <record>
  ...
 <controlfield tag="001">66</controlfield>
...
 <datafield tag="200" ind1="1" ind2="">
 <subfield code="a">Συνεργατική βασιζόμενη σε Οντολογία ευρετηρίαση και ανάκτηση
</subfield>
 </datafield>
 ...
 <datafield tag="321" ind1="" ind2="">
  <subfield code="a">Βιογραφικό Σημείωμα … </subfield>
 </datafield>

<datafield tag="463" ind1="_" ind2="1">
   ...
   <subfield code="1">2001</subfield>
   <subfield code="a"> Διαχείριση Γνώσης και Ανάκτηση Πληροφορίας. </subfield>
   <subfield code="v">2006</subfield>
 </datafield>
 <datafield tag="541" ind1="1" ind2="">
   <subfield code="a"> Collaborative ontology-based information indexing and
retrieval </subfield>
 </datafield>
 <datafield tag="606" ind1="|" ind2="">
   <subfield code="a">Διαχείριση Γνώσης</subfield>
 </datafield>
 <datafield tag="606" ind1="|" ind2="">
<subfield code="a">Knowledge Management</subfield>
 </datafield>
........
........
 <datafield tag="686" ind1="" ind2="">
```

```
<subfield code="2">jel</subfield>
<subfield code="a">19.04</subfield>
</datafield>
<datafield tag="701" ind1="" ind2="1">
<subfield code="a">Χριστόπουλος</subfield>
<subfield code="b">Βασίλειος Ν</subfield>
<subfield code="4">070</subfield>
</datafield>
<datafield tag="701" ind1="" ind2="1">
<subfield code="a">Christopoulos</subfield>
<subfield code="b">Vassilios N</subfield>
 <subfield code="4">070</subfield>
</datafield>
....
</record>
</bibliographic_records>
```

2.9.4 Πλαίσιο RDF

Το Πλαίσιο Περιγραφής Πόρων - Resource Description Framework (RDF) του W3C είναι ένα μοντέλο για την αναπαράσταση μεταδεδομένων που περιγράφουν πόρους (resources) του ιστού, π.χ., ένας πόρος είναι ένας ιστότοπος, ένα html έγγραφο. Σημαντικό ρόλο για την εδραίωση του σημασιολογικού ιστού αποτελεί η πρωτοβουλία του Dublin Core. Το Dublin Core περιλαμβάνει μια σειρά από "στοιχεία" (properties) για την περιγραφή εγγράφων και επομένως και για την καταγραφή metadata. Για παράδειγμα, στοιχεία του Dublin Core (που χρησιμοποιούνται για την τεκμηρίωση πηγών του Internet είναι: Title-όνομα πόρου (resource), Creator-οντότητα υπεύθυνη για το περιεχόμενο του πόρου, Subject–θέμα, Description-περιγραφή περιεχομένου, Language-Γλώσσα του περιεχομένου του πόρου κ.λπ. (βλέπε και Dublin Core Metadata Element Set: Reference Description)

DCMI: Dublin Core™ Metadata Element Set, Version 1.1: Reference Description

Το πλαίσιο RDF προσφέρει μια άμεση, επαρκή απεικόνιση για τις Dublin Core πληροφορίες. Ακολουθεί παράδειγμα σε XML.

```
<rdf:Description rdf:about="http://www.securityfocus.com">
      <rdf:type rdf:resource="&my_rdfs;Resource"/>
      <dc:title>Security Focus</dc:title>
      <dc:Description>SecurityFocus is the most
        comprehensive and trusted source
        of security information on the Internet…
      </dc:Description>
</rdf:Description>
```

Οι μηχανές αναζήτησης (search engines) του Παγκόσμιου Ιστού βασίζονται σε έννοιες και τεχνικές των συστημάτων ανάκτησης πληροφορίας (Information Retrieval) και επιτρέπουν αναζήτηση τεκμηρίων-εγγράφων (documents) με λέξεις-κλειδιά (keywords) ή την ελεύθερη αναζήτηση στο κείμενο των εγγράφων (free text searching). Στις μηχανές αναζήτησης του Σημασιολογικού Ιστού θέλουμε πλέον η αναζήτηση να γίνεται με έννοιες αντί λέξεις-κλειδιά. Αυτό γίνεται στις μέρες μας κυρίως με χρήση ελεγχόμενων λεξιλογίων και οντολογιών και περιλαμβάνει τη σημασιολογική εστίαση και διεύρυνση των ερωτήσεων (queries) του χρήστη.

Σαν παραδείγματα αναφέρουμε δύο διάσημα λεξιλόγια που χρησιμοποιούν τη γλώσσα RDF, την οντολογία FOAF και το πρότυπο Dublin Core (Στεφανιδάκης, Ανδρόνικος, Παπαδάκης, 2015).

2.9.4.1 Πρότυπο Dublin Core (Dublin Core, 2021)

Απώτερος στόχος του προτύπου είναι η διευκόλυνση της ανάκτησης των ψηφιακών πόρων στο Διαδίκτυο. Η σημασιολογία των όρων του Dublin Core έχει συνδιαμορφωθεί από μια διεθνή ομάδα επαγγελματιών που προέρχεται από τον χώρο της βιβλιοθηκονομίας, της επιστήμης των υπολογιστών, των μουσείων και άλλους συναφείς τομείς.

Χρήση Dublin Core και RDF για την περιγραφή των πόρων

Ένα σημαντικό βήμα για την εδραίωση του σημασιολογικού ιστού είναι η πρωτοβουλία Dublin Core. Η περιγραφή του συνόλου των στοιχείων μεταπληροφορίας της πρωτοβουλίας Dublin Core υπάρχει στις ιστοσελίδες της. Δείτε για παράδειγμα τις σελίδες:

DCMI: Dublin Core™

DCMI: DCMI Schemas (dublincore.org)

Παραπέμπει στις σελίδες XMLS Schemas και RDFS Schemas

DCMI: Metadata Basics (dublincore.org)

DCMI: Dublin Core™ Metadata Element Set, Version 1.1: Reference Description

Τα καθορισθέντα στοιχεία μεταπληροφορίας για την περιγραφή πόρων (resources) είναι προαιρετικά, μπορούν να επαναλαμβάνονται, να εμφανίζονται με οποιαδήποτε σειρά στην περιγραφή, έχουν ένα όνομα δηλωτικό της σημασίας τους και μια «επίσημη» ετικέτα (tag). Τα στοιχεία χωρίζονται σε τρεις ομάδες: (1) στοιχεία σχετικά με το Περιεχόμενο του πόρου (Τίτλος, Θέμα, Περιγραφή, Πηγή, Γλώσσα, Σχέση, Κάλυψη), (2) στοιχεία σχετικά με την Πνευματική Ιδιοκτησία της πηγής (Δημιουργός, Εκδότης, Συντελεστής, Δικαιώματα) και (3) στοιχεία σχετικά με το Στιγμιότυπο της πηγής (Ημερομηνία, Τύπος, Μορφότυπος, Κωδικός). Στη συνέχεια παρατίθενται οι ετικέτες και επεξηγήσεις των στοιχείων που χρησιμοποιούνται για την περιγραφή (τεκμηρίωση) πόρων του Internet (Μαρινάγη και Σκουρλάς, 2022):

1) «Title: Τίτλος». Περιέχει το όνομα του πόρου.
2) «Creator: Συγγραφέας ή Δημιουργός». Περιέχει την οντότητα που είναι υπεύθυνη για τη δημιουργία και το περιεχόμενο του πόρου.
3) «Subject: Θέμα και Λέξεις-Κλειδιά». Προτείνεται η χρήση καθορισμένων γλωσσαρίων και επίσημων σχημάτων ταξινόμησης.
4) «Description: Περιγραφή». Περιέχει περιγραφή περιεχομένου, π.χ., περίληψη κειμένου, περιγραφή περιεχομένου οπτικού πόρου.
5) «Publisher: Περιγραφή». Περιέχει οντότητα υπεύθυνη για τη διαθεσιμότητα του πόρου, π.χ., έναν εκδοτικό οίκο, ένα τμήμα Πανεπιστημίου.
6) «Contributor: Συντελεστής». Περιέχει οντότητα που δεν προσδιορίζεται στο στοιχείο Δημιουργός αλλά έχει συμβολή στο περιεχόμενο του πόρου, π.χ., εκδότης, μεταφραστής, εικονογράφος.
7) «Date: Ημερομηνία». Είναι ημερομηνία σχετική με γεγονός στον κύκλο ζωής του πόρου πχ 1954-06-28. Προτείνεται η χρήση του προτύπου ISO 8601.
8) «Type: Τύπος Πόρου». Περιέχει την κατηγορία του περιεχομένου του πόρου, π.χ., home page, τεχνική αναφορά, γλωσσάρι.

9) «Format: Μορφότυπος». Περιγραφή που αναφέρεται συνήθως στο περιβάλλον (software και ενδεχομένως και hardware) που χρησιμοποιήθηκε στην υλοποίηση του πόρου και άρα μπορεί να χρειαστεί για την εμφάνιση / έκθεση ή τη λειτουργία του πόρου.
10) «Identifier: Κωδικός Πόρου». Μοναδικός κωδικός για τον πόρο, π.χ., URL, International Standard Book Numbers (ISBN).
11) «Source: Πηγή». Πληροφορίες για την πηγή από την οποία προέρχεται ο παρών πόρος.
12) «Language: Γλώσσα». Η γλώσσα του περιεχομένου του πόρου, π.χ., gr, en, de, es.
13) «Relation: Σχέση». Αναγνωριστικό/αναφορά σε σχετικό πόρο. Παραδείγματα σχέσεων είναι: IsVersionOf, IsBasedOn, IsPartOf, IsFormatOf.
14) «Coverage: Κάλυψη». Αναφέρεται σε μέγεθος και σκοπό – τι καλύπτει - του περιεχομένου του πόρου.
15) «Rights: Δικαιώματα Χρήσης». Πληροφορίες σχετικές με τα δικαιώματα διαχείρισης/χρήσης του πόρου.

2.9.4.2 Συνδεδεμένα δεδομένα (linked data), πρότυπο RDF και γλώσσα XML.

Αντιγράφουμε από το σύγγραμμα των Μαρινάγη και Σκουρλά (2022).

«Τα συνδεδεμένα δεδομένα είναι ένα πρώτο και πολύ σημαντικό βήμα για το πέρασμα από τον Παγκόσμιο Ιστό στον Σημασιολογικό Ιστό. Οι αρχές στις οποίες πρέπει να υπακούει η υλοποίηση των συνδεδεμένων δεδομένων είναι εγγενείς στον Σημασιολογικό Ιστό. Για παράδειγμα, οι οντότητες δεδομένων (data entities) πρέπει να έχουν μοναδικά αναγνωριστικά (μοναδική ταυτότητα, unambiguous identification) που είναι προϋπόθεση για την ενσωμάτωση-ολοκλήρωση δεδομένων (data integration). Με τον τρόπο αυτό μπορούν εύκολα να εδραιωθούν-υποστηριχθούν απλές υπηρεσίες από τα υπάρχοντα αποθετήρια δεδομένων (data repositories). Με τις τεχνολογίες των συνδεδεμένων δεδομένων διευκολύνεται η μετατροπή δεδομένων από γλώσσα XML σε δεδομένα συμβατά με το πρότυπο RDF, η υλοποίηση προσόψεων RDF μπροστά από τις βάσεις δεδομένων SQL (RDF facades in front of SQL databases)».

Σύμφωνα με τους Στεφανιδάκη κ.ά. (2015): «Τα Συνδεδεμένα Δεδομένα αποτελούν απλουστευμένη εκδοχή των ιδεών του Σημασιολογικού Ιστού. Η αξιοποίησή τους απαιτεί συναίνεση, δηλαδή την εισαγωγή κοινών "καλών πρακτικών" υλοποίησης λογισμικού εφαρμογής. Κύριο χαρακτηριστικό τους είναι η ανοικτότητα, δηλαδή επιβάλλεται η ανοιχτή διάθεση των δεδομένων για την ανάπτυξη και διάθεση της σημασιολογικής πληροφορίας».

2.9.4.3 Το πρότυπο HTTP URI (Uniform Resource Identifier), τα μοναδικά αναγνωριστικά URI και Σημασιολογικός Ιστός

«Το πρότυπο αποτελεί θεμέλιο του Σημασιολογικού Ιστού επειδή προδιαγράφει την εκχώρηση με μονοσήμαντο τρόπο ενός αναγνωριστικού ονόματος σε οποιαδήποτε έννοια ή οντότητα του Ιστού. Κάθε οντότητα που συμπεριλαμβάνεται σε δηλώσεις σημασιολογικών δεδομένων έχει μία μοναδική ταυτότητα, το αναγνωριστικό ονόματος URI, σε παγκόσμιο επίπεδο. Τα αναγνωριστικά ονόματα αποτελούν το θεμέλιο και της τεχνολογίας των συνδεδεμένων δεδομένων, υποστηρίζουν την ανακάλυψη και την προσπέλαση σημασιολογικών δεδομένων που διατίθενται στον Παγκόσμιο Ιστό» (Μαρινάγη και Σκουρλάς, 2022).

«Τα URIs αυτά αποτελούν διευθύνσεις, τις οποίες οι εφαρμογές μπορούν να χρησιμοποιήσουν ως υπερσυνδέσμους για να ανακαλύψουν νέα γνώση» (Στεφανιδάκης κ.ά., 2015).

Ο Tim Berners-Lee πρότεινε τέσσερις αρχές για την έκδοση (δημοσίευση, publish), δηλαδή την ανάρτηση με στόχο τη δημόσια χρήση συνδεδεμένων δεδομένων (Linked Data):

1) «Χρησιμοποιήστε URIs για να αναγνωρίσετε οποιοδήποτε πράγμα (οντότητα)» (Use URIs as names for things).
2) «Χρησιμοποιήστε HTTP URIs έτσι ώστε να αποτελούν επισκέψιμες διευθύνσεις στον παγκόσμιο ιστό». (Use HTTP URIs so people can look up those names).
3) «Όταν κάποιος επισκέπτεται τη διεύθυνση ενός URI, δώστε χρήσιμη πληροφορία χρησιμοποιώντας τα πρότυπα (δηλαδή RDF, SPARQL)» (When someone looks up a URI, provide useful information using the standards).
4) «Στα δεδομένα που επιστρέφετε, παραθέστε συνδέσεις με νέα URIs, έτσι ώστε να μπορεί να ανακαλυφθεί περαιτέρω γνώση» (Include links to other things, so people can discover more). Έτσι, μια σημασιολογική εφαρμογή μπορεί να μετακινείται από διεύθυνση σε διεύθυνση συγκεντρώνοντας νέα δεδομένα, όπως ακριβώς ένας χρήστης μετακινείται από ιστοσελίδα σε ιστοσελίδα.

Το πλαίσιο περιγραφής πόρων (Resource Description Framework-RDF) καθορίζει τη βασική γλώσσα περιγραφής δεδομένων και μεταδεδομένων του Σημασιολογικού Ιστού. Χρησιμοποιεί έναν γράφο του μορφοτύπου δεδομένων (graph data format). Το γράφημα-μοντέλο RDF (Graph Model) βασίζεται στην ιδέα ότι κάθε πόρος, οντότητα, έννοια κ.λπ. έχει ένα μοναδικό αναγνωριστικό Ιστού, URI (Uniform Resource Identifier) (Μαρινάγη και Σκουρλάς, 2022).

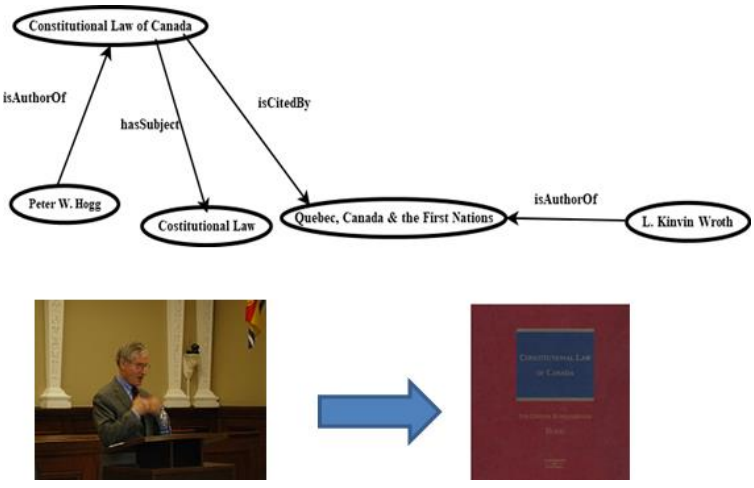
Στο μοντέλο RDF οι δηλώσεις δεδομένων (data statements) μπορούν να απεικονιστούν εννοιολογικά χρησιμοποιώντας μια οπτική αναπαράσταση. Θα εξετάσουμε ένα παράδειγμα δήλωσης δεδομένων σε τριάδα RDF (triple):

Peter W. Hogg → isAuthorOf → Constitutional Law of Canada

Το υποκείμενο (subject) είναι Peter W. Hogg, το predicate είναι isAuthorOf και το αντικείμενο (object) είναι Constitutional Law of Canada. Παραθέτουμε τριάδες οι οποίες σχετίζονται με το βιβλίο του Hogg:

1) Peter W. Hogg → isAuthorOf → Constitutional Law of Canada
2) Constitutional Law of Canada → hasSubject → Constitutional law
3) Constitutional Law of Canada → isCitedBy → Quebec, Canada and the First Nations: The Problem of Secession
4) L. Kinvin Wroth → isAuthorOf → Quebec, Canada and the First Nations: The Problem of Secession

Αυτές οι RDF τριάδες μπορούν να παρασταθούν με έναν γράφο (RDF graph) όπως βλέπουμε στην Εικόνα 2.25 (Μαρινάγη και Σκουρλάς, 2022) (Knight, 2011)



Εικόνα 2.25 Αναπαράσταση RDF τριάδων με γράφο (Μαρινάγη & Σκουρλάς, 2022) (Knight, 2011).

Σύμφωνα με τον Tim Knight (2011), πολλές κοινότητες πληροφοριών συνεισφέρουν πλέον ανοιχτά τα δεδομένα τους στον Ιστό:

« ... υπάρχει μια αυξανόμενη κοινότητα ανθρώπων και οργανισμών που έχουν στη διάθεσή τους μεταδεδομένα τα οποία έχουν δομήσει χρησιμοποιώντας κανόνες Σημασιολογικού Ιστού. Αυτά τα διαφορετικά σύνολα δεδομένων μπορούν να συνδυαστούν σε μια βάση δεδομένων παρέχοντας δυνατότητα δράσης. Αυτά τα σύνολα δεδομένων αναφέρονται ως «συνδεδεμένα δεδομένα» και το Linked Data Cloud είναι μια ανοιχτή και άτυπη αναπαράσταση συμβατών δεδομένων που διατίθενται μέσω του Διαδικτύου. Νέα συνδεδεμένα δεδομένα προστίθενται καθημερινά στο cloud. Κάθε νέος πόρος που προστίθεται στον Ιστό με αυτήν τη μορφή αυξάνει τον αριθμό των δυνατών συνδέσεων δεδομένων μεταξύ των υφιστάμενων συνόλων δεδομένων.

(... there is a growing community of people and organizations who have metadata available to them that they have structured using Semantic Web rules. These disparate sets of data can be combined into a base of actionable data. These sets of data are being referred to as 'linked data,' and the Linked Data Cloud is an open and informal representation of compatible data available over the Internet. New linked data is being added to the cloud daily. Each new resource that is added to the Web in this format increases the number of data connections possible between existing data sets.)

Δείτε σχετικά και το άρθρο,

Knight, T.F. (2011). Break On Through to the Other Side: The Library and Linked Data. TALL Quarterly, 30(1), 1–7, http://hdl.handle.net/10315/6760

Προσφέρονται υπηρεσίες επικύρωσης (validation) των δηλώσεων RDF, ώστε να διαπιστωθεί αν υπάρχουν προβλήματα με τον κώδικα. Για παράδειγμα, μπορεί να χρησιμοποιηθεί η υπηρεσία W3C (World Wide Web Consortium) RDF validation service (Εικόνα 2.26) που βρίσκεται στη διεύθυνση www.w3.org/RDF/Validator. Σε αυτόν τον ιστότοπο, μπορεί να πληκτρολογηθεί ο κώδικας, να γίνει επιλογή της ρύθμισης Triples and Graph από την ιστοσελίδα και γίνει κλικ στο κουμπί Parse RDF.

2.9.4.4 Οργάνωση και έλεγχος δεδομένων, Σημασιολογικός Ιστός, και συνδεδεμένα δεδομένα

Ο Knight (2011) υποστηρίζει ότι υπάρχουν πολλά εμπόδια για τη σύνδεση των «επιμελημένων» (από ειδικούς) βιβλιογραφικών δεδομένων που διαχειρίζονται βιβλιοθήκες και κέντρα τεκμηρίωσης και των δεδομένων που είναι διαθέσιμα στον Παγκόσμιο Ιστό. Αυτή η «ποιοτική απομόνωση» των βιβλιογραφικών δεδομένων είναι ένα πρόβλημα που επηρεάζει την επιτυχή ενσωμάτωση του καταλόγου της βιβλιοθήκης στον Σημασιολογικό Ιστό. Υπογραμμίζει τη σημασία του πλαισίου περιγραφής πόρων (RDF) για το οποίο θεωρεί ότι έχει αναδειχθεί ως το προτιμώμενο μοντέλο δεδομένων για τη διαχείριση συνδεδεμένων δεδομένων στον Ιστό αλλά και τη σημασία του προτύπου Resource Description and Access (RDA). Τέλος, πιστεύει ότι υπάρχουν πάρα πολλοί σύνδεσμοι σε έγγραφα ιστού, αλλά δεν υπάρχει κάποιο νόημα που να σχετίζεται με αυτούς (Μαρινάγη και Σκουρλάς, 2022).

Σχετικά με αυτό το θέμα, η Karen Coyle (2010) κάνει την ακόλουθη παρατήρηση:

Ο σημερινός ιστός είναι ένας ιστός εγγράφων που συνδέονται μεταξύ τους. [...] Οι σύνδεσμοι γενικά πηγαίνουν από ένα σημείο σε ένα έγγραφο σε ένα άλλο έγγραφο, με μερικές σελίδες να αποτελούνται σχεδόν εξ ολοκλήρου από συνδέσμους που χρησιμεύουν ως σημεία εισόδου σε συλλογές. Οι ίδιοι οι σύνδεσμοι, ωστόσο, δεν είναι πολύ ενημερωτικοί: δεν έχουν-εκφράζουν κάποιο νόημα πέρα από τη σύνδεση. Δεν εξηγούν γιατί γίνεται η σύνδεση, ούτε τι μπορεί να σημαίνει ο ίδιος ο σύνδεσμος. Ένας σύνδεσμος θα μπορούσε να είναι μια παραπομπή που υποστηρίζει ένα απόσπασμα, θα μπορούσε να είναι ένα έγγραφο που παρέχει

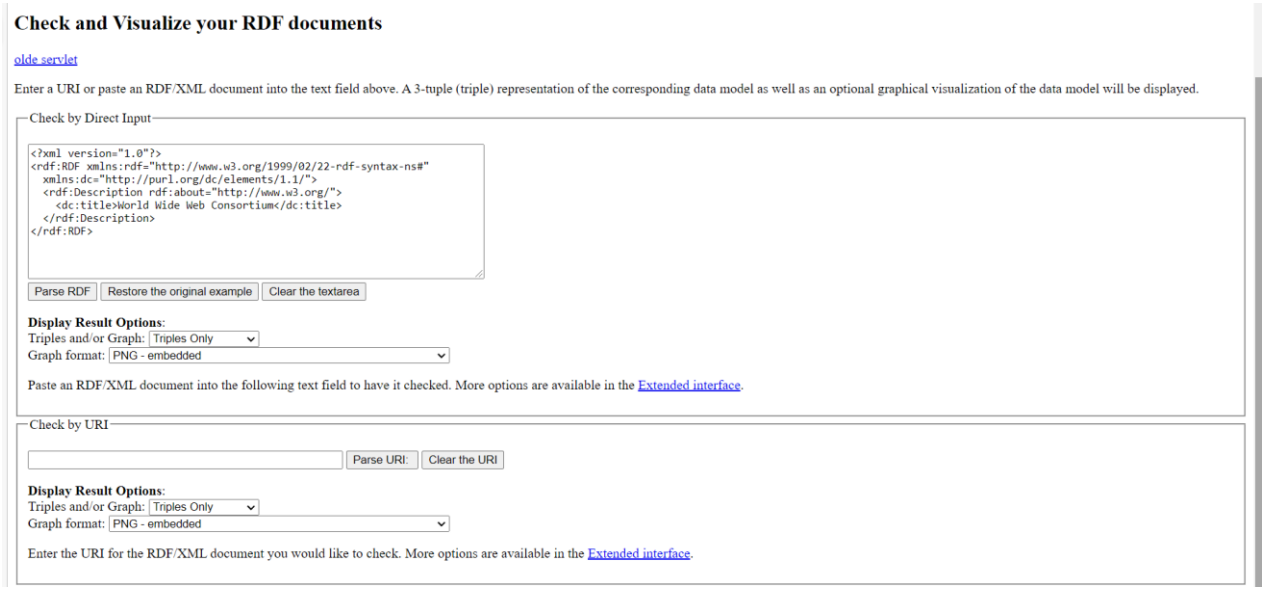
περισσότερες πληροφορίες [...] Σημειώστε επίσης ότι οι σύνδεσμοι πηγαίνουν μόνο προς μια κατεύθυνση, οπότε ένα έγγραφο που έχει συνδέσμους που οδηγούν σε αυτό δεν γνωρίζει αυτούς τους συνδέσμους. Αυτό που μας λείπει και το έργο της οδηγίας προς τον σημασιολογικό ιστό είναι το 'νόημα'».

(Today's web is a web of documents that link to each other. [...] The links generally go from a point in one document to another document, with some pages consisting almost entirely of links that serve as entry points to collections. The links themselves, however, are not very informative: they have no meaning beyond link. They do not explain why you have linked, nor what the link itself could mean. A link could be a citation that supports a quote, it could be [a] document that gives further information …. Note also that links go in only one direction, so a document that has links to it is not aware of those links)

Δείτε σχετικά και το άρθρο,

Coyle, K. (2010). Understanding the Semantic Web: Bibliographic Data and Metadata. Library Technology Reports, 46(1). Chicago, IL : ALA TechSource.

Στην εικόνα 2.26 βλέπουμε την υπηρεσία επικύρωσης XML/RDF (validation service) του W3C (World Wide Web Consortium).



Εικόνα 2.26 Υπηρεσία επικύρωσης XML/RDF του W3C

2.9.4.5 Δεδομένα μεγάλης κλίμακας και σημασιολογικός ιστός. Μία αμφίδρομη σχέση.

Η έρευνα και οι εφαρμογές στη διαχείριση δεδομένων μεγάλης κλίμακας σε περιβάλλον σημασιολογικού ιστού γνωρίζει μια έκρηξη τα τελευταία χρόνια. Παραθέτουμε μια ενδεικτική βιβλιογραφία για την αμφίδρομη σχέση των δύο τομέων.

2.9.4.6 Ενδεικτική βιβλιογραφία

Quboa, Q., Mehandjiev, N. (2017), Creating Intelligent Business Systems by Utilizing Big Data and Semantics, IEEE 19th Conference on Business Informatics, Volume 2, 39-46

Ostrowski, D., Rychtyckyj, N., MacNeille, P., Kim, M. (2016) Integration of Big Data Using Semantic Web Technologies, IEEE 10th International Conference on Semantic Computing, 382-385.

Knoblock, C.A. & Szekely, P. (2015), Exploiting Semantics for Big Data Integration. AI Magazine, 36(1), 25-39

Cure, O., Kerdjoudj, F., Le Duc, C. & Lamolle, M. (2012) On the Potential Integration of an Ontology-Based Data Access Approach in NoSQL Stores, Third International Conference on Emerging Intelligent Data and Web Technologies, 166-173.

Bansal, S.K., Kagemann, S. (2015) Integrating Big Data: A Semantic Extract-Transform-Load Framework, IEEE Computer Society, 48(3), 42-50.

Merelli, I., Pérez-Sánchez, H., Gesing, S., & D'Agostino, D. (2014). Managing, analysing, and integrating big data in medical bioinformatics: open problems and future perspectives. BioMed research international, 2014.

Panahiazar, M., Taslimitehrani, V., Jadhav, Pathak, A.J. (2014) Empowering Personalized Medicine with Big Data and Semantic Web Technology: Promises, Challenges, and Use Cases, IEEE International Conference on Big Data.

Eine, B., Jurisch, M., and Quint, W. (2017) Ontology-Based Big Data Management, Systems, 5(3), 1-14

Xiong, J., Liu, Y., and Liu, W. (2014) Ontology-based Integration and Sharing of Big Data Educational Resources, 11th Web Information System and Application Conference, 245-248.

Muqeem, J.A., Muqeem, A. (2018) Big data and semantic web, challenges and opportunities a survey, International Journal of Engineering & Technology 7(4.42), 631-633

2.10 Πολυμεσικές (multimedia database) ή και πολυτροπικές βάσεις δεδομένων (multimodal database)

Οι όροι πολυμεσικές βάσεις δεδομένων (ή βάσεις πολυμέσων, multimedia database) και πολυτροπικές βάσεις δεδομένων (multimodal database) είναι συνώνυμοι. Συνήθως, στην περίπτωση διεπαφής συστήματος διαχείρισης πολυμέσων, χρησιμοποιείται ο όρος multimodal interface.

Σύμφωνα με λήμμα του Ramesh Jain (2009) στην Encyclopedia of Database Systems (Springer), οι πολυμεσικές βάσεις δεδομένων (Multimedia database) MMDB είναι βάσεις δεδομένων που περιέχουν παραδοσιακά δεδομένα (traditional data) και δεδομένα πολυμέσων (multimedia data) και επιτρέπουν τις κύριες λειτουργίες διαχείρισης αυτών των δεδομένων (παραδοσιακών δεδομένων και πολυμέσων). Διαφέρουν από τις παραδοσιακές βάσεις δεδομένων, οι οποίες περιλαμβάνουν κυρίως αλφαριθμητικά δεδομένα, και απαιτούν όχι μόνο τον καθορισμό πρόσθετων τύπων δεδομένων (για τα πολυμέσα) αλλά και την ανάπτυξη πρόσθετων εξειδικευμένων λειτουργιών για την αποθήκευση, διαχείριση, πρόσβαση και την παρουσίαση (των πολυμέσων).

Συνήθως, «οι βάσεις δεδομένων που διαχειρίζονται εικόνες, ήχο και βίντεο εκτός από τα μεταδεδομένα που σχετίζονται με αυτές και άλλες αλφαριθμητικές πληροφορίες ονομάζονται βάσεις δεδομένων πολυμέσων.» ("databases that manage images, audio, and video in addition to metadata related to these and other alphanumeric information are called multimedia databases").

Όταν οι βάσεις δεδομένων περιέχουν μόνο εικόνες ή ήχο ή βίντεο, ονομάζονται αντίστοιχα βάσεις δεδομένων εικόνας, βάσεις δεδομένων ήχου και βάσεις δεδομένων βίντεο.

Σε ένα σύστημα βάσης δεδομένων πολυμέσων (multimedia database system) διακρίνουμε τέσσερις συνεργαζόμενες συνιστώσες οι οποίες και παρέχουν την απαιτούμενη λειτουργικότητα.

Ανάλυση δεδομένων και εξαγωγή χαρακτηριστικών (Data Analysis and Feature Extraction):

Η αποθήκευση δεδομένων πολυμέσων στα παραδοσιακά ΣΔΒΔ γίνεται με χρήση τύπου δεδομένων BLOB και δεν επιτρέπει ερωτήματα σχετικά με το περιεχόμενο των δεδομένων. Σε ένα σύστημα πολυμεσικής βάσης δεδομένων, MMDB, τα δεδομένα αναλύονται για να εξαχθούν χαρακτηριστικά (features) τα οποία μπορούν να χρησιμοποιηθούν για την «εξαγωγή της απαιτούμενης σημασιολογίας» ("to derive the required semantics"). Τα χαρακτηριστικά εξαρτώνται από τον τύπο των δεδομένων ή/και τον τομέα εφαρμογής. Τα χαρακτηριστικά χαμηλού επιπέδου είναι γενικά και δεν εξαρτώνται από τον τομέα εφαρμογής (application domain), π.χ., ιστογράμματα χρώματος (color histograms) και χαρακτηριστικά υφής (texture features) για εικόνες. Τα χαρακτηριστικά υψηλού επιπέδου συνδέονται άμεσα με τον τομέα εφαρμογής, όπως το σχήμα του όγκου καρκινοπαθούς (shape of the tumor).

Ερμηνεία δεδομένων (διερμήνευση) ενός τομέα (Domain Knowledge and Interpretation):

Η ερμηνεία δεδομένων πολυμέσων απαιτεί γνώση του τομέα. Για παράδειγμα, η εικόνα από σύστημα σχεδιασμού ακτινοθεραπείας (radiotherapy treatment planning) μπορεί να μελετηθεί από ειδικευμένους ιατρούς. Επισημαίνεται ότι η απαιτούμενη γνώση για την ερμηνεία των δεδομένων πολυμέσων δεν είναι μόνο η παραδοσιακή γνώση τομέα. Οι οντολογίες, η επεξεργασία και κατανόηση κειμένου φυσικής γλώσσας και άλλες παρόμοιες τεχνικές είναι καλά αναπτυγμένες για την αναπαράσταση της παραδοσιακής γνώσης αλλά δεν επαρκούν από μόνες τους. Απαιτούνται και «μοντέλα που εξαρτώνται από τα μέσα τα οποία απαιτούν εξελιγμένες προσεγγίσεις ταξινόμησης» ("media dependent models that require sophisticated classification approaches"). Επιπλέον, στα αρχεία ήχου και τα βίντεο που περιλαμβάνονται στα δεδομένα, θέλουμε να ανιχνεύονται συμβάντα-γεγονότα (events), δηλαδή απαιτούνται λειτουργίες που εξαρτώνται από το χρόνο επεξεργασίας.

Αλληλεπίδραση και διεπαφή χρήστη (Interaction and User Interface):

Οι γνωστές παραδοσιακές διεπαφές των βάσεων δεδομένων και των μηχανών αναζήτησης δεν είναι επαρκείς σε εφαρμογές πολυμέσων. Αντιγράφουμε από το λήμμα:

«Οι λέξεις κλειδιά ή τα ονόματα αντικειμένων μπορούν να χρησιμοποιηθούν σε ορισμένες περιορισμένες αναζητήσεις, αλλά πολλές εφαρμογές απαιτούν έννοιες και ιδέες που απαιτούν συνεχείς αλληλεπιδράσεις και διαδοχική βελτίωση των ερωτημάτων σε αυτό που ονομάζεται αναδυόμενο περιβάλλον σημασιολογίας» ("Using keywords or names of objects, some limited searches can be performed, but many applications require concepts and ideas that require both continuous interactions and successive refinement of queries in what is called emergent semantics environment").

Αποθήκευση, αντιστοίχιση και ευρετηρίαση (Storage, Matching, and Indexing):

Σε πολλές εφαρμογές, το μέγεθος των δεδομένων πολυμέσων απαιτεί ιδιαίτερη διαχείριση. Τα παραδοσιακά ΣΔΒΔ υποστηρίζουν τον τύπο δεδομένων BLOB αλλά συχνά η αποθήκευση των αρχείων βίντεο συνδέεται με περιορισμούς και έχει μεγάλο κόστος. Μια συνηθισμένη πρακτική είναι να αποθηκεύουμε στη βάση δεδομένων μόνο τα ονόματα των αρχείων δεδομένων πολυμέσων (τα οποία μπορεί να υπάρχουν σε κάποιο διανομέα-server ή σε κατανεμημένη βάση δεδομένων ή στο νέφος) και επιπλέον, τα χαρακτηριστικά (features) που εξάγουμε από τα δεδομένα. Επισημαίνεται ότι για ένα αρχείο δεδομένων πολυμέσων, π.χ., ένα βίντεο μπορούμε να έχουμε ένα πολύ μεγάλο αριθμό χαρακτηριστικών που πρέπει να αποθηκευτούν, π.χ., χιλιάδες ή εκατοντάδες χιλιάδες. Όμως αυτά τα χαρακτηριστικά μπορούν να χρησιμοποιηθούν την αναζήτηση σωστών αποτελεσμάτων

Βλέπε σχετικά, Ramesh Jain (2009) Multimedia Databases, pp. 1817-1820, Encyclopedia of Database Systems, Editors: Ling Liu, M. Tamer Özsu

10.1007/978-0-387-39940-9_1006.pdf (springer.com)

Multimedia Databases | SpringerLink

Οι βάσεις πολυμέσων ή και βάσεις πολλαπλών μέσων επηρεάζουν και θα επηρεάσουν σημαντικά τον κόσμο των εφαρμογών των βάσεων δεδομένων. Βλέπε ενδεικτικά και την πρόβλεψη στο σύγγραμμα,

Ullman, Widom, A first course in Database system, Prentice-Hall, pp. 14-23.

Η συζήτηση για τη διαχείριση των πολυμέσων βρίσκει πολλές εφαρμογές στα μεγάλα δεδομένα και τις βάσεις δεδομένων. Ήδη, όταν μιλάμε για δεδομένα (data), ακόμη και στην περίπτωση σχεσιακών βάσεων, έχουμε τη διευρυμένη αντίληψη που φαίνεται στην εικόνα 2.27. Πιο συγκεκριμένα, οι κοινές μορφές πολυμέσων (multimedia) περιλαμβάνουν, εκτός από την παλαιότερη γνωστή μορφή, το κείμενο, και τα παρακάτω:

1) video - βίντεο πχ. video clip σε MP4
2) audio - ήχος πχ. τραγούδι σε MP3.
3) documents - τεκμήρια (ή έγγραφα), πχ. ένα άρθρο που περιλαμβάνει κείμενο και εικόνες
4) images - εικόνες πχ. μία φωτογραφία
5) drawings - σχέδια πχ. ένα σκίτσο κ.τ.λ.

Οι μορφές αυτές υπάρχουν σε διάφορες κωδικοποιήσεις (encodings). Ένα χαρακτηριστικό όλων των μορφοτύπων (μορφών, format) είναι ότι συχνά οι απαιτήσεις αποθήκευσης τους είναι πολύ μεγαλύτερες από αυτές των παλαιότερων μορφών δεδομένων και βέβαια ποικίλουν από μέσο σε μέσο. Η αποθήκευση και η ανάκτηση πολυμέσων υποχρεώνει τα ΣΔΒΔ να επεκταθούν σε διάφορες κατευθύνσεις.

Παράδειγμα

Μια βάση εικόνων (image database), μπορεί να περιλαμβάνει μαγνητικές τομογραφίες (MRI), ακτινογραφίες (X rays), κείμενο (περιγραφή συμπτωμάτων), κ.λπ. Οι εικόνες μπορεί να είναι σε διάφορες μορφές (formats): GIF, TIFF, JPEG, PCX κ.λπ.

Η αναζήτηση (search) των εικόνων θα μπορούσε να βασίζεται στο κείμενο (textual input), π.χ. αναζήτηση εικόνων που έχουν χαρακτηριστεί με κάποιο σύμπτωμα)

Θα μπορούσε να βασίζεται στη σύγκριση εικόνων, π.χ., εύρεση εικόνων στις οποίες κυριαρχεί ένα χρώμα που υπάρχει σε συγκεκριμένη εικόνα.

Μία αντικειμενοστρεφής προσέγγιση (η οποία εστιάζει στο περιεχόμενο των δεδομένων πολυμέσων), σε συνδυασμό με τα σχεσιακά συστήματα (τα οποία διαχειρίζονται τα δομημένα δεδομένα, συνοδευτικό κείμενο κ.λπ.) και κυρίως η τεχνολογία των μεγάλων δεδομένων παρέχουν νέες δυνατότητες στους ενδιαφερόμενους χρήστες να υποστηρίξουν τις απαιτούμενες λειτουργίες (functions) στα πολυμέσα. Το μέγεθος των αντικειμένων των πολυμέσων υποχρεώνει τα συστήματα διαχείρισης να τροποποιήσουν τη διαχείριση της αποθήκευσης (storage managment) ώστε να διαχειρίζονται αντικείμενα μεγέθους gigabytes και άνω.

Ένα ΣΔΒΔ πολυμέσων (Multimedia DB, MMDB) πρέπει να μπορεί να απαντά μέσα από ένα διάλογο με το χρήστη και να δείχνει το σύνολο ή μέρος του ζητούμενου. Για παράδειγμα, αν ένα μέρος της απάντησης στο ερώτημα του χρήστη είναι ένα βίντεο, που απαιτεί κάποιο χρόνο μετάδοσης αλλά και μεγάλο χρόνο εκτέλεσης (to play) για να το δει ο χρήστης, τότε πρέπει, ενδεχομένως, ο χρήστης να μπορεί να αποφασίζει για ποιο συγκεκριμένο τμήμα του βίντεο (videoclip) ενδιαφέρεται, πότε θέλει να "κατέβει" το βίντεο κ.λπ.

Εικόνα 2.27 Τύποι δεδομένων πολυμέσων που συναντώνται στις εφαρμογές βάσεων δεδομένων πολυμέσων (video, documents, text, audio, images)

Η αναζήτηση εικόνων θεωρείται ένα δύσκολο πρόβλημα το οποίο προσελκύει το ενδιαφέρον της ερευνητικής κοινότητας από τις αρχές της δεκαετίας του 90. Η αναζήτηση μπορεί να βασίζεται σε οπτικές ιδιότητες (visual properties) (σχετικά εύκολη διαδικασία) ή στο περιεχόμενο (δύσκολη διαδικασία). Στη βιβλιογραφία, γίνεται αναφορά στο πρόβλημα του σημασιολογικού χάσματος (semantic gap problem) το οποίο αποτελεί κύριο εμπόδιο για τη βελτίωση της απόδοσης ενός συστήματος ανάκτησης εικόνων που βασίζεται στο περιεχόμενο (Content Based Image Retrieval) CBIR. Το όνομα του προβλήματος οφείλεται ακριβώς στο χάσμα μεταξύ των οπτικών χαρακτηριστικών (visual features) χαμηλού επιπέδου και της υψηλού επιπέδου σημασιολογικής κατανόησης των εικόνων (semantic understanding). Μία δημοφιλής προσέγγιση εστιάζει όχι στην περιγραφή του συνόλου του περιεχομένου (global content) της εικόνας αλλά στην τοπική περιγραφή περιεχομένου ανά περιοχές. Δηλαδή, μιλάμε για ανάκτηση εικόνας βάσει περιοχής (region-based image retrieval) ή ανάκτηση εικόνας βάσει αντικειμένων της (object-based image retrieval) (Wei Huang, Yan Gao, Kap Luk Chan, 2010).Βλέπε σχετικά και το άρθρο,

Huang, W., Gao, Y., & Chan, K. L. (2010). A review of region-based image retrieval. Journal of Signal Processing Systems, 59(2), 143-161.

Σημαντική έρευνα στοχεύει στην αναγνώριση κειμένου σε φωτογραφίες, προσώπων σε εικόνα ή βίντεο κ.λπ.

2.11 Συστήματα Ανάκτησης Πληροφοριών-ΣΑΠ (Information Retrieval Systems), Συστήματα Ανάκτησης Κειμένου (Text Retrieval) και Μηχανές Αναζήτησης

Τα Συστήματα Ανάκτησης Πληροφορίας (ΣΑΠ) βασίζονται σε μοντέλα ανάκτησης (retrieval models).

Ένα μοντέλο ανάκτησης πληροφορίας μπορεί να οριστεί ως η τετράδα $(D, Q, F, R(q_i, d_j))$ όπου:

- D είναι το σύνολο των λογικών αναπαραστάσεων των εγγράφων/κειμένων
- Q είναι το σύνολο των ερωτημάτων που περιγράφουν τις ανάγκες πληροφόρησης του χρήστη
- F είναι το πλαίσιο για τη μοντελοποίηση των εγγράφων, των ερωτημάτων και των σχέσεων μεταξύ τους.

$R(q_i, d_j)$ είναι μία συνάρτηση κατάταξης, η οποία συνδέει έναν πραγματικό αριθμό με ένα ερώτημα $q_i \in Q$ και μία αναπαράσταση κειμένου $d_j \in D$. Η κατάταξη αυτή ορίζει μία διάταξη των κειμένων με βάση το ερώτημα q_i

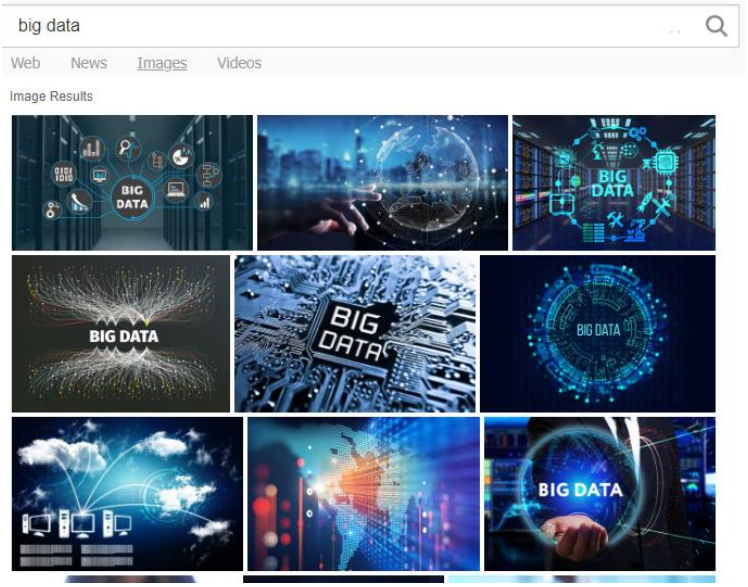
Η κατάταξη «εκφράζει» την ομοιότητα (similarity) των εγγράφων με το συγκεκριμένο ερώτημα. Η ομοιότητα ενός εγγράφου με ένα ερώτημα (similarity of a document against a submitted query) αποτελεί πεδίο πολυετούς, συνεχούς έρευνας που γνωρίζει άνθηση και στις μέρες μας.

Κλασσικά μοντέλα ανάκτησης είναι το λογικό (boolean), το πιθανοθεωρητικό ή Μπεϊσιανό μοντέλο ταξινόμησης (Bayes classification model) και το μοντέλο διανυσματικού χώρου (vector space model). Κοινό χαρακτηριστικό αυτών των μοντέλων είναι ότι αναπαριστούν τα κείμενα ως σύνολα όρων που περιγράφουν το περιεχόμενό τους. Οι όροι αυτοί ονομάζονται όροι ευρετηρίου (index terms).

Ειδική μνεία πρέπει να γίνει για το μοντέλο διανυσματικού χώρου και το ιστορικό σύστημα Smart (Shalton) το οποίο επηρέασε πάρα πολύ την έρευνα και εξέλιξη του τομέα της ανάκτησης.

Εκτός από τα τρία παραπάνω κλασσικά μοντέλα ανάκτησης έχουν προταθεί και πολλά άλλα μοντέλα τα οποία, συνήθως, αποτελούν επεκτάσεις των κλασσικών. Για παράδειγμα τα μοντέλα που βασίζονται στην "Αρχή Πιθανοθεωρητικής Κατάταξης" (Probability Ranking Principle) με χαρακτηριστικό παράδειγμα το ιστορικό σύστημα INQUERY. Ιδιαίτερο ενδιαφέρον υπάρχει για τη σχεδίαση συστημάτων που συνδυάζουν αποτελέσματα αναζητήσεων με βάση διαφορετικά μοντέλα ανάκτησης. Παράδειγμα συστημάτων που συνδυάζουν ετερογενείς αναζητήσεις είναι οι μηχανές "μετα-αναζήτησης" στον Παγκόσμιο Ιστό, π.χ., η ιστορική μηχανή MetaCrawler (http://metacrawler.com).

Στην εικόνα 2.28 μπορείτε να δείτε παράδειγμα απάντησης της μηχανής στην αναζήτηση εικόνων με θέμα Big Data.



Εικόνα 2.28 Απάντηση της ιστορικής μηχανής MetaCrawler (http://metacrawler.com) στην αναζήτηση εικόνων με θέμα Big Data

2.11.1 Μοντέλο διανυσματικού χώρου

Στην περίπτωση του δημοφιλούς μοντέλου διανυσματικού χώρου (vector space model) καθορίζεται ένα σύνολο (data set) n μοναδικών όρων (unique terms) οι οποίοι ονομάζονται όροι ευρετηρίου (index terms) της συλλογής των εγγράφων (document collection) και κάθε έγγραφο μπορεί να αναπαρασταθεί σαν ένα διάνυσμα (vector),

$$(T_1, T_2, \ldots, T_n)$$

όπου T_i=1, αν ο όρος ευρετηρίου (index term) i υπάρχει στο έγγραφο αλλιώς είναι ίσο με 0.

Τόσο οι ερωτήσεις όσο και τα έγγραφα μπορούν να αναπαρασταθούν με τον ίδιο τρόπο. Τα διανύσματα των εγγράφων και των ερωτημάτων αποτελούν n-διάστατο διανυσματικό χώρο (n-dimensional vector space). Μια λειτουργία «ταιριάσματος» (δηλαδή προσδιορισμού του βαθμού ομοιότητας) δύο διανυσμάτων (A vector matching operation), βασιζόμενη στο «νόμο του συνημιτόνου» (based on the cosine correlation) χρησιμοποιείται για να μετρήσει το συνημίτονο της γωνίας που σχηματίζουν τα δύο διανύσματα (to measure the cosine of the angle between vectors) και να υπολογίσει τον βαθμό ομοιότητας μεταξύ τους (to compute the similarity). Η επόμενη εξίσωση (βλέπε π.χ., Karanikolas, Skourlas, 2000, 2002) δίνει μία γνωστή μέθοδο για τη μέτρηση του βαθμού ομοιότητας (a well-known method to measure the similarity) ενός εγγράφου (document) Di με το ερώτημα (query) Q:

$$S(D_i, Q) = \frac{\sum_{j=1}^{n} q_j t_{ij}}{\sqrt{\sum_{j=1}^{n} q_j^2 \cdot \sum_{j-1}^{n} t_{ij}^2}} = \frac{\sum_{j=1}^{n} q_j t_{ij}}{L_Q \cdot L_{D_i}}$$

όπου n είναι ο αριθμός των όρων ευρετηρίου (index terms) που χρησιμοποιούνται στη συλλογή των εγγράφων, t_{ij} είναι το βάρος – weight (ο όρος αποδίδεται στα ελληνικά και ως συνάφεια) του όρου j στο έγγραφο D_i και q_j είναι το βάρος (weight) του όρου j στο ερώτημα (query).

Το βάρος ακριβώς εκφράζει πόσο αντιπροσωπευτικός είναι ένας όρος ευρετηρίου για το έγγραφο ή για το ερώτημα.

Οι επόμενες δύο εξισώσεις μπορούν να χρησιμοποιηθούν για να μετρήσουν (to measure) τους όρους (terms) t_{ij} και q_j:

$$t_{ij} = 0.5 + 0.5 \cdot \frac{F_{ij}}{\max F_i}$$

$$q_j = \log_2\left(\frac{N}{DOCFREQ_j}\right)$$

όπου F_{ij} είναι η συχνότητα (frequency) του όρου j στο έγγραφο D_i, $\max F_i$ είναι η μέγιστη συχνότητα (maximum frequency) των όρων στο έγγραφο D_i, N είναι ο αριθμός των εγγράφων στη συλλογή και $DOCFREQ_j$ είναι ο αριθμός των εγγράφων που περιλαμβάνουν τον όρο ευρετηρίου (index term) j.

Βλέπε σχετικά και τα άρθρα

Karanikolas, N.N. and Skourlas, C. (2000). Computer assisted information resources navigation. Medical Informatics and the Internet in Medicine, 25(2), 133-146.

Karanikolas, N.N. and Skourlas, C., 2002. Automatic Diagnosis Classification of Patient Discharge Letters. In MIE'2002, 23rd International Congress of the European Federation for Medical Informatics, Budapest.

Σημείωση

Στη βιβλιογραφία αντί «όρων ευρετηρίου», σε παρόμοιες περιπτώσεις ανάκτησης εγγράφων/κειμένου (documents/text) και ανάλογα με το περιεχόμενο της έρευνας, χρησιμοποιούνται οι όροι λέξεις κλειδιά (keywords), μη-ελεγχόμενοι όροι κλειδιά (uncontrolled terms), φράσεις κλειδιά (key phrases).

2.11.2 Αποσαφήνιση σημαντικών εννοιών των συστημάτων ανάκτησης πληροφοριών (Information Retrieval System) και των μηχανών αναζήτησης (search engines) με χρήση παραδειγμάτων

Θα προσπαθήσουμε να παρουσιάσουμε με απλά παραδείγματα και να κατανοήσουμε τις βασικές ιδέες που ενσωματώνονται σε ένα σύστημα ανάκτησης πληροφοριών (ΣΑΠ) και αξιοποιούνται στις μηχανές αναζήτησης. Θα θεωρήσουμε μία βιβλιογραφική βάση δεδομένων που είναι οργανωμένη με χρήση ενός ΣΑΠ. Για παράδειγμα, έστω ένας κατάλογος βιβλίων.

Πως είναι οργανωμένη η βάση δεδομένων

Υπάρχει ένας κύριος πίνακας (ή αρχείο), έστω με όνομα MAIN_FILE, που έχει τις καταχωρημένες εγγραφές (records) βιβλίων.

MAIN_FILE

1	Date, An introduction to database systems, volume 1, Addison-Wesley, 2004
2	Date, Εισαγωγή στα συστήματα βάσεων δεδομένων τόμος Α, Κλειδάριθμος, 2008
3	Date, Εισαγωγή στα συστήματα βάσεων δεδομένων τόμος Β, Κλειδάριθμος, 2008
4	Ullman, Widom, A first course in database systems, Prentice-Hall, 2014
5	Elmasri, Navathe, Fundamentals of database systems, The Benjamin/Cummings Publishing Co., 2017
6	Elmasri, Θεμελιώδεις αρχές συστημάτων βάσεων δεδομένων, τόμος Α, Δίαυλος, 2016
7	Elmasri, Θεμελιώδεις αρχές συστημάτων βάσεων δεδομένων, τόμος Β, Δίαυλος, 2016
8	Date, Relational database writings 1994-1997, Addison-Wesley

Η βάση συμπληρώνεται από σειρά αντεστραμμένων αρχείων (inverted files) τα οποία βοηθούν αποτελεσματικά στη γρήγορη αναζήτηση και εμφάνιση στοιχείων. Για παράδειγμα, θα έχουμε δύο αντεστραμμένα αρχεία:

Authors_Inverted_File

Date #1, #2, #3, #8
Elmasri #5, #6, #7
Navathe #5
Ullman #4
Widom #4

Publishers_Inverted_File

Addison-Wesley #1, #8
Benjamin/Cummings #5
Prentice-Hall #4
Δίαυλος #6, #7

Κλειδάριθμος #2, #3

Σημείωση

Βέβαια, για να κρατήσουμε όσο πιο απλή και συνοπτική γίνεται τη συζήτηση μας, παραλείψαμε, ίσως το πλέον σημαντικό αντεστραμμένο αρχείο, στην πράξη, το αρχείο με τα θέματα (ακριβέστερα με τις λέξεις κλειδιά - keywords).

Εισαγωγή νέας εγγραφής

Αν θελήσουμε να εισάγουμε στη βάση ένα νέο βιβλίο τότε το ΣΑΠ θα δημιουργήσει την ένατη εγγραφή στο MAIN_FILE και θα ενημερώσει αυτόματα και όλα τα αντεστραμμένα αρχεία.

1	Date, An introduction to database systems, volume 1, Addison-Wesley, 2004
2	Date, Εισαγωγή στα συστήματα βάσεων δεδομένων τόμος Α, Κλειδάριθμος, 2008
3	Date, Εισαγωγή στα συστήματα βάσεων δεδομένων τόμος Β, Κλειδάριθμος, 2008
4	Ullman, Widom, A first course in database systems, Prentice-Hall, 2014
5	Elmasri, Navathe, Fundamentals of database systems, The Benjamin/Cummings Publishing Co. 2017
6	Elmasri, Θεμελιώδεις αρχές συστημάτων βάσεων δεδομένων, τόμος Α, Δίαυλος, 2016
7	Elmasri, Θεμελιώδεις αρχές συστημάτων βάσεων δεδομένων, τόμος Β, Δίαυλος, 2016
8	Date, Relational database writings 1994-1997, Addison-Wesley
9	Date, Foundation for object/relational databases, Addison-Wesley

Authors_Inverted_File

Date #1, #2, #3, #8, #9
Elmasri #5, #6, #7
Navathe #5
Ullman #4
Widom #4

Publishers_Inverted_File

Addison-Wesley #1, #8, #9
Benjamin/Cummings #5
Prentice-Hall #4
Δίαυλος #6, #7
Κλειδάριθμος #2, #3

Ανάκτηση

Αν υποτεθεί ότι ο χρήστης ενδιαφέρεται να ανακτήσει κάποια πληροφορία τότε το ΣΑΠ ψάχνει πρώτα στα σχετικά αντεστραμμένα αρχεία. Για παράδειγμα, αν ο χρήστης ενδιαφέρεται για τα βιβλία του Ullman που

είναι καταχωρημένα στη βάση δεδομένων τότε το ΣΑΠ ψάχνει στο σχετικό αντεστραμμένο αρχείο Authors_Inverted_File, διαβάζει την εγγραφή του Ullman και στη συνέχεια διαβάζει το MAIN_FILE με τιμή κλειδιού ίση με 4.

Αν ο χρήστης ενδιαφέρεται για βιβλία του Date στον εκδοτικό οίκο Κλειδάριθμο που είναι καταχωρημένα στη βάση δεδομένων τότε το ΣΑΠ ψάχνει στα σχετικά αντεστραμμένα αρχεία Authors_Inverted_File, Publishers_Inverted_File και διαβάζει τις σχετικές εγγραφές.

Από τις δύο εγγραφές που διάβασε, δηλαδή τις

Date #1, #2, #3, #8, #9

Κλειδάριθμος #2, #3

υπολογίζει, με σύγκριση, ότι οι εγγραφές του MAIN_FILE που ενδιαφέρουν είναι οι #2, #3. Τότε, το ΣΑΠ διαβάζει διαδοχικά το MAIN_FILE με τιμή κλειδιού ίση με 2 και 3 και δείχνει τις εγγραφές που ζήτησε ο χρήστης.

Διαγραφή

Αν υποτεθεί ότι θέλουμε να διαγράψουμε το βιβλίο με αριθμό 2, τότε το ΣΑΠ διαγράφει το βιβλίο όχι μόνο από το MAIN_FILE αλλά και από τα αντεστραμμένα αρχεία.

Η διεπαφή του χρήστη

Παλαιότερα, συνήθως, ο χρήστης επικοινωνούσε με γλώσσα εντολών ή μενού και επιλογή τιμών. Σήμερα, υπάρχουν πάρα πολλές ιδέες πειραματικές και εμπορικά εφαρμοσμένες στο θέμα αυτό. Ο καλύτερος τρόπος να αποκτήσετε μία εμπειρία για το θέμα της διεπαφής είναι να χρησιμοποιήσετε τις μηχανές αναζήτησης του διαδικτύου.

Παραθέτουμε παράδειγμα χρήσης με γλώσσα εντολών.

FIND (AU=Date OR AU= Ullman) AND PU=Κλειδάριθμος

SHOW AU; TI; PU; PD

όπου AU=author, PU=publisher, TI=title, PD=publication date.

Παραθέτουμε και παράδειγμα χρήσης μενού και επιλογής τιμών,

Βασικό μενού
1. Συγγραφείς
2. Εκδότες
…
Έξοδος

Επιλογή ονόματος συγγραφέα
Date
Elmasri
Navathe
Ullman
Widow
....

Επιλογή Εκδοτικού οίκου
Addisson-Wesley
Benjamin/Cummings
Prentice-Hall
...

Οργάνωση Βάσεων πλήρους κειμένου

Θα προσπαθήσουμε να παρουσιάσουμε και να κατανοήσουμε τις βασικές ιδέες που χρησιμοποιούνται σε ένα ΣΑΠ αλλά και στις μηχανές αναζήτησης (search engines) με τη βοήθεια ενός ακόμη παραδείγματος. Έστω μία βάση πλήρους κειμένου που είναι οργανωμένη με χρήση ενός ΣΑΠ.

Πως είναι οργανωμένη η βάση

Υπάρχει ένα κύριο αρχείο, έστω με όνομα MAIN_FILE που έχει τις καταχωρημένες εγγραφές των κειμένων. Κάθε αριθμός αντιστοιχεί σε ένα κείμενο.

Acc no	Text document
1	Εκτός από το αρχείο MAIN με τα κείμενα, έχουμε και ένα ή περισσότερα αντεστραμμένα αρχεία.
2	Για κάθε σημαντική λέξη του κειμένου και αφού το ΣΑΠ προσδιορίσει το θέμα (stem) της γίνεται μία εγγραφή στο αντεστραμμένο αρχείο.
3	Για παράδειγμα, μέχρι τώρα στο κείμενο μας θα πρέπει να γίνουν εγγραφές για τις σημαντικές λέξεις «εκτός», «αρχείο», «πλήρους», «κειμένου», «υπάρχει», «αντεστραμμένο», «σημαντική», «λέξη», «ΣΑΠ", «προσδιορίσει» κ.λπ.
4	Λέξεις όπως «από", «το", "και" και άλλες «αγνοούνται» σαν περιττές (μη σημαντικές). Δηλαδή, δεν περιλαμβάνονται στα αντεστραμμένα ευρετήρια. Είναι όλες καταγεγραμμένες σε ειδικό αρχείο που είναι γνωστό στη βιβλιογραφία ως αρχείο stopwords.
5	Για τις σημαντικές λέξεις καταγράφεται στο αντεστραμμένο αρχείο ο αριθμός του κειμένου, η πρόταση στην οποία ανήκει η λέξη μέσα στο κείμενο, οι γειτονικές σημαντικές λέξεις κ.λπ.
6	Για παράδειγμα, η εγγραφή για τη σημαντική λέξη «ΕΓΓΡΑΦ-Η» περιλαμβάνει τα στοιχεία: #2, #3, #6, #7, δηλαδή η λέξη αυτή υπάρχει στα κείμενα που έχουν αυτούς τους αριθμούς.
7	Αν απορείτε, πως προέκυψε ότι η λέξη «ΕΓΓΡΑΦ-Η» ανήκει στο κείμενο με #3 θυμηθείτε ότι οι λέξεις «ΕΓΓΡΑΦ-Η» και «ΕΓΓΡΑΦ-ΕΣ» έχουν το ίδιο θέμα (stem) και, επίσης, ότι αντί για τη λέξη καταγράφουμε το θέμα της στο αντεστραμμένο αρχείο.
…….	……

Οι υπόλοιπες λεπτομέρειες του παραδείγματος μπορούν να συμπληρωθούν, όπως και στο προηγούμενο παράδειγμα.

Η συνδυασμένη αναζήτηση όρων, π.χ., αναζήτηση με δύο όρους, υλοποιείται με διάφορες τεχνικές. Για παράδειγμα γίνεται χρήση bit maps όπου με 0 δηλώνεται η απουσία και με 1 η παρουσία μιας λέξης σε ένα κείμενο. Αν όλα τα κείμενα της βάσης κειμένων είναι 7 τότε έχουμε:

όροι	Bit map
…….	……..
ΕΓΓΡΑΦ	0110011
……	……...

ΣΑΠ	0000010
……..	……

Η αναζήτηση για κείμενα με όρους ΕΓΓΡΑΦ και ΣΑΠ γίνεται με ένα λογικό AND των δύο bit maps και η απάντηση είναι το κείμενο 3.

2.11.2.1 Αξιολόγηση αποτελεσμάτων ανάκτησης. Τεχνική (μέτρο) TF−IDF

Ας υποθέσουμε ότι ο χρήστης πληκτρολογεί το ακόλουθο αίτημα: "“RapidMiner books that describe text mining.” (Βιβλία RapidMiner που περιγράφουν την εξόρυξη κειμένου). Οι μηχανές αναζήτησης λειτουργούν με την ακόλουθη βασική λογική:

1) Δώστε μεγάλη βαρύτητα σε εκείνες τις λέξεις-κλειδιά που είναι σχετικά "σπάνιες".
2) Δώστε μεγάλη βαρύτητα σε εκείνες τις ιστοσελίδες που περιέχουν μεγάλο αριθμό περιπτώσεων των «σπάνιων» λέξεων-κλειδιών.

Στο κείμενο θα λέγαμε ότι οι «σπάνιες» λέξεις κλειδιά είναι “RapidMiner” και “mining” με την έννοια ότι οι λέξεις “that,” “books,” “describe,” και “text” πιθανότατα εμφανίζονται σε ένα πολύ μεγαλύτερο αριθμό σελίδων (web pages) από τις λέξεις “RapidMiner” και “mining”.

Επομένως θα θέλαμε οι πιο «σπάνιες» λέξεις να λάβουν υψηλότερη βαθμολογία (higher rating). Από τις σελίδες που περιέχουν τις σπάνιες λέξεις-κλειδιά μας ενδιαφέρει να έχουν μεγαλύτερη στάθμιση (high weight) μόνο οι σελίδες που έχουν τον μεγαλύτερο αριθμό εμφάνισης των σπάνιων λέξεων-κλειδιών (επειδή θεωρούμε πιθανότερο να είναι σχετικές με το αίτημα). Δηλαδή, μας ενδιαφέρουν οι σελίδες με τη μεγαλύτερη στάθμιση και αυτές είναι οι σελίδες για τις οποίες το γινόμενο αυτών των δύο βαρών είναι το υψηλότερο.

Η τεχνική υπολογισμού αυτής της στάθμισης ονομάζεται TF–IDF (Term Frequency–Inverse Document Frequency). Ο υπολογισμός του TF είναι απλός, είναι ο λόγος του αριθμού εμφανίσεων μιας λέξης-κλειδιού k σε ένα δεδομένο έγγραφο, n_k , προς τον συνολικό αριθμό n των όρων στο έγγραφο:

$$TF = n_k/n$$

Επομένως, στο παράδειγμα, η κοινή λέξη "that" θα έχει αρκετά υψηλότερη βαθμολογία TF από τη λέξη "RapidMiner".

Η βαθμολογία IDF ορίζεται ως εξής:

$$IDF = \log_2 (N/N_k)$$

όπου N είναι ο αριθμός των εγγράφων που εξετάζονται, δηλαδή σε περιβάλλον μηχανής αναζήτησης N_k είναι ο αριθμός όλων των ιστοσελίδων που έχουν καταχωρηθεί στο ευρετήριο της. Στις περισσότερες αναλύσεις το N είναι ο αριθμός των εγγράφων που προσπαθούμε να ανακτήσουμε-εξορύξουμε και N_k είναι ο αριθμός των εγγράφων που περιέχουν τη λέξη-κλειδί, k. Η λέξη "that" θα εμφανίζεται σχεδόν (αν όχι οπωσδήποτε) σε κάθε έγγραφο και έτσι η αναλογία (N/ N_k) θα είναι κοντά στο 1 και η βαθμολογία IDF θα ήταν κοντά στο μηδέν. Αυτό δεν ισχύει για τη βαθμολογία μιας λέξης, όπως “RapidMiner”. Δηλαδή, η βαθμολογία IDF είναι υψηλότερη στις «σπάνιες» λέξεις. Τελικά,

$$TF−IDF = n_k /n * \log_2 (N/N_k)$$

Οι βαθμολογίες TF–IDF υπολογίζονται για κάθε λέξη στο σύνολο των εγγράφων. Αυτός ο υπολογισμός είναι μια τυπική τεχνική η οποία εφαρμόζεται σε όλες τις περιπτώσεις ανάλυσης και εξόρυξης δεδομένων.

2.11.3 Συστήματα Ανάκτησης Εικόνας Βασιζόμενα στο Περιεχόμενό της. Δεικτοδότηση και Ανάκτηση Εικόνας

Η Ανάκτηση εικόνας που βασίζεται στο περιεχόμενο της (Content Based Image Retrieval), CBIR, στα πλαίσια της πολυτροπικής (multimodal) ανάκτησης πληροφορίας, αποτελεί ένα σημαντικό ερευνητικό τομέα από τις αρχές της δεκαετίας του 90

Η επίπονη και ακριβή εργασία καταλογογογράφησης εικόνων με το χέρι και η δυσκολία αντιστοίχισης λέξεων-κλειδιών στις εικόνες, που να καλύπτουν όλες τις πιθανές ανάγκες κάθε χρήστη, οδήγησε στην ανάπτυξη CBIR αλγορίθμων. Στόχοι της ερευνητικής δραστηριότητας είναι: 1) η εξαγωγή χαρακτηριστικών και η δεικτοδότηση της εικόνας που εξασφαλίζει την αποτελεσματική ανάκτηση των εικόνων που έχουν κάποια ομοιότητα με την εικόνα-ερώτημα (query image), με κάποιο τμήμα της ή με κάποια αναπαράστασή της (π.χ. το χρωματικό της ιστόγραμμα) και 2) η σχεδίαση μηχανής αναζήτησης εικόνων με βάση το περιεχόμενό τους αλλά και το συνοδευτικό τους κείμενο.

Ένα Σύστημα ανάκτησης πληροφοριών και εικόνων πρέπει για να απαντά σε ερωτήματα του τύπου:

1) Να βρεθούν εικόνες με περιεχόμενο σαν το περιεχόμενο της εικόνας X (όρος αναζήτησης) σε έγγραφα (documents) που πραγματεύονται το θέμα Y (όρος αναζήτησης)
2) Να βρεθούν έγγραφα που πραγματεύονται το θέμα Y και περιλαμβάνουν εικόνες σαν τη X
3) Να βρεθούν εικόνες με περιεχόμενο σαν της εικόνας X με λεζάντες που αναφέρονται σε θέμα Y.

Η μεθοδολογία σχεδίασης και υλοποίησης ενός CBIR θα μπορούσε να περιγράφεται από τα παρακάτω βήματα:

1) Επιλογή – δημιουργία βάσεων ψηφιακών εικόνων για πειραματισμό (Test sets).
2) Εξαγωγή χαρακτηριστικών εικόνας π.χ., ιστογράμματα, ροπές, συντελεστές Fourier, χρήση αλγορίθμου αυτο-οργανούμενων χαρτών SOM (Kohonen, 1995) (Vassilas, 2000) κ.λπ.
3) Δεικτοδότηση εικόνων. Η επιλογή των καταλληλότερων χαρακτηριστικών (features) για τη δεικτοδότηση (indexing) των εικόνων μπορεί να γίνει για παράδειγμα με χρήση τεχνικών νευρωνικών δικτύων (Perantonis, 1999) ή τεχνικής μεγιστοποίησης εντροπίας (Petridis, 2003).

Στην περίπτωση που οι εικόνες της βάσης είναι έγχρωμες (π.χ. RGB), γίνεται μετατροπή σε ασπρόμαυρες εικόνες με 256 αποχρώσεις γκρίζου.

Όταν τροφοδοτούμε με μια βάση εικόνων ένα σύστημα CBIR, το σύστημα αναλύει (και αποθηκεύει) τις εικόνες σύμφωνα με ένα σύνολο τεχνικών ανάλυσης εικόνας τις οποίες διαθέτει. Στη συνέχεια, εξάγονται κάποιοι δείκτες από την κάθε εικόνα και αποτελούν το διάνυσμα χαρακτηριστικών της (feature vector). Οι πληροφορίες αυτές είναι αρκετά μικρότερες σε μέγεθος σε σχέση με την αρχική εικόνα και οι δείκτες είναι το συγκρίσιμο μέγεθος ομοιότητας ανάμεσα σε δύο εικόνες.

Γνωστές τεχνικές που χρησιμοποιούνται ως μέθοδοι ανάλυσης της εικόνας είναι οι εξής:

1) Ανάλυση εικόνας με την χρήση τεχνικών Wavelets
2) Εξαγωγή στοιχείων τιμών του γκρίζου και δημιουργία ιστογράμματος
3) Εξαγωγή στοιχείων τιμών του γκρίζου και δημιουργία ιστογράμματος για κάθε μία από τις τρεις χρωματικές μπάντες του RGB χρωματικού χώρου
4) Εξαγωγή Color Moments από την αρχική εικόνα στον HSV χρωματικό χώρο

2.11.3.1 Παράδειγμα ανάκτησης εικόνας

Τα συστήματα CBIR υπολογίζουν οπτικές ομοιότητες μεταξύ της εικόνας - ερώτημα και των εικόνων που αποθηκεύονται σε μία βάση εικόνων. Τα αποτελέσματα της ανάκτησης είναι μία λίστα εικόνων οι οποίες χαρακτηρίζονται από τις ομοιότητες τους με την εικόνα - ερώτημα. Πολλές μετρικές ομοιότητας έχουν προταθεί για τα συστήματα CBIR και οι διαφορετικές μετρικές επηρεάζουν σημαντικά την απόδοση ενός συστήματος.

Στη συνέχεια φαίνεται παράδειγμα των αποτελεσμάτων ενός πειράματος αναζήτησης με χρήση περιστραφείσης εικόνας και αλλοιωμένης («κομμένης») εικόνας (Σφήκας, 2000). Στην εικόνα 2.29 βλέπουμε το αποτέλεσμα της αναζήτησης όταν το ερώτημα (original image) είναι «κομμένη» εικόνα. Στην εικόνα 2.30 βλέπουμε το αποτέλεσμα της αναζήτησης όταν το ερώτημα (original image) είναι περιστραφείσα εικόνα.

Εικόνα 2.29 Αποτέλεσμα αναζήτησης σε ερώτημα (original image) που είναι «κομμένη» εικόνα

Εικόνα 2.30 Αποτέλεσμα αναζήτησης σε ερώτημα (original image) που είναι περιστραφείσα εικόνα

2.11.3.2 Εξαγωγή Χαρακτηριστικών εικόνας

Αναγκαία διαδικασία για την δεικτοδότηση των εικόνων, σε εφαρμογές ανάκτησης εικόνων με βάση το περιεχόμενο, είναι η περιγραφή των εικόνων με ένα πλήθος χαρακτηριστικών. Εξαγωγή χαρακτηριστικών

μπορεί να γίνει τόσο στο χωρικό πεδίο (ιστογράμματα και αναλλοίωτες ροπές) όσο και στο πεδίο των συχνοτήτων (μέτρα συντελεστών Fourier που αντιστοιχούν σε χαμηλές συχνότητες).

Στην περίπτωση που οι εικόνες της βάσης είναι έγχρωμες (π.χ. RGB), μπορεί να γίνει μετατροπή σε ασπρόμαυρες εικόνες με 256 αποχρώσεις γκρίζου μέσω του τύπου:

$$X = \frac{R + G + B}{3}$$

όπου η απόχρωση γκρίζου X ενός εικονοστοιχείου της τελικής ασπρόμαυρης εικόνας υπολογίζεται ως η μέση τιμή της φωτεινότητας των τριών καναλιών της έγχρωμης εικόνας.

Εξαγωγή χαρακτηριστικών στο πεδίο του χώρου

Τα χαρακτηριστικά που χρησιμοποιούνται συνήθως για την περιγραφή εικόνων στο χωρικό πεδίο είναι: τα ιστογράμματα, οι ροπές (απλές, κεντρικές, αναλλοίωτες), το μέσο χρώμα, καθώς και πλήθος γεωμετρικών, τοπολογικών και μορφολογικών χαρακτηριστικών (π.χ. αριθμός Euler, εκκεντρότητα, συμπάγεια, κλίση πρωτεύοντα άξονα, κ.λπ.) ανάλογα με το είδος των εικόνων και αφού προηγηθεί η κατάτμηση της εικόνας στα αντικείμενα ή τις περιοχές από τις οποίες αποτελείται. Λόγω της δυσκολίας που εμφανίζει η κατάτμηση των εικόνων, συνήθως επιλέγεται εξαγωγή χαρακτηριστικών που περιγράφουν το σύνολο της εικόνας αντί των επί μέρους αντικειμένων της. Τέτοια χαρακτηριστικά είναι τα ιστογράμματα και οι ροπές.

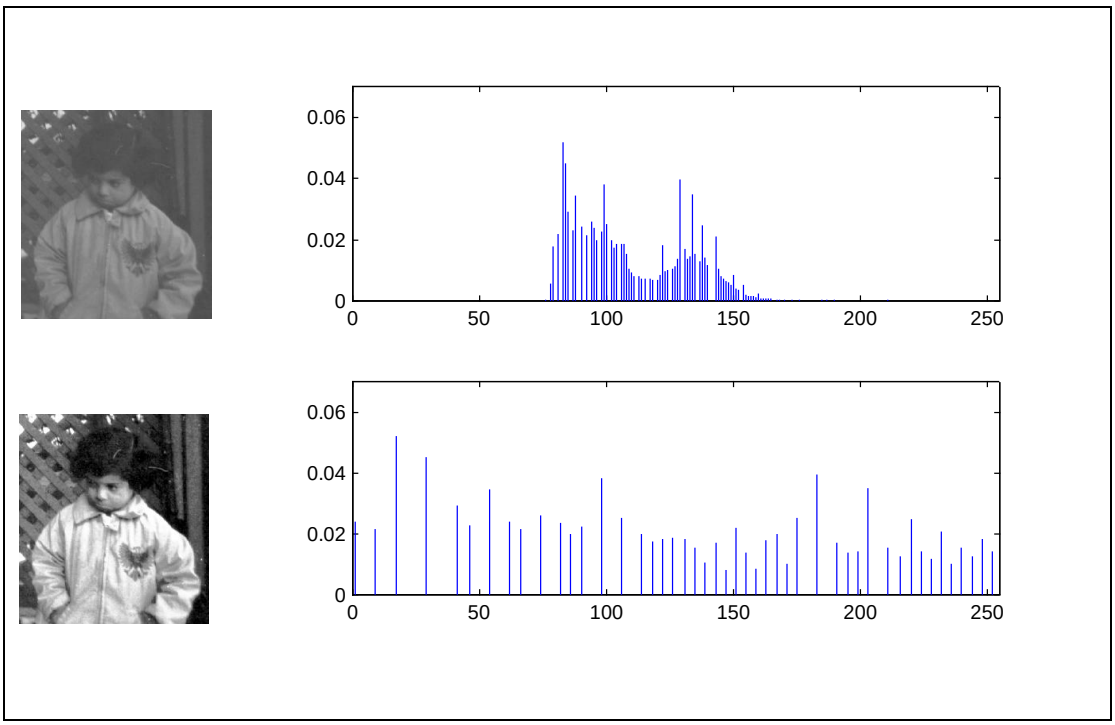
Ιστόγραμμα ισοσταθμισμένης εικόνας

Το ιστόγραμμα μιας εικόνας με τιμές γκρίζου k στο σύνολο {0, . . ., 255} δίνεται από τη διακριτή συνάρτηση

$$h(k) = n_k / n \qquad \text{για} \quad k = 0, 1, \ldots, 255$$

όπου n_k είναι το πλήθος των εικονοστοιχείων της εικόνας με τιμή γκρίζου ίση με k και n είναι το συνολικό πλήθος των εικονοστοιχείων της εικόνας. Με άλλα λόγια, το ιστόγραμμα της εικόνας απεικονίζει τις συχνότητες εμφάνισης των τιμών γκρίζου στα εικονοστοιχεία της εικόνας.

Προκειμένου να επιτευχθεί ένας βαθμός αναλλοιωσιμότητας στις συνθήκες φωτισμού της εικόνας και της χρωματικής της αντίθεσης, επιλέξαμε να γίνεται ισοστάθμιση του ιστογράμματος ώστε να αξιοποιείται όλη η παρεχόμενη δυναμική περιοχή των 256 τιμών γκρίζου με όσο το δυνατόν πιο ομοιόμορφη κατανομή. Το τελικό ιστόγραμμα που αποθηκεύεται στο διάνυσμα χαρακτηριστικών θα είναι αυτό της ισοσταθμισμένης εικόνας. Ένα παράδειγμα ισοστάθμισης ιστογράμματος παρουσιάζεται στην εικόνα 2.31.



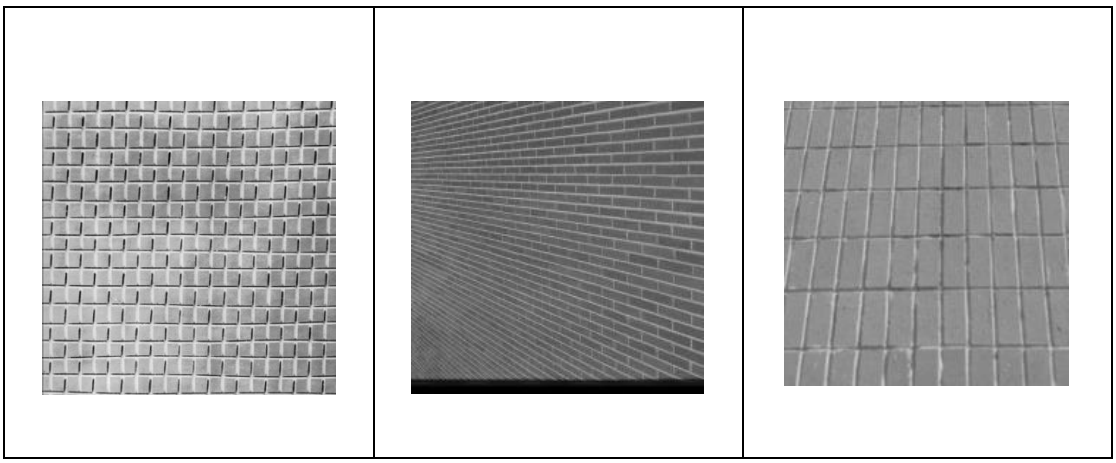
Εικόνα 2.31 Παράδειγμα ισοστάθμισης ιστογράμματος

2.11.3.3 Παραδείγματα εικόνων

Στη συνέχεια θα δώσουμε παραδείγματα εικόνων υφής και βιοϊατρικών εικόνων.

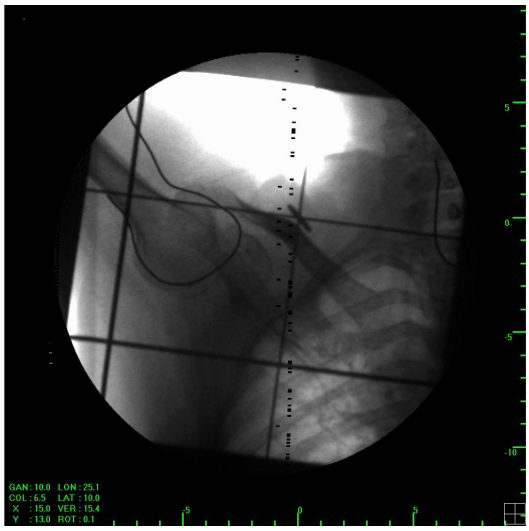
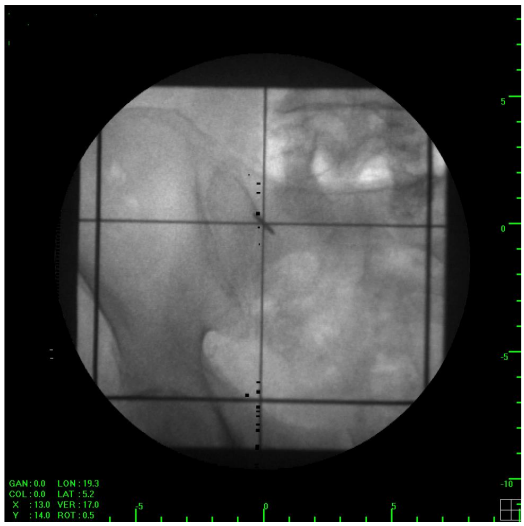
Βάση εικόνων υφής. Υπάρχουν βάσεις ψηφιακών εικόνων υφής (the brodatz textures) στον παγκόσμιο ιστό που επιτρέπουν την εξαγωγή χαρακτηριστικών υφής. Τα χαρακτηριστικά αυτά ενδεχομένως είναι σημαντικά για την ανάκτηση εικόνων με περιοχές που χαρακτηρίζονται από περιοδικότητες και επαναλαμβανόμενα πρότυπα. Στην εικόνα 2.32 παραθέτουμε παραδείγματα εικόνων υφής. Στο διαδίκτυο υπάρχουν βάσεις υφής. Δείτε για παράδειγμα τη συλλογή εικόνων, Brick walls - SIPI Image Database – Textures

http://sipi.usc.edu/database/database.php?volume=textures&image=40#top



Εικόνα 2.32 Εικόνες υφής

Βάσεις βιοϊατρικών εικόνων. Στη γενική περίπτωση, η ιδιαιτερότητα των βιοϊατρικών δεδομένων είναι ότι κατά την μεταφορά τους μέσα από τηλεπικοινωνιακά κανάλια ή την αποθήκευσή τους σε διάφορα μέσα (π.χ. σκληρός δίσκος, CD-ROM, USB κ.λπ.) δεν πρέπει να χάνεται πληροφορία. Με άλλα λόγια, επιτρέπεται συμπίεση των εικόνων με την προϋπόθεση ότι δεν έχουμε απώλεια πληροφορίας και με την αποσυμπιεσμένη εικόνα να είναι ακριβώς ίδια με την αρχική εικόνα. Μια τέτοια βάση έχει ιδιαίτερη σημασία στον έλεγχο κατά πόσον τεχνικές συμπίεσης όπως, παραδείγματος χάριν, με χρήση αυτο-οργανούμενων χαρτών, θα επηρεάζει τα εξαγόμενα χαρακτηριστικά και κατά συνέπεια και την ορθότητα της ανάκτησης βιοϊατρικών εικόνων με βάση το περιεχόμενο. Στην εικόνα 2.33 παραθέτουμε παραδείγματα ιατρικής εικόνας που έχουν συλλεγεί από ιατρικά μηχανήματα (Simulator, Treatment Planning) και ανήκουν σε καρκινοπαθείς.

Breast	Bone metastasis
	

Εικόνα 2.33 Παραδείγματα ιατρικής εικόνας από ιατρικά μηχανήματα (Simulator, Treatment Planning)

Επειδή οποιαδήποτε παρουσίαση και χρήση των εικόνων αυτού του τύπου υπόκειται σε περιορισμούς θα πρέπει να λείπουν/αλλοιωθούν οποιαδήποτε στοιχεία μπορούν να κάνουν γνωστή την ταυτότητα του ασθενή. Έτσι για λόγους προστασίας προσωπικών δεδομένων (για την εξασφάλιση της ανωνυμίας) από τις εικόνες πρέπει να απαλείφεται το ονοματεπώνυμο του ασθενούς, το όνομα του Νοσοκομείου και η ημερομηνία και ώρα λήψης της εικόνας από το ιατρικό μηχάνημα.

Παραπομπές και ενδεικτική βιβλιογραφία

Kohonen, T. (1995) *Self-Organizing Maps*, Springer-Verlag.

Perantonis, S. and V. Virvilis (1999) Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, Neural Processing Letters, 10(3), 243-252.

Petridis, S., & Perantonis, S. J. (2004). On the relation between discriminant analysis and mutual information for supervised linear feature extraction. Pattern Recognition, 37(5), 857-874.

Rui, Y., Huang, T. S., & Chang, S. F. (1999). Image retrieval: Current techniques, promising directions, and open issues. Journal of visual communication and image representation, 10(1), 39-62.

Tamura, H., & Yokoya, N. (1984). Image database systems: A survey. Pattern recognition, 17(1), 29-43.

Vassilas, N. (2018). Efficient neural network-based methodology for the design of multiple classifiers. In Recent Advances in Artificial Neural Networks (pp. 95-125). CRC Press.

Wang, W. (2001, September). Identifying salient risk factors by clamping neural networks. In IASTED International Conference on Artificial Intelligence and Applications.

Σκουρλάς, Χ. (2006) Εννοιολογικό πλαίσιο για Συστήματα Διαγλωσσικής, Βασιζόμενης στο Περιεχόμενο, Ανάκτησης Πληροφοριών και Διαχείρισης Γνώσης, σελ. 13-68, Συλλογικός τόμος «Παράλληλη, Βασιζόμενη σε Περιεχόμενο, Διαγλωσσική Ανάκτηση Πληροφοριών»

Βασιλάς, Ν., Περαντώνης, Σ., Σκουρλάς, Χ., Σφήκας, Κ. (2006) Εξαγωγή Χαρακτηριστικών, Δεικτοδότηση και Ανάκτηση Εικόνας Βασισμένη στο Περιεχόμενο, σελ. 95-116, Συλλογικός τόμος «Παράλληλη, Βασιζόμενη σε Περιεχόμενο, Διαγλωσσική Ανάκτηση Πληροφοριών»

2.12 Κατανεμημένες βάσεις δεδομένων. Βάσεις στο νέφος

Στην ενότητα θα προσεγγίσουμε τις κατανεμημένες βάσεις και τις βάσεις στο νέφος. Θα αναφέρουμε αρχικά αρχιτεκτονικές σύμφωνα με γνωστά ΣΔΒΔ και στη συνέχεια θα εξετάσουμε σύντομα τη σχέση των αρχιτεκτονικών με τη διαχείριση μεγάλων δεδομένων.

Ένα κατανεμημένο σύστημα βάσεων δεδομένων είναι μια συλλογή από λογικά σχετιζόμενες βάσεις δεδομένων που «συνεργάζονται» με διαφανή τρόπο (“co-operate in a transparent manner”, Rolland). Με τον όρο transparent εννοούμε ότι ο χρήστης έχει πρόσβαση στα δεδομένα που αποθηκεύονται σε διάφορες βάσεις δεδομένων σαν να ήταν μία και μόνο βάση δεδομένων.

Σύμφωνα με την Oracle,

«Ένα κατανεμημένο σύστημα βάσεων δεδομένων επιτρέπει στις εφαρμογές να έχουν πρόσβαση σε δεδομένα από τοπικές και απομακρυσμένες βάσεις δεδομένων. Σε ένα ομοιογενές σύστημα κατανεμημένης βάσης δεδομένων, κάθε βάση δεδομένων είναι μια βάση δεδομένων Oracle. Σε ένα ετερογενές κατανεμημένο σύστημα βάσεων δεδομένων, τουλάχιστον μία από τις βάσεις δεδομένων δεν είναι μια βάση δεδομένων Oracle. Οι κατανεμημένες βάσεις δεδομένων χρησιμοποιούν αρχιτεκτονική πελάτη/εξυπηρετητή για την επεξεργασία αιτημάτων πληροφοριών».

Oracle® Database (2021) Database Administrator’s Guide, 21c, F31835-11

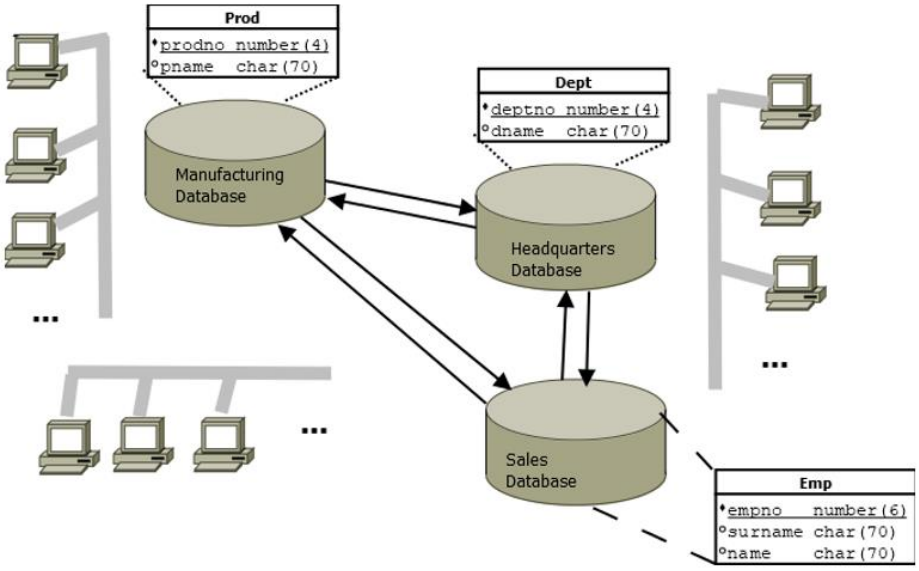
Database Administrator’s Guide (oracle.com)

Στη συνέχεια περιγράφουμε σύντομα κάποιες αρχιτεκτονικές κατανεμημένης βάσης δεδομένων (Distributed Database Architecture) με χρήση προϊόντων της Oracle. Ο αναγνώστης μπορεί να μελετήσει την προτεινόμενη βιβλιογραφία και επιπλέον να εξετάσει τη «φιλοσοφία» και τα προϊόντα των μεγάλων προμηθευτών (vendor) για να αποκτήσει μια σφαιρική εικόνα ενός ζητήματος που συνδέεται με όλες τις μεγάλες τεχνολογικές εξελίξεις των τελευταίων ετών.

2.12.1 Ομοιογενής κατανεμημένη βάση δεδομένων στο προϊόν της Oracle – Παράδειγμα

Η εικόνα 2.34 παρουσιάζει την αρχιτεκτονική μίας ομοιογενούς κατανεμημένης βάσης δεδομένων η οποία περιλαμβάνει τρεις βάσεις δεδομένων της υποθετικής εταιρείας ACME όλες υλοποιημένες με χρήση Oracle: Headquarters, Sales, Manufacturing.

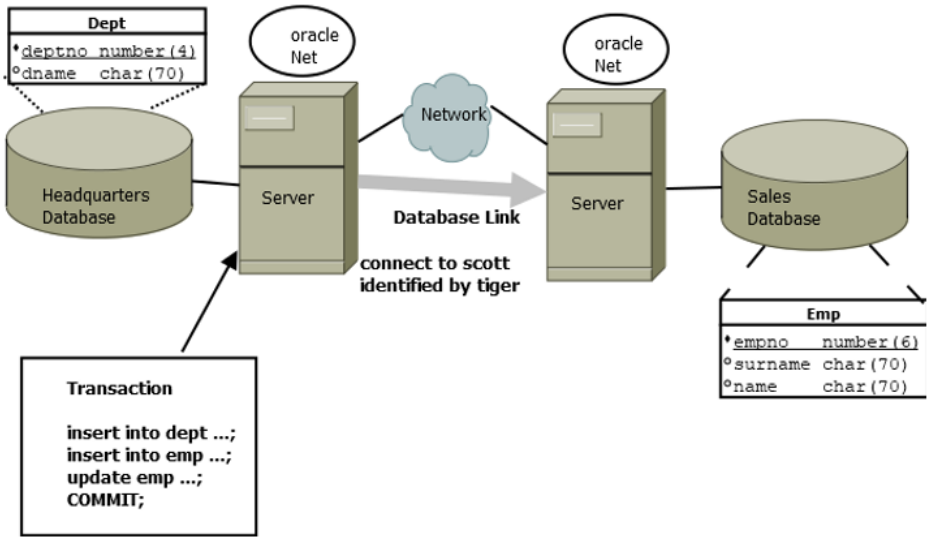
Κάθε βάση δεδομένων συνδέεται με διάφορα συστήματα πελατών (client systems) στα κεντρικά γραφεία και στα τμήματα πωλήσεων και κατασκευών. Η εικόνα βασίζεται σε Εικόνα της Oracle, βλέπε και κεφάλαιο 30.1 Distributed Database Architecture στο Oracle® Database (2021).



Εικόνα 2.34 Παράδειγμα ομοιογενούς κατανεμημένης βάσης δεδομένων (βασίζεται σε σχήμα της Oracle, βλέπε και Oracle® Database (2021).

2.12.2 Κατανεμημένο σύστημα βάσης δεδομένων στο προϊόν της Oracle

Η εικόνα 2.35 παρουσιάζει την αρχιτεκτονική ενός κατανεμημένου συστήματος βάσης δεδομένων στο προϊόν της Oracle. Υπάρχουν δύο βάσεις δεδομένων (Headquarters, Sales) που βρίσκονται σε χωριστούς διακομιστές (server). Η εικόνα παρουσιάζει παραδείγματα άμεσων και έμμεσων συνδέσεων πελατών. Τα ερωτήματα γίνονται απευθείας (directly) στη βάση δεδομένων Headquarters ενώ τα ερωτήματα γίνονται έμμεσα (indirectly) στη βάση δεδομένων Sales μέσω της βάσης δεδομένων Headquarters, η οποία στη συνέχεια ενεργεί ως πελάτης (servers). Η επικοινωνία μεταξύ των εξυπηρετητών γίνεται μέσω δικτύου και χρησιμοποιείται το προϊόν Oracle Net. Η εικόνα βασίζεται σε σχήμα της Oracle, βλέπε και κεφάλαιο 30.1 Distributed Database Architecture στο Oracle® Database (2021).



Εικόνα 2.35 Παράδειγμα αρχιτεκτονικής κατανεμημένου συστήματος βάσης δεδομένων (βασίζεται σε σχήμα της Oracle, βλέπε και Oracle® Database (2021).

2.12.3 Κατανεμημένη βάση δεδομένων, κατανεμημένη επεξεργασία (Distributed Computing) και ανάλυση μεγάλων δεδομένων

Η παρουσίαση του θέματος θα ξεκινήσει από την υπογράμμιση της διαφοράς του κατανεμημένου συστήματος βάσεων δεδομένων (distributed database) και του συστήματος κατανεμημένης επεξεργασίας (Distributed processing). Βέβαια, επισημαίνουμε ότι υπάρχει μία κοινή απαίτηση και στις δύο αντίστοιχες αρχιτεκτονικές. Τόσο η κατανεμημένη επεξεργασία όσο και οι κατανεμημένες βάσεις δεδομένων απαιτούν δίκτυο για τη σύνδεση όλων των στοιχείων τους και μάλιστα δίκτυο υψηλών ταχυτήτων, για να έχει νόημα ένα ποιοτικό κατανεμημένο περιβάλλον.

Μια κατανεμημένη βάση δεδομένων μπορεί να είναι ένα σύνολο βάσεων δεδομένων αποθηκευμένων σε πολλούς υπολογιστές σε διάφορες τοποθεσίες και να εμφανίζεται στις εφαρμογές και στον τελικό χρήστη ως μια ενιαία βάση δεδομένων. Επίσης, μια κατανεμημένη βάση δεδομένων μπορεί να είναι μία βάση δεδομένων η οποία αποθηκεύεται σε δύο ή περισσότερες φυσικά ανεξάρτητες τοποθεσίες οι οποίες συνδέονται μέσω δικτύου.

Στην κατανεμημένη επεξεργασία μία εφαρμογή κατανέμει τις εργασίες της μεταξύ διαφορετικών υπολογιστών σε ένα δίκτυο.

Παράδειγμα

Έχουμε μια (ενιαία) βάση δεδομένων η οποία βρίσκεται σε έναν μόνο υπολογιστή και το σύστημα κατανεμημένης επεξεργασίας μοιράζει τη λογική επεξεργασία των δεδομένων σε δύο τοποθεσίες που συνδέονται μέσω δικτύου. Η είσοδος/έξοδος των δεδομένων, η επιλογή και η επικύρωση δεδομένων ανατίθενται στον υπολογιστή Α και η οπτικοποίηση των δεδομένων και τα σχετικά εκτυπωτικά (report) στον υπολογιστή Β, αν και η κατανεμημένη επεξεργασία στο παράδειγμά μας να βασίζεται σε μια ενιαία βάση δεδομένων που βρίσκεται σε έναν μόνο υπολογιστή.

Η κατανεμημένη επεξεργασία δεν απαιτεί απαραίτητα μια κατανεμημένη βάση δεδομένων, αλλά μια κατανεμημένη βάση δεδομένων απαιτεί κατανεμημένη επεξεργασία και κάθε τμήμα της βάσης δεδομένων (database fragment) πρέπει να διαχειρίζεται από τη δική του διαδικασία τοπικής βάσης δεδομένων (local database process).

Η διαχείριση μεγάλων δεδομένων είναι άρρηκτα συνδεδεμένη με την κατανεμημένη επεξεργασία (distributed computing), την εικονικοποίηση (virtualization) και με την τεχνολογία REST.

Τεχνολογία REST

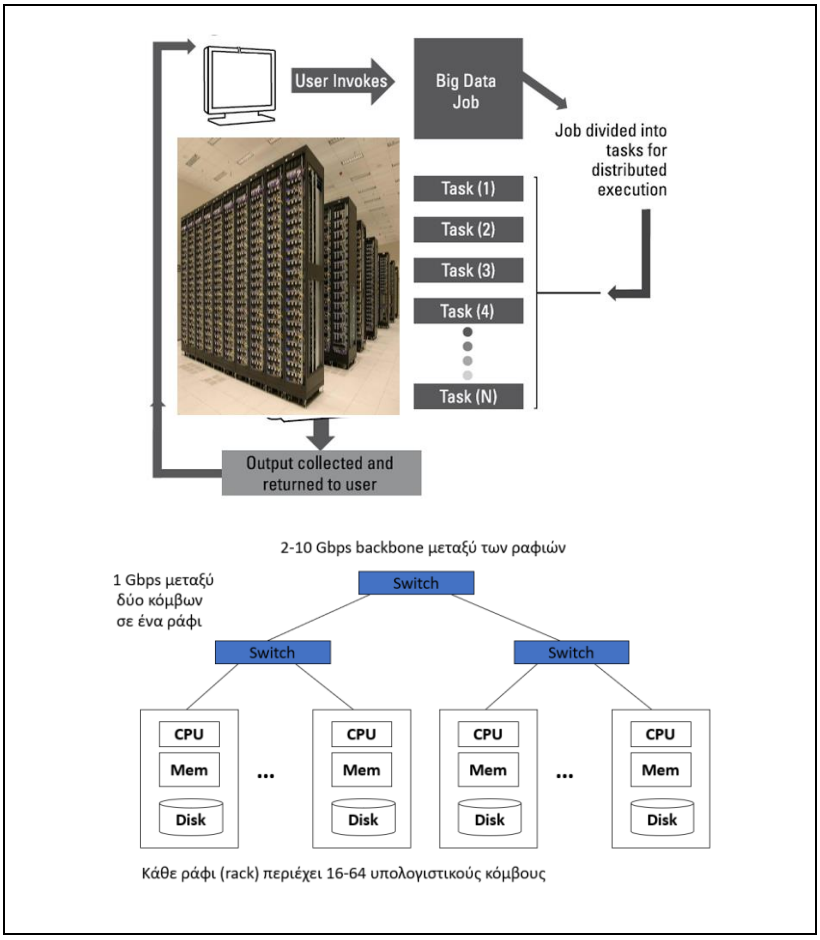
Οι τεχνολογικές λύσεις στις εφαρμογές μεγάλων δεδομένων υποστηρίζουν την τεχνολογία REST (Representational State Transfer) επειδή ένα RESTful API παρέχει έναν τυποποιημένο τρόπο δημιουργίας μιας προσωρινής σχέσης (ονομάζεται και χαλαρή σύζευξη, loose coupling) μεταξύ των πόρων του Παγκόσμιου Ιστού. Αυτό σημαίνει, βέβαια, ότι ένα αίτημά σας θα περιμένει και θα απαντηθεί μόνο όταν η υπηρεσία είναι διαθέσιμη. Βέβαια, τα αιτήματα REST σε μεγάλες εφαρμογές απευθύνονται σε Applications Servers οι οποίοι έχουν περιορισμό χρόνου στα εκκρεμή αιτήματα (requests).

Κατανεμημένη επεξεργασία

Στην κατανεμημένη επεξεργασία (ή κατανεμημένο υπολογισμό) πρέπει να κατανείμουμε τις συνιστώσες (components) μίας υπηρεσίας μεγάλων δεδομένων (big data service) σε μια σειρά κόμβων. Κάθε κόμβος είναι ένα στοιχείο απλής αρχιτεκτονικής που περιέχεται σε ένα σύμπλεγμα συστημάτων (cluster) ή μέσα σε «ραφιέρα» («ικρίωμα», rack) και μπορεί να περιλαμβάνει CPU, μνήμη και κάποιο είδος δίσκου (δείτε και

εικόνα 2.36). Στην εικόνα 2.36 βλέπουμε ότι κάθε «ράφι» περιέχει 16-64 κόμβους. Δύο κόμβοι σε ένα ράφι (rack) συνδέονται μέσω Ethernet με ταχύτητα 1 Gigabit/sec. Η ταχύτητα μεταξύ ραφιών είναι 2-10 Gbps.

Τελικά μπορούμε να έχουμε ένα κατανεμημένο σύστημα το οποίο αναλαμβάνει την κατανομή των εργασιών (jobs) και βασίζεται στην κατανεμημένη εκτέλεση στοιχειωδών εργασιών (tasks). Στην Εικόνα 2.36 μία συστάδα υπολογιστών αναλαμβάνει την επεξεργασία.



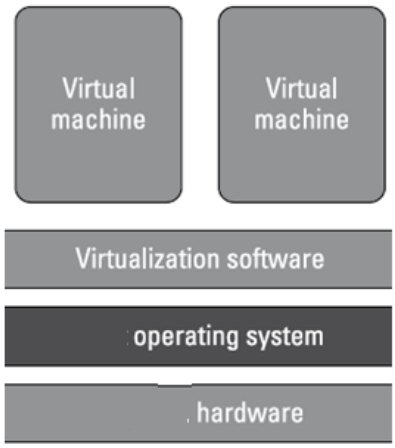
Εικόνα 2.36 Κόμβοι (CPU, μνήμη, δίσκος), ραφιέρες (rack), συστάδες (cluster). Κατανομή εργασιών (job) σε στοιχειώδεις εργασίες (task)

Μια προσέγγιση στην ανάλυση των μεγάλων δεδομένων θα μπορούσε να βασίζεται σε δύο αρχές:

1) Προσθήκη περισσοτέρων κόμβων σε ένα σύμπλεγμα (cluster) για κλιμάκωση (to scale out)
2) Κατανομή της ανάλυσης σε διαφορετικά περιβάλλοντα

Εικονικοποίηση (Virtualization)

Η εικονικοποίηση διαχωρίζει πόρους (resources) και υπηρεσίες (services) από το «υποκείμενο φυσικό περιβάλλον διανομής-εξυπηρέτησης-παράδοσης» (“the underlying physical delivery environment”), επιτρέποντας τη δημιουργία πολλών εικονικών συστημάτων σε ένα ενιαίο φυσικό σύστημα. Η μεγάλη διάδοση-εφαρμογή της εικονικοποίησης οφείλεται στη βελτίωση της απόδοσης και της αποτελεσματικότητας της επεξεργασίας. Αντί της αποκλειστικής εκχώρησης φυσικών πόρων σε κάθε σύνολο εργασιών (tasks) υπάρχει μία «δεξαμενή εικονικών πόρων» (“pool of virtual resources”) και οι φυσικοί πόροι κατανέμονται ταχύτατα ανάλογα με τους φόρτους εργασίας (workloads). Η «δεξαμενή εικονικών πόρων» επιτρέπει τη βελτίωση του «λανθάνοντος χρόνου εξυπηρέτησης» (latency), δηλαδή μειώνεται ο χρόνος εξυπηρέτησης του αιτήματος. Η εικόνα 2.37 απεικονίζει ένα τυπικό περιβάλλον εικονικοποίησης.



Εικόνα 2.37 Χρήση λογισμικού εικονικοποίησης για τη δημιουργία πολλών εικονικών συστημάτων σε ένα ενιαίο φυσικό σύστημα

Ο σχεδιασμός και οι διάφορες υλοποιήσεις του μοντέλου MapReduce αποτελούν παράδειγμα του τρόπου με τον οποίο ο κατανεμημένος υπολογισμός καθιστά την τεχνολογία μεγάλων δεδομένων λειτουργική και οικονομικά προσιτή.

2.12.4 Τύποι προϊόντων διαχείρισης βάσεων δεδομένων και μεγάλα δεδομένα

Η γλώσσα SQL είναι η πιο διαδεδομένη γλώσσα ερωτημάτων βάσης δεδομένων και χρησιμοποιείται και για τη διαχείριση των δομημένων σχεσιακών δεδομένων τα οποία περιλαμβάνονται στα μεγάλα δεδομένα. Υπάρχουν, όμως, και άλλες γλώσσες οι οποίες παρέχουν έναν αποδοτικό τρόπο επίλυσης των προβλημάτων διαχείρισης των μη δομημένων δεδομένων τα οποία, επίσης, περιλαμβάνονται στα μεγάλα δεδομένα. Είναι σημαντικό να ξεκαθαρίσουμε (αποφασίσουμε) ποιους τύπους δεδομένων μπορεί να διαχειριστεί η βάση δεδομένων και επιπλέον για ποια δεδομένα μπορούμε να έχουμε διαχείριση συναλλαγών. Οι σχεδιαστές συναλλαγών βάσεων δεδομένων περιγράφουν τη δυνατότητα διαχείρισης συναλλαγών και με το ακρωνύμιο ACID.

Πολλά προϊόντα NoSQL δεν υποστηρίζουν διαχείριση συναλλαγών και δεν υποστηρίζουν τις ιδιότητες ACID. Στον πίνακα 2.7 βλέπουμε κύρια χαρακτηριστικά των βάσεων δεδομένων SQL και βάσεων NoSQL και την υποστήριξή τους σε διαχείριση συναλλαγών. Πριν συνεχίσουμε την παρουσίαση, θυμίζουμε τις ιδιότητες ACID:

Ατομικότητα (Atomicity): Κάθε συναλλαγή είναι «ατομική», δηλαδή εκτελείται με τη λογική «όλα ή τίποτα». Αν οποιοδήποτε τμήμα της συναλλαγής αποτύχει ή συμβεί αστοχία συστήματος (λογισμικού ή υλικού) τότε η συναλλαγή αποτυγχάνει στο σύνολό της. Θυμηθείτε το παράδειγμα της μεταφοράς ποσού από λογαριασμό σε λογαριασμό και την απαιτούμενη «ατομικότητα» («όλα ή τίποτα») της συναλλαγής η οποία αφαιρεί το ποσό από τον ένα λογαριασμό και το πιστώνει στον άλλο.

Συνέπεια (Consistency): Μόνο οι συναλλαγές με έγκυρα δεδομένα θα εκτελούνται στη βάση δεδομένων. Εάν τα δεδομένα δεν είναι συνεπή τότε η συναλλαγή δεν πρέπει να ολοκληρωθεί και τα δεδομένα δεν θα εγγραφούν στη βάση δεδομένων. Θυμηθείτε ότι δεν μπορούμε να έχουμε αρνητικό υπόλοιπο σε τραπεζικό λογαριασμό και δεν επιτρέπεται ανάληψη μεγαλύτερη από το υπόλοιπο του λογαριασμού.

Απομόνωση (Isolation): Οι ταυτόχρονες συναλλαγές πρέπει να εκτελούνται σαν να είναι απομονωμένες, δηλαδή ανά δύο οι συναλλαγές πρέπει να εκτελούνται χωρίς να παρεμβαίνει η μία στην άλλη. Επομένως, κάθε έγκυρη συναλλαγή πρέπει να εκτελείται μέχρι να ολοκληρωθεί και οι διάφορες συναλλαγές πρέπει να

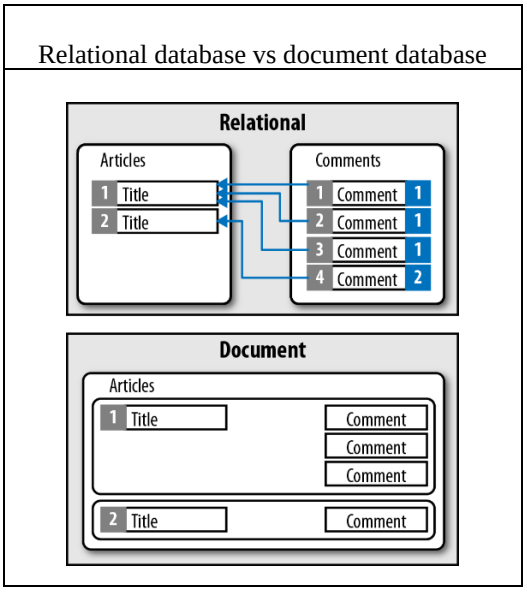
εκτελούνται με τη σειρά που υποβλήθηκαν για επεξεργασία. Υπογραμμίζουμε ότι η ιδιότητα περιγράφει μια ιδεατή (επιθυμητή) λειτουργία η οποία δεν εξασφαλίζεται πάντοτε στην πράξη. Δηλαδή, η συμμόρφωση με την ιδιότητα αυτή δεν είναι αυτονόητη έστω και αν στις εφαρμογές μας χρησιμοποιούμε σχεσιακά ΣΔΒΔ ακόμη και μεγάλων προμηθευτών (vendor). Στην ενότητα 2.8 αναφερθήκαμε σε προβλήματα (ανωμαλίες) τα οποία μπορούν να προκύψουν κατά την ταυτόχρονη εκτέλεση συναλλαγών και παραθέσαμε μια ανάλυση των επιπέδων απομόνωσης (isolation levels). Επισημαίνεται ότι η υποστήριξη των επιπέδων απομόνωσης, και ειδικότερα ο βαθμός υποστήριξης του επιπέδου SERIALIZABLE (για την εκτέλεση της συναλλαγής) ποικίλει στα διάφορα σχεσιακά ΣΔΒΔ. Επισημαίνουμε, επιπλέον, ότι το προϊόν MySQL συμμορφώνεται με το σχετικό πρότυπο.

Ανθεκτικότητα (Durability): Αν τα δεδομένα τα οποία διαχειρίζεται μία συναλλαγή εγγραφούν (commit work) στη βάση δεδομένων, και ο χρήστης έχει λάβει σχετικό μήνυμα, τότε δεν μπορούν να αναιρεθούν (rollback work), πρέπει να παραμείνουν στη βάση.

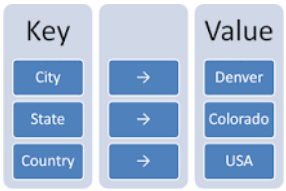
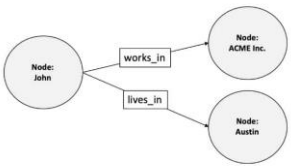
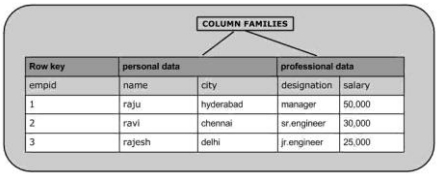
Στην εικόνα 2.38 βλέπουμε εικόνες-παραδείγματα με τις οποίες συγκρίνουμε τα χαρακτηριστικά των τύπων βάσεων δεδομένων Relational database, Document database. Στην εικόνα 2.39 βλέπουμε εικόνες-παραδείγματα με τις οποίες συγκρίνουμε τα χαρακτηριστικά των τύπων βάσεων δεδομένων Columnar Database, Graph database, Key-Value Database.

Πίνακας 2.7 Κύρια χαρακτηριστικά των βάσεων δεδομένων SQL και βάσεων NoSQL.

Type	Query language	MapReduce	Transactions	Examples
Relational database	SQL	No	ACID	Oracle, PostgreSQL
Columnar Database	Ruby	Hadoop	Yes, if enabled	HBase
Graph database	Walking, Search, Cypher	No	Acid	Neo4J
Document Data base	Commands	JavaScript	No	MongoDB, CouchDB
Key-Value Database	Lucene, Commands	JavaScript	No	Riak, Redis



Εικόνα 2.38 Σύγκριση των χαρακτηριστικών των τύπων βάσεων δεδομένων Relational database, Document database.

Key-value database	Graph database	Columnar database
kvd-pic1.png (544×342) (dv-website.s3.amazonaws.com)	kvd-pic1.png (544×342) (dv-website.s3.amazonaws.com)	columnar database example - Bing images
		

Εικόνα 2.39 Σύγκριση χαρακτηριστικών των τύπων βάσεων δεδομένων Columnar Database, Graph database, Key-Value Database

Σε ένα σχεσιακό ΣΔΒΔ (RDBMS) υπάρχει όριο στον μέγιστο όγκο δεδομένων που μπορεί να αποθηκεύσει και να επεξεργαστεί. Η ανάγκη επεξεργασίας και διαχείρισης πελώριου όγκου δεδομένων οδήγησε στη γέννηση του τομέα των Big Data. Μια κατανεμημένη βάση δεδομένων που αποθηκεύει τα δεδομένα με κατανεμημένο τρόπο σε πολλούς κόμβους μέσα σε ένα σύμπλεγμα (cluster) λύνει το πρόβλημα επειδή η λύση είναι οριζόντια κλιμακούμενη (horizontally scalable).

Μια βάση δεδομένων NoSQL είναι μια κατανεμημένη βάση δεδομένων που δεν υποστηρίζει την επεξεργασία συναλλαγών και επομένως και τις ιδιότητες ACID. Η βάση δεδομένων NoSQL χαρακτηρίζεται και ως «σύστημα κοινής χρήσης δεδομένων μέσω δικτύου» (network shared data system), δηλαδή συμμορφώνεται με το λεγόμενο θεώρημα CAP, όπου τα αρχικά σημαίνουν: C-Consistency, A-Availability, P-Partition Tolerance.

Σύμφωνα με το θεώρημα CAP (CAP theorem), το οποίο είναι γνωστό και ως θεώρημα Brewer, κάθε «σύστημα κοινής χρήσης δεδομένων μέσω δικτύου» (network shared data system) μπορεί να παρέχει μόνο δύο από τις ακόλουθες τρεις εγγυήσεις:

- Συνέπεια. Κάθε αίτημα ανάγνωσης (read) λαμβάνει ως απάντηση την πιο πρόσφατη εγγραφή (write) ή ένδειξη σφάλματος.
- Διαθεσιμότητα. Κάθε αίτημα λαμβάνει απάντηση χωρίς ένδειξη σφάλματος αλλά δεν υπάρχει εγγύηση ότι στην απάντηση έχουμε την πιο πρόσφατη εγγραφή (write).
- Ανοχή κατάτμησης (Partition tolerance).Το σύστημα συνεχίζει να λειτουργεί αν και υπάρχουν προβλήματα δικτύου μεταξύ κόμβων και παρά την απόρριψη ή την καθυστέρηση των μηνυμάτων μεταξύ των κόμβων.

Ενδεικτική βιβλιογραφία

Gilbert, S. and Lynch, N. (2002) Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services", ACM SIGACT News, Volume 33 Issue 2, pp. 51–59

Brewer, E. (2012) CAP twelve years later: How the ‘rules’ have changed, Computer, Volume 45, Issue 2, pp. 23–29.

Θα χρειαστεί στη συνέχεια να αναφερθούμε και στην έννοια της «κατάτμησης δικτύου» (network partitioning).

Η «κατάτμηση δικτύου» είναι μια αποτυχία του δικτύου η οποία προκαλεί το διαχωρισμό των μελών σε ομάδες, έτσι ώστε το μέλος μιας ομάδας να μην μπορεί να επικοινωνήσει με τα μέλη άλλων ομάδων και επομένως όλες οι πλευρές του αρχικού συμπλέγματος (cluster) λειτουργούν ανεξάρτητα.

Ουσιαστικά το θεώρημα CAP δηλώνει ότι ένα «σύστημα κοινής χρήσης δεδομένων μέσω δικτύου» μπορεί να εγγυηθεί συνέπεια και διαθεσιμότητα μόνο όταν το δίκτυο λειτουργεί χωρίς προβλήματα. Σε περίπτωση του φαινομένου «κατάτμησης δικτύου» (network partition) το σύστημα μπορεί να εγγυηθεί είτε μόνο τη συνέπεια είτε μόνο τη διαθεσιμότητα.

Ένα τέτοιο σύστημα μπορεί χαρακτηρίζεται ανάλογα με τις εγγυήσεις που παρέχει ως 02 τύπων:

CP-Consistent & Partition Tolerant. Σε περίπτωση βλάβης κατάτμησης του δικτύου (network partition failure) όλοι οι κόμβοι του συστήματος παρέχουν την πιο πρόσφατη επιτυχή εγγραφή ή δεν παρέχουν κάποια απάντηση. Το μοντέλο αυτό ονομάζεται μοντέλο ισχυρής συνέπειας (Strong Consistency).

AP-Available & Partition Tolerant. Σε βλάβη κατάτμησης δικτύου όλοι οι κόμβοι του συστήματος θα παρέχουν τα δεδομένα αλλά δεν θα εγγυώνται ότι τα δεδομένα αντιστοιχούν στην πιο πρόσφατη επιτυχή εγγραφή. Το μοντέλο αυτό ονομάζεται μοντέλο της ενδεχόμενης συνέπειας (Eventual Consistency model).

Ο Eric Brewer προτείνει αντί των ιδιοτήτων ACID ένα άλλο μοντέλο, τις ιδιότητες BASE, για τη διασφάλιση της λειτουργικότητας των βάσεων NoSQL:

BA - Basically Available,

S - Soft-state,

E- Eventual consistency.

Περιγράφουμε σύντομα με μία πρόταση τις απαιτήσεις (την άποψή του) για τη λειτουργία εφαρμογής-βάσης NoSQL:

Μια εφαρμογή σε γενικές γραμμές πρέπει να λειτουργεί συνεχώς (κατάσταση basically available) και δεν απαιτείται να είναι συνεχώς συνεπής (κατάσταση soft-state), αλλά τελικά θα πρέπει να περιέλθει σε κάποια γνωστή κατάσταση (κατάσταση eventual consistency) σε αντίθεση με τις ιδιότητες ACID που απαιτούν «αυστηρή συνέπεια» (strict consistency).

2.12.5 Βάσεις δεδομένων νέφους (Cloud database)

Στην ενότητα αυτή θα ξεκινήσουμε, επίσης, με την άποψη γνωστών προμηθευτών ΣΔΒΔ.

Σύμφωνα με την IBM, μια βάση δεδομένων νέφους (cloud database) είναι μια υπηρεσία βάσης δεδομένων (database service) που δημιουργήθηκε και είναι προσπελάσιμη μέσω μιας πλατφόρμας νέφους. Υποστηρίζει πολλές από τις λειτουργίες μιας παραδοσιακής βάσης δεδομένων με την πρόσθετη ευελιξία της υπολογιστικής νέφους (cloud computing). Οι χρήστες εγκαθιστούν λογισμικό σε υποδομή νέφους για τη διαχείριση και τη λειτουργία της βάσης δεδομένων. Παραθέτουμε κάποια βασικά χαρακτηριστικά της βάσης δεδομένων νέφους:

- Επιτρέπει στους εταιρικούς χρήστες να «φιλοξενούν» (host) βάσεις δεδομένων χωρίς να αγοράζουν ειδικό υλικό (hardware).
- Μπορεί να τη διαχειρίζεται ο χρήστης ή να προσφέρεται ως υπηρεσία και να τη διαχειρίζεται ο πάροχος
- Μπορεί να υποστηρίξει σχεσιακές βάσεις δεδομένων, π.χ., MySQL, PostgreSQL και βάσεις NoSQL, π.χ., MongoDB, Apache CouchDB.

- Η πρόσβαση γίνεται μέσω διεπαφής ιστού (web interface) ή API που παρέχεται από προμηθευτή (vendor-provided API).

Πηγή των στοιχείων είναι η ιστοσελίδα,

What is a cloud database? Learn about cloud database solutions from IBM

https://www.ibm.com/cloud/learn/what-is-cloud-database

Σύμφωνα με την Oracle υπάρχουν δύο κύρια μοντέλα ανάπτυξης βάσεων δεδομένων νέφους (Cloud Database):

1) Στο παραδοσιακό μοντέλο η ανάπτυξη προσομοιάζει με την επιτόπια (onsite), εσωτερική (inhouse) διαχείριση βάση δεδομένων εκτός βέβαια από τη χρησιμοποιούμενη υποδομή. Σχετικά με την υποδομή, ο οργανισμός αγοράζει χώρο εικονικής μηχανής (virtual machine space) από έναν πάροχο υπηρεσιών νέφους και η βάση δεδομένων αναπτύσσεται στο νέφος. Ο οργανισμός είναι υπεύθυνος για την εποπτεία και τη διαχείριση των βάσεων δεδομένων του και χρησιμοποιεί προσωπικό πληροφορικής (IT staff) για τον έλεγχο της βάσης δεδομένων.
2) Πολλοί οργανισμοί χρησιμοποιούν σήμερα το μοντέλο DevOps, δηλαδή οι δύο ομάδες, ομάδα ανάπτυξης και ομάδα λειτουργίας, συνεργάζονται σε ολόκληρο τον κύκλο ζωής της εφαρμογής λογισμικού που περιλαμβάνει ανάπτυξη, δοκιμή, λειτουργία. Συχνά στην πράξη δημιουργείται μία ομάδα μόνο η οποία αναλαμβάνει ανάπτυξη και λειτουργία.
3) Στο μοντέλο «Βάση δεδομένων ως υπηρεσία» (Database as a service, DBaaS), ο οργανισμός συνάπτει συμβόλαια με έναν πάροχο υπηρεσιών νέφους και χρησιμοποιεί συνδρομητική υπηρεσία. Ο πάροχος προσφέρει στον τελικό χρήστη μια ποικιλία λειτουργιών (tasks) συντήρησης και διαχείρισης βάσεων δεδομένων σε πραγματικό χρόνο. Η βάση δεδομένων λειτουργεί στην υποδομή του παρόχου. Αυτό το μοντέλο χρήσης συνήθως περιλαμβάνει αυτοματοποίηση λειτουργίας της βάσης, δημιουργίας αντιγράφων ασφαλείας (backup), κλιμάκωσης (scaling), υψηλής διαθεσιμότητας (high availability), ασφάλειας (security), επιδιόρθωσης (patching) και παρακολούθησης (monitoring). Το μοντέλο DBaaS παρέχει στους οργανισμούς τη δυνατότητα διαχείρισης βάσης δεδομένων με εξωτερική ανάθεση (outsourced database management) αντί να προσλαμβάνει και να διαχειρίζεται εσωτερικά εμπειρογνώμονες βάσεων δεδομένων (database experts).

Πηγή των στοιχείων είναι η ιστοσελίδα,

What is a Cloud Database?

https://www.oracle.com/database/what-is-a-cloud-database/

2.12.5.1 Είδη παροχής υπηρεσιών νέφους

Η υπολογιστική νέφους (cloud computing) είναι μια μέθοδος παροχής ενός συνόλου κοινόχρηστων υπολογιστικών πόρων που περιλαμβάνει:

- υπηρεσίες για τις εφαρμογές, π.χ. υπηρεσίες διαχείρισης σχέσεων πελατών (CRM), οι οποίες βασίζονται σε επιλογή Software as a Service (SaaS)
- παροχή υποδομής, π.χ., υπολογιστές με αποθηκευτικό χώρο, δικτύωση κ.λπ.
- πλατφόρμες για ανάπτυξη και διανομή εφαρμογών κ.λπ.

Τα πάντα, από την υπολογιστική ισχύ μέχρι την υπολογιστική υποδομή και από τις εφαρμογές και τις επιχειρηματικές διαδικασίες έως τα δεδομένα και τα αναλυτικά στοιχεία, μπορούν να παρέχονται ως υπηρεσία.

Υπάρχουν τέσσερα μοντέλα παρεχόμενων υπηρεσιών υπολογιστικού νέφους:

Η υποδομή ως υπηρεσία (Infrastructure as a Service)

Η υποδομή ως υπηρεσία (IaaS) παρέχει με βάση ένα μοντέλο ενοικίασης υπολογιστικών υπηρεσιών, π.χ., υλικό, δικτύωση, αποθήκευση. Ο πελάτης της υπηρεσίας μισθώνει έναν πόρο και χρεώνεται με βάση τη διάρκεια της χρήσης κ.λπ.

Πλατφόρμα ως υπηρεσία (Platform as a Service)

Η πλατφόρμα ως υπηρεσία (PaaS) είναι ένας μηχανισμός ο οποίος συνδυάζει την υπηρεσία IaaS με ένα σύνολο υπηρεσιών ενδιάμεσου λογισμικού (middleware services), δηλαδή υπηρεσιών ανάπτυξης λογισμικού και εργαλείων ανάπτυξης. Με την υπηρεσία αυτή η επιχείρηση εξασφαλίζει έναν αξιόπιστο και συνεπή τρόπο ανάπτυξης και εκμετάλλευσης εφαρμογών στο νέφος ή και σε άλλες εγκαταστάσεις ("on a cloud or on premises").

Λογισμικό ως υπηρεσία (Software as a Service)

Το λογισμικό ως υπηρεσία (SaaS) είναι μια επιχειρηματική εφαρμογή που δημιουργείται και φιλοξενείται σε έναν πάροχο. Ακολουθεί το μοντέλο πολλαπλών μισθώσεων. Η πολυμίσθωση αναφέρεται σε μεμονωμένη παρουσία μιας εφαρμογής ("a single instance of an application") η οποία εκτελείται σε περιβάλλον νέφους, και εξυπηρετεί πολλούς οργανισμούς πελατών («ενοικιαστές»), οι οποίοι διατηρούν τα δεδομένα τους ξεχωριστά.

Τα δεδομένα ως υπηρεσία (Data as a Service)

Τα δεδομένα ως υπηρεσία (DaaS) σχετίζονται στενά με την υπηρεσία SaaS. Είναι μια υπηρεσία ανεξάρτητη από την πλατφόρμα η οποία επιτρέπει σύνδεση στο νέφος για την αποθήκευση και ανάκτηση των δεδομένων. Επιπλέον, παρέχονται εξειδικευμένες υπηρεσίες γεγονός κρίσιμο σε ένα περιβάλλον μεγάλων δεδομένων. Για παράδειγμα, η Google παρέχει μια υπηρεσία η οποία για ένα ερώτημα μπορεί να επεξεργαστεί 5 terabyte δεδομένων σε 15 δευτερόλεπτα, δηλαδή σε χρόνο δέκα φορές λιγότερο από τον απαιτούμενο σε κέντρο διαχείρισης δεδομένων (data center).

Έχουν αναπτυχθεί από εταιρείες, π.χ., IBM, πάρα πολλές εξειδικευμένες υπηρεσίες ανάλυσης δεδομένων.

2.13 Ενεργές βάσεις δεδομένων (active databases) και τεχνολογία εναυσμάτων (triggers)

Η αρχή της παρουσίασης-συζήτησης στην ενότητα αυτή γίνεται με παράδειγμα εναύσματος (trigger) στο προϊόν της ORACLE. Θα αναφέρουμε κάποια απαραίτητα στοιχεία πριν δημιουργήσουμε το πρώτο έναυσμα.

Ο πίνακας DUAL είναι ένας πίνακας του συστήματος τον οποίο χρησιμοποιούμε όταν θέλουμε να περάσουμε τιμές που "κρατάει" το (ή για την ακρίβεια είναι ανά πάσα στιγμή καταχωρημένες στο) σύστημα σε μεταβλητές ή όταν θέλουμε απλά να τις δούμε ή ακόμη να κάνουμε κάποιες πράξεις. Για παράδειγμα, οι παρακάτω δηλώσεις:

- Η δήλωση select user from dual; δείχνει το χειριστή.
- Η δήλωση select sysdate from dual; δείχνει την ημερομηνία.
- Η δήλωση select ((15*4)+(4.10))/5 from dual; εκτελεί τις πράξεις και δείχνει το αποτέλεσμα.

Ακολουθεί η δημιουργία του πρώτου εναύσματος.

```
CREATE TRIGGER EMP_AUDIT
/* Το έναυσμα ενεργοποιείται κάθε φορά που ενημερώνεται ο μισθός του υπαλλήλου
στον πίνακα EMP */
AFTER UPDATE OF SAL
ON EMP
FOR EACH ROW
DECLARE
 CUR_USER VARCHAR2(8);
 CUR_DATE DATE;
BEGIN
/* Καταγράφονται στις δύο μεταβλητές CUR_USER, CUR_DATE το όνομα του χρήστη (που
έχει συνδεθεί στη βάση και χρησιμοποιεί κάποια προγράμματα που μεταβάλουν το
μισθό, οπότε καλείται και εκτελείται το έναυσμα) και η τρέχουσα  ημερομηνία */
SELECT USER, SYSDATE
INTO CUR_USER, CUR_DATE
FROM DUAL;
/* Στη συνέχεια γίνεται εισαγωγή στον πίνακα EMP_AUDIT του κωδικού του
υπαλλήλου, του μισθού του πριν από την αλλαγή και μετά, ημερομηνίας μεταβολής
και όνομα χειριστή που έκανε τη μεταβολή */
INSERT INTO EMP_AUDIT
VALUES(:OLD.EMPNO,CUR_DATE,:OLD.SAL,:NEW.SAL,CUR_USER);
/* Παρατηρήστε τα χαρακτηριστικά OLD, NEW με τη βοήθεια των οποίων
διαχειριζόμαστε παλαιά και νέα τιμή */
```

2.13.1 Ενεργές βάσεις δεδομένων (active database)

Ένα ενεργό σύστημα διαχείρισης βάσεων δεδομένων (active database management system) είναι ένα ΣΔΒΔ με τη πλήρη λειτουργικότητα διαχείρισης βάσης δεδομένων, το οποίο έχει δύο επιπρόσθετες δυνατότητες: 1) παρακολούθησης (monitoring) της κατάστασης της βάσης δεδομένων και 2) εκτέλεσης ορισμένων προκαθορισμένων ενεργειών (predefined actions) όταν εντοπίζονται κατάλληλα συμβάντα-γεγονότα (events). Δηλαδή, στο ενεργό σύστημα υπάρχουν οι κανόνες οι οποίοι συνδέονται με γεγονότα. Αν συμβεί κάποιο γεγονός τότε αυτόματα εκτελείται ο κανόνας.

Τα γνωστά εμπορικά ΣΔΒΔ υλοποιούν με ένα κάπως απλουστευμένο τρόπο τη διαχείριση των γεγονότων. Πιο συγκεκριμένα υποστηρίζουν τον ορισμό και τη χρήση εναυσμάτων (triggers), όπως είδαμε και στην αρχή της ενότητας. Η ιδέα που χρησιμοποιούν είναι απλή:

Το έναυσμα έχει μια συνθήκη αφύπνισης, η οποία συνδέεται με ένα γεγονός το οποίο μπορεί να συμβεί στη βάση δεδομένων, και έχει το «σώμα» του (body), δηλαδή ένα πρόγραμμα. Ακολουθεί ο σκελετός ενός παραδείγματος.

```
CREATE TRIGGER my_trigger
BEFORE INSERT INTO emp
….
BEGIN
UPDATE dept
SET no_of_employees= ….
….
END
```

Το έναυσμα αποθηκεύεται και είναι σε κατάσταση «ύπνωσης» στο λεξικό δεδομένων (data dictionary), δηλαδή στη βάση δεδομένων που χρησιμοποιεί το ΣΔΒΔ για να κάνει τη διαχείριση. Έχει το όνομα my_trigger

και η συνθήκη αφύπνισής του είναι BEFORE INSERT INTO emp. Δηλαδή, όταν γίνει εισαγωγή στοιχείων στον πίνακα emp τότε αυτόματα το έναυσμα ενεργοποιείται από το ΣΔΒΔ και εκτελείται το σώμα του, δηλαδή οι εντολές-δηλώσεις οι οποίες περικλείονται από BEGIN …. END. Δείτε ότι στο «σώμα» περιλαμβάνεται μία δήλωση UPDATE η οποία θα αλλάξει κάποια στοιχεία στον πίνακα dept.

Τα εναύσματα χρησιμοποιούνται στην αντιμετώπιση συμβάντων στη βάση δεδομένων, τη διασφάλιση της ακεραιότητας των δεδομένων, στην υλοποίηση επιχειρηματικών κανόνων, κ.λπ.

Επειδή ένα ενεργό σύστημα έχει και την «παραδοσιακή» πλευρά του μπορούμε να χρησιμοποιήσουμε για τη μοντελοποίησή του το μοντέλο οντοτήτων συσχετίσεων, ΜΟΣ (Entity-Relationship model-ER), ενσωματώνοντας την ενεργή συμπεριφορά βάσης δεδομένων στο μοντέλο με τη μορφή συμβάντων και κανόνων. Στη βιβλιογραφία υπάρχουν διάφορες προτάσεις για το πρόβλημα του εννοιολογικού σχεδιασμού ενεργών συστημάτων εδώ και πολλά χρόνια. Βλέπε σχετικά:

Asteriokiyoshi Tanaka (1992) On conceptual design of active databases, PhD thesis, Georgia Institute of Technology, School of Information & Computer Science

On conceptual design of active databases | Guide books (acm.org)

MM Zoet (2014) Methods and Concepts for Business Rules Management, PhD thesis, University of Applied Sciences Utrecht

Παράδειγμα σχεδίου υλοποίησης επιχειρηματικού κανόνα με έναυσμα

Ακολουθεί παράδειγμα επιχειρηματικού κανόνα κράτησης δύο διπλών δωματίων σε ένα σύστημα κρατήσεων δωματίων ξενοδοχείου.

```
IFnum_of_adults >=2 AND
num_of_adults <=4 AND
num_of_kids = 0
THEN rooms=2;
```

Στη συνέχεια παρατίθεται ένα μέρος του αντίστοιχου trigger.

```
create or replace trigger reserve_2_rooms
before insert on hotel_reservation
for each row
begin
if :new.NUM_OF_ADULTS >2
  and :new.NUM_OF_ADULTS <= 4
  and :new.NUM_OF_KIDS = 0
then :new.ROOMS := 2;
insert into hotel_reserve_rule_link values κ.λπ.;
end if;
end;
```

Μια λεπτομερής περιγραφή του συστήματος κρατήσεων, της μοντελοποίησής του, των επιχειρηματικών κανόνων του κ.λπ. υπάρχει στο επόμενο άρθρο:

Rajeev Kaula (2012) Business Rules Modeling for Business Process Events: An Oracle Prototype, Journal of computers, vol.7, no.9

Κανόνες Συμβάν-Κατάσταση-Δράση (ECA-Event-Condition-Action rules)

Αντιγράφουμε από το άρθρο (manifesto) του ACT-NET Consortium:

«Η προσθήκη της λειτουργικότητας απόκρισης-απάντησης (reactive functionality) (σημείωση, σε ενεργό σύστημα βάσης δεδομένων) χαρακτηρίζεται από κανόνες ECA-Event-Condition-Action rules (κανόνες Συμβάν-Κατάσταση-Δράση) οι οποίοι ορίζονται:

- από ένα συμβάν,
- έναν έλεγχο των συνθηκών και (εάν ισχύει ο έλεγχος τότε εκτελείται)
- μια ενέργεια»

("The addition of the reactive functionality is characterized by an ECA-rules (event-condition-action rules) defined by an event, a check of conditions, and if true, an action is executed. Once a group of rules is defined, an Active Database Management System will monitor the event").

Βλέπε σχετικά και το άρθρο:

ACT-NET Consortium. 1996. The Active Database Management System Manifesto: A Rulebase of ADBMS Features. ACM SIGMOD Record, vol.25, no.3, p.40-49.

The active database management system manifesto: a rulebase of ADBMS features: ACM SIGMOD Record: Vol 25, No 3

Γενικά ένας κανόνας Συμβάν-Κατάσταση-Δράση θα μπορούσε να έχει την παρακάτω δομή:

ON event

IF condition

THEN action 1

ELSE action 2

Παραδείγματα κανόνων ECA σε ένα σύστημα διαχείρισης μισθοδοσίας:

Τα παραδείγματα που ακολουθούν υπάρχουν στο άρθρο:

Li, X., Medina Marń, J., & Chapa, S. V. (2002, April). A structural model of ECA rules in active database. In Mexican International Conference on Artificial Intelligence (pp. 486-493). Springer, Berlin, Heidelberg.

Rule 1

```
on insert employee
if e.degree = "Phd"
then modify e.status = "A" and e.salary = e.salary * 1.50
```

Rule 2

```
on insert employee
if e.degree = "Master"
then modify e.status = "B" and e.salary = e.salary * 1.25
```

Rule 3

```
on insert employee
if e.degree = "Bachelor"
then modify e.status ="C" and e.salary = e.salary * 1.10
```

Rule 4

```
on insert employee
if e.salary > manager.salary
then modify e.salary = manager.salary-100
```

2.13.2 Εναύσματα σε εμπορικά ΣΔΒΔ

Όπως είδαμε στα παραδείγματα, τα γνωστά ΣΔΒΔ, Oracle, SQL Server, DB2, MySQL, κ.λπ., υποστηρίζουν κάποια χαρακτηριστικά των ενεργών βάσεων δεδομένων με τη μορφή εναυσμάτων. Ο όρος trigger έχει αποδοθεί στα ελληνικά και ως σκανδάλη/ες. Οι triggers περιλαμβάνονται στο πρότυπο SQL-99 (Chapter 24, SQL trigger) και στα μεταγενέστερα πρότυπα. Σύμφωνα με το πρότυπο ένας trigger ορίζεται από οκτώ τμήματα:

1) Το <Trigger name>, με πιστοποίηση (qualification) από το <Schema name> δηλαδή του σχήματος στο οποίο ανήκει.
2) Το όνομα του πίνακα του οποίου τα δεδομένα, όταν αλλάξουν, θα προκαλέσουν την ενεργοποίηση του Trigger.
3) Ο χρόνος δράσης του Trigger, ο οποίος ενημερώνει το ΣΔΒΔ (DBMS) πότε να εκτελέσει το σώμα του Trigger (BEFORE/AFTER το συμβάν του Trigger).
4) Το συμβάν που ενεργοποιεί τον Trigger. Το συμβάν ενημερώνει το ΣΔΒΔ για τη δήλωση που προκαλεί κάποια αλλαγή των δεδομένων (οι δηλώσεις που ενεργοποιούν μπορεί να είναι INSERT, UPDATE, DELETE) στον πίνακα του Trigger.
5) Η λίστα των στηλών (Trigger Column list) για το συμβάν του Trigger. Περιγράφεται, ειδικότερα στην περίπτωση UPDATE Trigger events, και εάν η λίστα καθορίστηκε ρητά ή σιωπηρά (explicitly or implicitly).
6) Παλιές τιμές <Correlation name>, νέες τιμές <Correlation name>, παλιές τιμές Ψευδώνυμου πίνακα (Table alias) ή νέες τιμές Ψευδώνυμου πίνακα (Table alias) που ορίζονται για τον Trigger.
7) Trigger body: οι δηλώσεις SQL που θέλετε να εκτελέσει το DBMS στον πίνακα του Trigger όταν ο Trigger ενεργοποιείται.
8) Η χρονική σήμανση (timestamp) του Trigger: όταν δημιουργήθηκε.

Σύνταξη CREATE TRIGGER Statement (SQL-99, Chapter 24)

```
CREATE TRIGGER <Trigger name>
{BEFORE | AFTER} <trigger event> ON <Table name>
[ REFERENCING <old or new values alias list> ]
<triggered action>
   <trigger event> ::=
   INSERT |
   DELETE |
   UPDATE [ OF <trigger Column list> ]
   <trigger Column list> ::= <Column name> [ {,<Column name>} ... ]
   <old or new values alias list> ::=
   <old or new values alias>...
   <old or new values alias> ::=
   OLD [ ROW ] [ AS ] old values <Correlation name> |
   NEW [ ROW ] [ AS ] new values <Correlation name> |
   OLD TABLE [ AS ] <old values Table alias> |
```

```
   NEW TABLE [ AS ] <new values Table alias>
   <old values Table alias> ::= <identifier>
   <new values Table alias> ::= <identifier>
   <triggered action> ::=
   [ FOR EACH {ROW | STATEMENT} ] [ WHEN (search condition) ]
   <triggered SQL statement>
   <triggered SQL statement> ::=
   SQL statement |
   BEGIN ATOMIC {SQL statement;}... END
```

2.13.3 Trigger στο προϊόν της Oracle

Στην περίπτωση του προϊόντος της Oracle χρησιμοποιείται η γλώσσα (τεχνολογία) PL/SQL για τη συγγραφή των triggers.

Επισημαίνουμε τα παρακάτω:

1) Οι Triggers αποθηκεύονται ως ανεξάρτητα αντικείμενα στη βάση δεδομένων,
2) Δεν μπορεί να γίνει κλήση του Trigger. Ενεργοποιείται από ένα συμβάν
3) Ο Trigger δεν μπορεί να λάβει παραμέτρους

Σύνταξη σε PL / SQL για τη δημιουργία των triggers

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER}
-- triggering_event ON table_name
{INSERT | DELETE | UPDATE [OF COLUMN[,COLUMN ... ]]} ON table_name
[FOR EACH ROW]
[FOLLOWS | PRECEDES another_trigger]
[ENABLE / DISABLE]
[WHEN condition]
DECLARE
declaration statements
--trigger_body;
BEGIN
executable statements
EXCEPTION
exception_handling statements
END;
```

Παραδείγματα

```
-- Use :old and :new operators
create or replace trigger tri_update
after
update on employees
for each row
begin
  dbms_ output.put_ Line ('before update: '|: old.salary ||'updated:' |:
new.salary );
end;

-- Write a trigger to print 'hello' when inserting records into the EMP -- table
```

```
create or replace trigger tri_update
after
insert on emp
begin
  dbms_output.put_line('ok');
end;

-- A HelloWorld level trigger
-- Create a trigger to trigger when updating the employees table
create or replace trigger tri_update
after update on employees
For each row
begin
   dbms_output.put_line('ok');
end;
--Execution
update employees
set salary = salary+1
where department_id = 80
```

Τα παραδείγματα που παρατέθηκαν υπάρχουν στα άρθρα:

What is an Oracle trigger, Oracle Trigger (oracletutorial.com) (accessed 25-1-2021)

Case comparison of using triggers in Oracle and MySQL (accessed 25-1-2021)

Case comparison of using triggers in Oracle and MySQL | Develop Paper

Το γεγονός που ενεργοποιεί ένα έναυσμα (trigger) μπορεί να είναι οποιοδήποτε από τα ακόλουθα:

1) Μια δήλωση Γλώσσας Χειρισμού Δεδομένων (DML) που εκτελέστηκε σε έναν πίνακα, π.χ. INSERT, UPDATE ή DELETE.
2) Μια δήλωση Γλώσσας Ορισμού Δεδομένων (DDL) που εκτελείται π.χ. δήλωση CREATE ή ALTER. Αυτοί οι triggers χρησιμοποιούνται συχνά για σκοπούς ελέγχου και για την καταγραφή αλλαγών του σχήματος της βάσης δεδομένων.

Προσοχή! Δεν υποστηρίζεται από όλα τα ΣΔΒΔ.

1) Ένα συμβάν συστήματος όπως εκκίνηση ή τερματισμός της βάσης δεδομένων της Oracle.
2) Ένα συμβάν χρήστη όπως σύνδεση ή αποσύνδεση (“A user event such as login or logout”).
3) Η πράξη εκτέλεσης ενός εναύσματος (trigger) είναι επίσης γνωστή ως ενεργοποίηση του εναύσματος (μιας σκανδάλης, firing a trigger).

Θυμίζουμε ότι σε κάποια μεταφρασμένα στα ελληνικά ξενόγλωσσα συγγράμματα χρησιμοποιείται η ορολογία «ενεργοποιείται η σκανδάλη».

Τα εναύσματα σε ΣΔΒΔ μεγάλων προμηθευτών θα αναλυθούν περαιτέρω σε επόμενο κεφάλαιο του συγγράμματος.

Σύνοψη για τις ενεργές βάσεις δεδομένων. Το μέλλον

Οι ενεργές βάσεις αποτελούν και σήμερα ένα σημαντικό τομέα έρευνας εφαρμογών. Δείτε σχετικά και την παρακάτω ενδεικτική βιβλιογραφία.

Επιπλέον, η έρευνα σε ενεργές βάσεις μεγάλων δεδομένων έχει ξεκινήσει. Δείτε ενδεικτικά:

Ordonez, C., & García-García, J. (2016, May). Managing big data analytics workflows with a database system. In 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) (pp. 649-655). IEEE.

Hongying X., Lingling, J., Liyou, C. (2017) The Application of Database Technology in Information Society and its Existing Problems, Big Data and Cloud Innovation

Jin, Y., Bharath, V., & Shah, J. (2020, March). Active Rules in a Graph Database Environment. In CATA, Proceedings of 35th International Conference on Computers and Their Application, EPiC Series in Computing, vol. 69, pp. 134–14

Csaszar, A. G., Furtenbacher, T., & Arendas, P. (2016). Small Molecules-Big Data. The Journal of Physical Chemistry A, 120(45), 8949-8969.

Το παρακάτω κείμενο αφορά μία σύνοψη των κανόνων Event-condition-action, και αποτελεί προσαρμογή αποσπάσματος από άρθρο (κεφάλαιο βιβλίου) των Milosevic & Rabhi (2016).

«Η προσέγγιση των κανόνων Συμβάντων-Συνθηκών-Δράσεων (ECA) αναπτύχθηκε για να υποστηρίξει την ανάγκη απόκρισης-απάντησης σε διαφορετικά είδη γεγονότων-συμβάντων που συμβαίνουν σε ενεργές βάσεις δεδομένων. Υπάρχουν τρεις συνιστώσες σε έναν κανόνα:

1) **Event**: Καθορίζει το συμβάν που ενεργοποιεί την επίκληση του κανόνα. Το ίδιο το συμβάν μπορεί να είναι μια σύνθεση διαφορετικών τύπων συμβάντων, οπότε ονομάζεται σύνθετο συμβάν.
2) **Συνθήκη**: Αποτελείται από τις προϋποθέσεις που πρέπει να πληρούνται για την εκτέλεση της συγκεκριμένης ενέργειας. Η συνθήκη ελέγχεται μόνο κατά την εμφάνιση του καθορισμένου συμβάντος.
3) **Ενέργεια**: Καθορίζει τις ενέργειες που πρέπει να γίνουν στα δεδομένα»

Παραδείγματα ενεργών συστημάτων βάσης δεδομένων που χρησιμοποιούν κανόνες ECA περιλαμβάνουν το ACCOOD και το Chimera. Εκτός από τις ενεργές βάσεις δεδομένων, οι κανόνες ECA έχουν επίσης εφαρμοστεί σε «συμβατικές» βάσεις δεδομένων», όπου η συνθήκη (condition) είναι ένα παραδοσιακό ερώτημα (query) για την τοπική βάση δεδομένων αλλά και σε «(μηχανισμούς) μηχανές κανόνων που βασίζονται στη μνήμη» (memory-based rule engines), όπου η συνθήκη είναι μια δοκιμή στα τοπικά δεδομένα.

Ενδεικτική βιβλιογραφία

Milosevic, Z., Rabhi, F.A. (2016) Real-Time Analytics, Big Data: Principles and Paradigms, Morgan Kaufmann, ISBN 978-0-12-805394-2

Milosevic, Z., Chen, W., Berry, A., & Rabhi, F. A. (2016). An open architecture for event-based analytics. International Journal of Data science and analytics, 2(1), 13-27. https://doi.org/10.1007/s41060-016-0029-7

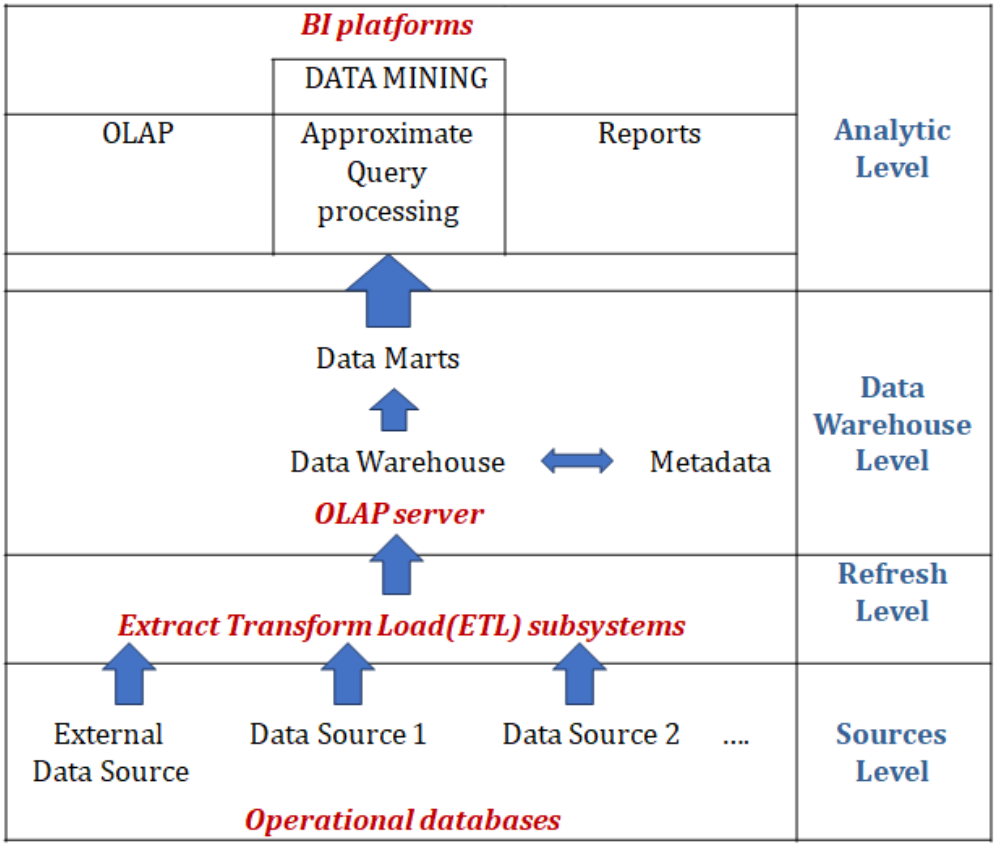
Muzammal, M., Qu, Q., & Nasrulin, B. (2019). Renovating blockchain with distributed databases: An open source system. Future generation computer systems, 90, 105-117.

Wagner, J., Rasin, A., & Grier, J. (2016). Database image content explorer: Carving data that does not officially exist. Digital Investigation, 18, S97-S107.

2.14 Βάσεις Δεδομένων και Επιχειρήσεις. Data Mining, Data Warehouse, Datamart, Business Intelligence, Knowledge Management

Η ενότητα 2.14 βασίζεται στην παρουσίαση του θέματος στο σύγγραμμα,

Μαρινάγη, Α., Σκουρλάς, Χ. (2022) Διαχείριση γνώσης, Κάλλιπος.



Εικόνα 2.40 Αρχιτεκτονική συστημάτων επιχειρηματικής ευφυΐας.

Στην εικόνα 2.40 παρατίθεται η αρχιτεκτονική ενός συστήματος επιχειρησιακής ευφυίας. Τα συστήματα αυτά και οι επιμέρους συνιστώσες τους είχαν και έχουν μία μεγάλη δυναμική στον κόσμο των οργανισμών και των επιχειρήσεων.

Η εικόνα 2.40 αποτελεί και ένα εννοιολογικό πλαίσιο για τα σύγχρονα συστήματα βάσεων δεδομένων και τις εφαρμογές στις επιχειρήσεις. Με αφετηρία το πλαίσιο αυτό θα γίνει παρουσίαση και σύντομη συζήτηση των εννοιών του εννοιολογικού πλαισίου: Relational database, Data Warehousing, ETL, Data Mining, OLAP, Business Intelligence.

Ο τομέας «αποθήκη δεδομένων» (Data Warehousing) αναφέρεται στη δημιουργία κοινόχρηστων «αποθηκών δεδομένων» (data warehouses) σε μεγάλες επιχειρήσεις-οργανισμούς. Οι «αποθήκες» τροφοδοτούνται από πολλές επιμέρους βάσεις τμημάτων (ονομάζονται legacy databases), με τις κατάλληλες «μετεγγραφές» (μετά από επεξεργασία) δεδομένων. Επειδή οι επιμέρους βάσεις (legacy databases) αλλάζουν, η "αποθήκη" πρέπει να ενημερώνεται, όχι βέβαια κατ' ανάγκην άμεσα. Συνήθως, η "αποθήκη" ανακατασκευάζεται περιοδικά, π.χ. κάθε βράδυ ή κάθε εβδομάδα, όταν οι επιμέρους βάσεις δεδομένων χρησιμοποιούνται λιγότερο.

Οι ανάγκες για ανάλυση δεδομένων (data analysis), σχεδιασμό (planning), στήριξη αποφάσεων (decision support) εξυπηρετούνται από την "αποθήκη" ή από μικρότερες ειδικές θεματικές αποθήκες οι οποίες ονομάζονται «πρατήρια δεδομένων» (data marts).

Η εξόρυξη δεδομένων (data mining), δηλαδή η έρευνα για ενδιαφέροντα και ασυνήθιστα μοτίβα (unusual patterns) τα οποία υπάρχουν (υποκρύπτονται) στα δεδομένα συνδέεται με τη δημιουργία των αποθηκών δεδομένων (data warehouses) και μάλιστα υπάρχουν πάρα πολλές εφαρμογές της εξόρυξης δεδομένων σε διάφορους τομείς όπως πωλήσεις κ.λπ.

Ειδικά τα δεδομένα μεγάλης κλίμακας, big data, αξιοποιούνται με χρήση τεχνικών-εργαλείων διαφόρων τομέων, όπως ο τομέας της εξόρυξης δεδομένων, η μηχανική μάθηση (machine learning), η βαθιά μάθηση (deep learning) κ.λπ.

Οι "πελώριες" βάσεις στο διαδίκτυο είναι ίσως το πιο ενδιαφέρον στοιχείο προώθησης νέων ιδεών στο θέμα της κατασκευής εφαρμογών βάσεων.

2.14.1 Αποθήκη δεδομένων (Data Warehouse)

Με τον όρο αποθήκη δεδομένων αναφερόμαστε σε μία τεχνολογία που παρέχει ένα περιβάλλον υποστήριξης λήψης αποφάσεων. Το περιβάλλον αυτό (η αποθήκη) τροφοδοτείται με δεδομένα που φυλάσσονται σε διαφορετικά σημεία του οργανισμού. Τα δεδομένα αποθηκεύονται περιοδικά, μετά από επεξεργασία, σε συγκεκριμένα χρονικά διαστήματα, και είναι οργανωμένα και ενοποιημένα έτσι ώστε να είναι διαθέσιμα στους υπεύθυνους για τη λήψη αποφάσεων.

Η Αποθήκη Δεδομένων ως διεπιστημονικός κλάδος ορίζεται με πολλούς διαφορετικούς τρόπους, συχνά όχι αυστηρά. Απλουστεύοντας λίγο είναι μια βάση δεδομένων υποστήριξης αποφάσεων που διατηρείται χωριστά από την επιχειρησιακή βάση δεδομένων του οργανισμού και υποστηρίζει την επεξεργασία-ανάλυση των πληροφοριών, παρέχοντας μια σταθερή πλατφόρμα ανάλυσης ενοποιημένων ιστορικών στοιχείων.

Ορισμός Inmon

"A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision-making process."—W. H. Inmon

Με βάση τον ορισμό του Inmon τα δεδομένα είναι:

1) Προσανατολισμένα στη θεματολογία (Subject-oriented). Η αποθήκη οργανώνεται γύρω από τα κύρια αντικείμενα της επιχείρησης (Πελάτες, Προϊόντα, Πωλήσεις) και όχι γύρω από τις συνήθεις εφαρμογές που χρησιμοποιεί η επιχείρηση, όπως έκδοση τιμολογίων, έλεγχος αποθεμάτων.
2) Ολοκληρωμένα (integrated) λόγω της «ένωσης» (ενοποίησης, integration) με δεδομένα από άλλα συστήματα που συχνά είναι ακόμη και ασύμβατα μεταξύ τους και απαιτείται ειδική επεξεργασία για την ενοποίηση τους.
3) Χρονικά κυμαινόμενα (time variant) επειδή τα δεδομένα στην αποθήκη είναι ακριβή και έγκυρα μόνο για κάποιο χρονικό διάστημα.
4) «Αναλλοίωτα» (non-volatile), τα δεδομένα είναι μόνο για ανάγνωση και ανανεώνονται περιοδικά (προστίθενται νέα)

Άλλοι ορισμοί

Κατά τους Lee, Son και Kim η αποθήκη δεδομένων αποτελεί πλέον μια κοινή τεχνολογία που χρησιμοποιείται για να εφοδιάζει αναλυτές και μάνατζερ με πληροφορίες στρατηγικής σημασίας για τον οργανισμό. Ο Bagchi ορίζει την αποθήκη δεδομένων ως ένα χώρο αποθήκευσης λειτουργικών στοιχείων (operational data) συγκεντρωμένων από βάσεις δεδομένων παραγωγής και άλλες πηγές ("A data Warehouse (DW) refers to a storehouse of business information gathered from production databases and multiple other sources"). Ο Date C.J. αναφέρεται στην αποθήκη δεδομένων ως ειδική κατηγορία βάσεων δεδομένων. Ο Huyn N. περιγράφει την αποθήκη δεδομένων ως μια συλλογή από υλοποιημένες όψεις (materialized views) με την έννοια ότι αποτελείται από δεδομένα που προκύπτουν με κάποια επεξεργασία από τα δεδομένα του οργανισμού ή σε κάποιες περιπτώσεις τα δεδομένα δεν υπάρχουν αποθηκευμένα στην πραγματικότητα αλλά προκύπτουν από

επεξεργασία. Στην τελευταία περίπτωση, όταν τίθεται ένα ερώτημα, απαντάται τοπικά χωρίς πρόσβαση στις πρωτότυπες πληροφοριακές πηγές.

Κύριος στόχος της λειτουργίας αποθήκης δεδομένων είναι η ενοποίηση-ενσωμάτωση όλων των δεδομένων της επιχείρησης ή του οργανισμού σε μια και μοναδική αποθήκη την οποία όλοι οι χρήστες μπορούν να χρησιμοποιήσουν για να υποβάλουν/εκτελέσουν ερωτήματα, να παράγουν αναφορές και να κάνουν ανάλυση δεδομένων (Μαρινάγη & Σκουρλάς, 2022).

Δείτε και τις παρακάτω πηγές,

Císaro, S. E. G. C., & Nigro, H.O. (2009). Architecture for Symbolic Object Warehouse. Encyclopedia of Data Warehouse, 58-65.

Connolly, T., & Begg, C. (2015). Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition.

Date, C.J. (2003). An Introduction to Database Systems. Addison-Wesley, 8th edition.

Di Tria, F., Lefons, E., & Tangorra, F. (2013). Evaluation of Business Intelligence Systems. International Journal of Information Processing and Management (IJIPM), 4(3).

Elmasri, R., & Navathe, S.B. (2016). Fundamentals of Database Systems, 7th edition, 841-874.

Gonzales, M., Bagchi, K., Kirs, P., & Udo, G. (2011). Diffusion of Business Intelligence and Data Warehousing: An Exploratory Investigation of Research and Practice, In Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS), Jan. 4-7, Hawaii, 1-9, IEEE Press. https://doi.ieeecomputersociety.org/10.1109/HICSS.2011.166

Huyn. N. (1997). Maintaining data warehouses under limited source access. Ph.D. Thesis STAN-CSTR-97-1595, Stanford University.

Inmon, B. (1999). Data Mart does not Equal Data Warehouse, DM Review, Nov. 1999 Ανακτήθηκε από http://www.dmreview.com/master.cfm?NavID=55&EdID=1675

Inmon, W.H. (2002). Building the Data Warehouse. John Wiley &Sons Inc, 3rd edition.

Lee, K.Y., Son, J.H. & Kim., M.H. (2001). Efficient incremental view maintenance in data warehouses, Proceedings of the tenth international conference on Information and knowledge management (CIKM 01), 349–356, https://doi.org/10.1145/502585.502644

Les Pang (2009). Best Practices in Data Warehousing. Encyclopedia of Data Warehouse, 146-152.

2.14.2 Extraction-Transformation-Loading (ETL) και αποθήκες δεδομένων

Η διαδικασία εξαγωγής, μετασχηματισμού και φόρτωσης (Extraction, Transformation, Loading) ETL αποτελεί τη θεμελιώδη διαδικασία στην τεχνολογία αποθήκης δεδομένων (DW). Περιλαμβάνει την εξαγωγή δεδομένων από διάφορες πηγές, τον καθαρισμό (cleansing), την προσαρμογή (customization) και την εισαγωγή τους σε αποθήκη. Τα δεδομένα συχνά περιλαμβάνουν κείμενο, ήχο, εικόνες, βίντεο κ.λπ., είναι πολύμορφα (multimodal) και μπορεί να έχουν πολύπλοκες δομές. Η ολοκλήρωση-ενσωμάτωση (integration) δεδομένων με διαφορετική σημασιολογία (different semantics) καθιστά τη διαδικασία ETL δύσκολη σε πραγματικές συνθήκες (Císaro & Nigro, 2009). Απαιτείται η εκπόνηση αποτελεσματικής στρατηγικής ETL που θα εξάγει δεδομένα από τα διάφορα συστήματα συναλλαγών (transactional systems), θα μετατρέπει τα δεδομένα σε μια κοινή μορφή (common format) και θα φορτώνει τα δεδομένα σε μια σχεσιακή ή πολυδιάστατη βάση δεδομένων (relational or multidimensional database) (Les Pang, 2009) (Μαρινάγη & Σκουρλάς, 2022).

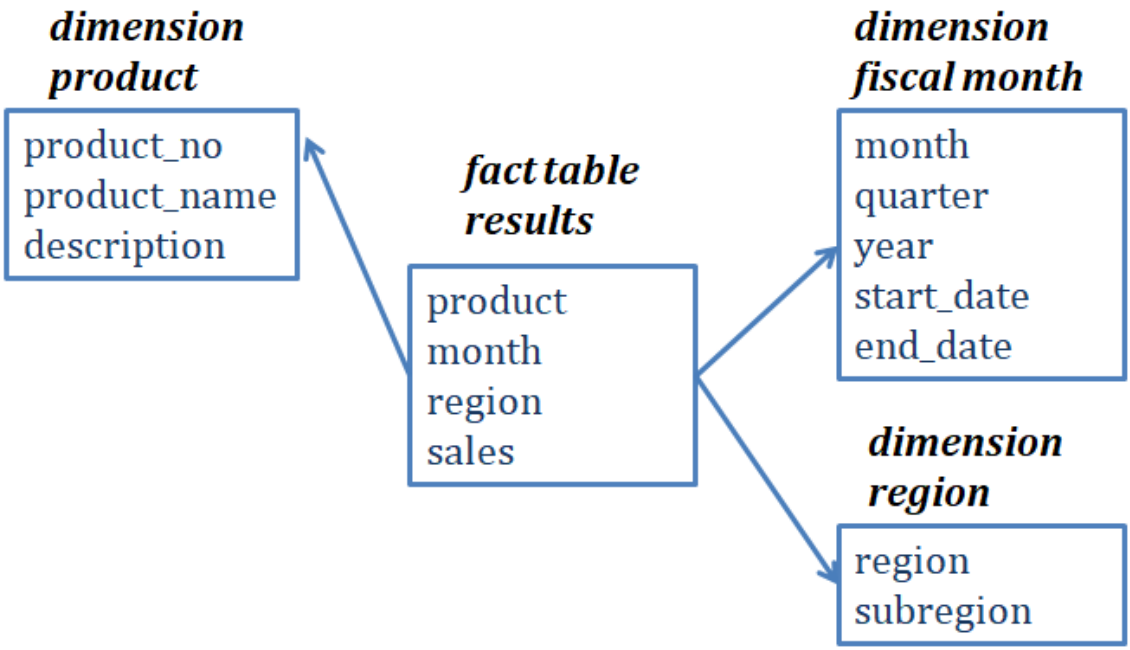
Σύμφωνα με τους Elmasri και Navathe η διαδικασία επεξεργασίας αποθηκών δεδομένων (Data warehouse processing) περιλαμβάνει (Elmasri & Navathe 2016, fig. 29.1):

1) Cleaning and reformatting of data,
2) ETL (Extract, Transform, Load),
3) OLAP – Data Analytics,
4) Data Mining.

2.14.3 Αρχιτεκτονικές αποθήκης δεδομένων. Πρατήρια δεδομένων (data marts)

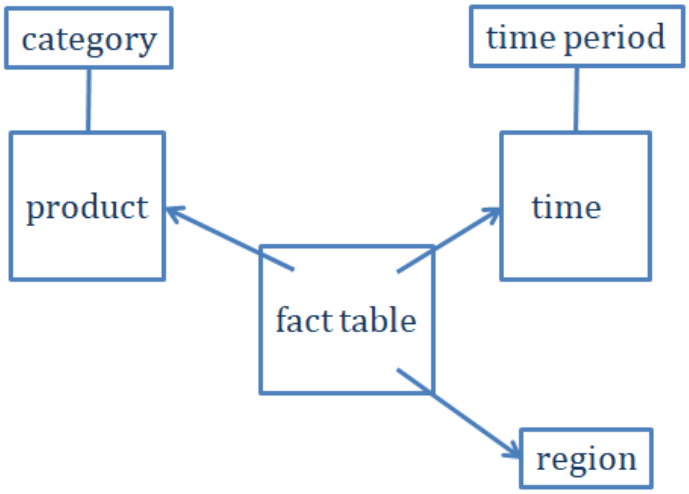
Δημοφιλείς αρχιτεκτονικές αποθήκης δεδομένων είναι η αρχιτεκτονική με αστεροειδές σχήμα και η αρχιτεκτονική με σχήμα χιονονιφάδας. Το σχήμα (schema) σε μια σχεσιακή βάση δεδομένων είναι η συλλογή αντικειμένων της βάσης δεδομένων, στα οποία συμπεριλαμβάνονται πίνακες, όψεις, ευρετήρια και συνώνυμα (tables, views, indexes, synonyms).

Στην εικόνα 2.41 παρουσιάζεται το αστεροειδές σχήμα (Star schema) της αρχιτεκτονικής αποθήκης δεδομένων. Περιλαμβάνει πίνακα γεγονότων (fact table) και τρεις πίνακες διαστάσεων (dimension tables).



Εικόνα 2.41 Αρχιτεκτονική σε αστεροειδές σχήμα (Μαρινάγη και Σκουρλάς, 2022)

Στην εικόνα 2.42 παρουσιάζεται ένα σχήμα χιονονιφάδας.



Εικόνα 2.42 Αρχιτεκτονική σε σχήμα χιονονιφάδας.

2.14.3.1 Βιβλιογραφία

Lee, K. Y., Son, J. H., & Kim, M. H. (2001, October). Efficient incremental view maintenance in data warehouses. In Proceedings of the tenth international conference on Information and knowledge management (pp. 349-356).

Gonzales, M. L., Bagchi, K., Udo, G., & Kirs, P. (2011, January). Diffusion of business intelligence and data warehousing: An exploratory investigation of research and practice. In 2011 44th Hawaii International Conference on System Sciences (pp. 1-9). IEEE.

Huyn, N. (1997) Maintaining data warehouses under limited source access. Ph.D. Thesis STAN-CS-TR-97-1595, Stanford University.

Abiteboul, S., & Duschka, O. M. (1998, May). Complexity of answering queries using materialized views. In Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (pp. 254-263).

2.14.3.2 Πρατήρια δεδομένων (Datamarts)

Μία αποθήκη δεδομένων σε επίπεδο οργανισμού ή επιχείρησης είναι συχνά πολύπλοκη και ευμεγέθης. Πολλές εταιρείες σχεδιάζουν και υλοποιούν μικρότερες θεματικές αποθήκες, datamarts, που ανταποκρίνονται στις ανάγκες εξειδικευμένων ομάδων τελικών χρηστών (end-users groups). Σύμφωνα με τους Connolly & Begg (2015), τα πρατήρια δεδομένων (datamarts) είναι υποσύνολα των αποθηκών δεδομένων τα οποία υποστηρίζουν τις απαιτήσεις συγκεκριμένων τμημάτων ή λειτουργιών. Λειτουργούν είτε αυτοτελώς (standalone) ή ως συνδεδεμένα πρατήρια με την κεντρική εταιρική αποθήκη δεδομένων. Κάποιες εταιρείες οικοδομούν αρχικά πρατήρια δεδομένων (datamarts) και προχωρούν στη συνέχεια σε ολοκληρωμένες αποθήκες δεδομένων (Connolly & Begg, 2015) (Μαρινάγη & Σκουρλάς, 2022). Δείτε σχετικά και το σύγγραμμα,

Thomas Connolly, Carolyn Begg, (2015) Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition

2.14.4 Τεχνολογία αποθηκών δεδομένων, ανάλυση δεδομένων και υποστήριξη αποφάσεων (Decision Support/Making)

Η λήψη αποφάσεων απαιτεί ανάλυση του οργανισμού, της επιχείρησης, των απαιτήσεων, των τάσεων κ.λπ. Συνεπώς, οι υπεύθυνοι για τη λήψη αποφάσεων πρέπει να έχουν:

1) Πρόσβαση σε όλα τα δεδομένα του οργανισμού ανεξάρτητα από το χώρο αποθήκευσης τους.
2) Πρόσβαση σε ενημερωμένα στοιχεία αλλά και σε («ιστορικά») στοιχεία προηγουμένων μηνών, ετών κ.λπ..
3) Συνόψεις/αναφορές και πρόσβαση σε οπτικοποιημένα δεδομένα

Η αποθήκη δεδομένων βασίζεται σε επεκτάσεις της τεχνολογίας βάσεων δεδομένων και παρέχει τη δυνατότητα διαχείρισης επικαλυπτόμενων, ετερογενών, κατανεμημένων δεδομένων και τη δυνατότητα υποστήριξης της λήψης αποφάσεων με τη χρησιμοποίηση ισχυρών εργαλείων. Παραθέτουμε συνιστώσες και εργαλεία της αποθήκης δεδομένων:

- Βάση δεδομένων
- Μεταδεδομένα (metadata), δηλαδή καταχωρημένες «συνόψεις» των δεδομένων κ.λπ. στη βάση δεδομένων που περιγράφουν και την αποθήκη δεδομένων
- Εργαλεία μοντελοποίησης δεδομένων (data modeling)
- Εργαλεία μεταφοράς/φόρτωσης δεδομένων. Τα εργαλεία έχουν πρόσβαση σε πηγές (βάσεις) δεδομένων που επιλέγονται επειδή έχουν δεδομένα χρήσιμα για την αποθήκη
- Εργαλεία ανακάλυψης και εξαγωγής δεδομένων
- Εργαλεία και εξοπλισμός διασύνδεσης (connectivity) για χειρισμό του περιβάλλοντος της αποθήκης δεδομένων
- Εργαλεία data mining, On-Line Analytical Processing κ.ά.

2.14.5 Επιχειρηματική Ευφυία (Business Intelligence)

Η **επιχειρηματική ευφυΐα–ΕΕ (Business Intelligence−BI)** είναι μια τεχνολογία που αποσκοπεί στη βελτίωση των επιχειρηματικών διαδικασιών και του πληροφοριακού συστήματος της επιχείρησης. Βασίζεται σε ένα σύνολο διαδικασιών και εργαλείων λογισμικού. Στόχος είναι να υποστηρίξει τη λήψη αποφάσεων. Η επιτυχία εφαρμογής της τεχνολογίας εξαρτάται από την αποτελεσματικότητα πολλών επιχειρηματικών διαδικασιών (business processes) και απαιτεί τη σωστή ενσωμάτωση διαφορετικών εργαλείων λογισμικού και την εξαγωγή υψηλής ποιότητας πληροφορίας (από σύνθετα δεδομένα) χρήσιμης για τη βελτίωση των επιχειρηματικών διαδικασιών (Di Tria, Lefons & Tangorra, 2013) (Μαρινάγη και Σκουρλάς, 2022)

Tria, F. D., Lefons, E., & Tangorra, F. (2013). Ontological approach to data warehouse source integration. In Information Sciences and Systems 2013 (pp. 251-259). Springer, Cham.

Ο ορισμός της επιχειρηματικής ευφυΐας τον οποίο έδωσαν οι Turban, Sharda, Delen, King, και Aronson (2011) είναι ο ακόλουθος:

Επιχειρηματική ευφυΐα (ΕΕ) είναι ένας γενικός όρος που συνδυάζει αρχιτεκτονικές, εργαλεία, βάσεις δεδομένων, αναλυτικά εργαλεία, εφαρμογές και μεθοδολογίες. Είναι μια έκφραση ελεύθερη περιεχομένου (content-free expression), έτσι ώστε να σημαίνει διαφορετικά πράγματα σε διαφορετικούς ανθρώπους. Κύριος στόχος της ΕΕ είναι να επιτρέπει την εύκολη πρόσβαση στα δεδομένα (και μοντέλα) για να παρέχει στους managers των επιχειρήσεων την ικανότητα να διεξάγουν ανάλυση. Η ΕΕ βοηθά τη μετατροπή των δεδομένων σε πληροφορίες (και γνώσεις), για την υποστήριξη αποφάσεων και τελικά στήριξη της επιχειρηματικής δράσης.

(BI is an umbrella term that combines architectures, tools, databases, analytical tools, applications, and methodologies. BI a content-free expression, so it means different things to different people. BI's major objective is to enable easy access to data (and models) to provide business managers with the ability to conduct analysis.BI helps transform data, to information (and knowledge), to decisions and finally to action).

Turban, E., Sharda, R., Delen, D., King, D., & Aronson, J. E. (2011). Business Intelligence: a managerial approach. 2nd ed. Upper Saddle River: Pearson Prentice Hall.

Στη βιβλιογραφία υπάρχουν διάφορες αρχιτεκτονικές των συστημάτων επιχειρηματικής ευφυΐας. Στην εικόνα 2.40 παρουσιάζεται μία αρχιτεκτονική. Βλέπε και άρθρο (Di Tria *et al.*, 2013).

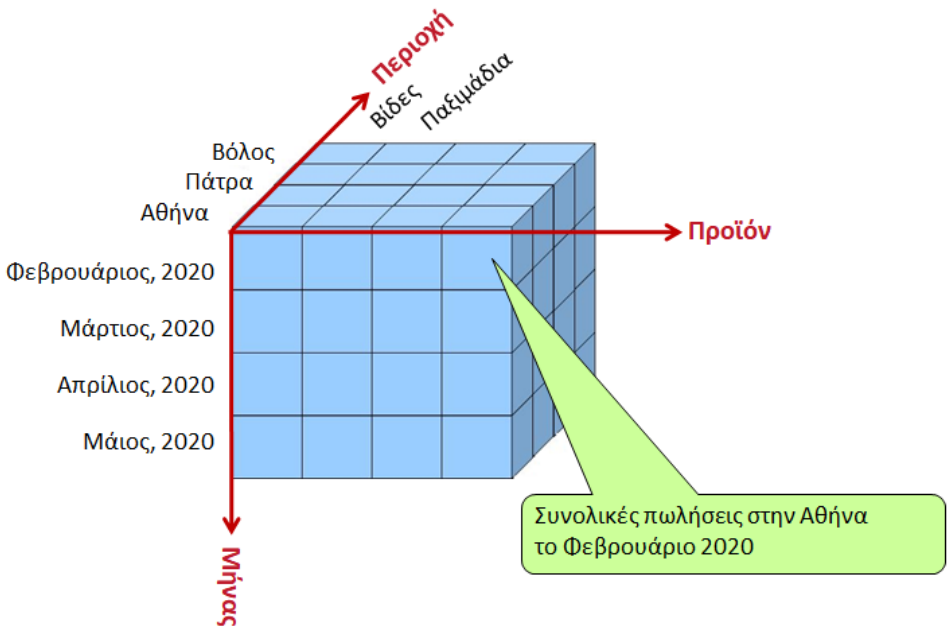
2.14.6 Online Analytical Processing−OLAP

Ο όρος **OLAP (Online Analytical Processing)** πρωτοεμφανίστηκε σε ένα κείμενο (white paper) της Arbor Software Corp., το 1993, αν και η έννοια είναι αρκετά παλαιότερη. Μπορεί να οριστεί ως η διαλογική διαδικασία δημιουργίας, διαχείρισης, ανάλυσης και παρουσίασης δεδομένων (the interactive process of creating, managing, analyzing, and reporting on data). Είναι σύνηθες να προστίθεται στον ορισμό ότι τα δεδομένα γίνονται αντιληπτά και τα επεξεργαζόμαστε (is perceived and manipulated) σαν να ήταν αποθηκευμένα σε πολυδιάστατους πίνακες (multi-dimensional array) (Μαρινάγη και Σκουρλάς, 2022).

Στην εικόνα 2.44 βλέπουμε ένα παράδειγμα δεδομένων δύο διαστάσεων και στην εικόνα 2.45 ένα παράδειγμα πολυδιάστατων δεδομένων (Multidimensional Data), πιο συγκεκριμένα, έναν κύβο δεδομένων (Data Cube). Βλέπουμε τον όγκο των πωλήσεων ως συνάρτηση τριών διαστάσεων (dimensions): προϊόν (product), μήνας (month), περιοχή (region).

	region1	region2	region3
product 1			
product 2			
product 3			

Εικόνα 2.43 Δεδομένα δύο διαστάσεων (Μαρινάγη και Σκουρλάς, 2022)



Εικόνα 2.44 Κύβος δεδομένων (Μαρινάγη και Σκουρλάς, 2022)1

2.14.7 Εξόρυξη δεδομένων (Data Mining)

Η διαδικασία ανακάλυψης και εξαγωγής γνώσης (Knowledge Discovery in Databases-KDD) είναι μια αλληλεπιδραστική, επαναληπτική διαδικασία που περιλαμβάνει ένα σύνολο βημάτων όπου πολλές αποφάσεις λαμβάνονται από τον χρήστη (Fayyad *et al.,* 1996). Τα σημαντικότερα από τα βήματα αυτά είναι (Μαρινάγη και Σκουρλάς, 2022):

1) **Κατανόηση του πεδίου εφαρμογής** και **προσδιορισμός των στόχων** της διαδικασίας KDD από την πλευρά του πελάτη.
2) **Δημιουργία συνόλου επιλεγμένων δεδομένων** (παρατηρήσεις (observations), ή παραδείγματα (examples)) και μεταβλητών (data set) στο οποίο θα επικεντρωθεί η διαδικασία της ανακάλυψης και εξόρυξης.
3) **Προ-επεξεργασία (pre-processing)** και καθαρισμός των δεδομένων ώστε να απαλλαγεί η πληροφορία από θόρυβο, απόφαση των στρατηγικών διαχείρισης των ελλιπών δεδομένων κ.λπ.
4) **Επιλογή της μεθόδου εξόρυξης** (data mining) και των αλγορίθμων που θα χρησιμοποιηθούν.
5) **Εξόρυξη δεδομένων** (data mining).
6) **Επεξεργασία (interpretation) των προτύπων** (patterns) που προέκυψαν. Πιθανή επιστροφή σε κάποιο από τα προηγούμενα βήματα και επαναληπτική εκτέλεσή τους. Επίσης το βήμα περιλαμβάνει την οπτικοποίηση (visualization) των αποτελεσμάτων.
7) **Χρήση της παραγόμενης γνώσης**, άμεσα, ή ενσωματώνοντάς της σε ένα άλλο σύστημα, ή τεκμηριώνοντάς την για χρήση από τους ενδιαφερομένους.
8) **Αξιολόγηση**. Επιλύονται πιθανές συγκρούσεις με προηγούμενη εξαγόμενη γνώση.

Η εφαρμογή της ανακάλυψης και εξαγωγής γνώσης (KDD) στις επιχειρήσεις είναι ιδιαίτερα σημαντική και περιλαμβάνει για παράδειγμα το μάρκετινγκ, τα οικονομικά (ιδιαίτερα τις επενδύσεις), τις κατασκευές, τις τηλεπικοινωνίες, την ανίχνευση απάτης.

Η εξόρυξη δεδομένων είναι το πιο σημαντικό από τα βήματα της διαδικασίας ανακάλυψης και εξαγωγής γνώσης (KDD) και μπορεί να περιγραφεί σαν μια διερευνητική («εξερευνητική») ανάλυση δεδομένων (exploratory data analysis). Στόχος είναι η αναζήτηση προτύπων (patterns) στα δεδομένα που μπορούν να χρησιμοποιηθούν σε πολύ διαφορετικές περιπτώσεις, από τον καθορισμό της στρατηγικής της επιχείρησης (business strategy) μέχρι την ανίχνευση ασυνήθιστης συμπεριφοράς. Για παράδειγμα, μια ξαφνική αύξηση στη χρήση μιας πιστωτικής κάρτας θα μπορούσε να σημαίνει ότι η κάρτα έχει κλαπεί (Μαρινάγη και Σκουρλάς, 2022). Συγγράμματα όπως του Κύρκου (2015, εκδόσεις Κάλλιπος) καλύπτουν με λεπτομέρεια τα θέματα της εξόρυξης δεδομένων. Στο παρόν κεφάλαιο 2 το θέμα της εξόρυξης δεδομένων προσεγγίζεται κυρίως μέσω παραδειγμάτων.

Παραθέτουμε κάποια παραδείγματα χρησιμοποίησης εξόρυξης δεδομένων σε διάφορους τομείς:

- Προσδιορισμός αγοραστικής συμπεριφοράς πελατών
- Ανακάλυψη μοτίβων-μοντέλων αγορών (shopping patterns) και τάσεων (trends) των πελατών
- Βελτίωση ποιότητας παροχής υπηρεσιών πελάτη
- Ικανοποίηση πελάτη
- Σχεδίαση αποτελεσματικότερης διανομής και μεταφοράς (delivery, transportation)
- Μείωση κόστους των επιχειρήσεων

Μεθοδολογία εξόρυξης δεδομένων

Στο πλαίσιο της διαχείρισης δεδομένων απαιτείται μια μεθοδολογία για την οργάνωση των εργασιών που σχετίζονται με την ανάκτηση, την επεξεργασία, την ανάλυση και την επαναχρησιμοποίηση των δεδομένων και την ανακάλυψη γνώσης **Invalid source specified.**. Η ανακάλυψη γνώσης περιλαμβάνει την επιλογή συχνά ετερογενών ή και μεγάλων δεδομένων από διαφορετικές πηγές, την προεπεξεργασία τους για τη διαχείριση ασυνεπών, εσφαλμένων ή ελλειπόντων δεδομένων, τον μετασχηματισμό τους ώστε τα δεδομένα να έχουν κοινό μορφότυπο (format) για τη διευκόλυνση της επεξεργασίας τους, την εξόρυξη δεδομένων για την κατασκευή μοντέλων και την οπτικοποίησή τους, και τέλος, την ερμηνεία, την αξιολόγηση και την εφαρμογή των αποτελεσμάτων (Παπαϊωάννου, 2022).

Παπαϊωάννου, Παναγιώτης (2022) Εξόρυξη δεδομένων και σεμινάρια επιμόρφωσης φορέων του δημόσιου τομέα, μεταπτυχιακή διπλωματική εργασία, Πανεπιστήμιο Δυτικής Αττικής.

Όπως, ήδη, αναφέρθηκε η εξόρυξη δεδομένων αποτελεί σημαντικό μέρος της διαδικασίας ανακάλυψης γνώσης. Μια δημοφιλής μεθοδολογία είναι Cross Industry Process for Data Mining (CRISP-DM) (βλέπε εικόνα 2.46), η οποία αφορά την προτυποποίηση των διεργασιών εξόρυξης δεδομένων και περιλαμβάνει έξι φάσεις (North, 2012): 1) Business understanding, 2) Data understanding, 3) Data preparation, 4) Modeling, 5) Evaluation, 6) Deployment.

Εικόνα 2.45 Οι φάσεις του μοντέλου CRISP-DM (https://commons.wikimedia.org/wiki/File:CRISP-DM_Process_Diagram.png, Kenneth Jensen, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons)

Δείτε σχετικά και το βιβλίο,

Erl, T., Khattak, W., & Buhler, P. (2016). Big data fundamentals: concepts, drivers & techniques. Prentice Hall Press.

2.14.7.1 Εξόρυξη δεδομένων και μεγάλα δεδομένα

Η σημασιολογική ανάλυση (Semantic analysis) των δεδομένων μεγάλης κλίμακας αφορά την ανάλυση δεδομένων κειμένου, ομιλίας, εικόνας ή βίντεο για να προσδιοριστεί το «νόημα», η «σημασία». Στη σχετική έρευνα αξιοποιούνται ερευνητικά αποτελέσματα σε τεχνολογικές δραστηριότητες, όπως “named entity recognition”, “word sense disambiguation”, “facial recognition”, “Optical Character Recognition”, κ.λπ.

Παραθέτουμε σύντομη αναφορά σε αυτές τις τεχνολογίες-διαδικασίες.

- Αναγνώριση επώνυμης οντότητας (Named-entity recognition, NER) είναι η τεχνολογία-διαδικασία εξαγωγής «ονομαστικών οντοτήτων» (named entities) από μη δομημένο κείμενο και ταξινόμησης των οντοτήτων σε προκαθορισμένες κατηγορίες, όπως ονόματα προσώπων, οργανισμοί, τοποθεσίες, ιατρικοί κώδικες, χρονικές περιόδους, κ.λπ.
- Αποσαφήνιση της έννοιας της λέξης (word sense disambiguation, WSD)) είναι η τεχνολογία-διαδικασία του προσδιορισμού της «νοήματος» (σημασίας) μιας λέξης σε ένα συγκεκριμένο πλαίσιο («από τα συμφραζόμενα»).
- Σύστημα αναγνώρισης προσώπου (facial recognition system) είναι η τεχνολογία- διαδικασία, η οποία χρησιμοποιεί μια βάση ψηφιακών εικόνων προσώπων για την ταυτοποίηση ενός προσώπου από ψηφιακή εικόνα ή βίντεο (καρέ βίντεο, video frame).
- Οπτική αναγνώριση χαρακτήρων (Optical Character Recognition, OCR), είναι η τεχνολογία-διαδικασία αναγνώρισης κειμένου μέσα σε εικόνες και η μετατροπή του σε ηλεκτρονική μορφή. Οι εικόνες θα μπορούσαν να είναι φωτογραφίες, σκαναρισμένα έγγραφα ή χειρόγραφα, κ.λπ. Τα συστήματα OCR λειτουργούν σε δύο φάσεις: 1) φάση ανίχνευση κειμένου μέσα στην εικόνα, π.χ. «εντοπισμός» της περιοχής του αριθμού κυκλοφορίας μέσα σε φωτογραφία αυτοκινήτου, 2) αναγνώριση κειμένου (δηλαδή το κείμενο εξάγεται) από την εικόνα.

Η σημασιολογική ανάλυση (Semantic analysis) των δεδομένων μεγάλης κλίμακας περιλαμβάνει τους εξής τομείς:

- Επεξεργασία φυσικής γλώσσας (natural language processing): Αποτελεί πεδίο της τεχνητής νοημοσύνης και αφορά την κατανόηση ανθρώπινης ομιλίας και κειμένου.
- Ανάλυση κειμένου (Text Analytics): Είναι η διαδικασία ανάλυσης αδόμητου κειμένου με στόχο την εξαγωγή πληροφορίας.
- Ανάλυση συναισθήματος ή ανάλυση «κλίματος» (Sentiment Analysis): Είναι η ανάλυση της άποψης του χρήστη για να δημιουργηθούν κριτικές, π.χ. για προϊόν (μάρκετινγκ, οικονομία, κοινωνικές-πολιτικές επιστήμες).

Παραθέτουμε παραδείγματα τύπων κειμένου: emails, blogs, forums, social network feeds, απαντήσεις σε έρευνες, έγγραφα, αρχεία καταγραφής τηλεφωνικών κέντρων κ.λπ.

Στον πίνακα 2.8 επισημαίνουμε τις διαφορές της εξόρυξης δεδομένων και της ανάλυσης μεγάλων δεδομένων (Big Data analytics).

Πίνακας 2.8 Διαφορές Εξόρυξης δεδομένων και Big Data analytics

Χαρακτηριστικά	Εξόρυξη δεδομένων	Big data analytics
Volume (Όγκος)	GB, TB	TB, PB, EB, ZB
Velocity (Ταχύτητα)	Κεντρικές Βάσεις Δεδομένων	Κατανεμημένες Βάσεις Δεδομένων
Variety (Ποικιλία)	Δομημένα δεδομένα	Ημιδομημένα, αδόμητα δεδομένα
Veracity & Validity (Ορθότητα & αξιοπιστία)	Αξιοπιστία δεδομένων	Κίνδυνος αναξιόπιστων δεδομένων
Value (Αξία)	Παραδοσιακές μέθοδοι δημιουργίας αξίας	Απαιτείται σωστή εκμετάλλευση νέων μεθόδων

Στην επόμενη ενότητα 2.15 θα παρουσιαστούν θέματα εξόρυξης κειμένου (Text mining), ανάλυσης συναισθήματος (Sentiment Analysis) και εξαγωγής όρων ευρετηρίου από ιστοσελίδες (Web pages) με χρήση του εργαλείου RapidMiner.

2.14.8 Αναφορές και ενδεικτική βιβλιογραφία

Bhavani, M. T. (1999). Data Mining: Technologies, Techniques, Tools and Trends. CRC Press

Witten and Frank (2006) Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann Publishers

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. AI magazine, 17(3), 37-37.

Frawley, W. J., Piatetsky-Shapiro, G., & Matheus, C. J. (1992). Knowledge discovery in databases: An overview. AI magazine, 13(3), 57-57.

Bissantz, N., Hagedorn, J. (2009) Data Mining. Bus. Inf. Syst. Eng. 1, 118–122

Matthew North (2012) Data Mining for the Masses, ISBN 978-0615684376, This book is licensed under a Creative Commons Attribution 3.0 License

2.14.9 Βιβλιογραφία. Συγγράμματα αποθετηρίου «Κάλλιπος»

Η ενότητα αυτή του κεφαλαίου 2 γράφτηκε συνοπτικά, αν και λόγω της σημασίας της θα μπορούσε να αποτελέσει ξεχωριστό κεφάλαιο, επειδή υπάρχουν συγγράμματα στο αποθετήριο «Κάλλιπος». Ο αναγνώστης παραπέμπεται στα συγγράμματα αυτά.

Σταλίδης, Γ., Καρδαράς, Δ. (2015) Διαχείριση Δεδομένων και Επιχειρηματική Ευφυία. Θεωρία και Εφαρμογές για στελέχη επιχειρήσεων, Εκδόσεις Κάλλιπος

Κύρκος, Ε. (2015) Επιχειρηματική ευφυΐα και εξόρυξη δεδομένων. Εκδόσεις Κάλλιπος.

Συμεωνίδης, Π., Γούναρης, Α. (2015) Βάσεις, Αποθήκες και εξόρυξη δεδομένων με τον SQL Server. Εργαστηριακός οδηγός. Εκδόσεις Κάλλιπος (Στις ασκήσεις του βιβλίου χρησιμοποιούνται Microsoft SQL Server, Access).

Βερύκιος, Β., Καγκλής, Β., Σταυρόπουλος, Η. (2015). Η επιστήμη των δεδομένων μέσα από τη γλώσσα R, Εκδόσεις Κάλλιπος

Μαρινάγη, Α., Σκουρλάς, Χ. (2022) Διαχείριση γνώσεις , Εκδόσεις Κάλλιπος

Βερύκιος, Β., & Βασιλακόπουλος, Μ. (2022). Συστήματα Βάσεων Δεδομένων [Προπτυχιακό εγχειρίδιο]. Κάλλιπος, Ανοικτές Ακαδημαϊκές Εκδόσεις. http://dx.doi.org/10.57713/kallipos-36

2.15 Εισαγωγή στην εξόρυξη κειμένου (Text mining) και ανάλυση συναισθήματος (Sentiment Analysis) με χρήση του εργαλείου RapidMiner

Σύμφωνα με τον Marti Hearst (October 17, 2003),

«Η εξόρυξη κειμένου είναι η ανακάλυψη από τον υπολογιστή νέων, προηγουμένως άγνωστων πληροφοριών, με αυτόματη εξαγωγή πληροφοριών από διαφορετικούς γραπτούς πόρους (από κείμενα). Ένα βασικό στοιχείο είναι η σύνδεση των εξαγόμενων πληροφοριών μεταξύ τους για να σχηματιστούν νέα γεγονότα ή νέες υποθέσεις που θα διερευνηθούν περαιτέρω με πιο συμβατικά μέσα πειραματισμού»

("Text Mining is the discovery by computer of new, previously unknown information, by automatically extracting information from different written resources. A key element is the linking together of the extracted information together to form new facts or new hypotheses to be explored further by more conventional means of experimentation").

Marti A. Hearst (1999) Untangling Text Data Mining, Proceedings of ACL'99: the 37th Annual Meeting of the Association for Computational Linguistics, University of Maryland, June 20-26, 1999 (invited paper)

Untangling Text Data Mining (berkeley.edu)

Marti Hearst: What Is Text Mining? (berkeley.edu)

Οι γραπτοί πόροι μπορεί να περιλαμβάνουν ιστότοπους, ηλεκτρονικά βιβλία, μηνύματα ηλεκτρονικού ταχυδρομείου, κριτικές και άρθρα.

Ανάλυση συναισθήματος (Sentiment Analysis) είναι η διαδικασία (process) της κατηγοριοποίησης τμημάτων κειμένου ως θετικών ή αρνητικών.

2.15.1 Παράδειγμα εξαγωγής όρων ευρετηρίου. Εξόρυξη κειμένου (Text mining) με το εργαλείο Rapid Miner

Ακολουθεί παράδειγμα, που έχει ως στόχο την ανάγνωση ενός ή περισσότερων κειμένων και την εξαγωγή όρων ευρετηρίου. Για την επεξεργασία κειμένου (ή κειμένων) πρέπει να πραγματοποιηθούν τα εξής βήματα:

1) **Διάσπαση κειμένου σε λέξεις** (tokenize).
2) **Φιλτράρισμα μη σημαντικών λέξεων** (filter stopwords). Υπάρχουν λεξικά με stopwords για διάφορες γλώσσες στο εργαλείο Rapid Miner, όχι όμως για τα ελληνικά. Βέβαια μπορεί να κατασκευαστεί λεξικό μη σημαντικών λέξεων για τα ελληνικά ή για οποιαδήποτε γλώσσα.
3) **Φιλτράρισμα λέξεων δεδομένου μήκους** (filter tokens by length), π.χ. μπορούν να εξαιρεθούν λέξεις με λιγότερα από 3 γράμματα και περισσότερα από 25 γράμματα.
4) **Transform cases**. Για τον υπολογισμό των συχνοτήτων των λέξεων μπορεί να οριστεί ότι οι λέξεις του κειμένου είναι οι ίδιες ανεξάρτητα αν γράφονται με πεζά ή κεφαλαία, π.χ. system, SYSTEM, System στο κείμενο είναι η ίδια λέξη.

Στο παράδειγμά μας, έχουμε ένα σύνολο κριτικών ταινιών. Οι κριτικές αποθηκεύονται στο αρχείο κειμένου IMDB_movie_sample.xls. Περιλαμβάνονται 1000 κριτικές και κάθε κριτική χαρακτηρίζεται ως θετική ή αρνητική.

Μπορείτε να χρησιμοποιήσετε ανάλογο δείγμα δεδομένων από διάφορους ιστοτόπους. Παραθέτουμε παραδείγματα από τον ιστότοπο Kaggle.

"IMDB Dataset of 50K Movie Reviews" (Large Movie Review Dataset)

IMDB Dataset of 50K Movie Reviews | Kaggle

"The Movies Dataset" (Metadata on over 45,000 movies. 26 million ratings from over 270,000 users) **The Movies Dataset | Kaggle**

"9000+ Movies Dataset" (Movies Dataset sorted by its popularity for Recommender System) **9000+ Movies Dataset | Kaggle**

Παραθέτουμε δείγμα δεδομένων στον πίνακα 2.9

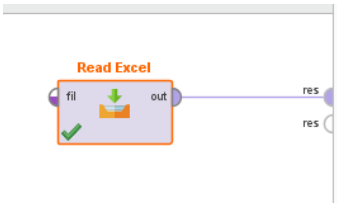
Πίνακας 2.9 Δείγμα κριτικών

Label	Text
Negative	Un-bleeping-believable! Meg Ryan doesn't even look her usual pert lovable self in this, which normally makes me forgive her shallow ticky acting schtick. Hard to believe she was the producer on this dog. Plus Kevin Kline: what kind of suicide trip has his career been on? Whoosh... Banzai!!! Finally this was directed by the guy who did Big Chill? Must be a replay of Jonestown - hollywood style. Wooofff!
Positive	This is a extremely well-made film. The acting, script and camera-work are all first-rate. The music is good, too, though it is mostly early in the film, when things are still relatively cheery. There are no really superstars in the cast, though several faces will be familiar. The entire cast does an excellent job with the script. But it is hard to watch, because there is no good end to a situation like the one presented. It is now fashionable to blame the British for setting Hindus and Muslims against each other, and then cruelly separating them into two countries. There is some merit in this view, but it's also true that no one forced Hindus and Muslims in the region to mistreat each other as they did around the time of partition. It seems more likely that the British simply saw the tensions between the religions and were clever enough to exploit them to their own ends. The result is that there is much cruelty and inhumanity in the situation and this is very unpleasant to remember and to see on the screen. But it is never painted as a black-and-white case. There is baseness and nobility on both sides, and also the hope for change in the younger generation. There is redemption of a sort, in the end, when Puro has to make a hard choice between a man who has ruined her life, but also truly loved her, and her family which has disowned her, then later come looking for her. But by that point, she has no option that is without great pain for her. This film carries the message that both Muslims and Hindus have their grave faults, and also that both can be dignified and caring people. The reality of partition makes that realisation all the more wrenching, since there can never be real reconciliation across the India/Pakistan border. In that sense, it is similar to "Mr & Mrs Iyer". In the end, we were glad to have seen the film, even though the resolution was heartbreaking. If the UK and US could deal with their own histories of racism with this kind of frankness, they would certainly be better off.
Negative	Every once in a long while a movie will come along that will be so awful that I feel compelled to warn people. If I labor all my days and I can save but one soul from watching this movie, how great will be my joy. Where to begin my discussion of pain. For starters, there was a musical montage every five minutes. There was no character development. Every character was a stereotype. We had swearing guy, fat guy who eats donuts, goofy foreign guy, etc. The script felt as if it were being written as the movie was being shot. The production value was so incredibly low that it felt like I was watching a junior high video presentation. Have the directors, producers, etc. ever even seen a movie before? Halestorm is getting worse and worse with every new entry. The concept for this movie sounded so funny. How could you go wrong with Gary Coleman and a handful of somewhat legitimate actors. But trust me when I say this, things went wrong, VERY WRONG.
Positive	Name just says it all. I watched this movie with my dad when it came out and having served in Korea he had great admiration for the man. The disappointing thing about this film is that it only concentrate on a short period of the man's life - interestingly enough the man's entire life would have made such an epic bio-pic that it is staggering to imagine the cost for production. Some posters elude to the flawed characteristics about the man, which are cheap shots. The theme of the movie "Duty, Honor, Country" are not just mere words blathered from the lips of a high-brassed officer - it is the deep declaration of one man's total devotion to his country. Ironically Peck being the liberal that he was garnered a better understanding of the man. He does a great job showing the fearless general tempered with the humane side of the man.

Ακολουθεί μια σειρά από βήματα για την επεξεργασία του δείγματος.

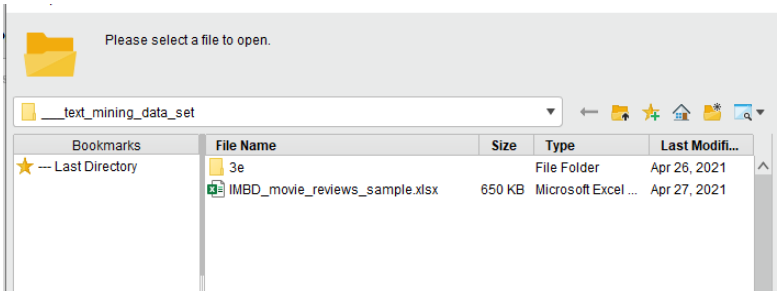
Βήμα 1

Αρχικά θα πρέπει να φορτώσουμε τα δεδομένα στο πρόγραμμα. Στην εικόνα 2.46 βλέπουμε το μοντέλο μας το οποίο περιλαμβάνει μόνο τον τελεστή Read Excel που θα χρησιμοποιήσουμε για να διαβάσουμε το αρχείο με τις κριτικές.



Εικόνα 2.46 Ανάγνωση αρχείου κειμένου

Στην εικόνα 2.47 βλέπουμε πως επιλέγουμε το αρχείο των κριτικών. Στο παράδειγμά μας το αρχείο βρίσκεται στην περιοχή _text_mining_data_set, στον υπολογιστή μας, και έχει όνομα IMDB_movie_sample.xls.



Εικόνα 2.47 Επιλογή αρχείου κριτικών

Στην εικόνα 2.48 βλέπουμε το αποτέλεσμα της εκτέλεσης του μοντέλου.

Row No.	label	text
1	negative	Un-bleeping-believable! Meg Ryan doesn't even look her usual pert lovable s...
2	positive	This is a extremely well-made film. The acting, script and camera-work are all...
3	negative	Every once in a long while a movie will come along that will be so awful that I f...
4	positive	Name just says it all. I watched this movie with my dad when it came out and ...
5	negative	This movie succeeds at being one of the most unique movies you've seen. H...
6	negative	From the start, you know how this movie will end. It's so full of clichÃ©s your ty...
7	negative	There were a lot of truly great horror movies produced in the seventies - but th...
8	negative	I was fortunate enough to meet George Pal (and still have my DS:TMOB poste...
9	negative	This film is the freshman effort of Stephanie Beaton and her new production c...
10	negative	Greg Davis and Bryan Daly take some crazed statements by a terrorists, add ...
11	positive	This movie is the first of Miikes triad society trilogy, and the trilogy kicks of to a ...
12	positive	I have screened this movie several times here at college, and every time I sho...
13	positive	An unusual film from Ringo Lam and one that's strangely under-appreciated. ...
14	positive	This is a wonderfully written and well acted psychological drama. It is not reall...
15	positive	If you find yourself in need of an escape, something that will hold your attentio...

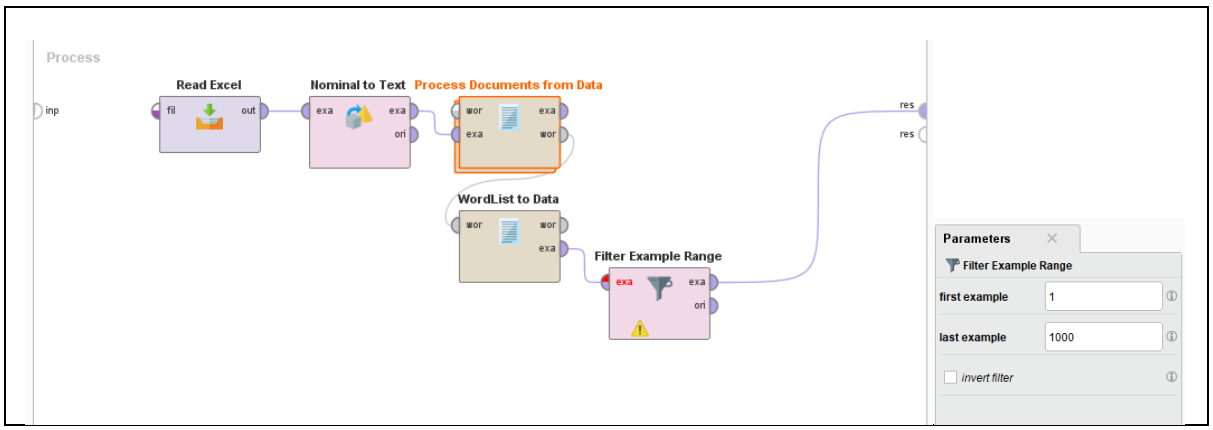
ExampleSet (1,000 examples, 0 special attributes, 2 regular attributes)

Εικόνα 2.48 Οι κριτικές είναι χαρακτηρισμένες ως θετικές ή αρνητικές

Βήμα 2

Κατασκευάζουμε το μοντέλο μας (process) που θα διεκπεραιώσει τη βασική επεξεργασία (Εικόνα 2.49). Το μοντέλο μας χρησιμοποιεί πέντε τελεστές (operators):

1) Ο τελεστής **Read Excel** χρησιμοποιείται για να διαβάσουμε το αρχείο με τις κριτικές
2) Ο τελεστής **Nominal to Text** χρησιμοποιείται για να καθορίσουμε ότι το εργαλείο RapidMiner πρέπει να διαχειριστεί τα δεδομένα (τις κριτικές) σαν κείμενο.
3) Ο τελεστής **Process Documents for data** χρησιμοποιείται για να διεκπεραιώσει τη βασική επεξεργασία. Για την ακρίβεια θα πρέπει να ορίσουμε τις ενέργειες που θέλουμε να πραγματοποιεί. Επομένως, θα χρειαστεί να κατασκευάσουμε στο Βήμα 3 ένα νέο (υπο)μοντέλο που θα υλοποιεί αυτά που θέλουμε να πραγματοποιεί ο τελεστής Process Documents for data.
4) Ο τελεστής **WordList to Data** δημιουργεί ένα σύνολο δεδομένων από μια λίστα λέξεων. Το σύνολο δεδομένων περιέχει μια γραμμή για κάθε λέξη και κάποια χαρακτηριστικά (attributes), π.χ., τον αριθμό των εγγράφων (documents) στα οποία εμφανίστηκε.
5) Ο τελεστής **Filter Example Range** επιλέγει ποια παραδείγματα–examples (δηλαδή γραμμές) ενός ExampleSet θα πρέπει να διατηρηθούν και ποια παραδείγματα θα πρέπει να αφαιρεθούν.

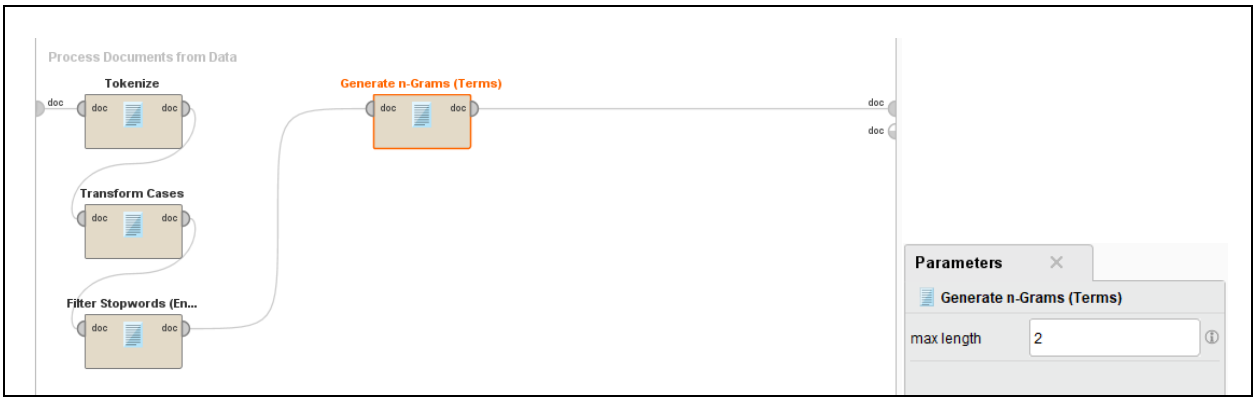


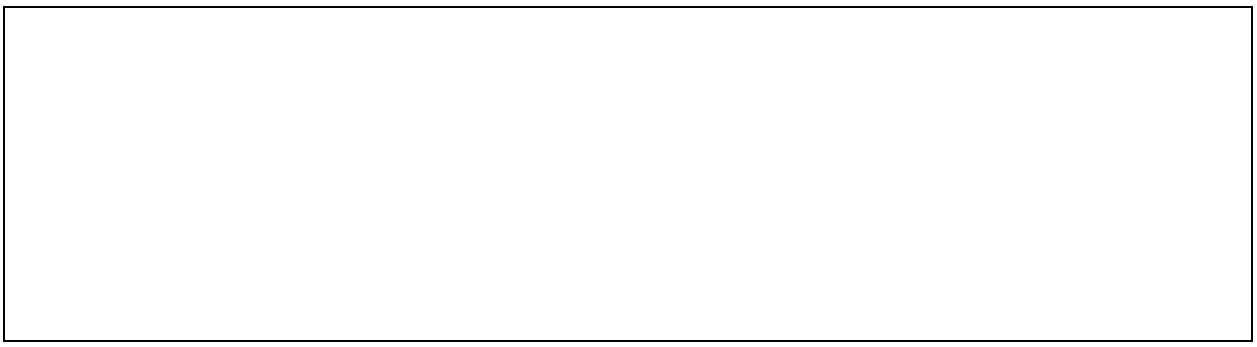
Εικόνα 2.49 Η κεντρική διαδικασία (process) περιλαμβάνει τελεστή "Process Documents for data" συνδεδεμένο με υποδιαδικασία (subprocess)

Βήμα 3

Κατασκευάζουμε ένα υπομοντέλο που υλοποιεί τις λειτουργίες του τελεστή Process Documents for data (Εικόνα 2.49). Το μοντέλο μας χρησιμοποιεί τέσσερις τελεστές (operators) (Εικόνα 2.50) που αντιστοιχούν στις παρακάτω λειτουργίες:

1) Το πρώτο βήμα στην προεπεξεργασία του κειμένου των κριτικών είναι το tokenization δηλαδή ο διαχωρισμός του κειμένου σε λέξεις (tokens). Για τον σκοπό αυτό χρησιμοποιείται ο τελεστής Tokenize. Προσοχή! Συνδέστε την είσοδο doc στα αριστερά με την αριστερή πόρτα (left port) doc του τελεστή Tokenize..
2) Στη συνέχεια μεταγράφετε όλες τις λέξεις σε πεζά. Με αυτόν τον τρόπο το εργαλείο RapidMiner υπολογιστής δεν θα βλέπει System, SYSTEM, system ως διαφορετικές λέξεις αλλά ως την ίδια λέξη system. Θα χρησιμοποιήσετε τον τελεστή Transform Cases.
3) Στο επόμενο βήμα θα αφαιρεθούν οι μη σημαντικές λέξεις (stopwords), π.χ.., "the", "and", "of", "is" κ.λπ. Οι λέξεις αυτές δεν βοηθούν τον αλγόριθμό μας να διαχωρίσει θετικές από αρνητικές κριτικές. Χρησιμοποιούμε τον τελεστή Filter Stopwords (English). Αν είχαμε κριτικές στα ελληνικά θα έπρεπε να «φορτώσουμε» στο εργαλείο αρχείο με ελληνικές μη σημαντικές λέξεις (stopwords).. Το εργαλείο δε διαθέτει τέτοιο αρχείο αλλά επιτρέπει να χρησιμοποιήσουμε δικό μας.
4) Τέλος, θα δημιουργήσουμε N-grams δηλαδή ακολουθίες n στοιχείων από τις λέξεις του κειμένου. Για παράδειγμα, .τα unigrams (1-grams) της πρότασης "Athens is the capital" είναι Athens, is, the, capital. Τα bigrams (2-grams) are " Athens is ", "is the", "the capital". Αν έχουμε ήδη χρησιμοποιήσει αρχείο stopwords τότε τα 1-grams για την πρόταση είναι Athens, capital και τα 2-grams είναι " Athens capital". Θα πρέπει να χρησιμοποιήσουμε τον τελεστή Generate n-Grams (Terms).





Εικόνα 2.50 Ορισμός subprocess για τις λειτουργίες του τελεστή Process Documents for data

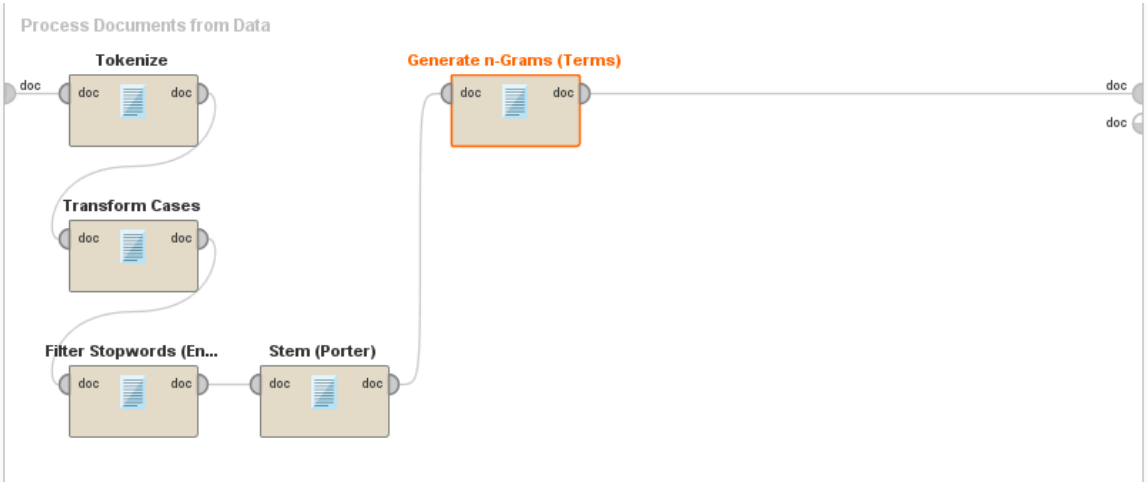
Στην εικόνα 2.51 βλέπουμε το αποτέλεσμα της εκτέλεσης του μοντέλου.

Row No.	word	in documents	total
1	ability	2	2
2	able	12	13
3	abound	2	2
4	absolute	2	2
5	absolutely	8	9
6	abuse	2	3
7	accent	3	3
8	accidentally	2	2
9	accomplished	2	2
10	accuracy	2	2
11	accurate	3	4
12	act	7	9
13	act_i	2	2
14	acted	6	6

Εικόνα 2.51 Συνολική συχνότητα λέξεων και συχνότητα ανά κριτική

Χρήση εξαγωγής θέματος λέξεων (stemming)

Στο Βήμα 3 μπορούμε να χρησιμοποιήσουμε «εύρεση ρίζας λέξης»,stemming, δηλαδή αντί των λέξεων να χρησιμοποιούμε το θέμα (ρίζα, stem) τους (Εικόνα 2.52). Οι μηχανές αναζήτησης χρησιμοποιούν stemming για την ευρετηρίαση των λέξεων. Δηλαδή, αντί να αποθηκεύει όλες τις μορφές μιας λέξης, μια μηχανή αναζήτησης μπορεί να αποθηκεύσει μόνο τα θέματα.



Εικόνα 2.52 Χρήση τελεστή stem για την εύρεση του θέματος (stem) των λέξεων

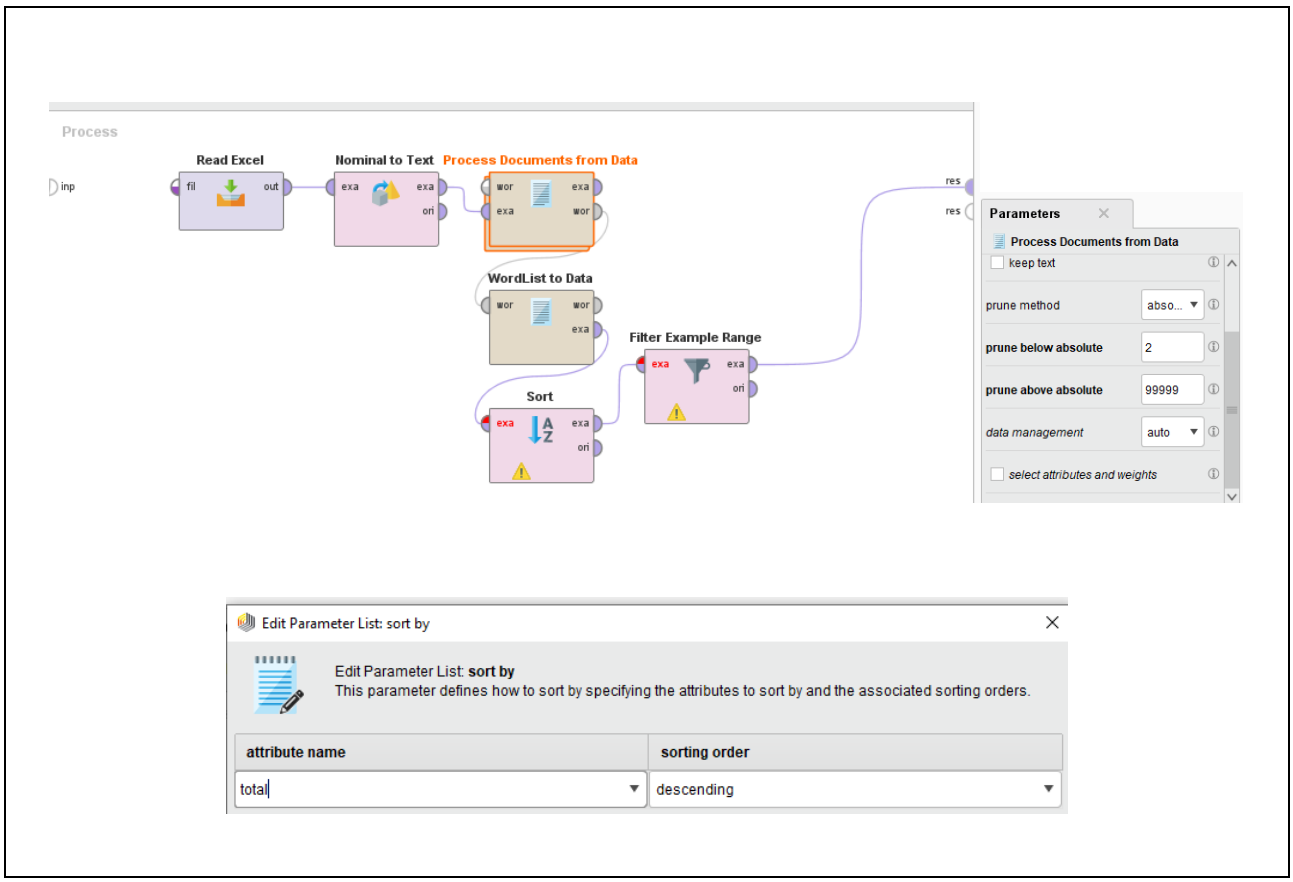
Στην εικόνα 2.53 βλέπουμε το αποτέλεσμα της εκτέλεσης του μοντέλου

Row No.	word	in documents	total
1	abil	3	3
2	abl	12	13
3	abound	2	2
4	absolut	10	11
5	absurd	2	2
6	abus	3	6
7	accent	3	3
8	accept	5	5
9	accident	2	2
10	accompani	2	2
11	accomplish	3	3
12	account	3	3
13	accur	3	4
14	accuraci	2	2

Εικόνα 2.53 Συχνότητες θεμάτων (stem)

Κατασκευή wordcloud

Στην εικόνα 2.54 βλέπουμε το μοντέλο που θα χρησιμοποιήσουμε. Στην παραμετροποίηση του τελεστή Process Documents from Data καταργούμε τις επιλογές παραμέτρων ‘create word vector’ και ‘add meta information’, θέτουμε prune method ως ‘absolute’, και ορίζουμε τιμές 2 για ‘prune below absolute’ και 99999 για ‘prune above absolute’. Με τον τρόπο αυτό αγνοούμε λέξεις που υπάρχουν μόνο σε ένα έγγραφο. Ο τελεστής Sort ταξινομεί τα δεδομένα εισόδου του κατά αύξουσα (ascending) ή φθίνουσα (descending) τάξη σύμφωνα με διάφορα χαρακτηριστικά (attributes).



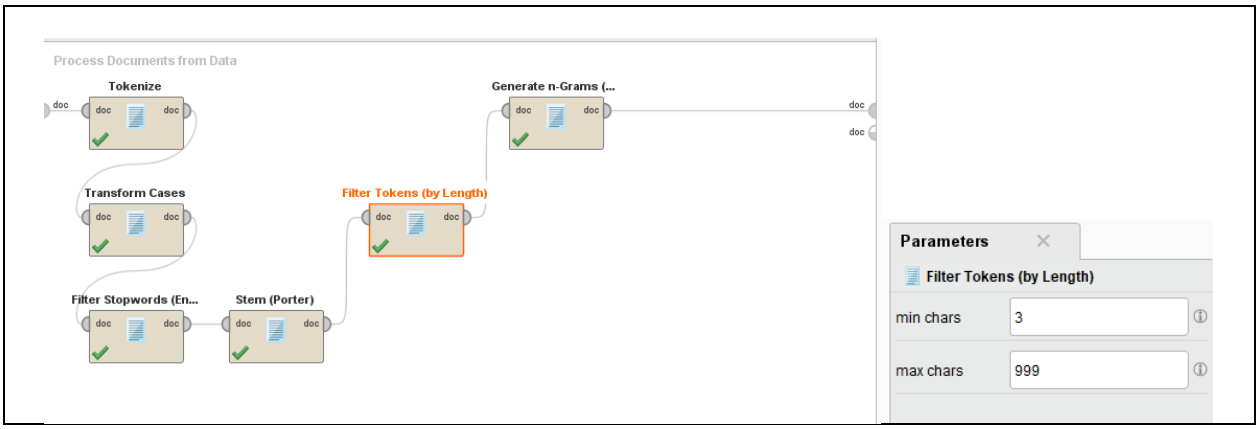
Εικόνα 2.54 Μοντέλο που θα χρησιμοποιήσουμε και παραμετροποίηση του τελεστή Process Documents from Data

Στο γενικό μοντέλο μας προσθέτουμε τελεστές Wordlist to Data, Sort (παράμετροι decreasing και attribute name ίσο με total) και Filter Example Range (παράμετροι first=1 και last=20, για να επιλέξουμε μόνο τις 20 πιο συχνές λέξεις). Στην Εικόνα 2.55 βλέπουμε το αποτέλεσμα εκτέλεσης του μοντέλου μας.

Row No.	word	in documents	total
1	film	120	425
2	movi	119	369
3	good	82	126
4	time	78	125
5	get	74	117
6	stori	62	114
7	make	76	108
8	see	73	105
9	scene	56	99
10	watch	69	99
11	look	67	93
12	peopl	63	86
13	end	55	81
14	charact	46	79

Εικόνα 2.55 Συχότητες θεμάτων λέξεων ανά τεκμήριο (document) και συνολικά

Στη συνέχεια προσθέτουμε στο μοντέλο μας τον τελεστή Filter Tokens (by Length). (Εικόνα 2.56). Στις παραμέτρους θέτουμε min chars (ελάχιστος αριθμός γραμμάτων λέξης) το 3 και maximum το 999.

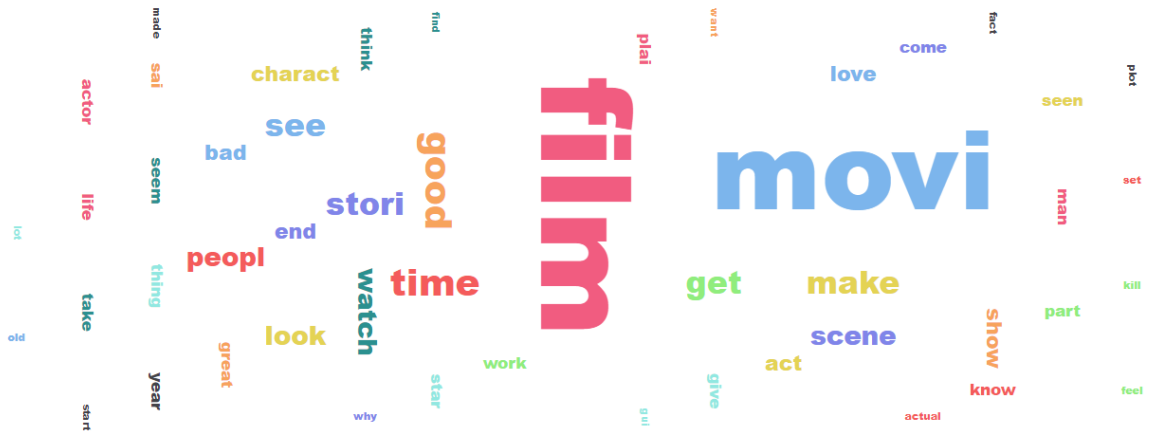


Εικόνα 2.56 Προσθήκη στο μοντέλο του τελεστή τελεστή Filter Tokens (by Length και παραμετροποίηση

Εκτελούμε το νέο μοντέλο και επιλέγουμε Visualizations. Στο plot type επιλέγουμε Word cloud και θέτουμε ως value την επιλογή word και ως weight την επιλογή total (εικόνα 2.57)

Εικόνα 2.57 Χρήση τύπου σχεδίασης(plot type)Word cloud

Στην εικόνα 2.58 βλέπουμε το γράφημα για τις 50 πρώτες λέξεις.

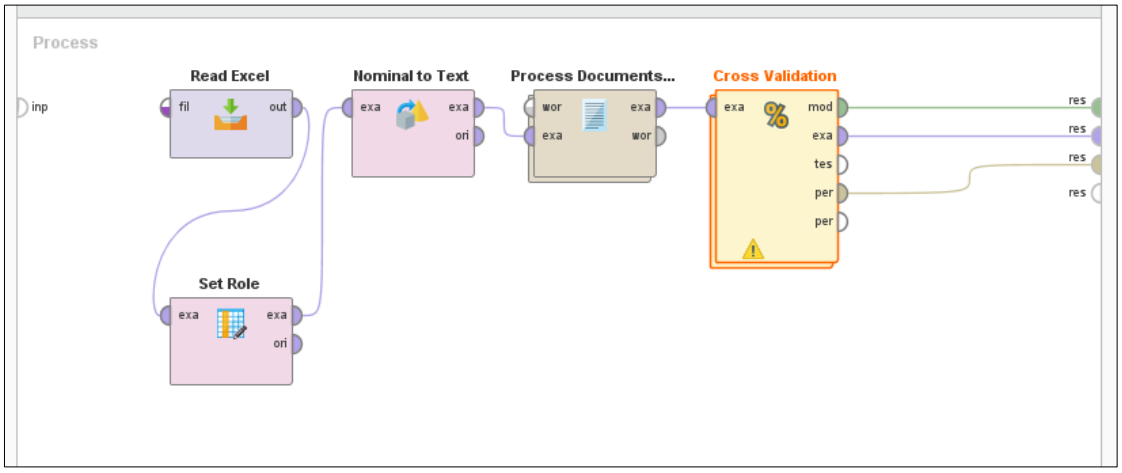


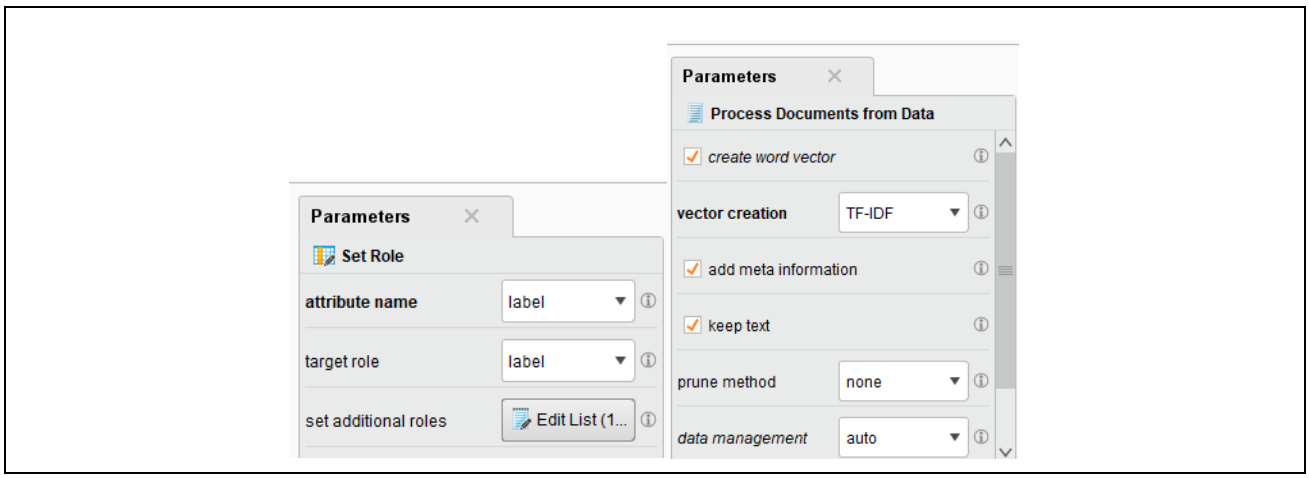
Εικόνα 2.58 Χρήση τύπου σχεδίασης(plot type)Word cloud για τις 50 πιο σημαντικές λέξεις

2.15.2 Ανάλυση συναισθήματος (Sentiment analysis)

Στην εικόνα 2.59 βλέπουμε το μοντέλο μας που θα προβλέπει αν μία νέα κριτική είναι θετική ή αρνητική. Χρησιμοποιούμε πέντε τελεστές:

1) Ο τελεστής Read Excel χρησιμοποιείται για να διαβάζουμε το αρχείο κειμένου. Το αρχείο έχει την μορφή που παραθέσαμε στον πίνακα 2.9.
2) Με τον τελεστή Set Role ορίζουμε ότι το χαρακτηριστικό label ότι είναι τύπου label.
3) Με τον τελεστή Nominal to text καθορίζουμε ότι θα διαχειριστούμε τα δεδομένα μας ως κείμενο.
4) Με τον τελεστή Process Documents from Data θα καθορίσουμε σε (sub) process πως ακριβώς θα γίνει η επεξεργασία του κειμένου κριτικών (Εικόνα 2.60).
5) Θα χρησιμοποιήσουμε τελεστή Cross Validation για να ορίσουμε (sub) process (Εικόνα 2.61). Πιο συγκεκριμένα, θα ορίσουμε ένα process για τη δημιουργία συνόλου εκπαίδευσης (training dataset) και συνόλου βαθμολόγησης (scoring dataset). Με τα σύνολα αυτά θα εκπαιδεύσουμε τον τελεστή για τη πρόβλεψη-πρόγνωση (prediction) και θα μετρήσουμε την επίδοσή του.





Εικόνα 2.59 Μοντέλο πρόβλεψης αν μία νέα κριτική είναι θετική ή αρνητική. Χρήση τελεστή Process Documents from Data και τελεστή Cross Validation

Στην εικόνα 2.60 βλέπουμε και την παραμετροποίηση για τον τελεστή Process Documents from Data. Επιλέξαμε create word vectors, keep text και add meta information. Στη συνέχεια επεξηγούμε τις επιλογές αυτές.

1) **create word vector**: Αν επιλεγεί τότε οι λέξεις του εγγράφου θα χρησιμοποιηθούν για τη δημιουργία ενός διανύσματος που αντιπροσωπεύει αριθμητικά το έγγραφο (“the tokens of a document will be used to generate a vector numerically representing the document”).
2) **add meta information**: Αν επιλεγεί τότε οι διαθέσιμες μετα-πληροφορίες του κειμένου όπως όνομα αρχείου, ημερομηνία προστίθενται ως χαρακτηριστικό (“available meta information of the text like filename, date is added as attribute”).
3) **keep text**: Αν επιλεγεί, “the input text will be stored as a special String attribute with the role text”).(RapidMiner 9 Operator Reference Manual)

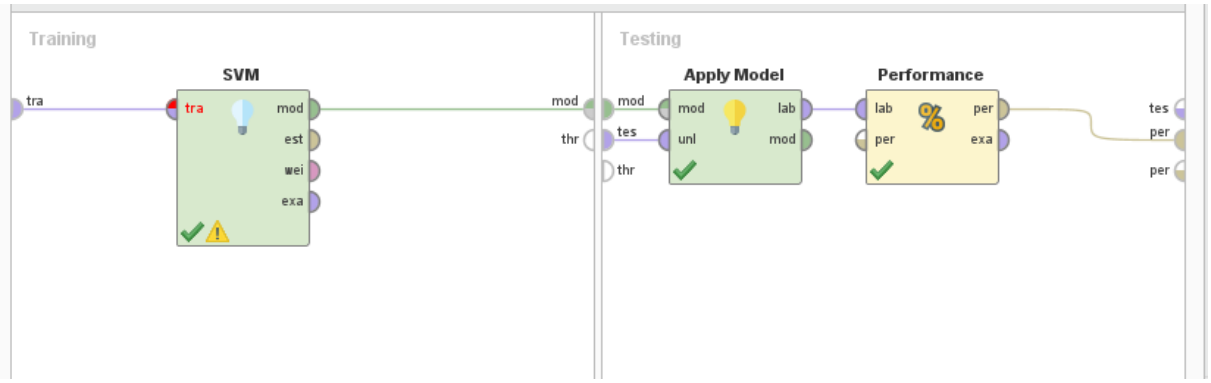
Στην εικόνα 2.59 βλέπουμε, επίσης, τον τελεστή Cross Validation. Ο τελεστής χρησιμοποιείται για να ορίσουμε το (sub) process για τη δημιουργία συνόλου εκπαίδευσης και συνόλου βαθμολόγησης, θα εκπαιδεύσουμε τελεστή για τη πρόγνωση (prediction) και θα μετρήσουμε την επίδοσή του.

Εικόνα 2.60 Ο τελεστής Process Documents from Data συνδέεται με το process για την επεξεργασία κειμένου

Ο τελεστής Cross Validation χρησιμοποιείται για την εκτίμηση της ακρίβειας ενός μοντέλου το οποίο βασίζεται σε έναν συγκεκριμένο τελεστή που «μαθαίνει», π.χ., Support Vector Machine. Στην εικόνα 2.61

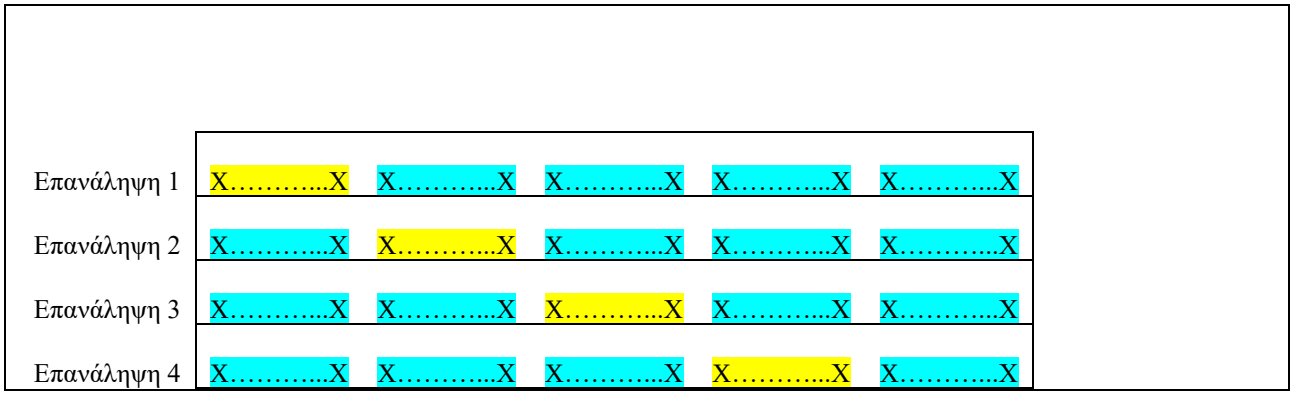
βλέπουμε ότι ο τελεστής Cross Validation έχει δύο εμφωλευμένες (φωλιασμένες) υποδιαδικασίες: μια υποδιαδικασία εκπαίδευσης (Training subprocess) και μια υποδιαδικασία δοκιμής (Testing subprocess). Η υποδιαδικασία Training έχει ως είσοδο το σύνολο εκπαίδευσης και χρησιμοποιείται για την εκπαίδευση του μοντέλου (στο παράδειγμα χρησιμοποιούμε Support Vector Machine). Στη συνέχεια, το εκπαιδευμένο μοντέλο εφαρμόζεται στην υποδιαδικασία Testing. Η απόδοση του μοντέλου μετράται κατά τη φάση της δοκιμής. Το ExampleSet εισόδου χωρίζεται σε k υποσύνολα ίσου μεγέθους. Από τα k υποσύνολα, ένα υποσύνολο διατηρείται ως σύνολο δεδομένων δοκιμής και τα υπόλοιπα (k – 1) υποσύνολα χρησιμοποιούνται ως σύνολο δεδομένων εκπαίδευσης. Στη συνέχεια, η διαδικασία διασταυρούμενης επικύρωσης (cross validation process) επαναλαμβάνεται k φορές, με καθένα από τα k υποσύνολα να χρησιμοποιείται ακριβώς μία φορά ως δεδομένα δοκιμής. Τα k αποτελέσματα από τις k επαναλήψεις συνδυάζονται για να παραχθεί η τελική εκτίμηση. Η τιμή k ρυθμίζεται στο προϊόν RapidMiner με ορισμό της παραμέτρου μπορεί να ρυθμιστεί χρησιμοποιώντας την παράμετρο folds (Εικόνα 2.62 και Εικόνα 2.63).

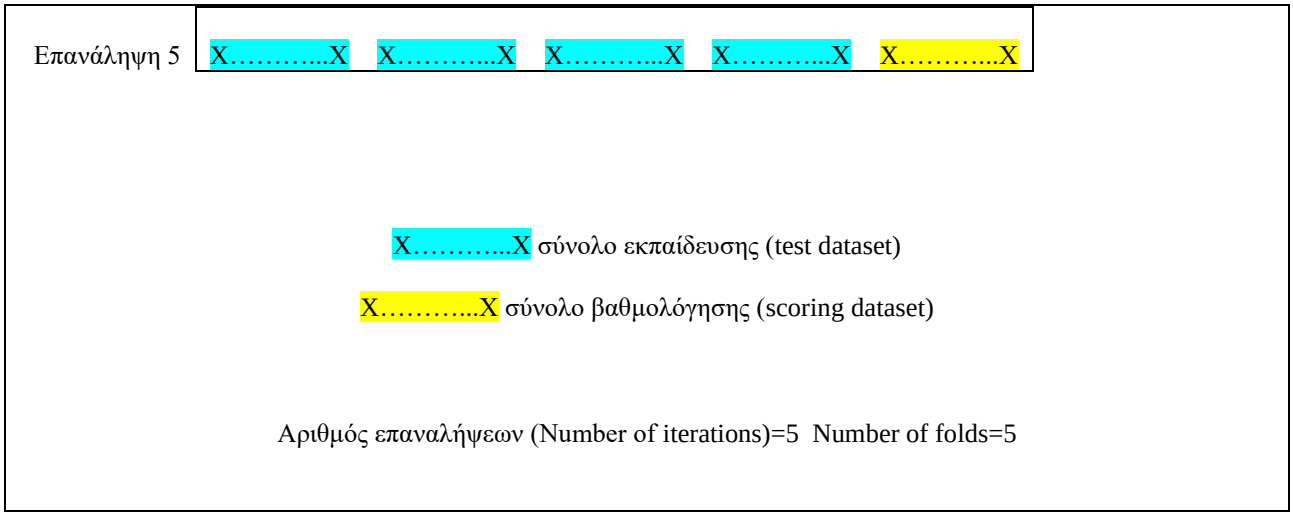
Είναι σημαντικό να επισημάνουμε ότι: «Η αξιολόγηση της απόδοσης ενός μοντέλου σε ανεξάρτητα σύνολα δοκιμών αποδίδει μια καλή εκτίμηση της απόδοσης του όταν θα εφαρμοστεί σε νέα σύνολα δεδομένων. Δείχνει επίσης εάν συμβαίνει «υπερβολική προσαρμογή. Το φαινόμενο της υπερβολικής προσαρμογής σημαίνει ότι το μοντέλο αντιπροσωπεύει πολύ καλά τα δεδομένα δοκιμής, αλλά δεν γενικεύεται για νέα δεδομένα. Έτσι, η απόδοση μπορεί να είναι πολύ χειρότερη στα δεδομένα δοκιμών» (“The evaluation of the performance of a model on independent test sets yields a good estimation of the performance on unseen data sets. It also shows if ‘overfitting’ occurs. This means that the model represents the testing data very well, but it does not generalize well for new data. Thus, the performance can be much worse on test data”). (RapidMiner 9 Operator Reference Manual, 2022)



Εικόνα 2.61 Ο τελεστής Cross Validation συνδέεται με το process εκπαίδευσης και υπολογισμού επίδοσης.

Στο αριστερό μέρος της Εικόνας 2.62 φαίνεται η εκπαίδευση. Χρησιμοποιείται τελεστής Support Vector Machine.Στο δεξί μέρος χρησιμοποιούμε τελεστή Apply Model για να εκτελέσει το μοντέλο μας και τελεστή Performance για να υπολογίσει την επίδοση του μοντέλου μας.



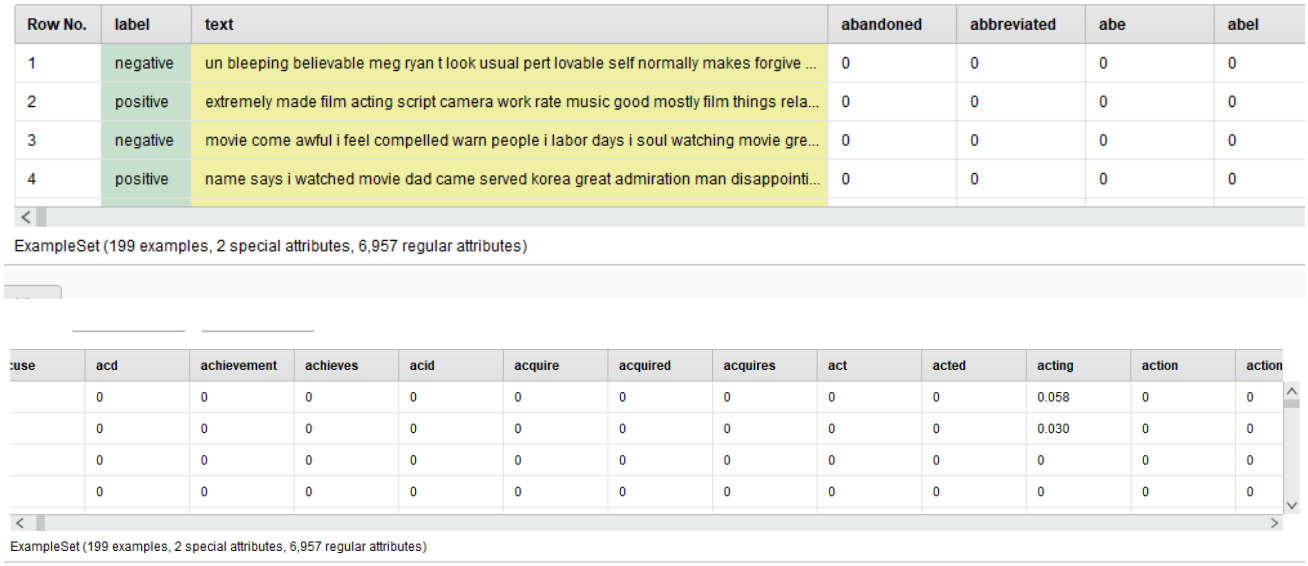


Εικόνα 2.62 Χωρισμός σε σύνολο δεδομένων δοκιμής και σύνολο δεδομένων εκπαίδευσης και επαναλήψεις διαδικασίας διασταυρούμενης επικύρωσης (cross validation process).

Με την εκτέλεση του μοντέλου μπορούμε να δούμε πολλά στοιχεία. Στην εικόνα 2.63 βλέπουμε απόσπασμα από το ExampleSet. Στο παράδειγμά μας ο αλγόριθμος Support Vector Machine εκπαιδεύεται με 800 κριτικές (examples) και γίνεται πρόβλεψη για 199 κριτικές. Στην εικόνα 2.63 βλέπουμε το κείμενο 4 κριτικών, τον χαρακτηρισμό (positive/negative) και τη βαρύτητα των λέξεων. Στο παράδειγμα της εικόνας οι πρώτες λέξεις δεν έχουν βαρύτητα. Στη συνέχεια, στην ίδια εικόνα, βλέπουμε ότι η λέξη acting έχει βαρύτητα στις 2 πρώτες κριτικές και μάλιστα μεγαλύτερη στην πρώτη κριτική.

Παρατηρήστε ότι κάθε κριτική είναι ένα διάνυσμα λέξεων. Η τιμή 0 στο διάνυσμα σημαίνει ότι η λέξη δεν υπάρχει στην κριτική. Για παράδειγμα η λέξη abandoned δεν υπάρχει στις 4 πρώτες κριτικές. Η ύπαρξη τιμής διάφορης του μηδενός στο διάνυσμα σημαίνει ότι η λέξη υπάρχει στην κριτική και έχει κάποια βαρύτητα.

Στην εικόνα 2.63 βλέπετε ακόμη ότι η λέξη acting έχει για το σύνολο του dataset αρνητικό βάρος. Βέβαια η πρώτη κριτική είναι αρνητική γεγονός που σημαίνει ότι η κριτική αυτή πρέπει να περιλαμβάνει κυρίως λέξεις με σημαντική βαρύτητα στην κριτική και αρνητικό βάρος. Αντιθέτως, η δεύτερη κριτική πρέπει να περιλαμβάνει κυρίως σημαντικές λέξεις για την κριτική που έχουν θετικό βάρος.



Row No.	label	text	abandoned	abbreviated	abe	abel
1	negative	un bleeping believable meg ryan t look usual pert lovable self normally makes forgive ...	0	0	0	0
2	positive	extremely made film acting script camera work rate music good mostly film things rela...	0	0	0	0
3	negative	movie come awful i feel compelled warn people i labor days i soul watching movie gre...	0	0	0	0
4	positive	name says i watched movie dad came served korea great admiration man disappointi...	0	0	0	0

ExampleSet (199 examples, 2 special attributes, 6,957 regular attributes)

:use	acd	achievement	achieves	acid	acquire	acquired	acquires	act	acted	acting	action	action
	0	0	0	0	0	0	0	0	0	0.058	0	0
	0	0	0	0	0	0	0	0	0	0.030	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0

ExampleSet (199 examples, 2 special attributes, 6,957 regular attributes)

Εικόνα 2.63 Κείμενο 4 κριτικών, με τον χαρακτηρισμό (positive/negative) και τη βαρύτητα των λέξεων

Στην εικόνα 2.64 βλέπουμε το Kernel Model (SVM) και το βάρος κάποιων όρων.

Kernel Model

Total number of Support Vectors: 199

Bias (offset): -0.089

w[abandoned] = 0.002

w[abbreviated] = 0.001

w[abe] = 0.001

w[abel] = 0.002

w[abiding] = -0.002

w[abilities] = 0.002

w[ability] = -0.002

w[able] = 0.002

w[abo] = -0.002

Example Set (Process Documents from Data)

```
w[accurate] = -0.000
w[accuse] = 0.001
w[acd] = 0.002
w[achievement] = -0.001
w[achieves] = 0.001
w[acid] = -0.001
w[acquire] = -0.001
w[acquired] = -0.001
w[acquires] = -0.001
w[act] = -0.003
w[acted] = 0.001
w[acting] = -0.003
w[action] = -0.001
w[actions] = -0.002
w[actor] = -0.002
```

Εικόνα 2.64 Kernel Model (SVM) και το βάρος κάποιων όρων

Αν επιλέξουμε Weight Table μπορούμε να δούμε τους όρους (λέξεις) με μέγιστο αρνητικό βάρος (Words with the highest negative weight) και τις λέξεις με μέγιστο θετικό βάρος (Words with the highest positive weight) (Εικόνα 2.65). Όπως προαναφέραμε, η ύπαρξη πολλών όρων με μεγάλο αρνητικό βάρος σε μία κριτική μπορούν να παίξουν σημαντικό ρόλο στο να χαρακτηριστεί η κριτική αρνητική. Επίσης, η ύπαρξη πολλών όρων με μεγάλο θετικό βάρος σε μία κριτική μπορούν να παίξουν σημαντικό ρόλο στο να χαρακτηριστεί η κριτική θετική.

Weight Table

Words with the highest positive weight

Attribute	Weight ↓
great	0.006
fantastic	0.005
ways	0.004
life	0.004
happy	0.004
woman	0.004

Words with the highest negative weight

Attribute	Weight ↑
d	-0.005
awful	-0.005
worst	-0.005
part	-0.005
rest	-0.005
t	-0.005

Εικόνα 2.65 Λέξεις με μέγιστο θετικό βάρος

Τέλος, Στην εικόνα 2.66 βλέπουμε το Performance Vector (Performance). Η ακρίβεια του μοντέλου στην πρόγνωση αρνητικών κριτικών είναι 64.06% και στην πρόγνωση θετικών 73.24%. Σχετικά καλή επίδοση για ένα τόσο μικρό δείγμα (1000 examples).

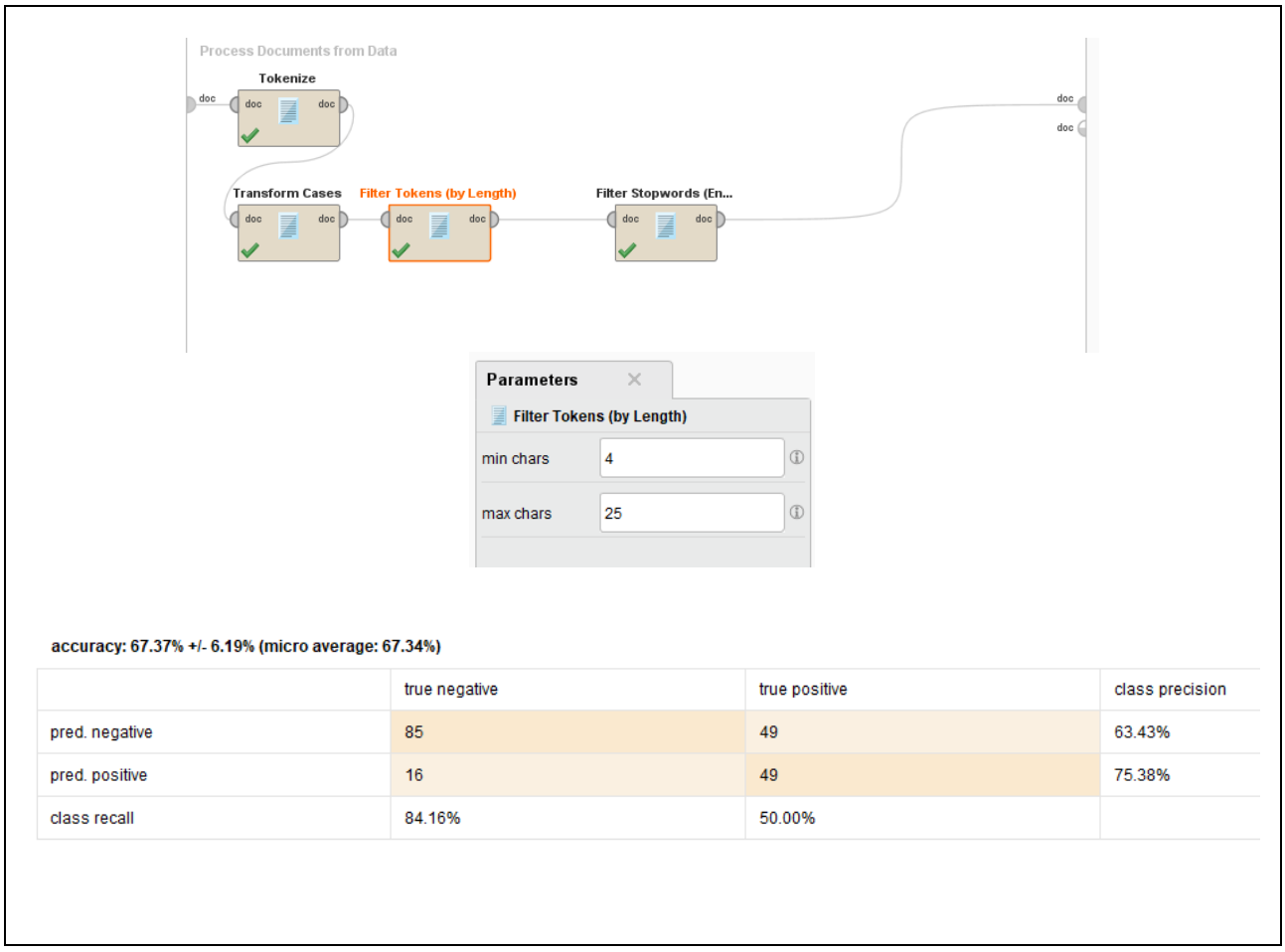
accuracy: 67.37% +/- 8.34% (micro average: 67.34%)

	true negative	true positive	class precision
pred. negative	82	46	64.06%
pred. positive	19	52	73.24%
class recall	81.19%	53.06%	

Εικόνα 2.66 Performance Vector

Διερεύνηση

Στην εικόνα 2.67 βλέπουμε το μοντέλο μας με την προσθήκη του τελεστή Filter Tokens (by Length). Η παραμετροποίηση μας ορίζει ότι το μέγεθος των όρων (λέξεων) που θα χρησιμοποιήσουμε στην ανάλυσή μας είναι γραμμένες το ελάχιστο από 4 γράμματα και το πολύ από 25 γράμματα. Βλέπουμε, επίσης, την απόδοση.



accuracy: 67.37% +/- 6.19% (micro average: 67.34%)

	true negative	true positive	class precision
pred. negative	85	49	63.43%
pred. positive	16	49	75.38%
class recall	84.16%	50.00%	

Εικόνα 2.67 Το μοντέλο μας με την προσθήκη του τελεστή Filter Tokens (by Length).

Στην εικόνα 2.68 βλέπουμε το διάνυσμα απόδοσης (Performance Vector) και τις λέξεις με ελάχιστο και λέξεις με μέγιστο βάρος.

Kernel Model

```
Total number of Support Vectors: 199
Bias (offset): -0.057

w[abandoned] = 0.002
w[abbreviated] = 0.001
w[abel] = 0.002
w[abiding] = -0.002
w[abilities] = 0.002
w[ability] = -0.002
w[able] = 0.002
w[abomination] = -0.001
```

Weight table

Words with the highest positive weight

Attribute	Weight ↓
great	0.006
fantastic	0.005
ways	0.005
life	0.005
woman	0.005
happy	0.005

Words with the highest negative weight

Attribute	Weight ↑
awful	-0.006
worst	-0.006
part	-0.005
rest	-0.005
okay	-0.005
watching	-0.005

Εικόνα 2.68 Διάνυσμα απόδοσης (Performance Vector, λέξεις με ελάχιστο και λέξεις με μέγιστο βάρος.

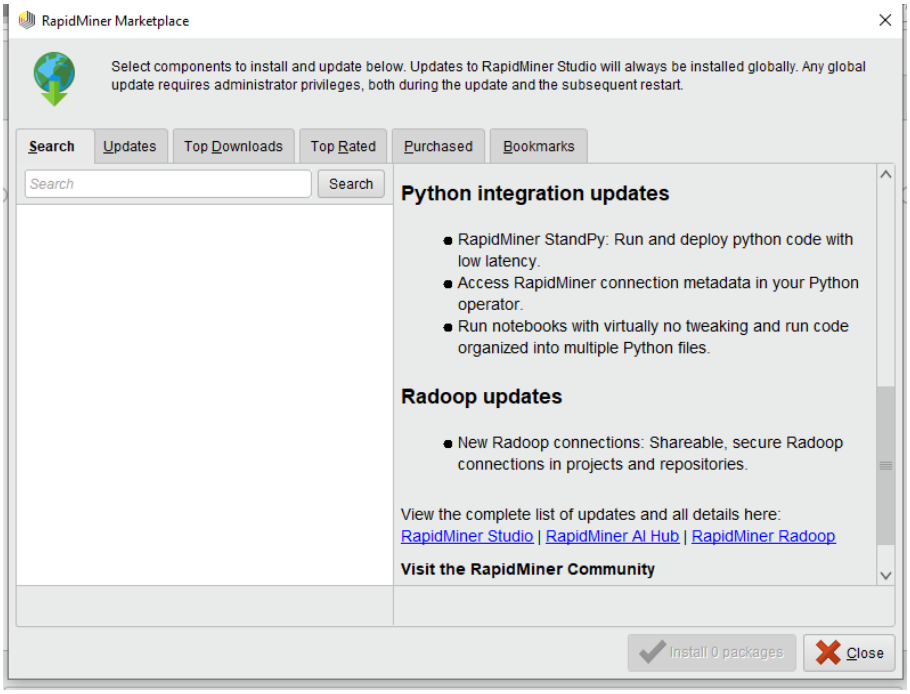
2.15.3 Παράδειγμα εξαγωγής όρων ευρετηρίου από ιστοσελίδες (Web)

Αρχικά θα πρέπει να επιλέξουμε από το βασικό μενού του εργαλείου RapidMiner την καρτέλα extensions. Μεταβαίνουμε στην αγορά RapidMiner Marketplace (εικόνα 2.69). Επιλέγουμε και εγκαθιστούμε την επέκταση Web Mining 9.7.0. (Εικόνα 2.70).Πλέον έχουμε στη διάθεσή μας μία σειρά από τελεστές με τους οποίους μπορούμε να «κατεβάσουμε» και να επεξεργαστούμε web pages. Στόχος μας να «διαβάσουμε» δύο σελίδες από τη Wikipedia και να προσδιορίσουμε τους όρους (n-gram=2) με τη μεγαλύτερη συχνότητα. Δείτε σχετικά και το αποτέλεσμα στην εικόνα 2.71.

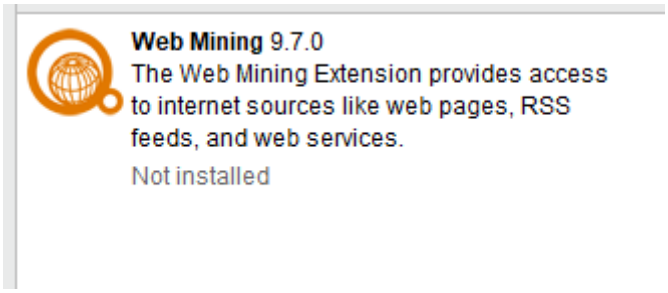
Οι σελίδες της Wikipedia στις οποίες εστιάζουμε αφορούν το λήμμα RapidMiner και το λήμμα Data mining:

https://en.wikipedia.org/wiki/RapidMiner

https://en.wikipedia.org/wiki/Data_mining



Εικόνα 2.69 Επεκτάσεις του RapidMiner. RapidMiner Marketplace



Εικόνα 2.70 Η επέκταση Web Mining 9.7.0

Word	Attribute Name	Total Occurences ↓	Docum...
data	data	258	2
mining	mining	158	2
data_mining	data_mining	140	2
learning	learning	44	2
machine	machine	39	2
machine_learning	machine_learning	34	2
rapidminer	rapidminer	33	2
software	software	30	2
knowledge	knowledge	25	2
analysis	analysis	23	2
patterns	patterns	23	1
discovery	discovery	19	2
privacy	privacy	19	2
research	research	18	2
knowledge_discovery	knowledge_discovery	17	2

Εικόνα 2.71 Αποτέλεσμα ανάγνωσης δύο σελίδων από τη Wikipedia και προσδιορισμός των όρων (n-gram=2) με τη μεγαλύτερη συχνότητα

Υλοποίηση

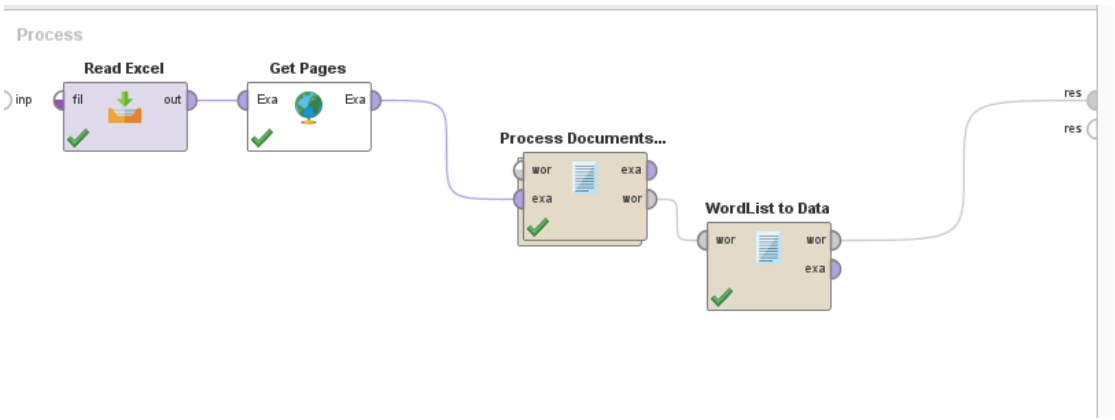
Γράφουμε στο αρχείο excel pages_1.xlsx τις παρακάτω τρεις γραμμές:

URLs

https://en.wikipedia.org/wiki/RapidMiner

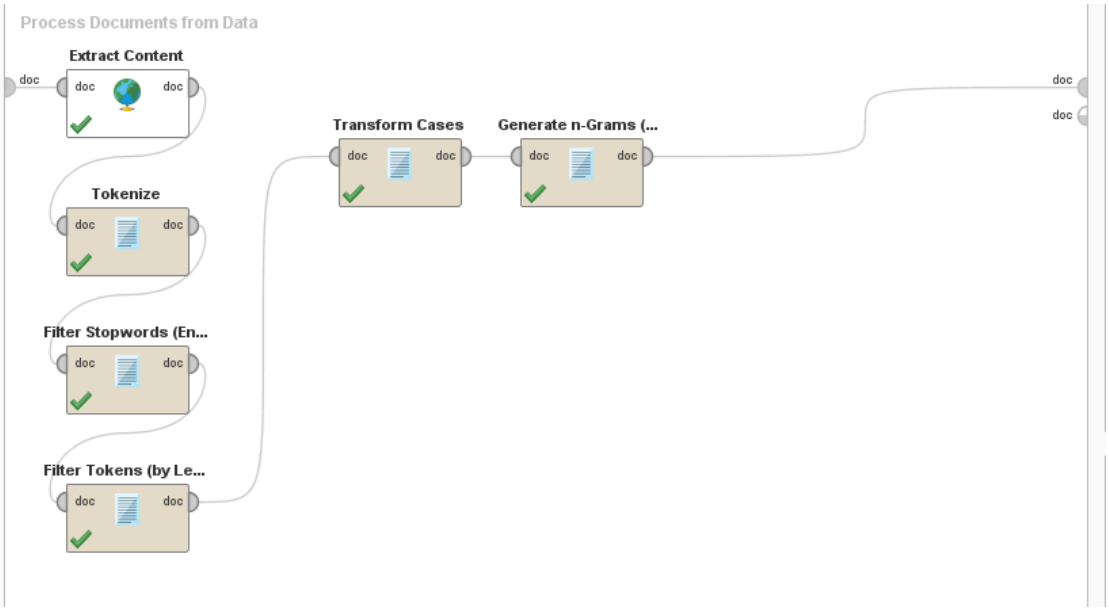
https://en.wikipedia.org/wiki/Data_mining

Στην πρώτη γραμμή η λέξη URLs έχει τη σημασία της γιατί θα δηλωθεί σαν link attribute του operator Get Pages. Μπορούμε να γράψουμε όποιο όνομα θέλουμε, π.χ., link, link_to_scan. Στις άλλες γραμμές του αρχείου γράφουμε όσες διευθύνσεις URLs θέλουμε. Στο παράδειγμα, γράφουμε δύο γραμμές. Το βασικό process φαίνεται στην εικόνα 2.72.



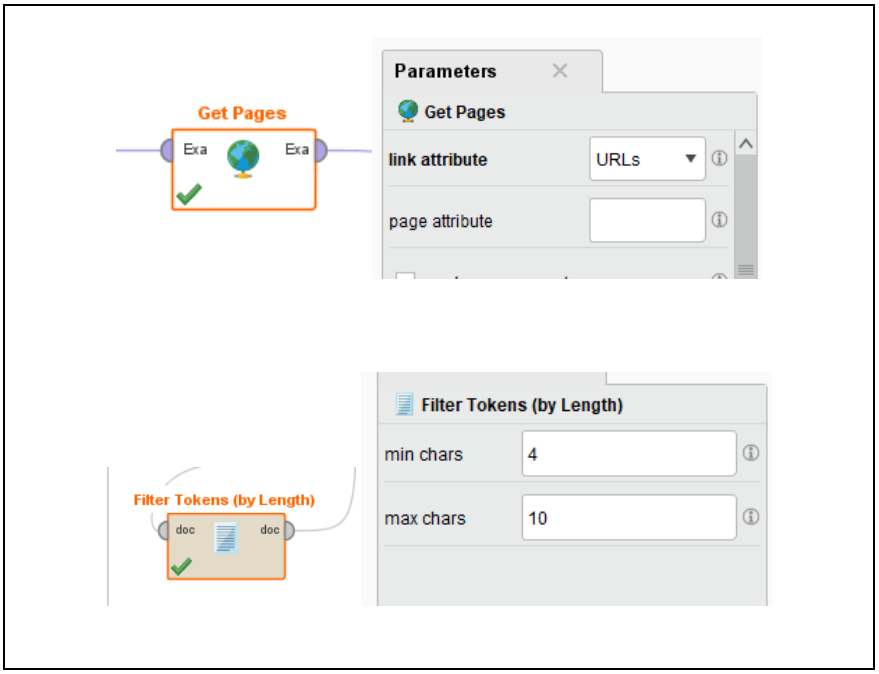
Εικόνα 2.72 Το βασικό process με τελεστή Get Pages

Ο τελεστής (operator) Process Document from Data ορίζεται ως sub-process στην εικόνα 2.73.



Εικόνα 2.73 Ορισμός του τελεστή Process Document from Data ως sub-process

Στην Εικόνα 2.73 βλέπουμε και τις παραμέτρους.



Εικόνα 2.74 Παραμετροποίηση τελεστών Get Pages, Filter Tokens (by Length)

Ενδεικτική βιβλιογραφία

Kotu, V., & Deshpande, B. (2014). Predictive analytics and data mining: concepts and practice with rapidminer. Morgan Kaufmann., ISBN: 978-0-12-801460-8

Hofmann, M., & Klinkenberg, R. (Eds.). (2016). RapidMiner: Data mining use cases and business analytics applications. CRC Press. Taylor & Francis Group

North, M.A. (2012). Data Mining for the Masses, Creative Commons Attribution 3.0 License. Ανακτήθηκε από https://docs.rapidminer.com/downloads/DataMiningForTheMasses.pdf

2.16 Η περίπτωση του προϊόντος MySQL. Συνδυασμός σχεσιακού ΣΔΒΔ και αποθήκης δεδομένων (Document Store)

Ένα σημαντικό χαρακτηριστικό του προϊόντος MySQL είναι ο συνδυασμός σχεσιακού ΣΔΒΔ και αποθήκης δεδομένων (Document Store). Το προϊόν παρέχει επιπλέον της σχεσιακής διαχείρισης τη δυνατότητα αποθήκευσης και διαχείρισης εγγράφων τύπου JSON. Τα έγγραφα JSON αποθηκεύονται σε συλλογές και είναι διαχειρίσιμα με τις λειτουργίες CRUD.

CRUD είναι μια ακροστιχίδα για τις λειτουργίες Create, Read, Update, και Delete. Οι λειτουργίες CRUD διεκπεραιώνουν τη βασική επεξεργασία δεδομένων (data manipulation) για μία βάση δεδομένων. Μπορούμε να δούμε την αντιστοιχία τους με δηλώσεις SQL: η λειτουργία create αντιστοιχεί με insert, η λειτουργία read με select, και οι λειτουργίες update και delete είναι αντίστοιχες των δηλώσεων update και delete της γλώσσας SQL.

Οι προγραμματιστές έχουν τη δυνατότητα να χρησιμοποιούν το συνδυασμό σχεσιακής και βάσης δεδομένων εγγράφων (Document Store) JSON. Οι βάσεις δεδομένων εγγράφων (Document Store) είναι μια από τις κατηγορίες των μη σχεσιακών (non-relational) βάσεων δεδομένων NoSQL και το προϊόν MySQL είναι ένα παράδειγμα προϊόντος που υλοποιεί-αποδίδει τον όρο “NoSQL” ως “Not only SQL”.

Ακολουθεί μικρή αναφορά στους όρους και περιγραφή στις ορολογίες NoSQL, Document Store και JSON document.

Βάσεις Δεδομένων NoSQL

Οι βάσεις δεδομένων NoSQL αποτελούν μη σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων (non-relational DBMS), δεν έχουν σταθερό σχήμα (schema), δεν χρειάζονται συνδέσεις (joins) και είναι εύκολα επεκτάσιμες. Η χρήση βάσεων αυτού του τύπου είναι κατάλληλη για κατανεμημένα συστήματα αποθήκευσης και διαχείρισης δεδομένων μεγάλης κλίμακας (big data) σε εφαρμογές ιστού πραγματικού χρόνου (real-time web apps). Εταιρείες όπως Google, Facebook και Amazon χρησιμοποιούν NoSQL βάσεις δεδομένων γιατί συλλέγουν δεδομένα μεγέθους terabyte, σε καθημερινή βάση. Τα παραδοσιακά σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων (RDBMS) χρησιμοποιούν τη γλώσσα SQL για την αποθήκευση και την ανάκτηση των δεδομένων τους. Τα συστήματα βάσης δεδομένων NoSQL χρησιμοποιεί ένα ευρύ φάσμα τεχνολογιών βάσεων δεδομένων οι οποίες μπορούν να αποθηκεύουν δομημένα, ημι-δομημένα, μη δομημένα και πολυμορφικά δεδομένα. Μια από αυτές τις τεχνολογίες είναι και το μοντέλο εγγράφων.

Βάσεις εγγράφων

Οι βάσεις δεδομένων που βασίζονται σε αυτό το μοντέλο είναι ουσιαστικά συλλογές από έγγραφα. Κάθε έγγραφο είναι ένα σύνολο τιμών κλειδιών όπου το κλειδί επιτρέπει την πρόσβαση στην τιμή του. Μπορούν όμως να περιέχουν και πολλά διαφορετικά ζεύγη κλειδιού-τιμής ή ζεύγη συστοιχιών κλειδιών ή ακόμη και ένθετα έγγραφα. Επειδή τα έγγραφα δεν είναι υποχρεωτικό να έχουν σχήμα, είναι ευέλικτα και εύκολα στις αλλαγές. Τέλος, επειδή αποθηκεύονται σε συλλογές μπορούν να ομαδοποιούν διαφορετικά είδη δεδομένων. Ο πίνακας 2.10 παρουσιάζει την αντιστοίχιση ενός κλασσικού σχεσιακού μοντέλου με το μοντέλο εγγράφων.

Πίνακας 2.10 Αντιστοίχιση σχεσιακού μοντέλου και μοντέλου εγγράφων

Σχεσιακό Μοντέλο	Μοντέλο Εγγράφων
Πίνακες	Συλλογές
Γραμμές	Έγγραφα
Στήλες	Ζεύγη κλειδιών-τιμών
Συνδέσεις (Joins)	Δεν υποστηρίζεται

Βάσεις Δεδομένων Αποθήκευσης Εγγράφων (Document Store databases)

Στις βάσεις εγγράφων τα δεδομένα αποθηκεύονται ως τιμή και συσχετίζονται με το κλειδί τους. Η τιμή μπορεί να έχει δομημένα ή ημι-δομημένα δεδομένα τα οποία αποτελούν το έγγραφο. Τα δομημένα/ημι-δομημένα δεδομένα που απαρτίζουν το έγγραφο μπορούν να κωδικοποιηθούν χρησιμοποιώντας διάφορες μορφές (μορφότυπους, format). Οι μορφότυποι XML και JSON είναι οι πλέον διαδεδομένοι σε εφαρμογές web και αποτυπώνουν τα δεδομένα με τρόπο είναι κατανοητό και αναγνώσιμο και από τον άνθρωπο και από τη μηχανή. Ακολουθούν παραδείγματα αποτύπωσης των δεδομένων.

JSON format	XML format
{	<student>
'Name' : { 'Jim Adams' },	<name>Jim Adams</name>
'Address' : { 'Thisseos 15' },	<address>Thisseos 15</address>
'Semester' : { '6' },	<semester>6</semester>
'Courses' : [	<courses>
{	<course>
'coursename' : 'Database I',	<coursename>Database I</coursename>

```
'mark' : 7
}, {
'coursename' : 'Database II',
'mark' : 8
}, {
'coursename' : 'Big data',
'mark' : 9
}
]
}
```

```
      <mark>7</mark>
   </course>
   <course>
      <coursename>Database II</coursename>
      <mark>8</mark>
   </course>
   <course>
      <coursename> Big data</coursename>
      <mark>9</degree>
   </course>
</courses>
</student>
```

Η εισαγωγή των δεδομένων θα μπορούσε να γίνει σε τρεις διαφορετικούς πίνακες σχεσιακής βάσης δεδομένων (πίνακες φοιτητών, μαθημάτων και εγγραφής φοιτητών σε μαθήματα) οι οποίοι θα συνδέονται μεταξύ τους με συσχέτιση (μέσω του μηχανισμού) των στηλών πρωτεύοντος και ξένου κλειδιού. Σε μια βάση εγγράφων, όλα τα δεδομένα μπορούν να αποθηκευτούν σε ένα πίνακα και πιθανώς και σε μια στήλη ανάλογα με το σκοπό της βάσης. Δηλαδή, οι σχεσιακές βάσεις δεδομένων αποθηκεύουν τα δεδομένα μίας οντότητας σε πολλούς συσχετιζόμενους πίνακες ενώ οι βάσεις εγγράφων αποθηκεύουν τα δεδομένα μιας οντότητας σε ένα μόνο έγγραφο. Όλα τα συσχετιζόμενα δεδομένα αποθηκεύονται μέσα σε αυτό το έγγραφο.

Στις σχεσιακές βάσεις δεδομένων πρέπει να δημιουργηθεί το σχήμα πριν εισαχθούν τα δεδομένα ενώ στις βάσεις εγγράφων, και σε άλλες NoSQL βάσεις δεδομένων, τα δεδομένα εισάγονται απευθείας στη βάση χωρίς να έχει προκαθοριστεί κάποιο σχήμα και κάθε επιπλέον κάθε έγγραφο μπορεί να έχει διαφορετική δομή και τύπο δεδομένων. Στις βάσεις εγγράφων δεν έχουμε το μηχανισμό πρωτεύοντος κλειδιού και ξένων κλειδιών αλλά τα δεδομένα τα οποία σχετίζονται με ένα έγγραφο συνήθως αποθηκεύονται στο ίδιο έγγραφο.

Έγγραφα τύπου JSON

Η γλώσσα JSON (JavaScript Object Notation) βασίζεται σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript, ακολουθεί τις συνηθισμένες συμβάσεις των γλωσσών προγραμματισμού στη γραφή της, και είναι μία μορφή κειμένου (μορφή-format JSON) κατανοητή σε άνθρωπο και μηχανή. Είναι μία εύκολη γλώσσα ανταλλαγής δεδομένων.

Ένα αντικείμενο (object) JSON περιλαμβάνει μία συλλογή δεδομένων της μορφής ζευγών (key, value), όπου value (τιμή) μπορεί να είναι μια μεμονωμένη τιμή ή πίνακες ή συνδυασμός των δύο. Περιβάλλεται από αριστερό και δεξιό σύμβολο τύπου { και }. Κάθε ζεύγος (key, value) διαχωρίζεται με κόμμα από το επόμενο, το key (όνομα) περικλείεται σε εισαγωγικά (") και αν value (τιμή) είναι συμβολοσειρά (string) τότε και αυτή περικλείεται σε εισαγωγικά. Ο διαχωρισμός του ονόματος με την τιμή γίνεται με άνω-κάτω τελεία (:) . Τέλος η τιμή ενός key-ονόματος μπορεί να είναι συμβολοσειρά, αριθμός, αντικείμενο, πίνακας, Boolean τιμή (true, false) ή null. Αν η τιμή είναι πίνακας, τότε το περιεχόμενο του πίνακα περιβάλλεται από αγκύλες τύπου [και]. Ακολουθούν δύο παραδείγματα JSON αντικειμένων.

Παράδειγμα 1

Το JSON object περιέχει ζεύγη ονόματος/τιμής

```
{ "name":"Jim", "age":35, "car":null }
```

Παράδειγμα 2

Το JSON object περιέχει ζεύγη ονόματος/τιμής και η τιμή ενός ονόματος είναι πίνακας.

```
{
"name":"Jim",
"age":35,
"cars":[ "Ford", "BMW" ]
}
```

2.16.1 Το προϊόν MySQL και συναρτήσεις διαχείρισης αντικειμένων JSON

Το προϊόν MySQL, στην «υβριδική» του μορφή (σχεσιακή βάση και αποθήκη εγγράφων), υποστηρίζει τον τύπο δεδομένων JSON και παρέχει συναρτήσεις που επιτρέπουν στους προγραμματιστές να διαχειρίζονται JSON δεδομένα, όπως ακριβώς συμβαίνει και με τους άλλους τύπους δεδομένων που υποστηρίζει το σχεσιακό προϊόν MySQL. Ο Πίνακας 2.11 παρουσιάζει μερικές από τις συναρτήσεις αυτές και την περιγραφή τους. Στην ενδεικτική βιβλιογραφία μπορείτε να μελετήσετε το σύνολο των JSON συναρτήσεων.

Πίνακας 2.11 Συναρτήσεις JSON

JSON_ARRAY()	Δημιουργία πίνακα JSON
JSON_ARRAY_APPEND()	Προσθήκη δεδομένων σε πίνακα JSON
JSON_ARRAY_INSERT()	Εισαγωγή σε πίνακα JSON
JSON_ARRAYAGG()	Επιστροφή συνόλου αποτελεσμάτων σε μορφή πίνακα JSON
JSON_CONTAINS()	Εξετάζει αν το JSON έγγραφο περιέχει συγκεκριμένα δεδομένα στη διαδρομή (path)
JSON_CONTAINS_PATH()	Εξετάζει αν το JSON έγγραφο περιέχει δεδομένα στη διαδρομή που ορίζεται
JSON_DEPTH()	Μέγιστο βάθος του εγγράφου JSON
JSON_EXTRACT()	Επιστροφή δεδομένων από το έγγραφο JSON
JSON_INSERT()	Εισαγωγή δεδομένων στο έγγραφο JSON
JSON_KEYS()	Επιστρέφει πίνακα κλειδιών από JSON έγγραφο
JSON_LENGTH()	Αριθμός στοιχείων στο έγγραφο JSON
JSON_MERGE_PATCH()	Συγχώνευση εγγράφων JSON αντικαθιστώντας τιμές διπλών κλειδιών
JSON_MERGE_PRESERVE()	Συγχώνευση εγγράφων JSON, διατηρώντας διπλά κλειδιά
JSON_OBJECT()	Δημιουργία αντικειμένου JSON
JSON_OBJECTAGG()	Επιστρέφει σύνολο αποτελεσμάτων σε JSON αντικείμενο
JSON_OVERLAPS() (εισήχθη 8.0.17)	Συγκρίνει δύο έγγραφα JSON, επιστρέφει TRUE (1) εάν αυτά έχουν κοινά ζεύγη κλειδιών-τιμών ή κοινά στοιχεία πίνακα, διαφορετικά επιστρέφει FALSE (0)
JSON_PRETTY()	Εκτυπώνει ένα έγγραφο JSON σε μορφή αναγνώσιμη από τον άνθρωπο
JSON_QUOTE()	Προσθέτει διπλούς χαρακτήρες εισαγωγικών σε μια συμβολοσειρά και την παραθέτει ως JSON τιμή
JSON_REMOVE()	Απομακρύνει δεδομένα από JSON έγγραφο
JSON_REPLACE()	Αντικαθιστά τιμές σε JSON έγγραφο
JSON_SCHEMA_VALID()	Επικυρώνει ένα JSON έγγραφο κατά JSON σχήμα

JSON_SCHEMA_VALIDATION_REPORT()	Επικυρώνει ένα JSON έγγραφο κατά JSON σχήμα και επιστρέφει αναφορά σε μορφή JSON σχετικά με το αποτέλεσμα της επικύρωσης
JSON_SEARCH()	Αναζητά τιμή σε JSON έγγραφο
JSON_SET()	Εισάγει δεδομένα σε JSON έγγραφο
JSON_STORAGE_FREE()	Δείχνει πόσος αποθηκευτικός χώρος απελευθερώθηκε στη δυαδική αναπαράστασή του μετά από μερική ενημέρωση
JSON_STORAGE_SIZE()	Δείχνει το χώρο που χρησιμοποιείται για την αποθήκευση δυαδικής αναπαράστασης ενός εγγράφου JSON
JSON_TABLE()	Εξάγει δεδομένα από ένα έγγραφο JSON και τα επιστρέφει ως σχεσιακό πίνακα
JSON_TYPE()	Επιστρέφει τον τύπο του JSON εγγράφου
JSON_UNQUOTE()	Απομακρύνει ειδικούς χαρακτήρες και επιστρέφει την τιμή σαν συμβολοσειρά
JSON_VALID()	Επιστρέφει εάν η τιμή JSON είναι έγκυρη
JSON_VALUE()	Εξάγει την τιμή από το έγγραφο JSON στη θέση που δείχνει η διαδρομή που παρέχεται.
MEMBER OF()	Επιστρέφει true (1) εάν ο πρώτος τελεστής ταιριάζει με οποιοδήποτε στοιχείο της συστοιχίας JSON που έχει περάσει ως δεύτερος τελεστής, διαφορετικά επιστρέφει ψευδής (0)

Ομαδοποίηση συναρτήσεων JSON

Κάποιες συναρτήσεις JSON μπορούν να ομαδοποιηθούν με βάση τις κυριότερες κατηγορίες εργασιών με τις οποίες σχετίζονται. Έτσι, έχουμε τις παρακάτω κατηγορίες:

- Συναρτήσεις που δημιουργούν τιμές JSON
- Συναρτήσεις που αναζητούν τιμές JSON
- Συναρτήσεις που τροποποιούν τιμές JSON
- Συναρτήσεις που επιστρέφουν τιμές JSON

Συναρτήσεις που δημιουργούν τιμές JSON

Στην ομάδα αυτή κατατάσσονται οι συναρτήσεις:

- JSON_ARRAY([val[, val] ...])
- JSON_OBJECT([key, val[, key, val] ...])
- JSON_QUOTE(string)
- JSON_ARRAYAGG()
- JSON_OBJECTAGG()

Συναρτήσεις που αναζητούν τιμές JSON

Σε αυτή την ομάδα ανήκουν οι συναρτήσεις:

- JSON_CONTAINS(target, candidate[, path])
- JSON_CONTAINS_PATH(json_doc, one_or_all, path[, path] ...)

- JSON_EXTRACT(json_doc, path[, path] ...)
- JSON_KEYS(json_doc[, path])
- JSON_OVERLAPS(json_doc1, json_doc2)
- JSON_SEARCH(json_doc, one_or_all, search_str[, escape_char[, path] ...])
- JSON_VALUE(json_doc, path)
- value MEMBER OF(json_array)

Συναρτήσεις που τροποποιούν τιμές JSON

Οι συναρτήσεις αυτής της ομάδας είναι οι παρακάτω:

- JSON_ARRAY_APPEND(json_doc, path, val[, path, val] ...)
- JSON_ARRAY_INSERT(json_doc, path, val[, path, val] ...)
- JSON_INSERT(json_doc, path, val[, path, val] ...)
- JSON_REPLACE(json_doc, path, val[, path, val] ...)
- JSON_SET(json_doc, path, val[, path, val] ...)
- JSON_MERGE_PATCH(json_doc, json_doc[, json_doc] ...)
- JSON_MERGE_PRESERVE(json_doc, json_doc[, json_doc] ...)
- JSON_REMOVE(json_doc, path[, path] ...)
- JSON_UNQUOTE(json_val)

Συναρτήσεις που επιστρέφουν τιμές JSON

Τέλος στην ομάδα των συναρτήσεων που επιστρέφουν τιμές JSON, ανήκουν οι συναρτήσεις:

- JSON_DEPTH(json_doc)
- JSON_LENGTH(json_doc[, path])
- JSON_TYPE(json_val)
- JSON_VALID(val)

Εκφράσεις μονοπατιού (διαδρομής)

Το μονοπάτι ή διαδρομή (path) χρησιμοποιείται για να προσδιορίσει το στοιχείο στο οποίο θα εφαρμοστεί η ενέργεια της συνάρτησης μέσα σε JSON έγγραφο ή πίνακα. Μια έκφραση μονοπατιού περικλείεται σε απλά (μονά) εισαγωγικά και ξεκινά πάντα με το σύμβολο $. Ανάλογα με το τι προσδιορίζει, ακολουθεί είτε η θέση του στοιχείου μέσα σε αγκύλες [], δηλαδή ένας αριθμός, είτε το όνομα του στοιχείου, είτε και τα δύο. Ας δούμε ένα παράδειγμα από κάθε περίπτωση για να γίνει πιο κατανοητός ο τρόπος σύνταξης.

Ας υποθέσουμε ότι έχουμε έναν JSON πίνακα και θέλουμε να διαγράψουμε το δεύτερο στοιχείο του, δηλαδή τον αριθμό 5. Θα χρησιμοποιούσαμε τη συνάρτηση JSON_REMOVE() με τον παρακάτω τρόπο:

```
mysql> SET @a = '[3, 5, 7, 9 ]';
mysql> SELECT JSON_REMOVE(@a, '$[1]');
```

Ας υποθέσουμε ότι έχουμε ένα JSON έγγραφο που αποτελείται από ζεύγη της μορφής (όνομα, τιμή) και θέλουμε να αντικαταστήσουμε την τιμή ενός ονόματος. Θα χρησιμοποιούσαμε τη συνάρτηση JSON_REPLACE() με τον ακόλουθο τρόπο:

```
mysql> SET @x = '{"city": "Athens", "country": "Greece"}';
mysql> SELECT JSON_REPLACE(@j, '$. city, 'Athens');
```

Αν τώρα το στοιχείο "city", στο παραπάνω παράδειγμα, είναι ένας πίνακας που περιλαμβάνει πολλές πόλεις και θέλαμε να αντικαταστήσουμε το όνομα της πρώτης πόλης του πίνακα, δηλαδή την τιμή "Athens", τότε θα προσδιορίζαμε τη διαδρομή με τον παρακάτω τρόπο:

```
mysql> SET @y = '{"city": ["Athens", "Patras", "Piraeus"], "country":
"Greece"}';
mysql> SELECT JSON_REPLACE(@y, '$. city[0], 'Thessaloniki');
```

Ακολουθούν παραδείγματα.

2.16.2 Παραδείγματα χρήσης συναρτήσεων JSON σε προϊόν MySQL. Μελέτη περίπτωσης

Σκοπός της ενότητας είναι να ξεκαθαρίσει τη χρήση JSON συναρτήσεων μέσα από μία απλή μελέτη περίπτωσης (case study). Θα υποθέσουμε ότι έχουμε ένα ηλεκτρονικό βιβλιοπωλείο που πουλά βιβλία πληροφορικής. Στη βάση δεδομένων e_bookstore αποθηκεύονται στοιχεία εκδοτών, θεματικών κατηγοριών και βιβλίων. Η μελέτη περίπτωσης θα διεκπεραιωθεί σε πέντε βήματα.

Βήμα 1. Καθορισμός του σχήματος της βάσης δεδομένων

Θα δημιουργήσετε το σχήμα που ορίζει τη βάση δεδομένων ενός διαδικτυακού βιβλιοπωλείου που πουλά μια ποικιλία βιβλίων πληροφορικής. Η βάση δεδομένων θα ονομάζεται e_bookstore και θα έχει τρεις πίνακες που ονομάζονται publishers (εκδότες), categories (κατηγορίες) και books (βιβλία).

Δημιουργήστε τη βάση δεδομένων e_bookstore:

```
CREATE DATABASE IF NOT EXISTS `e_bookstore`
DEFAULT CHARACTER SET utf8
DEFAULT COLLATE utf8_general_ci;
SET default_storage_engine = INNODB;
```

Οι πίνακες εκδοτών και κατηγοριών θα έχουν ο καθένας ένα αναγνωριστικό (id) και μία στήλη ονόματος.

Δημιουργήστε τον πίνακα εκδοτών:

```
CREATE TABLE `e_bookstore`.`publishers`(
      `id` INT UNSIGNED NOT NULL auto_increment ,
      `name` VARCHAR(250) NOT NULL ,
      PRIMARY KEY(`id`)
);
```

Δημιουργούμε τον πίνακα κατηγοριών:

```
CREATE TABLE `e_bookstore`.`categories`(
      `id` INT UNSIGNED NOT NULL auto_increment ,
      `name` VARCHAR(250) NOT NULL ,
      PRIMARY KEY(`id`)
);
```

Στη συνέχεια, προσθέτουμε ένα μικρό δείγμα εκδοτών:

```
INSERT INTO `e_bookstore`.`publishers`(`name`)
VALUES ('Springer');
INSERT INTO `e_bookstore`.`publishers`(`name`)
VALUES ('National Geographic Society');
INSERT INTO `e_bookstore`.`publishers`(`name`)
VALUES ('Chronicle Books');
```

Βλέπουμε τους εκδότες.

```
SELECT * FROM publishers;
```

```
mysql> SELECT * FROM publishers;
+----+----------------------------+
| id | name                       |
+----+----------------------------+
|  1 | Springer                   |
|  2 | National Geographic Society |
|  3 | Chronicle Books            |
+----+----------------------------+
3 rows in set (0.01 sec)
```

Στη συνέχεια, προσθέτουμε μερικές κατηγορίες:

```
INSERT INTO `e_bookstore`.`categories`(`name`)
VALUES ('Information technology');
INSERT INTO `e_bookstore`.`categories`(`name`)
VALUES ('Medical Informatics');
INSERT INTO `e_bookstore`.`categories`(`name`)
VALUES ('Computer science');
```

Βλέπουμε τους εκδότες.

```
SELECT * FROM categories;
```

```
mysql>
mysql> SELECT * FROM categories;
+----+-----------------------+
| id | name                  |
+----+-----------------------+
|  1 | Information technology |
|  2 | Medical Informatics   |
|  3 | Computer science      |
+----+-----------------------+
3 rows in set (0.00 sec)
```

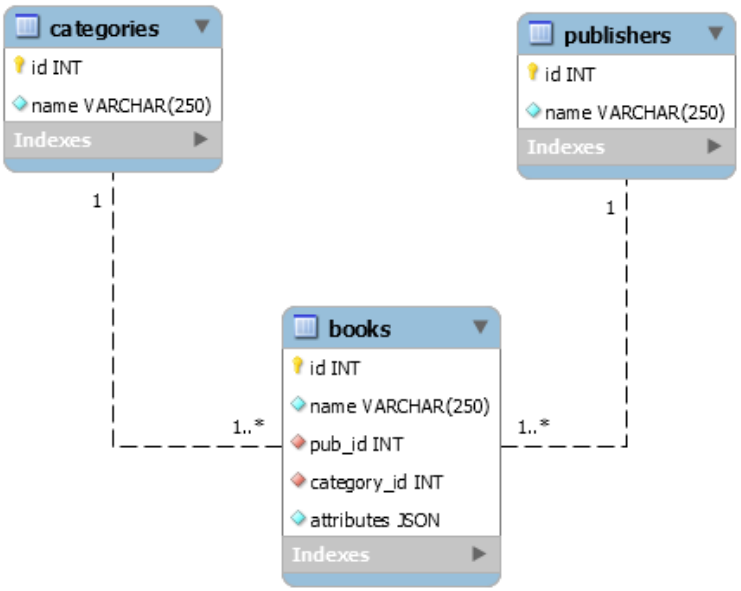
Στη συνέχεια, δημιουργούμε έναν πίνακα βιβλίων με στήλες id, name, pub_id, category_id:

```
CREATE TABLE `e_bookstore`.`books`(
      `id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
      `name` VARCHAR(250) NOT NULL ,
      `pub_id` INT UNSIGNED NOT NULL ,
      `category_id` INT UNSIGNED NOT NULL ,
      `attributes` JSON NOT NULL ,
      PRIMARY KEY(`id`) ,
```

```
      INDEX `CATEGORY_ID`(`category_id` ASC) ,
      INDEX `PUB_ID`(`pub_id` ASC) ,
      CONSTRAINT `pub_id` FOREIGN KEY(`pub_id`) REFERENCES
`e_bookstore`.`publishers`(`id`) ON DELETE RESTRICT ON UPDATE CASCADE ,
      CONSTRAINT `category_id` FOREIGN KEY(`category_id`) REFERENCES
`e_bookstore`.`categories`(`id`) ON DELETE RESTRICT ON UPDATE CASCADE
);
```

Ο ορισμός του πίνακα καθορίζει τους περιορισμούς ξένων κλειδιών για τις στήλες pub_id και category_id και προσδιορίζει ότι οι στήλες αυτές αναφέρονται (referenced) στον πίνακα εκδοτών και κατηγοριών αντίστοιχα. Επίσης καθορίζει ότι οι αναφερόμενες γραμμές (των πινάκων publishers και categories αντίστοιχα) δεν επιτρέπεται να διαγραφούν και εάν ενημερωθούν, οι αλλαγές θα πρέπει να αντικατοπτρίζονται και στις αναφερόμενες γραμμές. Ο τύπος στήλης JSON είναι πλέον εγγενής τύπος δεδομένων στο προϊόν SQL.

Στην εικόνα 2.75 παραθέτουμε το μοντέλο οντοτήτων συσχετίσεων.



Εικόνα 2.75 ΜΟΣ ηλεκτρονικού βιβλιοπωλείου

Επισημαίνουμε ότι ο σχεδιασμός της βάσης δεδομένων δεν είναι άρτιος επειδή στο παράδειγμά μας δεν περιλαμβάνονται σημαντικές στήλες στον πίνακα των βιβλίων, π.χ. η ημερομηνία έκδοσης, η τιμή, και κυρίως επειδή ο σχεδιασμός δεν επιτρέπει τη διαχείριση προϊόντων που ανήκουν σε πολλές κατηγορίες. Μας επιτρέπει όμως, επειδή είναι πολύ απλός, να επικεντρωθούμε στις δυνατότητες διαχείρισης JSON του προϊόντος MySQL.

Βήμα 2. Δημιουργία δεδομένων σε στήλη τύπου JSON

Στο βήμα αυτό θα προσθέσουμε βιβλία στη βάση δεδομένων. Ακολουθούν μερικά παραδείγματα εισαγωγής αντικειμένων JSON (JSON objects) τα οποία περιλαμβάνουν στοιχεία για το μέγεθος του βιβλίου (size), την έκδοση (edition), μία λέξη-κλειδί (keyword), τη βιβλιοδέτηση (binding):

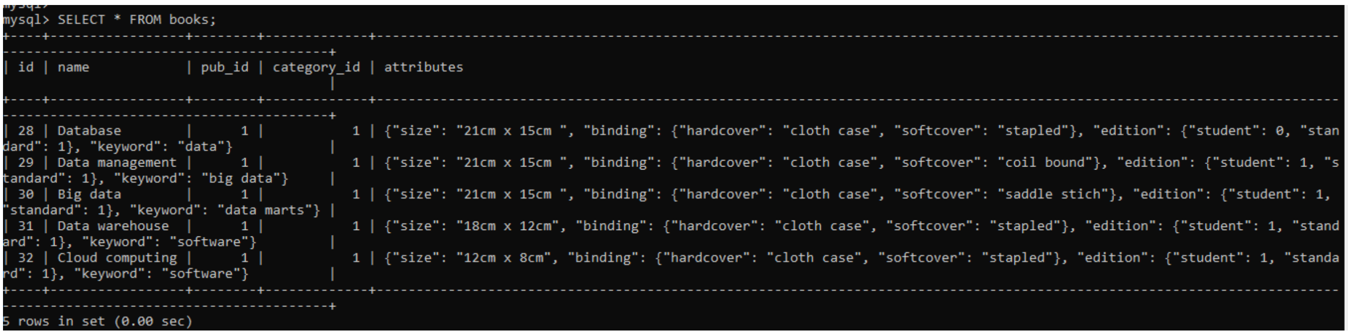
```
INSERT INTO `e_bookstore`.`books`(`name` , `pub_id` , `category_id` ,
`attributes`)
VALUES('Database' , '1' , '1' ,
'{"keyword": "data", "size": "21cm x 15cm ", "edition": {"student": 0,
"standard": 1}, "binding": {"hardcover": "cloth case", "softcover":
```

```
"stapled"}}’);
INSERT INTO `e_bookstore`.`books`(  `name` , `pub_id` , `category_id` ,
`attributes`)
VALUES(‘Data management’ ,  ‘1’ , ‘1’ ,
‘{"keyword": "big data", "size": "21cm x 15cm ", "edition": {"student": 1,
"standard": 1}, "binding": {"hardcover": "cloth case", "softcover": "coil
bound"}}’);

INSERT INTO `e_bookstore`.`books`(`name` , `pub_id` , `category_id` ,
`attributes`)
VALUES(‘Big data’ , ‘1’ , ‘1’ ,
‘{"keyword": "data marts", "size": "21cm x 15cm ", "edition": {"student": 1,
"standard": 1}, "binding": {"hardcover": "cloth case", "softcover": "saddle
stich"}}’);

INSERT INTO `e_bookstore`.`books`(`name` , `pub_id` , `category_id` ,
`attributes`)
VALUES(‘Data warehouse’ , ‘1’ , ‘1’ ,
‘{"keyword": "software", "size": "18cm x 12cm", "edition": {"student": 1,
"standard": 1}, "binding": {"hardcover": "cloth case", "softcover":
"stapled"}}’);

INSERT INTO `e_bookstore`.`books`(`name` , `pub_id` , `category_id` ,
`attributes`)
VALUES(‘Cloud computing’ , ‘1’ , ‘1’ ,
‘{"keyword": "software", "size": "12cm x 8cm", "edition": {"student": 1,
"standard": 1}, "binding": {"hardcover": "cloth case", "softcover":
"stapled"}}’);
```

Ακολουθούν κάποιες διευκρινίσεις που αφορούν τον τρόπο που βλέπουμε τα στοιχεία. Μια πληρέστερη συζήτηση γίνεται στο επόμενο βήμα.

Βλέπουμε τα στοιχεία των βιβλίων.

```
SELECT * FROM books;
```



```
mysql> SELECT * FROM books;
+----+-----------------+--------+-------------+--------------------------------------------------------------------------------------------------
--------------------------------------+
| id | name            | pub_id | category_id | attributes
                                      |
+----+-----------------+--------+-------------+--------------------------------------------------------------------------------------------------
--------------------------------------+
| 28 | Database        |      1 |           1 | {"size": "21cm x 15cm ", "binding": {"hardcover": "cloth case", "softcover": "stapled"}, "edition": {"student": 0, "stan
dard": 1}, "keyword": "data"}         |
| 29 | Data management |      1 |           1 | {"size": "21cm x 15cm ", "binding": {"hardcover": "cloth case", "softcover": "coil bound"}, "edition": {"student": 1, "s
tandard": 1}, "keyword": "big data"}  |
| 30 | Big data        |      1 |           1 | {"size": "21cm x 15cm ", "binding": {"hardcover": "cloth case", "softcover": "saddle stich"}, "edition": {"student": 1,
"standard": 1}, "keyword": "data marts"} |
| 31 | Data warehouse  |      1 |           1 | {"size": "18cm x 12cm", "binding": {"hardcover": "cloth case", "softcover": "stapled"}, "edition": {"student": 1, "stand
ard": 1}, "keyword": "software"}      |
| 32 | Cloud computing |      1 |           1 | {"size": "12cm x 8cm", "binding": {"hardcover": "cloth case", "softcover": "stapled"}, "edition": {"student": 1, "standa
rd": 1}, "keyword": "software"}       |
+----+-----------------+--------+-------------+--------------------------------------------------------------------------------------------------
--------------------------------------+
5 rows in set (0.00 sec)
```

```
SELECT  books.id, books.name, books.pub_id, books.category_id
FROM books
JOIN publishers ON books.id=publishers.id
JOIN categories ON books.id=categories.id;
SELECT  books.id, books.name,
  books.pub_id, publishers.name,
  books.category_id, categories.name
FROM books
```

```
JOIN publishers ON books.id=publishers.id
JOIN categories ON books.id=categories.id;
```

```
mysql>
mysql> SELECT  books.id, books.name, books.pub_id, books.category_id
    -> FROM books
    -> JOIN publishers ON books.id=publishers.id
    -> JOIN categories ON books.id=categories.id;
+----+-----------------+--------+-------------+
| id | name            | pub_id | category_id |
+----+-----------------+--------+-------------+
|  1 | Database        |      1 |           1 |
|  2 | Data management |      1 |           1 |
|  3 | Big data        |      1 |           1 |
+----+-----------------+--------+-------------+
3 rows in set (0.00 sec)

mysql>
mysql> SELECT  books.id, books.name,
    ->       books.pub_id, publishers.name,
    ->       books.category_id, categories.name
    -> FROM books
    -> JOIN publishers ON books.id=publishers.id
    -> JOIN categories ON books.id=categories.id;
+----+-----------------+--------+------------------------------+-------------+------------------------+
| id | name            | pub_id | name                         | category_id | name                   |
+----+-----------------+--------+------------------------------+-------------+------------------------+
|  1 | Database        |      1 | Springer                     |           1 | Medical Informatics    |
|  2 | Data management |      1 | National Geographic Society  |           1 | Computer science       |
|  3 | Big data        |      1 | Chronicle Books              |           1 | Information technology |
+----+-----------------+--------+------------------------------+-------------+------------------------+
3 rows in set (0.00 sec)
```

Στο παράδειγμά μας δηλώσαμε πέντε διαφορετικά βιβλία τεχνολογίας πληροφοριών.

```
SELECT books.id, books.name, attributes
FROM books;
```

```
mysql>
mysql> SELECT books.id, books.name, attributes
    -> FROM books;
+----+-----------------+------------------------------------------------------------------------------------------------------------------------------
-----------------+
| id | name            | attributes
                 |
+----+-----------------+------------------------------------------------------------------------------------------------------------------------------
-----------------+
|  1 | Database        | {"size": "21cm x 15cm ", "binding": {"hardcover": "cloth case", "softcover": "stapled"}, "edition": {"student": 0, "standard": 1}, "keyword": "
data"}           |
|  2 | Data management | {"size": "21cm x 15cm ", "binding": {"hardcover": "cloth case", "softcover": "coil bound"}, "edition": {"student": 1, "standard": 1}, "keyword"
: "big data"}    |
|  3 | Big data        | {"size": "21cm x 15cm ", "binding": {"hardcover": "cloth case", "softcover": "saddle stich"}, "edition": {"student": 1, "standard": 1}, "keywor
d": "data marts"} |
|  4 | Data warehouse  | {"size": "18cm x 12cm", "binding": {"hardcover": "cloth case", "softcover": "stapled"}, "edition": {"student": 1, "standard": 1}, "keyword": "s
oftware"}        |
|  5 | Cloud computing | {"size": "12cm x 8cm", "binding": {"hardcover": "cloth case", "softcover": "stapled"}, "edition": {"student": 1, "standard": 1}, "keyword": "so
ftware"}         |
+----+-----------------+------------------------------------------------------------------------------------------------------------------------------
-----------------+
5 rows in set (0.00 sec)
```

Αν θέλουμε να βλέπουμε συγκεκριμένα στοιχεία τα οποία υπάρχουν στο αντικείμενο JASON πρέπει να χρησιμοποιήσουμε τη συνάρτηση JSON_EXTRACT.

```
SELECT books.id, books.name, JSON_EXTRACT(`attributes` , '$.keyword')
FROM books;
```

```
mysql>
mysql> SELECT books.id, books.name, JSON_EXTRACT(`attributes` , '$.keyword')
    -> FROM books;
+----+-----------------+-----------------------------------------+
| id | name            | JSON_EXTRACT(`attributes` , '$.keyword') |
+----+-----------------+-----------------------------------------+
|  1 | Database        | "data"                                  |
|  2 | Data management | "big data"                              |
|  3 | Big data        | "data marts"                            |
|  4 | Data warehouse  | "software"                              |
|  5 | Cloud computing | "software"                              |
+----+-----------------+-----------------------------------------+
5 rows in set (0.00 sec)
```

Αν θέλουμε να βλέπουμε για ένα συγκεκριμένο βιβλίο τον τίτλο του και τα στοιχεία τα οποία υπάρχουν στο αντικείμενο JSON με έναν κομψό (pretty) τρόπο πρέπει να χρησιμοποιήσουμε τη συνάρτηση JSON_PRETTY.

```
SELECT books.id, books.name, books.attributes
INTO @id, @name, @see_details
FROM books
WHERE id=1 ;
SELECT @id, @name, JSON_PRETTY(@see_details);
```

```
mysql> SELECT books.id, books.name, books.attributes
    -> INTO @id, @name, @see_details
    -> FROM books
    -> WHERE id=1 ;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT @id, @name, JSON_PRETTY(@see_details);
+------+----------+----------------------------------------
-----------------------------------+
| @id  | @name    | JSON_PRETTY(@see_details)
                                   |
+------+----------+----------------------------------------
-----------------------------------+
|    1 | Database | {
  "size": "21cm x 15cm ",
  "binding": {
    "hardcover": "cloth case",
    "softcover": "stapled"
  },
  "edition": {
    "student": 0,
    "standard": 1
  },
  "keyword": "data"
} |
+------+----------+----------------------------------------
-----------------------------------+
1 row in set (0.00 sec)
```

Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε την ενσωματωμένη συνάρτηση JSON_OBJECT για να δημιουργήσουμε αντικείμενα τύπου JSON. Η συνάρτηση JSON_OBJECT δέχεται μία λίστα ζευγών (κλειδί, τιμή), (key, value), και επιστρέφει ένα αντικείμενο JSON. Ακολουθείται η σύνταξη:

JSON_OBJECT(key1, value1, key2, value2, ... key(n), value(n)).

Μπορούμε, επιπλέον, να χρησιμοποιήσουμε αντί για ένα κλειδί, key, ως τιμή, value, έναν πίνακα. Ακολουθούν μερικά παραδείγματα δηλώσεων εισαγωγής στοιχείων εγχειριδίων οι οποίες χρησιμοποιούν τη συνάρτηση JSON_OBJECT.

```
INSERT INTO `e_bookstore`.`books`(
  `name` , `pub_id` ,`category_id` , `attributes`)
VALUES('Big data' , '2' , '2' ,
      JSON_OBJECT("manual" ,
JSON_ARRAY("reference", "user" , "DBA" , "developer") ,
            "body" ,
            "20cm x 14cm" ,
            "weight" ,
            "200 grams" ,
            "edition" ,
            "Standard" ,
            "Binding" ,
            "hardcover"
```

```
      )
);
```

```
INSERT INTO `e_bookstore`.`books`(
  `name` , `pub_id` ,`category_id` , `attributes`)
VALUES('MapReduce' , '2' , '2' ,
      JSON_OBJECT("manual" ,
JSON_ARRAY("reference", "user" , "DBA" , "developer") ,
            "body" ,
            "20cm x 14cm" ,
            "weight" ,
            "200 grams" ,
            "edition" ,
            "Standard" ,
            "Binding" ,
            "spiral"
      )
);
```

Στο παράδειγμα εισάγουμε στη βάση δύο διαφορετικά εγχειρίδια. Χρησιμοποιούμε τη συνάρτηση JSON_ARRAY η οποία επιστρέφει έναν πίνακα JSON όταν τις «περάσουμε» ένα σύνολο τιμών. Εάν χρησιμοποιήσουμε ένα κλειδί πολλές φορές, θα διατηρηθεί μόνο το πρώτο ζεύγος (κλειδιού, τιμής). Επιπλέον, θα μπορούσαμε να χρησιμοποιήσουμε τις ενσωματωμένες συναρτήσεις JSON_MERGE_PRESERVE ή JSON_MERGE_PATCH για να δημιουργήσουμε αντικείμενα JSON.

Παραθέτουμε παραδείγματα χρήσης της συνάρτησης JSON_MERGE_PRESERVE:

```
INSERT INTO `e_bookstore`.`books`(
 `name` , `pub_id` , `category_id` , `attributes`)
VALUES(     'Radoop' , '3' , '3' ,
      JSON_MERGE_PRESERVE(
            '{"keyword": "software"}' ,
            '{"edition": "standard"}' ,
            '{"binding": "hard cover"}' ,
            '{"size": "18cm x 12cm"}'
      )
);
```

```
INSERT INTO `e_bookstore`.`books`(
 `name` , `pub_id` , `category_id` , `attributes`)
VALUES(     'Hadoop' , '3' , '3' ,
      JSON_MERGE_PRESERVE(
            '{"keyword": "software"}' ,
            '{"edition": "standard"}' ,
            '{"binding": "softcover"}' ,
            '{"size": "18cm x 12cm"}'
      )
);
```

```
INSERT INTO `e_bookstore`.`books`(
 `name` , `pub_id` , `category_id` , `attributes`)
VALUES(     'business intelligence' ,     '3' ,'3' ,
  JSON_MERGE_PRESERVE(
  JSON_OBJECT("book_type" , "textbook") ,
            '{"keyword": "OLAP"}' ,
            '{"edition": "student"}'
      )
);
```

```
INSERT INTO `e_bookstore`.`books`(
 `name` , `pub_id` , `category_id` , `attributes`)
VALUES(     'ETL' ,     '3' ,'3' ,
JSON_MERGE_PRESERVE(
   JSON_OBJECT("binding" , "hardcover") ,
   JSON_OBJECT("edition" , "student")
      )
);
```

Στο παράδειγμα δηλώνουμε τέσσερα διαφορετικά βιβλία πληροφορικής. Υπογραμμίζουμε ότι μόνο αντικείμενα μεταβιβάζονται στη συνάρτηση JSON_MERGE_PRESERVE.

Ορισμένα αντικείμενα έχουν κατασκευαστεί χρησιμοποιώντας τη συνάρτηση JSON_OBJECT. Άλλα έχουν διαβιβαστεί («περαστεί») ως συμβολοσειρές JSON (JSON strings). Στην περίπτωση της συνάρτησης JSON_MERGE_PRESERVE, εάν ένα κλειδί επαναλαμβάνεται πολλές φορές τότε η τιμή του διατηρείται ως πίνακας.

Για παράδειγμα, στη συνέχεια βλέπουμε (εικόνα 2.76) μια συλλογή αντικειμένων JSON με το ίδιο κλειδί:

```
SELECT JSON_MERGE_PRESERVE(
      '{"keyword": "data"}' ,
      '{"keyword": "datamarts"}' ,
      '{"keyword": "Big data"}'
);
```

```
mysql>
mysql> SELECT JSON_MERGE_PRESERVE(
    -> '{"keyword": "data"}' ,
    -> '{"keyword": "datamarts"}' ,
    -> '{"keyword": "Big data"}'
    -> );
+-----------------------------------------------------------------+
| JSON_MERGE_PRESERVE(
'{"keyword": "data"}' ,
'{"keyword": "datamarts"}' ,
'{"keyword": "Big data"}'
) |
+-----------------------------------------------------------------+
| {"keyword": ["data", "datamarts", "Big data"]}
+-----------------------------------------------------------------+
1 row in set (0.00 sec)
```

Εικόνα 2.76 Συλλογή αντικειμένων JSON με το ίδιο κλειδί

Μπορείτε να επαληθεύσετε τα ερωτήματά σας χρησιμοποιώντας τη συνάρτηση JSON_TYPE για να εμφανίσετε τον τύπο στήλης (εικόνα 2.77).

```
SELECT books.id, name, JSON_TYPE(attributes) FROM `e_bookstore`.`books`;
```

```
mysql>
mysql> SELECT books.id, name, JSON_TYPE(attributes) FROM `e_bookstore`.`books`;
+----+----------------------+-----------------------+
| id | name                 | JSON_TYPE(attributes) |
+----+----------------------+-----------------------+
|  1 | Database             | OBJECT                |
|  2 | Data management      | OBJECT                |
|  3 | Big data             | OBJECT                |
|  4 | Data warehouse       | OBJECT                |
|  5 | Cloud computing      | OBJECT                |
|  6 | Big data             | OBJECT                |
|  7 | MapReduce            | OBJECT                |
|  8 | Radoop               | OBJECT                |
|  9 | Hadoop               | OBJECT                |
| 10 | business intelligence | OBJECT                |
| 11 | ETL                  | OBJECT                |
+----+----------------------+-----------------------+
11 rows in set (0.00 sec)
```

Εικόνα 2.77 Εμφάνιση τύπου στήλης

Αυτό το ερώτημα θα παράγει 11 αποτελέσματα αντικειμένων που αντιπροσωπεύουν όλα τα βιβλία που έχουμε εισάγει στη βάση δεδομένων.

Μέχρι τώρα είδαμε πως μπορούμε να δημιουργήσουμε δεδομένα σε στήλη τύπου JSON.

Βήμα 3. Ανάγνωση των δεδομένων από στήλη τύπου JSON

Τώρα που έχουμε εισάγει μερικά βιβλία στη βάση δεδομένων, μπορούμε να πειραματιστούμε με την ανάγνωση των δεδομένων. Για τις συνηθισμένες στήλες που δεν είναι τύπου JSON, θα βασιστούμε στη συνθήκη WHERE.

Αν θέλουμε να επιλέξουμε γραμμές χρησιμοποιώντας αντικείμενο τύπου JSON θα πρέπει να εξοικειωθούμε με την έννοια της έκφρασης διαδρομής (path expression). Οι εκφράσεις διαδρομής χρησιμοποιούν το σύμβολο δολαρίου ($) και τα κλειδιά (keys) το αντικειμένου που μας ενδιαφέρει, π.χ., '$.keyword'. Όταν χρησιμοποιείται σε συνδυασμό με τη συνάρτηση JSON_EXTRACT, μπορούμε να ανακτήσουμε τις τιμές για την καθορισμένη στήλη, π.χ.,

JSON_EXTRACT(`attributes` , '$.keyword')

Ακολουθούν παραδείγματα.

```
SELECT *
FROM  `e_bookstore`.`books`
WHERE `category_id` = 1
AND JSON_EXTRACT(`attributes` , '$.edition.standard') > 0
AND JSON_EXTRACT(`attributes` , '$.edition.student') > 0;
```

```
mysql>
mysql> SELECT *
    -> FROM `e_bookstore`.`books`
    -> WHERE `category_id` = 1
    -> AND JSON_EXTRACT(`attributes` , '$.edition.standard') > 0
    -> AND JSON_EXTRACT(`attributes` , '$.edition.student') > 0;
| id | name            | pub_id | category_id | attributes
|  2 | Data management |      1 |           1 | {"size": "21cm x 15cm ", "binding": {"hardcover": "cloth case", "softcover": "coil bound"}, "edition": {"student": 1, "standard": 1}, "keyword": "big data"}      |
|  3 | Big data        |      1 |           1 | {"size": "21cm x 15cm ", "binding": {"hardcover": "cloth case", "softcover": "saddle stich"}, "edition": {"student": 1, "standard": 1}, "keyword": "data marts"} |
|  4 | Data warehouse  |      1 |           1 | {"size": "18cm x 12cm", "binding": {"hardcover": "cloth case", "softcover": "stapled"}, "edition": {"student": 1, "standard": 1}, "keyword": "software"}        |
|  5 | Cloud computing |      1 |           1 | {"size": "12cm x 8cm", "binding": {"hardcover": "cloth case", "softcover": "stapled"}, "edition": {"student": 1, "standard": 1}, "keyword": "software"}         |
4 rows in set (0.00 sec)
```

Το πρώτο όρισμα στη συνάρτηση JSON_EXTRACT είναι το αντικείμενο JSON για τον καθορισμό της «έκφρασης διαδρομής» (path expression) στην οποία βρίσκεται το χαρακτηριστικό που ενδιαφέρει. Το σύμβολο $ δηλώνει το αντικείμενο με το οποίο θα εργαστούμε. Για παράδειγμα, η «έκφραση διαδρομής» $.edition.standard μεταφράζεται ως «λάβετε το κλειδί (key) standard που βρίσκεται κάτω από το κλειδί edition». Όταν εξάγουμε τα κλειδιά που μας ενδιαφέρουν, μπορούμε να χρησιμοποιήσουμε τους τελεστές του προϊόντος MySQL, π.χ. τελεστές σύγκρισης.

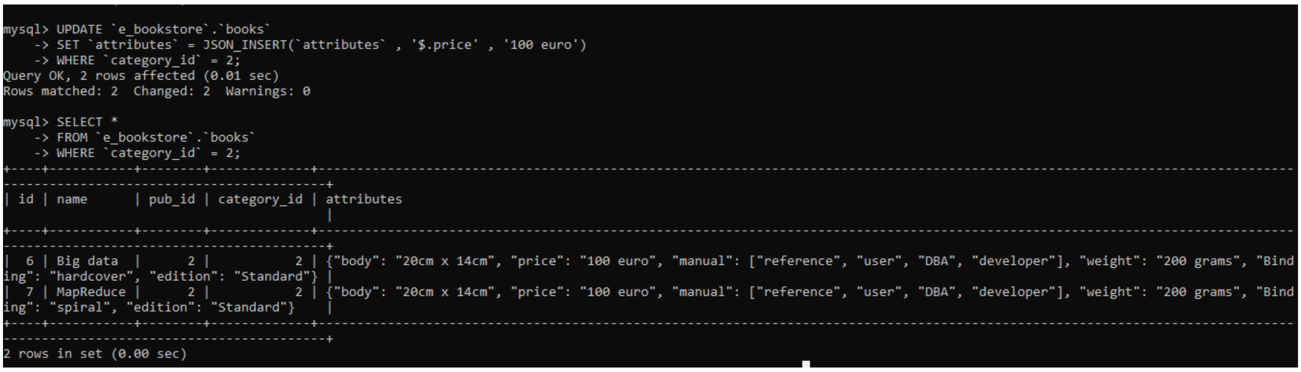
Εναλλακτικά, η συνάρτηση JSON_EXTRACT χρησιμοποιείται με το ψευδώνυμο (alias) -> που μπορούμε να χρησιμοποιήσουμε για να κάνουμε τα ερωτήματά μας πιο ευανάγνωστα. Ακολουθεί παράδειγμα.

```
SELECT *
FROM  `e_bookstore`.`books`
WHERE `category_id` = 1
AND `attributes` -> '$.edition.standard' > 0
AND `attributes` -> '$.edition.student' > 0;
```

Βήμα 4. Ενημέρωση δεδομένων σε στήλη τύπου JSON

Μπορούμε να ενημερώσουμε δεδομένα σε στήλη JSON με τις συναρτήσεις JSON_INSERT, JSON_REPLACE και JSON_SET. Αυτές οι συναρτήσεις απαιτούν επίσης μια «έκφραση διαδρομής» για να καθορίσουμε ποια μέρη του αντικειμένου JSON θέλουμε να τροποποιήσουμε.

```
UPDATE `e_bookstore`.`books`
SET `attributes` = JSON_INSERT(`attributes` ,  '$.price' , '100 euro')
WHERE `category_id` = 2;

SELECT *
FROM  `e_bookstore`.`books`
WHERE `category_id` = 2;
```



```
mysql> UPDATE `e_bookstore`.`books`
    -> SET `attributes` = JSON_INSERT(`attributes` , '$.price' , '100 euro')
    -> WHERE `category_id` = 2;
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> SELECT *
    -> FROM `e_bookstore`.`books`
    -> WHERE `category_id` = 2;
| id | name      | pub_id | category_id | attributes
|  6 | Big data  |      2 |           2 | {"body": "20cm x 14cm", "price": "100 euro", "manual": ["reference", "user", "DBA", "developer"], "weight": "200 grams", "Binding": "hardcover", "edition": "Standard"} |
|  7 | MapReduce |      2 |           2 | {"body": "20cm x 14cm", "price": "100 euro", "manual": ["reference", "user", "DBA", "developer"], "weight": "200 grams", "Binding": "spiral", "edition": "Standard"}    |
2 rows in set (0.00 sec)
```

Στη συνέχεια θα χρησιμοποιήσουμε τη συνάρτηση JSON_REPLACE.

```
UPDATE `e_bookstore`.`books`
SET `attributes` = JSON_REPLACE(`attributes` , '$.price' , 'sales')
WHERE `category_id` = 2;
```

Τέλος, θα ενημερώσουμε τη στήλη JSON με χρήση συνάρτησης JSON_SET και θα προσθέσουμε ένα νέο κλειδί cover_color (βασικό χρώμα εξωφύλλου) με την τιμή 'red' (κόκκινο).

```
UPDATE `e_bookstore`.`books`
SET `attributes` = JSON_SET(`attributes` , '$.cover_color' , 'red')
WHERE `category_id` = 1;
```

Οι συναρτήσεις που χρησιμοποιήσαμε διαφέρουν σε συμπεριφορά. Η συνάρτηση JSON_INSERT θα προσθέσει την ιδιότητα στο αντικείμενο μόνο εάν δεν υπάρχει ήδη. Η συνάρτηση JSON_REPLACE αντικαθιστά την ιδιότητα μόνο εάν βρεθεί. Η συνάρτηση JSON_SET θα προσθέσει την ιδιότητα εάν δεν βρεθεί, διαφορετικά θα την αντικαταστήσει.

Βήμα 5. Διαγραφή δεδομένων από στήλη τύπου JSON

Μπορούμε να διαγράψουμε δεδομένα σε στήλη JSON με τη συνάρτηση JSON_REMOVE και τη δήλωση DELETE. Η συνάρτηση JSON_REMOVE μας επιτρέπει να διαγράψουμε ένα συγκεκριμένο ζεύγος (κλειδί, τιμή) από τις στήλες JSON.

```
UPDATE `e_bookstore`.`books`
SET `attributes` = JSON_REMOVE(`attributes` , '$.sales')
WHERE `category_id` = 3;
```

Η συνάρτηση JSON_REMOVE επιστρέφει το ενημερωμένο αντικείμενο JSON μετά την αφαίρεση του καθορισμένου κλειδιού με βάση την «έκφραση διαδρομής». Εναλλακτικά, μπορείτε να διαγράψετε ολόκληρες γραμμές οι οποίες περιλαμβάνουν μια στήλη JSON.

Μπορούμε να χρησιμοποιήσουμε σε δήλωση DELETE και τη συνάρτηση JSON_EXTRACT και τελεστή LIKE για να διαγράψουμε γραμμές βιβλίων.

```
DELETE FROM `e_bookstore`.`books`
WHERE `category_id` = 1
AND JSON_EXTRACT(`attributes` , '$.keyword') LIKE '%data%';
```

```
mysql> DELETE FROM `e_bookstore`.`books`
    -> WHERE `category_id` = 1
    -> AND JSON_EXTRACT(`attributes` , '$.keyword') LIKE '%data%';
Query OK, 3 rows affected (0.01 sec)

mysql> select * from books;
+----+-----------------------+--------+-------------+------------
| id | name                  | pub_id | category_id | attributes
+----+-----------------------+--------+-------------+------------
|  4 | Data warehouse        |      1 |           1 | {"size": "18cm x 12cm", "binding": {"hardcover": "cloth case", "softcover": "stapled"}, "edition": {"student": 1, "standard": 1}, "keyword": "software", "cover_color": "red"} |
|  5 | Cloud computing       |      1 |           1 | {"size": "12cm x 8cm", "binding": {"hardcover": "cloth case", "softcover": "stapled"}, "edition": {"student": 1, "standard": 1}, "keyword": "software", "cover_color": "red"} |
|  6 | Big data              |      2 |           2 | {"body": "20cm x 14cm", "price": "sales", "manual": ["reference", "user", "DBA", "developer"], "weight": "200 grams", "Binding": "hardcover", "edition": "Standard"} |
|  7 | MapReduce             |      2 |           2 | {"body": "20cm x 14cm", "price": "sales", "manual": ["reference", "user", "DBA", "developer"], "weight": "200 grams", "Binding": "spiral", "edition": "Standard"} |
|  8 | Radoop                |      3 |           3 | {"size": "18cm x 12cm", "binding": "hard cover", "edition": "standard", "keyword": "software"} |
|  9 | Hadoop                |      3 |           3 | {"size": "18cm x 12cm", "binding": "softcover", "edition": "standard", "keyword": "software"} |
| 10 | business intelligence |      3 |           3 | {"edition": "student", "keyword": "OLAP", "book_type": "textbook"} |
| 11 | ETL                   |      3 |           3 | {"binding": "hardcover", "edition": "student"} |
+----+-----------------------+--------+-------------+------------
8 rows in set (0.00 sec)
```

2.17 Βιβλιογραφία

MySQL Document Store: Top 10 Reasons https://www.mysql.com/why-mysql/white-papers/mysql-document-store-top-10-reasons, πρόσβαση στις 18/5/2022

NoSQL Tutorial: What is, Types of NoSQL Databases & Example https://www.guru99.com/nosql-tutorial.html, πρόσβαση στις 18/5/2022

w3 resources NoSQL https://www.w3resource.com/mongodb/nosql.php, πρόσβαση στις 18/5/2022

What is a Document Store Database? https://database.guide/what-is-a-document-store-database, πρόσβαση στις 18/5/2022

Introducing JSON https://www.json.org/json-en.html, πρόσβαση στις 18/5/2022

MySQL 8.0 Reference Manual 2022-06-22 (revision: 73559)

https://downloads.mysql.com/docs/refman-8.0-en.a4.pdf

Μαρία Χριστοδουλάκη (2020) Εφαρμογές Βάσεων Δεδομένων με χρήση SQL και NoSQL. Η περίπτωση του προϊόντος MySQL 8, διπλωματική εργασία, Πανεπιστήμιο Δυτικής Αττικής

Κεφάλαιο 3
Υλοποίηση σχεσιακών βάσεων δεδομένων. Η γλώσσα SQL (Structured Query Language)

Σύνοψη
Στο κεφάλαιο αυτό γίνεται παρουσίαση θεμάτων της υλοποίηση σχεσιακών βάσεων δεδομένων με χρήση της Δομημένης Γλώσσας Επερωτήσεων SQL (Structured Query Language). Περιλαμβάνονται:

Εισαγωγή στη γλώσσα SQL και τα Πρότυπα ANSI. *Η γλώσσα SQL ως γλώσσα διαχείρισης σχεσιακών βάσεων δεδομένων και οι τρεις υπογλώσσες της: (1) Η γλώσσα ορισμού δεδομένων-DDL (Data Definition Language) συνδέεται με τις δηλώσεις (SQL statements) δημιουργίας και διαχείρισης (CREATE, ALTER, DROP) αντικειμένων της βάσης, όπως TABLES, VIEWS, INDEXES κ.λπ. (2) Η γλώσσα χειρισμού δεδομένων-DML (Data Manipulation Language) περιλαμβάνει τις δηλώσεις εισαγωγής, ενημέρωσης και ανάκτησης δεδομένων INSERT, UPDATE, DELETE, SELECT. (3) Η γλώσσα ελέγχου δεδομένων-DCL (Data Control Language) περιλαμβάνει δηλώσεις εκχώρησης (GRANT) και αφαίρεσης (REVOKE) δικαιωμάτων δημιουργίας χρηστών, σύνδεσης στη βάση κλπ. και δικαιωμάτων χρήσης αντικείμενων της βάσης δεδομένων. Η χρησιμοποίησή τους μαζί με τον ορισμό views και τη διαχείριση συναλλαγών (transactions) διασφαλίζει την ακεραιότητα (integrity), τη συνέπεια (consistency) και την ασφάλεια (security) των δεδομένων της βάσης δεδομένων.*

Σύνταξη δηλώσεων ορισμού δεδομένων, δηλώσεων επεξεργασίας και αναζήτησης δεδομένων, δηλώσεων ελέγχου δεδομένων. *Εισαγωγή στη χρήση της Oracle SQL και της mySQL. Επισήμανση διαφορών στην υλοποίηση βάσης δεδομένων στα δύο γνωστά προϊόντα MySQL, Oracle. Πώς γράφονται τα ονόματα πινάκων, στηλών, δεικτών (index) και όψεων. Ποιοι τύποι δεδομένων χρησιμοποιούνται συνήθως. Διαφορές ανάμεσα στους τύπους δεδομένων που υποστηρίζουν γνωστά προϊόντα διαχείρισης βάσεων δεδομένων.*

Αναζήτηση δεδομένων (SELECT) με τη γλώσσα SQL. *Αναζητήσεις που οδηγούν σε αποτελέσματα χωρίς επανάληψη τιμών (χρήση DISTINCT). Περιορισμός των αποτελεσμάτων με απλές συνθήκες (χρήση WHERE) και αναζητήσεις που περιλαμβάνουν υπολογισμούς, πράξεις σε συμβολοσειρές (strings), συναρτήσεις κ.λπ. Σχηματισμός σύνθετων συνθηκών (χρήση WHERE) που περιλαμβάνουν υπολογισμούς, τελεστές σύγκρισης, AND, OR, NOT, LIKE, NOT LIKE, BETWEEN ..... AND, NOT BETWEEN ……... AND, σύνολα (IN), υποαναζητήσεις (φωλιασμένες δηλώσεις SELECT), τελεστές ANY, ALL, EXIST και συναρτήσεις. Ταξινόμηση αποτελεσμάτων (χρήση ORDER). Αναζήτηση στοιχείων και εξαγωγή στατιστικών με ομαδοποίηση των αποτελεσμάτων (χρήση GROUP BY και χρήση GROUP BY .... HAVING). Αναζήτηση στοιχείων που απαιτεί σύνδεση (JOIN) δύο ή περισσότερων πινάκων. Συνδέσεις INNER JOIN, JOIN, OUTER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN. Σύνδεση πίνακα με τον εαυτό του. Σύνδεση πινάκων και χρήση τελεστών σύγκρισης.* ***Χρήσιμες συναρτήσεις και η συνηθισμένη σύνταξή τους.*** *Αναφορά σε συναρτήσεις στο προϊόν της Oracle. Συναρτήσεις NVL, POWER, ROUND, TRUNC, DECOD, SUBSTR, UPPER, LOWER, LENGTH, INSTR, AVG, SUM, MAX, MIN, COUNT, COUNT(*). Αντίστοιχες συναρτήσεις στο προϊόν MySQL. Σύγκριση Oracle DECODE, CASE mySQL.* ***Αναζήτηση με τους γνωστούς από τη θεωρία συνόλων τελεστές UNION, INTERSECT, MINUS.*** *Επισήμανση ότι η MySQL δεν έχει τελεστές INTERSECT, MINUS και τρόπος αντιμετώπισης του προβλήματος. Ιεραρχική αναζήτηση στην ORACLE. Πώς διαχειριζόμαστε ημερομηνίες στο προϊόν της Oracle..*

Εισαγωγή στη δημιουργία, χρήση και ενημερωσιμότητα όψεων (view). *Δημιουργία view βασιζόμενης σε έναν ή περισσότερους πίνακες*

Προαπαιτούμενη γνώση
Κεφάλαιο 1 Εισαγωγή στις έννοιες της τεχνολογίας βάσεων δεδομένων του παρόντος συγγράμματος.

3.1 Εισαγωγή στη γλώσσα SQL

Η γλώσσα διαχείρισης Βάσεων Δεδομένων SQL-Structured Query Language γνωρίζει εξαιρετικά μεγάλη διάδοση εδώ και πολλά χρόνια Ο αναγνώστης που ενδιαφέρεται για τα ιστορικά στοιχεία, τις προσπάθειες προτυποποίησης αλλά και για μια "κριτική" θεώρηση της γλώσσας, και την επισήμανση δυνατών σημείων και αδυναμιών της, παραπέμπεται στο βιβλίο:

C.J.Date (1987) A guide to the SQL standard, Addison-Wesley, ISBN:978-0-201-05777-5

Όλοι οι μεγάλοι διεθνείς οργανισμοί εκπόνησης προτύπων, American National Standards Institute (ANSI), International Organization for Standardization / International Electrotechnical Commission (ISO/IEC) και γνωστοί κατασκευαστές Προϊόντων Διαχείρισης Βάσεων Δεδομένων, όπως οι εταιρείες Oracle, Microsoft, IBM κ.λπ. δέχονται τη γλώσσα SQL ως πρότυπη γλώσσα (standard language) για τις σχεσιακές βάσεις δεδομένων και συμμετέχουν ενεργά στις επιτροπές, γνωστές ως "SQL standards committees", για την εξέλιξής της.

Βλέπε σχετικά και τους συνδέσμους (links):

Oracle, Database SQL Language Reference-SQL Standards

http://docs.oracle.com/cd/B28359_01/server.111/b28286/intro002.htm

Oracle, Database SQL Language Reference-Oracle and Standard SQL

http://docs.oracle.com/cd/B28359_01/server.111/b28286/ap_standard_sql.htm#g21788

Το τελευταίο πρότυπο SQL υιοθετήθηκε τον Ιούλιο του 2003 και είναι γνωστό ως "SQL:2003" ή "SQL3. Ο" ενδιαφερόμενος αναγνώστης παραπέμπεται στο πρότυπο:

- ANSI/ISO/IEC 9075:2003, "Database Language SQL"
- Ένα τμήμα του προτύπου, "Part 14, SQL/XML (ISO/IEC 9075-14)" αναθεωρήθηκε το 2006 και είναι γνωστό ως "SQL/XML:2006". Ο ενδιαφερόμενος αναγνώστης παραπέμπεται στο πρότυπο:
- ANSI/ISO/IEC 9075-14:2006, "Database Language SQL", Part 14 ("SQL/XML")
- Το πρότυπο SQL:2011/ISO/IEC 9075:2011 ("Information technology – Database languages – SQL") είναι η έβδομη αναθεώρηση (revision) των προτύπων ISO (1987) και ANSI (1986) για την SQL database query language. Τυπικά έγινε δεκτό το Δεκέμβρη του 2011.

Το πρότυπο SQL2 ήταν το πρότυπο που ακολούθησε η πλειοψηφία των εμπορικών προϊόντων. Το νέο πρότυπο, SQL3, περιλαμβάνει αρκετές επεκτάσεις όπως υποστήριξη εναυσμάτων (triggers), περιορισμών (constraints) κτλ. Το πρότυπο υποστηρίζει κάποιες αντικειμενοστρεφείς επεκτάσεις που «γεφυρώνουν» σχεσιακές βάσεις με αντικειμενοστρεφή προγραμματισμό κ.λπ. (βλέπε πχ. Ullman and Widom, “A first course in database systems”, Prentice-Hall, 1997). Για τη γλώσσα απαντώνται κάποιες διαφορετικές "διάλεκτοι" ανάλογα με τις επιμέρους προσθήκες και επεκτάσεις που υιοθετεί ο κάθε κατασκευαστής ΣΔΒΔ . Μάλιστα υπάρχουν περιπτώσεις όπου ο ίδιος κατασκευαστής έχει διαφορετική υλοποίηση και υποστήριξη της γλώσσας από προϊόν σε προϊόν πχ. υπάρχουν κάποιες διαφορές Microsoft Access και SQL Server αναφορικά με την SQL. Το ίδιο ανάμεσα σε Oracle, MySQL.

Το υποχρεωτικό μέρος (mandatory portion) του "SQL:2003" είναι γνωστό ως "Core SQL:2003" και υπάρχει στα "SQL:2003 Part 2 (Foundation), Part 11 (Schemata) ".

3.1.1 Πρότυπα ANSI για SQL

Τα παρακάτω έγγραφα του Ινστιτούτου American National Standards Institute (ANSI) αναφέρονται στη γλώσσα SQL:

1) ANSI/ISO/IEC 9075-1:2003, Information technology—Database languages—SQL—Part 1: Framework (SQL/Framework)
2) ANSI/ISO/IEC 9075-2:2003, Information technology—Database languages—SQL—Part 2: Foundation (SQL/Foundation)
3) ANSI/ISO/IEC 9075-3:2003, Information technology—Database languages—SQL—Part 3: Call-Level Interface (SQL/CLI)
4) ANSI/ISO/IEC 9075-4:2003, Information technology—Database languages—SQL—Part 4: Persistent Stored Modules (SQL/PSM)
5) ANSI/ISO/IEC 9075-9:2003, Information technology—Database languages—SQL—Part 9: Management of External Data (SQL/MED)
6) ANSI/ISO/IEC 9075-10:2003, Information technology—Database languages—SQL—Part 10: Object Language Bindings (SQL/OLB)
7) ANSI/ISO/IEC 9075-11:2003, Information technology—Database languages—SQL—Part 11: Information and Definition Schemas (SQL/Schemata)
8) ANSI/ISO/IEC 9075-13:2003, Information technology—Database languages—SQL—Part 13: SQL Routines and Types using the Java Programming Language (SQL/JRT)
9) ANSI/ISO/IEC 9075-14:2005, Information technology—Database languages—SQL—Part 14: XML-Related Specifications (SQL/XML)

Στη συνέχεια θα εξετάσουμε τη γλώσσα όπως υλοποιείται (υποστηρίζεται) στα γνωστά σχεσιακά προϊόντα διαχείρισης βάσης δεδομένων (ORACLE κ.λπ.).

Η γλώσσα SQL ως γλώσσα διαχείρισης βάσης δεδομένων περιλαμβάνει τρεις υπογλώσσες:

- τη γλώσσα ορισμού δεδομένων - DDL (Data Definition Language),
- τη γλώσσα χειρισμού δεδομένων - DML (Data Manipulation Language),
- τη γλώσσα ελέγχου δεδομένων - DCL (Data Control Language).

Στο σημείο αυτό κρίνεται σκόπιμο να υπενθυμίσουμε ότι κάθε σχεσιακό προϊόν διαχείρισης βάσης δεδομένων εκτός από την υλοποίηση της γλώσσας SQL προσφέρει Γεννήτριες εφαρμογών (application generators), εκτυπώσεων (report generators) κ.λπ.

Στις επόμενες ενότητες δίνονται κανόνες για τη χρήση της SQL δήλωσης (SQL statement) αναζήτησης – SELECT- που είναι το βασικότερο τμήμα της υπογλώσσας χειρισμού δεδομένων, κανόνες για τη χρήση δηλώσεων CREATE που αφορούν τη δημιουργία της βάσης δεδομένων, τη δημιουργία δεικτών-ευρετηρίων (indexes), όψεων (views) κ.λπ., κανόνες για τις δηλώσεις INSERT, UPDATE, DELETE που δίνουν τη δυνατότητα εισαγωγής, ενημέρωσης, διαγραφής στοιχείων και τέλος κανόνες για τις δηλώσεις GRANT, REVOKE που εξασφαλίζουν την ασφάλεια του συστήματος. Οι κανόνες συνοδεύονται από παραδείγματα και Case Studies. Ακολουθεί η συνηθισμένη σύνταξη των δηλώσεων της γλώσσας SQL. Χρησιμοποιείστε την επόμενη παράγραφο σαν σημείο αναφοράς και ελέγχου αφού διαβάσετε τους πρακτικούς κανόνες σύνταξης των δηλώσεων SQL.

3.1.2 Σύνταξη των δηλώσεων της γλώσσας SQL

Η γλώσσα SQL έχει δηλώσεις για τον ορισμό δεδομένων (data definition SQL statements), δηλώσεις για την επεξεργασία και την αναζήτηση δεδομένων (data manipulation and retrieval SQL statements) και δηλώσεις ελέγχου δεδομένων (data control SQL statements).

Ο συμβολισμός που ακολουθείται είναι ο εξής:

- Με κεφαλαία γράφουμε τις δεσμευμένες λέξεις (reserved keywords) της γλώσσας SQL και με πεζά τα ονόματα μεταβλητών κ.λπ. που καθορίζει ο προγραμματιστής.
- Μια υποπρόταση σε [ ] είναι προαιρετική.
- Η χρήση / σημαίνει ότι μπορούμε να διαλέξουμε ένα μόνο από αυτά που χωρίζονται με /.
- Η χρήση { ... / ... / κτλ } σημαίνει ότι πρέπει υποχρεωτικά να διαλέξουμε ένα από αυτά που οριοθετούνται από σύμβολα / .
- Οι συντομογραφίες expr και spec σημαίνουν αντίστοιχα expression (έκφραση ή παράσταση) και specification (προδιαγραφή).
- Η λέξη alias σημαίνει συνώνυμο για το όνομα ενός πίνακα
- Τέλος, ότι είναι σε (...) είναι υποχρεωτικό.

3.1.2.1 Συνηθισμένη σύνταξη δηλώσεων ορισμού δεδομένων

Παραθέτουμε τη συνηθισμένη σύνταξη:

```
CREATE TABLE table (column spec[NOT NULL]
   [, column spec[NOT NULL] ... ] )
  [CLUSTER cluster (column [, column] ...) ]
[As query] ;

CREATE [UNIQUE] INDEX indexname
  ON table (column [ASC/DESC] [, column [ASC/DESC] ] ... ) ;

CREATE VIEW view-name [ (alias [, alias] ... ) ]
  As query
  [WITH CHECK OPTION] ;

ALTER TABLE table
  ADD ( column spec [NULL/NOT NULL]
   [, column spec [NULL/NOT NULL] ] ... ) ;

    ALTER TABLE table
  MODIFY( column [spec] [NULL/NOT NULL]
 [, column [spec] [NULL/NOT NULL] ] ... ) ;

COMMENT ON {TABLE table/COLUMN table.column} IS comment-characters;

DROP { CLUSTER cluster / INDEX index [ON TABLE] / TABLE table / VIEW view} ;

RENAME old-name  TO new-name;

CREATE CLUSTER cluster ( column spec [, column spec] ... ) ;
```

3.1.2.2 Συνηθισμένη σύνταξη δηλώσεων επεξεργασίας και αναζήτησης δεδομένων

```
INSERT
  INTO  table [( column [, column ] ... ) ]
  VALUES  (value [, value ] ...) ;

INSERT
  INTO  table [ ( column [, column] ... ) ]
  query ;

UPDATE  table [ alias ]
  SET  column = expression [, column = expression ] ...
  [WHERE condition] ;

UPDATE  table [ alias ]
  SET  (column [, column] ... ) = (query)
  [WHERE condition ] ;

DELETE
  FROM  table
  [WHERE condition] ;

SELECT [ ALL/DISTINCT ] { [table.] * / expression [ alias ], ... }
  FROM  table [ alias ] [, table [ alias ] ] ...
  [WHERE condition]
  [GROUP BY expr [, expr] ... [HAVING condition] ]
  [ORDER BY expr [ASC/DESC] [, expr [ASC/DESC] ] ... ]
  [ { UNION / INTERSECT / MINUS } query ] ;
```

Προσοχή! Σε κάποια προϊόντα, π.χ. MySQL, δεν υποστηρίζονται INTERSECT, MINUS.

3.1.2.3 Συνηθισμένη σύνταξη δηλώσεων ελέγχου δεδομένων

```
GRANT CONNECT [, RESOURCE] [, DBA]
  TO user [, user] ...
  IDENTIFIED BY password [, password] ... ;

GRANT {privilege [, privilege] ... / ALL}
  ON table
  TO {user [,user] ... / PUBLIC}
  [WITH GRANT OPTION] ;

REVOKE [CONNECT] [, RESOURCE] [,DBA]
  FROM user [,user] ... ;

REVOKE {privilege [,privilege] ... / ALL}
  ON  table
  FROM {user [, user] ... / PUBLIC } ;
```

3.2 Αρχή με παράδειγμα υλοποίησης σχεσιακής βάσης δεδομένων

Στην Εικόνα 3.1 βλέπουμε απόσπασμα δεδομένων (sample of data) μιας απλοποιημένης βάσης δεδομένων προσωπικού εταιρείας. Η βάση περιλαμβάνει δύο πίνακες, τον πίνακα emp (πίνακας υπαλλήλων) με στήλες Empno-κωδικός υπαλλήλου, Ename-όνομα, Job-θέση, Hiredate- ημερομηνία πρόσληψης, Mgr-μάνατζερ, Sal-

μισθός, Comm-προμήθεια, Deptno-κωδικός τμήματος υπαλλήλου, και τον πίνακα dept (πίνακας τμημάτων) με στήλες Deptno-κωδικός τμήματος, Dname-όνομα, Loc-έδρα τμήματος.

emp (πίνακας υπαλλήλων)

Empno	Ename	Job	Hiredate	Mgr	Sal	Comm	Deptno
10	CODD	ANALYST	1/1/1989	15	3000		10
15	ELMASRI	ANALYST	2/5/1995	15	1200	150	10
20	NAVATHE	SALESMAN	7/7/1977	20	2000		20
30	DATE	PROGRAMMER	4/5/2004	15	1800	200	10

dept (πίνακας τμημάτων)

Deptno	Dname	Loc
10	ACCOUNTING	ATHENS
20	SALES	LONDON
30	RESEARCH	ATHENS
40	PAYROLL	LONDON

Εικόνα 3.1 Απόσπασμα δεδομένων μιας βάσης δεδομένων προσωπικού εταιρείας. Παρατηρήστε το μορφότυπο στις ημερομηνίες (HH/MM/EEEE).

3.2.1 Εισαγωγή στη χρήση της γλώσσας SQL του προϊόντος της Oracle

Ακολουθεί σειρά δηλώσεων για τη δημιουργία της βάσης δεδομένων στο προϊόν της Oracle, Αρχικά θα δημιουργήσετε τη βάση και τους δύο πίνακες και θα εισάγετε στοιχεία.

```
CREATE TABLE DEPT(DEPTNO NUMBER(2) NOT NULL,
 DNAME VARCHAR2(14), LOC VARCHAR2(14));

CREATE TABLE EMP(EMPNO NUMBER(4) NOT NULL, ENAME VARCHAR2(10),
JOB VARCHAR2(25), HIREDATE DATE, MGR NUMBER(4),
SAL NUMBER(7,2), COMM NUMBER(7,2),
DEPTNO NUMBER(2));

INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');

INSERT INTO EMP
 VALUES (10, 'CODD', 'ANALYST', '01/01/1989', 15, 3000, NULL, 10);

SELECT * FROM EMP;
SELECT * FROM DEPT;

DROP TABLE EMP;
DROP TABLE DEPT;
```

3.2.2 Εισαγωγή στη χρήση της γλώσσας SQL του προϊόντος mySQL

Ακολουθεί σειρά δηλώσεων για τη δημιουργία της βάσης δεδομένων στο προϊόν της mySQL, Αρχικά θα δημιουργήσετε τη βάση και τους δύο πίνακες και θα εισάγετε στοιχεία

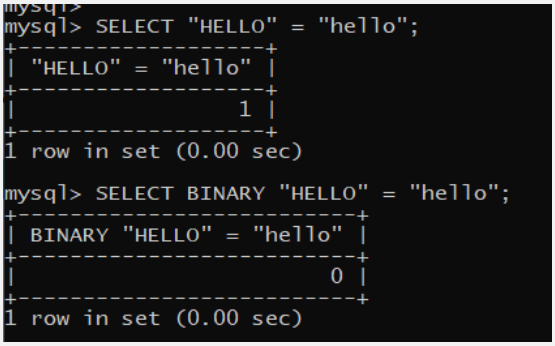
```
DROP DATABASE NEW_PERSONNEL;

CREATE DATABASE new_personnel;
```

```
USE new_personnel;

CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL,
 DNAME VARCHAR(14), LOC VARCHAR(14));

CREATE TABLE EMP(EMPNO INT(4) NOT NULL,
ENAME VARCHAR(10), JOB VARCHAR(25),
HIREDATE DATE, MGR INT(4),  SAL FLOAT(7,2), COMM FLOAT(7,2),
DEPTNO INT(2));

SHOW TABLES;

INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');

INSERT INTO EMP
 VALUES (10, 'CODD', 'ANALYST', '1989/01/01', 15, 3000, NULL, 10);

SELECT * FROM EMP;
SELECT * FROM DEPT;
```

Ακολουθεί ο συγκριτικός Πίνακας 3.1 όπου επισημαίνονται οι διαφορές στην υλοποίηση βάσης δεδομένων στα δύο γνωστά προϊόντα MySQL, Oracle.

Πίνακας 3.1 Διαφορές στις δηλώσεις SQL στα γνωστά προϊόντα MySQL, Oracle

mySQL	Oracle
CREATE DATABASE new_personnel;	
USE new_personnel;	
CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL, DNAME VARCHAR(14), LOC VARCHAR(14));	CREATE TABLE DEPT(DEPTNO NUMBER(2) NOT NULL, DNAME VARCHAR2(14), LOC VARCHAR2(14));
CREATE TABLE EMP(EMPNO INT(4) NOT NULL, ENAME VARCHAR(10), JOB VARCHAR(25), HIREDATE DATE, MGR INT(4), SAL FLOAT(7,2), COMM FLOAT(7,2), DEPTNO INT(2));	CREATE TABLE EMP(EMPNO NUMBER(4) NOT NULL, ENAME VARCHAR2(10), JOB VARCHAR2(25), HIREDATE DATE, MGR NUMBER(4), SAL NUMBER(7,2), COMM NUMBER(7,2), DEPTNO NUMBER(2));
INSERT INTO DEPT(DEPTNO, DNAME, LOC) VALUES (10, 'ACCOUNTING', 'NEW YORK'); INSERT INTO EMP VALUES (10, 'CODD', 'ANALYST', '1989/01/01', 15, 3000, NULL, 10);	INSERT INTO DEPT(DEPTNO, DNAME, LOC) VALUES (10, 'ACCOUNTING', 'NEW YORK'); INSERT INTO EMP VALUES (10, 'CODD', 'ANALYST', '01/01/1989', 15, 3000, NULL, 10);
SELECT * FROM EMP; SELECT * FROM DEPT;	SELECT * FROM EMP; SELECT * FROM DEPT;
DROP TABLE EMP; DROP TABLE DEPT;	DROP TABLE EMP; DROP TABLE DEPT;
DROP DATABASE NEW_PERSONNEL;	

SHOW TABLES;	SELECT * FROM Tab;
Οι παρακάτω δηλώσεις επιστρέφουν το ίδιο αποτέλεσμα, δηλαδή δείχνουν όλους τους αναλυτές του πίνακα emp, όπως και αν είναι γραμμένη η τιμή της θέσης (job): SELECT * FROM emp WHERE job=”ANALYST”; SELECT * FROM emp WHERE job=”analyst”;	Έχει σημασία πως γράφουμε τις τιμές σε δήλωση SELECT. Οι παρακάτω δηλώσεις δεν επιστρέφουν το ίδιο αποτέλεσμα, SELECT * FROM emp WHERE job=‘ANALYST’; SELECT * FROM emp WHERE job=‘analyst’; Αν η τιμή της θέσης (job) για ένα υπάλληλο είναι αποθηκευμένη στη βάση ως ‘ANALYST’ τότε η δεύτερη δήλωση δεν τον δείχνει.
Δηλαδή βλέπουμε ότι στο προϊόν MySQL πραγματοποιούμε μια σύγκριση χωρίς διάκριση πεζών-κεφαλαίων (case-insensitive comparison). Μπορούμε να διαφοροποιήσουμε δύο συμβολοσειρές που είναι γραμμένες με κεφαλαία και πεζά γράμματα και στο προϊόν MySQL χρησιμοποιώντας τη συνάρτηση binary. Στο παράδειγμά μας δεξιά το προϊόν MySQL εκτελεί μια σύγκριση byte-byte των συμβολοσειρών "HELLO" και "hello" και επιστρέφει 0 επειδή δεν είναι ισοδύναμες.	

3.2.3 Σενάριο χρήσης (use case). Δημιουργία βάσης δεδομένων με κύρια και ξένα κλειδιά στο προϊόν mySQL

Μας αναθέτουν να κατασκευάσουμε βάση δεδομένων προσωπικού για την εταιρεία που εργαζόμαστε. Η βάση μας θα έχει δύο πίνακες, έναν πίνακα για τους υπαλλήλους και έναν πίνακα για τα τμήματα. Τα στοιχεία που θα γράψουμε στους πίνακες είναι: EMPNO = κωδικός υπαλλήλου, ENAME = όνομα υπαλλήλου, JOB = θέση, HIREDATE = ημερομηνία πρόσληψης, SAL = μισθός, COMM = προμήθεια, DEPTNO = κωδικός τμήματος, DNAME = όνομα τμήματος, LOC = έδρα τμήματος.

Έστω ότι στη βάση δεδομένων του προσωπικού της εταιρείας θέλουμε να εισάγουμε τα στοιχεία που βλέπουμε στην Εικόνα 3.2,

Πίνακας τμημάτων εταιρείας

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Πίνακας υπαλλήλων εταιρείας

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17	800.00	NULL	20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	2002-02-01	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981-12-24	2975.00	NULL	20
7654	MARTIN	SALESMAN	7698	1981-10-28	1250.00	1400.00	30
7698	BLAKE	MANAGER	7839	2001-05-02	2850.00	NULL	30
7782	CLARK	MANAGER	7839	1981-11-27	2450.00	NULL	10
7788	SCOTT	ANALYST	7566	1987-04-29	3000.00	NULL	20
7839	KING	PRESIDENT	NULL	1987-11-12	5000.00	NULL	10
7844	TURNER	SALESMAN	7698	2007-10-19	1500.00	0.00	30
7876	ADAMS	CLERK	7788	2003-05-07	1100.00	NULL	20
7900	JAMES	CLERK	7698	2003-12-12	950.00	NULL	30
7902	FORD	ANALYST	7566	2003-12-19	3000.00	NULL	20
7934	MILLER	CLERK	7782	2003-01-19	1300.00	NULL	10
7999	BATES	ANALYST	7566	2004-01-04	1300.00	NULL	NULL

Εικόνα 3.2 Απόσπασμα δεδομένων μίας βάσης δεδομένων προσωπικού εταιρείας. Οι πίνακες της βάσης φαίνονται στο προϊόν mySQL. Παρατηρήστε το μορφότυπο των ημερομηνιών (EEEE/MM/HH)

Παραθέτουμε τις σχετικές δηλώσεις SQL.

Δημιουργία βάσης και πινάκων.

```
CREATE DATABASE personnel;
USE personnel;
CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL,
 DNAME VARCHAR(14), LOC VARCHAR(14),
 PRIMARY KEY(DEPTNO));

CREATE TABLE EMP(EMPNO INT(4) NOT NULL,
ENAME VARCHAR(10), JOB VARCHAR(9),
MGR INT(4), HIREDATE DATE,
SAL FLOAT(7,2), COMM FLOAT(7,2),
DEPTNO INT(2),
 PRIMARY KEY(EMPNO),
 FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO));
```

Πως βλέπουμε τους πίνακες της βάσης δεδομένων

```
SHOW TABLES;
```

Πως βλέπουμε τη δομή των πινάκων

```
DESCRIBE dept;
DESCRIBE emp;
```

Εισαγωγή δεδομένων

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (30, 'SALES', 'CHICAGO');
```

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (40, 'OPERATIONS', 'BOSTON');

INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
 VALUES (7369, 'SMITH', 'CLERK', 7902, '1980/12/17', 800, NULL, 20);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
  VALUES (7499, 'ALLEN', 'SALESMAN', 7698, '1981/02/20', 1600, 300, 30);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
  VALUES (7521, 'WARD', 'SALESMAN', 7698, '2002/02/01', 1250, 500, 30);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
  VALUES (7566, 'JONES', 'MANAGER', 7839, '1981/12/24', 2975, NULL, 20);
INSERT INTO EMP
   (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
   VALUES (7654, 'MARTIN', 'SALESMAN', 7698, '1981/10/28', 1250, 1400, 30);
INSERT INTO EMP
    (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7698, 'BLAKE', 'MANAGER', 7839, '2001/05/02', 2850, NULL, 30);
INSERT INTO EMP
(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
   VALUES (7782, 'CLARK', 'MANAGER', 7839, '1981/11/27', 2450, NULL, 10);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
 VALUES (7788, 'TT', 'ANALYST', 7566, '1987/04/29', 3000, NULL, 20);
INSERT INTO EMP
    (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7839, 'KING', 'PRESIDENT', NULL, '1987/11/12', 5000, NULL, 10);
INSERT INTO EMP
    (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7844, 'TURNER', 'SALESMAN', 7698, '2007/10/19', 1500, 0, 30);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7876, 'ADAMS', 'CLERK', 7788, '2003/05/07', 1100, NULL, 20);
INSERT INTO EMP
  (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7900, 'JAMES', 'CLERK', 7698, '2003/12/12', 950, NULL, 30);
INSERT INTO EMP
    (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7902, 'FORD', 'ANALYST', 7566, '2003/12/19', 3000, NULL, 20);
INSERT INTO EMP
  (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7934, 'MILLER', 'CLERK', 7782, '2003/01/19', 1300, NULL, 10);
INSERT INTO EMP
  (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7999, 'BATES', 'ANALYST', 7566, '2004/01/04', 1300, NULL, NULL);

SELECT * FROM EMP;
SELECT * FROM DEPT;
```

Πίνακας τμημάτων εταιρείας

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Πίνακας υπαλλήλων εταιρείας

EMPNO	ENAME	JOB	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	17/12/80	800	-	20
7499	ALLEN	SALESMAN	20/02/81	1600	300	30
7521	WARD	SALESMAN	22/02/81	1250	500	30
7566	JONES	MANAGER	02/04/81	2975	-	20
7654	MARTIN	SALESMAN	28/10/81	1250	1400	30
7698	BLAKE	MANAGER	01/05/81	2850	-	30
7782	CLARK	MANAGER	09/06/81	2450	-	10
7788	SCOTT	ANALYST	19/04/87	3000	-	20
7839	KING	PRESIDENT	17/11/81	5000	-	10
7844	TURNER	SALESMAN	08/10/81	1500	0	30
7876	ADAMS	CLERK	23/05/87	1100	-	20
7900	JAMES	CLERK	03/12/81	950	-	30
7902	FORD	ANALYST	03/12/81	3000	-	20
7934	MILLER	CLERK	23/01/82	1300	-	10
7999	BATES	ANALYST	23/01/04	1300	-	-

Εικόνα 3.3 Οι πίνακες της βάσης φαίνονται στο προϊόν της ORACLE.

3.2.4 Δημιουργία βάσης δεδομένων με κύρια και ξένα κλειδιά στο προϊόν Oracle.

Έστω η βάση δεδομένων προσωπικού εταιρείας της Εικόνας 3.3.

Δημιουργία πινάκων με κύρια και ξένα κλειδιά

```
CREATE TABLE DEPT(DEPTNO NUMBER(2) NOT NULL,
  DNAME CHAR(14), LOC CHAR(14),
  PRIMARY KEY(DEPTNO));

CREATE TABLE EMP(EMPNO NUMBER(4) NOT NULL,
   ENAME CHAR(10), JOB CHAR(9),
   HIREDATE DATE,
   SAL NUMBER(7,2), COMM NUMBER(7,2),
   DEPTNO NUMBER(2),
   PRIMARY KEY(EMPNO),
   FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO));
```

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
  VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
  VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
  VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
  VALUES (40, 'OPERATIONS', 'BOSTON');
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
   VALUES (7369, 'SMITH', 'CLERK', '17/12/1980', 800, NULL, 20);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
   VALUES (7499, 'ALLEN', 'SALESMAN', '20/02/1981', 1600, 300, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
   VALUES (7521, 'WARD', 'SALESMAN', '22/02/1981', 1250, 500, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
   VALUES (7566, 'JONES', 'MANAGER', '02/04/1981', 2975, NULL, 20);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
   VALUES (7654, 'MARTIN', 'SALESMAN', '28/10/1981', 1250, 1400, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
   VALUES (7698, 'BLAKE', 'MANAGER', '01/05/1981', 2850, NULL, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
   VALUES (7782, 'CLARK', 'MANAGER', '09/06/1981', 2450, NULL, 10);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7788, 'SCOTT', 'ANALYST', '19/04/1987', 3000, NULL, 20);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7839, 'KING', 'PRESIDENT', '17/11/1981', 5000, NULL, 10);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7844, 'TURNER', 'SALESMAN', '08/10/1981', 1500, 0, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7876, 'ADAMS', 'CLERK', '23/05/1987', 1100, NULL, 20);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7900, 'JAMES', 'CLERK', '03/12/1981', 950, NULL, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7902, 'FORD', 'ANALYST', '03/12/1981', 3000, NULL, 20);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7934, 'MILLER', 'CLERK', '23/01/1982', 1300, NULL, 10);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7999, 'BATES', 'ANALYST', '23/01/2004', 1300, NULL, NULL);
```

3.3 Πως ορίζουμε τους πίνακες βάσης δεδομένων και πως εισάγουμε στοιχεία με τη γλώσσα SQL

Η γλώσσα διαχείρισης βάσεων δεδομένων SQL, όπως αναφέρθηκε, περιλαμβάνει τρεις υπογλώσσες:

1) Τη γλώσσα ορισμού δεδομένων - DDL (data definition language)
2) Τη γλώσσα χειρισμού δεδομένων - DML (data manipulation language)
3) Τη γλώσσα ελέγχου δεδομένων - DCL (data control language)

Στη συνέχεια θα ενδιαφερθούμε για την υπογλώσσα DDL και την υπογλώσσα DML δηλαδή για τις δηλώσεις (SQL statements) CREATE, SELECT, INSERT, UPDATE, DELETE.

3.3.1 Τι μπορεί να κάνει ο χρήστης με τη Γλώσσα Ορισμού Δεδομένων (DDL)

Η DDL ως (υπο)γλώσσα της γλώσσας SQL δίνει στον προγραμματιστή τη δυνατότητα να ορίσει πίνακες (tables), ευρετήρια (δείκτες, indexes), όψεις (views), εναύσματα (triggers) και στη συνέχεια να τροποποιήσει ή να διαγράψει αυτούς τους ορισμούς.

Προσοχή! Τροποποιήσεις και διαγραφές δεν αφορούν μόνο ορισμούς!

3.3.2 Πώς γράφονται τα ονόματα πινάκων, στηλών, δεικτών και όψεων

Για να είναι νόμιμο ένα όνομα στα περισσότερα προϊόντα διαχείρισης βάσεων δεδομένων ακολουθούμε τις παρακάτω βασικές αρχές:

- Αρχίστε τα ονόματα με γράμμα
- Χρησιμοποιήστε γράμματα του λατινικού αλφαβήτου, αριθμούς και τον χαρακτήρα "_"
- Υπάρχουν κάποια προϊόντα που δέχονται στα ονόματα και γράμματα του ελληνικού αλφαβήτου, π.χ., MS-ACCESS
- Θυμηθείτε ότι υπάρχουν περιορισμοί στο μήκος των ονομάτων. Σε πολλά προϊόντα μπορείτε να χρησιμοποιήσετε μέχρι 30 χαρακτήρες
- Μη χρησιμοποιείτε λέξεις κλειδιά της SQL. Για παράδειγμα, δν μπορείτε να ονομάσετε έναν πίνακα με το όνομα SELECT.

3.3.3 Ποιοί τύποι δεδομένων χρησιμοποιούνται συνήθως.

Υπάρχουν διαφορές ανάμεσα στους τύπους δεδομένων που υποστηρίζουν γνωστά προϊόντα διαχείρισης βάσεων δεδομένων. Το προϊόν της MySQL υποστηρίζει τρεις τύπους δεδομένων: TEXT (Πίνακας 3.2), NUMBER (Πίνακας 3.3), DATE (Πίνακας 3.4). Δείτε σχετικά και την ιστοσελίδα MySQL Data Types. https://www.w3schools.com/sql/sql_datatypes.asp

Πίνακας 3.2 Τύποι δεδομένων Text στο προϊόν MySQL

Data type	Περιγραφή
CHAR(size)	Διαχειρίζεται συμβολοσειρές σταθερού μήκους (fixed length string) μέχρι 255 characters
VARCHAR(size)	Διαχειρίζεται συμβολοσειρές μεταβλητού μήκους (variable length string) μέχρι 255 characters. Ο μέγιστος αριθμός καθορίζεται στην παρένθεση. Αν εκχωρηθεί μεγαλύτερη τιμή από 255 μετατρέπεται σε TEXT type
TINYTEXT	Διαχειρίζεται συμβολοσειρές (string) μέχρι 255 characters
TEXT	Διαχειρίζεται συμβολοσειρές (string) μέχρι 65,535 characters
BLOB	Διαχειρίζεται BLOBs (Binary Large OBjects) μέχρι 65,535 bytes
MEDIUMTEXT	Διαχειρίζεται συμβολοσειρές (string) μέχρι 16,777,215 characters
MEDIUMBLOB	Διαχειρίζεται BLOBs (Binary Large OBjects) μέχρι bytes
LONGTEXT	Διαχειρίζεται συμβολοσειρές (string) μέχρι 4,294,967,295 characters
LONGBLOB	Διαχειρίζεται BLOBs (Binary Large OBjects) μέχρι 4,294,967,295 bytes
ENUM(x,y,z,etc.)	Επιτρέπει την εισαγωγή λίστας τιμών. Η λίστα ENUM περιλαμβάνει μέχρι 65535 τιμές. Αν εισαχθεί μία τιμή εκτός λίστας τότε στην πραγματικότητα θα εισαχθεί κενή τιμή (blank value).

	Οι τιμές ταξινομούνται με την σειρά εισαγωγής. Η εισαγωγή τιμών γίνεται με το μορφότυπο (format) ENUM('X','Y','Z')
SET	Παρόμοιος τύπος με τον τύπο ENUM με τον περιορισμό ότι περιλαμβάνει μέχρι 64 list items και αποθηκεύει περισσότερες από μία επιλογές

Πίνακας 3.3 Τύποι δεδομένων Number στο προϊόν MySQL

Data type	Περιγραφή
TINYINT(size)	-128 έως 127. 0 έως το 255 για μη προσημασμένους αριθμούς (UNSIGNED). Ο μέγιστος αριθμός καθορίζεται στην παρένθεση.
SMALLINT(size)	-32768 έως 32767. 0 έως 65535 για μη προσημασμένους αριθμούς (UNSIGNED). Ο μέγιστος αριθμός καθορίζεται στην παρένθεση.
MEDIUMINT(size)	-8388608 έως 8388607. 0 έως 16777215 για μη προσημασμένους αριθμούς (UNSIGNED). Ο μέγιστος αριθμός καθορίζεται στην παρένθεση.
INT(size)	-2147483648 έως 2147483647. 0 έως 4294967295 για μη προσημασμένους αριθμούς (UNSIGNED). Ο μέγιστος αριθμός καθορίζεται στην παρένθεση.
BIGINT(size)	-9223372036854775808 έως 9223372036854775807. 0 έως 18446744073709551615 για μη προσημασμένους αριθμούς (UNSIGNED). Ο μέγιστος αριθμός καθορίζεται στην παρένθεση.
FLOAT(size,d)	Μικροί πραγματικοί αριθμοί με υποδιαστολή (floating decimal point). Ο μέγιστος ακέραιος αριθμός καθορίζεται στην παρένθεση από την παράμετρο size και το δεκαδικό μέρος από την παράμετρο d.
DOUBLE(size,d)	Μεγάλοι πραγματικοί αριθμοί με υποδιαστολή (floating decimal point). Ο μέγιστος ακέραιος αριθμός καθορίζεται στην παρένθεση από την παράμετρο size parameter και το δεκαδικό μέρος από την παράμετρο d.
DECIMAL(size,d)	Αριθμός DOUBLE αποθηκευμένος σαν ορμαθός (string).

Πίνακας 3.4 Τύποι δεδομένων Date στο προϊόν MySQL

Data type	Περιγραφή
DATE()	Ημερομηνία. Format: YYYY-MM-DD Από '1000-01-01' έως '9999-12-31'
DATETIME()	Ημερομηνία και ώρα. Format: YYYY-MM-DD HH:MM:SS Από '1000-01-01 00:00:00' έως '9999-12-31 23:59:59'
TIMESTAMP()	Timestamp. Format: YYYY-MM-DD HH:MM:SS Από '1970-01-01 00:00:01' έως '2038-01-09 03:14:07'
TIME()	time. Format: HH:MM:SS Από '-838:59:59' έως '838:59:59'
YEAR()	Year Από 1901 έως 2155 (four-digit format) Από 70 έως 69 (two-digit format). Αναπαρίστανται έτη από το 1970 έως το 2069

Οι τύποι δεδομένων DATETIME και TIMESTAMP είναι διαφορετικοί. Σε δηλώσεις INSERT, UPDATE η στήλη με τύπο TIMESTAMP αυτόματα λαμβάνει τιμή την τρέχουσα ημερομηνία και ώρα. Επίσης, δέχεται τιμές σε μορφότυπα (formats) όπως YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD, YYMMDD.

Στην περίπτωση του προϊόντος της ORACLE χρησιμοποιούνται πάρα πολλοί τύποι δεδομένων. Στον Πίνακα 3.5 παραθέτουμε ενδεικτικά λίγους βασικούς τύπους. Δείτε και το εγχειρίδιο αναφοράς (reference manual):

Oracle Database SQL Language Reference, 19c, E96310-11, August 2021

Πίνακας 3.5 Παράθεση βασικών τύπων δεδομένων του προϊόντος Oracle

Data type	Περιγραφή
CHAR (size)	μέχρι 2000 bytes
VARCHAR2(size)	μέχρι 4000 bytes
NUMBER	προσημασμένοι δεκαδικοί μέχρι 38 ψηφία
NUMBER (p)	προσημασμένοι δεκαδικοί μέχρι p ψηφία
NUMBER (p, s)	όπως παραπάνω για p ψηφία, τα s δεκαδικά The precision p can range from 1 to 38. The scale s can range from -84 to 127.
DATE	ημερομηνίες από 1/ΙΑΝ/4712 πχ - 31/ΔΕΚ/9999 μχ (January 1, 4712 BC to December 31, 9999 AD)
CLOB	Character Large Object (8 TB έως128 TB)
BLOB	Binary Large Object (8 TB έως128 TB)
BFILE	pointer to binary file on disk (8 TB έως 128 TB)
XML	XML data (έως 4 Gigabytes)

3.3.4 Πώς ορίζουμε πίνακες. Δήλωση CREATE TABLE

‘Εστω ότι θέλουμε να ορίσουμε πίνακες με στοιχεία υπαλλήλου (EMP), τμήματος (DEPT), και έργων (PROJ) στο περιβάλλον SQL*PLUS της ORACLE. Στην περίπτωση αυτή υποτίθεται ότι ένας υπάλληλος ανήκει σε ένα τμήμα και επιπλέον, εργάζεται σε ένα μόνο έργο.

Στη γραμμή εκτέλεσης των δηλώσεων (prompt) του προϊόντος διαχείρισης βάσης δεδομένων γράφουμε τη δήλωση CREATE για κάθε πίνακα. Ακολουθεί παράδειγμα στην περίπτωση της Oracle:

```
SQL> CREATE TABLE DEPT (DEPTNO NUMBER(2),
2  DNAME CHAR(14),
3  LOC CHAR(13));
SQL> CREATE TABLE EMP (EMPNO NUMBER(4) NOT NULL,
2 ENAME CHAR(10),
3 JOB CHAR(9),
4 MGR NUMBER(4),
5 HIREDATE DATE,
6 SAL NUMBER(7,2),
7 COMM NUMBER(7,2),
8 DEPTNO NUMBER(2));
SQL> CREATE TABLE PROJ (PROJNO NUMBER(3) NOT NULL,
2  PNAME CHAR(5),
3  BUDGET NUMBER (7,2));
```

Στα διάφορα προϊόντα διαφέρει η γραμμή εντολών (prompt) του συστήματος, π.χ., **SQL>**, **.SQL**

Προσοχή! Στους ορισμούς των πινάκων υποθέσαμε ότι κάθε υπάλληλος εργάζεται σε ένα μόνο έργο αλλά δεν συμπεριλάβαμε τον κωδικό του έργου στον πίνακα emp. Αυτό σημαίνει ότι δε γνωρίζουμε σε ποιό έργο εργάζεται ο υπάλληλος. Επομένως θα πρέπει να προσθέσετε τη στήλη αυτή στον πίνακα. Επιπλέον, μπορούμε να έχουμε ονόματα υπαλλήλων μόνο μέχρι 10 χαρακτήρες. Ακολουθούν δηλώσεις SQL που τροποποιούν-διορθώνουν τους ορισμούς αυτούς.

Σημείωση

Αν ένας υπάλληλος μπορεί να εργάζεται σε πολλά έργα με κάποιο ποσοστό του χρόνου του θα πρέπει εκτός από τους τρεις παραπάνω πίνακες να ορίσουμε ακόμη έναν πίνακα, ως εξής:

```
CREATE TABLE ASSIGN(EMPNO NUMBER(4) NOT NULL,
   PROJNO NUMBER(3) NOT NULL,
   PTIME NUMBER(3));
```

3.3.5 Πώς τροποποιούμε τον ορισμό πίνακα. Δήλωση ALTER TABLE

Στη συνέχεια υποθέτουμε ότι κάθε υπάλληλος εργάζεται αποκλειστικά σε ένα έργο. Θα προσθέσουμε στον πίνακα EMP τη στήλη PROJNO :

```
SQL> ALTER TABLE EMP ADD (PROJNO NUMBER(3));
```

Τι τιμή έχει η στήλη PROJNO στο πίνακα EMP;

Για να αλλάξουμε στον πίνακα EMP το μήκος της στήλης ENAME:

```
SQL> ALTER TABLE EMP MODIFY (ENAME CHAR(20));
```

3.3.6 Πώς διαγράφουμε πίνακα. Δήλωση DROP TABLE

```
SQL> DROP TABLE EMP;
```

Προσοχή στις επιπτώσεις!

3.3.7 Πώς διαχειριζόμαστε ευρετήρια-δείκτες. Δήλωση CREATE INDEX

Μόλις οριστεί ένα ευρετήριο (δείκτης, index) ενημερώνεται και χρησιμοποιείται αυτόματα από το σύστημα π.χ. ΣΔΒΔ της ORACLE. Επίσης, όταν σχηματίζετε μια δήλωση αναζήτησης, δηλαδή δήλωση SELECT, δεν είναι απαραίτητο να ορίσετε δείκτη για να κάνετε αναζήτηση. Ο δείκτης ορίζεται για να επιταχύνει την απάντηση κάποιων ερωτημάτων. Δείτε σχετικά και Κεφάλαιο 1.

3.3.7.1 Πώς ορίζουμε δείκτες για απλές στήλες πίνακα

Για να ορίσουμε μία στήλη, π.χ., το όνομα υπαλλήλου (ename), σαν δείκτη-ευρετήριο INAME στον πίνακα EMP:

```
SQL> CREATE INDEX INAME ON EMP (ENAME);
```

3.3.7.2 Πώς ορίζουμε δείκτες για συνδυασμό στηλών πίνακα

Για να ορίσουμε δείκτη SALCOM το συνδυασμό Μισθού (SAL), προμήθειας (COMM):

```
SQL> CREATE INDEX SALCOM ON EMP (SAL,COMM);
```

ή

```
SQL> CREATE INDEX SALCOM ON EMP (COMM,SAL);
```

Προσοχή! Οι δύο ορισμοί ευρετηρίων δεν είναι ισοδύναμοι. Αν οι αναζητήσεις συνήθως περιέχουν υποπροτάσεις WHERE, π.χ.., WHERE SAL< 1180 AND SAL> 1160, τότε πρέπει να χρησιμοποιήσετε τον πρώτο ορισμό. Επισημαίνουμε ότι ανάλογα με το προϊόν υπάρχει περιορισμός στον αριθμό στηλών (του πίνακα) που μπορούμε να χρησιμοποιήσουμε για να ορίσουμε δείκτη. Για παράδειγμα, σε κάποια προϊόντα μπορούμε να έχουμε συνδυασμό μέχρι 16 στηλών στο ευρετήριο.

3.3.7.3 Κατάργηση δείκτη – Δήλωση DROP INDEX

```
SQL> DROP INDEX SALCOM ON EMP;
```

3.3.8 Δυνατότητες της υπογλώσσας χειρισμού δεδομενων (DML).

Η υπο-γλώσσα DML είναι μια γλώσσα που δίνει στο χρήστη τη δυνατότητα εισαγωγής, τροποποίησης, διαγραφής και αναζήτησης στοιχείων της ΒΔ.

3.3.8.1 Πώς θα εισάγετε στοιχεία σε πίνακα. Δήλωση INSERT

```
SQL> INSERT INTO EMP
   2   VALUES(7522, ' ΑΝΔΡΕΟΥ Ν.', 'ΚΛΗΤΗΡΑΣ',7890,
   3   '10/11/1998', 1000,NULL,30);
```

Αν δεν επιθυμείτε να εισάγετε όλα τα στοιχεία:

```
 SQL >  INSERT INTO EMP (empno, ename, deptno)
   2  VALUES (69569, 'ΣΚΟΥΡΑΣ', 30 ) ;
```

3.3.8.2 Πώς εισάγουμε στοιχεία ημερομηνίας και ώρας με χρήση της συνάρτησης TO_DATE στο προϊόν της Oracle.

Αρχίζουμε με απλά παραδείγματα χρήσης της συνάρτησης TO_DATE του προϊόντος της Oracle:

```
SELECT TO_DATE('2026/12/10', 'yyyy/mm/dd') FROM DUAL;
SELECT TO_DATE('10122026', 'DDMMYYYY') FROM DUAL;
SELECT TO_DATE('20261210', 'yyyymmdd') FROM DUAL;
```

Η συνάρτηση TO_DATE μετατρέπει ημερομηνία και ώρα στον επιθυμητό μορφότυπο (format) :

DD/MON/YY HH:MI:SS.

Ο πίνακας DUAL είναι ένας μονοθέσιος πίνακας (έχει μία στήλη και μία γραμμή) του ΣΔΒΔ της Oracle και χρησιμοποιείται για να σχηματιστεί ορθή δήλωση SQL. Στο σημείο αυτό μπορούμε να παρουσιάσουμε μία διαφορά ανάμεσα σε ΣΔΒΔ. Στο προϊόν MySQL επιτρέπονται δηλώσεις του τύπου

```
SELECT 15*40; ή και SELECT 15*40 FROM DUAL;
```

ενώ στο προϊόν της Oracle πρέπει να γράψουμε υποχρεωτικά

```
SELECT 15*40 FROM DUAL;
```

Επισημαίνουμε ότι σε Oracle η συνάρτηση TO_DATE μετατρέπει μία τιμή συμβολοσειράς (a string value) σε τιμή τύπου δεδομένων DATE χρησιμοποιώντας τον μορφότυπο (format) που καθορίσαμε στη δήλωση, π.χ.,

‘DDMMYYYY’. Σε MySQL πρέπει να χρησιμοποιήσετε τη συνάρτηση STR_TO_DATE. Σημειώστε ότι στις συναρτήσεις TO_DATE και STR_TO_DATE οι μορφότυποι συμβολοσειρών διαφέρουν.

```
   Oracle: SELECT TO_DATE('2026-02-11', 'YYYY-MM-DD') FROM dual;
   MySQL:  SELECT STR_TO_DATE('2013-02-11', '%Y-%m-%d');
```

Ακολουθούν δηλώσεις εισαγωγής γραμμών λίγο πιο πολύπλοκες.

```
SQL> INSERT INTO EMP
   2 VALUES(7612, 'ΑΝΔΡΕΟΥ', 'ΠΩΛΗΤΗΣ',7890,
   3   TO_DATE('10/11/1988 9:30', 'DD/MM/YY HH:MI'),
   4   1000,NULL,30);
SQL> INSERT INTO EMP
   2 VALUES(7512, 'ΝΙΚΟΥ Ν.', 'ΠΩΛΗΤΗΣ',7890,
   3 TO_DATE('3/4/1987 9:30:15', 'DD-MM-YYYY HH:MI:SS'),
   4   1000,NULL,30);
```

3.3.8.3 Πώς εισάγετε στοιχεία με χρήση αναζήτησης (query) SELECT

Ακολουθεί παράδειγμα στο οποίο «φωλιάζουμε» δήλωση SELECT σε δήλωση INSERT,

Περίπτωση «φωλιασμένης» (Embedded) δήλώσης SELECT: Είναι καλό να επανέλθετε στα παραδείγματα δηλώσεων INSERT/UPDATE/DELETE που περιλαμβάνουν «φωλιασμένη» δήλωση SELECT όταν μελετήσετε περισσότερο τη δήλωση SELECT στη συνέχεια του κεφαλαίου.

Παράδειγμα

Θέλετε να εισάγετε τα στοιχεία των υπαλλήλων σε έναν πίνακα με όνομα Bonus και επιπλέον να δώσετε δώρο για την απόδοση (bonus) σε όλους τους υπαλλήλους που είναι επικεφαλείς έργων και σε υπαλλήλους με προμήθεια μεγαλύτερη του 25% του μισθού τους. Αρχικά ορίζετε τον πίνακα.

```
SQL> CREATE TABLE BONUS (EMPNO NUMBER(4) NOT NULL,
   2  ENAME CHAR(20),
   3  JOB CHAR(9),
   4  SAL NUMBER(7,2),
   5  COMM NUMBER(7,2),
   6  DEPTNO NUMBER(2));
```

Παραθέτουμε και τη δήλωση που δείχνει τα στοιχεία EMPNO, ENAME, JOB, SAL, COMM, DEPTNO των υπαλλήλων που μας ενδιαφέρουν, δηλαδή των υπαλλήλων που είναι επικεφαλείς έργων και των υπαλλήλων με προμήθεια μεγαλύτερη του 25% του μισθού τους.

```
SELECT EMPNO,ENAME,JOB,SAL,COMM,DEPTNO
FROM   EMP
WHERE  JOB = 'MANAGER' OR  COMM > 0.25*SAL;
```

Ακολουθεί η δήλωση INSERT:

```
INSERT INTO BONUS(EMPNO,ENAME,JOB,SAL,COMM,DEPTNO)
SELECT EMPNO,ENAME,JOB,SAL,COMM,DEPTNO
FROM   EMP
WHERE  JOB = 'MANAGER' OR  COMM > 0.25*SAL;
```

Βλέπουμε όλα τα δεδομένα του πίνακα με δήλωση SELECT.

```
SELECT * FROM BONUS;
```

3.3.8.4 Πώς τροποποιούμε στοιχεία. Δήλωση UPDATE

Ακολουθούν παραδείγματα.

Παράδειγμα 1

Θέλετε ο υπάλληλος με όνομα WIDOM να τοποθετηθεί στους πωλητές με 10% αύξηση μισθού. Η πρόσληψη γίνεται σήμερα (SYSDATE).

```
SQL> UPDATE EMP
2    SETJOB='ΠΩΛΗΤΗΣ', HIREDATE = SYSDATE, SAL=1.1*SAL
3    WHERE  ENAME = 'WIDOM';
```

Προσοχή! Αν έχουμε περισσότερους υπάλληλους με το ίδιο όνομα οι μεταβολές θα γίνουν για όλους αυτούς τους υπάλληλους.

Παράδειγμα 2

Θέλετε να δώσετε 15% αύξηση στους αναλυτές και τους πωλητές του Τμήματος 30

```
SQL> UPDATE EMP
2    SET  SAL = SAL*1.15
3    WHERE  (JOB='ΑΝΑΛΥΤΗΣ' OR JOB='ΠΩΛΗΤΗΣ')
4    AND DEPTNO = 30;
```

Παράδειγμα 3

Θέλετε να δώσετε αύξηση 5% σε όλους τους υπαλλήλους που περιλαμβάνονται στον πίνακα BONUS. Δείτε αρχικά το ερώτημα που υπολογίζει τους κωδικούς των υπαλλήλων αυτών.

```
SELECT EMPNO FROM  BONUS;
```

Ακολουθεί η δήλωση.

```
SQL> UPDATE EMP
2    SETSAL = SAL*1.05
3    WHERE  EMPNO IN
4    (SELECT EMPNO
5    FROM  BONUS);
```

Παράδειγμα 4

Θέλετε να εξισώσετε το μισθό όλων των πωλητών με το μέσο μισθό πωλητών αυξημένο κατά 10%. Αρχικά υπολογίζουμε το μέσο μισθό.

```
SELECT 1.1*AVG(SAL) FROM EMP WHERE JOB = 'ΠΩΛΗΤΗΣ';
```

Ακολουθεί η δήλωση.

```
SQL> UPDATE  EMP
2    SET SAL = (SELECT 1.1*AVG(SAL)
3    FROM EMP WHERE JOB = 'ΠΩΛΗΤΗΣ')
4    WHERE   JOB = 'ΠΩΛΗΤΗΣ';
```

Παράδειγμα 5

Θέλετε η διεύθυνση και η ημερομηνία γεννήσεως του υπαλλήλου 7522 να εξισωθούν με τα στοιχεία του υπαλλήλου 30.

```
SQL>  UPDATE  emp
   2  SET (job, hiredate) =
   3 (SELECT job, hiredate
   4   FROM emp
   5   WHERE empno = 30)
   6   WHERE  empno = 7522;
```

3.3.9 Πώς διαγράφουμε στοιχεία - δήλωση DELETE

```
SQL> DELETE FROM EMP
2    WHERE ENAME =  'ΣΚΟΥΡΑΣ';
```

3.3.10 Διαχειριστής βάσεως δεδομένων, προγραμματιστής εφαρμογών βάσεων και η υπογλώσσα ελέγχου δεδομένων (DCL)

Η DCL είναι γλώσσα που δίνει στον χρήστη τη δυνατότητα να ορίσει πότε θα οριστικοποιήθουν ή θα ακυρωθούν οι μεταβολές των δεδομένων, και να καθορίσει τα δικαιώματα των χρηστών στη δημιουργία ή χρήση πινάκων, δεικτών-ευρετηρίων, όψεων κ.λπ. Η γλώσσα χρησιμοποιείται από τους διαχειριστές βάσης δεδομένων, DBA–Data Base Administrator, αλλά και τους προγραμματιστές εφαρμογών σε συνεργασία με τους διαχειριστές. Θα περιγραφεί στο Κεφάλαιο 7 του παρόντος συγγράμματος.

3.4 Πως αναζητούμε στοιχεία με τη γλώσσα SQL. Σύνταξη της δήλωσης SELECT. Πρακτικοί κανόνες

Αρχικά θα παρουσιάσουμε τη βάση δεδομένων που θα χρησιμοποιηθεί στα παραδείγματα αναζήτησης. Πιο συγκεκριμένα, τα περισσότερα παραδείγματα που θα αναφέρουμε στη συνέχεια είναι αναζητήσεις, ερωτήματα SELECT, σε μία απλή βάση δεδομένων που περιλαμβάνει στοιχεία των υπαλλήλων μίας εταιρείας, των τμημάτων της εταιρείας (πχ. Τμήμα Προσωπικού) στα οποία ανήκουν οι υπάλληλοι και τα στοιχεία των έργων (projects) της εταιρείας στα οποία εργάζονται οι υπάλληλοι. Υποθέτουμε ότι οι υπάλληλοι μπορεί να εργάζονται σε ένα ή περισσότερα έργα και σε κάθε έργο εργάζονται υπάλληλοι. Όλα τα στοιχεία είναι οργανωμένα στους τέσσερις πίνακες της εικόνας 3.4 όπου :

EMPNO= κωδικός υπαλλήλου, ENAME= ονοματεπώνυμο υπαλλήλου, JOB= θέση στην εταιρεία, MGR= ο επικεφαλής του υπαλλήλου, SAL= μισθός, COMM = προμήθεια, DEPTNO= κωδικός Τμήματος. Υποτίθεται ότι κάθε υπάλληλος ανήκει σε ένα Τμήμα. DNAME = όνομα Τμήματος, LOC= έδρα Τμήματος, PROJNO= κωδικός έργου, PNAME= περιγραφή έργου, BUDGET= προϋπολογισμός έργου, PTIME= ποσοστό χρόνου απασχόλησης υπαλλήλου σε έργο. Υποτίθεται ότι κάθε υπάλληλος μπορεί να εργάζεται σε περισσότερα από ένα έργα.

Αν θέλετε να δοκιμάσετε τα παρακάτω παραδείγματα στον υπολογιστή, π.χ. στο περιβάλλον SQL*PLUS της ORACLE, θα πρέπει να ορίσετε τη βάση δεδομένων και να εισάγετε στοιχεία. Οι πίνακες δημιουργούνται με τη δήλωση CREATE και "γεμίζουν" με δήλωση INSERT. Οι δηλώσεις αυτές είναι δηλώσεις της γλώσσας SQL.

(πίνακας στοιχείων υπαλλήλου) EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO

(πίνακας στοιχείων Τμημάτων στις οποίες ανήκουν οι υπάλληλοι) DEPT

DEPTNO	DNAME	LOC

(πίνακας στοιχείων έργων της εταιρείας) PROJ

PROJNO	PNAME	BUDGET

(πίνακας απασχόλησης υπαλλήλων σε έργα της εταιρείας) ASSIGN

EMPNO	PROJNO	PTIME

Εικόνα 3.4 Βάση δεδομένων που θα χρησιμοποιηθεί στα παραδείγματα της ενότητας.

Ακολουθεί η δημιουργία της βάσης σε Oracle. Οι πίνακες δημιουργούνται χωρίς κύρια και ξένα κλειδιά. Μπορείτε να τα προσθέσετε με χρήση δήλωσης ALTER TABLE.

```
CREATE TABLE dept(deptno NUMBER(2),
   dname VARCHAR2(24),
   loc CHAR(23));

CREATE TABLE EMP (empno NUMBER(4) NOT NULL,
  ename VARCHAR2(20),
  job VARCHAR2(19),
  mgr NUMBER(4),
  hiredate DATE,
  sal NUMBER(10,2),
  comm NUMBER(10,2),
  deptno NUMBER(2));

CREATE TABLE PROJ (projno NUMBER(3) NOT NULL,
   pname VARCHAR2(15),
   budget NUMBER (12,2));

CREATE TABLE ASSIGN (empno NUMBER(4) NOT NULL ,
 projno NUMBER(3) NOT NULL,
 ptime  NUMBER(3));
```

Αν θέλουμε να δούμε τη δομή των πινάκων (δηλαδή, πως έχουν οριστεί οι πίνακες) χρησιμοποιούμε την εντολή (command) describe π.χ. describe emp;

Προσοχή! Η εντολή (command) αυτή δεν ανήκει στις δηλώσεις (statements) της γλώσσας SQL. Κάθε ΣΔΒΔ έχει τη δική του εντολή.

Η εισαγωγή στοιχείων γίνεται με τις παρακάτω δηλώσεις:

```
INSERT INTO dept(deptno, dname, loc)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO dept(deptno, dname, loc)
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO dept(deptno, dname, loc)
   VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO dept(deptno, dname, loc)
   VALUES (40, 'OPERATIONS', 'BOSTON');
```

```
INSERT INTO emp(empno,ename,job,mgr,hiredate,sal,comm,deptno)
  VALUES(10,'CODD','ANALYST',15,'01/01/1989',3000,NULL,10);
INSERT INTO  emp(empno,ename,job,mgr,hiredate,sal,comm,deptno)
  VALUES(15,'ELMASRI','ANALYST',15,'02/05/1995',1200,150,10);
INSERT INTO  emp(empno,ename,job,mgr,hiredate,sal,comm,deptno)
   VALUES   (20,'NAVATHE','SALESMAN',20,'07/07/1977',2000,NULL,20);
INSERT INTO  emp(empno,ename,job,mgr,hiredate,sal,comm,deptno)
VALUES (30,'DATE','PROGRAMMER',15,'04/05/2004',1800,200,10);

INSERT INTO proj(projno, pname, budget)
VALUES(100, 'PAYROLL', 100000);
INSERT INTO proj(projno, pname, budget)
VALUES(200, 'PERSONNEL',200000 );
INSERT INTO proj(projno, pname, budget)
VALUES(300, 'SALES', 150000);
INSERT INTO assign(empno, projno, ptime)
 VALUES(10,100, 40);
INSERT INTO assign(empno, projno, ptime)
  VALUES(10, 200, 60);
INSERT INTO assign(empno, projno, ptime)
  VALUES(15, 100, 100);
INSERT INTO assign(empno, projno, ptime)
   VALUES(20, 200, 100);
INSERT INTO assign(empno, projno, ptime)
   VALUES(30, 100, 100);
```

Ακολουθεί η δημιουργία της βάσης σε mySQL. Οι πίνακες δημιουργούνται με κύρια και ξένα κλειδιά.

Βλέπουμε τις υπάρχουσες βάσεις δεδομένων.

```
SHOW DATABASES;
```

Δημιουργία της βάσης

```
CREATE DATABASE new_personnel;
```

Εντολή για τη χρησιμοποίηση της βάσης δεδομένων.

```
USE new_personnel;
```

Δημιουργία πινάκων

```
CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL,
 DNAME VARCHAR(14), LOC VARCHAR(14),
 PRIMARY KEY(DEPTNO));

CREATE TABLE EMP(EMPNO INT(4) NOT NULL,
  ENAME VARCHAR(10), JOB VARCHAR(25),
  HIREDATE DATE, MGR INT(4),  SAL FLOAT(7,2), COMM FLOAT(7,2),
  DEPTNO INT(2),
  PRIMARY KEY(EMPNO),
  FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO));

CREATE TABLE PROJ (projno INT(3) NOT NULL,
   pname VARCHAR(15),
   budget FLOAT(12,2),
 PRIMARY KEY(projno));
```

```
CREATE TABLE ASSIGN(
 EMPNO INT(4) NOT NULL, PROJNO INT(3) NOT NULL,
 PTIME INT(3),
PRIMARY KEY(EMPNO,PROJNO),
FOREIGN KEY(EMPNO) REFERENCES EMP(EMPNO),
FOREIGN KEY(PROJNO) REFERENCES PROJ(PROJNO));
```

Βλέπουμε τους πίνακες.

```
SHOW TABLES;
```

Αν θέλουμε να δούμε τη δομή των πινάκων (δηλαδή, πως έχουν οριστεί) χρησιμοποιούμε τη δήλωση describe, π.χ. describe emp;

Ακολουθεί η εισαγωγή δεδομένων στους πίνακες.

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (40, 'OPERATIONS', 'BOSTON');

INSERT INTO EMP
  VALUES (10, 'CODD', 'ANALYST', '1989/01/01', 15, 3000, NULL, 10);
INSERT INTO EMP
  VALUES (15, 'ELMASRI', 'ANALYST', '1995/05/02', 15, 1200, 150, 10);
INSERT INTO EMP
  VALUES (20, 'NAVATHE', 'SALESMAN', '1977/07/07', 20, 2000, NULL, 20);
INSERT INTO EMP
  VALUES (30, 'DATE', 'PROGRAMMER', '2004/05/04', 15, 1800, 200, 10);

INSERT INTO proj(projno, pname, budget)
VALUES(100, 'PAYROLL', 100000);
INSERT INTO proj(projno, pname, budget)
VALUES(200, 'PERSONNEL',200000 );
INSERT INTO proj(projno, pname, budget)
VALUES(300, 'SALES', 150000);

INSERT INTO assign(empno, projno, ptime)
 VALUES(10,100, 40);
INSERT INTO assign(empno, projno, ptime)
 VALUES(10, 200, 60);
INSERT INTO assign(empno, projno, ptime)
 VALUES(15, 100, 100);
INSERT INTO assign(empno, projno, ptime)
 VALUES(20, 200, 100);
INSERT INTO assign(empno, projno, ptime)
 VALUES(30, 100, 100);
```

Πως βλέπουμε όλα τα δεδομένα των πινάκων.

```
SELECT * FROM DEPT;
SELECT * FROM EMP;
```

```
SELECT * FROM PROJ;
SELECT * FROM ASSIGN;
```

Πως διαγράφουμε πίνακες

```
DROP TABLE assign;
DROP TABLE emp;
DROP TABLE proj;
DROP TABLE dept;
```

Η διαγραφή της βάσης γίνεται με τη δήλωση:

```
DROP DATABASE NEW_PERSONNEL;
```

Η αναζήτηση στοιχείων από μία βάση δεδομένων στη γλώσσα SQL γίνεται με τη δήλωση SELECT. Στη συνέχεια θα περιγραφεί η σύνταξη της δήλωσης για απλές αλλά και σύνθετες περιπτώσεις αναζήτησης. Η περιγραφή της σύνταξης συνοδεύεται από θέματα-παραδείγματα.

Για τη συνέχεια προτείνουμε τη μελέτη όλων των ενοτήτων που ακολουθούν, τη δοκιμή των θεμάτων-παραδειγμάτων σε ΣΔΒΔ αλλά και έναν παραπέρα προσωπικό πειραματισμό με παρόμοιες δηλώσεις.

Στην εικόνα 3.5 βλέπουμε απόσπασμα δεδομένων της βάσης δεδομένων που θα χρησιμοποιηθεί στα παραδείγματα στις επόμενες ενότητες.

SELECT * FROM dept;

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

SELECT * FROM emp;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	15	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	15	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	20	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	15	04/05/2004	1800	200	10

SELECT * FROM proj;

PROJNO	PNAME	BUDGET
100	PAYROLL	100000
200	PERSONNEL	200000
300	SALES	150000

SELECT * FROM assign;

EMPNO	PROJNO	PTIME
10	100	40
10	200	60

15	100	100
20	200	100
30	100	100

Εικόνα 3.5 Απόσπασμα δεδομένων (sample of data) της βάσης δεδομένων που θα χρησιμοποιηθεί στα παραδείγματα της ενότητας.

3.4.1 Πως βλέπουμε όλα τα στοιχεία ενός πίνακα.

Η σύνταξη της δήλωσης είναι:

```
SELECT  *  FROM  όνομα-πίνακα;
```

Παράδειγμα

Η παρακάτω δήλωση δείχνει τα στοιχεία του πίνακα DEPT.

```
SELECT * FROM dept;
```

Ακολουθεί η απάντηση,

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

3.4.2 Πως αλλάζουμε τη σειρά των στηλών των αποτελεσμάτων.

Η συνηθισμένη σύνταξη της δήλωσης είναι:

```
SELECT  όνομα-πίνακα.όνομα-στήλης, ...
FROM όνομα-πίνακα;
```

Παράδειγμα

Η παρακάτω δήλωση δείχνει τις στήλες DNAME, LOC,

```
SELECT  DEPT.DNAME, DEPT.LOC  FROM  DEPT;
```

DNAME	LOC
ACCOUNTING	NEW YORK
RESEARCH	DALLAS
SALES	CHICAGO
OPERATIONS	BOSTON

Προσοχή! Παρατηρήστε ότι στις στήλες αναφέρουμε και το όνομα του πίνακα στον οποίο ανήκουν, π.χ., **DEPT.DNAME.** Αν είναι σαφές σε ποιές στήλες αναφερόμαστε χωρίς να χρειάζεται να εξετάζουμε το όνομα του πίνακα μπορούμε να το παραλείψουμε. Έτσι η παραπάνω αναζήτηση μπορεί να τροποποιηθεί:

```
SELECT  DNAME, LOC  FROM DEPT;
```

DNAME	LOC
ACCOUNTING	NEW YORK
RESEARCH	DALLAS
SALES	CHICAGO
OPERATIONS	BOSTON

3.4.3 Πως αλλάζουμε τα ονόματα των στηλών των αποτελεσμάτων.

Για να αλλάξουμε το όνομα μιας στήλης στα αποτελέσματα αρκεί να γράψουμε σε εισαγωγικά το νέο όνομα δίπλα στο όνομα της στήλης.

Η παρακάτω δήλωση παραθέτει τους υπαλλήλους της εταιρείας με το μισθό τους.

```
SELECT  ENAME  "ΟΝΟΜΑ ΥΠΑΛΛΗΛΟΥ", SAL "ΜΙΣΘΟΣ" FROM EMP;
```

Ακολουθούν τα αποτελέσματα.

ΟΝΟΜΑ ΥΠΑΛΛΗΛΟΥ	ΜΙΣΘΟΣ
CODD	3000
ELMASRI	1200
NAVATHE	2000
DATE	1800

Σημείωση 1

Όπως θα δούμε στη συνέχεια η γλώσσα SQL επιτρέπει δηλώσεις της μορφής

```
SELECT ENAME, SAL/(25*8) "ΩΡΙΑΙΑ ΑΜΟΙΒΗ",
   SAL+NVL(COMM,0) "ΣΥΝΟΛΟ ΑΠΟΔΟΧΩΝ"
FROM EMP;
```

ENAME	ΩΡΙΑΙΑ ΑΜΟΙΒΗ	ΣΥΝΟΛΟ ΑΠΟΔΟΧΩΝ
CODD	15	3000
ELMASRI	6	1350
NAVATHE	10	2000
DATE	9	2000

Σε mySQL

```
SELECT ENAME, SAL/(25*8) "ΩΡΙΑΙΑ ΑΜΟΙΒΗ",
   SAL+IFNULL(COMM,0) "ΣΥΝΟΛΟ ΑΠΟΔΟΧΩΝ"
FROM EMP;
```

Σημείωση 2

Αν το όνομα που δίνουμε είναι μόνο μια λέξη δεν είναι απαραίτητο να το γράφουμε σε εισαγωγικά. Αυτό ισχύει σε κάποια μόνο προϊόντα, π.χ. Oracle

```
SELECT  ENAME  "ΟΝΟΜΑ ΥΠΑΛΛΗΛΟΥ", SAL ΜΙΣΘΟΣ
FROM EMP;
```

ΟΝΟΜΑ ΥΠΑΛΛΗΛΟΥ	ΜΙΣΘΟΣ
CODD	3000
ELMASRI	1200
NAVATHE	2000
DATE	1800

Σε MySQL χρησιμοποιούμε πάντα εισαγωγικά:

```
SELECT  ENAME  "ΟΝΟΜΑ ΥΠΑΛΛΗΛΟΥ", SAL "ΜΙΣΘΟΣ"
FROM EMP;
```

Σημείωση 3

Την εμφάνιση των στηλών μπορούμε να τη συμπληρώσουμε με φράσεις, και επίσης μπορούμε να εμφανίσουμε στοιχεία δύο ή περισσότερων στηλών σαν να ανήκουν στην ίδια στήλη όπως φαίνεται στο παράδειγμα που ακολουθεί.

```
SELECT  empno||ename FROM emp;
```

EMPNO||ENAME
10CODD
15ELMASRI
20NAVATHE
30DATE

Παρατηρήστε ότι γίνεται χρήση του συμβόλου || της παράθεσης (concatenation) .

Σημείωση 4

Ανάλογα με το προϊόν θα πρέπει να χρησιμοποιήσετε απλά εισαγωγικά ‘ ... ‘ ή διπλά " ... "

3.4.4 Αναζητήσεις που οδηγούν σε αποτελέσματα χωρίς επανάληψη τιμών - τελεστής DISTINCT

Η συνηθισμένη σύνταξη της δήλωσης είναι:

```
SELECT  DISTINCT  όνομα-πίνακα.όνομα-στήλης,
 όνομα-πίνακα.όνομα-στήλης, ...
FROM όνομα-πίνακα κ.λπ
```

Παράδειγμα

Η παρακάτω αναζήτηση προσδιορίζει τα επαγγέλματα των υπαλλήλων επαναλαμβάνοντας τις τιμές στα αποτελέσματα:

```
SELECT  JOB  FROM EMP;
```

JOB
ANALYST
ANALYST

SALESMAN
PROGRAMMER

Αντίθετα η παρακάτω αναζήτηση προσδιορίζει τα επαγγέλματα των υπαλλήλων αλλά δεν επαναλαμβάνει τις τιμές, δηλαδή, αναγράφει όλα τα επαγγέλματα (τις θέσεις) που υπάρχουν στο πίνακα EMP μόνο μία φορά.

```
SELECT DISTINCT JOB FROM EMP;
```

JOB
SALESMAN
ANALYST
PROGRAMMER

Παράδειγμα

Παρατηρήστε τη διαφορά και στις επόμενες δηλώσεις.

```
SELECT  job, deptno from  emp;
```

JOB	DEPTNO
ANALYST	10
ANALYST	10
SALESMAN	20
PROGRAMMER	10

```
SELECT  DISTINCT job, deptno from  emp;
```

JOB	DEPTNO
ANALYST	10
PROGRAMMER	10
SALESMAN	20

Προσοχή! Η δήλωση

```
SELECT DISTINCT job, deptno from  emp;
```

θα εμφανίσει τρεις γραμμές επειδή το ζεύγος στοιχείων (ANALYST, 10) θα εμφανίζεται μία μόνο φορά.

Σημείωση

Ο τελεστής DISTINCT μπορεί να χρησιμοποιηθεί και με συναρτήσεις και γενικά σε σύνθετες αναζητήσεις (που περιγράφονται παρακάτω).

3.4.5 Πως περιορίζετε της στήλες των αποτελεσμάτων με απλές συνθήκες . Υποπρόταση WHERE.

Η συνηθισμένη σύνταξη της δήλωσης είναι:

```
SELECT  ...
 FROM...
 WHERE   συνθήκη;
```

Παράδειγμα 1

Η παρακάτω αναζήτηση βρίσκει τμήματα της εταιρείας που έχουν έδρα New York.

```
SELECT  DNAME "ΟΝΟΜΑ ΤΜΗΜΑΤΟΣ", LOC  ΕΔΡΑ
FROM  DEPT
WHERE  LOC = 'NEW YORK';
```

ΟΝΟΜΑ ΤΜΗΜΑΤΟΣ	ΕΔΡΑ
ACCOUNTING	NEW YORK

Σε MySQL:

```
SELECT  DNAME "ΟΝΟΜΑ ΤΜΗΜΑΤΟΣ", LOC  "ΕΔΡΑ"
FROM  DEPT
WHERE  LOC = 'NEW YORK';
```

Παράδειγμα 2

Δείξτε μόνο τους υπάλληλους του Τμήματος 10.

```
SELECT *
FROM EMP
WHERE DEPTNO = 10;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	15	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	15	02/05/1995	1200	150	10
30	DATE	PROGRAMMER	15	04/05/2004	1800	200	10

Παράδειγμα 3

Δείξτε όνομα, κωδικό και τμήμα για της αναλυτές.

```
SELECT ENAME,EMPNO,JOB, DEPTNO
FROM EMP
WHERE JOB = 'ANALYST';
```

ENAME	EMPNO	JOB	DEPTNO
CODD	10	ANALYST	10
ELMASRI	15	ANALYST	10

Παράδειγμα 4

Δείξτε ονόματα και κωδικούς Τμημάτων μόνο για τα τμήματα με κωδικό αριθμό μεγαλύτερο του 20.

```
SELECT DNAME, DEPTNO
FROM DEPT
WHERE DEPTNO > 20;
```

DNAME	DEPTNO
SALES	30
OPERATIONS	40

Παράδειγμα 5

Δείξτε ονόματα και αριθμούς τμημάτων για τμήματα με κωδικό μεγαλύτερο ή ίσο του 20.

```
SELECT DNAME,DEPTNO
FROM DEPT
WHERE DEPTNO >= 20;
```

DNAME	DEPTNO
RESEARCH	20
SALES	30
OPERATIONS	40

Παράδειγμα 6

Δείξτε υπαλλήλους με προμήθεια μεγαλύτερη του δέκατου του μισθού τους.

```
SELECT ENAME,SAL,SAL/10, COMM
FROM EMP
WHERE COMM > SAL/10;
```

ENAME	SAL	SAL/10	COMM
ELMASRI	1200	120	150
DATE	1800	180	200

Προσοχή! Αν το ζητούμενο τροποποιηθεί και είναι «Δείξτε υπαλλήλους με προμήθεια μικρότερη του δέκατου του μισθού τους» τότε η παρακάτω αναζήτηση είναι λανθασμένη και θα πρέπει να χρησιμοποιηθεί η συνάρτηση NVL (αντίστοιχα η συνάρτηση IFNULL στο προϊόν MySQL)

```
SELECT ENAME,SAL,SAL/10, COMM
FROM EMP
WHERE COMM < SAL/10; -- διόρθωση NVL(COMM,0) < SAL/10;
```

Σημείωση

Σε επόμενες παραγράφους δίνονται της «μορφές» συνθηκών που χρησιμοποιούν ειδικούς τελεστές, παρενθέσεις κ.λπ.

3.4.6 Αναζητήσεις που περιλαμβάνουν υπολογισμούς, πράξεις σε συμβολοσειρές (strings) κ.λπ.

Για να υπολογίσετε αθροίσματα, γινόμενα, μέσους όρους κ.λπ. μπορείς να χρησιμοποιήσετε κατά την επιλογή στηλών:

1) αριθμητικούς τελεστές (+, -, *, /), δηλαδή μπορείτε να κάνετε πράξεις μεταξύ των τιμών των στηλών
2) συναρτήσεις, της αυτές που χρησιμοποιούνται στο προϊόν της Oracle, π.χ., NVL-Null Value (IFNULL στο προϊόν MySQL), POWER-ύψωση σε δύναμη, ROUND-στρογγύλευση, TRUNC-αποκοπή, SUBSTR-substring, UPPER-μεταγραφή σε κεφαλαία, AVG-μέσος όρος κ.λπ.,
3) παρενθέσεις,
4) τελεστή DISTINCT.

Παράδειγμα 1

Βλέπουμε τη σημερινή ημερομηνία με τη συνάρτηση sysdate χρησιμοποιώντας τον πίνακα dual του συστήματος.

```
SELECT sysdate FROM  dual;
```

SYSDATE
24/01/2022

Σε mySQL:

```
SELECT current_date;
```

Σε διάφορα προϊόντα, της για παράδειγμα στο προϊόν της **ORACLE**, είναι δυνατές κάποιες αριθμητικές πράξεις σε ημερομηνίες.

```
SELECT sysdate + 5  FROM  dual;
```

SYSDATE+5
29/01/2021

```
SELECT sysdate - 5  FROM  dual;
```

SYSDATE-5
19/01/2021

Υποτίθεται ότι η σημερινή ημερομηνία είναι 24/1/2021.

Σε mySQL:

```
SELECT CURRENT_DATE  FROM  dual;
SELECT CURRENT_DATE+ 5 FROM  dual;
SELECT CURRENT_DATE- 5 FROM  dual;
```

Βλέπουμε τα αποτελέσματα στην εικόνα 3.6

```
mysql>
mysql> SELECT CURRENT_DATE  FROM  dual;
+--------------+
| CURRENT_DATE |
+--------------+
| 2021-01-24   |
+--------------+
1 row in set (0.00 sec)

mysql> SELECT CURRENT_DATE+ 5 FROM  dual;
+-----------------+
| CURRENT_DATE+ 5 |
+-----------------+
|        20210129 |
+-----------------+
1 row in set (0.00 sec)

mysql> SELECT CURRENT_DATE- 5 FROM  dual;
+-----------------+
| CURRENT_DATE- 5 |
+-----------------+
|        20210119 |
+-----------------+
1 row in set (0.00 sec)
```

Εικόνα 3.6 Χρήση συνάρτησης CURRENT_DATE σε MySQL

Παράδειγμα 2

Αναζήτηση που εμφανίζει τον αριθμό ημερών μεταξύ δύο ημερομηνιών.

Σε Oracle:

```
SELECT  sysdate - hiredate, ename
FROM  emp;
```

Σε MySQL:

```
SELECT  CURRENT_DATE - hiredate, ename
FROM  emp;
```

Δείτε την εικόνα 3.7

```
mysql> SELECT  CURRENT_DATE - hiredate, ename
    -> FROM  emp;
+-------------------------+-------------------+
| CURRENT_DATE - hiredate | ename             |
+-------------------------+-------------------+
|                   20023 | ΣΚΟΥΡΑΣ           |
|                   20023 | ΝΙΚΟΥ Ν.          |
|                   40023 | ΑΝΔΡΕΟΥ Ν.        |
|                  120023 | ΑΝΔΡΕΟΥ           |
|                  140023 | ΑΝΔΡΕΟΥ Ν.        |
+-------------------------+-------------------+
5 rows in set (0.03 sec)
```

Εικόνα 3.7 Χρήση συνάρτησης CURRENT_DATE και υπολογισμός αριθμού ημερών μεταξύ δύο ημερομηνιών σε MySQL

Στο προϊόν της MySQL μπορούμε να χρησιμοποιούμε και τη συνάρτηση datediff: DATEDIFF(date1,date2).

Παράδειγμα 3

Όπως είδαμε και στα προηγούμενα η παρακάτω αναζήτηση βρίσκει την ωριαία και τη συνολική αμοιβή υπαλλήλων

```
SELECT  ENAME ONOMA,
SAL/(25*8) "ΩΡΙΑΙΑ ΑΜΟΙΒΗ",
SAL+NVL(COMM,0) "ΣΥΝΟΛΙΚΗ ΑΜΟΙΒΗ"
FROM EMP;
```

Η συνάρτηση NVL εξασφαλίζει ότι το άθροισμα "Μισθός+Προμήθεια" έχει τιμή στην περίπτωση υπαλλήλων που δεν έχουν προμήθεια (έχουν τιμή προμήθειας «τίποτα», NULL).

Σε MySQL:

```
SELECT  ENAME "ONOMA",
SAL/(25*8) "ΩΡΙΑΙΑ ΑΜΟΙΒΗ",
SAL+IFNULL(COMM,0) "ΣΥΝΟΛΙΚΗ ΑΜΟΙΒΗ"
FROM EMP;
```

Δείτε τι θα συμβεί αν δεν χρησιμοποιήσουμε τη συνάρτηση.

```
SELECT  ENAME ONOMA,
SAL/(25*8) "ΩΡΙΑΙΑ ΑΜΟΙΒΗ",SAL+COMM,
SAL+NVL(COMM,0) "ΣΥΝΟΛΙΚΗ ΑΜΟΙΒΗ"
FROM EMP;
```

ΟΝΟΜΑ	ΩΡΙΑΙΑ ΑΜΟΙΒΗ	SAL+COMM	ΣΥΝΟΛΙΚΗ ΑΜΟΙΒΗ
CODD	15	-	3000
ELMASRI	6	1350	1350
NAVATHE	10	-	2000
DATE	9	2000	2000

3.4.7 Πως σχηματίζουμε σύνθετες συνθήκες σε υποπρόταση WHERE. Τελεστές σύγκρισης, Αριθμητικοί, Boole, LIKE, NOT LIKE, BETWEEN ...AND, NOT BETWEEN ...AND, σύνολα (IN).

Για να σχηματίσουμε συνθήκες μπορούμε να χρησιμοποιήσουμε:

1) τελεστές σύγκρισης (>, <, >=, <=, != , <>,^ =), όπου οι τρεις τελευταίοι (ή και άλλοι ανάλογα με το προϊόν) συμβολίζουν το διάφορο.
2) αριθμητικούς τελεστές (+, -, /, *),
3) τελεστές Boole (AND, OR, NOT),
4) τελεστή LIKE ή NOT LIKE (π.χ. ENAME NOT LIKE "%ΑΣ"),
5) τελεστές BETWEEN ...AND ή NOT BETWEEN ...AND
6) (π.χ. SAL BETWEEN 2500 AND 3000)
7) σύνολα τιμών (τελεστής IN),
8) (π.χ. JOB IN ("ΠΩΛΗΤΗΣ", "ΑΝΑΛΥΤΗΣ ΣΥΣΤΗΜΑΤΩΝ")),
9) παρενθέσεις.

Παράδειγμα 1

Η παρακάτω αναζήτηση βρίσκει πωλητές με μισθό πάνω από 1,300 ή προμήθεια μεγαλύτερη του ενός δεκάτου του μισθού τους

```
SELECT  ENAME ΟΝΟΜΑ, SAL ΜΙΣΘΟΣ,COMM  ΠΡΟΜΗΘΕΙΑ
FROM EMP
WHERE JOB = 'SALESMAN'
AND ((SAL> 1300) OR (COMM > SAL/10));
```

ΟΝΟΜΑ	ΜΙΣΘΟΣ	ΠΡΟΜΗΘΕΙΑ
ELMASRI	1200	150
NAVATHE	2000	-
DATE	1800	200

Σε mySQL:

```
SELECT  ENAME "ΟΝΟΜΑ", SAL "ΜΙΣΘΟΣ", COMM "ΠΡΟΜΗΘΕΙΑ"
FROM EMP
WHERE JOB = 'SALESMAN'
AND ((SAL> 1300) OR (COMM > SAL/10));
```

Παράδειγμα 2

Η παρακάτω αναζήτηση βρίσκει πωλητές που έχουν μισθό μεταξύ 1800 και 2000. Οι τιμές 1800 και 2000 περιλαμβάνονται.

```
SELECT  ENAME, JOB, SAL
FROM EMP
WHERE JOB = 'SALESMAN'
AND SAL BETWEEN 1800 AND 2000;
```

ENAME	JOB	SAL
NAVATHE	SALESMAN	2000

Παράδειγμα 3

Η παρακάτω αναζήτηση βρίσκει πωλητές ή αναλυτές συστημάτων που έχουν μισθό μεγαλύτερο από 1250

```
SELECT  ENAME, JOB, SAL
FROM EMP
WHERE JOB IN ('SALESMAN', 'ANALYST')
AND SAL> 1250;
```

ENAME	JOB	SAL
CODD	ANALYST	3000
NAVATHE	SALESMAN	2000

Παράδειγμα 4

Η παρακάτω αναζήτηση βρίσκει αναλυτές με όνομα που αρχίζει από C, D, E.

```
SELECT  ename, job
FROM   emp
WHERE  (ename > = 'C'  AND ename < 'F' )
AND  job = 'ANALYST';
```

ENAME	JOB
CODD	ANALYST
ELMASRI	ANALYST

Σε mySQL δε γίνεται διάκριση για κεφαλαία-πεζά στη σύγκριση με τις τιμές 'ANALYST', 'analyst'

```
SELECT  ename, job
FROM   emp
WHERE  (ename >= 'C'  AND ename < 'F' )
AND  job = 'ANALYST';

SELECT  ename, job
FROM   emp
WHERE  (ename >= 'C'  AND ename < 'F' )
AND  job = 'analyst';
```

Παρατήρηση

Σε mySQL μπορεί να γίνει διάκριση για κεφαλαία-πεζά στη σύγκριση με τις τιμές 'ANALYST', 'analyst':

```
SELECT  ename, job
FROM   emp
WHERE  (ename >= "C"  AND ename < "F" )
AND  job = "ANALYST";

SELECT  ename, job
FROM   emp
WHERE  (ename >= "C"  AND ename < "F" )
AND  BINARY job = "ANALYST";
```

```
SELECT  ename, job
FROM   emp
WHERE  (ename >= "C"  AND ename < "F" )
AND  BINARY job = "analyst";
```

```
mysql> SELECT  ename, job
    -> FROM   emp
    -> WHERE  (ename >= "C"  AND ename < "F" )
    -> AND  job = "ANALYST";
+---------+---------+
| ename   | job     |
+---------+---------+
| CODD    | ANALYST |
| ELMASRI | ANALYST |
+---------+---------+
2 rows in set (0.00 sec)

mysql> SELECT  ename, job
    -> FROM   emp
    -> WHERE  (ename >= "C"  AND ename < "F" )
    -> AND  BINARY job = "ANALYST";
+---------+---------+
| ename   | job     |
+---------+---------+
| CODD    | ANALYST |
| ELMASRI | ANALYST |
+---------+---------+
2 rows in set (0.00 sec)

mysql> SELECT  ename, job
    -> FROM   emp
    -> WHERE  (ename >= "C"  AND ename < "F" )
    -> AND  BINARY job = "analyst";
Empty set (0.00 sec)
```

Συνεχίζουμε με άλλη εναλλακτική λύση στο παράδειγμά μας. Το ίδιο αποτέλεσμα έχει και η παρακάτω αναζήτηση η οποία χρησιμοποιεί τη συνάρτηση substr για να απομονώσει το πρώτο γράμμα του ονόματος του υπαλλήλου.

```
SELECT  ename, job
FROM   emp
WHERE (substr(ename,1,1) IN ('C', 'D', 'E'))
AND  job = 'ANALYST';
```

ENAME	JOB
CODD	ANALYST
ELMASRI	ANALYST

Σημείωση

Η χρήση LIKE για την σύγκριση συμβολοσειρών (strings) συνδέεται συνήθως με το σύμβολο % όπως φαίνεται στα παραδείγματα:

Παράδειγμα 5

Η παρακάτω δήλωση δείχνει την εργασία υπαλλήλων με όνομα CODD

```
SELECT ename, job
FROM emp
WHERE ename LIKE 'CODD';
```

ENAME	JOB
CODD	ANALYST

Η παρακάτω δήλωση περιλαμβάνει ... like 'C%' και δείχνει την εργασία αυτών με επώνυμο που αρχίζει από το γράμμα C

```
SELECT ename, job
FROM emp
WHERE ename LIKE 'C%';
```

ENAME	JOB
CODD	ANALYST

Η παρακάτω δήλωση περιλαμβάνει ... like '%TE' και δείχνει την εργασία αυτών που το όνομα τους τελειώνει σε TE

```
SELECT ename, job
FROM emp
WHERE ename LIKE'%TE';
```

ENAME	JOB
DATE	PROGRAMMER

Η παρακάτω δήλωση περιλαμβάνει ... like '%D%' και δείχνει την εργασία αυτών με ονοματεπώνυμο που περιέχει το γράμμα D

```
SELECT ename, job
FROM emp
WHERE ename LIKE '%D%';
```

ENAME	JOB
CODD	ANALYST
DATE	PROGRAMMER

Η αναζήτηση δείχνει θέσεις εργασίας που το όνομά τους αρχίζει από A.

```
SELECT DISTINCT job
FROM emp
WHERE job LIKE 'A%';
```

JOB
ANALYST

Σε mySQL δε γίνεται διάκριση για κεφαλαία-πεζά στη σύγκριση με τις τιμές 'A%', 'a%':

```
SELECT DISTINCT job
FROM emp
WHERE job like “A%”;

SELECT DISTINCT job
FROM emp
WHERE job like “a%”;
```

Παράδειγμα 6

Η παρακάτω αναζήτηση δείχνει υπαλλήλους που το επώνυμο τους αρχίζει από C ή D και προσλήφθηκαν από το 1989 έως και το 1995

```
SELECT  ename, hiredate
from  emp
where  ename between ‘C’  and ‘F’
and  hiredate Between ‘01/01/1989’  and ‘12/31/1995’;
```

ENAME	HIREDATE
CODD	01/01/1989
ELMASRI	02/05/1995

Σε mySQL

```
SELECT  ename, hiredate
from  emp
where  ename between “C”  and “F”
and  hiredate Between “1989/01/01”  and “1995/12/31”;
```

Στο παράδειγμα μπορείτε να χρησιμοποιήσετε και τη συνάρτηση substr.

Παράδειγμα 7

Δείξτε όνομα, θέση και μισθό των υπαλλήλων του Τμήματος 10 που κερδίζουν πάνω από 2000.

```
SELECT ENAME,JOB,SAL,DEPTNO
FROM EMP
WHERE DEPTNO = 10
AND SAL > 2000;
```

ENAME	JOB	SAL	DEPTNO
CODD	ANALYST	3000	10

Παράδειγμα 8

Δείξτε τους πωλητές του Τμήματος 10 με μισθό μεγαλύτερο ή ίσο των 1,500 ευρώ.

```
SELECT ENAME, SAL, JOB, DEPTNO
FROM EMP
WHERE JOB = ‘ANALYST’
 AND DEPTNO = 10
 AND SAL >= 1500;
```

ENAME	SAL	JOB	DEPTNO
CODD	3000	ANALYST	10

Παράδειγμα 9

Δείξτε τους υπάλληλους που είναι πωλητές ή αναλυτές

```
SELECT ENAME, JOB, SAL, DEPTNO
FROM EMP
WHERE JOB = 'ANALYST'
OR JOB = 'SALESMAN';
```

ENAME	JOB	SAL	DEPTNO
CODD	ANALYST	3000	10
ELMASRI	ANALYST	1200	10
NAVATHE	SALESMAN	2000	20

Παράδειγμα 10

Δείξτε τους πωλητές ή αναλυτές που δεν ανήκουν στο τμήμα 10.

```
SELECT ENAME, JOB, SAL, DEPTNO
FROM EMP
WHERE (JOB = 'ANALYST' OR JOB = 'SALESMAN')
AND DEPTNO <> 10;
```

ENAME	JOB	SAL	DEPTNO
NAVATHE	SALESMAN	2000	20

Παράδειγμα 11

Δείξτε τους πωλητές ή αναλυτές που ανήκουν στο τμήμα 10.

```
SELECT ENAME, JOB, SAL, DEPTNO
FROM EMP
WHERE (JOB = 'ANALYST' OR JOB = 'SALESMAN')
AND DEPTNO=10;
```

ENAME	JOB	SAL	DEPTNO
CODD	ANALYST	3000	10
ELMASRI	ANALYST	1200	10

Παράδειγμα 12

Δείξτε πωλητές οποιουδήποτε τμήματος και αναλυτές του Τμήματος 10.

```
SELECT *
FROM EMP
WHERE JOB = 'SALESMAN'
OR (JOB = 'ANALYST' AND DEPTNO = 10);
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	15	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	15	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	20	07/07/1977	2000	-	20

Ποιό είναι το αποτέλεσμα της επόμενης αναζήτησης;

```
SELECT *
FROM EMP
WHERE (JOB = 'SALESMAN' OR JOB = 'ANALYST')
AND DEPTNO = 20;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
20	NAVATHE	SALESMAN	20	07/07/1977	2000	-	20

Παράδειγμα 13

Δείξτε υπάλληλους του Τμήματος 10 που δεν είναι πωλητές ή αναλυτές.

```
SELECT *
FROM EMP
WHERE NOT (JOB = 'SALESMAN' OR JOB = 'ANALYST')
AND DEPTNO = 10;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
30	DATE	PROGRAMMER	15	04/05/2004	1800	200	10

Παράδειγμα 14

Δείξτε ποιοί υπάλληλοι έχουν μισθό μεταξύ 1,500 και 1,800.

```
SELECT ENAME,JOB,SAL
FROM EMP
WHERE SAL BETWEEN 1500 AND 1800;
```

ENAME	JOB	SAL
DATE	PROGRAMMER	1800

Παράδειγμα 15

Χρησιμοποιήστε τον τελεστή BETWEEN με τον τελεστή NOT για να δείξετε τους υπόλοιπους υπαλλήλους.

```
SELECT ENAME,JOB,SAL
FROM EMP
WHERE SAL NOT BETWEEN 1500 AND 1800;
```

ENAME	JOB	SAL
CODD	ANALYST	3000
ELMASRI	ANALYST	1200
NAVATHE	SALESMAN	2000

Παράδειγμα 16

Δείξτε υπάλληλους που είναι αναλυτές ή πωλητές και στη συνέχεια δείξτε τους υπόλοιπους υπαλλήλους.

```
SELECT ENAME,JOB,DEPTNO
FROM EMP
WHERE JOB IN ('ANALYST','SALESMAN');
```

ENAME	JOB	DEPTNO
CODD	ANALYST	10
ELMASRI	ANALYST	10
NAVATHE	SALESMAN	20

```
SELECT ENAME,JOB,DEPTNO
FROM EMP
WHERE JOB NOT IN ('ANALYST','SALESMAN');
```

ENAME	JOB	DEPTNO
DATE	PROGRAMMER	10

Παράδειγμα 17

Δείξτε υπαλλήλους που το επώνυμο τους αρχίζει με το γράμμα C ή το γράμμα E .

```
SELECT ENAME,JOB,DEPTNO
FROM EMP
WHERE (ENAME LIKE 'C%' OR ENAME LIKE 'E%');
```

ENAME	JOB	DEPTNO
CODD	ANALYST	10
ELMASRI	ANALYST	10

Θα μπορούσαμε να χρησιμοποιήσουμε και τη συνάρτηση substr.

Παράδειγμα 18

Δείξτε υπαλλήλους που το όνομα τους τελειώνει σε THE και είναι Πωλητές .

```
SELECT ENAME,JOB,DEPTNO
FROM EMP
WHERE ENAME LIKE '%THE'
AND JOB = 'SALESMAN';
```

ENAME	JOB	DEPTNO
NAVATHE	SALESMAN	20

3.4.8 Πως σχηματίζουμε συνθήκες που περιλαμβάνουν υποαναζήτηση (SELECT).

Για το σχηματισμό συνθήκης με «φωλιασμένη» (embedded) υποαναζήτηση μπορείτε να χρησιμοποιήσετε:

- τελεστές σύγκρισης (>, <, >=, <=, =, !=, <> , ^ =)
- τελεστές Boole (AND, OR, NOT)

- παρενθέσεις
- τελεστές ANY, ALL, IN, EXIST
- συναρτήσεις.

Παράδειγμα 1

Η παρακάτω αναζήτηση δείχνει υπαλλήλους που κάνουν την ίδια εργασία (έχουν την ίδια θέση) με τον υπάλληλο CODD ή τον υπάλληλο ELMASRI και έχουν μισθό μεγαλύτερο των 1250.

```
SELECT  ENAME, JOB, SAL
 FROM EMP
 WHERE  JOB IN (SELECT JOB
FROM EMP
WHERE  ENAME='CODD' OR  ENAME='ELMASRI')
 AND  SAL > 1250;
```

ENAME	JOB	SAL
CODD	ANALYST	3000

Παράδειγμα 2

Η παρακάτω αναζήτηση βρίσκει ποιοί υπάλληλοι παίρνουν περισσότερο μισθό από όλους τους πωλητές και ποιά είναι η εργασία (θέση) τους.

```
SELECT  ENAME ONOMA, JOB ΘΕΣΗ, SAL ΜΙΣΘΟΣ
 FROM  EMP
 WHERE  SAL > (SELECT  MAX(SAL)
  FROM  EMP
  WHERE  JOB = 'SALESMAN');
```

ΟΝΟΜΑ	ΘΕΣΗ	ΜΙΣΘΟΣ
CODD	ANALYST	3000

Σε MySQL:

```
SELECT  ENAME "ONOMA", JOB "ΘΕΣΗ", SAL "ΜΙΣΘΟΣ"
 FROM  EMP
 WHERE  SAL > (SELECT  MAX(SAL)
  FROM  EMP
  WHERE  JOB = "SALESMAN");
```

Παραδείγμα 3

Έστω βάση δεδομένων εταιρείας που περιλαμβάνει ανάμεσα σε άλλους πίνακες και τους πίνακες πελατών και προμηθευτών:

CUSTOMER(cust_name, cust_city)

SUPPLIER(suppl_name, suppl_city)

Δημιουργούμε τους πίνακες.

```
CREATE TABLE customer(cust_name CHAR(15),
  cust_city CHAR(15));
```

```
CREATE TABLE supplier(suppl_name CHAR(15),
   suppl_city CHAR(15));
```

Εισάγουμε στοιχεία

```
INSERT INTO customer VALUES('CODD', 'NEW YORK');
INSERT INTO customer VALUES('DATE', 'BERLIN');

INSERT INTO supplier VALUES('NAVATHE', 'NEW YORK');
INSERT INTO supplier VALUES('ELMASRI', 'ATHENS');
```

Βλέπουμε τα δεδομένα.

```
SELECT * FROM supplier;
```

SUPPL_NAME	SUPPL_CITY
NAVATHE	NEW YORK
ELMASRI	ATHENS

CUST_NAME	CUST_CITY
CODD	NEW YORK
DATE	BERLIN

Η παρακάτω αναζήτηση βρίσκει τους πελάτες (customers) που έχουν έδρα την ίδια πόλη με κάποιο προμηθευτή (supplier):

```
SELECT   cust_name
FROM customer
WHERE   EXISTS
(SELECT suppl_city
 FROM supplier
 WHERE supplier.suppl_city = customer.cust_city);
```

CUST_NAME
CODD

Σε MySQL:

```
CREATE DATABASE company;
USE company:
CREATE TABLE customer(cust_name CHAR(15),
  cust_city CHAR(15));

CREATE TABLE supplier(suppl_name CHAR(15),
   suppl_city CHAR(15));

INSERT INTO customer VALUES('CODD', 'NEW YORK');
INSERT INTO customer VALUES('DATE', 'BERLIN');
INSERT INTO supplier VALUES('NAVATHE', 'NEW YORK');
INSERT INTO supplier VALUES('ELMASRI', 'ATHENS');

SELECT * FROM supplier;
SELECT * FROM customer;
```

```
SELECT  cust_name
FROM customer
WHERE EXISTS (SELECT suppl_city
 FROM supplier
 WHERE supplier.suppl_city = customer.cust_city);


DROP TABLE customer;
DROP TABLE supplier;
```

Σε επόμενες παραγράφους θα χρησιμοποιήσουμε δήλωση SELECT που χρησιμοποιεί σύνδεση (join) πινάκων. Δείτε μια πρώτη μορφή της σύνταξης της δήλωσης αυτής :

```
SELECT *
FROM customer, supplier
WHERE customer.cust_city=supplier.suppl_city;
```

Σε πολλές περιπτώσεις με χρήση σύνδεσης μπορούμε να αποφύγουμε τις δηλώσεις τύπου SELECT …. SELECT,

Παράδειγμα 4

Ποιοι υπάλληλοι ανήκουν στο ίδιο τμήμα και έχουν την ίδια θέση με τον CODD.

```
SELECT  ename, deptno, sal
FROM  emp
WHERE  (deptno, job) IN (SELECT  deptno, job
 FROM  emp
 WHERE  ename = 'CODD');
```

ENAME	DEPTNO	SAL
ELMASRI	10	1200
CODD	10	3000

Παράδειγμα 5

Εστω ότι ενδιαφερόμαστε για τη θέση του CODD.

```
SELECT JOB
FROM EMP
WHERE ENAME = 'CODD';
```

JOB
ANALYST

Τώρα που βλέπουμε ότι ο CODD είναι αναλυτής ζητάμε όλους τους αναλυτές.

```
SELECT ENAME,JOB
FROM EMP
WHERE JOB = 'ANALYST';
```

ENAME	JOB
CODD	ANALYST
ELMASRI	ANALYST

ή εναλλακτικά με χρήση υποαναζήτησης

```
SELECT ENAME,JOB
FROM EMP
WHERE JOB =
   (SELECT JOB
FROM EMP
WHERE ENAME = 'CODD');
```

ENAME	JOB
CODD	ANALYST
ELMASRI	ANALYST

Ερώτηση

Στα παραδείγματα 4 και 5 θεωρούμε ότι έχουμε έναν υπάλληλο με το όνομα Codd. Θα αλλάξει κάτι αν έχουμε πολλούς υπαλλήλους με το ίδιο όνομα; (Υπόδειξη: Εισάγετε και άλλους υπαλλήλους με το όνομα Codd και δοκιμάστε τις

Πώς χρησιμοποιούμε τον τελεστή ANY

Παράδειγμα 6

Δείξτε όλους τους υπάλληλους που κερδίζουν περισσότερα από υπάλληλο του Τμήματος 20.

```
SELECT EMPNO,SAL,JOB,ENAME,DEPTNO
FROM EMP
WHERE SAL > ANY
   (SELECT SAL
FROM   EMP
WHERE  DEPTNO = 20)
ORDER BY SAL DESC;
```

EMPNO	SAL	JOB	ENAME	DEPTNO
10	3000	ANALYST	CODD	10

Πώς χρησιμοποιούμε τον τελεστή ALL

Παράδειγμα 7

Δείξτε τους υπάλληλους που κερδίζουν περισσότερα από όλους τους υπάλληλους του Τμήματος 20.

Πρώτα θα δούμε τα στοιχεία όλων των υπαλλήλων.

```
SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	15	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	15	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	20	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	15	04/05/2004	1800	200	10

```
SELECT SAL,JOB,ENAME,DEPTNO
FROM EMP
WHERE SAL > ALL
   (SELECT SAL
FROM   EMP
WHERE  DEPTNO = 20)
ORDER BY SAL DESC;
```

SAL	JOB	ENAME	DEPTNO
3000	ANALYST	CODD	10

Πώς χρησιμοποιούμε τον τελεστή IN

Παράδειγμα 8

Δείξτε τους αναλυτές που ανήκουν στο ίδιο τμήμα με προγραμματιστή.

```
SELECT ENAME,DEPTNO
FROM EMP
WHERE JOB = 'ANALYST'
 AND DEPTNO IN
   (SELECT DEPTNO
FROM   EMP
WHERE  JOB = 'PROGRAMMER');
```

ENAME	DEPTNO
ELMASRI	10
CODD	10

Παράδειγμα 9

Δείξτε τους αναλυτές που δεν ανήκουν στο ίδιο τμήμα με πωλητή.

Πρώτα βλέπετε όλους τους υπαλλήλους.

```
SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	15	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	15	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	20	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	15	04/05/2004	1800	200	10

```
SELECT ENAME,DEPTNO
FROM EMP
WHERE JOB = 'ANALYST'
 AND DEPTNO NOT IN
   (SELECT DEPTNO
FROM   EMP
WHERE  JOB = 'SALESMAN');
```

ENAME	DEPTNO
ELMASRI	10
CODD	10

Παράδειγμα 10

Δείξτε όνομα, θέση και μισθό υπαλλήλων που έχουν την ίδια θέση και ανήκουν στο ίδιο τμήμα με υπάλληλο που ονομάζεται CODD.

```
SELECT ENAME,JOB,SAL, DEPTNO
FROM EMP
WHERE  (JOB,DEPTNO) =
   (SELECT JOB,DEPTNO
FROM EMP
WHERE ENAME = 'CODD');
```

ENAME	JOB	SAL	DEPTNO
CODD	ANALYST	3000	10
ELMASRI	ANALYST	1200	10

Παράδειγμα 11

Δείξτε όνομα, θέση και τμήμα υπαλλήλων που έχουν ίδια θέση με τον υπάλληλο CODD ή μισθό μεγαλύτερο ή ίσο του DATE.

```
SELECT ENAME,JOB,DEPTNO,SAL
FROM EMP
WHERE JOB IN
   (SELECT JOB
FROM EMP
WHERE ENAME = 'CODD')
OR SAL >=
   (SELECT SAL
FROM EMP
WHERE ENAME = 'DATE')
ORDER BY JOB,SAL;
```

ENAME	JOB	DEPTNO	SAL
ELMASRI	ANALYST	10	1200
CODD	ANALYST	10	3000
DATE	PROGRAMMER	10	1800
NAVATHE	SALESMAN	20	2000

Παράδειγμα 12

Δείξτε τους υπάλληλους που έχουν θέση που συναντάται στα τμήματα SALES, ACCOUNTING.

Βλέπετε πρώτα τα στοιχεία των τμημάτων και των υπαλλήλων.

```
SELECT * FROM dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

```
SELECT * FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	15	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	15	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	20	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	15	04/05/2004	1800	200	10

```
SELECT ENAME, JOB, DEPTNO
FROM EMP
WHERE JOB IN
  (SELECT JOB
   FROM   EMP
   WHERE  DEPTNO IN (SELECT DEPTNO
 FROM   DEPT
 WHERE  DNAME IN ('SALES', 'ACCOUNTING')));
```

ENAME	JOB	DEPTNO
ELMASRI	ANALYST	10
CODD	ANALYST	10
DATE	PROGRAMMER	10

Ακολουθεί τμηματική εκτέλεση της δήλωσης ώστε να την κατανοήσουμε καλύτερα.

```
SELECT DEPTNO
FROM  DEPT
WHERE  DNAME IN ('SALES', 'ACCOUNTING');
```

DEPTNO
10
30

```
SELECT JOB
FROM   EMP
WHERE  DEPTNO IN (10, 30);
```

JOB
ANALYST
ANALYST
PROGRAMMER

```
SELECT ENAME, JOB, DEPTNO
FROM EMP
WHERE JOB IN ('ANALYST', 'PROGRAMMER');
```

ENAME	JOB	DEPTNO
CODD	ANALYST	10
ELMASRI	ANALYST	10
DATE	PROGRAMMER	10

Παράδειγμα 13

Δείξτε τους υπάλληλους με έδρα NEW YORK ή BOSTON και θέση σαν τη θέση υπαλλήλου με το όνομα CODD. Θυμηθείτε ότι μπορεί να υπάρχουν πολλοί υπάλληλοι με το ίδιο όνομα.

Βλέπετε πρώτα τα στοιχεία των τμημάτων και των υπαλλήλων.

```
SELECT * FROM dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

```
SELECT * FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	15	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	15	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	20	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	15	04/05/2004	1800	200	10

```
SELECT ENAME,LOC,SAL,JOB, EMP.DEPTNO
FROM EMP,DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO
AND (LOC = 'NEW YORK' OR LOC = 'BOSTON ')
AND JOB IN (SELECT JOB
FROM EMP
WHERE ENAME = 'CODD')
ORDER BY ENAME;
```

ENAME	LOC	SAL	JOB	DEPTNO
CODD	NEW YORK	3000	ANALYST	10
ELMASRI	NEW YORK	1200	ANALYST	10

Παράδειγμα 14

Ποιοί υπάλληλοι κερδίζουν περισσότερα από το μέσο μισθό των υπαλλήλων του τμήματός τους.

```
SELECT DEPTNO,ENAME,SAL
FROM EMP X
WHERE SAL >
   (SELECT AVG(SAL)
FROM   EMP
WHERE  X.DEPTNO = DEPTNO)
ORDER BY DEPTNO;
```

DEPTNO	ENAME	SAL
10	CODD	3000

3.4.9 Πως σχηματίζουμε συνθήκες που περιλαμβάνουν πολλά επίπεδα υποαναζητήσεων.

Για το σχηματισμό συνθήκης σε αναζήτηση μπορείς να χρησιμοποιήσεις πολλά επίπεδα υποαναζητήσεων (πολλές «φωλιασμένες» δηλώσεις SELECT, τη μία μέσα στην άλλη).

Παράδειγμα

Η παρακάτω αναζήτηση βρίσκει ποιοί υπάλληλοι έχουν την ίδια θέση με κάποιο υπάλληλο του τμήματος ACCOUNTING.

```
SELECT  ENAME, JOB
 FROM EMP
 WHERE JOB IN (SELECT  JOB
   FROM  EMP
   WHERE  DEPTNO IN
(SELECT DEPTNO
 FROM DEPT
 WHERE  DNAME = 'ACCOUNTING'));
```

ENAME	JOB
ELMASRI	ANALYST
CODD	ANALYST
DATE	PROGRAMMER

3.4.10 Αναζήτηση στοιχείων με διάταξη των αποτελεσμάτων - υποπρόταση ORDER BY.

Η συνηθισμένη μορφή σύνταξης της δήλωσης SELECT που ταξινομεί τα αποτελέσματα είναι:

```
  SELECT  ονόματα στηλών ή αριθμητικές εκφράσεις των στηλών ή ..
  FROM όνομα πίνακα ή ονόματα πινάκων ή ....
  WHERE  συνθήκη
  ORDER BY  όνομα_πίνακα.όνομα_στήλης ASC ή DESC ή τίποτα,
   όνομα_πίνακα.όνομα_στήλης κ.λπ.;
```

Δηλαδή, ο προγραμματιστής μπορεί να επιλέξει πολλά επίπεδα ταξινόμησης κατά αύξουσα σειρά (όταν η αντίστοιχη στήλη συνοδεύεται από ASC ή δεν συνοδεύεται από κάποια δήλωση) ή φθίνουσα σειρά (όταν η αντίστοιχη στήλη συνοδεύεται από δήλωση DESC).

Για χαρακτήρες η ταξινόμηση ακολουθεί τη σειρά κατάταξης χαρακτήρων του πίνακα ASCII. Έτσι το 0 προηγείται του 9, το A του Z κ.λπ.

Στην περίπτωση του προϊόντος της ORACLE οι τιμές NULL πάντοτε εμφανίζονται τελευταίες στα αποτελέσματα και απαιτείται τέχνασμα με χρήση συνάρτησης NVL-Null Value για να ανατραπεί το γεγονός αυτό. Τι συμβαίνει στην περίπτωση του προϊόντος της mySQL;

Στην υποπρόταση ORDER BY μπορούμε να χρησιμοποιήσουμε και εκφράσεις των στηλών με πράξεις, παρενθέσεις κ.λπ. ή και τον αύξοντα αριθμό ενός χαρακτηριστικού (attribute) στον ορισμό του πίνακα.

π.χ. ... ORDER BY COMM/SAL

... ORDER BY 3 (αντί του ORDER BY JOB)

Παράδειγμα 1

Η παρακάτω αναζήτηση ταξινομεί τους υπαλλήλους κατά Τμήμα και φθίνουσα τάξη μισθού.

```
SELECT  ENAME, DEPTNO, JOB, SAL
FROM  EMP
ORDER BY DEPTNO, SAL DESC;
```

ENAME	DEPTNO	JOB	SAL
CODD	10	ANALYST	3000
DATE	10	PROGRAMMER	1800
ELMASRI	10	ANALYST	1200
NAVATHE	20	SALESMAN	2000

Παράδειγμα 2

Δείξτε τους υπάλληλους του Τμήματος 10 ταξινομημένους ανά μισθό.

```
SELECT SAL,JOB,ENAME
FROM EMP
WHERE DEPTNO = 10
ORDER BY SAL;
```

SAL	JOB	ENAME
1200	ANALYST	ELMASRI
1800	PROGRAMMER	DATE
3000	ANALYST	CODD

Παράδειγμα 3

Δείξτε τους υπάλληλους του Τμήματος 10 ταξινομημένους ανά φθίνουσα σειρά μισθού.

```
SELECT SAL,JOB,ENAME
FROM EMP
WHERE DEPTNO = 10
ORDER BY SAL DESC;
```

SAL	JOB	ENAME
3000	ANALYST	CODD

1800	PROGRAMMER	DATE
1200	ANALYST	ELMASRI

Παράδειγμα 4

Να ταξινομήσετε τους υπαλλήλους ανά θέση, Οι υπάλληλοι που έχουν την ίδια θέση να ταξινομηθούν ανά φθίνουσα τάξη μισθού.

```
SELECT JOB,SAL,ENAME
FROM EMP
ORDER BY JOB,SAL DESC;
```

JOB	SAL	ENAME
ANALYST	3000	CODD
ANALYST	1200	ELMASRI
PROGRAMMER	1800	DATE
SALESMAN	2000	NAVATHE

Παράδειγμα 5

Να διατάξετε τους υπαλλήλους του τμήματος 10 ανάλογα με τη προμήθειά τους.

```
SELECT COMM,SAL,ENAME
FROM EMP
WHERE DEPTNO = 10
ORDER BY COMM;
```

COMM	SAL	ENAME
150	1200	ELMASRI
200	1800	DATE
-	3000	CODD

Ερώτηση

Τι θα συμβεί αν χρησιμοποιήσετε ORDER BY NVL(COMM,0);

Παράδειγμα 6

Να διατάξετε τους υπαλλήλους του τμήματος 10 ανάλογα με τη προμήθεια τους και επιπλέον κατά φθίνουσα τάξη.

```
SELECT COMM,SAL,ENAME
FROM EMP
WHERE DEPTNO = 10
ORDER BY COMM DESC;
```

COMM	SAL	ENAME
-	3000	CODD
200	1800	DATE
150	1200	ELMASRI

Ερώτηση

Τι θα συμβεί αν χρησιμοποιήσετε ORDER BY NVL(COMM,0);

MySQL:

COMM	SAL	ENAME
200	1800	DATE
150	1200	ELMASRI
-	3000	CODD

3.4.11 Αναζήτηση στοιχείων με ομαδοποίηση αποτελεσμάτων. Υποπρόταση GROUP BY.

Η συνηθισμένη σύνταξη μίας δήλωσης SELECT που ομαδοποιεί τα αποτελέσματα είναι:

```
SELECT   στήλες ή .... (Προσοχή! Υπάρχουν σοβαροί περιορισμοί.)
FROM όνομα_πίνακα ή ονόματα πινάκων ή ....
GROUP BY όνομα_πίνακα.όνομα_στήλης;
```

Βέβαια υπάρχει δυνατότητα ομαδοποίησης με περισσότερα κριτήρια, πχ.,

.... GROUP BY EMP.JOB , EMP.SAL ... ;

Παράδειγμα

Η παρακάτω αναζήτηση ομαδοποιεί τους υπαλλήλους ανά τμήμα και βρίσκει το μέσο όρο μισθού.

```
SELECT DEPTNO, AVG(SAL)
 FROM  EMP
 GROUP BY DEPTNO;
```

DEPTNO	AVG(SAL)
20	2000
10	2000

3.4.12 Αναζήτηση στοιχείων με ομαδοποίηση αποτελεσμάτων και περιορισμό των στηλών. Χρήση της υποπρότασης HAVING σε συνδυασμό με υποπρόταση GROUP BY.

Η συνηθισμένη σύνταξη της δήλωσης SELECT που ομαδοποιεί τα αποτελέσματα και περιορίζει τις στήλες των αποτελεσμάτων είναι:

```
SELECT   ονόματα στηλών ή ....   (Προσοχή! Υπάρχουν
  σοβαροί περιορισμοί στην επιλογή στηλών)
FROM όνομα_πίνακα ή ονόματα πίνακων ή ....
GROUP BY ονόματα στηλών
HAVING   συνθήκη;
```

Παράδειγμα 1

Η παρακάτω αναζήτηση βρίσκει τον μέσο όρο μισθού υπαλλήλων που κάνουν την ίδια εργασία όταν ο μέσος μισθός τους (των υπαλλήλων που κάνουν την ίδια εργασία) ξεπερνά τα 1250 ευρώ.

```
SELECT  JOB, AVG(SAL)
 FROM  EMP
 GROUP BY JOB
 HAVING AVG(SAL) > 1250;
```

JOB	AVG(SAL)
SALESMAN	2000
ANALYST	2100
PROGRAMMER	1800

Παράδειγμα 2

Η παρακάτω αναζήτηση βρίσκει το μέσο μισθό αυτών που κάνουν την ίδια εργασία όταν είναι τουλάχιστον δύο.

```
SELECT  JOB, AVG(SAL), COUNT(*)
 FROM  EMP
 GROUP BY JOB
 HAVING COUNT(*) >= 2;
```

JOB	AVG(SAL)	COUNT(*)
ANALYST	2100	2

Προσοχή! Η SELECT JOB, SAL ... δίνει συντακτικό λάθος επειδή απαγορεύεται η χρήση στήλης στην υποπρόταση SELECT αν δεν αναφέρεται στο GROUP BY ή αν δεν χρησιμοποιείται σαν όρισμα συνάρτησης ομαδοποίησης.

Ειδικά στην περίπτωση MySQL δεν έχουμε μήνυμα λάθους αλλά τα αποτελέσματα είναι λανθασμένα.

3.4.13 Αναζήτηση στοιχείων που απαιτεί σύνδεση δύο πινάκων.

Η συνηθισμένη σύνταξη αναζήτησης που συνδέει δύο πίνακες είναι:

```
SELECT  όνομα_πίνακα1.όνομα_στήλης, ....
όνομα_πίνακα2.όνομα_στήλης, ....
FROM όνομα_πίνακα1 , όνομα_πίνακα2
WHERE   όνομα_πίνακα1.όνομα_στήλης =
όνομα_πίνακα2.όνομα_στήλης  κτλ.
```

Παράδειγμα 1

Η παρακάτω δήλωση βρίσκει όνομα, μισθό και έδρα των πωλητών.

```
SELECT  ENAME, JOB, SAL, LOC
 FROM  EMP,DEPT
 WHERE  JOB = 'SALESMAN'
 AND  EMP.DEPTNO = DEPT.DEPTNO;
```

ENAME	JOB	SAL	LOC
NAVATHE	SALESMAN	2000	DALLAS

Επεξήγηση

Η συνθήκη σύνδεσης (join) EMP.DEPTNO=DEPT.DEPTNO εξασφαλίζει τη λογική (conceptually) "ενοποίηση" των δύο πινάκων. Η παρακάτω δήλωση δεν περιλαμβάνει κάποια συνθήκη και έχει ως αποτέλεσμα ακριβώς αυτήν την «ενοποίηση» των δύο πινάκων:

```
SELECT  *
FROM  EMP,DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO;
```

Παραθέτουμε το αποτέλεσμα της δήλωσης αυτής.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
30	DATE	PROGRAMMER	15	04/05/2004	1800	200	10	10	ACCOUNTING	NEW YORK
15	ELMASRI	ANALYST	15	02/05/1995	1200	150	10	10	ACCOUNTING	NEW YORK
10	CODD	ANALYST	15	01/01/1989	3000	-	10	10	ACCOUNTING	NEW YORK
20	NAVATHE	SALESMAN	20	07/07/1977	2000	-	20	20	RESEARCH	DALLAS

Η τελική δήλωση επιλέγει τις στήλες ENAME, JOB, SAL, LOC για τους πωλητές (salesman) «βασιζόμενη» στον πίνακα αυτό.

Προσοχή! Δεν είναι απαραίτητο οι πίνακες που θα συνδεθούν να έχουν κοινή στήλη ή τις ίδιες τιμές (values) στις στήλες της σύνδεσης. Αρκεί να έχουν στήλες με κοινό πεδίο ορισμού. Για παράδειγμα, οι πίνακες που είδαμε παραπάνω:

CUSTOMER(cust_name, cust_city)

SUPPLIER(suppl_name, suppl_city)

```
CREATE TABLE customer(cust_name CHAR(15),
  cust_city CHAR(15));
CREATE TABLE supplier(suppl_name CHAR(15),
   suppl_city CHAR(15));

INSERT INTO customer VALUES('CODD', 'NEW YORK');
INSERT INTO customer VALUES('DATE', 'BERLIN');

INSERT INTO supplier VALUES('NAVATHE', 'NEW YORK');
INSERT INTO supplier VALUES('ELMASRI', 'ATHENS');

SELECT * FROM supplier;
```

SUPPL_NAME	SUPPL_CITY
NAVATHE	NEW YORK
ELMASRI	ATHENS

```
SELECT * FROM customer;
```

CUST_NAME	CUST_CITY
CODD	NEW YORK
DATE	BERLIN

Η παρακάτω αναζήτηση βρίσκει ποιοί πελάτες (customers) έχουν έδρα την ίδια πόλη με κάποιο προμηθευτή (supplier):

```
SELECT *
FROM customer, supplier
WHERE customer.cust_city=supplier.suppl_city;
```

CUST_NAME	CUST_CITY	SUPPL_NAME	SUPPL_CITY
CODD	NEW YORK	NAVATHE	NEW YORK

Βλέπουμε ότι οι πίνακες μπορούν να συνδεθούν με συνθήκη

customer.cust_city=supplier.suppl_city

επειδή οι δύο στήλες έχουν το ίδιο πεδίο ορισμού.

Σε MySQL:

```
CREATE TABLE customer(cust_name CHAR(15),
  cust_city CHAR(15));
CREATE TABLE supplier(suppl_name CHAR(15),
   suppl_city CHAR(15));

INSERT INTO customer VALUES('CODD', 'NEW YORK');
INSERT INTO customer VALUES('DATE', 'BERLIN');

INSERT INTO supplier VALUES('NAVATHE', 'NEW YORK');
INSERT INTO supplier VALUES('ELMASRI', 'ATHENS');

SELECT * FROM supplier;
SELECT * FROM customer;

SELECT *
FROM customer, supplier
WHERE customer.cust_city=supplier.suppl_city;

DROP TABLE customer;
DROP TABLE supplier;
```

Προσοχή! Η παρακάτω δήλωση υπολογίζει το καρτεσιανό γινόμενο.

```
SELECT emp.ename, emp.sal, dept.loc
from  emp, dept;
```

ENAME	SAL	LOC
CODD	3000	NEW YORK
CODD	3000	DALLAS
CODD	3000	CHICAGO
CODD	3000	BOSTON
ELMASRI	1200	NEW YORK
ELMASRI	1200	DALLAS

ELMASRI	1200	CHICAGO
ELMASRI	1200	BOSTON
NAVATHE	2000	NEW YORK
NAVATHE	2000	DALLAS
NAVATHE	2000	CHICAGO
NAVATHE	2000	BOSTON
DATE	1800	NEW YORK
DATE	1800	DALLAS
DATE	1800	CHICAGO
DATE	1800	BOSTON

Δεν μπορεί να θεωρηθεί σαν εναλλακτική της αντίστοιχης αναζήτησης που περιλαμβάνει join γιατί υπολογίζει το καρτεσιανό γινόμενο των δύο πινάκων γεγονός που οδηγεί στην εμφάνιση στοιχείων που δεν έχουν πραγματική σημασία, π.χ. κάθε υπάλληλος εμφανίζεται να έχει έδρα όλες τις πόλεις που είναι έδρες τμημάτων.

Παράδειγμα 2

Βρείτε όλους τους CODD από τον πίνακα EMP και τη θέση της έδρας τους από τον πίνακα DEPT .

```
SELECT ENAME,LOC
FROM EMP,DEPT
WHERE ENAME = 'CODD'
AND EMP.DEPTNO = DEPT.DEPTNO;
```

ENAME	LOC
CODD	NEW YORK

Παράδειγμα 3

Για κάθε τμήμα βρείτε κωδικό αριθμό, όνομα, τους υπαλλήλους του και τη θέση κάθε υπαλλήλου του. Διάταξε τα αποτελέσματα κατά αύξοντα αριθμό τμήματος.

```
SELECT DEPT.DEPTNO,DNAME,ENAME,JOB
FROM DEPT,EMP
WHERE DEPT.DEPTNO = EMP.DEPTNO
ORDER BY DEPT.DEPTNO;
```

DEPTNO	DNAME	ENAME	JOB
10	ACCOUNTING	CODD	ANALYST
10	ACCOUNTING	DATE	PROGRAMMER
10	ACCOUNTING	ELMASRI	ANALYST
20	RESEARCH	NAVATHE	SALESMAN

Παράδειγμα 4

Βρείτε τμήματα με υπαλλήλους αλλά και τμήματα χωρίς υπαλλήλους. Ταξινομήστε κατά αύξοντα αριθμό τμήματος.

```
SELECT DEPT.DEPTNO,DNAME,ENAME,JOB
FROM DEPT,EMP
WHERE DEPT.DEPTNO = EMP.DEPTNO(+)
ORDER BY DEPT.DEPTNO;
```

DEPTNO	DNAME	ENAME	JOB
10	ACCOUNTING	CODD	ANALYST
10	ACCOUNTING	ELMASRI	ANALYST
10	ACCOUNTING	DATE	PROGRAMMER
20	RESEARCH	NAVATHE	SALESMAN
30	SALES	-	-
40	OPERATIONS	-	-

LEFT JOIN σε MySQL:

Πρώτα δίδεται μια γενική μορφή σύνδεσης πινάκων:

```
SELECT * FROM t1
LEFT JOIN (t2)
ON (t2.a=t1.a);
```

Ακολουθεί δήλωση SELECT η οποία περιλαμβάνει LEFT JOIN:

```
SELECT DEPT.DEPTNO, DNAME,ENAME, JOB FROM DEPT
 LEFT JOIN EMP ON (DEPT.DEPTNO = EMP.DEPTNO)
ORDER BY DEPT.DEPTNO;
```

Παράδειγμα 5

Βρείτε τμήματα χωρίς υπαλλήλους.

Σε Oracle:

```
SELECT DEPT.DEPTNO,DNAME,LOC
FROM DEPT,EMP
WHERE DEPT.DEPTNO = EMP.DEPTNO(+)
AND EMPNO IS NULL;
```

DEPTNO	DNAME	LOC
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Σε Oracle και MySQL:

```
SELECT DEPT.DEPTNO,DNAME,ENAME,JOB FROM DEPT
 LEFT JOIN EMP ON (DEPT.DEPTNO = EMP.DEPTNO)
WHERE EMPNO IS NULL
ORDER BY DEPT.DEPTNO;
```

Παράδειγμα 6. Συνηθισμένες συνδέσεις

Δείξε όνομα Τμήματος και όλες τις πληροφορίες για τους υπάλληλους που εργάζονται σε New York.

```
SELECT DNAME,EMPNO,ENAME,JOB,MGR,HIREDATE,
   SAL,COMM,EMP.DEPTNO
FROM EMP,DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO
 AND LOC = 'NEW YORK'
ORDER BY EMP.DEPTNO;
```

DNAME	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
ACCOUNTING	10	CODD	ANALYST	15	01/01/1989	3000	-	10
ACCOUNTING	30	DATE	PROGRAMMER	15	04/05/2004	1800	200	10
ACCOUNTING	15	ELMASRI	ANALYST	15	02/05/1995	1200	150	10

ή εναλλακτικά με χρήση συνωνύμων για τους πίνακες

```
SELECT DNAME,E.*
FROM EMP E,DEPT D
WHERE E.DEPTNO = D.DEPTNO
 AND LOC = 'NEW YORK'
ORDER BY E.DEPTNO;
```

Παράδειγμα 7

Βρείτε υπαλλήλους με μισθό μεγαλύτερο από τον επικεφαλής τους.

```
SELECT WORKER.ENAME,WORKER.SAL,
   MANAGER.ENAME,MANAGER.SAL
FROM EMP WORKER,EMP MANAGER
WHERE WORKER.MGR = MANAGER.EMPNO
AND WORKER.SAL > MANAGER.SAL;
```

ENAME	SAL	ENAME	SAL
DATE	1800	ELMASRI	1200
CODD	3000	ELMASRI	1200

Παράδειγμα 8

Ποιοί υπάλληλοι κερδίζουν περισσότερα από τους υπάλληλους που ονομάζονται DATE.

```
SELECT X.ENAME,X.SAL,X.JOB,Y.ENAME,Y.SAL,Y.JOB
FROM EMP X,EMP Y
WHERE X.SAL > Y.SAL
AND Y.ENAME = 'DATE';
```

ENAME	SAL	JOB	ENAME	SAL	JOB
CODD	3000	ANALYST	DATE	1800	PROGRAMMER
NAVATHE	2000	SALESMAN	DATE	1800	PROGRAMMER

3.4.14 Αναζήτηση στοιχείων (SELECT) που απαιτεί σύνδεση περισσότερων από δύο πινάκων.

Η σύνδεση (join) δύο πινάκων μέσω στηλών τους με κοινό πεδίο ορισμού γενικεύεται ως εξής:

Αρχικά, καθορίζονται οι πίνακες των οποίων στήλες μας ενδιαφέρουν. Στη συνέχεια ο προγραμματιστής γράφει απλές συνδέσεις των πινάκων ανά δύο και κατασκευάζει μια σύνθετη συνθήκη χρησιμοποιώντας τους κατάλληλους κατά περίπτωση τελεστές Boole και παρενθέσεις (όταν είναι απαραίτητο).

Παράδειγμα

Έστω ότι το σύστημα διαχείρισης παραγγελιών περιλαμβάνει τέσσερις πίνακες.

- Πίνακας με στοιχεία πελατών: CUSTOMERS (CUSTNO, CNAME, LOC)
- Μητρώο προϊόντων: STOCKS (STOCKNO, DESCRIPTION, LIST_PRICE)
- Πίνακας με βασικά στοιχεία παραγγελίας: ORDERS (ORDERNO, CUSTNO, ODATE, TOTAL)
- Πίνακας με στοιχεία γραμμών παραγγελίας: ORDERLINES (ORDERNO, STOCKNO, QTY, PTOTAL)

```
SELECT * FROM customers;
```

CUSTNO	CNAME	CADDRESS
1	SMITH	ATHENS
2	JONES	VOLOS
3	BATES	NEW YORK

```
SELECT * FROM stocks;
```

STOCKNO	DESCR	LISTPRICE
1	APPLE	1
2	ORANGE	1.5
3	LEMON	1.7

```
SELECT * FROM orders;
```

ORDERNO	CUSTNO	ODATE	TOTAL
1	1	10/10/2012	17.5

```
SELECT * FROM orderlines;
```

ORDERNO	STOCKNO	QTY	PTOTAL
1	1	10	10
1	2	5	7.5

Δημιουργήστε τους πίνακες της βάσης. Προσοχή! Έχει σημασία η σειρά δημιουργίας των πινάκων!

```
CREATE TABLE customers(custno NUMBER(3) NOT NULL,
cname VARCHAR2(10), caddress VARCHAR2(15),
PRIMARY KEY(custno));
```

```
CREATE TABLE stocks(stockno NUMBER(3) NOT NULL,
descr VARCHAR2(10), listprice NUMBER(7,2),
PRIMARY KEY (stockno));

CREATE TABLE orders(orderno NUMBER NOT NULL,
custno NUMBER(3), odate DATE, total NUMBER(9,2),
PRIMARY KEY (orderno),
  FOREIGN KEY (custno) REFERENCES customers(custno));

CREATE TABLE orderlines(orderno NUMBER NOT NULL,
stockno NUMBER (3) NOT NULL, qty NUMBER(2),
ptotal NUMBER (8,2), PRIMARY KEY (orderno,stockno),
FOREIGN KEY(orderno) REFERENCES orders(orderno),
FOREIGN KEY (stockno) REFERENCES stocks (stockno));
```

Εισάγετε στοιχεία στους πίνακες. Προσοχή! Έχει σημασία η σειρά αναγραφής των δηλώσεων εισαγωγής γραμμών.

Δοκιμάστε και λανθασμένες δηλώσεις εισαγωγής στοιχείων. Για παράδειγμα δοκιμάστε να εισάγετε τιμές μεγαλύτερου μήκους από αυτό που ορίσατε κατά τη δημιουργία των πινάκων ώστε να δείτε τα μηνύματα του συστήματος. Δοκιμάστε να εισάγετε γραμμή στον πίνακα των βασικών στοιχείων παραγγελίας που να περιλαμβάνει τιμή για τον κωδικό του πελάτη που δεν έχει ήδη εισαχθεί στον πίνακα πελατών. Το ίδιο για γραμμή παραγγελίας που περιλαμβάνει είδος που δεν υπάρχει στο μητρώο ειδών.

```
INSERT INTO customers VALUES(1, 'SMITH', 'ATHENS');
INSERT INTO customers VALUES(2, 'JONES', 'VOLOS');
INSERT INTO customers VALUES(3, 'BATES', 'NEW YORK');

INSERT INTO stocks VALUES(1, 'APPLE', 1);
INSERT INTO stocks VALUES(2, 'ORANGE', 1.5);
INSERT INTO stocks VALUES(3, 'LEMON', 1.7);

INSERT INTO orders VALUES(1, 1, '10/10/2012', 17.5);

INSERT INTO orderlines VALUES(1, 1, 10, 10);
INSERT INTO orderlines VALUES(1, 2, 5, 7.5);
```

Η παρακάτω δήλωση SELECT βρίσκει ποιοί πελάτες παράγγειλαν είδος σε ποσότητα μεγαλύτερη του 5.

```
SELECT  ORDERS.ORDERNO, ORDERS.CUSTNO, CNAME, CADDRESS,
STOCKS.STOCKNO, DESCR, QTY
FROM CUSTOMERS, ORDERS, ORDERLINES, STOCKS
WHERE CUSTOMERS.CUSTNO = ORDERS.CUSTNO
AND  ORDERS.ORDERNO = ORDERLINES.ORDERNO
AND ORDERLINES.QTY > 5;
```

ORDERNO	CUSTNO	CNAME	CADDRESS	STOCKNO	DESCR	QTY
1	1	SMITH	ATHENS	1	APPLE	10
1	1	SMITH	ATHENS	2	ORANGE	10
1	1	SMITH	ATHENS	3	LEMON	10

Προσοχή! Δίχως να είναι αναγκαίο, είναι σκόπιμο η σειρά που θα γράψετε τις συνδέσεις των πινάκων ανά δύο να μην είναι τυχαία! Αφού "ενοποιήσετε" λογικά τους δύο πρώτους συνεχίζεις με την "ενσωμάτωση" του επόμενου κ.λπ.

Γράψτε δηλώσεις διαγραφής των πινάκων. Η σειρά διαγραφής δεν είναι τυχαία. Αν αρχικά προσπαθήσω να εκτελέσω τη δήλωση drop table customers τι θα συμβεί; Γιατί; Τι πρέπει να κάνω για να εκτελεστεί η δήλωση drop table stocks;

Ακολουθεί μία ορθή σειρά διαγραφής των πινάκων:

```
DROP TABLE orderlines;
DROP TABLE orders;
DROP TABLE stocks;
DROP TABLE customers;
```

Παραθέτουμε τις ίδιες δηλώσεις στην περίπτωση του προϊόντος mySQL

```
CREATE TABLE customers(custno INT(3) NOT NULL,
cname VARCHAR(10), caddress VARCHAR(15),
PRIMARY KEY(custno));

CREATE TABLE stocks(stockno INT(3) NOT NULL,
descr VARCHAR(10), listprice FLOAT(7,2),
PRIMARY KEY (stockno));

CREATE TABLE orders(orderno INT(4) NOT NULL,
custno INT(3), odate DATE, total FLOAT(9,2),
PRIMARY KEY (orderno),
  FOREIGN KEY (custno) REFERENCES customers(custno));

CREATE TABLE orderlines(orderno INT(4) NOT NULL,
stockno INT(3) NOT NULL, qty INT(2),
ptotal FLOAT(8,2), PRIMARY KEY (orderno,stockno),
FOREIGN KEY(orderno) REFERENCES orders(orderno),
FOREIGN KEY (stockno) REFERENCES stocks (stockno));

INSERT INTO customers VALUES(1, 'SMITH', 'ATHENS');
INSERT INTO customers VALUES(2, 'JONES', 'VOLOS');
INSERT INTO customers VALUES(3, 'BATES', 'NEW YORK');

INSERT INTO stocks VALUES(1, 'APPLE', 1);
INSERT INTO stocks VALUES(2, 'ORANGE', 1.5);
INSERT INTO stocks VALUES(3, 'LEMON', 1.7);

INSERT INTO orders VALUES(1, 1, '2012/10/10', 17.5);
INSERT INTO orderlines VALUES(1, 1, 10, 10);
INSERT INTO orderlines VALUES(1, 2, 5, 7.5);

SELECT * FROM CUSTOMERS;
SELECT * FROM STOCKS;
SELECT * FROM ORDERS;
SELECT * FROM ORDERLINES;

SELECT  ORDERS.ORDERNO, ORDERS.CUSTNO, CNAME, CADDRESS,STOCKS.STOCKNO, DESCR,
QTY
FROMCUSTOMERS, ORDERS, ORDERLINES, STOCKS
WHERE CUSTOMERS.CUSTNO = ORDERS.CUSTNO
AND  ORDERS.ORDERNO = ORDERLINES.ORDERNO AND ORDERLINES.QTY > 5;

DROP TABLE ORDERLINES;
DROP TABLE ORDERS;
```

```
DROP TABLE CUSTOMERS;
DROP TABLE STOCKS;
```

3.4.15 Αναζήτηση στοιχείων που απαιτεί σύνδεση πίνακα με τον εαυτό του.

Στην περίπτωση αυτή χρησιμοποιούμε μια τεχνική αναγραφής και χρήσης συνωνύμων του πίνακα. Η τεχνική αυτή μπορεί να χρησιμοποιηθεί και γενικότερα φυσικά.

Παράδειγμα 1

Η παρακάτω αναζήτηση βρίσκει ποιοί είναι οι υπάλληλοι που παίρνουν μεγαλύτερο μισθό από τον επικεφαλής τους.

```
SELECT  WORKER.ENAME WORKER,WORKER.SAL, MANAGER.ENAME MANAGER, MANAGER.SAL
 FROMEMP WORKER, EMP MANAGER
 WHERE   WORKER.MGR = MANAGER.EMPNO
 AND WORKER.SAL > MANAGER.SAL;
```

WORKER	SAL	MANAGER	SAL
DATE	1800	ELMASRI	1200
CODD	3000	ELMASRI	1200

3.4.16 Αναζητήσεις που απαιτούν σύνδεση πινάκων και χρήση τελεστών σύγκρισης.

Ακολουθούν παραδείγματα.

Παράδειγμα 1

Η παρακάτω αναζήτηση συνδέει ένα πίνακα με τον εαυτό του χρησιμοποιώντας τον τελεστή σύγκρισης > και βρίσκει ποιοί υπάλληλοι έχουν μισθό μεγαλύτερο από τον DATE.

```
SELECT  X.ENAME, X.SAL, X.JOB,
 Y.ENAME, Y.SAL, Y.JOB
 FROM  EMP  X , EMP Y
 WHERE  X.SAL > Y.SAL
 AND  Y.ENAME = 'DATE';
```

ENAME	SAL	JOB	ENAME	SAL	JOB
CODD	3000	ANALYST	DATE	1800	PROGRAMMER
NAVATHE	2000	SALESMAN	DATE	1800	PROGRAMMER

3.4.17 Χρήσιμες συναρτήσεις και η συνηθισμένη σύνταξή τους.

Θα αναφερθούν μερικές πολύ χρήσιμες συναρτήσεις και η συνηθισμένη σύνταξη τους. Αρχικά θα αναφερθούμε σε συναρτήσεις του προϊόντος της Oracle και στον τρόπο χρήσης:

Συνάρτηση	Παράδειγμα
NVL-διαχείριση null value στήλης	NVL (column-name, integer)
POWER-ύψωση σε δύναμη	POWER (column-name, integer)

ROUND-στρογγύλευση	ROUND (column-name, integer)
TRUNC-αποκοπή	TRUNC (column-name, integer)
DECODE-αποκωδικοποίηση	DECODE(column-name, "value1", number1, "value2", number2, ...)
SUBSTR-καθορισμός τμήματος συμβολοσειράς (substring)	SUBSTR (column, nstart, nend)
UPPER-μεταγραφή string σε κεφαλαία	UPPER (column)
LOWER-μεταγραφή string σε πεζά	LOWER (column)
LENGTH-μήκος string	LENGTH (column)
INSTR-εισαγωγή χαρακτήρα σε string	INSTR (column, "letter", position)
AVG-μέσος όρος τιμών	AVG (column)
SUM-άθροισμα τιμών	SUM (column)
MAX-μέγιστη τιμή	MAX (column)
MIN-ελάχιστη τιμή	MIN (column)
COUNT-πλήθος γραμμών	COUNT (column) ή COUNT (DISTINCT column) ή COUNT(*)

Σημειώνουμε ότι αντί του column-name μπορείτε να χρησιμοποιήσετε εκφράσεις με αριθμητικούς τελεστές, παρενθέσεις, τελεστή DISTINCT. Επιπλέον, σε πολλές περιπτώσεις αντί του column στις συναρτήσεις ομαδοποίησης AVG, SUM, MAX, MIN χρησιμοποιούμε DISTINCT column.

3.4.18 Συναρτήσεις ομαδοποίησης (group functions) - AVG-SUM-MAX-MIN-COUNT

Οι συνηθισμένες συναρτήσεις ομαδοποίησης υπάρχουν σε όλα τα γνωστά προϊόντα και είναι οι εξής:

- AVG- μέσος όρος
- SUM-άθροισμα
- MAX-μέγιστη τιμή
- MIN-ελάχιστη τιμή
- COUNT-πλήθος γραμμών

Η συνηθισμένη σύνταξη είναι η εξής:

```
group-function ( [DISTINCT/ALL] {column / expression} )
```

Ακολουθούν παραδείγματα χρήσης συναρτήσεων ομαδοποίησης. Βασίζονται στον παρακάτω πίνακα:

EMPNO	ENAME	JOB	SAL	COMM	DEPTNO
10	CODD	ANALYST	3000	-	10
15	ELMASRI	ANALYST	1200	150	10
20	NAVATHE	SALESMAN	2000	-	20
30	DATE	PROGRAMMER	1800	200	10

Η δήλωση

```
SELECT  deptno, AVG(sal)
FROM  emp
GROUP BY  deptno
HAVING  AVG (sal) > 1200;
```

υπολογίζει το μέσο μισθό για κάθε τμήμα με μέσο μισθό μεγαλύτερο των 1200.

DEPTNO	AVG(SAL)
10	2000
20	2000

Η δήλωση υπολογίζει μέγιστο, ελάχιστο μισθό υπαλλήλου και την διαφορά τους.

```
SELECT  MAX(sal), MIN(sal), MAX(sal)-MIN(sal)
FROM  emp;
```

MAX(SAL)	MIN(SAL)	MAX(SAL)-MIN(SAL)
3000	1200	1800

Δείτε και την παρακάτω αναζήτηση.

```
SELECT  MAX(comm), MIN(comm), MAX(comm) - MIN(comm)
FROM  emp;
```

MAX(COMM)	MIN(COMM)	MAX(COMM)-MIN(COMM)
200	150	50

Ειδική μνεία πρέπει να γίνει για τη συνάρτηση COUNT. Η σύνταξη της COUNT πέρα από τα ισχύοντα για τις άλλες συναρτήσεις ομαδοποίησης δέχεται και τη μορφή COUNT(*). Στη συνέχεια θα αναφέρουμε τη διαφορά των πιθανών μορφών της συνάρτησης COUNT:

- Η συνάρτηση COUNT (column) μετρά τις τιμές που δεν είναι NULL του χαρακτηριστικού column.
- Η συνάρτηση COUNT (DISTINCT column) μετρά τις διακεκριμένες τιμές
- Η συνάρτηση COUNT(*) μετρά όλες τις τιμές.

Παραδείγματα χρήσης της COUNT σε δηλώσεις SELECT:

Η δήλωση υπολογίζει διακεκριμένες θέσεις υπαλλήλων.

```
SELECT  COUNT(DISTINCT  job)   FROM  emp;
```

COUNT(DISTINCTJOB)
3

Δείτε και τη δήλωση

```
SELECT  COUNT(job) FROM  emp;
```

COUNT(JOB)
4

Η δήλωση

```
SELECT  deptno, AVG(sal),COUNT(*)
FROM  emp
GROUP BY deptno
HAVING  AVG(sal) >= 1200
ORDER BY  deptno;
```

εμφανίζει για τα τμήματα τα οποία έχουν μέσο μισθό υπαλλήλων πάνω από 1200, τον μέσο μισθό των υπαλλήλων του τμήματος και τον αριθμό γραμμών που ανακτήθηκαν, δηλαδή πόσοι είναι οι υπάλληλοι του τμήματος.

DEPTNO	AVG(SAL)	COUNT(*)
10	2000	3
20	2000	1

Προσοχή! Θυμηθείτε ότι στην περίπτωση δήλωσης SELECT με υποπρόταση GROUP BY δεν επιτρέπεται να χρησιμοποιηθεί όνομα χαρακτηριστικού αν δεν περιλαμβάνεται στην υποπρόταση GROUP BY. Βεβαίως, επιτρέπεται η χρήση συνάρτησης ομαδοποίησης του χαρακτηριστικού, όπως είδαμε στα παραδείγματα, Δηλαδή, δεν επιτρέπεται δήλωση

```
SELECT deptno, sal ... group by deptno ...
```

ενώ επιτρέπεται

```
SELECT  deptno, avg (sal) ...
```

Προσοχή επίσης στις τιμές NULL! Οι τιμές αυτές δεν συμμετέχουν στον υπολογισμό των συναρτήσεων ομαδοποίησης. Επομένως, οι δηλώσεις SELECT AVG (sal) ... και SELECT SUM (sal) / COUNT (*) ... ενδεχομένως διαφέρουν!

Δείτε και το παρακάτω παράδειγμα δήλωσης στο προϊόν mySQL.

```
SELECT * FROM EMP;
SELECT AVG(COMM), SUM(COMM)/COUNT(COMM),
SUM(COMM)/COUNT(*) FROM EMP;
```

```
mysql>
mysql> SELECT * FROM EMP;
+-------+------------------+------------------+------+------------+---------+--------+--------+
| EMPNO | ENAME            | JOB              | MGR  | HIREDATE   | SAL     | COMM   | DEPTNO |
+-------+------------------+------------------+------+------------+---------+--------+--------+
|  6956 | ΣΚΟΥΡΑΣ          | ΠΩΛΗΤΗΣ          | 7890 | 2019-01-01 | 1000.00 | 400.00 |     30 |
|  7512 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ          | 7890 | 2019-01-01 | 1000.00 | 400.00 |     30 |
|  7522 | ΑΝΔΡΕΟΥ Ν.       | ΚΛΗΤΗΡΑΣ         | 7890 | 2017-01-01 | 1000.00 |   NULL |     30 |
|  7612 | ΑΝΔΡΕΟΥ          | ΠΩΛΗΤΗΣ          | 7890 | 2009-01-01 | 1000.00 | 400.00 |     30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       | MANAGER          | 7890 | 2007-01-01 | 3000.00 | 500.00 |     30 |
+-------+------------------+------------------+------+------------+---------+--------+--------+
5 rows in set (0.00 sec)

mysql> SELECT AVG(COMM), SUM(COMM)/COUNT(COMM),
    ->        SUM(COMM)/COUNT(*) FROM EMP;
+------------+-----------------------+--------------------+
| AVG(COMM)  | SUM(COMM)/COUNT(COMM) | SUM(COMM)/COUNT(*) |
+------------+-----------------------+--------------------+
| 425.000000 |            425.000000 |          340.000000 |
+------------+-----------------------+--------------------+
1 row in set (0.00 sec)
```

3.4.19 Συναρτήσεις στο προϊόν της Oracle

Στη συνέχεια παραθέτουμε τις σημαντικότερες συναρτήσεις στο προϊόν της Oracle, μια περιγραφή τους και παραδείγματα χρήσης και αποτέλεσμα.

ABS απόλυτη τιμή

```
SELECT ABS(-3) FROM DUAL;
3
```

CEIL ελάχιστος ακέραιος μεγαλύτερος ή ίσος με τον αριθμό

```
SELECT CEIL(-5.2) FROM DUAL;
-5
SELECT CEIL(5.2) FROM DUAL;
6
```

FLOOR μεγαλύτερος ακέραιος μικρότερος ή ίσος με τον αριθμό

```
SELECT FLOOR(5.2) FROM DUAL;
5
SELECT FLOOR(-5.2) FROM DUAL;
-6
```

MOD υπόλοιπο διαίρεσεως

```
SELECT MOD(12, 5) FROM DUAL;
2
SELECT MOD(12.3, 7.1) FROM DUAL;
5.2
```

POWER ύψωση σε δύναμη ακεραίου

```
SELECT POWER(2, 3) FROM DUAL;
8
```

ROUND στρογγύλευση αριθμού κατά συγκεκριμένο αριθμό δεκαδικών θέσεων

```
SELECT ROUND(25.229, 2) FROM DUAL;
25.23
```

SIGN πρόσημο αριθμού

```
SELECT SIGN(5.7) FROM DUAL;
1
SELECT SIGN(-3) FROM DUAL;
-1
SELECT SIGN(0) FROM DUAL;
0
```

SQRT τετραγωνική ρίζα αριθμού

```
SELECT SQRT(64) FROM DUAL;
8
SELECT SQRT(-64) FROM DUAL;
NULL
```

TRUNC αποκοπή κατά συγκεκριμένο αριθμό δεκαδικών θέσεων

```
SELECT TRUNC(32.4) FROM DUAL;
32
```

```
SELECT TRUNC(37.565, 2) FROM DUAL;
37.56
```

ASCII θέση στον πίνακα ASCII του πρώτου χαρακτήρα της συμβολοσειράς (string)

```
SELECT ASCII('JOHN') FROM DUAL;
74
```

CHR κάνει το αντίθετο από την ASCII

```
SELECT CHR(74) FROM DUAL;
J
```

INITCAP μεταγραφή σε κεφαλαία του πρώτου χαρακτήρα συμβολοσειράς

```
SELECT INITCAP('john') FROM DUAL;
John
```

LENGTH δείχνει το μήκος συμβολοσειράς

```
SELECT LENGTH('JOHN') FROM DUAL;
4
```

LOWER μεταγράφει συμβολοσειρά σε πεζά

```
SELECT LOWER('JOHN') FROM DUAL;
john
```

UPPER μεταγράφει συμβολοσειρά σε κεφαλαία

```
SELECT UPPER('john') FROM DUAL;
JOHN
```

LPAD προσθέτει κάποιο χαρακτήρα αριστερά της συμβολοσειράς

```
SELECT LPAD('JOHN', 10, 9) FROM DUAL;
999999JOHN
```

RPAD προσθέτει χαρακτήρα δεξιά της συμβολοσειράς

```
SELECT RPAD('JOHN', 10, 9) FROM DUAL;
JOHN999999
```

SUBSTR καθορισμός τμήματος συμβολοσειράς το οποίο αρχίζει σε συγκεκριμένη θέση και έχει δεδομένο μήκος

```
SELECT SUBSTR('JOHN', 2, 3) FROM DUAL;
OHN
```

ADD_MONTH πρόσθεση μηνών σε ημερομηνία

```
SELECT ADD_MONTH(birthdate, 804) FROM employee;
Ημερομηνία συνταξιοδότησης υπαλλήλων στα 67 χρόνια (804=67*12)
```

MONTHS_BETWEEN αριθμός μηνών μεταξύ ημερομηνιών

```
SELECT MONTHS_BETWEEN(sysdate, birthdate) FROM employee;
Ηλικία σε μήνες υπαλλήλων σήμερα
```

NEXT_DAY ημερομηνία επόμενης ημέρας

```
SELECT NEXT_DAY(sysdate, 'mon') FROM DUAL;
Ημερομηνία επόμενης Δευτέρας (monday)
```

DECODE αποκωδικοποιήση

```
SELECT empno, DECODE(job,
 'ANALYST', 'ΑΝΑΛΥΤΗΣ',
 'ΛΟΙΠΑ ΕΠΑΓΓΕΛΜΑΤΑ')
FROM employee;
```

Για όλους τους υπάλληλους δείχνει τον κωδικό τους (empno) και αν η θέση (job) τους είναι ANALYST δείχνει ΑΝΑΛΥΤΗΣ διαφορετικά δείχνει ΛΟΙΠΑ ΕΠΑΓΓΕΛΜΑΤΑ.

```
SELECT empno, DECODE(comm, NULL, 0, comm)
FROM employee;
Για όλους τους υπάλληλους δείχνει τον κωδικό τους (empno) και αν η προμήθειά
(comm) τους είναι NULL δείχνει 0 διαφορετικά δείχνει την τιμή της comm.
```

GREATEST μέγιστη τιμή

```
SELECT GREATEST(10, 20, 140, 60) FROM DUAL;
140
```

LEAST ελάχιστη τιμή

```
SELECT LEAST(10, 20, 140, 60) FROM DUAL;
10
```

NVL αντικατάσταση τιμής NULL με ...

```
SELECT empno, NVL(comm, 0)
FROM employee;
Για όλους τους υπάλληλους δείχνει τον κωδικό τους (empno) και αν η προμήθειά
(comm) τους είναι NULL δείχνει 0 διαφορετικά δείχνει την τιμή της προμήθειας
(comm).
```

TO_NUMBER εμφάνιση αντί αριθμητικής συμβολοσειράς (string) της αριθμητικής τιμής της

```
SELECT TO_NUMBER('22122')+5 FROM DUAL;
22127
```

TO_DATE εμφάνιση συμβολοσειράς ημερομηνίας σύμφωνα με format (το προκαθορισμένο –default-format είναι 'dd-mon-yy')

```
SELECT TO_DATE('280654', 'ddmmyy') FROM DUAL;
28-JUN-54
SELECT TO_DATE('280654') FROM DUAL;
28-Jun-54
```

TO_CHAR εμφάνιση συμβολοσειράς (αριθμητικής ή ημερομηνίας) σύμφωνα με format

```
SELECT TO_CHAR(TO_DATE('280654', 'ddmmyy') 'Month ddth Syear')
FROM DUAL;
JUNE 28th Nineteen Fifty Four
```

Άσκηση

Δείτε τις συναρτήσεις αυτές και στο στο προϊόν MySQL. Ποιες ισχύουν;

3.4.20 Παρατήρηση για τη χρήση των συναρτήσεων RTRIM, LTRIM

Σε κάποιες περιπτώσεις η χρήση συναρτήσεων όπως η decode φαίνεται να μας οδηγεί σε σωστά αποτελέσματα (π.χ. έχουμε σωστά αποτελέσματα αν η μεταβλητή στην οποία εφαρμόζεται είναι τύπου VARCHAR2) ενώ σε άλλες (π.χ. αν η μεταβλητή στην οποία εφαρμόζεται είναι τύπου CHAR) δεν μας οδηγεί σε σωστά αποτελέσματα. Αυτό συμβαίνει επειδή η τιμή της μεταβλητής είναι διαφορετικά αποθηκευμένη (ανάλογα με το αν χρησιμοποιούμε τύπο δεδομένων VARCHAR2 ή CHAR). Για να διορθώσουμε το πρόβλημα μπορούμε να χρησιμοποιούμε συναρτήσεις αποκοπής των κενών των αποθηκευμένων τιμών που συνδέονται με τον τύπο δεδομένων CHAR. Ακολουθεί απλό παράδειγμα.

```
SELECT DNAME ,
  DECODE(RTRIM(DNAME),
 'SALES', 'BARGAIN',
 'DEPARTMENTS')
FROM DEPT;
```

DNAME	DECODE(RTRIM(DNAME),'SALES','BARGAIN','DEPARTMENTS')
ACCOUNTING	DEPARTMENTS
RESEARCH	DEPARTMENTS
SALES	BARGAIN
OPERATIONS	DEPARTMENTS

Κρίνουμε επίσης σκόπιμο να γράψουμε αναλυτικά το παράδειγμα διαχείρισης τιμών NULL που φαίνεται στον πίνακα.

```
SELECT ENAME, DECODE (COMM,
 NULL,0,
 COMM)
  FROM EMP
  ORDER BY ENAME;
```

ENAME	DECODE(COMM,NULL,0,COMM)
CODD	0
DATE	200
ELMASRI	150
NAVATHE	0

3.4.21 Παραδείγματα χρήσης συναρτήσεων σε MySQL

Η συνάρτηση TRUNCATE έχει ως αποτέλεσμα την αποκοπή δεκαδικών θέσεων (ψηφίων) κατά συγκεκριμένο αριθμό.

```
SELECT TRUNCATE(37.565, 2) FROM DUAL;
```

Η συνάρτηση ASCII υπολογίζει τη θέση στον πίνακα ASCII του πρώτου χαρακτήρα της συμβολοσειράς (string).

```
SELECT ASCII('JOHN') FROM DUAL;
```

```
ysql>
ysql> # TRUNCATE αποκοπή κατά συγκεκριμένο αριθμό δεκαδικών θέσεων
ysql> SELECT TRUNCATE(37.565, 2) FROM DUAL;
+--------------------+
| TRUNCATE(37.565, 2) |
+--------------------+
|              37.56 |
+--------------------+
row in set (0.00 sec)

ysql> # ASCII θέση στον πίνακα ASCII του πρώτου χαρακτήρα της συμβολοσειράς (string)
ysql> SELECT ASCII('JOHN') FROM DUAL;
+---------------+
| ASCII('JOHN') |
+---------------+
|            74 |
+---------------+
row in set (0.00 sec)
```

Δημιουργία νέου πίνακα βασιζόμενου σε υπάρχοντα

```
SELECT * FROM DEPT;

DROP TABLE IF EXISTS NEW_DEPT;

CREATE TABLE NEW_DEPT(DEPTNO INT(2) NOT NULL, DNAME CHAR(14));
  INSERT INTO NEW_DEPT
  SELECT DEPTNO,
         CASE dname
          WHEN 'ΛΟΓΙΣΤΗΡΙΟ' THEN 'ACCOUNTING'
          WHEN 'ΠΩΛΗΣΕΙΣ' THEN 'SALES'
          ELSE 'Unknown level'
         end
  FROM dept;

SELECT * FROM NEW_DEPT;
```

```
mysql> SELECT * FROM DEPT;
+--------+----------------------+-------------+
| DEPTNO | DNAME                | LOC         |
+--------+----------------------+-------------+
|     10 | ΛΟΓΙΣΤΗΡΙΟ           | ΑΘΗΝΑ       |
|     30 | ΠΩΛΗΣΕΙΣ             | ΑΘΗΝΑ       |
+--------+----------------------+-------------+
2 rows in set (0.00 sec)

mysql> DROP TABLE IF EXISTS NEW_DEPT;
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE NEW_DEPT(DEPTNO INT(2) NOT NULL, DNAME CHAR(14));
Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> INSERT INTO NEW_DEPT
    -> SELECT DEPTNO, CASE dname
    -> WHEN 'ΛΟΓΙΣΤΗΡΙΟ' THEN 'ACCOUNTING'
    -> WHEN 'ΠΩΛΗΣΕΙΣ' THEN 'SALES'
    -> ELSE 'Unknown level'
    -> end
    -> FROM dept;
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM NEW_DEPT;
+--------+------------+
| DEPTNO | DNAME      |
+--------+------------+
|     10 | ACCOUNTING |
|     30 | SALES      |
+--------+------------+
2 rows in set (0.00 sec)
```

Παρατήρηση για τη χρήση των συναρτήσεων RTRIM, LTRIM όταν εφαρμόζονται σε στήλη που έχει οριστεί με τύπο δεδομένων CHAR.

```
SELECT * FROM EMP;
SELECT * FROM EMP WHERE ENAME="ΣΚΟΥΡΑΣ";
SELECT * FROM EMP WHERE ENAME=LTRIM("ΣΚΟΥΡΑΣ");
```

```
mysql>
mysql> SELECT * FROM EMP;
+-------+----------------+-----------------+------+------------+---------+---------+--------+
| EMPNO | ENAME          | JOB             | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+----------------+-----------------+------+------------+---------+---------+--------+
|  6956 | ΣΚΟΥΡΑΣ        | ΠΩΛΗΤΗΣ         | 7890 | 2019-01-01 | 1000.00 | 400.00  |     30 |
|  7512 | ΝΙΚΟΥ Ν.       | ΠΩΛΗΤΗΣ         | 7890 | 2019-01-01 | 1000.00 | 400.00  |     30 |
|  7522 | ΑΝΔΡΕΟΥ Ν.     | ΚΛΗΤΗΡΑΣ        | 7890 | 2017-01-01 | 1000.00 |    NULL |     30 |
|  7612 | ΑΝΔΡΕΟΥ        | ΠΩΛΗΤΗΣ         | 7890 | 2009-01-01 | 1000.00 | 400.00  |     30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.     | MANAGER         | 7890 | 2007-01-01 | 3000.00 | 500.00  |     30 |
+-------+----------------+-----------------+------+------------+---------+---------+--------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM EMP WHERE ENAME="    ΣΚΟΥΡΑΣ";
Empty set (0.00 sec)

mysql> SELECT * FROM EMP WHERE ENAME=LTRIM("    ΣΚΟΥΡΑΣ");
+-------+----------------+-----------------+------+------------+---------+---------+--------+
| EMPNO | ENAME          | JOB             | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+----------------+-----------------+------+------------+---------+---------+--------+
|  6956 | ΣΚΟΥΡΑΣ        | ΠΩΛΗΤΗΣ         | 7890 | 2019-01-01 | 1000.00 | 400.00  |     30 |
+-------+----------------+-----------------+------+------------+---------+---------+--------+
1 row in set (0.00 sec)
```

```
SELECT * FROM EMP;
SELECT * FROM EMP WHERE ENAME="ΣΚΟΥΡΑΣ ";
SELECT * FROM EMP WHERE ENAME=RTRIM("ΣΚΟΥΡΑΣ ");
```

```
mysql>
mysql> SELECT * FROM EMP;
+-------+------------------+------------------+------+------------+---------+---------+--------+
| EMPNO | ENAME            | JOB              | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+------------------+------------------+------+------------+---------+---------+--------+
|  6956 | ΣΚΟΥΡΑΣ          | ΠΩΛΗΤΗΣ          | 7890 | 2019-01-01 | 1000.00 | 400.00  |     30 |
|  7512 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ          | 7890 | 2019-01-01 | 1000.00 | 400.00  |     30 |
|  7522 | ΑΝΔΡΕΟΥ Ν.       | ΚΑΗΤΗΡΑΣ         | 7890 | 2017-01-01 | 1000.00 |    NULL |     30 |
|  7612 | ΑΝΔΡΕΟΥ          | ΠΩΛΗΤΗΣ          | 7890 | 2009-01-01 | 1000.00 | 400.00  |     30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       | MANAGER          | 7890 | 2007-01-01 | 3000.00 | 500.00  |     30 |
+-------+------------------+------------------+------+------------+---------+---------+--------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM EMP WHERE ENAME="ΣΚΟΥΡΑΣ     ";
Empty set (0.00 sec)

mysql> SELECT * FROM EMP WHERE ENAME=RTRIM("ΣΚΟΥΡΑΣ     ");
+-------+----------------+----------------+------+------------+---------+---------+--------+
| EMPNO | ENAME          | JOB            | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+----------------+----------------+------+------------+---------+---------+--------+
|  6956 | ΣΚΟΥΡΑΣ        | ΠΩΛΗΤΗΣ        | 7890 | 2019-01-01 | 1000.00 | 400.00  |     30 |
+-------+----------------+----------------+------+------------+---------+---------+--------+
1 row in set (0.00 sec)
```

Σύγκριση Oracle DECODE, CASE mySQL

Oracle	mySQL

`select dname,`
` DECODE(dname,`
`  'SALES', '1',`
`  'ACCOUNTING', '2',`
`  'Unknown level')`
`FROM dept;`
	`select dname, CASE dname`
	`WHEN 'SALES' THEN '1'`
	`WHEN 'ACCOUNTING' THEN '2'`
	`ELSE 'Unknown level'`
	`end`
	`FROM dept;`

3.4.22 Αναζήτηση με τους γνωστούς από τη θεωρία συνόλων τελεστές UNION, INTERSECT, MINUS

Οι τελεστές (πράξεις της σχεσιακής άλγεβρας) UNION (ένωση), INTERSECT (τομή), MINUS (διαφορά) είναι πολύ γνωστοί από τη θεωρία συνόλων.

Αρχικά ορίζουμε δύο πίνακες:

```
CREATE TABLE staff(ename CHAR(30), hospital CHAR(30))
CREATE TABLE doctor(ename CHAR(30), hospital CHAR(30))

INSERT INTO staff VALUES('CODD', 'METAXA')
INSERT INTO staff VALUES('DATE', 'CARE')
INSERT INTO staff VALUES('WIDOM', 'METAXA')

INSERT INTO doctor VALUES('ELMASRI', 'METAXA')
INSERT INTO doctor VALUES('NAVATHE', 'CARE')
INSERT INTO doctor VALUES('WIDOM', 'CARE')

SELECT * FROM staff
```

ENAME	HOSPITAL
CODD	METAXA
DATE	CARE
WIDOM	METAXA

```
SELECT * FROM doctor
```

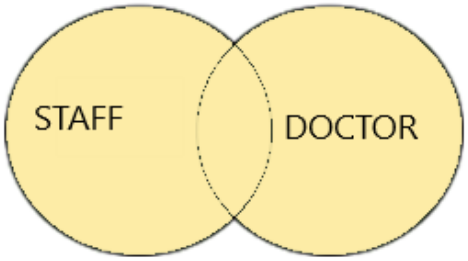
ENAME	HOSPITAL

ELMASRI	METAXA
NAVATHE	CARE
WIDOM	CARE

Η δήλωση

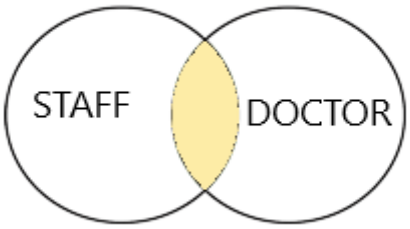
```
SELECT ename,hospital
FROM staff
WHERE hospital = 'METAXA'
UNION
SELECT ename, hospital
FROM doctor
WHERE hospital = 'METAXA';
```

βρίσκει την ένωση των συνόλων



ENAME	HOSPITAL
CODD	METAXA
ELMASRI	METAXA
WIDOM	METAXA

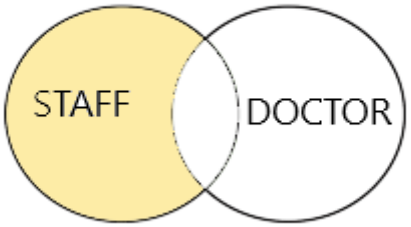
Αν αντί του UNION χρησιμοποιηθεί η τομή (intersect) υπολογίζεται η τομή των συνόλων:




```
SELECT ename
FROM staff
INTERSECT
SELECT ename
FROM doctor;
```

ENAME
WIDOM

Τέλος, αν χρησιμοποιηθεί η διαφορά (MINUS) υπολογίζεται η διαφορά των συνόλων:




```
SELECT ename,hospital
FROM staff
WHERE hospital = 'METAXA'
MINUS
SELECT ename, hospital
FROM doctor
WHERE hospital = 'METAXA';
```

ENAME	HOSPITAL
CODD	METAXA
WIDOM	METAXA

Προσοχή στη χρήση των τελεστών! Για παράδειγμα, η αναζήτηση

(select ... UNION select ...) INTERSECT select ... διαφέρει από

select ... UNION (select ... INTERSECT ... select).

Η MySQL δεν έχει τελεστές intersect, minus, προς το παρόν.

```
CREATE TABLE staff(ename CHAR(30), hospital CHAR(30));
CREATE TABLE doctor(ename CHAR(30), hospital CHAR(30));

INSERT INTO staff VALUES('CODD', 'METAXA');
INSERT INTO staff VALUES('DATE', 'CARE');
INSERT INTO staff VALUES('WIDOM', 'METAXA');
INSERT INTO doctor VALUES('ELMASRI', 'METAXA');
INSERT INTO doctor VALUES('NAVATHE', 'CARE');
INSERT INTO doctor VALUES('WIDOM', 'CARE');

SELECT * FROM staff;
SELECT * FROM doctor;
```

Η αναζήτηση

```
SELECT ename, hospital
FROM staff
WHERE hospital = 'METAXA'
UNION
SELECT ename, hospital
FROM doctor
WHERE hospital = 'METAXA'
```

βρίσκει την ένωση των συνόλων.

ORACLE	MySQL

`SELECT ENAME, HOSPITAL FROM staff`
`INTERSECT`
`SELECT ENAME, HOSPITAL FROM doctor;`
	`SELECT STAFF.ENAME, STAFF.HOSPITAL`
	`FROM STAFF INNER JOIN DOCTOR`
	`USING (ENAME, HOSPITAL);`

`SELECT ENAME FROM staff`
`INTERSECT`
`SELECT ENAME FROM doctor;`
	`SELECT STAFF.ENAME`
	`FROM STAFF INNER JOIN DOCTOR`
	`USING (ENAME);`

`SELECT ename FROM staff`
`MINUS`
`SELECT ename FROM doctor;`
	`SELECT DISTINCT ENAME, HOSPITAL`
	`FROM STAFF`
	`WHERE (ENAME, HOSPITAL) NOT IN`
	`(SELECT ENAME, HOSPITAL`
	`FROM DOCTOR);`
	`SELECT DISTINCT staff.ename,`
	`hospital`
	`FROM staff LEFT JOIN doctor`
	`USING(ename, hospital)`
	`WHERE doctor.ename IS NULL;`

3.4.23 Ιεραρχική αναζήτηση στην ORACLE

Η ιεραρχική αναζήτηση χρησιμοποιείται για την ανάκτηση γραμμών ταξινομημένων σε δομή δένδρου, αν υποτεθεί ότι τα στοιχεία είναι έτσι δομημένα. Η δήλωση αυτή υποστηρίζεται από το προϊόν της ORACLE και έχει τη γενική μορφή:

- SELECT εκφράσεις στηλών
- LEVEL αριθμοί επιπέδου των κόμβων (nodes) στο δένδρο
- FROM πίνακες ή όψεις
- WHERE συνθήκη
- CONNECT BY τα κριτήρια που ορίζουν τη δομή του δένδρου
- START WITH καθοριστικό της ρίζας του δένδρου
- ORDER BY κριτήρια ταξινόμησης ;

Παράδειγμα

Αλλάζουμε λίγο τα στοιχεία των υπαλλήλων.

```
UPDATE emp
SET mgr=20
WHERE ename='CODD';

UPDATE emp
SET mgr=20
WHERE ename='ELMASRI';

UPDATE emp
SET mgr=20
WHERE ename='CODD';
```

```
UPDATE emp
SET mgr=10
WHERE ename='DATE';

SELECT * FROM emp;
```

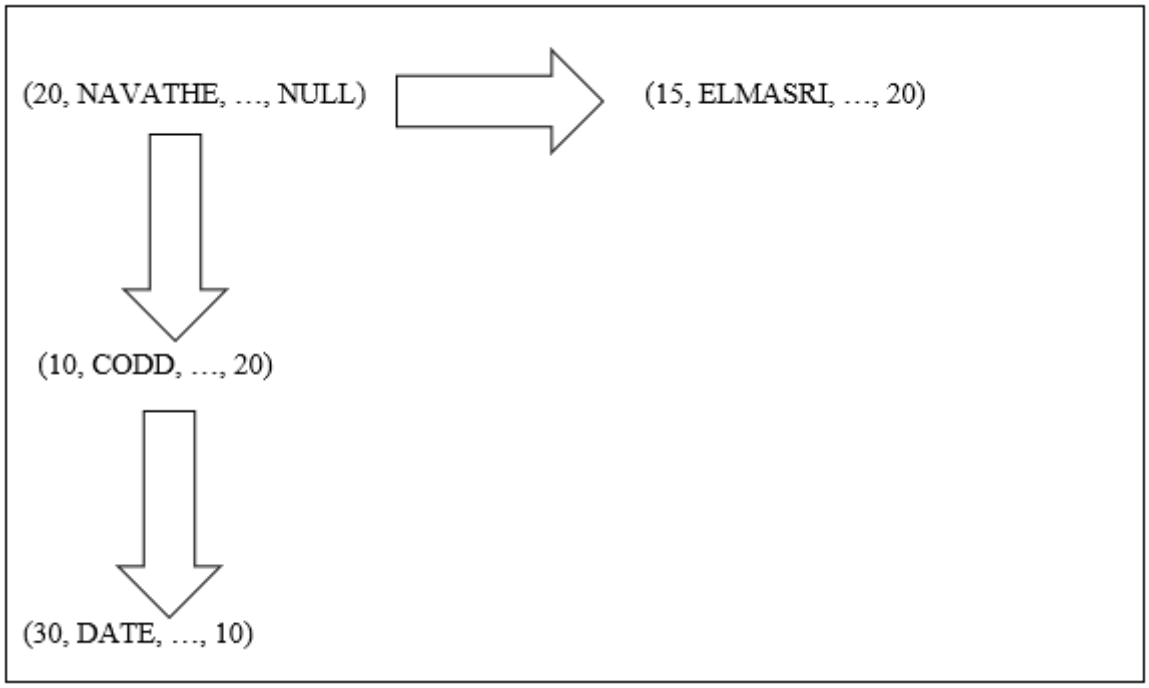
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	20	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	20	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	-	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	10	04/05/2004	1800	200	10

Η παρακάτω αναζήτηση

```
SELECT ename, empno, job, deptno, mgr
FROM emp
CONNECT BY PRIOR empno=mgr
START WITH ename = 'NAVATHE';
```

ENAME	EMPNO	JOB	DEPTNO	MGR
NAVATHE	20	SALESMAN	20	-
CODD	10	ANALYST	10	20
DATE	30	PROGRAMMER	10	10
ELMASRI	15	ANALYST	10	20

εμφανίζει σε δένδρο τους υπαλλήλους ανάλογα με τον επικεφαλής τους αρχίζοντας με τον NAVATHE ().



Εικόνα 3.8 Απεικόνιση υπαλλήλων με τον επικεφαλής τους με τη μορφή δέντρου

Να τι θα συμβεί αν ξεκινήσουμε από τον CODD.

```
SELECT ename, empno, job, deptno, mgr
FROM emp
CONNECT BY PRIOR empno=mgr
START WITH ename = 'CODD' ;
```

ENAME	EMPNO	JOB	DEPTNO	MGR
CODD	10	ANALYST	10	20
DATE	30	PROGRAMMER	10	10

3.4.24 Πώς διαχειριζόμαστε ημερομηνίες στο προϊόν της Oracle.

Παραθέτουμε παραδείγματα διαχείρισης ημερομηνιών.

Παράδειγμα 1

Δείξτε όνομα, θέση και ημερομηνία πρόσληψης για τους υπαλλήλους του τμήματος 10.

```
SELECT ENAME,JOB,HIREDATE
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	JOB	HIREDATE
CODD	ANALYST	01/01/1989
ELMASRI	ANALYST	02/05/1995
DATE	PROGRAMMER	04/05/2004

Παράδειγμα 2

Για τη στήλη HIREDATE χρησιμοποιήστε format MM/DD/YY.

```
SELECT ENAME,JOB,TO_CHAR(HIREDATE,'MM/DD/YY') HIREDATE
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	JOB	HIREDATE
CODD	ANALYST	01/01/89
ELMASRI	ANALYST	02/05/95
DATE	PROGRAMMER	04/05/04

Παράδειγμα 3

Εμφανίστε τις τιμές για τη στήλη HIREDATE ως εξής:

DAY MONTH DD , YYYY .

π.χ. SATURDAY 12 2 , 1989

```
SELECT ENAME,JOB,
  TO_CHAR(HIREDATE,'DAY MONTH DD, YYYY')  HIREDATE
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	JOB	HIREDATE
CODD	ANALYST	SUNDAY JANUARY 01, 1989
ELMASRI	ANALYST	SUNDAY FEBRUARY 05, 1995
DATE	PROGRAMMER	MONDAY APRIL 05, 2004

Παράδειγμα 4

Το ίδιο για DY DD MON YYYY.

π.χ. SY 2 DEC 1989

```
SELECT ENAME,JOB,
 TO_CHAR(HIREDATE,'DY DD MON YYYY')  HIREDATE
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	JOB	HIREDATE
CODD	ANALYST	SUN 01 JAN 1989
ELMASRI	ANALYST	SUN 05 FEB 1995
DATE	PROGRAMMER	MON 05 APR 2004

Παράδειγμα 5

Το ίδιο για Dy DD Mon YYYY.

π.χ. Sy 2 Dec 1989.

```
SELECT ENAME,JOB,
TO_CHAR(HIREDATE,'Dy DD Mon YYYY') HIREDATE
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	JOB	HIREDATE
CODD	ANALYST	Sun 01 Jan 1989
ELMASRI	ANALYST	Sun 05 Feb 1995
DATE	PROGRAMMER	Mon 05 Apr 2004

Παράδειγμα 6

Γράψτε πλήρη πρόταση για την ημερομηνία.

```
SELECT ENAME,JOB,
  TO_CHAR(HIREDATE,'Day "the" DDth "of" Month YYYY') HIREDATE
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	JOB	HIREDATE
CODD	ANALYST	Sunday the 01ST of January 1989
ELMASRI	ANALYST	Sunday the 05TH of February 1995
DATE	PROGRAMMER	Monday the 05TH of April 2004

Παράδειγμα 7

Δείξτε ημέρα και ώρα.

```
SELECT ENAME,
TO_CHAR(HIREDATE,'MONTH DD, YYYY HH:MIPM')  HIREDATE
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	HIREDATE
CODD	JANUARY 01, 1989 12:00AM
ELMASRI	FEBRUARY 05, 1995 12:00AM
DATE	APRIL 05, 2004 12:00AM

Παράδειγμα 8

Προσδιορίστε την ημερομηνία της πρώτης αξιολόγησης της απόδοσης των υπαλλήλων στο τμήμα 10. Η αξιολόγηση έγινε 3 μήνες μετά από την πρόσληψή τους.

```
SELECT ENAME, HIREDATE,HIREDATE + 90 REVIEWDATE
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	HIREDATE	REVIEWDATE
CODD	01/01/1989	04/01/1989
ELMASRI	02/05/1995	05/06/1995
DATE	04/05/2004	07/04/2004

3.4.25 Πώς χρησιμοποιούμε την ημερομηνία του συστήματος

Παραθέτουμε παραδείγματα.

Παράδειγμα 1

```
SELECT ENAME,HIREDATE,SYSDATE TODAY, DEPTNO
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	HIREDATE	TODAY	DEPTNO
CODD	01/01/1989	24/01/2021	10
ELMASRI	02/05/1995	24/01/2021	10
DATE	04/05/2004	24/01/2021	10

Παράδειγμα 2

Υπολόγισε πόσα έτη πέρασαν από την πρόσληψη των υπαλλήλων του τμήματος 10

```
SELECT ENAME,HIREDATE, (SYSDATE TODAY)/365,
   SYSDATE - HIREDATE, DEPTNO
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	HIREDATE	TODAY	SYSDATE-HIREDATE	DEPTNO
CODD	01/01/1989	11/04/2012	32.0849	10
ELMASRI	02/05/1995	11/04/2012	25.9863	10
DATE	04/05/2004	11/04/2012	16.7370	10

3.4.26 Πώς χρησιμοποιούμε τις συναρτήσεις NEXT_DAY, LAST_DAY

Παραθέτουμε παραδείγματα.

Παράδειγμα 1

Υπολογίστε την ημερομηνία της πρώτης Παρασκευής και την ημερομηνία τρεις μήνες μετά την πρόσληψη.

```
SELECT ENAME,
TO_CHAR(HIREDATE,'DAY MON-DD-YY') HIREDATE,
TO_CHAR(NEXT_DAY(HIREDATE,'FRIDAY'),
  'DAY MON-DD-YY') FIRSTFRIDAY,
TO_CHAR(LAST_DAY(HIREDATE+90),
  'DAY MON-DD-YY') ADD_3_MONTHS
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	HIREDATE	FIRSTFRIDAY	ADD_3_MONTHS
CODD	SUNDAY JAN-01-89	FRIDAY JAN-06-89	SUNDAY APR-30-89
ELMASRI	SUNDAY FEB-05-95	FRIDAY FEB-10-95	WEDNESDAY MAY-31-95
DATE	MONDAY APR-05-04	FRIDAY APR-09-04	SATURDAY JUL-31-04

Παράδειγμα 2

Υπολογίστε την ημερομηνία πρώτης αξιολόγησης κάθε υπαλλήλου προσθέτωντας (χρήση συνάρτησης ADD_MONTHS) 3 μήνες στη στήλη HIREDATE.

```
SELECT ENAME,HIREDATE,ADD_MONTHS(HIREDATE,3) REVIEWDATE
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	HIREDATE	REVIEWDATE
CODD	01/01/1989	04/01/1989
ELMASRI	02/05/1995	05/05/1995
DATE	04/05/2004	07/05/2004

Παράδειγμα 3

Μετρήστε πόσοι προσλήφθηκαν σε κάθε τετράμηνο μιας χρονιάς.

```
SELECT TO_CHAR(HIREDATE,'"Q-"Q YYYY') HIREDATE ,COUNT(*)
FROM EMP
GROUP BY TO_CHAR(HIREDATE,'"Q-"Q YYYY');
```

HIREDATE	COUNT(*)
Q-1 1995	1
Q-2 2004	1
Q-1 1989	1
Q-3 1977	1

3.5 Εισαγωγή στη δημιουργία, χρήση και ενημερωσιμότητα όψεων (view)

Στην ενότητα αυτή εξετάζουμε τις όψεις και πως ορίζουμε και διαχειριζόμαστε όψεις (views)

3.5.1 Τι είναι όψη (view)

Μια όψη (view) είναι ένας ιδεατός (virtual) πίνακας που περιλαμβάνει στοιχεία από ένα ή περισσότερους πραγματικούς πίνακες ή και άλλες όψεις της βάσης δεδομένων. Η όψη δεν έχει "φυσική" υπόσταση, δηλαδή δεν υπάρχει σαν πίνακας με αποθηκευμένα στοιχεία. Παρ' όλα αυτά τα στοιχεία της όψης αντανακλούν άμεσα τις αλλαγές που γίνονται στο περιεχόμενο των πινάκων στους οποίους βασίζεται.

Ο χρήστης διαχειρίζεται τις όψεις σαν πραγματικούς πίνακες με κάποιους περιορισμούς ανάλογα με τον ορισμό της. Η χρήση τους συνίσταται σε περιπτώσεις που είναι επιβεβλημένη:

- απλοποίηση της προσπέλασης στοιχείων
- ακεραιότητα στοιχείων
- ανεξαρτησία των στοιχείων
- ασφάλεια των στοιχείων
- προστασία του ιδιωτικού απόρρητου.

3.5.2 Πως ορίζουμε και πως καταργούμε όψη. Παραδείγματα με χρήση Oracle

Παραθέτουμε μία απλή μορφή της σύνταξη της δήλωσης CREATE VIEW στο προϊόν της ORACLE.

```
CREATE [OR REPLACE] VIEW view_name [(column_aliases)] AS
defining-query
[WITH READ ONLY]
[WITH CHECK OPTION]
```

Ακολουθούν παραδείγματα.

Παράδειγμα 1

Θέλουμε να ορίσουμε την όψη των υπαλλήλων που έχουν επικεφαλής (manager) τον υπάλληλο ΑΝΔΡΕΟΥ Ν. με κωδικό αριθμό 7890.

Στο προϊόν της Oracle δημιουργούμε τους πίνακες της βάσης δεδομένων.

```
CREATE TABLE DEPT (DEPTNO NUMBER(2),
  DNAME CHAR(14),
  LOC CHAR(13));
```

```
CREATE TABLE EMP (EMPNO NUMBER(4) NOT NULL,
 ENAME CHAR(10),
 JOB CHAR(9),
 MGR NUMBER(4),
 HIREDATE DATE,
 SAL NUMBER(7,2),
 COMM NUMBER(7,2),
 DEPTNO NUMBER(2));

CREATE TABLE PROJ (PROJNO NUMBER(3) NOT NULL,
  PNAME CHAR(5),
  BUDGET NUMBER (7,2));

ALTER TABLE EMP MODIFY (ENAME CHAR(20)); -- αυξάνουμε το επιτρεπτό μήκος των
ονομάτων

INSERT INTO EMP
VALUES(7512, 'NIKOY N.', 'ΠΩΛΗΤΗΣ',7890,
  TO_DATE('3/4/1987 9:30:15', 'DD-MM-YYYY HH:MI:SS'), 1000,NULL,30);
INSERT INTO EMP
VALUES(7612, 'ΑΝΔΡΕΟΥ', 'ΠΩΛΗΤΗΣ',7890,
   TO_DATE('10/11/1988 9:30', 'DD/MM/YY HH:MI'), 1000,NULL,30);
INSERT INTO EMP
VALUES(7522, 'ΑΝΔΡΕΟΥ Ν.', 'ΚΛΗΤΗΡΑΣ',7890,
   '10/11/1998', 1000,NULL,30);
INSERT INTO EMP (empno, ename, deptno)
  VALUES (6956, 'ΣΚΟΥΡΑΣ', 30 ) ;
INSERT INTO EMP
VALUES(7890, 'ΑΝΔΡΕΟΥ Ν.', 'MANAGER',7890,
   '12/12/1989', 3000, 500,30);
```

Παραθέτουμε τον ορισμό της όψης.

```
CREATE VIEW PROJMGR (EMPLOYEE,SALARY)
  AS SELECT  ENAME,SAL
 FROM   EMP
 WHERE  EMP.MGR = 7890;
```

Για να δούμε το περιεχόμενο της όψης:

```
SQL> SELECT  EMPLOYEE, SALARY FROM PROJMGR;
```

Για να καταργήσουμε την όψη δίνουμε:

```
SQL > DROP VIEW projmgr;
```

3.5.3 Πώς ορίζουμε όψη με χρήση join

Παραθέτουμε παραδείγματα.

Παράδειγμα 1

Να σχεδιάσετε αναφορά η οποία όταν εκτελείται (run) θα παρουσιάζει τους υπαλλήλους του έργου 101.

Πρώτα δημιουργείτε μια όψη βασιζόμενη στους πίνακες EMP και PROJ :

```
SQL> CREATE VIEW PROJSTAFF
 2  (EMPLOYEE,PROJECT,PROJECT_NUMBER)
 3  AS SELECT ENAME,PNAME,EMP.PROJNO
 4  FROM   EMP,PROJNO
 5  WHERE  EMP.PROJNO = PROJ.PROJNO ;
```

Έπειτα, στην περίπτωση του προϊόντος της Oracle, γράφετε τη δήλωση αναζήτησης στοιχείων προσθέτοντας τις απαραίτητες εντολές "φορμαρίσματος" των αποτελεσμάτων:

```
SQL> COLUMN EMPLOYEE  FORMAT A8;
SQL> COLUMN PROJECT   FORMAT A7;
SQL> SELECT EMPLOYEE,PROJECT
  2  FROM  PROJSTAFF
  3  WHERE PROJECT_NUMBER = 101;
```

Παράδειγμα 2

Υποθέτουμε ότι κάθε υπάλληλος εργάζεται σε ένα έργο (project). Προσθέτουμε (ALTER TABLE) στον πίνακα των υπαλλήλων τη στήλη κωδικός του έργου (projno) και τοποθετούμε (UPDATE) όλους τους υπάλληλους με επικεφαλής τον υπάλληλο 7980 στο έργο με κωδικό 101. Τέλος, δημιουργούμε την όψη PROJSTAFF, όπως βλέπετε στη συνέχεια. Ποια στοιχεία διαχειριζόμαστε με την όψη αυτή;

Μπορείτε να δοκιμάσετε τις επόμενο δηλώσεις.

```
ALTER TABLE EMP ADD (PROJNO NUMBER(3));
UPDATE emp
SET projno=101
WHERE mgr=7890;
INSERT INTO proj VALUES(101, 'P#1', 15000);
SELECT empno, ename, mgr, projno FROM emp;
CREATE VIEW PROJSTAFF
(CODE, EMPLOYEE, PROJECT, PROJECT_NUMBER)
  AS SELECT EMPNO, ENAME, PNAME, EMP.PROJNO
FROM   EMP, PROJ
WHERE  EMP.PROJNO = PROJ.PROJNO ;
COLUMN EMPLOYEE  FORMAT A8;
COLUMN PROJECT   FORMAT A7;
```

Ποια στοιχεία διαχειριζόμαστε με την όψη PROJSTAFF;

Υπόδειξη

Εκτελέστε τη δήλωση:

```
SELECT CODE, EMPLOYEE, PROJECT
  FROM  PROJSTAFF
  WHERE PROJECT_NUMBER = 101;
```

Παράδειγμα 3

Δείξτε στοιχεία υπαλλήλων και τα έργα στα οποία εργάζονται για τους υπάλληλους που ανήκουν σε τμήμα με έδρα CHICAGO.

Πρώτα δημιουργείτε μία όψη η οποία βασίζεται στους πίνακες PROJ, EMP, DEPT:

```
SQL> CREATE VIEW CHICAGO_PROJECTS
  2  (PROJECT, EMPLOYEE, EMP_NUMBER, LOCATION)
  3  AS SELECT  PNAME, ENAME, EMPNO, LOC
  4  FROM   PROJ, EMP, DEPT
  5  WHERE  EMP.DEPTNO = DEPT.DEPTNO
  6  ANDEMP.PROJNO = PROJ.PROJNO;
```

Έπειτα γράφετε την παρακάτω αναζήτηση.

```
SQL> SELECT PROJECT,EMPLOYEE,LOCATION
 2   FROM CHICAGO_PROJECTS
 3   WHERE LOCATION = 'CHICAGO';
```

3.5.4 Ορισμός όψης με υποπρόταση check option

Η χρήση της υποπρότασης WITH CHECK OPTION επιτρέπει τον έλεγχο των ενεργειών του χρήστη κατά την ενημέρωση στοιχείων των πινάκων της βάσης δεδομένων, δηλαδή εξασφαλίζει την ακεραιότητα των στοιχείων, κ.λπ.

Παράδειγμα 1

Έστω ότι δημιουργείτε την όψη των υπαλλήλων με μισθό πάνω από 1250.

```
SQL > CREATE VIEW CHECKSAL AS
2     SELECT  *  FROM emp
3     WHERE sal > 1800
4     WITH  check OPTION ;
```

Η δήλωση

```
 UPDATE  CHECKSAL
 SET  sal = 1500
 WHERE empno=10;
```

δεν επιτρέπεται επειδή προσπαθούμε να εκχωρήσουμε μισθό στον υπάλληλο 10 (που έχει μισθό 3000, άρα εμφανίζεται στην όψη) ίσο με 1500 (με την όψη διαχειριζόμαστε μισθούς μικρότερους των 1800).

Παράδειγμα 2

Η δήλωση

```
SQL >  CREATE  VIEW  checkemp
2   AS  SELECT  *  FROM emp
3   WHERE  deptno  IN (SELECT  deptno  FROM  dept)
4   WITH CHECK OPTION ;
```

επιτρέπει εισαγωγή και ενημέρωση στοιχείων μόνο αν ο κωδικός τμήματος (deptno) υπάρχει ήδη στον πίνακα DEPT.

Επομένως, η παρακάτω δήλωση δεν εκτελείται.

```
INSERT INTO EMP
VALUES(7777, 'ΑΝΔΡΕΟΥ Ν.', 'ΠΩΛΗΤΗΣ',7890,
   '12/12/1998', 1000,NULL,50, 101);
```

3.5.5 Σενάριο χρήσης. Δημιουργία όψεων για βάση δεδομένων παραγγελιών

Στην Εικόνα 3.9 βλέπουμε τους πίνακες μιας βάσης δεδομένων παραγγελιών. Τα ζητούμενα είναι:

- Δημιουργήστε όψη που να περιλαμβάνει τους πελάτες με έδρα την περιοχή του Αιγάλεω και εισάγετε νέους πελάτες αυτής της περιοχής
- Εξετάστε αν μπορείτε να εισάγετε στην όψη τα στοιχεία πελάτη του οποίου η έδρα είναι σε άλλη περιοχή, π.χ., στην Ηλιούπολη; Δηλαδή, εξετάστε αν μπορείτε να παραβιάσετε τον ορισμό της όψης.
- Πώς θα μπορούσατε να αποκλείσετε τη δυνατότητα παραβίασης της όψης με τον με κατάλληλο ορισμό της. Όπως θα δούμε η χρήση της υποπρότασης WITH CHECK OPTION στον ορισμό της όψης εξασφαλίζει το απαραβίαστο της συνθήκης που περιλαμβάνεται στον ορισμό.
- Να ορίσετε όψη η οποία θα βασίζεται σε συνδεδεμένους πίνακες, π.χ., στους πίνακες βασικών στοιχείων παραγγελίας και πελατών.
- Να εξετάσετε αν μπορείτε να εισάγετε στοιχεία σε όψεις βασιζόμενες σε συνδεδεμένους πίνακες. Δώστε παραδείγματα. Πότε θα μπορούσατε να εισάγετε στοιχεία;
- Δώστε εντολές αναζήτησης που περιλαμβάνουν υποπρόταση με δήλωση ομαδοποίησης. Μήπως δεν μπορείτε να εισάγετε στοιχεία σε τέτοιες όψεις;

```
mysql>
mysql> SELECT * FROM customers;
+--------+-------+----------+
| custno | cname | caddress |
+--------+-------+----------+
|      1 | SMITH | ATHENS   |
|      2 | JONES | VOLOS    |
|      3 | BATES | NEW YORK |
+--------+-------+----------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM stocks;
+---------+--------+-----------+
| stockno | descr  | listprice |
+---------+--------+-----------+
|       1 | APPLE  |      1.00 |
|       2 | ORANGE |      1.50 |
|       3 | LEMON  |      1.70 |
+---------+--------+-----------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM orders;
+---------+--------+------------+-------+
| orderno | custno | odate      | total |
+---------+--------+------------+-------+
|       1 |      1 | 2021-10-10 | 17.50 |
+---------+--------+------------+-------+
1 row in set (0.00 sec)

mysql> SELECT * FROM orderlines;
+---------+---------+------+--------+
| orderno | stockno | qty  | ptotal |
+---------+---------+------+--------+
|       1 |       1 |   10 |  10.00 |
|       1 |       2 |    5 |   7.50 |
+---------+---------+------+--------+
2 rows in set (0.00 sec)
```

Εικόνα 3.9 Βάση δεδομένων παραγγελιών

```
CREATE TABLE customers(custno NUMBER(3) NOT NULL,
cname VARCHAR2(10), caddress VARCHAR2(15),
PRIMARY KEY(custno));

CREATE TABLE stocks(stockno NUMBER(3) NOT NULL,
descr VARCHAR2(10), listprice NUMBER(7,2),
PRIMARY KEY (stockno));

CREATE TABLE orders(orderno NUMBER NOT NULL,
custno NUMBER(3), odate DATE, total NUMBER(9,2),
PRIMARY KEY (orderno),
FOREIGN KEY (custno) REFERENCES customers(custno));

CREATE TABLE orderlines(orderno NUMBER NOT NULL,
stockno NUMBER (3) NOT NULL, qty NUMBER(2),
ptotal NUMBER (8,2), PRIMARY KEY (orderno,stockno),
FOREIGN KEY(orderno) REFERENCES orders(orderno),
FOREIGN KEY (stockno) REFERENCES stocks (stockno));

INSERT INTO customers VALUES(1, 'SMITH', 'ATHENS');
INSERT INTO customers VALUES(2, 'JONES', 'VOLOS');
INSERT INTO customers VALUES(3, 'BATES', 'NEW YORK');

INSERT INTO stocks VALUES(1, 'APPLE', 1);
INSERT INTO stocks VALUES(2, 'ORANGE', 1.5);
INSERT INTO stocks VALUES(3, 'LEMON', 1.7);

INSERT INTO orders VALUES(1, 1, '10/10/2012', 17.5);
INSERT INTO orderlines VALUES(1, 1, 10, 10);
INSERT INTO orderlines VALUES(1, 2, 5, 7.5);

SELECT * FROM customers;
SELECT * FROM stocks;
SELECT * FROM orders;
SELECT * FROM orderlines;

SELECT  ORDERS.ORDERNO, ORDERS.CUSTNO, CNAME, CADDRESS,
STOCKS.STOCKNO, DESCR, QTY
FROMCUSTOMERS, ORDERS, ORDERLINES, STOCKS
WHERE CUSTOMERS.CUSTNO = ORDERS.CUSTNO
```

```
AND  ORDERS.ORDERNO = ORDERLINES.ORDERNO
AND ORDERLINES.QTY > 5;
```

Αν θέλουμε να εκτελέσουμε τα ίδια παραδείγματα σε MySQL μπορούμε να δημιουργήσουμε τη βάση ως εξής:

```
DROP TABLE ORDERLINES;
DROP TABLE ORDERS;
DROP TABLE STOCKS;
DROP TABLE CUSTOMERS;

CREATE TABLE customers(custno INT(3) NOT NULL,
cname CHAR(10), caddress CHAR(15),
PRIMARY KEY(custno));

CREATE TABLE stocks(stockno INT(3) NOT NULL,
descr CHAR(10), listprice FLOAT(7, 2),
PRIMARY KEY (stockno));

CREATE TABLE orders(orderno INT(5) NOT NULL,
custno INT(3), odate DATE, total FLOAT(9,2),
PRIMARY KEY (orderno),
  FOREIGN KEY (custno) REFERENCES customers(custno));

CREATE TABLE orderlines(orderno INT NOT NULL,
stockno INT (3) NOT NULL, qty INT(2),
ptotal FLOAT(8,2), PRIMARY KEY (orderno,stockno),
FOREIGN KEY(orderno) REFERENCES orders(orderno),
FOREIGN KEY (stockno) REFERENCES stocks (stockno));

INSERT INTO customers VALUES(1, 'SMITH', 'ATHENS');
INSERT INTO customers VALUES(2, 'JONES', 'VOLOS');
INSERT INTO customers VALUES(3, 'BATES', 'NEW YORK');

INSERT INTO stocks VALUES(1, 'APPLE', 1);
INSERT INTO stocks VALUES(2, 'ORANGE', 1.5);
INSERT INTO stocks VALUES(3, 'LEMON', 1.7);
INSERT INTO orders VALUES(1, 1, '2021-10-10', 17.5);
INSERT INTO orderlines VALUES(1, 1, 10, 10);
INSERT INTO orderlines VALUES(1, 2, 5, 7.5);

SELECT * FROM customers;
SELECT * FROM stocks;
SELECT * FROM orders;
SELECT * FROM orderlines;

SELECT  ORDERS.ORDERNO, ORDERS.CUSTNO, CNAME, CADDRESS,
STOCKS.STOCKNO, DESCR, QTY
FROMCUSTOMERS, ORDERS, ORDERLINES, STOCKS
WHERE CUSTOMERS.CUSTNO = ORDERS.CUSTNO
AND  ORDERS.ORDERNO = ORDERLINES.ORDERNO
AND ORDERLINES.QTY > 5;
```

```
mysql>
mysql> SELECT  ORDERS.ORDERNO, ORDERS.CUSTNO, CNAME, CADDRESS,
    ->                 STOCKS.STOCKNO, DESCR, QTY
    -> FROM    CUSTOMERS, ORDERS, ORDERLINES, STOCKS
    -> WHERE CUSTOMERS.CUSTNO = ORDERS.CUSTNO
    -> AND      ORDERS.ORDERNO = ORDERLINES.ORDERNO
    -> AND     ORDERLINES.QTY > 5;
+---------+--------+-------+----------+---------+--------+------+
| ORDERNO | CUSTNO | CNAME | CADDRESS | STOCKNO | DESCR  | QTY  |
+---------+--------+-------+----------+---------+--------+------+
|       1 |      1 | SMITH | ATHENS   |       1 | APPLE  |   10 |
|       1 |      1 | SMITH | ATHENS   |       2 | ORANGE |   10 |
|       1 |      1 | SMITH | ATHENS   |       3 | LEMON  |   10 |
+---------+--------+-------+----------+---------+--------+------+
3 rows in set (0.00 sec)
```

Έστω ότι θέλουμε να δημιουργήσουμε μια εξωτερική όψη στον πίνακα του πελάτη έτσι ώστε να διαχειριζόμαστε μόνο τους πελάτες με έδρα (διεύθυνση) ATHENS.

```
CREATE VIEW ATHENS_CUST(CUST,NAME,ADDRESS)
AS SELECT CUSTNO,CNAME,CADDRESS
FROM CUSTOMERS
WHERE CADDRESS='ATHENS';
```

Η όψη βασίζεται σε έναν μόνο πίνακα και η δήλωση δημιουργίας της μας επιτρέπει να εισάγουμε στοιχεία στον πίνακα Customers στον οποίο βασίζεται. Επομένως, η όψη είναι ενημερώσιμη, δηλαδή επιτρέπει δηλώσεις INSERT, UPDATE, DELETE στον πίνακα Customers. Όλες οι όψεις εξ' ορισμού επιτρέπουν δήλωση SELECT.

```
SELECT * FROM ATHENS_CUST;
```

CUST	NAME	ADDRESS
1	SMITH	ATHENS

Προσοχή! Θα εισάγουμε υπάλληλους οι οποίοι δεν έχουν διεύθυνση ATHENS. Δηλαδή,

```
INSERT INTO ATHENS_CUST VALUES(4,'SPANOS','AIGALEW');
INSERT INTO ATHENS_CUST VALUES(5,'STAMOS','HELIOUPOLIS');
```

Δείτε ότι έγινε εισαγωγή στοιχείων στον πίνακα CUSTOMERS.

```
SELECT * FROM CUSTOMERS;
```

CUSTNO	CNAME	CADDRESS
1	SMITH	ATHENS
2	JONES	VOLOS
3	BATES	NEW YORK
4	SPANOS	AIGALEW
5	STAMOS	HELIOUPOLIS

Διαγράψτε τις γραμμές αυτές.

```
DELETE FROM CUSTOMERS WHERE CUSTNO IN (4, 5);
SELECT * FROM CUSTOMERS;
```

CUSTNO	CNAME	CADDRESS
1	SMITH	ATHENS
2	JONES	VOLOS
3	BATES	NEW YORK

Συμπέρασμα

Η όψη την οποία δημιουργήσαμε είναι ενημερώσιμη και κατά συνέπεια είναι ένα εργαλείο στα χέρια μας το οποίο επιτρέπει να τη χρησιμοποιούν χειριστές χωρίς να γνωρίζουν τους πραγματικούς πίνακες της βάσης δεδομένων.

Αλλά ο τρόπος που τη δημιουργήσαμε οδηγεί σε προβλήματα παραβίασης του ορισμού της.

3.5.5.1 Δημιουργία ασφαλούς όψης με υποπρόταση WITH CHECK OPTION

Διαγράφουμε την όψη και την επαναδημιουργούμε.

```
DROP VIEW ATHENS_CUST;

CREATE VIEW ATHENS_CUST AS SELECT * FROM CUSTOMERS
  WHERE CADDRESS='ATHENS'
  WITH CHECK OPTION;
```

Τώρα μπορούμε να εισάγουμε με τη βοήθεια της όψης μόνο υπάλληλους που έχουν διεύθυνση ATHENS.

```
INSERT INTO ATHENS_CUST VALUES(4,'SPANOS','AIGALEW');
INSERT INTO ATHENS_CUST VALUES(5,'STAMOS','HELIOUPOLIS');

SELECT * FROM CUSTOMERS;
```

CUSTNO	CNAME	CADDRESS
1	SMITH	ATHENS
2	JONES	VOLOS
3	BATES	NEW YORK

```
SELECT * FROM athens_cust;
```

CUSTNO	CNAME	CADDRESS
1	SMITH	ATHENS

3.5.5.2 Δημιουργία view βασιζόμενης σε περισσότερους πίνακες

Η σύνταξη της δήλωσης ακολουθεί το σχήμα:

```
CREATE VIEW ….
AS  SELECT … JOIN

DROP VIEW ATH_CUST;


CREATE VIEW ORD_CUST(ORDERNO,CUSTNO,CNAME,ODATE,
 TOTAL)
```

```
 AS SELECT ORDERNO, ORDERS.ORDERNO,
 CNAME, ODATE, TOTAL
   FROM ORDERS,CUSTOMERS
   WHERE ORDERS.CUSTNO=CUSTOMERS.CUSTNO;
SELECT * FROM ORD_CUST;
```

ORDERNO	CUSTNO	CNAME	ODATE	TOTAL
1	1	SMITH	10/10/2021	17.5

```
INSERT INTO ORD_CUST VALUES (7,4,'TAKIDIS','15/10/2021',0);
ORA-01776: cannot modify more than one base table through a join view

INSERT INTO ORD_CUST(ORDERNO,CUSTNO) VALUES(7,3);
ORA-00001: unique constraint (CHRISTOS.SYS_C007242) violated

INSERT INTO ORD_CUST(CUSTNO,CNAME) VALUES(7,'AAAAA');
ORA-01776: cannot modify more than one base table through a join view
```

Σε αυτή την περίπτωση βλέπουμε ότι η όψη δεν είναι ενημερώσιμη, δηλαδή επιτρέπει μόνο δήλωση SELECT.

Κατάργηση των πινάκων.

```
DROP TABLE orderlines;
DROP TABLE orders;
DROP TABLE stocks;
DROP TABLE customers;
```

Περισσότερα θα δούμε στην επόμενη ενότητα.

3.6 Παράρτημα

Ακολουθεί script για τη δημιουργία των πινάκων της βάσης δεδομένων, την εισαγωγή στοιχείων, τη δημιουργία και τη διαχείριση όψεων.

```
DROP TABLE EMP;
DROP TABLE DEPT;
DROP TABLE PROJ;
DROP TABLE BONUS;

CREATE TABLE DEPT (DEPTNO NUMBER(2),
  DNAME CHAR(14),
  LOC CHAR(13));

CREATE TABLE EMP (EMPNO NUMBER(4) NOT NULL,
 ENAME CHAR(10),
 JOB CHAR(20),
 MGR NUMBER(4),
 HIREDATE DATE,
 SAL NUMBER(7,2),
 COMM NUMBER(7,2),
 DEPTNO NUMBER(2));

CREATE TABLE PROJ (PROJNO NUMBER(3) NOT NULL,
  PNAME CHAR(5),
  BUDGET NUMBER (7,2));
```

```
ALTER TABLE EMP MODIFY (ENAME CHAR(30));

INSERT INTO DEPT(DEPTNO, DNAME, LOC)
   VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
 VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
  VALUES (40, 'OPERATIONS', 'BOSTON');

SELECT * FROM DEPT;

INSERT INTO EMP
VALUES(10,'CODD','ΠΩΛΗΤΗΣ',15,'3/4/1987',1000, NULL,30);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES (15,
'ELMASRI', 'ΠΩΛΗΤΗΣ', 15, '20/02/1981', 1600, 300, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES (20,
'NAVATHE', 'DBA', 15, '22/02/1981', 1250, 500, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES (30,
'DATE', 'ΠΩΛΗΤΗΣ', 15, '02/04/1981', 2975, NULL, 20);
INSERT INTO EMP
VALUES(7512, 'ΝΙΚΟΥ Ν.', 'ΠΩΛΗΤΗΣ',7890,
  TO_DATE('3/4/1987 9:30:15', 'DD-MM-YYYY HH:MI:SS'),
  1000,NULL,30);
INSERT INTO EMP
VALUES(7612, 'ΑΝΔΡΕΟΥ', 'ΠΩΛΗΤΗΣ',7890,
   TO_DATE('10/11/1988 9:30', 'DD/MM/YY HH:MI'),
   1000,NULL,30);
INSERT INTO EMP
VALUES(7522, 'ΑΝΔΡΕΟΥ Ν.', 'ΚΛΗΤΗΡΑΣ',7890,
   '10/11/1998', 1000,NULL,30);
INSERT INTO EMP (empno, ename, deptno)
  VALUES (6956, 'ΣΚΟΥΡΑΣ', 30 ) ;
INSERT INTO EMP
VALUES(7890, 'ΑΝΔΡΕΟΥ Ν.', 'MANAGER',7890,
   '12/12/1989', 3000, 500,30);

SELECT * FROM EMP;
SELECT EMPNO, ENAME, MGR,SAL, COMM FROM EMP;

DROP VIEW PROJMGR;
CREATE VIEW PROJMGR (EMPLOYEE,SALARY)
  AS SELECT  ENAME,SAL
 FROM   EMP
 WHERE  EMP.MGR = 7890;
SELECT * FROM PROJMGR;

ALTER TABLE EMP ADD (PROJNO NUMBER(3));

UPDATE emp
SET projno=101
WHERE mgr=7890;

INSERT INTO proj VALUES(101, 'P#1', 15000);
```

```
SELECT empno, ename, mgr, projno FROM emp;
DROP VIEW PROJSTAFF;
CREATE VIEW PROJSTAFF
(CODE, EMPLOYEE, PROJECT, PROJECT_NUMBER)
  AS SELECT EMPNO, ENAME, PNAME, EMP.PROJNO
FROM   EMP, PROJ
WHERE  EMP.PROJNO = PROJ.PROJNO ;

COLUMN EMPLOYEE  FORMAT A8;
COLUMN PROJECT   FORMAT A7;
SELECT CODE, EMPLOYEE, PROJECT
  FROM  PROJSTAFF
  WHERE PROJECT_NUMBER = 101;
DROP VIEW CHICAGO_PROJECTS;
CREATE VIEW CHICAGO_PROJECTS
  (PROJECT, EMPLOYEE, EMP_NUMBER, LOCATION)
   AS  SELECT  PNAME, ENAME, EMPNO, LOC
  FROM   PROJ, EMP, DEPT
  WHERE  EMP.DEPTNO = DEPT.DEPTNO
  ANDEMP.PROJNO = PROJ.PROJNO;

SELECT * FROM chicago_projects;
SELECT PROJECT,EMPLOYEE,LOCATION
  FROM CHICAGO_PROJECTS WHERE LOCATION = 'CHICAGO';

DROP VIEW CHECKSAL;
CREATE VIEW CHECKSAL
AS SELECT  *  FROM emp
WHERE sal > 1800
WITH  check OPTION ;

UPDATE  CHECKSAL
SET  sal = 1500
WHERE empno=10;

SELECT * FROM CHECKSAL;

DROP VIEW CHECKEMP;
CREATE  VIEW  checkemp
 AS  SELECT  *  FROM emp
 WHERE  deptno  IN (SELECT  deptno  FROM  dept)
 WITH CHECK OPTION ;

INSERT INTO EMP
VALUES(7777, 'ΑΝΔΡΕΟΥ Ν.', 'ΠΩΛΗΤΗΣ',7890,
   '12/12/1998', 1000,NULL,50, 101);

SELECT * FROM CHECKEMP;

CREATE TABLE BONUS (EMPNO NUMBER(4) NOT NULL,
 ENAME CHAR(20),
 JOB CHAR(20),
 SAL NUMBER(7,2),
```

```
 COMM NUMBER(7,2),
 DEPTNO NUMBER(2));

INSERT INTO BONUS(EMPNO, JOB, SAL, COMM, DEPTNO)   SELECT EMPNO, JOB, SAL, COMM,
DEPTNO
FROM EMP
WHERE  JOB = 'MANAGER' OR  COMM > 0.25*SAL;

SELECT * FROM BONUS;

UPDATE EMP
SETJOB='ΠΩΛΗΤΗΣ', HIREDATE = SYSDATE, SAL=1.1*SAL
WHERE  ENAME = 'WIDOM';

UPDATE EMP
   SET  SAL = SAL*1.15
   WHERE  (JOB='ΠΩΛΗΤΗΣ' OR JOB='ΠΩΛΗΤΗΣ')
   ANDDEPTNO = 30;

UPDATE EMP
  SET SAL = SAL*1.05
WHERE  EMPNO IN
  (SELECT  EMPNO
FROM  BONUS);

UPDATE  EMP
  SET  SAL = (SELECT 1.1*AVG(SAL)
  FROM EMP WHERE JOB = 'ΠΩΛΗΤΗΣ')
  WHERE   JOB = 'ΠΩΛΗΤΗΣ';

UPDATE  emp
  SET (job, hiredate) =
   (SELECT job, hiredate
 FROM emp
 WHERE empno = 30)
  WHERE  empno = 7522;

DELETE FROM EMP
 WHERE ENAME =  'ΣΚΟΥΡΑΣ';

SELECT * FROM DEPT;
SELECT * FROM EMP;
SELECT * FROM PROJ;
SELECT * FROM BONUS;
```

Κεφάλαιο 4
Περιηγήσεις στην υλοποίηση σχεσιακών βάσεων δεδομένων με γλώσσα Structured Query Language (Tours on SQL)

Σύνοψη
Στο κεφάλαιο αυτό παρουσιάζουμε θέματα υλοποίησης σχεσιακών βάσεων δεδομένων με χρήση γλώσσας SQL (Structured Query Language). Υιοθετείται μία προσέγγιση «περιήγησης» με την έννοια του διαλόγου του προγραμματιστή με το ΣΔΒΔ. Ο προγραμματιστής θέτει ερωτήματα (queries) προς εκτέλεση και το ΣΔΒΔ εκτελεί και απαντά. Ο αναγνώστης-προγραμματιστής καλείται να δοκιμάσει στον υπολογιστή όλες τις δηλώσεις που παρατίθενται και να διεκπεραιώσει τα σχετικά παραδείγματα. Στο κεφάλαιο περιλαμβάνονται οι παρακάτω περιηγήσεις:

Περιήγηση πρώτη. *Διαχείριση βάσης δεδομένων με τη γλώσσα Oracle SQL. Χρήση Γλώσσας Ορισμού Δεδομένων (DDL) και Γλώσσας Χειρισμού Δεδομένων (DML).Η περιήγηση αποσκοπεί στην εξοικείωση του αναγνώστη με τη Γλώσσα Ορισμού Δεδομένων (DDL) και τη Γλώσσας Χειρισμού Δεδομένων (DML) της Oracle SQL. Με παραδείγματα επεξηγείται πως ορίζουμε τους πίνακες βάσης δεδομένων και διαχειριζόμαστε τους ορισμούς (CREATE TABLE, ALTER TABLE, DROP TABLE), πώς διαχειριζόμαστε ευρετήρια (CREATE INDEX, DROP INDEX). Πως ορίζουμε και διαχειριζόμαστε όψεις (views). Πώς ορίζουμε όψη με χρήση σύνδεσης (join). Παρουσιάζεται ο ορισμός όψης με υποπρόταση WITH CHECK OPTION και το ζήτημα της ενημερωσιμότητας όψεων (view). Παρουσιάζεται η εκχώρηση και η αφαίρεση δικαιωμάτων χρήσης δεδομένων (GRANT και REVOKE). Παρουσιάζεται το περιβάλλον SQL*PLUS της ORACLE και η αποθήκευση και η επανεκτέλεση ερωτημάτων. Η περιήγηση συμπληρώνεται με λυμένες ασκήσεις στις οποίες χρησιμοποιείται ORACLE SQL.*

Περιήγηση δεύτερη. *Αναλυτική παρουσίαση συνδέσεων (join) πινάκων, φωλιασμένων αναζητήσεων, χρήσης συναρτήσεων (functions) και ομαδοποίησης δεδομένων (GROUP BY) σε Oracle και MySQL. Στην περιήγηση γίνεται αναλυτική συζήτηση σύνθετων θεμάτων συνδέσεων (join) πινάκων, φωλιασμένων αναζητήσεων, χρήσης συναρτήσεων (functions) και ομαδοποίησης δεδομένων (GROUP BY) κ.λπ.*

Περιήγηση τρίτη. *Παρουσίαση και περαιτέρω συζήτηση σημαντικών για τις εφαρμογές δηλώσεων SELECT σε Oracle και MySQL. Στα παραδείγματα χρησιμοποιούνται διάφορες βάσεις δεδομένων, π.χ., προσωπικού - διαχείρισης έργων, και προμηθευτών - ανταλλακτικών. Η περιήγηση συμπληρώνεται με παραδείγματα και σε διάφορες βάσεις δεδομένων και παρατίθενται πολλές λυμένες ασκήσεις.*

Προαπαιτούμενη γνώση:
Προτείνεται η μελέτη του κεφαλαίου 3.

4.1 Περιήγηση πρώτη. Διαχείριση βάσης δεδομένων με τη γλώσσα Oracle SQL. Χρήση Γλώσσας Ορισμού Δεδομένων (DDL) και Γλώσσας Χειρισμού Δεδομένων (DML).

Ακολουθεί η πρώτη περιήγηση στην οποία περιγράφουμε και συζητάμε τον ορισμό βάσης δεδομένων και την εισαγωγή, ενημέρωση και αναζήτηση στοιχείων με χρήση του προϊόντος της ORACLE. Η βάση που χρησιμοποιείται είναι μια απλοποιημένη βάση προσωπικού που περιλαμβάνει πίνακες της εικόνας 4.1.

(table)Emp primary key=empno

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO

(table) Dept primary key=deptno

DEPTNO	DNAME	LOC

όπου,

EMPNO = κωδικός υπαλλήλου, π.χ. 7369

ENAME = όνομα, π.χ. ΝΙΚΟΥ

JOB = είδος εργασίας, π.χ. ΑΝΑΛΥΤΗΣ

MGR = επικεφαλής, π.χ. 7902

HIREDATE= ημερομηνία πρόσληψης, π.χ. 17-12-2010

SAL=μισθός, π.χ. 3000

COMM=προμήθεια, π.χ. 1400, NULL

DEPTNO=κωδικός Τμήματος (department), π.χ. 100

DNAME=όνομα, π.χ. ACCOUNTING

LOC=έδρα, π.χ. ΑΘΗΝΑ

Εικόνα 4.1 Πίνακες τμημάτων και υπαλλήλων

Ακολουθεί ο ορισμός και η διαχείριση δεδομένων.

4.1.1 Ορισμός πινάκων τμημάτων (DEPT), υπαλλήλων (EMP) και εισαγωγή στοιχείων.

Αν υπάρχουν οι πίνακες και θέλεις να τους καταργήσεις εκτελείς (run) τις δηλώσεις.

```
DROP TABLE emp;
DROP TABLE dept;
```

Ακολουθούν δηλώσεις ορισμού των πινάκων. Οι πίνακες ορίζονται με κύρια και ξένα κλειδιά,

```
CREATE TABLE DEPT(DEPTNO NUMBER(2) NOT NULL,
 DNAME VARCHAR2(14), LOC VARCHAR2(14),
 PRIMARY KEY(DEPTNO));
```

```
CREATE TABLE EMP(EMPNO NUMBER(4) NOT NULL,
  ENAME VARCHAR2(15), JOB VARCHAR2(25),
  MGR NUMBER(4),HIREDATE DATE, SAL NUMBER(7,2),
  COMM NUMBER(7,2),
  DEPTNO NUMBER(2),
  PRIMARY KEY(EMPNO),
  FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO));
```

Παράδειγμα εισαγωγής στοιχείων

Εισαγωγή στοιχείων στον πίνακα των τμημάτων.

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (40, 'OPERATIONS', 'BOSTON');
```

Εισαγωγή στοιχείων στον πίνακα των υπαλλήλων.

Υποδεικνύουμε το μορφότυπο των ημερομηνιών με την εντολή

alter session set nls_date_format = 'dd/mm/rrrr';

Oracle Help Center

SQL*Plus Quick Start (oracle.com)

```
INSERT INTO EMP
VALUES (10, 'CODD', 'ANALYST', 20,'01/01/1989', 3000, NULL, 10);
INSERT INTO EMP
VALUES (15, 'ELMASRI', 'ANALYST', 20,'02/05/1995',1200, 150, 10);
INSERT INTO EMP
VALUES (20, 'NAVATHE', 'SALESMAN', NULL,'07/07/1977', 2000, NULL, 20);
INSERT INTO EMP
VALUES (30, 'DATE', 'PROGRAMMER', 10,'04/05/2004', 1800, 200, 10);
```

Ακολουθούν δηλώσεις για να δούμε το περιεχόμενο των πινάκων.

Χρησιμοποιούνται απλές εντολές της συνιστώσας (component) SQL*PLUS για τη μορφοποίηση των αποτελεσμάτων.

```
COLUMN ENAME FORMAT A10;
COLUMN JOB FORMAT A10;
COLUMN EMPNO FORMAT 9999;
COLUMN SAL FORMAT 9999;
COLUMN DEPTNO FORMAT 99;
COLUMN MGR FORMAT 9999;
COLUMN DNAME FORMAT A10;
COLUMN LOC FORMAT A10;
```

```
SELECT * FROM EMP;

SELECT * FROM DEPT;
```

Βλέπουμε τα αποτελέσματα στην Εικόνα 4.2.

```
SQL>
SQL> COLUMN ENAME FORMAT A10;
SQL> COLUMN JOB FORMAT A10;
SQL> COLUMN EMPNO FORMAT 9999;
SQL> COLUMN SAL FORMAT 9999;
SQL> COLUMN DEPTNO FORMAT 99;
SQL> COLUMN MGR FORMAT 9999;
SQL> COLUMN DNAME FORMAT A10;
SQL> COLUMN LOC FORMAT A10;
SQL> SELECT * FROM EMP;

EMPNO ENAME      JOB          MGR HIREDATE     SAL       COMM DEPTNO
----- ---------- ---------- ----- ---------- ----- ---------- ------
   10 CODD       ANALYST       20 01/01/1989  3000                 10
   15 ELMASRI    ANALYST       20 02/05/1995  1200        150      10
   20 NAVATHE    SALESMAN         07/07/1977  2000                 20
   30 DATE       PROGRAMMER    10 04/05/2004  1800        200      10

SQL> SELECT * FROM DEPT;

DEPTNO DNAME      LOC
------ ---------- ----------
    10 ACCOUNTING NEW YORK
    20 RESEARCH   DALLAS
    30 SALES      CHICAGO
    40 OPERATIONS BOSTON
```

*Εικόνα 4.2 Δεδομένα πινάκων ως αποτέλεσμα δηλώσεων SELECT και εντολών μορφοποίησης αποτελεσμάτων. Χρησιμοποιείται η συνιστώσα ORACLE SQL*PLUS η οποία αποτελεί το περιβάλλον διαλόγου σε γλώσσα SQL του προγραμματιστή με τη βάση δεδομένων*

4.1.2 Προσθήκη στηλών στον ορισμό του πίνακα του υπαλλήλου

Η δήλωση ALTER TABLE παρέχει πολλές δυνατότητες στον σχεδιαστή της βάσης δεδομένων και είναι ένα εργαλείο για τη διαχείριση της δομής των πινάκων. Η παρακάτω δήλωση προσθέτει στον ορισμό του πίνακα του υπαλλήλου τις στήλες ADDRESS, ZIP .

```
ALTER TABLE EMP ADD (ADDRESS CHAR(15), ZIP NUMBER(7));
```

Θα πρέπει να ενημερώσεις τώρα με δηλώσεις UPDATE τα στοιχεία της διεύθυνσης και του ταχυδρομικού κώδικα για κάθε υπάλληλο.

Μπορείς να προσθέτεις στήλες, να διαγράφεις και να τροποποιείς στήλες να αλλάζεις ονόματα στηλών.

4.1.3 Ενημέρωση (update) του πίνακα για την αλλαγή στοιχείων συγκεκριμένου υπαλλήλου ή υπαλλήλων

Η δήλωση UPDATE επιτρέπει την ενημέρωση των στοιχείων ενός συγκεκριμένου υπαλλήλου. Ακολουθεί παράδειγμα ενημέρωσης των στοιχείων του υπαλλήλου 15, δηλαδή του ELMASRI.

```
UPDATE EMP
SET ADDRESS = 'THISSEOS 55', ZIP = 12345
WHERE EMPNO=15;
```

Η δήλωση UPDATE επιτρέπει, επίσης, την ενημέρωση των στοιχείων πολλών υπαλλήλων. Ακολουθεί παράδειγμα ενημέρωσης των στοιχείων όλων των υπαλλήλων οι οποίοι είναι αναλυτές.

```
UPDATE EMP
SET   ADDRESS = 'THISSEOS 55', ZIP = 12345
WHERE JOB= 'ANALYST';
```

Στη συνέχεια θα αναφέρουμε και δηλώσεις οριστικοποίησης (COMMIT), και αναίρεσης (ROLLBACK) των μεταβολών των στοιχείων.

4.1.4 Οριστικοποίηση της εισαγωγής/ενημέρωσης στοιχείων σε πίνακα. Δήλωση commit.

Εκτέλεσε δηλώσεις εισαγωγής στοιχείων νέων υπαλλήλων.

```
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
VALUES (1234, 'MONRO', 'SALESMAN', '22/02/1988', 1600, 200, 10);
INSERT INTO EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
VALUES (4321, 'GREEN', 'SALESMAN', '28/05/1989', 1750, 400, 30);
```

Προσοχή! Αν έχεις αλλάξει τη δομή του πίνακα των υπαλλήλων θα πρέπει να προσαρμόσεις ανάλογα τις δηλώσεις INSERT.

Επικύρωσε τις μεταβολές.

```
COMMIT;
```

Διαπίστωσε ότι οι μεταβολές οριστικοποιήθηκαν.

```
SELECT empno, ename, job, deptno FROM emp;
```

Προσοχή! Αν έχεις ορίσει ότι το ΣΔΒΔ είναι σε λειτουργία αυτόματης επικύρωσης μεταβολών (AUTO COMMIT MODE) δεν απαιτείται να εκτελέσεις δήλωση COMMIT.

4.1.5 Διαγραφή υπαλλήλου ή υπαλλήλων.

Η δήλωση DELETE επιτρέπει τη διαγραφή των στοιχείων ενός συγκεκριμένου υπαλλήλου.

```
DELETE FROM EMP
WHERE EMPNO = 1234;
```

Η δήλωση UPDATE επιτρέπει, επίσης, τη διαγραφή των στοιχείων πολλών υπαλλήλων, π.χ., των αναλυτών.

```
DELETE FROM EMP
WHERE JOB= 'ANALYST';
```

Μπορείς να δεις τις μεταβολές στα δεδομένα.

```
SELECT empno, ename, job, deptno FROM emp;
```

4.1.6 Ανάκληση διαγραφής. Δήλωση rollback.

Ακύρωσε τις μεταβολές.

```
ROLLBACK;
```

Διαπίστωσε ότι όσες μεταβολές στους πίνακες δεν είχαν οριστικοποιηθεί προηγουμένως με δήλωση commit ακυρώθηκαν.

```
SELECT empno, ename, job, deptno FROM emp;
```

Προσοχή! Αν έχεις ορίσει ότι το ΣΔΒΔ είναι σε λειτουργία αυτόματης επικύρωσης μεταβολών (AUTO COMMIT MODE) τότε η δήλωση ROLLBACK δεν έχει αποτέλεσμα.

4.1.7 Παράθεση της δομής των πινάκων. Εντολή describe

Στην Εικόνα 4.3 παραθέτουμε το σχήμα των πινάκων της βάσης δεδομένων. Αν έχουμε αλλάξει τη δομή των πινάκων η εικόνα θα είναι λίγο διαφορετική.

SQL> DESCRIBE EMP;

Name	Null	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME		CHAR(13)
JOB		CHAR(16)
MGR		NUMBER(40
HIREDATE		DATE
SAL		NUMBER(7, 2)
COMM		NUMBER(7, 2)
DEPTNO	NOT NULL	NUMBER(2)
ADDRESS		CHAR(15)
ZIP		NUMBER(7)

SQL>DESCRIBE DEPT;

Name	Null	Type
DEPTNO		NUMBER(2)
DNAME	NOT NULL	CHAR(14)
LOC		CHAR(13)

Εικόνα 4.3 Η δομή των πινάκων των τμημάτων και των υπαλλήλων

Άσκηση.

Σχολιάστε τα μήκη των στηλών των πινάκων. Είναι επαρκή για πραγματικές εφαρμογές;

Στη συνέχεια, στην Εικόνα 4.4 παραθέτουμε το περιεχόμενο των δύο πινάκων (EMP, DEPT). Με τους πίνακες αυτούς θα ελέγχονται τα αποτελέσματα των αναζητήσεων.

SQL> select *

2 from emp;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	20	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	20	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	-	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	10	04/05/2004	1800	200	10

SQL> select *

2 from dept;

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Εικόνα 4.4 Δεδομένα των πινάκων των τμημάτων και των υπαλλήλων

4.1.8 SQL*PLUS της ORACLE και αποθήκευση και επανεκτέλεση αναζητήσεων

Αν εργάζεσαι σε περιβάλλον SQL*PLUS της ORACLE μπορείς να αποθηκεύσεις αναζητήσεις και να τις ανακαλείς για επανεκτέλεση.

4.2 Συνθήκη WHERE σε δήλωση SELECT. Ταξινόμηση αποτελεσμάτων με χρήση order by.

Ακολουθούν παραδείγματα.

4.2.1 Ποιοί υπάλληλοι έχουν μισθό μικρότερο από 1200 και μεγαλύτερο από 1500.

Βλέπεις αρχικά τα στοιχεία των υπαλλήλων.

```
SELECT * FROM EMP
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	20	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	20	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	-	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	10	04/05/2004	1800	200	10

Η παρακάτω δήλωση χρησιμοποιεί τελεστές σύγκρισης.

```
SELECT EMPNO,ENAME,DEPTNO,SAL
FROM EMP
WHERE (SAL>1200) AND (SAL<2000);
```

EMPNO	ENAME	DEPTNO	SAL
30	DATE	10	1800

Προσοχή στην παρακάτω δήλωση!

```
SELECT EMPNO,ENAME,DEPTNO,SAL
FROM EMP
WHERE SAL BETWEEN 1200 AND 2000;
```

EMPNO	ENAME	DEPTNO	SAL
15	ELMASRI	10	1200
20	NAVATHE	20	2000
30	DATE	10	1800

Δηλαδή, δεν μπορείς να χρησιμοποιήσεις τον τελεστή BETWEEN … AND …. Η χρήση του θα ισοδυναμούσε με την εκτέλεση της δήλωσης

```
SELECT EMPNO,ENAME,DEPTNO,SAL
FROM EMP
WHERE (SAL>=1200) AND (SAL=<2000);
```

Βέβαια θα μπορούσες να δοκιμάσεις κάποια λύση κατά προσέγγιση, π.χ.,

```
SELECT EMPNO,ENAME,DEPTNO,SAL
FROM EMP
WHERE SAL BETWEEN 1200.01 AND 1999.99;
```

4.2.2 Ποιοί υπάλληλοι έχουν ετήσιες αποδοχές πάνω από 3000;

Βλέπεις αρχικά τα στοιχεία των υπαλλήλων.

```
SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	20	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	20	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	-	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	10	04/05/2004	1800	200	10

Για τον υπολογισμό των ετήσιων αποδοχών συνυπολογίζουμε μισθό και προμήθεια.

```
SELECT EMPNO,ENAME,DEPTNO,(SAL+NVL(COMM,0))*12 "YEARLY PAY"
FROM EMP
WHERE (SAL+NVL(COMM,0))*12 > 30000;
```

EMPNO	ENAME	DEPTNO	YEARLY PAY
10	CODD	10	36000

4.2.3 Δείξτε όλους τους υπάλληλους που είναι προγραμματιστές ή αναλυτές

Παραθέτουμε λύση με χρήση συνόλου και λύση με χρήση OR.

```
SELECT EMPNO,ENAME,DEPTNO,JOB
FROM EMP
WHERE JOB IN ('ANALYST','PROGRAMMER');

SELECT EMPNO,ENAME,DEPTNO,JOB
FROM EMP
WHERE JOB='ANALYST' OR JOB='PROGRAMMER';
```

EMPNO	ENAME	DEPTNO	JOB
10	CODD	10	ANALYST
15	ELMASRI	10	ANALYST
30	DATE	10	PROGRAMMER

4.2.4 Ποιοί υπάλληλοι δεν έχουν προμήθεια. Χρήση συνάρτησης NVL

Υποθέτουμε ότι ένας υπάλληλος δεν έχει προμήθεια αν στην αντίστοιχη στήλη έχει τιμή 0 ή NULL.

```
SELECT EMPNO,ENAME,COMM
FROM EMP
WHERE NVL(COMM,0)=0;
```

EMPNO	ENAME	COMM
10	CODD	-
20	NAVATHE	-

Προσοχή στην παρακάτω δήλωση!

```
SELECT EMPNO,ENAME,COMM
FROM EMP
WHERE COMM=NULL;
```

Το αποτέλεσμα ακολουθεί.

```
no rows selected
```

4.2.5 Ποιοί οι διαφορετικοί κωδικοί αριθμοί managers

Βλέπουμε τα στοιχεία των υπαλλήλων.

SELECT * FROM EMP;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	20	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	20	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	-	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	10	04/05/2004	1800	200	10

Βλέπουμε ότι κάποιοι υπάλληλοι είναι επικεφαλείς άλλων υπαλλήλων, π.χ., ο CODD εποπτεύει τον υπάλληλο DATE.

Αν υποθέσουμε ότι δεν μας ενδιαφέρει να έχει δηλωθεί και θέση (job) MANAGER τότε αρκεί να εκτελέσουμε την επόμενη δήλωση.

Στη δήλωση αυτή θα χρησιμοποιήσετε έναν από τους τελεστές που διαθέτει το ΣΔΒΔ της ORACLE για τον τελεστή διάφορο.

```
SELECT DISTINCT MGR
FROM EMP
WHERE NVL(MGR,0) <> 0;
```

MGR
20
10

Έτσι εμφανίζονται όσοι διευθύνουν υπαλλήλους έστω και αν δεν είναι δηλωμένοι ως MANAGER. Αυτό επειδή γίνεται αναζήτηση στον πίνακα βάσει της στήλης MGR.

Προσθέτουμε νέο υπάλληλο που έχει θέση MANAGER.

```
INSERT INTO emp(empno, ename, job, mgr, hiredate, sal, comm, deptno)
VALUES(40, 'WIDOM', 'MANAGER', 40, '01/01/1989', 4000, NULL, NULL);
```

Ο πίνακας διαμορφώθηκε ως εξής:

```
SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	20	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	20	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	-	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	10	04/05/2004	1800	200	10
40	WIDOM	MANAGER	40	01/01/1989	4000	-	-

Τότε με την παρακάτω δήλωση εμφανίζονται μόνον όσοι έχουν δηλωθεί ως MANAGER.

```
SELECT EMPNO
FROM EMP
WHERE JOB = 'MANAGER';
```

EMPNO
40

Πως θα εμφανίζατε όσους εποπτεύουν και όσους έχουν δηλωθεί ως MANAGER;

4.2.6 Διάταξη των υπαλλήλων κατά τρία επίπεδα ταξινόμησης, κατά τμήμα, φθίνουσα τάξη μισθού και όνομα

Θέλουμε να ταξινομήσουμε τους υπάλληλους ανά τμήμα. Οι υπάλληλοι που ανήκουν στο ίδιο τμήμα πρέπει ταξινομούνται κατά φθίνουσα τάξη μισθού. Οι υπάλληλοι οι οποίοι ανήκουν στο ίδιο τμήμα και έχουν τον ίδιο μισθό πρέπει να ταξινομούνται αλφαβητικά.

Η δήλωση των τριών επιπέδων ταξινόμησης γίνεται με χρήση της υποπρότασης

ORDER BY DEPTNO,SAL DESC,ENAME

alter session set nls_date_format = 'dd/mm/rrrr';

Υποθέτουμε ότι προσλαμβάνονται δύο νέοι υπάλληλοι:

Η υπάλληλος WIDOM,

```
INSERT INTO emp(empno, ename, job, mgr, hiredate, sal, comm, deptno)
VALUES(50, 'WIDOM', 'SALESMAN', 40, '01/01/1989', 2000, NULL, NULL);
```

και ο υπάλληλος RAMAKRISHNAN,

```
INSERT INTO emp(empno, ename, job, mgr, hiredate, sal, comm, deptno) VALUES (60,
'RAMAKRISHNAN', 'ANALYST', 40, '01/01/1989', 3000, NULL, 10);
```

Βλέπουμε τα στοιχεία των υπαλλήλων (πίνακας 4.1).

```
SELECT * FROM EMP;
```

Πίνακας 4.1 Με τα στοιχεία του πίνακα EMP ελέγχουμε το αποτέλεσμα των δηλώσεων SELECT

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	20	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	20	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	-	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	10	04/05/2004	1800	200	10
40	WIDOM	MANAGER	40	01/01/1989	4000	-	-
50	WIDOM	SALESMAN	40	01/01/1989	2000	-	-
60	RAMAKRISHNAN	ANALYST	40	01/01/1989	3000	-	10

Η παρακάτω δήλωση κάνει τη ζητούμενη ταξινόμηση.

```
SELECT DEPTNO,SAL,ENAME,EMPNO
FROM EMP
ORDER BY DEPTNO,SAL DESC,ENAME;
```

DEPTNO	SAL	ENAME	EMPNO
10	3000	CODD	10
10	3000	RAMAKRISHNAN	60
10	1800	DATE	30
10	1200	ELMASRI	15
20	2000	NAVATHE	20
-	4000	WIDOM	40
-	2000	WIDOM	50

4.2.7 Ταξινόμηση και τιμές NULL.

Η τιμή NULL, π.χ., στη στήλη COMM μπορεί να σημίνει διαφορετικά πράγματα.

Ο υπάλληλος δεν έχει προμήθεια, δε γνωρίζουμε αν ο υπάλληλος έχει προμήθεια, δεν έχει περαστεί η προμήθεια στα στοιχεία του.

Δύο εύλογα ερωτήματα είναι τα ακόλουθα:

- Τι γίνεται όταν εφαρμόζεις order by σε στήλη που περιέχει τιμές NULL;
- Πώς μπορείς να εμφανίζεις τιμές NULL πριν ή μετά από τιμές NOT NULL;

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.1.

```
SELECT * FROM EMP;
```

Στην παρακάτω δήλωση οι τιμές NULL τυπώνονται στο τέλος μετά από τις τιμές NOT NULL.

```
SELECT ENAME,COMM
FROM EMP
ORDER BY COMM;
```

ENAME	COMM
ELMASRI	150
DATE	200
NAVATHE	-
RAMAKRISHNAN	-
WIDOM	-
WIDOM	-
CODD	-

Το ίδιο ισχύει και για τη δήλωση

```
SELECT ENAME,COMM
FROM EMP
ORDER BY COMM ASC;
```

Ακολουθεί παράδειγμα χρήσης order by κατά φθίνουσα σειρά σε στήλη με NULL τιμές .

```
SELECT ENAME,COMM
FROM EMP
ORDER BY COMM DESC;
```

ENAME	COMM
CODD	-
WIDOM	-
NAVATHE	-
RAMAKRISHNAN	-
WIDOM	-
DATE	200
ELMASRI	150

Για να εμφανίσουμε τιμές NULL μετά από τιμές NOT NULL μπορούμε να χρησιμοποιήσουμε, ανάλογα με την περίπτωση, και κάποιο «τέχνασμα». Για παράδειγμα:

```
SELECT ENAME,COMM
FROM EMP
ORDER BY NVL(COMM,-1) DESC;
```

ENAME	COMM
DATE	200
ELMASRI	150
NAVATHE	-
RAMAKRISHNAN	-
WIDOM	-
WIDOM	-
CODD	-

4.2.8 Ονόματα υπαλλήλων και τριγράμματη συντομογραφία τους. Χρήση SUBSTR

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.1.

```
SELECT * FROM EMP;
```

Η παρακάτω δήλωση χρησιμοποιεί τη συνάρτηση SUBSTR, δείχνει το όνομα υπαλλήλου και επιπλέον πηγαίνει στην αρχή (πρώτη θέση) της συμβολοσειράς κάθε ονόματος και δείχμει τρεις χαρακτήρες.

```
SELECT EMPNO, ENAME,SUBSTR(ENAME,1,3) ABR
FROM EMP
ORDER BY ENAME;
```

EMPNO	ENAME	ABR
10	CODD	COD
30	DATE	DAT
15	ELMASRI	ELM
20	NAVATHE	NAV
60	RAMAKRISHNAN	RAM
40	WIDOM	WID
50	WIDOM	WID

4.2.9 Τίτλος λίστας με ονόματα και είδος εργασίας των υπαλλήλων. Χρήση συνάρτησης LPAD

Θέλουμε μια λίστα με τα ονόματα και το είδος εργασίας των υπαλλήλων. Η λίστα θα έχει τίτλο:

"Employee and Job".

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.1. με δήλωση

```
SELECT * FROM EMP;
```

Στη δήλωση θα χρησιμοποιήσουμε τη συνάρτηση LPAD (left pad). Η συνάρτηση διαβάζει τη συμβολοσειρά του ονόματος της θέσης (job) κάθε υπαλλήλου, π.χ., ANALYST, συμπληρώνει με κενά το όνομα μέχρι 15 χαρακτήρες και το δείχνει. Υποτίθεται ότι η στήλη JOB είναι τύπου CHAR(16). Μπορούμε αντί του κενού να κάνουμε τη συμπλήρωση (pad) με άλλο χαρακτήρα.

Παραθέτουμε τη ζητούμενη δήλωση.

```
SELECT ENAME "EMPLOYEE   ", LPAD(JOB,15,' ') " ANDJOB"
FROM EMP;
```

Με ανάλογο τρόπο χρησιμοποιείται η συνάρτηση RPAD (right pad).

Επίσης, οι συναρτήσεις TRIM, LTRIM, RTRIM μπορούν να χρησιμοποιηθούν για να αποκόψουν κενά ή χαρακτήρες από συμβολοσειρές.

Στο επόμενο παράδειγμα χρησιμοποιούμε αντί του κενού τον χαρακτήρα X.

```
SELECT ENAME "EMPLOYEE   ", LPAD(JOB,15,' X') " ANDJOB"
FROM EMP;
```

Employee	and Job
CODD	XXXXXXXXANALYST
ELMASRI	XXXXXXXXANALYST
NAVATHE	XXXXXXXSALESMAN
DATE	XXXXXPROGRAMMER
WIDOM	XXXXXXXXMANAGER
WIDOM	XXXXXXXSALESMAN
RAMAKRISHNAN	XXXXXXXXANALYST

4.2.10 Εκτυπωτικό με λίστα

Θέλουμε να χρησιμοποιούμε μια λίστα υπαλλήλων της μορφής της εικόνας 4.5.

Who, what and when
- - - - - - - - - - - - - - -
Ο κύριος CODD είναι ANALYST από 01-01-1989
.....
.....

Εικόνα 4.5 Λίστα υπαλλήλων

Ακολουθεί η δήλωση.

```
SELECT 'Ο ΚΥΡΙΟΣ '||ENAME||' ΕΙΝΑΙ '||JOB||' ΑΠΟ '|| TO_CHAR(HIREDATE,'DD-MM-
YYYY') "Who, what and when "
FROM EMP;
```

Who, what and when
Ο ΚΥΡΙΟΣ CODD ΕΙΝΑΙ ANALYST ΑΠΟ 01-01-1989

Ο ΚΥΡΙΟΣ ELMASRI EINAI ANALYST ΑΠΟ 05-02-1995
Ο ΚΥΡΙΟΣ NAVATHE EINAI SALESMAN ΑΠΟ 07-07-1977
Ο ΚΥΡΙΟΣ DATE EINAI PROGRAMMER ΑΠΟ 05-04-2004
Ο ΚΥΡΙΟΣ WIDOM EINAI MANAGER ΑΠΟ 01-01-1989
Ο ΚΥΡΙΟΣ WIDOM EINAI SALESMAN ΑΠΟ 01-01-1989
Ο ΚΥΡΙΟΣ RAMAKRISHNAN EINAI ANALYST ΑΠΟ 01-01-1989

4.2.11 Ποιοί άρχισαν να εργάζονται κάποιο συγκεκριμένο μήνα

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.1. με δήλωση

```
SELECT * FROM EMP;
```

Στη δήλωση θα χρησιμοποιήσουμε τη συνάρτηση TO_CHAR για να «απομονώσουμε» τον μήνα από την ημερομηνία πρόσληψης και να εξετάσουμε,π.χ., αν ο μήνας είναι ο Ιανουλριος.

Η συνάρτηση TO_CHAR μετατρέπει μία ημερομηνία ή ένα διάστημα (interval) σε συμβολοσειρά με το επιθυμητό μορφότυπο (format).

```
SELECT ENAME,EMPNO,HIREDATE
FROM EMP
WHERE TO_CHAR(HIREDATE,'MM')='01';
```

ENAME	EMPNO	HIREDATE
CODD	10	01/01/1989
WIDOM	40	01/01/1989
WIDOM	50	01/01/1989
RAMAKRISHNAN	60	01/01/1989

Η δήλωση

```
SELECT
  TO_CHAR( sysdate, 'YYYY-MM-DD' )
FROM   dual;
```

δείχνει τη σημερινή ημερομηνία με συγκεκριμένο format, π.χ., 2022-02-02

4.2.12 Λίστα υπαλλήλων εταιρείας με ταξινόμηση ανά μήνα πρόσληψης

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.1. με τη δήλωση

```
SELECT * FROM EMP;
```

Στην παρακάτω δήλωση θα χρησιμοποιήσουμε τη συνάρτηση TO_CHAR για να απομονώσουμε τον μήνα πρόσληψης και να ταξινομήσουμε τα αποτελέσματα.

```
SELECT ENAME,HIREDATE
FROM EMP
ORDER BY TO_CHAR(HIREDATE,'MM');
```

ENAME	HIREDATE
CODD	01/01/1989
WIDOM	01/01/1989
RAMAKRISHNAN	01/01/1989
WIDOM	01/01/1989
ELMASRI	02/05/1995
DATE	04/05/2004
NAVATHE	07/07/1977

4.3 Υποπρόταση GROUP BY και υποπρόταση GROUP BY και HAVING

Μία αρκετά γενική μορφή της σύνταξης της υποπρόταση GROUP BY (GROUP BY clause) παρατίθεται στη συνέχεια:

```
SELECT C1, C2,..., Cn, aggregate_function(Ci) [one or more]
FROM table
WHERE where_conditions (join is possible)
GROUP BY C1, C2,..., Cn;
```

Προσοχή! Συχνά τα ΣΔΒΔ αποκλίνουν από το πρότυπο της γλώσσας SQL. Για παράδειγμα το πρότυπο δεν επιτρέπει χρήση ψευδωνύμων (alias) στην υποπρόταση GROUP BY αλλά το προϊόν MySQL το επιτρέπει, π.χ.,

Οι επόμενες δηλώσεις είναι επιτρεπτές (βλέπε Εικόνα 4.6) στο προϊόν.

```
SELECT YEAR(Hiredate) AS year, COUNT(empno)
FROM emp
GROUP BY year;
```

```
mysql>
mysql> SELECT * FROM emp;
+-------+-------------------+-------------------+------+------------+---------+---------+---------+
| EMPNO | ENAME             | JOB               | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+-------------------+-------------------+------+------------+---------+---------+---------+
|  6956 | ΣΚΟΥΡΑΣ           | ΠΩΛΗΤΗΣ           | 7890 | 2019-01-01 | 1000.00 | 400.00 |      30 |
|  7512 | ΝΙΚΟΥ Ν.          | ΠΩΛΗΤΗΣ           | 7890 | 2019-01-01 | 1000.00 | 400.00 |      30 |
|  7522 | ΑΝΔΡΕΟΥ Ν.        | ΚΛΗΤΗΡΑΣ          | 7890 | 2017-01-01 | 1000.00 |   NULL |      30 |
|  7612 | ΑΝΔΡΕΟΥ           | ΠΩΛΗΤΗΣ           | 7890 | 2009-01-01 | 1000.00 | 400.00 |      30 |
|  7777 | ΝΙΚΟΥ Ν.          | ΠΩΛΗΤΗΣ           | 7890 | 2021-01-01 | 2000.00 |   NULL |      30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.        | MANAGER           | 7890 | 2007-01-01 | 3000.00 | 700.00 |      30 |
+-------+-------------------+-------------------+------+------------+---------+---------+---------+
6 rows in set (0.00 sec)

mysql> SELECT YEAR(Hiredate) AS year, COUNT(empno)
    -> FROM emp
    -> GROUP BY year
    -> ;
+------+--------------+
| year | COUNT(empno) |
+------+--------------+
| 2019 |            2 |
| 2017 |            1 |
| 2009 |            1 |
| 2021 |            1 |
| 2007 |            1 |
+------+--------------+
5 rows in set (0.00 sec)

mysql> SELECT YEAR(Hiredate) AS year, SUM(sal) AS total
    -> FROM emp
    -> INNER JOIN dept
    ->     USING (deptno)
    -> GROUP BY YEAR(Hiredate);
+------+---------+
| year | total   |
+------+---------+
| 2019 | 2000.00 |
| 2017 | 1000.00 |
| 2009 | 1000.00 |
| 2021 | 2000.00 |
| 2007 | 3000.00 |
+------+---------+
5 rows in set (0.00 sec)
```

Εικόνα 4.6 Το προϊόν MySQL επιτρέπει δήλωση SELECT με σύνδεση πινάκων και χρήση ψευδωνύμων (alias) στην υποπρόταση GROUP BY

```
SELECT YEAR(Hiredate) AS year, SUM(sal) AS total
FROM emp
INNER JOIN dept
USING (deptno)
GROUP BY YEAR(Hiredate);
```

```
SELECT YEAR(Hiredate) AS year, SUM(sal) AS total
FROM emp
INNER JOIN dept
USING (deptno)
GROUP BY YEAR;
```

Δείτε το αποτέλεσμα των δηλώσεων στην Εικόνα 4.6.

Μία αρκετά γενική σύνταξη της υποπρότασης GROUP BY και HAVING παρατίθεται στη συνέχεια.

```
SELECT C1, C2,..., Cn, aggregate_function(Ci) [one or more]
FROM table
WHERE where_conditions (join is possible)
GROUP BY C1, C2,..., Cn
HAVING group_condition;
```

Υπενθυμίζουμε ότι η υποπρόταση HAVING (HAVING clause) αποτελεί μία συνθήκη φίλτρου (filter condition) και εξετάζει κάθε γραμμή (row, group) αποτελεσμάτων που επιστρέφει η υποπρόταση GROUP

BY. Αν το αποτέλεσμα της συνθήκης φίλτρου είναι true τότε η γραμμή που υπολόγισε η group by περιλαμβάνεται στα αποτελέσματα της δήλωσης SELECT.

4.3.1 Μέσος όρος αμοιβής πωλητών

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.1. με δήλωση

```
SELECT * FROM EMP;
```

και επιβεβαιώνουμε το αποτέλεσμα της παρακάτω δήλωσης.Οι μηνιαίες αποδοχές υπολογίζονται ως άθροισμα μισθού και προμήθειας.

```
SELECT AVG(SAL+NVL(COMM,0)) "AVG PAY FOR SALESMEN"
FROM EMP
WHERE JOB='SALESMAN';
```

Avg pay for salesmen
2000

Υπολόγισε και το αποτέλεσμα της παρακάτω δήλωσης.

```
SELECT AVG(SAL+NVL(COMM,0)) "AVG PAY FOR SALESMEN"
FROM EMP
WHERE JOB='ANALYST';
```

((3000+1200+3000)/3)=(7200/3)=2400

4.3.2 Συνολικό έξοδο εταιρείας για αμοιβή πωλητών, αναλυτών:

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.1. με δήλωση

```
SELECT * FROM EMP;
```

και επιβεβαιώνουμε το αποτέλεσμα των παρακάτω δηλώσεων. Οι μηνιαίες αποδοχές υπολογίζονται ως άθροισμα μισθού και προμήθειας.

Εάν θέλουμε τα συνολικά έξοδα για τους πωλητές και αναλυτές ξεχωριστά, τότε δίνουμε :

```
SELECT JOB,SUM(SAL+NVL(COMM,0)) "sum pay"
FROM EMP
GROUP BY JOB
HAVING JOB IN('ANALYST','SALESMAN');
```

JOB	sum pay
SALESMAN	4000
ANALYST	7350

Εάν θέλουμε το συνολικό έξοδο για τους αναλυτές και τους πωλητές, δίνουμε :

```
SELECT SUM(SAL+NVL(COMM,0)) "sum pay"
FROM EMP
WHERE JOB IN ('ANALYST','SALESMAN');
```

sum pay
11350

Μέγιστη, ελάχιστη αμοιβή υπαλλήλων και διαφορά τους

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.1. με δήλωση

```
SELECT * FROM EMP;
```

και επιβεβαιώνουμε το αποτέλεσμα της παρακάτω δήλωσης. Οι μηνιαίες αποδοχές υπολογίζονται ως άθροισμα μισθού και προμήθειας.

```
SELECT MAX(SAL+NVL(COMM,0)) MAX,MIN(SAL+NVL(COMM,0)) MIN,
MAX(SAL+NVL(COMM,0))-MIN(SAL+NVL(COMM,0)) DIF
FROM EMP;
```

MAX	MIN	DIF
4000	1350	2650

4.3.3 Διακεκριμένες θέσεις συνολικά στα τμήματα 10, 20, 30

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.1. με δήλωση

```
SELECT * FROM EMP;
```

και επιβεβαιώνουμε το αποτέλεσμα της παρακάτω δήλωσης.

Η παρακάτω δήλωση υπολογίζει διακεκριμένες θέσεις ανά τμήμα.

```
SELECT COUNT(DISTINCT JOB) "Number of jobs", DEPTNO
FROM EMP
GROUP BY DEPTNO
HAVING DEPTNO IN (10,20,30);
```

Number of jobs	DEPTNO
1	20
2	10

Επίσης για να δούμε ποιές είναι αυτές οι θέσεις, δίνουμε :

```
SELECT DISTINCT JOB,DEPTNO
FROM EMP
GROUP BY DEPTNO,JOB
HAVING DEPTNO IN (10,20,30);
```

JOB	DEPTNO
ANALYST	10
PROGRAMMER	10
SALESMAN	20

```
SELECT DISTINCT JOB,DEPTNO, COUNT(JOB) "Number of jobs"
FROM EMP
```

```
GROUP BY DEPTNO,JOB
HAVING DEPTNO IN (10,20,30);
```

JOB	DEPTNO	Number of jobs
ANALYST	10	3
PROGRAMMER	10	1
SALESMAN	20	1

4.3.4 Μέση αμοιβή για κάθε θέση εργασία κάθε τμήματος

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.1. με δήλωση

```
 SELECT * FROM EMP;
```

και επιβεβαιώνουμε το αποτέλεσμα της παρακάτω δήλωσης.

```
SELECT DEPTNO,JOB,AVG(SAL+NVL(COMM,0)) "avg per job"
FROM EMP
GROUP BY DEPTNO,JOB;
```

DEPTNO	JOB	avg per job
10	ANALYST	2450
10	PROGRAMMER	2000
-	SALESMAN	2000
-	MANAGER	4000
20	SALESMAN	2000

4.3.5 Τμήματα στα οποια εργάζονται πάνω από δύο υπάλληλοι οι οποίοι έχουν την ίδια θέση.

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.1. με δήλωση

```
 SELECT * FROM EMP;
```

και επιβεβαιώνουμε το αποτέλεσμα της παρακάτω δήλωσης. Το κριτήριο της ομαδοποίησης είναι σύνθετο (deptno, job).

Η συνάρτηση COUNT(*) υπολογίζει τις γραμμές (τους υπάλληλους) κάθε ομάδας.

```
SELECT DEPTNO,COUNT(*) NUM, JOB
FROM EMP
GROUP BY DEPTNO,JOB
HAVING COUNT(*)>2;
```

DEPTNO	NUM	JOB
10	3	ANALYST

4.4 Δηλώσεις SELECT με συνδέσεις (Join) δύο πινάκων

Υπάρχουν διαφορετικοί τύποι JOIN δύο πινάκων στη γλώσσα SQL (Εικόνα 4.7):

(INNER) JOIN: Επιστρέφει τις γραμμές των οποίων οι τιμές «ταιριάζουν» (matching values) στους δύο πίνακες.

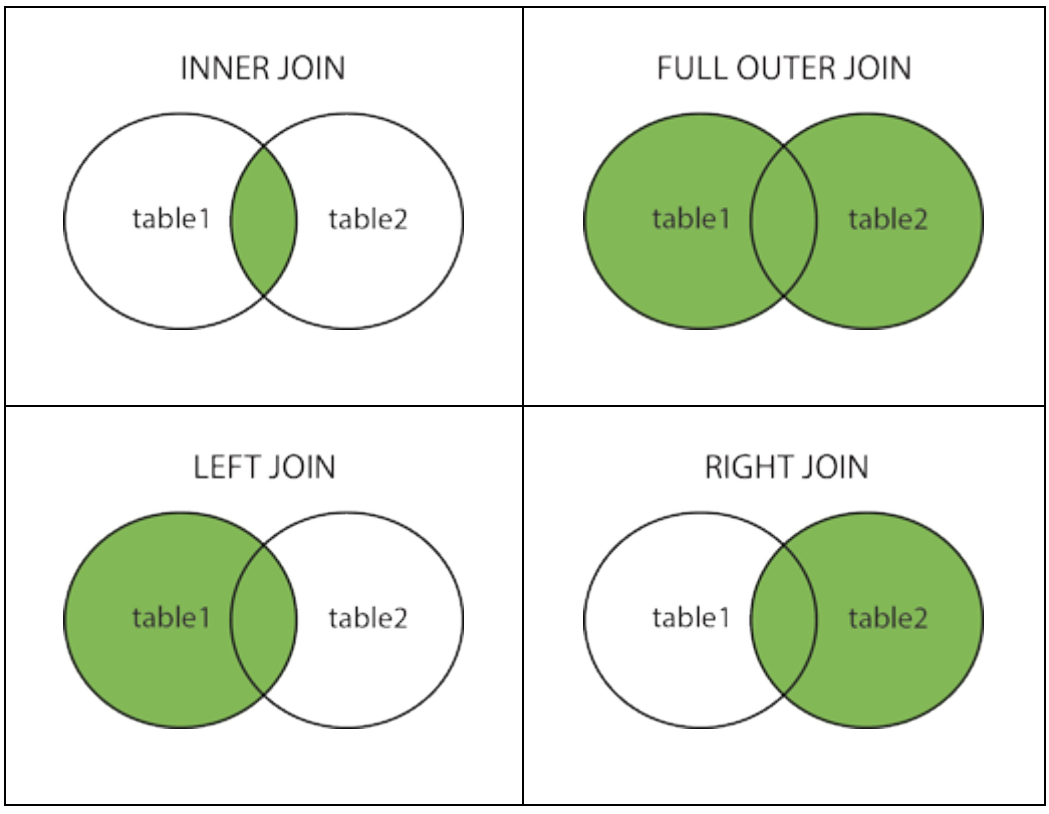
LEFT (OUTER) JOIN: Επιστρέφει όλες τις γραμμές του αριστερού πίνακα και τις γραμμές του δεξιού πίνακα των οποίων οι τιμές «ταιριάζουν».

RIGHT (OUTER) JOIN: Επιστρέφει όλες τις γραμμές του δεξιού πίνακα και τις γραμμές του αριστερού πίνακα των οποίων οι τιμές «ταιριάζουν».

FULL (OUTER) JOIN: Επιστρέφει όλες τις γραμμές για τις οποίες υπάρχει «ταίριασμα» είτε στον αριστερό πίνακα ή στον δεξιό πίνακα.

Βλέπουμε και σχετικά παραδείγματα.

```
SELECT empno, ename, emp.deptno, dname
FROM emp INNER JOIN dept ON emp.deptno=dept.deptno;

SELECT empno, ename, emp.deptno, dname
FROM emp LEFT JOIN dept ON emp.deptno=dept.deptno;

SELECT empno, ename, emp.deptno, dname
FROM emp RIGHT JOIN dept ON emp.deptno=dept.deptno;

SELECT empno, ename, emp.deptno, dname
FROM emp FULL JOIN dept ON emp.deptno=dept.deptno;
```



Εικόνα 4.7 *Τύποι JOIN δύο πινάκων στη γλώσσα SQL, Το προϊόν MySQL δεν υποστηρίζει FULL OUTER JOIN*

4.4.1 Σύνδεση πίνακα με τον εαυτό του. Υπάλληλοι που προσλήφθηκαν στην εταιρεία πριν από τον επικεφαλής τους:

Κάποιες αλλαγές στα δεδομένα.

```
UPDATE EMP
SET HIREDATE='01/01/1999'
WHERE EMPNO IN (40, 50, 60);
```

```
UPDATE EMP
SET MGR=40
WHERE ENAME='NAVATHE';
```

Στον πίνακα 4.4 βλέπουμε τα ενημερωμένα στοιχεία του πίνακα EMP με τα οποία ελέγχουμε το αποτέλεσμα των δηλώσεων SELECT.

Πίνακας 4.2 Ενημερωμένα στοιχεία του πίνακα EMP με τα οποία ελέγχουμε το αποτέλεσμα των δηλώσεων SELECT

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	20	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	20	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	40	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	10	04/05/2004	1800	200	10
40	WIDOM	MANAGER	40	01/01/1989	4000	-	-
50	WIDOM	SALESMAN	40	01/01/1999	2000	-	-
60	RAMAKRISHNAN	ANALYST	40	01/01/199	3000	-	10

Η παρακάτω δήλωση χρησιμοποιεί ψευδώνυμα (alias) για τον πίνακα του υπαλλήλου έτσι ώστε να επιτρέψει τη σύνδεσή του με τον εαυτό του, Με τον τρόπο αυτό μπορεί να συγκρίνει τα στοιχεία κάθε υπαλλήλου με τα στοιχεία του επικεφαλής του, π.χ., ο CODD δεν έχει προσληφθεί πριν από τον επικεφαλής του NAVATHE.

MNO	MNAME	MHDATE	MGR	ENAME	EHDATE
10	CODD	01/01/1989	20	NAVATHE	307/07/1977

```
SELECT M.EMPNO  MNO, M.ENAME MNAME,M.HIREDATE MHDATE,  M.MGR EMGR,
       E.ENAME NAME, E.HIREDATE EHDATE
FROM EMP M, EMP E
WHERE M.EMPNO = E.MGR
AND E.HIREDATE < M.HIREDATE;
```

MNAME	MHDATE	ENAME	HIREDATE
WIDOM	01/01/1999	NAVATHE	07/07/1977

Μπορούμε να γράψουμε τη δήλωση με πιο ευανάγνωστο τρόπο.

```
SELECT Manager.EMPNO  MNO, Manager.ENAME MNAME,Manager.HIREDATE MHDATE,
Manager.MGR EMGR, Employee.ENAME NAME, Employee.HIREDATE EHDATE
FROM EMP Manager, EMP Employee
WHERE Manager.EMPNO = Employee.MGR
AND Employee.HIREDATE < Manager.HIREDATE;
```

4.4.2 Λίστα υπαλλήλων με τον επικεφαλής τους. Ο πρόεδρος της εταιρείας συμπεριλαμβάνεται χωρίς κάποιον υπάλληλο ως επικεφαλή.

Εισάγουμε τα στοιχεία του προέδρου της εταιρείας. Βλέπουμε τα ενημερωμένα στοιχεία υπαλλήλων στον πίνακα 4.5,

```
INSERT INTO emp(empno, ename, job, mgr, hiredate, sal, comm, deptno) VALUES
(70, 'GEHRKE', 'PRESIDENT', NULL, '01/10/2012', 5000, NULL, NULL);
```

Πίνακας 4.3 Ενημερωμένα στοιχεία του πίνακα EMP με προσθήκη του προέδρου της εταιρίας

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	20	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	20	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	40	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	10	04/05/2004	1800	200	10
40	WIDOM	MANAGER	40	01/01/1989	4000	-	-
50	WIDOM	SALESMAN	40	01/01/1999	2000	-	-
60	RAMAKRISHNAN	ANALYST	40	01/01/199	3000	-	10
70	GEHRKE	PRESIDENT	-	01/10/2012	5000	-	-

Οι παρακάτω δηλώσεις χρησιμοποιούν outer join. Η σύνταξη της πρώτης δήλωσης δεν υποστηρίζεται από το προϊόν της MySQL.

```
SELECT MANAGER.EMPNO, MANAGER.ENAME MANAGER, WORKER.EMPNO,
WORKER.ENAME WORKER
FROM EMP MANAGER,EMP WORKER
WHERE WORKER.MGR = MANAGER.EMPNO(+)
ORDER BY MANAGER.ENAME;
```

```
SELECT MANAGER.EMPNO, MANAGER.ENAME MANAGER, WORKER.EMPNO,
WORKER.ENAME WORKER
FROM EMP MANAGER
RIGHT JOIN EMP WORKER  ON WORKER.MGR = MANAGER.EMPNO
ORDER BY MANAGER.ENAME;
```

EMPNO	MANAGER	EMPNO	WORKER
10	CODD	30	DATE
20	NAVATHE	15	ELMASRI
20	NAVATHE	10	CODD
40	WIDOM	60	RAMAKRISHNAN
40	WIDOM	50	WIDOM
40	WIDOM	40	WIDOM
40	WIDOM	20	NAVATHE
-	-	70	GEHRKE

4.5 Δηλώσεις SELECT οι οποίες περιλαμβάνουν εμφωλευμένες υποαναζητήσεις

Παραθέτουμε μία απλή κάπως γενική μορφή δήλωσης SELECT η οποία περιλαμβάνει εμφωλευμένη δήλωση (embedded select).

```
< - outer query ----------------- >
SELECT columns
FROM table1
WHERE column1 IN (SELECT column1
 FROM table2
WHERE condition);
   <---- subquery or inner query ---->
```

4.5.1 Ποιοί υπάλληλοι κερδίζουν περισσότερα από τους CODD και ELMASRI

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.3 με δήλωση

```
SELECT * FROM EMP;
```

και στη συνέχεια επιβεβαιώνουμε το αποτέλεσμα των παρακάτω δηλώσεων.

1ος τρόπος

```
SELECT EMPNO, ENAME,SAL+NVL(COMM,0) PAY
FROM EMP
WHERE SAL+NVL(COMM,0) > (SELECT MAX(SAL+NVL(COMM,0))
 FROM EMP
 WHERE ENAME IN ('CODD','ELMASRI'));
```

EMPNO	ENAME	PAY
40	WIDOM	4000
70	GEHRKE	5000

2ος τρόπος

```
SELECT EMPNO,ENAME,SAL+NVL(COMM,0) PAY
FROM EMP
WHERE SAL+NVL(COMM,0) >
   (SELECT SAL+NVL(COMM,0)
    FROM EMP
    WHERE ENAME = 'CODD')
 AND SAL+NVL(COMM,0) >
    (SELECT SAL+NVL(COMM,0)
FROM EMP
WHERE ENAME = 'ELMASRI');
```

Παραθέτουμε τις δηλώσεις και στο προϊόν MySQL.

```
SELECT EMPNO, ENAME,SAL+IFNULL(COMM,0)
FROM EMP
WHERE SAL+IFNULL(COMM,0) > (SELECT MAX(SAL+IFNULL(COMM,0))
```

```
 FROM EMP
 WHERE ENAME IN ('CODD','ELMASRI'));
```

```
SELECT EMPNO,ENAME,SAL+ IFNULL PAY
FROM EMP
WHERE SAL+ IFNULL (COMM,0) >
  (SELECT SAL+NVL(COMM,0)
   FROM EMP
   WHERE ENAME = 'CODD') AND SAL+ IFNULL (COMM,0) >
   (SELECT SAL+ IFNULL (COMM,0)
FROM EMP
WHERE ENAME = 'ELMASRI');
```

4.5.2 Ποιός εργάζεται για το τμήμα ACCOUNTING χωρίς να είναι αναλυτής.

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.3 με δήλωση

```
SELECT * FROM EMP;
```

και στη συνέχεια επιβεβαιώνουμε το αποτέλεσμα των παρακάτω δηλώσεων.

```
SELECT EMPNO, ENAME,JOB, DEPTNO
FROM EMP
WHERE JOB != 'ANALYST' AND DEPTNO =
   (SELECT DEPTNO
FROM   DEPT
WHERE DNAME = 'ACCOUNTING');
```

EMPNO	ENAME	JOB	DEPTNO
30	DATE	PROGRAMMER	10

Επίσης, το ίδιο επιτυγχάνεται και με χρήση σύνδεσης-join.

```
SELECT EMPNO, ENAME,JOB,EMP.DEPTNO, DNAME
FROM EMP,DEPT
WHERE EMP.DEPTNO=DEPT.DEPTNO
AND DNAME ='ACCOUNTING'
AND JOB != 'ANALYST';
```

EMPNO	ENAME	JOB	DEPTNO	DNAME
30	DATE	PROGRAMMER	10	ACCOUNTING

4.6 Οψεις

Παραθέτουμε μια κάπως γενική μορφή της σύνταξης της δήλωσης δημιουργίας όψης.

```
CREATE [OR REPLACE] VIEW [db_name.]view_name [(C1, C2, ,,,,, Ck)]
AS SELECT D1, D2, ,,,,,, Dk  -- select-statement
FROM ….;
```

4.6.1 Δημιουργία όψης η οποία περιλαμβάνει όνομα, μισθό, προμήθεια και ετήσια αμοιβή υπαλλήλου

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.3 με δήλωση

```
SELECT * FROM EMP;
```

και στη συνέχεια δημιουργούμε την όψη.

```
CREATE VIEW VIEW_SAL(EMPLOYEE,SALARY,COMMISSION,YEAR_PAY)
 AS SELECT ENAME,SAL,COMM,(SAL+NVL(COMM,0))*12
 FROM EMP;
View created. 1.01 seconds
```

Επιβεβαιώνουμε το αποτέλεσμα με χρήση της επόμενης δήλωσης.

```
SELECT *
FROM VIEW_SAL;
```

EMPLOYEE	SALARY	COMMISSION	YEAR_PAY
CODD	3000	-	36000
ELMASRI	1200	150	16200
NAVATHE	2000	-	24000
DATE	1800	200	24000
WIDOM	4000	-	48000
WIDOM	2000	-	24000
GEHRKE	5000	-	60000
RAMAKRISHNAN	3000	-	36000

Διαγράφουμε την όψη με την επόμενη δήλωση.

```
DROP VIEW view_sal;
View dropped.
```

4.6.2 Δημιουργία όψης η οποία περιλαμβάνει όνομα τμήματος και άθροισμα αμοιβών υπαλλήλων για τα τμήματα

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.3 με δήλωση

```
SELECT * FROM EMP;
```

και στη συνέχεια δημιουργούμε την όψη. Η αμοιβή υπολογίζεται ως άθροισμα μισθού και προμήθειας.

```
CREATE VIEW VIEW_DEPT(D_NO,D_NAME,PAY_SUM)
AS SELECT DEPT.DEPTNO, DNAME,SUM(SAL+NVL(COMM,0))
FROM DEPT,EMP
WHERE DEPT.DEPTNO = EMP.DEPTNO(+)
GROUP BY DNAME,DEPT.DEPTNO;
```

Επιβεβαιώνουμε το αποτέλεσμα με χρήση της επόμενης δήλωσης.

```
SELECT * FROM view_dept;
```

D_NO	D_NAME	PAY_SUM
20	RESEARCH	2000
10	ACCOUNTING	9350
30	SALES	-
40	OPERATIONS	-

Διαγράφουμε την όψη με την επόμενη δήλωση.

```
DROP VIEW view_dept;

View dropped.
```

4.6.3 Δημιουργία όψης η οποία δείχνει τον πίνακα των υπαλλήλων μόνο απόγευμα

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.3 με δήλωση

```
 SELECT * FROM EMP
```

Αρχικά βλέπουμε τη σημερινή ημερομηνία.

```
 SELECT TO_CHAR(SYSDATE,'DD.MM.YYYY:HH24:MI:SS') "SYSDATE"
 FROM dual;
```

SYSDATE
02.02.2021:18:05:27

και στη συνέχεια δημιουργούμε την όψη.

```
CREATE VIEW V_EMP(EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO)
AS SELECT EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO
FROM EMP
WHERE TO_CHAR(SYSDATE,'HH:MIPM') BETWEEN '04:00PM' AND '07:00PM'
WITH CHECK OPTION;
View created.
```

Δοκιμάζουμε τη δήλωση

```
SELECT * FROM v_emp;
no data found
```

επειδή η αναζήτηση δεν έγινε μέσα στα χρονικά πλαίσια που έχουν ορισθεί .

Διαγραφή της όψης.

```
DROP VIEW v_emp;
```

4.6.4 Δημιουργία όψης στον πίνακα των υπαλλήλων με χρήση υποπρότασης check option η οποία δεν θα επιτρέπει την εισαγωγή ενός ανύπαρκτου αριθμού (κωδικού) τμήματος και αμοιβή μεγαλύτερη της μέγιστης αμοιβής του πίνακα των υπαλλήλων.

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.3 με δήλωση

```
SELECT * FROM EMP
```

Δημιουργούμε την όψη.

```
CREATE VIEW V_EMPLOYEE(EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO)
  AS SELECT *
FROM EMP
WHERE DEPTNO IN (SELECT DEPTNO
  FROM DEPT)
AND SAL+NVL(COMM,0) <= (SELECT MAX(SAL+NVL(COMM,0))
  FROM EMP)
WITH CHECK OPTION;
View created.
```

Επιβεβαιώνουμε το αποτέλεσμα με χρήση της επόμενης δήλωσης.

```
SELECT * FROM v_employee;
```

4.7 Σύνθετες Αναζητήσεις

Ακολουθούν σύνθετες αναζητήσεις.

4.7.1 Υπάλληλοι που εργάζονται σε άλλο τμήμα από αυτό του επικεφαλής τους

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.3 με δήλωση

```
SELECT * FROM EMP;
```

Δημιουργούμε την όψη.

```
SELECT WORKER.EMPNO,WORKER.ENAME "NAME " ,
  WORKER.DEPTNO " WORKER DEPTNO",
   MANAGER.EMPNO, MANAGER.ENAME "NAME ",
  MANAGER.DEPTNO " MANAGER DEPTNO "
FROM EMP WORKER, EMP MANAGER
WHERE WORKER.MGR = MANAGER.EMPNO
AND WORKER.DEPTNO != MANAGER.DEPTNO;
```

Επιβεβαιώνουμε το αποτέλεσμα με χρήση της επόμενης δήλωσης

EMPNO	NAME	WORKER DEPTNO	EMPNO	NAME	MANAGER DEPTNO
15	ELMASRI	10	20	NAVATHE	20
10	CODD	10	20	NAVATHE	20

4.7.2 Πόσοι υπάλληλοι προσλήφθηκαν ανά έτος πρόσληψης

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.3 με δήλωση

```
 SELECT * FROM EMP
```

Η παρακάτω δήλωση χρησιμοποιεί τη συνάρτηση TO_CHAR για να απομονώσει το έτος πρόσληψης, και την υποπρόταση GROUP BY για να υπολογίσει των αριθμό προσλήψεων ανά έτος.

```
SELECT COUNT(*) "number of employees",TO_CHAR(HIREDATE,'YYYY') YEAR
FROM EMP
GROUP BY TO_CHAR(HIREDATE,'YYYY');
```

number of employees	YEAR
1	1977
1	1995
1	1989
3	1999
1	2004
1	2012

4.7.3 Ποιά χρονιά προσλήφθηκαν οι περισσότεροι υπάλληλοι.

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.3 με δήλωση

```
SELECT * FROM EMP
```

Η παρακάτω δήλωση χρησιμοποιεί τη συνάρτηση TO_CHAR για να απομονώσει το έτος πρόσληψης, και την υποπρόταση GROUP BY για να υπολογίσει των αριθμό προσλήψεων ανά έτος και την υποπρόταση GROUP BY …. HAVING για να φιλτράρει τα αποτελέσματα και να δείξει τελικά τη χρονιά προσλήφθηκαν οι περισσότεροι υπάλληλοι.

```
SELECT TO_CHAR(HIREDATE,'YYYY') YEAR,COUNT(*) NUM
FROM EMP
GROUP BY TO_CHAR(HIREDATE,'YYYY')
HAVING COUNT(*) =
   (SELECT MAX(COUNT(*))
 FROM EMP
 GROUP BY TO_CHAR(HIREDATE,'YYYY'));
```

YEAR	NUM
1999	3

4.7.4 Λίστα των managers και του αριθμού υπαλλήλων που έχουν κάτω από την επίβλεψή τους

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.3 με δήλωση

```
SELECT * FROM EMP
```

Η παρακάτω δήλωση εφαρμόζει την τεχνική με τα ψευδώνυμα στον πίνακα του υπαλλήλου για να παραθέσει με χρήση σύνδεσης δίπλα στα στοιχεία κάθε επικεφαλής τα στοιχεία των υπαλλήλων που επιβλέπει και χρησιμοποιεί την υποπρόταση GROUP BY για να υπολογίσει τον αριθμό των υπαλλήλων που επιβλέπει ο επικεφαλής.

```
SELECT MANAGER.EMPNO, MANAGER.ENAME, COUNT(*) NUM
FROM EMP MANAGER,EMP WORKER
WHERE MANAGER.EMPNO = WORKER.MGR
GROUP BY MANAGER.EMPNO, MANAGER.ENAME;
```

EMPNO	ENAME	NUM
20	NAVATHE	2
10	CODD	1
40	WIDOM	4

4.7.5 Πιο καλοπληρωμένοι υπάλληλοι ανά εργασία :

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.3 με δήλωση

```
SELECT * FROM EMP
```

Η παρακάτω δήλωση εφαρμόζει την τεχνική με τα ψευδώνυμα στον πίνακα του υπαλλήλου.

Επιπλέον χρησιμοποιεί τη συνάρτηση MAX.

Στην εμφωλευμένη δήλωση συνδέουμε τον πίνακα υπαλλήλου με τον εαυτό του για να παραθέσουμε μαζί τα στοιχεία των υπαλλήλων οι οποίοι έχουν την ίδια θέση (job).

Υπολογίζουμε, επίσης, τη μέγιστη αμοιβή ανά θέση.

Συγκρίνουμε την αμοιβή κάθε υπαλλήλου με τη μέγιστη αμοιβή που αντιστοιχεί στη θέση του.

```
SELECT  JOB,EMPNO, ENAME,SAL+NVL(COMM,0)
FROM EMP X
WHERE X.SAL+NVL(X.COMM,0) =
   (SELECT MAX(SAL+NVL(COMM,0))
  FROM EMP Y
  WHERE Y.JOB = X.JOB)
ORDER BY X.JOB;
```

JOB	EMPNO	ENAME	SAL+NVL(COMM,0)
ANALYST	10	CODD	3000
ANALYST	60	RAMAKRISHNAN	3000
MANAGER	40	WIDOM	4000
PRESIDENT	70	GEHRKE	5000
PROGRAMMER	30	DATE	2000
SALESMAN	20	NAVATHE	2000
SALESMAN	50	WIDOM	2000

4.7.6 Ο πιο καλοπληρωμένος υπάλληλος του τμήματος με την μικρότερη μέση αμοιβή

Βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.3 με δήλωση

```
SELECT * FROM EMP
```

Στην παρακάτω δήλωση χρησιμοποιούνται οι συναρτήσεις MIN, MAX., AVG.

Η εμφωλευμένη δήλωση

```
   SELECT MIN(AVG(SAL+NVL(COMM,0)))
   FROM EMP
   GROUP BY DEPTNO
```

υπολογίζει την ελάχιστη μέση αμοιβή ανά τμήμα.

Η εμφωλευμένη δήλωση

```
SELECT DEPTNO
FROM EMP
GROUP BY DEPTNO
HAVING …(SELECT  ….)
```

υπολογίζει το τμήμα με την ελάχιστη μέση αμοιβή,

```
SELECT EMPNO, ENAME,SAL+NVL(COMM,0) PAYMENT,DEPTNO
FROM EMP
WHERE SAL+NVL(COMM,0) =
   (SELECT MAX(SAL+NVL(COMM,0))
 FROM EMP
 WHERE DEPTNO =
  (SELECT DEPTNO
FROM EMP
GROUP BY DEPTNO
HAVING AVG(SAL+NVL(COMM,0)) =
 (SELECT MIN(AVG(SAL+NVL(COMM,0)))
   FROM EMP
   GROUP BY DEPTNO));
```

EMPNO	ENAME	PAYMENT	DEPTNO
20	NAVATHE	2000	20
30	DATE	2000	10
50	WIDOM	2000	-

4.8 Λυμένες ασκήσεις με χρήση του προϊόντος της ORACLE.

Ακολουθούν ασκήσεις με απαντήσεις. Για τον έλεγχο των απαντήσεων πρέπει να βλέπουμε τα στοιχεία των υπαλλήλων στον πίνακα 4.3 με τη δήλωση

```
 SELECT * FROM EMP;
```

Άσκηση 1. Δείξε τους ANALYST κατα φθίνουσα τάξη του λόγου προμήθεια/μισθός.

```
SELECT ENAME,COMM/SAL,COMM,SAL
FROM EMP
WHERE JOB = 'ANALYST'
ORDER BY COMM/SAL DESC;
```

ENAME	COMM/SAL	COMM	SAL
CODD	-	-	3000
RAMAKRISHNAN	-	-	3000
ELMASRI	.125	150	1200

Άσκηση 2. Υπολόγισε την ετήσια αμοιβή των υπαλλήλων με θέση ANALYST.

```
SELECT ENAME,SAL,COMM,(SAL + COMM) * 12
FROM EMP
WHERE JOB = 'ANALYST';
```

ENAME	SAL	COMM	(SAL+COMM)*12
CODD	3000	-	-
ELMASRI	1200	150	16200
RAMAKRISHNAN	3000	-	-

Άσκηση 3. Υπολόγισε το άθροισμα μισθού προμήθειας για όλους τους υπαλλήλους του τμήματος 10.

Προσοχή! Υπάρχουν υπάλληλοι με τιμή προμήθειας NULL. 'Αρα το άθροισμα SAL+COM, στην περίπτωσή τους, έχει τιμή NULL.

```
SELECT ENAME,JOB,SAL,COMM,SAL + NVL(COMM,0)
FROM EMP
WHERE DEPTNO = 10;
```

Θα μπορούσαμε να χρησιμοποιήσουμε εμφωλευμένη δήλωση SELECT για να απαντήσουμε στο ερώτημα «Υπολόγισε το άθροισμα μισθού προμήθειας για όλους τους υπαλλήλους του τμήματος ACCOUNTING».

ENAME	JOB	SAL	COMM	SAL+NVL(COMM,0)
CODD	ANALYST	3000	-	3000
ELMASRI	ANALYST	1200	150	1350
DATE	PROGRAMMER	1800	200	2000
RAMAKRISHNAN	ANALYST	3000	-	3000

Άσκηση 4. Τοποθέτησε τίτλους στις στήλες των αποτελεσμάτων της προηγούμενης αναζήτησης.

```
SELECT ENAME,JOB,SAL SALARY,COMM COMMISSION,
 SAL + NVL(COMM,0) TOTAL
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	JOB	SALARY	COMMISSION	TOTAL
CODD	ANALYST	3000	-	3000
ELMASRI	ANALYST	1200	150	1350
DATE	PROGRAMMER	1800	200	2000
RAMAKRISHNAN	ANALYST	3000	-	3000

Αν οι τίτλοι αποτελούνται από περισσότερες λέξεις:

```
SELECT ENAME,JOB,SAL,COMM,SAL+NVL(COMM,0) "TOTAL COMPENSATION"
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	JOB	SAL	COMM	TOTAL COMPENSATION
CODD	ANALYST	3000	-	3000
ELMASRI	ANALYST	1200	150	1350
DATE	PROGRAMMER	1800	200	2000
RAMAKRISHNAN	ANALYST	3000	-	3000

Άσκηση 5. 'Υψωσε τους αριθμούς Τμημάτων στις δυνάμεις του 2 και του 3.

```
SELECT * FROM dept
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

```
SELECT DNAME,DEPTNO,POWER(DEPTNO,2),POWER(DEPTNO,3)
FROM DEPT;
```

DNAME	DEPTNO	POWER(DEPTNO,2)	POWER(DEPTNO,3)
ACCOUNTING	10	100	1000
RESEARCH	20	400	8000
SALES	30	900	27000
OPERATIONS	40	1600	64000

Άσκηση 6. Υπολόγισε την ημερήσια αποζημίωση των υπαλλήλων του Τμήματος 10 υποθέτοντας ότι ο μήνας έχει 25 εργάσιμες ημέρες. Δείξε τα αποτελέσματα σε τρείς στήλες: Χωρίς στρογγύλευση, με στρογγύλευση σε ακέραιο αριθμό και στρογγύλευση σε ένα δεκαδικό ψηφίο.

```
SELECT EMPNO, ENAME,SAL,SAL/25,ROUND(SAL/25),ROUND(SAL/25,2)
FROM EMP
WHERE DEPTNO = 10;
```

EMPNO	ENAME	SAL	SAL/25	ROUND(SAL/25)	ROUND(SAL/25,2)
10	CODD	3000	120	120	120
15	ELMASRI	1200	48	48	48
30	DATE	1800	72	72	72
60	RAMAKRISHNAN	3000	120	120	120

Άσκηση 7. Υπολόγισε ημερήσια και ωριαία αποζημίωση για τους υπάλληλους της τμήματος 10.

```
SELECT ENAME,SAL MONTHLY,ROUND(SAL/25,2) DAILY,
  ROUND(SAL/(25*8),2) HOURLY
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	MONTHLY	DAILY	HOURLY
CODD	3000	120	15
ELMASRI	1200	48	6
DATE	1800	72	9
RAMAKRISHNAN	3000	120	15

Άσκηση 8. Το ίδιο μόνο αντί να χρησιμοποιήσεις ROUND χρησιμοποίησε TRUNC.

```
SELECT ENAME,SAL MONTHLY,TRUNC(SAL/25,2) DAILY,
  TRUNC(SAL/(25*8),2) HOURLY
FROM EMP
WHERE DEPTNO = 10;
```

ENAME	MONTHLY	DAILY	HOURLY
CODD	3000	120	15
ELMASRI	1200	48	6
DATE	1800	72	9
RAMAKRISHNAN	3000	120	15

Άσκηση 9. Δείξε τους υπάλληλους και χρησιμοποίησε τη συνάρτηση DECODE για να αντιστοιχίσεις στη θέση τους σε κατηγορίες.

Ακολούθησε τη παρακάτω ταξινόμηση:

ANALYST = 1, PROGRAMMER = 3 , PRESIDENT = 5, και όλες οι άλλες θέσεις = 2.

```
SELECT EMPNO, ENAME,JOB,DECODE(JOB,'ANALYST',1,'PROGRAMMER',3,
   'PRESIDENT',5,2) JOB_CLASS
FROM EMP;
```

EMPNO	ENAME	JOB	JOB_CLASS
10	CODD	ANALYST	1
15	ELMASRI	ANALYST	1
20	NAVATHE	SALESMAN	2
30	DATE	PROGRAMMER	3
40	WIDOM	MANAGER	2
50	WIDOM	SALESMAN	2
70	GEHRKE	PRESIDENT	5
60	RAMAKRISHNAN	ANALYST	1

Άσκηση 10. Δείξε τους μισθούς υπαλλήλων ορίζοντας τους υπάλληλους του Τμήματος 10 με τη θέση τους και όλους τους άλλους με το όνομα τους.

```
SELECT EMPNO, ENAME, DECODE(DEPTNO,10,JOB,ENAME) "EMP - ID", DEPTNO,SAL
FROM EMP;
```

EMPNO	ENAME	EMP - ID	DEPTNO	SAL
10	CODD	ANALYST	10	3000
15	ELMASRI	ANALYST	10	1200
20	NAVATHE	NAVATHE	20	2000
30	DATE	PROGRAMMER	10	1800
40	WIDOM	WIDOM	-	4000
50	WIDOM	WIDOM	-	2000
70	GEHRKE	GEHRKE	-	5000
60	RAMAKRISHNAN	ANALYST	10	3000

Στις επόμενες ασκήσεις χρησιμοποιούμε τις συναρτήσεις SUBSTR, UPPER, LOWER, INSTR, LENGTH

Άσκηση 11. Υπολόγισε την 3-γράμματη συντομογραφία των ονομάτων των τμημάτων.

```
SELECT DEPTNO, DNAME, SUBSTR(DNAME,1,3)
FROM DEPT;
```

DEPTNO	DNAME	SUBSTR(DNAME,1,3)
10	ACCOUNTING	ACC
20	RESEARCH	RES
30	SALES	SAL
40	OPERATIONS	OPE

Άσκηση 12. Δείξε ονόμα CODD με κεφαλαία και με πεζά.

```
SELECT ENAME,UPPER(ENAME),LOWER(ENAME)
FROM EMP
WHERE UPPER(ENAME) = UPPER('Codd');
```

ENAME	UPPER(ENAME)	LOWER(ENAME)
CODD	CODD	codd

Άσκηση 13. Βρές το γράμμα Α στα ονόματα υπαλλήλων στη πρώτη και την τέταρτη θέση της συμβολοσειράς του ονόματος.

```
SELECT EMPNO,  ENAME,INSTR(ENAME,'A',1) ,INSTR(ENAME,'A',4)
FROM EMP;
```

EMPNO	ENAME	INSTR(ENAME,'A',1)	INSTR(ENAME,'A',4)
10	CODD	0	0

15	ELMASRI	4	4
20	NAVATHE	2	4
30	DATE	2	0
40	WIDOM	0	0
50	WIDOM	0	0
70	GEHRKE	0	0
60	RAMAKRISHNAN	2	4

Άσκηση 14. Αριθμός χαρακτήρων (μήκος) στα ονόματα των τμημάτων.

```
SELECT DNAME,LENGTH(DNAME)
FROM DEPT;
```

DNAME	LENGTH(DNAME)
ACCOUNTING	10
RESEARCH	8
SALES	5
OPERATIONS	10

Άσκηση 15. Ποιός ο μέσος μισθός υπαλλήλων με θέση ANALYST

```
SELECT AVG(SAL)
FROM EMP
WHERE JOB = 'ANALYST';
```

AVG(SAL)
2400

Άσκηση 16. Ποιός ο συνολικός μισθός και η προμήθεια των υπαλλήλων με θέση ANALYST.

```
SELECT SUM(SAL),SUM(COMM)
FROM EMP
WHERE JOB = 'ANALYST';
```

SUM(SAL)	SUM(COMM)
7200	150

Προσοχή στην παρακάτω δήλωση!

```
SELECT job,SUM(SAL),SUM(COMM)
FROM EMP
WHERE JOB = 'ANALYST';
ORA-00937: not a single-group group function
```

Τι ισχύει στο προϊόν της MySQL;

Άσκηση 17. Ποιά η μέση ετήσια αμοιβή των υπαλλήλων με θέση ANALYST.

```
SELECT AVG(SAL + COMM) * 12
FROM EMP
WHERE JOB = 'ANALYST';
```

AVG(SAL+COMM)*12
16200

Άσκηση 18. Ποιός ο καλύτερα και ποιός ο χειρότερα αμοιβόμενος υπάλληλος και ποιά η διαφορά μισθού τους.

```
SELECT MAX(SAL), MIN(SAL), MAX(SAL) - MIN(SAL)
FROM EMP;
```

MAX(SAL)	MIN(SAL)	MAX(SAL)-MIN(SAL)
5000	1200	3800

Άσκηση 19. Ποιό το μεγαλύτερο μήκος ονόματος τμήματος.

```
SELECT MAX(LENGTH(DNAME))
FROM DEPT;
```

MAX(LENGTH(DNAME))
10

Άσκηση 20. Στοιχεία υπαλλήλου με το μεγαλύτερο μισθό.

```
SELECT ENAME,JOB,SAL
FROM EMP
WHERE SAL = (SELECT MAX(SAL)
 FROM EMP);
```

ENAME	JOB	SAL
GEHRKE	PRESIDENT	5000

Άσκηση 21. Πόσοι υπάλληλοι του Τμήματος 10 παίρνουν προμήθεια.

```
SELECT COUNT(COMM)
FROM EMP;
```

COUNT(COMM)
2

Άσκηση 22. Πόσες οι διαφορετικές θέσεις (jobs) στο τμήμα 10.

```
SELECT COUNT(DISTINCT JOB)
FROM EMP
WHERE DEPTNO = 10;
```

COUNT(DISTINCTJOB)
2

Άσκηση 23. Πόσοι εργάζονται στο τμήμα 10.

```
SELECT COUNT(*)
FROM EMP
WHERE DEPTNO = 10;
```

COUNT(*)
4

Άσκηση 24. Ποιός ο μέσος μισθός σε κάθε τμήμα,

```
SELECT DEPTNO,AVG(SAL)
FROM EMP
GROUP BY DEPTNO;
```

DEPTNO	AVG(SAL)
-	3666.6666666666666666666666666666666666667
20	2000
10	2250

Άσκηση 25. Ποιός ο μέσος ετήσιος μισθός για όλους τους υπάλληλους εκτός από τους υπαλλήλους που έχουν θέση MANAGER ή PRESIDENT

Η υποπρόταση GROUP BY DEPTNO διαμερίζει τους υπάλληλους (το σύνολο των υπαλλήλων) σε τρεις διακεκριμένες ομάδες (τρία ξένα υποσύνολα).

EMPNO	ENAME	….	DEPTNO
10	CODD		10
15	ELMASRI		10
30	DATE		10
60	RAMAKRISHNAN		10

20	NAVATHE	…….	20

40	WIDOM	MANAGER	-
50	WIDOM	SALESMAN	-
70	GEHRKE	PRESIDENT	-

```
SELECT DEPTNO,AVG(SAL) * 12
FROM EMP
WHERE JOB NOT IN ('MANAGER','PRESIDENT')
GROUP BY DEPTNO;
```

DEPTNO	AVG(SAL)*12
-	24000

20	24000
10	27000

Άσκηση 26. Χώρισε τους υπάλληλους ανά τμήμα και κατά θέση μέσα στο τμήμα. Μέτρησε τους υπάλληλους και υπολόγισε μέσο μισθό για κάθε ομάδα.

Η διαμέριση του συνόλου των υπαλλήλων γίνεται με την υποπρόταση GROUP BY και σύνθετο κριτήριο (DEPTNO, JOB).

```
SELECT DEPTNO,JOB,COUNT(*),AVG(SAL) * 12
FROM EMP
GROUP BY DEPTNO,JOB;
```

DEPTNO	JOB	COUNT(*)	AVG(SAL)*12
10	ANALYST	3	28800
10	PROGRAMMER	1	21600
-	SALESMAN	1	24000
-	PRESIDENT	1	60000
-	MANAGER	1	48000
20	SALESMAN	1	24000

Όπως βλέπουμε στην απάντηση οι υπάλληλοι οι οποίοι δεν έχουν ακόμη τοποθετηθεί σε τμήμα αποτελούν τρεις ομάδες σε σύνολο επτά ομάδων. Κάθε μία από τις τρεις ομάδες έχει ένα μόνο υπάλληλο.

40	WIDOM	MANAGER	-
50	WIDOM	SALESMAN	-
70	GEHRKE	PRESIDENT	-

Προσοχή στην παρακάτω δήλωση. Επειδή χρησιμοποιεί τη σύνδεση DEPT.DEPTNO = EMP.DEPTNO στα αποτελέσματα δεν περιλαμβάνονται οι υπάλληλοι 40, 50, 70.

```
SELECT DNAME,JOB,COUNT(*),AVG(SAL) * 12
FROM EMP,DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY DNAME,JOB;
```

DNAME	JOB	COUNT(*)	AVG(SAL)*12
ACCOUNTING	PROGRAMMER	1	21600
ACCOUNTING	ANALYST	3	28800
RESEARCH	SALESMAN	1	24000

Άσκηση 27. Ποιός ο μέσος μισθός των υπαλλήλων οι οποίοι κατέχουν την ίδια θέση για τις θέσεις που έχουν περισσότερους από δύο υπάλληλους.

Η δήλωση όταν περιλαμβάνει την υποπρόταση GROUP BY χωρίς HAVING υπολογίζει στοιχεία για τις παρακάτω εξ ομάδες.

JOB	COUNT(*)	…..
ANALYST	3	….

SALESMAN	2	…..
PROGRAMMER	1	…..
MANAGER	1	…..
PRESIDENT	1	…...

Άρα η πλήρης δήλωση δείχνει στοιχεία μόνο για τους υπαλλήλους που έχουν θέση ANALYST.

```
SELECT JOB,COUNT(*),AVG(SAL) * 12
FROM EMP
GROUP BY JOB
HAVING COUNT(*) > 2;
```

JOB	COUNT(*)	AVG(SAL)*12
ANALYST	3	28800

Άσκηση 28. Ποια τμήματα έχουν περισσότερους από έναν υπάλληλους με θέση ANALYST.

Σχηματίζουμε ομάδες με σύνθετο κριτήριο (job, deptno).

```
SELECT COUNT(*), JOB, DEPTNO
FROM EMP
GROUP BY DEPTNO, JOB;
```

COUNT(*)	JOB	DEPTNO
3	ANALYST	10
1	SALESMAN	20
1	PROGRAMMER	10
1	MANAGER	-
1	SALESMAN	-
1	PRESIDENT	-

Όλες οι γραμμές πλην της πρώτης εξαιρούνται λόγω της συνθήκης JOB = ‘ANALYST’. Η πρώτη γραμμή ικανοποιεί και τη συνθήκη της υποπρότασης HAVING.

```
SELECT DEPTNO, JOB, COUNT(*)
FROM EMP
WHERE JOB = ‘ANALYST’
GROUP BY DEPTNO, JOB
HAVING COUNT(*) >= 1;
```

DEPTNO	JOB	COUNT(*)
10	ANALYST	3

Άσκηση 29. Ποια τμήματα έχουν μέση προμήθεια μεγαλύτερη του 5% του μέσου μισθού.

```
SELECT DEPTNO,AVG(SAL),AVG(COMM),AVG(SAL) * 0.05
FROM EMP
GROUP BY DEPTNO
HAVING AVG(COMM) > AVG(SAL) * 0.05;
```

DEPTNO	AVG(SAL)	AVG(COMM)	AVG(SAL)*0.05
10	2250	175	112.5

Άσκηση 30. Θέσεις υπαλλήλων οι οποίες έχουν μέσο μισθό των υπαλλήλων τους ο οποίος είναι μεγαλύτερος του μέσου μισθού όλων των υπαλλήλων οι οποίοι είναι MANAGER.

```
SELECT JOB,AVG(SAL)
FROM EMP
GROUP BY JOB
HAVING AVG(SAL) > (SELECT AVG(SAL)
   FROM EMP
   WHERE JOB = 'MANAGER');
```

JOB	AVG(SAL)
PRESIDENT	5000

Άσκηση 31. Πόσοι υπάλληλοι στο τμήμα 10 έχουν μισθό και πόσοι προμήθεια.

```
SELECT COUNT(SAL),COUNT(COMM)
FROM EMP
WHERE DEPTNO = 10;
```

COUNT(SAL)	COUNT(COMM)
4	2

Η δήλωση

```
SELECT COUNT(SAL),COUNT(COMM)
FROM EMP
WHERE DEPTNO = 30;    -- το τμήμα 30 δεν έχει υπαλλήλους
```

έχει ως αποτέλεσμα

COUNT(SAL)	COUNT(COMM)
0	0

Άσκηση 32. Ποιά η μέση προμήθεια για τους υπάλληλους.

```
SELECT AVG(COMM)
FROM EMP;
```

AVG(COMM)
175

Προσοχή στην παρακάτω δήλωση!.

```
SELECT AVG(COMM),AVG(NVL(COMM,0))
FROM EMP;
```

AVG(COMM)	AVG(NVL(COMM,0))
175	43.75

Στις επόμενες ασκήσεις βλέπουμε πώς εργαζόμαστε με τιμές NULL

Άσκηση 33. Ποιοί υπάλληλοι δεν παίρνουν προμήθεια.

```
SELECT EMPNO, ENAME,SAL,COMM,JOB
FROM EMP
WHERE COMM IS NULL;
```

EMPNO	ENAME	SAL	COMM	JOB
10	CODD	3000	-	ANALYST
20	NAVATHE	2000	-	SALESMAN
40	WIDOM	4000	-	MANAGER
50	WIDOM	2000	-	SALESMAN
70	GEHRKE	5000	-	PRESIDENT
60	RAMAKRISHNAN	3000	-	ANALYST

Άσκηση 34. Ποιοί υπάλληλοι παίρνουν προμήθεια.

```
SELECT ENAME,SAL,COMM,JOB
FROM EMP
WHERE COMM IS NOT NULL;
```

ENAME	SAL	COMM	JOB
ELMASRI	1200	150	ANALYST
DATE	1800	200	PROGRAMMER

Άσκηση 35. Βρες υπάλληλους του τμήματος 10 με προμήθεια μικρότερη από 1000.

```
SELECT *
FROM EMP
WHERE DEPTNO = 10
AND COMM < 1000;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
15	ELMASRI	ANALYST	20	02/05/1995	1200	150	10
30	DATE	PROGRAMMER	10	04/05/2004	1800	200	10

Προσοχή! Δεν επιλέγονται όσοι έχουν προμήθεια με τιμή null.

4.9 Περιήγηση δεύτερη. Αναλυτική παρουσίαση συνδέσεων (join) πινάκων, φωλιασμένων αναζητήσεων, χρήσης συναρτήσεων (functions) και ομαδοποίησης δεδομένων (GROUP BY) σε Oracle.

Στη δεύτερη περιήγηση γίνεται συζήτηση των παρακάτω θεμάτων:

1) Εμβάθυνση στις συνδέσεις (join) πινάκων στις δηλώσεις SELECT. Περιπτώσεις JOIN, INNER JOIN, OUTER JOIN, RIGHT JOIN, LEFT JOIN, FULL JOIN.

2) Εμβάθυνση στις υποαναζητήσεις, δηλαδή στις εμφωλευμένες δηλώσεις SELECT μέσα σε δηλώσεις SELECT, π.χ., SELECT … SELECT

3) Εμβάθυνση στις συναρτήσεις (functions) και στη χρησιμοποίησή τους

4) Εμβάθυνση στην ομαδοποίηση δεδομένων (GROUP BY) και τη εξαγωγή στατιστικών

Στην Εικόνα 4.8 βλέπουμε τη βάση δεδομένων διεύθυνσης προσωπικού την οποία θα χρησιμοποιήσουμε στα παραδείγματα αναζήτησης.

(πίνακας στοιχείων υπαλλήλου) EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DNO

(πίνακας στοιχείων Τμημάτων στις οποίες ανήκουν οι υπάλληλοι) DEPT

DEPTNO	DNAME	LOC

όπου EMPNO = κωδικός υπαλλήλου, ENAME= ονοματεπώνυμο υπαλλήλου, JOB = θέση στην εταιρεία, MGR = ο επικεφαλής του, HIREDATE = ημερομηνία πρόσληψης, SAL= μισθός, COMM = προμήθεια, EMP.DNO και DEPT.DEPTNO = κωδικός Τμήματος, DNAME = όνομα Τμήματος, LOC = έδρα Τμήματος. Υποτίθεται ότι κάθε υπάλληλος ανήκει σε ένα Τμήμα.

Εικόνα 4.8 Βάση δεδομένων διεύθυνσης προσωπικού

Στην Εικόνα 4.9 βλέπουμε τον πίνακα των τμημάτων της εταιρείας και τον πίνακα των υπαλλήλων της εταιρείας και δείγμα δεδομένων.

4.9.1 Δημιουργία Πινάκων στο προϊόν της Oracle και δείγμα δεδομένων

Δημιουργία πινάκων

```
CREATE TABLE DEPT(DEPTNO NUMBER(2) NOT NULL,
 DNAME VARCHAR2(14), LOC VARCHAR(14),
 PRIMARY KEY(DEPTNO));
```

```
CREATE TABLE EMP(EMPNO NUMBER(4) NOT NULL,
ENAME VARCHAR2(10), JOB VARCHAR2(9),
MGR NUMBER(4), HIREDATE DATE,
SAL NUMBER(7,2), COMM NUMBER(7,2),
DNO NUMBER(2),
 PRIMARY KEY(EMPNO),
 FOREIGN KEY(DNO) REFERENCES DEPT(DEPTNO));
```

Πως βλέπουμε τους πίνακες της βάσης δεδομένων

```
SELECT * FROM TAB;
```

Πως βλέπουμε τη δομή των πινάκων

```
DESCRIBE dept;
DESCRIBE emp;
```

Εισαγωγή δεδομένων

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
```

```
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (40, 'OPERATIONS', 'BOSTON');
```

```
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
 VALUES (7369, 'SMITH', 'CLERK', 7902, '17/12/2000', 800, NULL, 20);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
  VALUES (7499, 'ALLEN', 'SALESMAN', 7698, '20/02/2001', 1600, 300, 30);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
  VALUES (7521, 'WARD', 'SALESMAN', 7698, '22/02/2001', 1250, 500, 30);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
  VALUES (7566, 'JONES', 'MANAGER', 7839, '02/04/2001', 2975, NULL, 20);
INSERT INTO EMP
  (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
   VALUES (7654, 'MARTIN', 'SALESMAN', 7698, '28/09/2001', 1250, 1400, 30);
INSERT INTO EMP
   (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7698, 'BLAKE', 'MANAGER', 7839, '01/05/2001', 2850, NULL, 30);
INSERT INTO EMP
(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
   VALUES (7782, 'CLARK', 'MANAGER', 7839, '09/06/2001', 2450, NULL, 10);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
 VALUES (7788, 'SCOTT', 'ANALYST', 7566, '19/04/2007', 3000, NULL, 20);
INSERT INTO EMP
   (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7839, 'KING', 'PRESIDENT', NULL, '17/11/2001', 5000, NULL, 10);
INSERT INTO EMP
   (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7844, 'TURNER', 'SALESMAN', 7698, '08/09/2001', 1500, 0, 30);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7876, 'ADAMS', 'CLERK', 7788, '23/05/2007', 1100, NULL, 20);
INSERT INTO EMP
  (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7900, 'JAMES', 'CLERK', 7698, '2003/12/12', 950, NULL, 30);
INSERT INTO EMP
   (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7902, 'FORD', 'ANALYST', 7566, '03/12/2001', 3000, NULL, 20);
INSERT INTO EMP
  (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7934, 'MILLER', 'CLERK', 7782, '23/01/2002', 1300, NULL, 10);
INSERT INTO EMP
  (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
```

```
VALUES (7999, 'BATES', 'ANALYST', 7566, '04/01/2014', 1300, NULL, NULL);


SELECT * FROM EMP;
SELECT * FROM DEPT;
```

4.9.2 Σύνταξη δήλωσης SELECT η οποία περιλαμβάνει σύνδεση δύο πινάκων ή υποαναζήτηση.

Μία συνηθισμένη σύνταξη δήλωσης αναζήτησης (SELECT) η οποία συνδέει δύο πίνακες είναι:

```
SELECT  όνομα_πίνακα1.όνομα_στήλης, ....
όνομα_πίνακα2.όνομα_στήλης, ....
FROM όνομα_πίνακα1 , όνομα_πίνακα2
WHERE όνομα_πίνακα1.όνομα_στήλης=όνομα_πίνακα2.όνομα_στήλης;
```

Εναλλακτικά,

```
SELECT  όνομα_πίνακα1.όνομα_στήλης, ....
όνομα_πίνακα2.όνομα_στήλης, ....
FROM όνομα_πίνακα1
INNER JOIN όνομα_πίνακα2
ON όνομα_πίνακα1.όνομα_στήλης=όνομα_πίνακα2.όνομα_στήλης;
```

```
SELECT  όνομα_πίνακα1.όνομα_στήλης, ....
όνομα_πίνακα2.όνομα_στήλης, ....
FROM όνομα_πίνακα1
JOIN όνομα_πίνακα2
ON όνομα_πίνακα1.όνομα_στήλης=όνομα_πίνακα2.όνομα_στήλης;
```

Μία συνηθισμένη σύνταξη δήλωσης αναζήτησης (SELECT) η οποία περιλαμβάνει υποαναζήτηση (Subquery Syntax):

SELECT statement

(SELECT statement) ….;

δηλαδή έχουμε δήλωση SELECT εμφωλευμένη μέσα σε άλλη δήλωση SELECT (statement). Τα προϊόντα συχνά υποστηρίζουν το πρότυπο (SQL standard) με κάποιες επεκτάσεις.

Ακολουθούν απλά παραδείγματα σύνταξης δήλωσης η οποία περιλαμβάνει υποαναζήτηση με χρήση τελεστή σύγκρισης (>,<,>=,<=,<>) κ.λπ.

Υποτίθεται ότι column1, column2 λαμβάνουν τιμές στο ίδιο πεδίο ορισμού, π.χ., column1 number(7,2), column2 number(7,2)

```
SELECT columns
FROM table1
WHERE column1=(SELECT column2 FROM table2 ….);
```

```
SELECT columns
FROM table1
WHERE column1 IN (SELECT column2 FROM table2 ….);
```

```
SELECT columns
FROM table1
WHERE (column1, …, column_x) IN (SELECT column1, …, column_x FROM table2 ….);
```

```
SELECT columns
FROM table1
WHERE column1 > (SELECT column2 FROM table2 ….);
```

```
SELECT columns
FROM table1
WHERE column1 >= (SELECT column2 FROM table2 …);
```

```
SELECT columns
FROM table1
WHERE column1 < (SELECT column2 FROM table2 ….);
```

```
SELECT columns
FROM table1
WHERE column1 <= (SELECT column2 FROM table2 ….);
```

```
SELECT columns
FROM table1
WHERE column1 <> (SELECT column2 FROM table2 ….);
```

Στην Εικόνα 4.9, στη συνέχεια δίνεται δείγμα δεδομένων των πινάκων της βάσης δεδομένων διεύθυνσης προσωπικού το οποίο θα χρησιμοποιηθεί στα επόμενα παραδείγματα. Εμφανίζονται ο πίνακας με τα στοιχεία των υπαλλήλων και ο πίνακας με τα στοιχεία των τμημάτων.

Σημειώνουμε ότι ο κωδικός τμήματος στον πίνακα emp είναι η στήλη dno ενώ στον πίνακα dept είναι η στήλη deptno.

Πίνακας τμημάτων εταιρείας DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Πίνακας υπαλλήλων εταιρείας EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DNO
7369	SMITH	CLERK	7902	17-12-2000	800.00	NULL	20
7499	ALLEN	SALESMAN	7698	20-02-2001	1600.00	300.00	30
7521	WARD	SALESMAN	7698	22-02-2001	1250.00	500.00	30
7566	JONES	MANAGER	7839	02-04-2001	2975.00	NULL	20
7654	MARTIN	SALESMAN	7698	28-09-2001	1250.00	1400.00	30
7698	BLAKE	MANAGER	7839	01-05-2001	2850.00	NULL	30
7782	CLARK	MANAGER	7839	09-06-2001	2450.00	NULL	10
7788	SCOTT	ANALYST	7566	19-04-2007	3000.00	NULL	20
7839	KING	PRESIDENT	NULL	17-11-2001	5000.00	NULL	10
7844	TURNER	SALESMAN	7698	08-09-2001	1500.00	0.00	30
7876	ADAMS	CLERK	7788	23-05-2007	1100.00	NULL	20
7900	JAMES	CLERK	7698	03-12-2001	950.00	NULL	30
7902	FORD	ANALYST	7566	03-12-2001	3000.00	NULL	20
7934	MILLER	CLERK	7782	23-01-2002	1300.00	NULL	10
7999	BATES	ANALYST	7566	04-01-2014	1300.00	NULL	NULL

Εικόνα 4.9 Πίνακας με στοιχεία υπαλλήλων (EMP) και πίνακας με στοιχεία τμημάτων (DEPT)

4.9.3 Παραδείγματα σύνταξης δηλώσεων SELECT που περιλαμβάνουν σύνδεση δύο πινάκων

Ακολουθούν τρεις τρόποι σύνταξης δήλωσης SELECT που περιλαμβάνει σύνδεση δύο πινάκων. Οι δηλώσεις είναι ισοδύναμες, δηλαδή οδηγούν στα ίδια αποτελέσματα.

```
SELECT *
FROM emp, dept
WHERE emp.dno=dept.deptno;
```

```
SELECT *
FROM emp
INNER JOIN dept
ON emp.dno=dept.deptno;
```

```
SELECT *
FROM emp
JOIN dept
ON emp.dno=dept.deptno;
```

Προσοχή! Η δήλωση **SELECT * FROM emp, dept;** δεν είναι ισοδύναμη με τις τρεις δηλώσεις αλλά εμφανίζει το καρτεσιανό γινόμενο των δύο πινάκων.

Τα αποτελέσματα της σύνδεσης παρατίθενται στη συνέχεια στον πίνακα 4.6. Παρατηρήστε ότι στα αποτελέσματα δεν περιλαμβάνονται:

1) Η υπάλληλος BATES επειδή δεν έχει τοποθετηθεί σε τμήμα. Αν θέλουμε να την συμπεριλάβουμε στα αποτελέσματα απαιτείται LEFT JOIN.
SELECT * FROM emp LEFT JOIN dept ON emp.dno=dept.deptno;

2) Το τμήμα OPERATIONS επειδή δεν έχει ακόμη υπαλλήλους. Αν θέλουμε να το συμπεριλάβουμε απαιτείται RIGHT JOIN.

3) SELECT * FROM emp RIGHT JOIN dept ON emp.dno=dept.deptno;
Οι δύο πίνακες συνδέονται με βάση τις κοινές τιμές στις στήλες dno, deptno. Για παράδειγμα, στην πρώτη γραμμή βλέπουμε τα στοιχεία του υπαλλήλου 7782 (CLARK)

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DNO
7782	CLARK	MANAGER	7839	09-06-2001	2450.00	NULL	10

να παρατίθενται με τα πλήρη στοιχεία του τμήματός του

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK

Πίνακας 4.4. Οι δύο πίνακες συνδέονται με βάση τις κοινές τιμές στις στήλες dno, deptno.

Empno	Ename	Job	Mgr	Hiredate	Sal	Comm	Dno	Deptno	Dname	Loc
7782	CLARK	MANAGER	7839	09/06/01	2450		10	10	ACCOUNTING	NEW YORK
7839	KING	PRESIDENT		17/11/01	5000		10	10	ACCOUNTING	NEW YORK
7934	MILLER	CLERK	7782	23/01/02	1300		10	10	ACCOUNTING	NEW YORK
7369	SMITH	CLERK	7902	17/12/00	800		20	20	RESEARCH	DALLAS
7566	JONES	MANAGER	7839	02/04/01	2975		20	20	RESEARCH	DALLAS
7788	SCOTT	ANALYST	7566	19/04/07	3000		20	20	RESEARCH	DALLAS
7876	ADAMS	CLERK	7788	23/05/07	1100		20	20	RESEARCH	DALLAS
7902	FORD	ANALYST	7566	03/12/01	3000		20	20	RESEARCH	DALLAS
7499	ALLEN	SALESMAN	7698	20/02/01	1600	300	30	30	SALES	CHICAGO
7521	WARD	SALESMAN	7698	22/02/01	1250	500	30	30	SALES	CHICAGO
7654	MARTIN	SALESMAN	7698	28/09/01	1250	1400	30	30	SALES	CHICAGO
7698	BLAKE	MANAGER	7839	01/05/01	2850		30	30	SALES	CHICAGO
7844	TURNER	SALESMAN	7698	08/09/01	1500	0	30	30	SALES	CHICAGO
7900	JAMES	CLERK	7698	03/12/01	950		30	30	SALES	CHICAGO

4.9.4 Δηλώσεις SELECT που περιλαμβάνουν σύνδεση δύο πινάκων και υπολογισμούς

Ακολουθεί παράδειγμα και δηλώσεις SELECT που περιλαμβάνουν σύνδεση πινάκων, υπολογισμούς και χρήση της συνάρτησης NVL.

Δείξτε κωδικό, όνομα, θέση, μισθό, προμήθεια και σύνολο μισθού και προμήθειας, κωδικό τμήματος και τμήμα όλων των υπαλλήλων.

Δείτε πίνακες DEPT, EMP της εικόνας 4.9 για τον υπολογισμό των αποτελεσμάτων τα οποία βλέπουμε στον πίνακα 4.7.

Η συνάρτηση εξασφαλίζει ότι όταν η στήλη COMM δεν έχει τιμή τότε η τιμή της γίνεται 0 ώστε να υπολογίζεται σωστά το άθροισμα μισθού και προμήθειας.

```
SELECT empno, ename, job, sal, comm, sal+NVL(comm,0) "sal+comm",
 dno, deptno, dname
FROM emp, dept
WHERE emp.dno = dept.deptno;
```

```
SELECT empno, ename, job, sal, comm, sal+NVL(comm,0) "sal+comm",
dno, deptno, dname
FROM emp
INNER JOIN dept ON emp.dno = dept.deptno;
```

```
SELECT empno, ename, job, sal, comm, sal+NVL(comm,0) "sal+comm",
 dno, deptno, dname
FROM emp
JOIN dept ON emp.dno = dept.deptno;
```

4.9.5 Δηλώσεις SELECT που περιλαμβάνουν σύνδεση δύο πινάκων και συνθήκη (WHERE)

Ακολουθεί παράδειγμα και δηλώσεις SELECT που περιλαμβάνουν σύνδεση πινάκων, υπολογισμούς και συνθήκη (υποπρόταση) WHERE.

Δείξτε κωδικό, όνομα, θέση, μισθό, προμήθεια και σύνολο μισθού και προμήθειας, κωδικό τμήματος και τμήμα των αναλυτών και των πωλητών.

Δείτε πίνακες DEPT, EMP της εικόνας 4.9 για τον υπολογισμό των αποτελεσμάτων.

Πίνακας 4.5 Αποτέλεσμα δηλώσεως που περιλαμβάνει σύνδεση πινάκων DEPT, EMP

Empno	Ename	Job	Sal	Comm	Sal+comm	Dno	Deptno	Dname
7782	CLARK	MANAGER	2450		2450	10	10	ACCOUNTING
7839	KING	PRESIDENT	5000		5000	10	10	ACCOUNTING
7934	MILLER	CLERK	1300		1300	10	10	ACCOUNTING

7369	SMITH	CLERK	800		800	20	20	RESEARCH
7566	JONES	MANAGER	2975		2975	20	20	RESEARCH
7788	SCOTT	ANALYST	3000		3000	20	20	RESEARCH
7876	ADAMS	CLERK	1100		1100	20	20	RESEARCH
7902	FORD	ANALYST	3000		3000	20	20	RESEARCH
7499	ALLEN	SALESMAN	1600	300	1900	30	30	SALES
7521	WARD	SALESMAN	1250	500	1750	30	30	SALES
7654	MARTIN	SALESMAN	1250	1400	2650	30	30	SALES
7698	BLAKE	MANAGER	2850		2850	30	30	SALES
7844	TURNER	SALESMAN	1500	0	1500	30	30	SALES
7900	JAMES	CLERK	950		950	30	30	SALES

```
SELECT empno, ename, job, sal, comm, sal+IFNULL(comm,0) "sal+comm",
  dno, deptno, dname
FROM emp, dept
WHERE emp.dno = dept.deptno
AND job IN ('ANALYST', 'SALESMAN');
```

```
SELECT empno, ename, job, sal, comm, sal+IFNULL(comm,0) "sal+comm",
  dno, deptno, dname
FROM emp
INNER JOIN dept ON emp.dno = dept.deptno
AND job IN ('ANALYST', 'SALESMAN');
```

```
SELECT empno, ename, job, sal, comm, sal+IFNULL(comm,0) "sal+comm",
  dno, deptno, dname
FROM emp
JOIN dept ON emp.dno = dept.deptno
WHERE job IN ('ANALYST', 'SALESMAN');
```

Δείτε τα αποτελέσματα στον πίνακα 4.6

Πίνακας 4.6 Αποτελέσματα δήλωσης που περιλαμβάνει σύνδεση πινάκων και συνθήκη WHERE

Empno	Ename	Job	Sal	Comm	Sal+ Comm	Dno	Deptno	Dname
7788	SCOTT	ANALYST	3000		3000	20	20	RESEARCH
7902	FORD	ANALYST	3000		3000	20	20	RESEARCH
7499	ALLEN	SALESMAN	1600	300	1900	30	30	SALES
7521	WARD	SALESMAN	1250	500	1750	30	30	SALES

7654	MARTIN	SALESMAN	1250	1400	2650	30	30	SALES
7844	TURNER	SALESMAN	1500	0	1500	30	30	SALES

Σημείωση. Το προϊόν MySQL αντί της συνάρτησης NVL-Null VaLue διαθέτει τη συνάρτηση IFNULL.

4.9.6 Συνδέσεις SELECT που περιλαμβάνουν LEFT JOIN, RIGHT JOIN, FULL JOIN

Στην ενότητα αυτή παραθέτουμε παραδείγματα δηλώσεων που περιλαμβάνουν συνδέσεις στο προϊόν της ORACLE..

 # left outer join - oracle only

```
SELECT empno, ename, job, sal, comm, sal+NVL(comm,0), dno, deptno, dname
FROM emp, dept
WHERE emp.dno=dept.deptno(+);
```

 # right outer join - oracle only

```
SELECT empno, ename, job, sal, comm, sal+NVL(comm,0), dno, deptno, dname
FROM emp, dept
WHERE emp.dno(+)=dept.deptno;
```

Σημαντική παρατήρηση

Emp No	Ename	Job	Sal	Comm	Sal+ Comm	Dno	Deptno	Dname
7788	SCOTT	ANALYST	3000		3000	20	20	RESEARCH
7902	FORD	ANALYST	3000		3000	20	20	RESEARCH
7499	ALLEN	SALESMAN	1600	300	1900	30	30	SALES
7521	WARD	SALESMAN	1250	500	1750	30	30	SALES
7654	MARTIN	SALESMAN	1250	1400	2650	30	30	SALES
7844	TURNER	SALESMAN	1500	0	1500	30	30	SALES

Η παραπάνω σύνταξη της δήλωσης SELECT με left (outer) join και της δήλωσης SELECT με right (outer) join δεν ισχύουν στο προϊόν mySQL:

```
mysql> SELECT empno, ename, job, sal, emp.dno, dname
-> FROM emp, dept
-> WHERE emp.dno = dept.deptno(+);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use …
```

Στη συνέχεια βλέπουμε τη σύνταξη δήλωσης SELECT που περιλαμβάνει LEFT JOIN ή RIGHT JOIN.

 # ORACLE: LEFT JOIN

```
SELECT empno, ename, job, sal, comm, sal+NVL(comm,0) "sal+comm",
 dno, deptno, dname
```

```
FROM emp
LEFT JOIN dept ON emp.dno = dept.deptno;
```

 # ORACLE: RIGHT JOIN

```
SELECT empno, ename, job, sal, comm, sal+NVL(comm,0) "sal+comm",
 dno, deptno, dname
FROM emp
RIGHT JOIN dept ON emp.dno = dept.deptno;
```

Η ίδια σύνταξη ισχύει και στο προϊόν MySQL.

 # MySQL: LEFT JOIN, RIGHT JOIN

```
SELECT empno, ename, job, sal, comm, sal+IFNULL(comm,0) "sal+comm",
 dno, deptno, dname
FROM emp
LEFT JOIN dept ON emp.dno = dept.deptno;
```

```
SELECT empno, ename, job, sal, comm, sal+IFNULL(comm,0) "sal+comm",
 dno, deptno, dname
FROM emp
RIGHT JOIN dept ON emp.dno = dept.deptno;
```

 # full join - oracle only

```
SELECT empno, ename, job, sal, comm, sal+NVL(comm,0), dno, deptno, dname
FROM emp
FULL JOIN dept
ON emp.dno=dept.deptno;
```

Η σύνταξη δήλωσης SELECT με χρήση FULLJOIN δεν υποστηρίζεται στο προϊόν MySQL.

Η παρακάτω δήλωση εκτελείται και στα δύο προϊόντα.

 # FULL JOIN. MySQL and ORACLE

```
SELECT empno, ename, job, sal, emp.dno, dname
FROM emp
LEFT JOIN dept ON emp.dno = dept.deptno
UNION
SELECT empno, ename, job, sal, emp.dno, dname
FROM emp
RIGHT JOIN dept ON emp.dno = dept.deptno;
```

4.9.7 Δήλωση SELECT με χρήση JOIN vs δήλωση SELECT με χρήση Subquery

Σε πολλές περιπτώσεις έχουμε τη δυνατότητα να απαντήσουμε σε ερωτήματα είτε με χρήση σύνδεσης ή με χρήση υποαναζήτησης. Ακολουθεί παράδειγμα.

Δείξτε τους υπάλληλους του Τμήματος ACCOUNTING.

Δείτε πίνακες DEPT, EMP της εικόνας4.9 για τον υπολογισμό των αποτελεσμάτων.

 # χρήση υποαναζήτησης

```
SELECT *
FROM emp
WHERE dno IN (SELECT deptno
  FROM dept
  WHERE dname='ACCOUNTING');
```

 # χρήση σύνδεσης

```
SELECT *
FROM emp
JOIN dept ON emp.dno=dept.deptno
AND dname='ACCOUNTING';
```

 # σύνδεση

```
SELECT *
FROM emp
INNER JOIN dept ON emp.dno=dept.deptno
WHERE dname='ACCOUNTING';
```

 # σύνδεση

```
SELECT *
FROM emp, dept
WHERE emp.dno=dept.deptno
AND dname='ACCOUNTING';
```

4.9.8 Ομαδοποίηση δεδομένων (GROUP BY) και υπολογισμός στατιστικών

Όταν θέλουμε να υπολογίσουμε στατιστικά συνήθως χρησιμοποιούμε δήλωση SELECT με GROUP BY. Ακολουθεί παράδειγμα.

Δείξτε για τους υπαλλήλους κάθε τμήματος μέσο όρο μισθού, μέγιστο και ελάχιστο μισθό, άθροισμα μισθών, πόσοι έχουν μισθό και πόσοι είναι οι υπάλληλοι του τμήματος.

Δείτε πίνακες DEPT, EMP της εικόνας 4.9 για τον υπολογισμό των αποτελεσμάτων.

 # Σωστή δήλωση

```
SELECT dno, AVG(sal), MAX(sal), MIN(sal), SUM(sal), COUNT(sal), COUNT(*)
FROM emp
GROUP BY dno;
```

 # Λανθασμένη απάντηση

```
SELECT dno, AVG(sal), MAX(sal), MIN(sal), SUM(sal), COUNT(sal), COUNT(*)
FROM emp;
```

4.9.9 Περιπτώσεις συνδέσεων σε ορολογία Oracle/PLSQL και άλλων γνωστών προϊόντων.

Έχουμε τις παρακάτω περιπτώσεις συνδέσεων (Joins):

- Oracle INNER JOIN (ή simple join)
- Oracle LEFT OUTER JOIN (ή LEFT JOIN)

- Oracle RIGHT OUTER JOIN (ή RIGHT JOIN)
- Oracle FULL OUTER JOIN (ή FULL JOIN)

4.9.10 Θέματα προς επίλυση

Αφήνεται ως άσκηση η διεκπεραίωση των θεμάτων της περιήγησης, που είδαμε στο παρόν κεφάλαιο, στο προϊόν της MySQL. Ακολουθούν οι δηλώσεις δημιουργίας βάσης δεδομένων και πινάκων και οι δηλώσεις εισαγωγής στοιχείων στους πίνακες.

Δημιουργία βάσης και πινάκων.

```
CREATE DATABASE personnel;

USE personnel;

CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL,
 DNAME VARCHAR(14), LOC VARCHAR(14),
 PRIMARY KEY(DEPTNO));

CREATE TABLE EMP(EMPNO INT(4) NOT NULL,
ENAME VARCHAR(10), JOB VARCHAR(9),
MGR INT(4), HIREDATE DATE,
SAL FLOAT(7,2), COMM FLOAT(7,2),
DNO INT(2),
 PRIMARY KEY(EMPNO),
 FOREIGN KEY(DNO) REFERENCES DEPT(DEPTNO));
```

Πως βλέπουμε τους πίνακες της βάσης δεδομένων

```
SHOW TABLES;
```

Πως βλέπουμε τη δομή των πινάκων

```
DESCRIBE dept;
DESCRIBE emp;
```

Εισαγωγή δεδομένων

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (40, 'OPERATIONS', 'BOSTON');
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
 VALUES (7369, 'SMITH', 'CLERK', 7902, '1980/12/17', 800, NULL, 20);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
  VALUES (7499, 'ALLEN', 'SALESMAN', 7698, '1981/02/20', 1600, 300, 30);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
```

```
  VALUES (7521, 'WARD', 'SALESMAN', 7698, '2002/02/01', 1250, 500, 30);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
  VALUES (7566, 'JONES', 'MANAGER', 7839, '1981/12/24', 2975, NULL, 20);
INSERT INTO EMP
  (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
   VALUES (7654, 'MARTIN', 'SALESMAN', 7698, '1981/10/28', 1250, 1400, 30);
INSERT INTO EMP
   (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7698, 'BLAKE', 'MANAGER', 7839, '2001/05/02', 2850, NULL, 30);
INSERT INTO EMP
(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
   VALUES (7782, 'CLARK', 'MANAGER', 7839, '1981/11/27', 2450, NULL, 10);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
 VALUES (7788, 'SCOTT', 'ANALYST', 7566, '1987/04/29', 3000, NULL, 20);
INSERT INTO EMP
   (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7839, 'KING', 'PRESIDENT', NULL, '1987/11/12', 5000, NULL, 10);
INSERT INTO EMP
   (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7844, 'TURNER', 'SALESMAN', 7698, '2007/10/19', 1500, 0, 30);
INSERT INTO EMP
 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7876, 'ADAMS', 'CLERK', 7788, '2003/05/07', 1100, NULL, 20);
INSERT INTO EMP
  (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7900, 'JAMES', 'CLERK', 7698, '2003/12/12', 950, NULL, 30);
INSERT INTO EMP
   (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7902, 'FORD', 'ANALYST', 7566, '2003/12/19', 3000, NULL, 20);
INSERT INTO EMP
  (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7934, 'MILLER', 'CLERK', 7782, '2003/01/19', 1300, NULL, 10);
INSERT INTO EMP
  (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DNO)
VALUES (7999, 'BATES', 'ANALYST', 7566, '2004/01/04', 1300, NULL, NULL);

SELECT * FROM EMP;
SELECT * FROM DEPT;
```

4.10 Περιήγηση (Tour on SQL). Περαιτέρω συζήτηση σημαντικών για τις εφαρμογές δηλώσεων SELECT σε MySQL. Χρήση Βάσης Δεδομένων Προσωπικού και Διαχείρισης Έργων, Βάσης Ανταλλακτικών. Λυμένες ασκήσεις με χρήση MySQL

Ακολουθεί βάση δεδομένων που θα χρησιμοποιηθεί στα παραδείγματα αναζήτησης.

(πίνακας στοιχείων υπαλλήλου) EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO

(πίνακας στοιχείων Τμημάτων στις οποίες ανήκουν οι υπάλληλοι)

DEPT

DEPTNO	DNAME	LOC

(πίνακας στοιχείων έργων της εταιρείας) PROJ

PROJNO	PNAME	BUDGET

(πίνακας απασχόλησης υπαλλήλων σε έργα της εταιρείας) ASSIGN

EMPNO	PROJNO	PTIME

όπου EMPNO = κωδικός υπαλλήλου, ENAME= ονοματεπώνυμο υπαλλήλου, JOB = θέση στην εταιρεία, MGR = ο επικεφαλής του, SAL= μισθός, COMM = προμήθεια, DEPTNO = κωδικός Τμήματος.

Υποτίθεται ότι κάθε υπάλληλος ανήκει σε ένα Τμήμα, DNAME = όνομα Τμήματος, LOC = έδρα Τμήματος, PROJNO = κωδικός έργου, PNAME = περιγραφή έργου, BUDGET = προϋπολογισμός έργου, PTIME = ποσοστό χρόνου απασχόλησης υπαλλήλου σε έργο. Υποτίθεται ότι κάθε υπάλληλος μπορεί να εργάζεται σε περισσότερα από ένα έργα.

Εικόνα 4.10 Τρίτη κανονική μορφή βάσης δεδομένων που θα χρησιμοποιηθεί στα παραδείγματα

Παραθέτουμε τις δηλώσεις δημιουργίας της βάσης δεδομένων και των πινάκων της και τις δηλώσεις εισαγωγής στοιχείων στους πίνακες με χρήση του προϊόντος mySQL.

Δημιουργία της βάσης

```
DROP DATABASE IF EXISTS join_examples;
CREATE DATABASE join_examples;
```

Χρήση βάσης

```
USE join_examples;
```

Δημιουργία πινάκων

```
CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL,
 DNAME VARCHAR(14), LOC VARCHAR(14),
 PRIMARY KEY(DEPTNO));
CREATE TABLE EMP(EMPNO INT(4) NOT NULL,
  ENAME VARCHAR(10), JOB VARCHAR(25),
  HIREDATE DATE, MGR INT(4),  SAL FLOAT(7,2), COMM FLOAT(7,2),
  DEPTNO INT(2),
  PRIMARY KEY(EMPNO),
  FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO));
CREATE TABLE PROJ (projno INT(3) NOT NULL,
   pname VARCHAR(15),
   budget FLOAT(12,2),
 PRIMARY KEY(projno));
CREATE TABLE ASSIGN(
 EMPNO INT(4) NOT NULL, PROJNO INT(3) NOT NULL,
 PTIME INT(3),
PRIMARY KEY(EMPNO,PROJNO),
```

```
FOREIGN KEY(EMPNO) REFERENCES EMP(EMPNO),
FOREIGN KEY(PROJNO) REFERENCES PROJ(PROJNO));


SHOW TABLES;
```

Αν θέλουμε να δούμε τη δομή των πινάκων γράφουμε εντολή describe.

```
describe emp;
```

Εισαγωγή στοιχείων στους πίνακες

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (40, 'OPERATIONS', 'BOSTON');
INSERT INTO EMP
  VALUES (10, 'CODD', 'ANALYST', '1989/01/01', 15, 3000, NULL, 10);
INSERT INTO EMP
  VALUES (15, 'ELMASRI', 'ANALYST', '1995/05/02', 15, 1200, 150, 10);
INSERT INTO EMP
  VALUES (20, 'NAVATHE', 'SALESMAN', '1977/07/07', 20, 2000, NULL, 20);
INSERT INTO EMP
  VALUES (30, 'DATE', 'PROGRAMMER', '2004/05/04', 15, 1800, 200, 10);
INSERT INTO EMP
  VALUES (11, 'CODD', 'ANALYST', '1989/01/01', 15, 2900, NULL, 10);
INSERT INTO EMP
  VALUES (12, 'CODD', 'PROGRAMMER', '1995/05/02', 15, 1200, 150, 10);
INSERT INTO EMP
  VALUES (21, 'CODD', 'SALESMAN', '1977/07/07', 20, 2000, NULL, 10);
INSERT INTO EMP
  VALUES (22, 'CODD', 'PROGRAMMER', '1995/05/02', 15, 1200, 150, 20);
INSERT INTO EMP
  VALUES (23, 'CODD', 'SALESMAN', '1977/07/07', 20, 2000, NULL, 20);
INSERT INTO EMP
  VALUES (77, 'BATES', 'SALESMAN', '1987/07/07', 20, 2000, NULL, NULL);
```

```
INSERT INTO proj(projno, pname, budget)
VALUES(100, 'PAYROLL', 100000);
INSERT INTO proj(projno, pname, budget)
VALUES(200, 'PERSONNEL',200000 );
INSERT INTO proj(projno, pname, budget)
VALUES(300, 'SALES', 150000);
INSERT INTO assign(empno, projno, ptime)
 VALUES(10,100, 40);
INSERT INTO assign(empno, projno, ptime)
 VALUES(10, 200, 60);
INSERT INTO assign(empno, projno, ptime)
 VALUES(15, 100, 100);
```

```
INSERT INTO assign(empno, projno, ptime)
 VALUES(20, 200, 100);
INSERT INTO assign(empno, projno, ptime)
 VALUES(30, 100, 100);
```

Πως βλέπουμε τα δεδομένα.

```
SELECT * FROM DEPT;
SELECT * FROM EMP;
```

Στην Εικόνα 4.11 βλέπουμε τους πίνακες με δείγμα δεδομένων που θα χρησιμοποιηθεί στα παραδείγματα.

SELECT * FROM dept

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

SELECT * FROM emp

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	15	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	15	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	20	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	15	04/05/2004	1800	200	10

SELECT * FROM proj

PROJNO	PNAME	BUDGET
100	PAYROLL	100000
200	PERSONNEL	200000
300	SALES	150000

SELECT * FROM assign

EMPNO	PROJNO	PTIME
10	100	40
10	200	60
15	100	100
20	200	100
30	100	100

Εικόνα 4.11 Δείγμα δεδομένων στους πίνακες της τρίτης κανονικής μορφής που θα χρησιμοποιηθούν στα παραδείγματα

4.10.1 Συναρτήσεις

Τις συναρτήσεις μπορούμε να τις χρησιμοποιήσουμε αυτοτελώς ή μαζί με στοιχεία των πινάκων. Ακολουθούν παραδείγματα με χρήση του προϊόντος mySQL.

Δείξτε συμβολοσειρές (string) μαζί.

```
SELECT CONCAT(' Mr ', 'and Mrs ');
SELECT CONCAT(' Mr ', 'and Mrs ') FROM dual;
```

Δείξτε κωδικό, όνομα μαζί για κάθε υπάλληλο

```
SELECT CONCAT(empno,ename) FROM emp;
SELECT CONCAT(empno, ' ', ename) FROM emp;
SELECT CONCAT(empno, ' Mr ', ename) FROM emp;
SELECT CONCAT('code=', empno, ' name=', ename) FROM emp;
```

Δώστε παραδείγματα των συναρτήσεων CURRENT_DATE, NOW, FLOOR, CEIL, DATEDIFF, ROUND.

```
SELECT CURRENT_DATE ;(απάντηση: '2017-12_09') (απάντηση: '2017-12_09')
```

```
SELECT NOW() ; (απάντηση: '2017-12_09 21:51:45') (απάντηση: '2017-12_09
21:51:45')
```

```
SELECT FLOOR(15.2);    (απάντηση: 15)
SELECT FLOOR(-15.2);   (απάντηση: -16)
SELECT FLOOR(15.8);    (απάντηση: 15)
SELECT FLOOR(-15.8);   (απάντηση: -16)
```

```
SELECT CEIL(15.2);       (απάντηση: 16)
SELECT CEIL(-15.2);      (απάντηση: -15)
SELECT CEIL(15.8);       (απάντηση: 16)
SELECT CEIL(-15.8);      (απάντηση: -15)
```

```
SELECT ROUND(15.2);      (απάντηση: 15)
SELECT ROUND(-15.2);   (απάντηση: -15)
SELECT ROUND(15.8);      (απάντηση: 16)
SELECT ROUND(-15.8);   (απάντηση: -16)
```

```
SELECT DATEDIFF('2017-12_09', '2017-12-25');   (απάντηση: -16)
SELECT DATEDIFF('2017-12-25', '2017-12_09');    (απάντηση: 16)
```

Δείξτε πόσο χρόνο εργάζεται κάθε υπάλληλος στην εταιρεία από την ημερομηνία πρόσληψης

```
SELECT empno, ename,
FLOOR(DATEDIFF(CURRENT_DATE, HIREDATE)/365)
FROM emp;
```

```
SELECT empno, ename,
ROUND(DATEDIFF(CURRENT_DATE, HIREDATE)/365)
FROM emp;
```

Δείξτε για τους υπάλληλους μέσο όρο μισθού, μέγιστο και ελάχιστο μισθό, άθροισμα μισθών, πόσοι έχουν μισθό και πόσοι είναι όλοι οι υπάλληλοι.

```
SELECT AVG(sal), MAX(sal), MIN(sal), SUM(sal), COUNT(sal), COUNT(*)
FROM emp;
```

Δείξτε για τους υπάλληλους μέσο όρο προμήθειας, μέγιστη και ελάχιστη

προμήθεια, άθροισμα προμηθειών, πόσοι έχουν προμήθεια και πόσοι είναι όλοι οι υπάλληλοι.

```
SELECT AVG(comm), MAX(comm), MIN(comm), SUM(comm), COUNT(comm), COUNT(*)
FROM emp;
```

Δείξτε για τους αναλυτές μέσο όρο μισθού, μέγιστο και ελάχιστο μισθό, άθροισμα μισθών, πόσοι έχουν μισθό και πόσοι είναι όλοι οι αναλυτές.

```
SELECT AVG(sal), MAX(sal), MIN(sal), SUM(sal), COUNT(sal), COUNT(*)
FROM emp
WHERE job='ANALYST';
```

4.10.2 Αναζήτηση στοιχείων που απαιτεί σύνδεση δύο πινάκων.

Η συνηθισμένη σύνταξη εντολής αναζήτησης που συνδέει δύο πίνακες είναι:

```
SELECT  όνομα_πίνακα1.όνομα_στήλης, ....
όνομα_πίνακα2.όνομα_στήλης, ....
FROM όνομα_πίνακα1 , όνομα_πίνακα2
WHERE   όνομα_πίνακα1.όνομα_στήλης =
όνομα_πίνακα2.όνομα_στήλης  κτλ.
```

```
SELECT  όνομα_πίνακα1.όνομα_στήλης, ....
όνομα_πίνακα2.όνομα_στήλης, ....
FROM όνομα_πίνακα1
JOIN όνομα_πίνακα2
ON   όνομα_πίνακα1.όνομα_στήλης = όνομα_πίνακα2.όνομα_στήλης  κτλ.
```

```
SELECT  όνομα_πίνακα1.όνομα_στήλης, ....
όνομα_πίνακα2.όνομα_στήλης, ....
FROM όνομα_πίνακα1
INNER JOIN όνομα_πίνακα2
ON   όνομα_πίνακα1.όνομα_στήλης = όνομα_πίνακα2.όνομα_στήλης  κτλ.
```

Ακολουθεί παράδειγμα χρήσης σύνδεσης και συνθήκης σε δήλωση SELECT

Η παρακάτω δήλωση βρίσκει όνομα, μισθό και έδρα των πωλητών.

```
SELECT  ENAME, JOB, SAL, LOC
 FROM  EMP,DEPT
 WHERE  JOB = 'SALESMAN'
 AND  EMP.DEPTNO = DEPT.DEPTNO;
```

ENAME	JOB	SAL	LOC
NAVATHE	SALESMAN	2000	DALLAS

Ακολουθεί επεξήγηση.

Η συνθήκη σύνδεσης (join)

EMP.DEPTNO = DEPT.DEPTNO

είναι σαν να μας εξασφαλίζει λογικά (conceptually) την "ενοποίηση" («οξυγονοκόλληση») των δύο πινάκων, όπως φαίνεται στην παρακάτω ανάλυση.(βλέπε πίνακα 4.7).

```
SELECT  *
FROM  EMP,DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO;
```

*Πίνακας 4.7 Αποτελέσματα δήλωσης SELECT * ... που περιλαμβάνει σύνδεση πινάκων EMP, DEPT.*

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
30	DATE	PROGRAMMER	15	04/05/2004	1800	200	10	10	ACCOUNTING	NEW YORK
15	ELMASRI	ANALYST	15	02/05/1995	1200	150	10	10	ACCOUNTING	NEW YORK
10	CODD	ANALYST	15	01/01/1989	3000	-	10	10	ACCOUNTING	NEW YORK
20	NAVATHE	SALESMAN	20	07/07/1977	2000	-	20	20	RESEARCH	DALLAS

```
SELECT  ENAME, JOB, SAL, LOC
FROM  EMP,DEPT
WHERE  JOB = 'SALESMAN'
AND  EMP.DEPTNO = DEPT.DEPTNO;
```

ENAME	JOB	SAL	LOC
NAVATHE	SALESMAN	2000	DALLAS

4.10.3 Σύνδεση πινάκων με χρήση στηλών με κοινό πεδίο ορισμού

Δεν είναι απαραίτητο οι πίνακες που θα συνδεθούν να έχουν στήλες με ίδιο όνομα, ίδιο τύπο δεδομένων και ίδιο μήκος, π.χ. char(5). Αρκεί να έχουν στήλες με κοινό πεδίο ορισμού. Επίσης, δεν είναι αναγκαίο οι δύο στήλες της σύνδεσης να έχουν τις ίδιες ακριβώς τιμές (values).

Για παράδειγμα, οι πίνακες:

```
CUSTOMER(AMKA, cust_name, cust_city)
SUPPLIER(AMKA, suppl_name, suppl_city)

DROP DATABASE cust_sup;
CREATE DATABASE cust_sup;
USE cust_sup;
```

```
CREATE TABLE customer(AMKA CHAR(11),
   cust_name CHAR(15),cust_city CHAR(15));
```

```
CREATE TABLE supplier(AMKA CHAR(11),
   suppl_name CHAR(15),suppl_city CHAR(15));
```

```
INSERT INTO customer VALUES('01019900470','CODD','NEW YORK');
INSERT INTO customer VALUES('01019900380','DATE','BERLIN');
```

```
INSERT INTO supplier VALUES('01019900470','CODD', 'NEW YORK');
INSERT INTO supplier VALUES('01019900570','NAVATHE','NEW YORK');
INSERT INTO supplier VALUES('01019900880','ELMASRI', 'ATHENS');
```

```
SELECT * FROM supplier;
```

AMKA	SUPPL_NAME	SUPPL_CITY
01019900470	CODD	NEW YORK
01019900570	NAVATHE	NEW YORK
01019900880	ELMASRI	ATHENS

```
SELECT * FROM customer;
```

AMKA	CUST_NAME	CUST_CITY
01019900470	CODD	NEW YORK
01019900380	DATE	BERLIN

Παρατηρήστε ότι ένας προμηθευτής μπορεί να είναι και πελάτης.

Η παρακάτω δήλωση δείχνει πελάτες (customers) και προμηθευτές (supplier) που έχουν έδρα την ίδια πόλη.

```
SELECT *
FROM customer, supplier
WHERE customer.cust_city=supplier.suppl_city;
```

```
SELECT cust_name, cust_city, suppl_name, suppl_city
FROM customer, supplier
WHERE customer.cust_city=supplier.suppl_city;
```

CUST_NAME	CUST_CITY	SUPPL_NAME	SUPPL_CITY
CODD	NEW YORK	CODD	NEW YORK
CODD	NEW YORK	NAVATHE	NEW YORK

Βλέπουμε ότι οι πίνακες μπορούν να συνδεθούν με τη συνθήκη

customer.cust_city=supplier.suppl_city

επειδή οι δύο στήλες **cust_city, suppl_city** έχουν το ίδιο πεδίο ορισμού.

Δείτε και την παρακάτω δήλωση.

```
SELECT cust_name, cust_city, suppl_name, suppl_city
FROM customer, supplier
WHERE customer.cust_city=supplier.suppl_city
AND customer.AMKA <> supplier.AMKA;
```

CUST_NAME	CUST_CITY	SUPPL_NAME	SUPPL_CITY
CODD	NEW YORK	NAVATHE	NEW YORK

4.10.4 Αποφυγή χρήσης καρτεσιανού γινομένου σε δηλώσεις SELECT

Συχνά οι αρχάριοι προγραμματιστές βάσεων δεδομένων παραλείπουν τη συνθήκη σύνδεσης (join) πινάκων όταν γράφουν δηλώσεις SELECT οι οποίες βασίζονται σε δύο ή περισσότερους πίνακες. Για παράδειγμα οι παρακάτω δηλώσεις είναι λανθασμένες.

```
SELECT *
FROM emp, dept
WHERE sal>2000;
```

```
SELECT *
FROM emp, dept
WHERE job='ANALYST';
```

Αυτό συμβαίνει επειδή η παράλειψη της σύνδεσης των πινάκων οδηγεί στον υπολογισμό του καρτεσιανού γινομένου.

Δείτε και την ακόλουθη δήλωση.

```
SELECT emp.ename, emp.sal, dept.loc
from  emp, dept;
```

ENAME	SAL	LOC
CODD	3000	NEW YORK
CODD	3000	DALLAS
CODD	3000	CHICAGO
CODD	3000	BOSTON
ELMASRI	1200	NEW YORK
ELMASRI	1200	DALLAS
ELMASRI	1200	CHICAGO
ELMASRI	1200	BOSTON
NAVATHE	2000	NEW YORK
NAVATHE	2000	DALLAS
NAVATHE	2000	CHICAGO
NAVATHE	2000	BOSTON
DATE	1800	NEW YORK

DATE	1800	DALLAS
DATE	1800	CHICAGO
DATE	1800	BOSTON

Προφανώς η δήλωση αυτή δεν μπορεί να θεωρηθεί σαν εναλλακτική της αντίστοιχης αναζήτησης που περιλαμβάνει join.

```
SELECT emp.ename, emp.sal, dept.loc
FROM  emp, dept
WHERE emp.deptno=dept.deptno;
```

Επειδή υπολογίζει το καρτεσιανό γινόμενο των δύο πινάκων γεγονός που οδηγεί στην εμφάνιση στοιχείων που δεν έχουν πραγματική σημασία.

4.10.5 Συνδέσεις στο προϊόν MySQL. Περιπτώσεις SQL JOINs

Πως βλέπουμε τα δεδομένα.

```
SELECT * FROM DEPT;
SELECT * FROM EMP;
SELECT * FROM PROJ;
SELECT * FROM ASSIGN;
```

```
mysql> SELECT * FROM DEPT;
+--------+------------+----------+
| DEPTNO | DNAME      | LOC      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
+--------+------------+----------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM EMP;
+-------+---------+------------+------------+------+---------+--------+--------+
| EMPNO | ENAME   | JOB        | HIREDATE   | MGR  | SAL     | COMM   | DEPTNO |
+-------+---------+------------+------------+------+---------+--------+--------+
|    10 | CODD    | ANALYST    | 1989-01-01 |   15 | 3000.00 |   NULL |     10 |
|    11 | CODD    | ANALYST    | 1989-01-01 |   15 | 2900.00 |   NULL |     10 |
|    12 | CODD    | PROGRAMMER | 1995-05-02 |   15 | 1200.00 | 150.00 |     10 |
|    15 | ELMASRI | ANALYST    | 1995-05-02 |   15 | 1200.00 | 150.00 |     10 |
|    20 | NAVATHE | SALESMAN   | 1977-07-07 |   20 | 2000.00 |   NULL |     20 |
|    21 | CODD    | SALESMAN   | 1977-07-07 |   20 | 2000.00 |   NULL |     10 |
|    22 | CODD    | PROGRAMMER | 1995-05-02 |   15 | 1200.00 | 150.00 |     20 |
|    23 | CODD    | SALESMAN   | 1977-07-07 |   20 | 2000.00 |   NULL |     20 |
|    30 | DATE    | PROGRAMMER | 2004-05-04 |   15 | 1800.00 | 200.00 |     10 |
|    77 | BATES   | SALESMAN   | 1987-07-07 |   20 | 2000.00 |   NULL |   NULL |
+-------+---------+------------+------------+------+---------+--------+--------+
10 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM PROJ;
+-------+----------+----------+
| projno | pname     | budget    |
+-------+----------+----------+
|    100 | PAYROLL   | 100000.00 |
|    200 | PERSONNEL | 200000.00 |
|    300 | SALES     | 150000.00 |
+-------+----------+----------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM ASSIGN;
+-------+-------+-------+
| EMPNO | PROJNO | PTIME |
+-------+-------+-------+
|    10 |    100 |    40 |
|    10 |    200 |    60 |
|    15 |    100 |   100 |
|    20 |    200 |   100 |
|    30 |    100 |   100 |
+-------+-------+-------+
5 rows in set (0.00 sec)
```

Πρώτο παράδειγμα σύνδεσης

```
SELECT EMPNO, ENAME, EMP.DEPTNO, DNAME, SAL, COMM
FROM EMP, DEPT
WHERE EMP.DEPTNO=DEPT.DEPTNO;
```

```
mysql> SELECT EMPNO, ENAME, EMP.DEPTNO, DNAME, SAL, COMM
    -> FROM EMP, DEPT
    -> WHERE EMP.DEPTNO=DEPT.DEPTNO;
+-------+---------+--------+------------+---------+--------+
| EMPNO | ENAME   | DEPTNO | DNAME      | SAL     | COMM   |
+-------+---------+--------+------------+---------+--------+
|    10 | CODD    |     10 | ACCOUNTING | 3000.00 |   NULL |
|    11 | CODD    |     10 | ACCOUNTING | 2900.00 |   NULL |
|    12 | CODD    |     10 | ACCOUNTING | 1200.00 | 150.00 |
|    15 | ELMASRI |     10 | ACCOUNTING | 1200.00 | 150.00 |
|    21 | CODD    |     10 | ACCOUNTING | 2000.00 |   NULL |
|    30 | DATE    |     10 | ACCOUNTING | 1800.00 | 200.00 |
|    20 | NAVATHE |     20 | RESEARCH   | 2000.00 |   NULL |
|    22 | CODD    |     20 | RESEARCH   | 1200.00 | 150.00 |
|    23 | CODD    |     20 | RESEARCH   | 2000.00 |   NULL |
+-------+---------+--------+------------+---------+--------+
9 rows in set (0.00 sec)
```

4.10.6 Περιπτώσεις SQL JOIN. Οι έννοιες (concept) join, left join, right join, full join με «απλά» λόγια

Θα λέμε ότι υπάρχει μια «αντιστοιχία γραμμών» (one match) ανάμεσα στους πίνακες emp, dept αν υπάρχουν τουλάχιστον δύο γραμμές των πινάκων αυτών που «ταιριάζουν» ως προς τη σύνδεση (join), δηλαδή «ταιριάζουν» όταν εστιάσουμε στις στήλες των πινάκων που συμμετέχουν στη σύνδεση. Για παράδειγμα, η γραμμή του υπαλλήλου (10, CODD , …, 10) «ταιριάζει» με τη γραμμή του τμήματος (10, ACCOUNTING, …)

Παρατηρήστε και τη συνθήκη της σύνδεσης:

```
EMP.DEPTNO=DEPT.DEPTNO
```

Υπάρχουν οι εξής περιπτώσεις:

- **JOIN**: Επιστρέφει γραμμές όταν υπάρχει τουλάχιστον μία «αντιστοιχία γραμμών» (one match) ανάμεσα στους δύο πίνακες
- **LEFT JOIN**: Επιστρέφει όλες τις γραμμές από τον αριστερό πίνακα, ακόμη και αν δεν υπάρχουν «αντίστοιχες γραμμές» (there are no matches) στο δεξιό πίνακα
- **RIGHT JOIN**: Επιστρέφει όλες τις γραμμές από το δεξιό πίνακα, ακόμα και αν δεν υπάρχουν «αντίστοιχες γραμμές» (there are no matches) στον αριστερό πίνακα
- **FULL JOIN**: RIGHT JOIN και LEFT JOIN μαζί

Σύνταξη SQL INNER JOIN

```
SELECT column_names
FROM table1
INNER JOIN table2
ON table1.column_name=table2.column_name
```

Το INNER JOIN είναι ίδιο με το JOIN.

Παράδειγμα SQL INNER JOIN

```
SELECT empno, ename, dept.deptno
FROM emp
INNER JOIN dept
ON emp.deptno=dept.deptno
ORDER BY ename;
```

```
SELECT empno, ename, dept.deptno
FROM emp
JOIN dept
ON emp.deptno=dept.deptno
ORDER BY ename;
```

```
SELECT empno, ename, dept.deptno
FROM emp
JOIN dept
ON emp.deptno=dept.deptno
ORDER BY ename;
```

```
mysql>
mysql> SELECT empno, ename, dept.deptno
    -> FROM emp
    -> JOIN dept
    -> ON emp.deptno=dept.deptno
    -> ORDER BY ename;
+-------+---------+--------+
| empno | ename   | deptno |
+-------+---------+--------+
|    21 | CODD    |     10 |
|    10 | CODD    |     10 |
|    11 | CODD    |     10 |
|    22 | CODD    |     20 |
|    12 | CODD    |     10 |
|    23 | CODD    |     20 |
|    30 | DATE    |     10 |
|    15 | ELMASRI |     10 |
|    20 | NAVATHE |     20 |
+-------+---------+--------+
9 rows in set (0.00 sec)
```

```
SELECT EMPNO, ENAME, EMP.DEPTNO, DEPT.DNAME, SAL, COMM
FROM emp
INNER JOIN dept
ON emp.deptno=dept.deptno
ORDER BY ename;
```

```
mysql> SELECT EMPNO, ENAME, EMP.DEPTNO, DEPT.DNAME, SAL, COMM
    -> FROM emp
    -> INNER JOIN dept
    -> ON emp.deptno=dept.deptno
    -> ORDER BY ename;
+-------+---------+--------+------------+---------+---------+
| EMPNO | ENAME   | DEPTNO | DNAME      | SAL     | COMM    |
+-------+---------+--------+------------+---------+---------+
|    11 | CODD    |     10 | ACCOUNTING | 2900.00 |    NULL |
|    21 | CODD    |     10 | ACCOUNTING | 2000.00 |    NULL |
|    22 | CODD    |     20 | RESEARCH   | 1200.00 |  150.00 |
|    12 | CODD    |     10 | ACCOUNTING | 1200.00 |  150.00 |
|    23 | CODD    |     20 | RESEARCH   | 2000.00 |    NULL |
|    10 | CODD    |     10 | ACCOUNTING | 3000.00 |    NULL |
|    30 | DATE    |     10 | ACCOUNTING | 1800.00 |  200.00 |
|    15 | ELMASRI |     10 | ACCOUNTING | 1200.00 |  150.00 |
|    20 | NAVATHE |     20 | RESEARCH   | 2000.00 |    NULL |
+-------+---------+--------+------------+---------+---------+
9 rows in set (0.00 sec)
```

Σύνταξη SQL LEFT JOIN

```
SELECT column_names
FROM table1
LEFT JOIN table2
ON table1.column_name=table2.column_name
```

Σε κάποια προϊόντα το LEFT JOIN ονομάζεται LEFT OUTER JOIN.

Παράδειγμα SQL LEFT JOIN

```
SELECT empno, ename, dept.deptno
FROM emp
LEFT JOIN dept
ON emp.deptno=dept.deptno
ORDER BY ename;
```

```
mysql> SELECT empno, ename, dept.deptno
    -> FROM emp
    -> LEFT JOIN dept
    -> ON emp.deptno=dept.deptno
    -> ORDER BY ename;
+-------+---------+--------+
| empno | ename   | deptno |
+-------+---------+--------+
|    77 | BATES   |   NULL |
|    23 | CODD    |     20 |
|    22 | CODD    |     20 |
|    21 | CODD    |     10 |
|    12 | CODD    |     10 |
|    11 | CODD    |     10 |
|    10 | CODD    |     10 |
|    30 | DATE    |     10 |
|    15 | ELMASRI |     10 |
|    20 | NAVATHE |     20 |
+-------+---------+--------+
10 rows in set (0.00 sec)
```

Σύνταξη SQL RIGHT JOIN

```
SELECT column_names
FROM table1
RIGHT JOIN table2
ON table1.column_name=table2.column_name
```

Σε κάποια προϊόντα το RIGHT JOIN ονομάζεται RIGHT OUTER JOIN.

Παράδειγμα SQL RIGHT JOIN

```
SELECT empno, ename, dept.deptno
FROM emp
RIGHT JOIN dept
ON emp.deptno=dept.deptno
ORDER BY ename;
```

```
mysql> SELECT empno, ename, dept.deptno
    -> FROM emp
    -> RIGHT JOIN dept
    -> ON emp.deptno=dept.deptno
    -> ORDER BY ename;
+-------+---------+--------+
| empno | ename   | deptno |
+-------+---------+--------+
|  NULL | NULL    |     30 |
|  NULL | NULL    |     40 |
|    12 | CODD    |     10 |
|    21 | CODD    |     10 |
|    22 | CODD    |     20 |
|    23 | CODD    |     20 |
|    10 | CODD    |     10 |
|    11 | CODD    |     10 |
|    30 | DATE    |     10 |
|    15 | ELMASRI |     10 |
|    20 | NAVATHE |     20 |
+-------+---------+--------+
11 rows in set (0.00 sec)
```

Σύνταξη SQL FULL JOIN

```
SELECT column_names
FROM table1
FULL JOIN table2
ON table1.column_name=table2.column_name;
```

Δεν υποστηρίζεται στο προϊόν mySQL.

Στο προϊόν mySQL αντί της δήλωσης full join μπορούμε να χρησιμοποιούμε μία από τις παρακάτω δηλώσεις:

```
SELECT empno, ename, dept.deptno
FROM emp
LEFT JOIN dept
ON emp.deptno=dept.deptno
UNION
SELECT empno, ename, dept.deptno
FROM emp
RIGHT JOIN dept
ON emp.deptno=dept.deptno
ORDER BY ename;
```

```
mysql> SELECT empno, ename, dept.deptno
    -> FROM emp
    -> LEFT JOIN dept
    -> ON emp.deptno=dept.deptno
    -> UNION
    -> SELECT empno, ename, dept.deptno
    -> FROM emp
    -> RIGHT JOIN dept
    -> ON emp.deptno=dept.deptno
    -> ORDER BY ename;
+-------+---------+--------+
| empno | ename   | deptno |
+-------+---------+--------+
|  NULL | NULL    |     30 |
|  NULL | NULL    |     40 |
|    77 | BATES   |   NULL |
|    21 | CODD    |     10 |
|    10 | CODD    |     10 |
|    22 | CODD    |     20 |
|    11 | CODD    |     10 |
|    23 | CODD    |     20 |
|    12 | CODD    |     10 |
|    30 | DATE    |     10 |
|    15 | ELMASRI |     10 |
|    20 | NAVATHE |     20 |
+-------+---------+--------+
12 rows in set (0.04 sec)
```

```
SELECT empno, ename, dept.deptno
FROM emp
LEFT JOIN dept
ON emp.deptno=dept.deptno
UNION ALL
SELECT empno, ename, dept.deptno
FROM emp
```

```
RIGHT JOIN dept
ON emp.deptno=dept.deptno
WHERE EMP.DEPTNO IS NULL
ORDER BY ename;
```

```
mysql> SELECT empno, ename, dept.deptno
    -> FROM emp
    -> LEFT JOIN dept
    -> ON emp.deptno=dept.deptno
    -> UNION ALL
    -> SELECT empno, ename, dept.deptno
    -> FROM emp
    -> RIGHT JOIN dept
    -> ON emp.deptno=dept.deptno
    -> WHERE EMP.DEPTNO IS NULL
    -> ORDER BY ename;
+-------+---------+--------+
| empno | ename   | deptno |
+-------+---------+--------+
|  NULL | NULL    |     30 |
|  NULL | NULL    |     40 |
|    77 | BATES   |   NULL |
|    21 | CODD    |     10 |
|    10 | CODD    |     10 |
|    22 | CODD    |     20 |
|    11 | CODD    |     10 |
|    23 | CODD    |     20 |
|    12 | CODD    |     10 |
|    30 | DATE    |     10 |
|    15 | ELMASRI |     10 |
|    20 | NAVATHE |     20 |
+-------+---------+--------+
12 rows in set (0.00 sec)
```

Προσοχή!

```
SELECT empno, ename, dept.deptno
FROM emp
LEFT JOIN dept
ON emp.deptno=dept.deptno
UNION ALL
SELECT empno, ename, dept.deptno
FROM emp
RIGHT JOIN dept
ON emp.deptno=dept.deptno
ORDER BY ename;
```

```
mysql> SELECT empno, ename, dept.deptno
    -> FROM emp
    -> LEFT JOIN dept
    -> ON emp.deptno=dept.deptno
    -> UNION ALL
    -> SELECT empno, ename, dept.deptno
    -> FROM emp
    -> RIGHT JOIN dept
    -> ON emp.deptno=dept.deptno
    -> ORDER BY ename;
+-------+---------+--------+
| empno | ename   | deptno |
+-------+---------+--------+
|  NULL | NULL    |     30 |
|  NULL | NULL    |     40 |
|    77 | BATES   |   NULL |
|    10 | CODD    |     10 |
|    21 | CODD    |     10 |
|    11 | CODD    |     10 |
|    22 | CODD    |     20 |
|    10 | CODD    |     10 |
|    22 | CODD    |     20 |
|    12 | CODD    |     10 |
|    23 | CODD    |     20 |
|    11 | CODD    |     10 |
|    23 | CODD    |     20 |
|    12 | CODD    |     10 |
|    21 | CODD    |     10 |
|    30 | DATE    |     10 |
|    30 | DATE    |     10 |
|    15 | ELMASRI |     10 |
|    15 | ELMASRI |     10 |
|    20 | NAVATHE |     20 |
|    20 | NAVATHE |     20 |
+-------+---------+--------+
21 rows in set (0.00 sec)
```

4.10.7 Σύνδεση πολλών πινάκων

Στη συνέχεια παραθέτουμε παράδειγμα σύνδεσης πολλών πινάκων με πολλούς τρόπους σύνταξης.

```
SELECT ASSIGN.EMPNO, ENAME, DNAME, ASSIGN.PROJNO, PTIME
FROM DEPT, EMP, ASSIGN, PROJ
WHERE DEPT.DEPTNO=EMP.DEPTNO
AND EMP.EMPNO=ASSIGN.EMPNO
AND ASSIGN.PROJNO=PROJ.PROJNO;
```

```
SELECT assign.empno, ename, dname, assign.projno, ptime
FROM dept
INNER JOIN emp ON dept.deptno=emp.deptno
JOIN assign ON emp.empno=assign.empno
JOIN proj ON assign.projno=proj.projno;
```

```
SELECT ASSIGN.EMPNO, ENAME, DNAME, ASSIGN.PROJNO, PTIME
FROM DEPT INNER JOIN EMP
ON DEPT.DEPTNO=EMP.DEPTNO
JOIN ASSIGN ON EMP.EMPNO=ASSIGN.EMPNO
JOIN PROJ ON ASSIGN.PROJNO=PROJ.PROJNO;
```

```
SELECT c.EMPNO, ENAME, DNAME, c.PROJNO, PTIME
FROM EMP a
INNER JOIN DEPT b ON a.DEPTNO=b.DEPTNO
INNER JOIN ASSIGN c ON a.EMPNO=c.EMPNO
INNER JOIN PROJ d ON c.PROJNO=d.PROJNO;
```

Ακολουθεί παράδειγμα σύνδεσης πίνακα με το εαυτό του.

```
SELECT WORKER.EMPNO "emp", MANAGER.EMPNO "mgr",
  WORKER.SAL "emp_sal", MANAGER.SAL "mgr_sal"
FROM EMP WORKER, EMP MANAGER
WHERE WORKER.MGR=MANAGER.EMPNO
AND WORKER.SAL > MANAGER.SAL;
```

```
SELECT assign.empno, ename, dname, assign.projno, ptime
FROM dept
INNER JOIN emp ON dept.deptno=emp.deptno
JOIN assign ON emp.empno=assign.empno
JOIN proj ON assign.projno=proj.projno;
```

Προσοχή στη λανθασμένη σύνδεση στο επόμενο παράδειγμα.

```
SELECT assign.empno, ename, dname, assign.projno, ptime
FROM dept
INNER JOIN emp ON dept.deptno=emp.empno
JOIN assign ON emp.empno=assign.empno
JOIN proj ON assign.projno=proj.projno;
```

```
mysql>
mysql> SELECT ASSIGN.EMPNO, ENAME, DNAME, ASSIGN.PROJNO, PTIME
    ->       FROM DEPT
    ->       INNER JOIN EMP ON DEPT.DEPTNO=EMP.EMPNO
    ->       JOIN ASSIGN ON EMP.EMPNO=ASSIGN.EMPNO
    ->       JOIN PROJ ON ASSIGN.PROJNO=PROJ.PROJNO;
+-------+---------+------------+--------+-------+
| EMPNO | ENAME   | DNAME      | PROJNO | PTIME |
+-------+---------+------------+--------+-------+
|    10 | CODD    | ACCOUNTING |    100 |    40 |
|    10 | CODD    | ACCOUNTING |    200 |    60 |
|    20 | NAVATHE | RESEARCH   |    200 |   100 |
|    30 | DATE    | SALES      |    100 |   100 |
+-------+---------+------------+--------+-------+
4 rows in set (0.00 sec)
```

```
SELECT ASSIGN.EMPNO, ENAME, DNAME, ASSIGN.PROJNO, PTIME
FROM DEPT INNER JOIN EMP
ON DEPT.DEPTNO=EMP.DEPTNO
JOIN ASSIGN ON EMP.EMPNO=ASSIGN.EMPNO
JOIN PROJ ON ASSIGN.PROJNO=PROJ.PROJNO;
```

```
mysql> SELECT ASSIGN.EMPNO, ENAME, DNAME, ASSIGN.PROJNO, PTIME
    -> FROM DEPT INNER JOIN EMP
    -> ON DEPT.DEPTNO=EMP.EMPNO
    -> JOIN ASSIGN ON EMP.EMPNO=ASSIGN.EMPNO
    -> JOIN PROJ ON ASSIGN.PROJNO=PROJ.PROJNO;
+-------+---------+------------+--------+-------+
| EMPNO | ENAME   | DNAME      | PROJNO | PTIME |
+-------+---------+------------+--------+-------+
|    10 | CODD    | ACCOUNTING |    100 |    40 |
|    10 | CODD    | ACCOUNTING |    200 |    60 |
|    20 | NAVATHE | RESEARCH   |    200 |   100 |
|    30 | DATE    | SALES      |    100 |   100 |
+-------+---------+------------+--------+-------+
4 rows in set (0.47 sec)
```

```
SELECT c.EMPNO, ENAME, DNAME, c.PROJNO, PTIME
FROM EMP a
INNER JOIN DEPT b ON a.DEPTNO=b.DEPTNO
INNER JOIN ASSIGN c ON a.EMPNO=c.EMPNO
INNER JOIN PROJ d ON c.PROJNO=d.PROJNO;
```

```
SELECT ASSIGN.EMPNO, ENAME, DNAME, ASSIGN.PROJNO, PTIME
FROM DEPT, EMP, ASSIGN, PROJ
WHERE DEPT.DEPTNO=EMP.DEPTNO
AND EMP.EMPNO=ASSIGN.EMPNO
AND ASSIGN.PROJNO=PROJ.PROJNO;
```

4.10.8 Συνδέσεις δύο πινάκων σε περισσότερες στήλες

```
DROP DATABASE IF EXISTS join_2_tables_3_columns;
CREATE DATABASE join_2_tables_3_columns;
USE join_2_tables_3_columns;
```

```
CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL,
 DNAME VARCHAR(14), LOC VARCHAR(14),
PRIMARY KEY(DEPTNO));
```

```
CREATE TABLE EMP(EMPNO INT(4) NOT NULL,
  ENAME VARCHAR(10), JOB VARCHAR(25),
  HIREDATE DATE, MGR INT(4),  SAL FLOAT(7,2), COMM FLOAT(7,2),
  DEPTNO INT(2),
  PRIMARY KEY(EMPNO),
  FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO));
```

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (30, 'SALES', 'CHICAGO');
```

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (40, 'OPERATIONS', 'BOSTON');
INSERT INTO EMP
  VALUES (100, 'CODD', 'ANALYST', '1989/01/01', 15, 3000, NULL, 10);
INSERT INTO EMP
  VALUES (150, 'ELMASRI', 'ANALYST', '1995/05/02', 15, 1200, 150, 10);
INSERT INTO EMP
  VALUES (200, 'NAVATHE', 'SALESMAN', '1977/07/07', 20, 2000, NULL, 20);
INSERT INTO EMP
  VALUES (300, 'DATE', 'PROGRAMMER', '2004/05/04', 15, 1800, 200, 10);
INSERT INTO EMP
  VALUES (110, 'CODD', 'ANALYST', '1989/01/01', 15, 2900, NULL, 10);
INSERT INTO EMP
  VALUES (120, 'CODD', 'PROGRAMMER', '1995/05/02', 15, 1200, 150, 10);
INSERT INTO EMP
  VALUES (210, 'CODD', 'SALESMAN', '1977/07/07', 20, 2000, NULL, 10);
INSERT INTO EMP
  VALUES (220, 'CODD', 'PROGRAMMER', '1995/05/02', 15, 1200, 150, 20);
INSERT INTO EMP
  VALUES (230, 'CODD', 'SALESMAN', '1977/07/07', 20, 2000, NULL, 20);
INSERT INTO EMP
  VALUES (770, 'BATES', 'SALESMAN', '1987/07/07', 20, 2000, NULL, NULL);
```

```
SELECT * FROM emp;
SELECT * FROM dept;
```

```
SELECT a.empno, a.ename, a.job, a.deptno, b.empno, b.ename, b.job, b.deptno FROM
EMP a, EMP b
WHERE a.job=b.job
AND a.deptno=b.deptno;
```

4.10.9 Θέμα προς επίλυση. Διαχείριση βάσης δεδομένων παραγγελιών ανταλλακτικών

Έστω ότι σας αναθέτουν τη σχεδίαση και την υλοποίηση μίας βάσης δεδομένων παραγγελιών ανταλλακτικών. Ακολουθούν κάποιοι περιορισμοί:

1) Μία παραγγελία δίδεται από έναν πελάτη και κάθε πελάτης μπορεί να δώσει πολλές παραγγελίες.
2) Μία παραγγελία μπορεί να περιλαμβάνει πολλά ανταλλακτικά και ένα ανταλλακτικό μπορεί να περιλαμβάνεται σε πολλές παραγγελίες.

Σαν άσκηση καταγράψτε όλους τους περιορισμούς.

Η διαδικασία της ανάλυσης δεδομένων οδήγησε στις παρακάτω στήλες:

customer_id int(5) κωδικός πελάτη,

customer_name varchar(14) όνομα πελάτη,

loc varchar(14) έδρα πελάτη,

credit_limit float(7,2) πιστωτικό όριο πελάτη,

order_id int(6) κωδικός παραγγελίας,

order_date DATE ημερομηνία παραγγελίας,

total float(7,2) συνολικό ποσό παραγγελίας,

part_id int(5) κωδικός ανταλλακτικού,

part_name VARCHAR(14) όνομα,

price float(6,2) τιμή,

quantity int(4) ποσότητα ανταλλακτικού,

ptotal float(7,2) αξία ανταλλακτικού ανά παραγγελία.

Αν θέλουμε να περιγράψουμε τις στήλες σε μορφή εγγύτερη στο προϊόν Oracle θα έχουμε τα εξής:

customer_id NUMBER(5) κωδικός πελάτη,

customer_name varchar2(14) όνομα πελάτη,

loc varchar2(14) έδρα πελάτη,

credit_limit NUMBER(7,2) πιστωτικό όριο πελάτη,

order_id NUMBER(6) κωδικός παραγγελίας,

order_date DATE ημερομηνία παραγγελίας,

total NUMBER(7,2) συνολικό ποσό παραγγελίας,

part_id NUMBER(5) κωδικός ανταλλακτικού,

part_name VARCHAR2(14) όνομα,

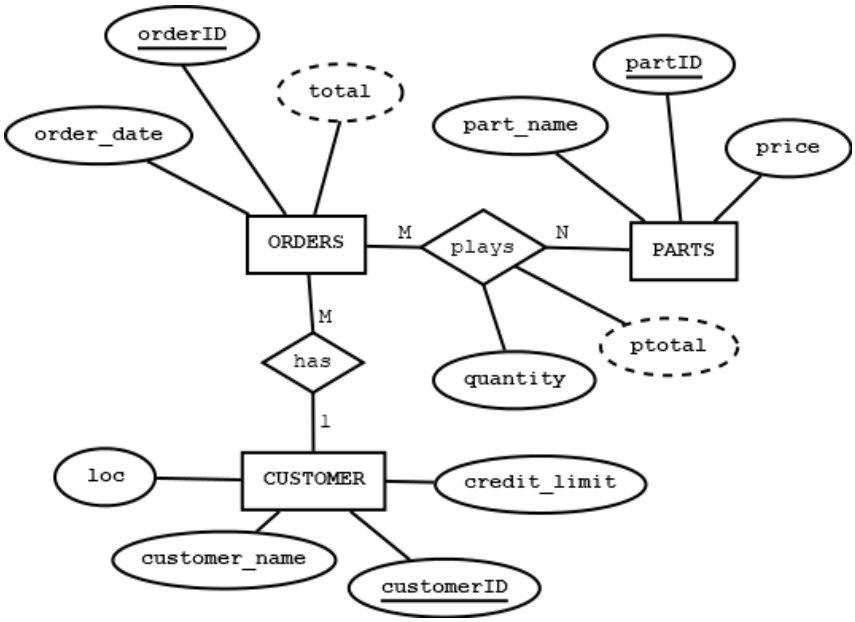
price NUMBER(6,2) τιμή,

quantity NUMBER(4) ποσότητα ανταλλακτικού,

ptotal NUMBER(7,2) αξία ανταλλακτικού ανά παραγγελία.

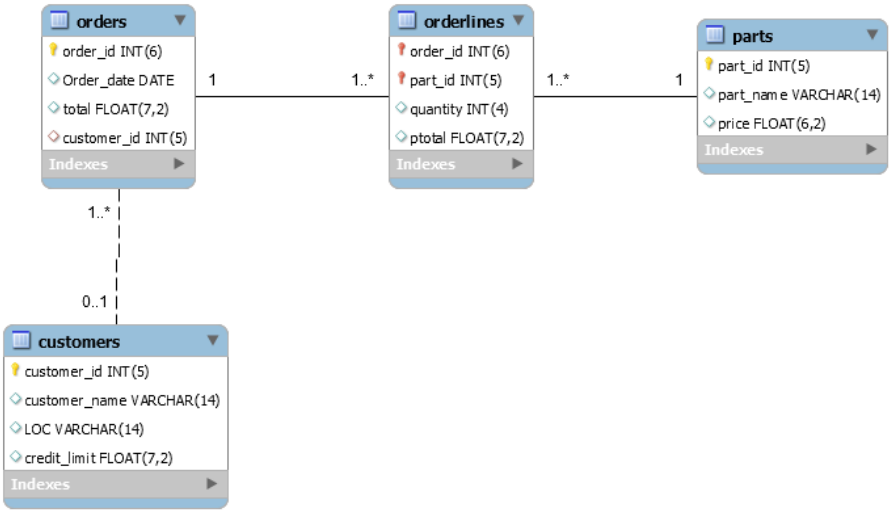
Σχεδιάστε το Μοντέλο Οντοτήτων Συσχετίσεων.

Ακολουθεί ενδεικτική λύση στην Εικόνα 4.12.



Εικόνα 4.12 Μοντέλο οντοτήτων συσχετίσεων (ΜΟΣ) βάσης δεδομένων παραγγελιών ανταλλακτικών.

Παραθέτουμε ενδεικτικό μοντέλο σχεδιασμένο σε MySQL workbench με συμβολισμό UML (εικόνα 4.13).



Εικόνα 4.13 Μοντέλο σε MySQL workbench με συμβολισμό UML

Στη συνέχεια θα πρέπει να ορίσουμε τους πίνακες αυτούς.

Ακολουθεί ενδεικτική λύση στο προϊόν της Oracle.

```
CREATE TABLE customers(customer_id NUMBER(5) NOT NULL,
 customer_name VARCHAR2(14), LOC VARCHAR2(14),
 credit_limit NUMBER(7,2),
 PRIMARY KEY(customer_id));
CREATE TABLE orders(order_id NUMBER(6) NOT NULL,
   Order_date DATE, total NUMBER(7,2), customer_id NUMBER(5),
  PRIMARY KEY(order_id),
FOREIGN KEY(customer_id) REFERENCES customers(customer_id));
CREATE TABLE parts(part_id NUMBER(5) NOT NULL,
 part_name VARCHAR2(14), price NUMBER(6,2),
```

```
 PRIMARY KEY(part_id));
CREATE TABLE orderlines(order_id NUMBER(6) NOT NULL,
   part_id NUMBER(5), quantity NUMBER(4), ptotal NUMBER(7,2),
  PRIMARY KEY(order_id, part_id),
  FOREIGN KEY(order_id) REFERENCES orders(order_id),
  FOREIGN KEY(part_id) REFERENCES parts(part_id));
```

Εισάγετε τα δεδομένα των πινάκων.

Παραθέτουμε τους πίνακες της βάσης δεδομένων με δείγμα δεδομένων (sample of data) και αντίστοιχες δηλώσεις.

```
SELECT * FROM customers;
```

Customer_id	Customer_name	Loc	Credit_limit
100	CODD	NEW YORK	10000
200	DATE	DALLAS	20000
300	CHEN	CHICAGO	15000
400	ELMASRI	BOSTON	35000

```
INSERT INTO customers VALUES (100, 'CODD', 'NEW YORK', 10000);
INSERT INTO customers VALUES (200, 'DATE', 'DALLAS', 20000);
INSERT INTO customers VALUES (300, 'CHEN', 'CHICAGO', 15000);
INSERT INTO customers VALUES (400, 'ELMASRI', 'BOSTON', 35000);
```

```
SELECT * FROM orders;
```

Order_id	Order_date	total	Customer_id
10	1/1/2017	3000	100
11	1/1/2017	2900	100
12	2/5/2017	1200	100
15	2/5/2017	1200	100
20	7/7/2017	2000	200
21	7/7/2017	2000	100
22	7/7/2017	1200	200
23	7/7/2017	2000	200
30	4/8/2017	1800	100
77	7/12/2017	2000	NULL

Είναι σωστό το δείγμα δεδομένων που βλέπουμε στον πίνακα orders; Σχολιάστε.

Προσοχή! Η παραγγελία 77 θα ήταν απαράδεκτη αν η στήλη customer_id είχε οριστεί ως NOT NULL επειδή η στήλη customer_id δεν έχει τιμή. Επίσης, αν προσπαθήσουμε να θέσουμε στη στήλη customer_id τιμή 500 (δηλαδή κωδικό πελάτη που δεν υπάρχει) τότε θα έχουμε πρόβλημα («αστοχία») επειδή ορίσαμε στοςπίνακες PRIMARY KEY, FOREIGN KEY.

```
INSERT INTO ORDERS
  VALUES (10, '01/01/2017', 3000, 100);
INSERT INTO ORDERS
  VALUES (11, '01/01/2017', 2900, 100);
INSERT INTO ORDERS
  VALUES (12, '02/05/2017', 1200, 100);
INSERT INTO ORDERS
  VALUES (15, '02/05/2017', 1200, 100);
INSERT INTO ORDERS
  VALUES (20, '07/07/2017', 2000, 200);
INSERT INTO ORDERS
  VALUES (21, '07/07/2017', 2000, 100);
INSERT INTO ORDERS
  VALUES (22, '07/07/2017', 1200, 200);
INSERT INTO ORDERS
  VALUES (23, '07/07/2017', 2000, 200);
INSERT INTO ORDERS
  VALUES (30, '04/08/2017', 1800, 100);
INSERT INTO ORDERS
  VALUES (77, '07/12/2017', 2000, NULL);
```

SELECT * FROM parts;

part_id	Part_name	Price
12345	BOLT	1
56789	SCREWDRIVER	4

```
INSERT INTO parts VALUES (12345, 'BOLT',1);
INSERT INTO parts VALUES (56789, 'SCREWDRIVER',4);
```

SELECT * FROM orderlines;

Order_id	Part_id	Quantity	Ptotal
10	12345	1800	1800
10	56789	300	1200

```
INSERT INTO orderlines VALUES (10,12345,1800,1800);
INSERT INTO orderlines VALUES (10,56789,300,1200);
```

Δημιουργήστε τη βάση δεδομένων και τους πίνακες και εισάγετε στοιχεία στο προϊόν MySQL

Ακολουθεί η δημιουργία της βάσης με χρήση mySQL και στη συνέχεια οι πίνακες δημιουργούνται με κύρια και ξένα κλειδιά.

 # Δημιουργία της βάσης

```
DROP DATABASE IF EXISTS orders_customers;
CREATE DATABASE orders_customers;
```

 # Χρήση βάσης

```
USE orders_customers;
```

 # Δημιουργία πινάκων

```
CREATE TABLE customers(customer_id INT(5) NOT NULL,
 customer_name VARCHAR(14), LOC VARCHAR(14),
 credit_limit float(7,2),
 PRIMARY KEY(customer_id));
CREATE TABLE orders(order_id INT(6) NOT NULL,
   Order_date DATE, total float(7,2), customer_id INT(5),
  PRIMARY KEY(order_id),
  FOREIGN KEY(customer_id) REFERENCES customers(customer_id));
CREATE TABLE parts(part_id int(5) NOT NULL,
 part_name VARCHAR(14), price float(6,2),
 PRIMARY KEY(part_id));
CREATE TABLE orderlines(order_id int(6) NOT NULL,
   part_id int(5) NOT NULL, quantity int(4), ptotal float(7,2),
  PRIMARY KEY(order_id, part_id),
  FOREIGN KEY(order_id) REFERENCES orders(order_id),
  FOREIGN KEY(part_id) REFERENCES parts(part_id));
```

```
SHOW TABLES;
```

 # Αν θέλουμε να δούμε τη δομή των πινάκων γράφουμε εντολή describe.

```
Describe customers;
Describe orders;
Describe parts;
Describe orderlines;
```

 # Εισαγωγή στοιχείων στους πίνακες

```
INSERT INTO customers VALUES (100, 'CODD', 'NEW YORK', 10000);
INSERT INTO customers VALUES (200, 'DATE', 'DALLAS', 20000);
INSERT INTO customers VALUES (300, 'CHEN', 'CHICAGO', 15000);
INSERT INTO customers VALUES (400, 'ELMASRI', 'BOSTON', 35000);
INSERT INTO ORDERS
  VALUES (10, '2017/01/01', 3000, 100);
INSERT INTO ORDERS
  VALUES (15, '2017/05/02', 1200, 100);
INSERT INTO ORDERS
  VALUES (20, '2017/07/07', 2000, 200);
INSERT INTO ORDERS
  VALUES (30, '2017/05/04', 1800, 100);
INSERT INTO ORDERS
  VALUES (11, '2017/01/01', 2900, 100);
INSERT INTO ORDERS
  VALUES (12, '2017/05/02', 1200, 100);
INSERT INTO ORDERS
  VALUES (21, '2017/07/07', 2000, 100);
INSERT INTO ORDERS
  VALUES (22, '2017/05/02', 1200, 200);
INSERT INTO ORDERS
  VALUES (23, '2017/07/07', 2000, 200);
INSERT INTO ORDERS
```

```
  VALUES (77, '2017/07/07', 2000, NULL);
INSERT INTO parts VALUES (12345, 'BOLT',1);
INSERT INTO parts VALUES (56789, 'SCREWDRIVER',4);
INSERT INTO orderlines VALUES (10,12345,1800,1800);
INSERT INTO orderlines VALUES (10,56789,300,1200);
```

Πως βλέπουμε τα δεδομένα.

```
SELECT * FROM customers;
SELECT * FROM orders;
SELECT * FROM parts;
SELECT * FROM orderlines;
```

Γράψτε τη σύνταξη δήλωσης SELECT η οποία περιλαμβάνει χρήση INNER JOIN (simple join).

Παραθέτουμε τη σύνταξη σε MySQL και Oracle.

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```

Δείξτε κωδικό πελάτη (customer_id), όνομα πελάτη (customer_name), κωδικό παραγγελίας και ημερομηνία (order_date) παραγγελίας

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers
INNER JOIN orders
ON customers.customer_id = orders.customer_id;
```

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers
INNER JOIN orders
ON customers.customer_id = orders.customer_id;
```

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers, orders
WHERE customers.customer_id = orders.customer_id;
```

Γράψτε τη σύνταξη δήλωσης SELECT η οποία περιλαμβάνει χρήση LEFT OUTER JOIN

Παραθέτουμε τη σύνταξη σε MySQL και Oracle

```
SELECT columns
FROM table1
LEFT [OUTER] JOIN table2
ON table1.column = table2.column;
```

Δείξτε κωδικό πελάτη (customer_id), όνομα πελάτη (customer_name), κωδικό παραγγελίας και ημερομηνία (order_date) παραγγελίας. Στα αποτελέσματα να φαίνονται και πελάτες που δεν έδωσαν παραγγελία.

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers
LEFT OUTER JOIN orders
ON customers.customer_id = orders.customer_id;
```

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers
LEFT JOIN orders
ON customers.customer_id = orders.customer_id;
```

Μόνο στο προϊόν της Oracle ισχύει:

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers, orders
WHERE customers.customer_id = orders.customer_id(+);
```

Γράψτε τη σύνταξη δήλωσης SELECT η οποία περιλαμβάνει χρήση RIGHT OUTER JOIN

Παραθέτουμε τη σύνταξη σε MySQL και Oracle

```
SELECT columns
FROM table1
RIGHT [OUTER] JOIN table2
ON table1.column = table2.column;
```

Δείξτε κωδικό πελάτη (customer_id), όνομα πελάτη (customer_name), κωδικό παραγγελίας και ημερομηνία (order_date) παραγγελίας. Στα αποτελέσματα να φαίνονται και παραγγελίες στις οποίες δεν υπάρχει συμπληρωμένος ο πελάτης.

```
SELECT orders.order_id, orders.order_date, customers.customer_name,
orders.order_id
FROM customers
RIGHT OUTER JOIN orders
ON customers.customer_id = orders.customer_id;
SELECT orders.order_id, orders.order_date, customers.customer_name,
orders.order_id
FROM customers
RIGHT JOIN orders
ON customers.customer_id = orders.customer_id;
```

Μόνο στο προϊόν της Oracle ισχύει:

```
SELECT orders.order_id, orders.order_date, customers.customer_name
FROM customers, orders
WHERE customers.customer_id(+) = orders.customer_id;
```

Γράψτε τη σύνταξη δήλωσης SELECT η οποία περιλαμβάνει χρήση FULL OUTER JOIN

Παραθέτουμε τη σύνταξη σε Oracle

```
SELECT columns
FROM table1
```

```
FULL [OUTER] JOIN table2
ON table1.column = table2.column;
```

```
SELECT customers.customer_id, customers.customer_name, orders.order_date
FROM customers
FULL OUTER JOIN orders
ON customers.customer_id = orders.customer_id;
```

Σε ORACLE και MySQL:

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers
LEFT JOIN orders
ON customers.customer_id = orders.customer_id
UNION
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers
RIGHT JOIN orders
ON customers.customer_id = orders.customer_id;
```

Γράψτε σε MySQL και Oracle SQL σύνδεση πίνακα με τον εαυτό του (SELF JOIN)

```
SELECT a.column_name, b.column_name...
FROM table1 a, table1 b
WHERE a.common_field = b.common_field;
```

```
SELECT  a.customer_ID, a.customer_name, a.credit_limit,
  b.customer_ID, b.customer_name, b.credit_limit
 FROM CUSTOMERS a, CUSTOMERS b
 WHERE a.credit_limit = b.credit_limit;
```

```
SELECT  a.customer_ID, a.customer_name, a.credit_limit,
  b.customer_ID, b.customer_name, b.credit_limit
 FROM CUSTOMERS a, CUSTOMERS b
 WHERE a.credit_limit < b.credit_limit;
```

Δείξτε για κάθε υπάλληλο τον κωδικό του (customer_ID) και το πιστωτικό του όριο (credit_limit) μαζί με τους κωδικούς και το πιστωτικό όριο των υπαλλήλων που έχουν μεγαλύτερο πιστωτικό όριο από αυτόν.

```
SELECT  a.customer_ID, a.customer_name, a.credit_limit,
  b.customer_ID, b.customer_name, b.credit_limit
 FROM CUSTOMERS a, CUSTOMERS b
 WHERE a.credit_limit < b.credit_limit
ORDER BY a.customer_ID;
```

Γράψτε δήλωση SELECT που περιλαμβάνει INNER JOIN με διάφορους τρόπους.

Να η σύνταξη με τρεις τρόπους.

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```

```
SELECT columns
FROM table1
JOIN table2
ON table1.column = table2.column;
```

```
SELECT columns
FROM table1, table2
WHERE table1.column = table2.column;
```

Δείξτε κωδικό πελάτη (customer_id), όνομα πελάτη (customer_name), κωδικό παραγγελίας και ημερομηνία (order_date) παραγγελίας

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers
INNER JOIN orders
ON customers.customer_id = orders.customer_id;
```

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers
JOIN orders
ON customers.customer_id = orders.customer_id;
```

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers, orders
WHERE customers.customer_id = orders.customer_id;
```

Γράψτε δήλωση SELECT που περιλαμβάνει LEFT OUTER JOIN/LEFT OUTER JOIN

Ακολουθεί η σύνταξη της δήλωσης.

```
SELECT columns
FROM table1
LEFT [OUTER] JOIN table2
ON table1.column = table2.column;
```

Δείξτε κωδικό πελάτη (customer_id), όνομα πελάτη (customer_name), κωδικό παραγγελίας και ημερομηνία (order_date) παραγγελίας. Στα αποτελέσματα να φαίνονται και πελάτες που δεν έδωσαν παραγγελία.

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers
```

```
LEFT OUTER JOIN orders
ON customers.customer_id = orders.customer_id;
```

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers
LEFT JOIN orders
ON customers.customer_id = orders.customer_id;
```

Στην περίπτωση του προϊόντος της Oracle επιπλέον ισχύει:

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers, orders
WHERE customers.customer_id = orders.customer_id(+);
```

Γράψτε δήλωση SELECT που περιλαμβάνει RIGHT JOIN/RIGHT OUTER JOIN

Ακολουθεί η σύνταξη της δήλωσης.

```
SELECT columns
FROM table1
RIGHT [OUTER] JOIN table2
ON table1.column = table2.column;
```

Δείξτε κωδικό πελάτη (customer_id), όνομα πελάτη (customer_name), κωδικό παραγγελίας και ημερομηνία (order_date) παραγγελίας. Στα αποτελέσματα να φαίνονται και παραγγελίες στις οποιες δεν υπάρχει συμπληρωμένος ο πελάτης.

```
SELECT orders.order_id, orders.order_date, customers.customer_name,
orders.order_id
FROM customers
RIGHT OUTER JOIN orders
ON customers.customer_id = orders.customer_id;
```

```
SELECT orders.order_id, orders.order_date, customers.customer_name,
orders.order_id
FROM customers
RIGHT JOIN orders
ON customers.customer_id = orders.customer_id;
```

Στην περίπτωση του προϊόντος της Oracle επιπλέον ισχύει:

```
SELECT orders.order_id, orders.order_date, customers.customer_name
FROM customers, orders
WHERE customers.customer_id(+) = orders.customer_id;
```

Επίσης, στην περίπτωση του προϊόντος της Oracle επιπλέον έχουμε και FULL OUTER JOIN:

```
SELECT columns
FROM table1
FULL [OUTER] JOIN table2
ON table1.column = table2.column;
```

```
SELECT customers.customer_id, customers.customer_name, orders.order_date
FROM customers
FULL OUTER JOIN orders
ON customers.customer_id = orders.customer_id;
```

Να πως υλοποιείται FULL OUTER JOIN στο προϊόν MySQL

```
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers
LEFT JOIN orders
ON customers.customer_id = orders.customer_id
UNION
SELECT customers.customer_id, customers.customer_name, orders.order_date,
orders.order_id
FROM customers
RIGHT JOIN orders
ON customers.customer_id = orders.customer_id;
```

Γράψτε σύνδεση πίνακα με τον εαυτό του (SELF JOIN).

```
SELECT a.column_name, b.column_name...
FROM table1 a, table1 b
WHERE a.common_field = b.common_field;
```

```
SELECT  a.customer_ID, a.customer_name, a.credit_limit,
  b.customer_ID, b.customer_name, b.credit_limit
 FROM CUSTOMERS a, CUSTOMERS b
 WHERE a.credit_limit = b.credit_limit;
```

Δείξτε για κάθε υπάλληλο τον κωδικό (customer_ID) και το πιστωτικό του όριο (credit_limit) μαζί με τους κωδικούς και το πιστωτικό όριο των υπαλλήλων που έχουν μεγαλύτερο πιστωτικό όριο από αυτόν.

```
SELECT  a.customer_ID, a.customer_name, a.credit_limit,
  b.customer_ID, b.customer_name, b.credit_limit
 FROM CUSTOMERS a, CUSTOMERS b
 WHERE a.credit_limit < b.credit_limit
ORDER BY a.customer_ID;
```

Κεφάλαιο 5
Μελέτη περίπτωσης. Πληροφοριακό Σύστημα Νοσοκομείων. Υλοποίηση σχεσιακής βάσης δεδομένων με τη γλώσσα SQL (Structured Query Language). Σχεδίαση της διεπαφής χρήστη.

Σύνοψη
Στη Μελέτη Περίπτωσης επιχειρείται να προσεγγιστεί σφαιρικά ένα Σύστημα Βάσης Δεδομένων μεγαλύτερης κλίμακας και να περιγραφεί ο σχεδιασμός και η υλοποίησή του. Παρατίθεται περιγραφή του συστήματος, ανάλυση δεδομένων (data analysis), μοντελοποίηση και κανονικοποίηση. Στη συνέχεια παρατίθενται δηλώσεις για τη δημιουργία και την αξιοποίηση του συστήματος. Τέλος, παρατίθεται σχεδίαση της διεπαφής του χρήστη.

Παράλληλα με την παρουσίαση της Μελέτης Περίπτωσης ο αναγνώστης καλείται μέσα από σειρά λυμένων θεμάτων και λυμένων ασκήσεων να κάνει τις δικές του επιλογές και να σχεδιάσει και να υλοποιήσει το δικό του σύστημα. Κατά συνέπεια, η παρουσίαση του θέματος απαιτεί την ενεργή συμμετοχή του αναγνώστη. Αφου δοθεί μια πρώτη περιγραφή του συστήματος ο αναγνώστης καλείται να μελετήσει θέματα της ανάλυσης των δεδομένων του συστήματος, να σκεφτεί κάποιες προτεινόμενες λύσεις και τα αντίστοιχα μοντέλα δεδομένων και τελικά να σχεδιάσει το δικό του μοντέλο και τη δική του προσέγγιση στη σχεσιακή βάση δεδομένων. Η παρατιθέμενη υλοποίηση βασίζεται στο σχεδιασμό που προτάθηκε αλλά εύκολα ο αναγνώστης μπορεί να την προσαρμόσει στη δική του προσέγγιση. Η Μελέτη Περίπτωσης ολοκληρώνεται με μία σχεδίαση της διεπαφής του χρήστη αρκετά σαφή και απλή την οποία μπορεί να προσαρμόσει ο προγραμματιστής υλοποιώντας ένα αντίστοιχο «πραγματικό» σύστημα.

5.1 Περιγραφή του πληροφοριακού συστήματος νοσοκομείων. Ανάλυση δεδομένων και μοντελοποίηση

Παραθέτουμε αρχικά μία περιγραφή του πληροφοριακού συστήματος νοσοκομείων και το αποτέλεσμα της ανάλυση δεδομένων (data analysis) που αποτελεί την αφετηρία της διαδικασίας μοντελοποίησης.

Η εταιρεία Hospital Information System H.I.S. πρόκειται να εγκαταστήσει ένα πληροφοριακό σύστημα για ένα σύνολο νοσοκομείων. Η βάση δεδομένων (Hospital Database) του πληροφοριακού συστήματος περιλαμβάνει στοιχεία για:

Νοσοκομεία, κλινικές, γιατρούς, λοιπό προσωπικό, ασθενείς κ.λ.π.

Μετά από τη σχετική ανάλυση δεδομένων καταγράφονται μια σειρά χαρακτηριστικών (attributes) και περιορισμοί (constraints) ως εξής:

α) Χαρακτηριστικά (attributes)

Hosp _ code = κωδικός Νοσοκομείου π.χ., 99
Hosp _ name = Ονομασία π.χ., ΜΕΤΑΞΑ
Hosp _ address = Διεύθυνση π.χ., Μπόταση 20
Hosp _ city = έδρα π.χ., Αθήνα
Hosp _ phone = τηλεφωνικό κέντρο π.χ., 2104518411, 2104284444-54
Hosp _ Sumbed = αριθμός κρεβατιών π.χ., 2088

Ward _ code = κωδικός κλινικής π.χ., 3,4,5
Ward _ name = ονομασία κλινικής π.χ., Καρδιολογική , Ακτινοθεραπευτική
Ward _ sumbed =αριθμός κρεβατιών κλινικής π.χ., 120

Lab _ no = κωδικός εργαστηρίου π.χ., 6226
Lab _ name = ονομασία εργαστηρίου π.χ., Beta
Lab _ address = διεύθυνση π.χ., Αγ. Σπυρίδωνα
Lab _ city = έδρα π.χ., Αθήνα
Lab _ phone = τηλέφωνο π.χ., 2105910974, 2105910975

Doctor _ no = κωδικός γιατρού π.χ., 999
Doctor _ name = ονοματεπώνυμο π.χ., Αδαμόπουλος
Doctor _ speciality = ειδικότητα π.χ., καρδιολογία

Entry_date = ημερομηνία εισαγωγής ασθενούς στο Νοσοκομείο/ κλινική
Pat _ registration = κωδικός ασθενή π.χ., 12345
Pat _ name = ονοματεπώνυμο π.χ., Νίκου Η.
Pat _ address = διεύθυνση π.χ., Αγ. Νικολάου
Pat _ city = πόλη π.χ., Καβάλα
Pat _ birthdate = ημερομηνία γέννησης π.χ.,1954
Pat _ sex = φύλο π.χ., Α
Pat _ ssn = κωδικός ασφάλισης π.χ., 000Θ695697

Bed _ no = αριθμός κρεβατιού ασθενή π.χ., 9

Emp _ no = κωδικός υπαλλήλου, που ανήκει στο νοσηλευτικό/λοιπό προσωπικό π.χ., 9999
Emp _ name = ονοματεπώνυμο π.χ., Σπύρου Μ.
Emp _ duty = καθήκον π.χ., νοσοκόμα
Emp _ shift = βάρδια π.χ., Β - βράδυ
Emp _ salary = μισθός, π.χ., 99999.99

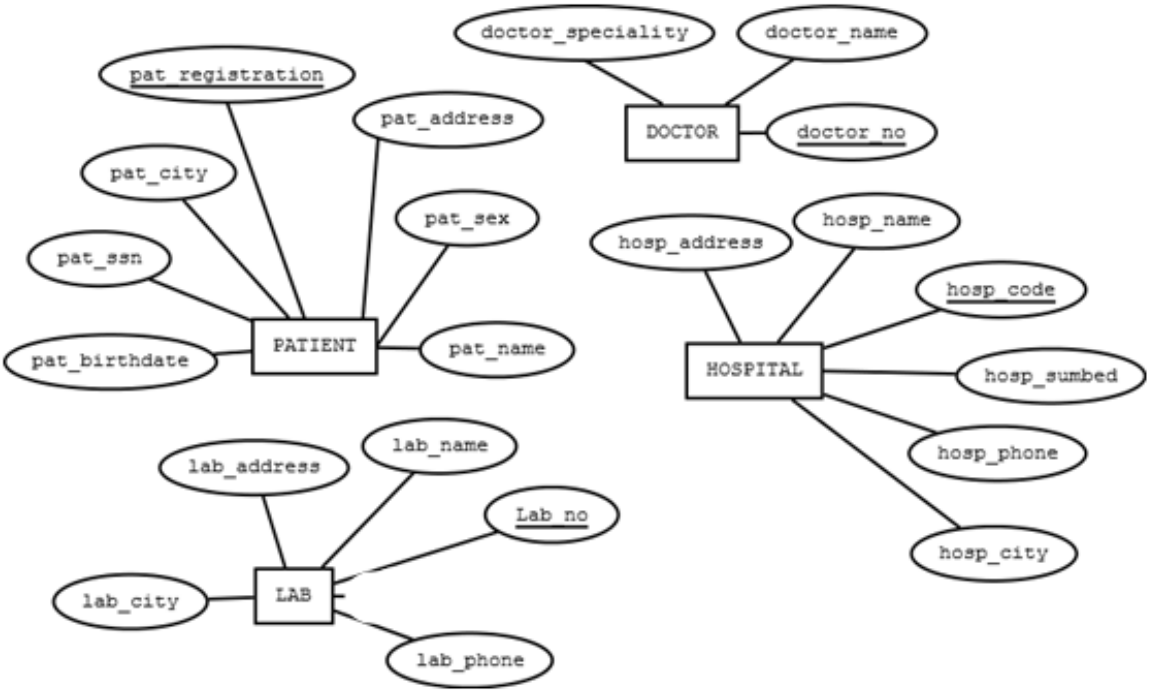
Test = κωδικός εργαστηριακής εξέτασης π.χ., T3
Result = αποτέλεσμα εργαστηριακής εξέτασης π.χ., Αρνητικό
Tdate = ημερομηνία εργαστηριακής εξέτασης

β) Περιορισμοί

Παραθέτουμε παραδείγματα περιορισμών:

1) Το νοσοκομείο έχει ένα μοναδικό κωδικό (hosp_code). Με διαφορετική διατύπωση, το χαρακτηριστικό (attribute) hosp_code είναι μοναδικό για το νοσοκομείο, δηλαδή καθορίζει (είναι κλειδί) για τα υπόλοιπα χαρακτηριστικά: Hosp_code → hosp_name, hosp_address κ.λπ.
2) Το χαρακτηριστικό pat_registration είναι μοναδικό για τον ασθενή, δηλαδή pat_registration → pat_name, pat_address κ.λπ.
3) Το χαρακτηριστικό doctor_no είναι μοναδικό για τον γιατρό, δηλαδή doctor_no → doctor_name, speciality
4) Το χαρακτηριστικό Lab_no είναι μοναδικό για το εργαστήριο, δηλαδή Lab _ no → Lab _ name, Lab _ address κλπ.
5) Η συσχέτιση μεταξύ HOSPITAL, DOCTOR είναι τύπου 1:N (Εικόνα 5.3). Κάθε νοσοκομείο έχει πολλούς γιατρούς. Κάθε γιατρός ανήκει σε ένα μόνο νοσοκομείο, δηλαδή Doctor_no □ hosp_code.

Οι οντότητες του Μοντέλου Οντοτήτων-Συσχετίσεων φαίνονται στην Εικόνα 5.1



Εικόνα 5.1 Οι οντότητες του Μοντέλου Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων

Στην Εικόνα 5.2 βλέπουμε τους πίνακες που αντιστοιχούν στο μοντέλο χωρίς να λάβουμε υπόψη κάποιες συσχετίσεις που υπάρχουν ανάμεσα στις οντότητες.

Πίνακας HOSPITAL

HOSP_CODE	HOSP_NAME	HOSP_ADDRESS	HOSP_CITY	HOSP_PHONE	HOSP_SUMBEDH

Πίνακας PATIENT

PAT_ REGISTRATION	PAT_NAME	PAT_ADDRESS	PAT_CITY	PAT_ BIRTHDATE	PAT_SEX	PAT_SSN

Πίνακας LAB

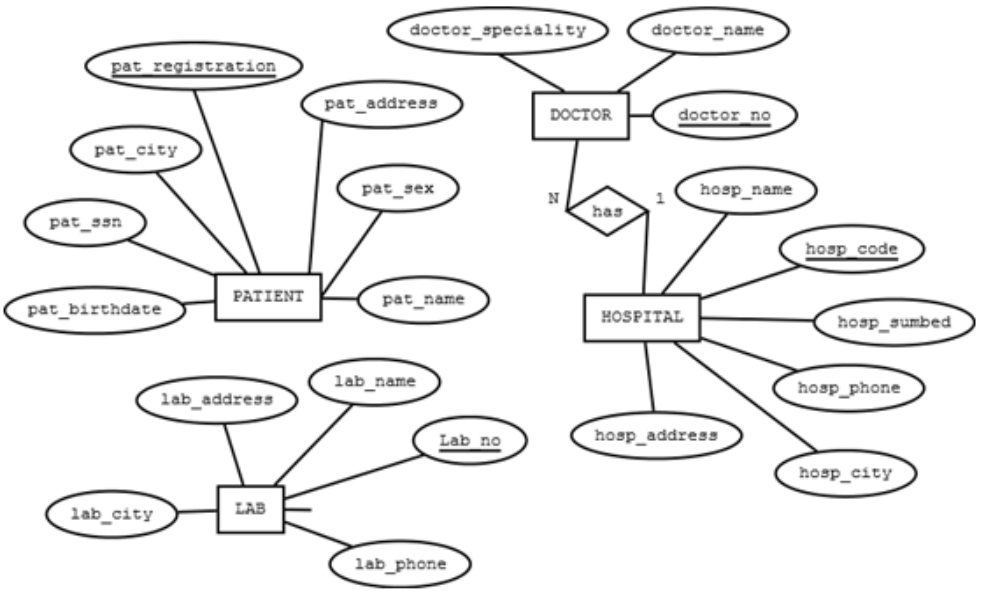
LAB_NO	LAB_NAME	LAB_ADDRESS	LAB_CITY	LAB_PHONE

Πίνακας DOCTOR

Θα συμπληρωθεί στη συνέχεια	DOCTOR_NO	NAME	SPECIALITY

Εικόνα 5.2 Πίνακες οι οποίοι αντιστοιχούν στις οντότητες του Μοντέλου Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων

Στην Εικόνα 5.3 βλέπουμε το ΜΟΣ και ειδικότερα τις οντότητες της εικόνας 5.2 και τη συσχέτιση has τύπου 1:N ανάμεσα στις οντότητες HOSPITAL, DOCTOR. Επίσης, βλέπουμε πως διαμορφώνεται ο πίνακας DOCTOR.



Πίνακας DOCTOR

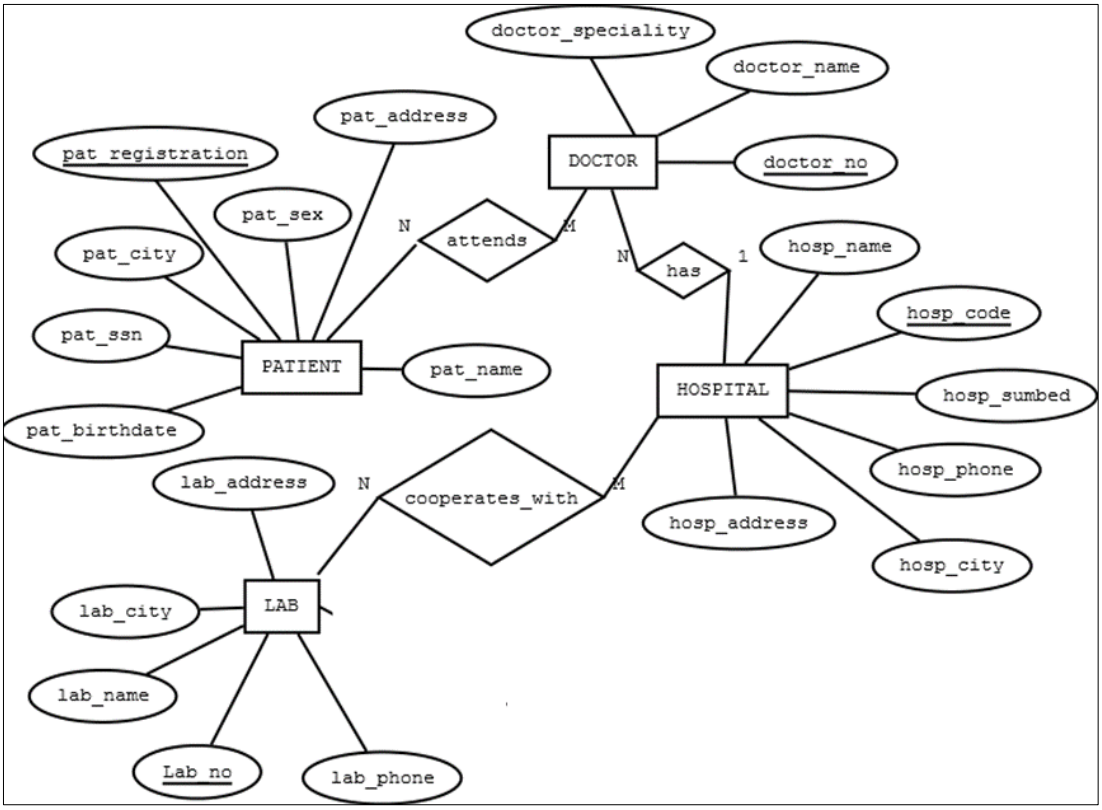
HOSP_CODE	DOCTOR_NO	NAME	SPECIALITY
Ξένο κλειδί	Πρωτεύον κλειδί		

Εικόνα 5.3Οι οντότητες του Μοντέλου Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων, με τη συσχέτιση has τύπου 1:N ανάμεσα στις οντότητες HOSPITAL, DOCTOR και την αντίστοιχη διαμόρφωση του πίνακα DOCTOR.

6) Η συσχέτιση μεταξύ HOSPITAL, LAB είναι τύπου M:N (Εικόνα 5.4). Ένα νοσοκομείο συνεργάζεται με πολλά εργαστήρια και ένα εργαστήριο συνεργάζεται με πολλά νοσοκομεία. Για τη συσχέτισης τύπου M:N θα κατασκευάσουμε πίνακα χρησιμοποιώντας τα κύρια κλειδιά των πινάκων HOSPITAL, LAB: (Hosp_code, lab_no) →

7) Η συσχέτιση μεταξύ DOCTOR, PATIENT είναι τύπου M:N (Εικόνα 5.4). Ένας γιατρός παρακολουθεί πολλούς ασθενείς και ένας ασθενής παρακολουθείται από πολλούς γιατρούς. Για τη συσχέτιση τύπου M:N θα κατασκευάσουμε πίνακα χρησιμοποιώντας τα κύρια κλειδιά των πινάκων DOCTOR, PATIENT: (Doctor_no, pat_registration) →

Στην Εικόνα 5.4 βλέπουμε το αντίστοιχο ΜΟΣ.



Εικόνα 5.4 Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων

Από τις συσχετίσεις τύπου M:N προκύπτουν δύο πίνακες.

Πίνακας ATT_DOCTOR

DOCTOR_NO	PAT_REGISTRATION
Ξένο κλειδί	Ξένο κλειδί

Το πρωτεύον κλειδί είναι σύνθετο (doctor_no, registration) και κάθε τμήμα του πρωτεύοντος κλειδιού είναι ξένο κλειδί.

Πίνακας HOSP_LAB

HOSPITAL_CODE	LAB_NO
Ξένο κλειδί	Ξένο κλειδί

Το πρωτεύον κλειδί είναι σύνθετο (hospital_code, lab_no) και κάθε τμήμα του πρωτεύοντος κλειδιού είναι ξένο κλειδί.

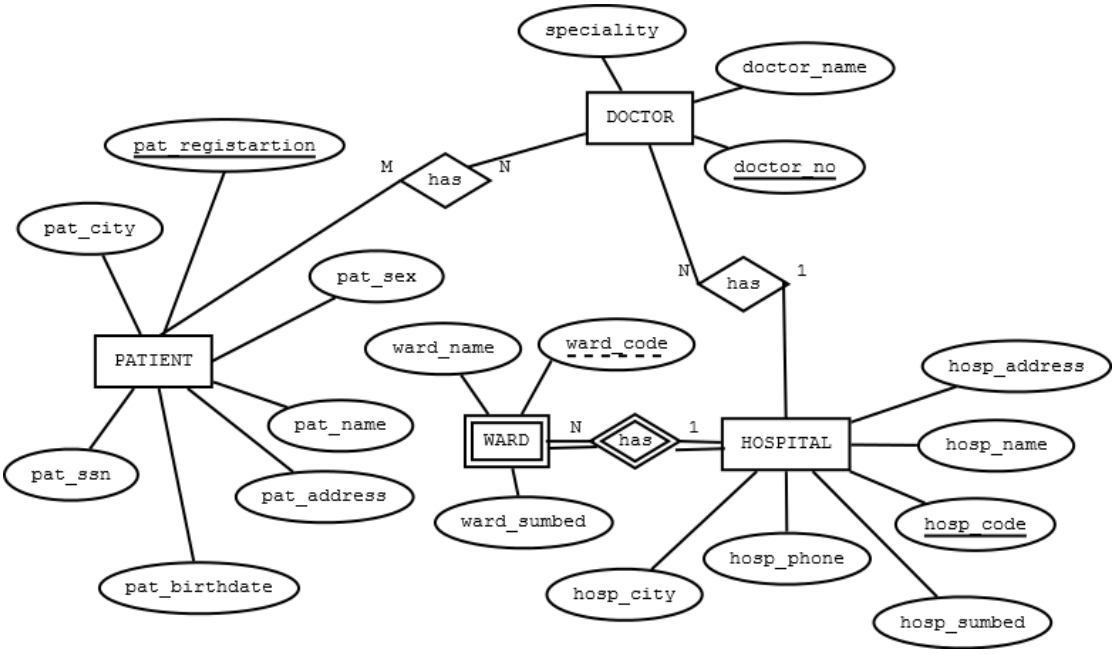
8) Θα εξετάσουμε τη συσχέτιση μεταξύ HOSPITAL, WARD. Προσοχή! Αρχικά θα θεωρήσουμε ότι η οντότητα WARD είναι εξαρτώμενη (weak entity) από την οντότητα HOSPITAL και η καθορίζουσα συσχέτιση has είναι τύπου 1:N (Εικόνα 5.5). Δηλαδή, υποθέτουμε σε κάθε νοσοκομείο η κλινική μπορεί να έχει διαφορετικό κωδικό. Ακολουθεί ο πίνακας για τις κλινικές και το αντίστοιχο ΜΟΣ

Πίνακας WARD

HOSP_CODE	WARD_CODE	WARD_NAME	WARD_SUMBEDW
Ξένο κλειδί			
10	101	ΚΑΡΔΙΟΛΟΓΙΚΟ	15
10	201	ΧΕΙΡΟΥΡΓΙΚΟ	35
20	110	ΚΑΡΔΙΟΛΟΓΙΚΟ	20

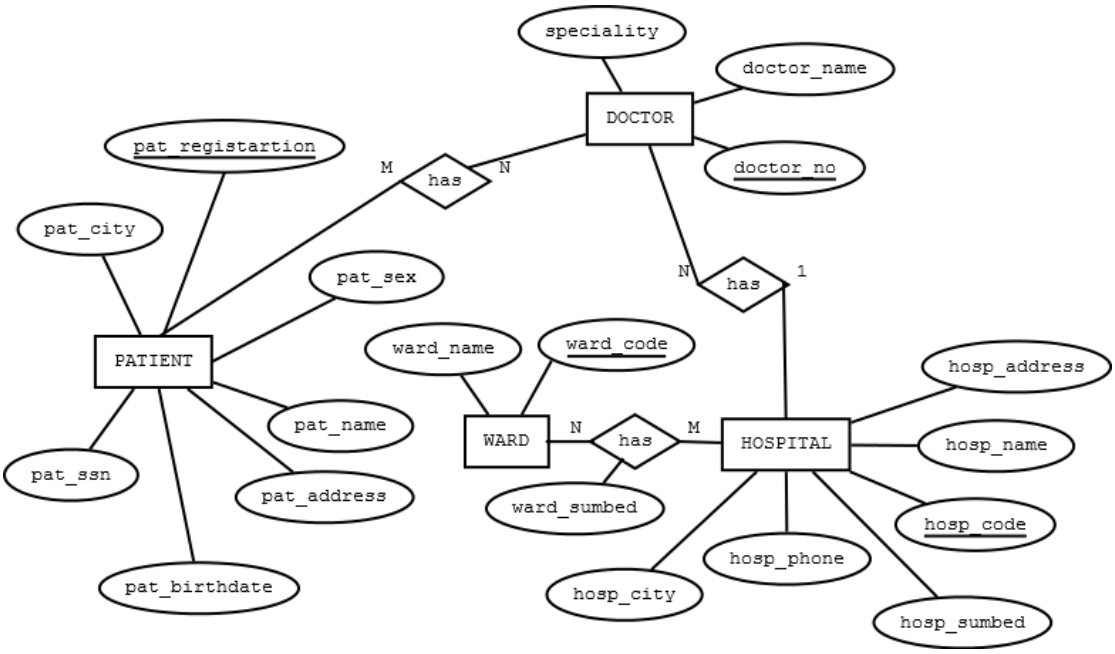
Το πρωτεύον κλειδί του πίνακα είναι (HOSP_CODE, WARD_CODE)

Στην υλοποίηση της βάσης δεδομένων παρακάτω θα χρησιμοποιούμε τον πίνακα αυτό.



Εικόνα 5.5 Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων. Η οντότητα WARD είναι εξαρτώμενη οντότητα (weak entity)

Στη συνέχεια, θα θεωρήσουμε ότι η οντότητα WARD κωδικοποιεί τις κλινικές για όλα τα νοσοκομεία του συστήματος βάσης δεδομένων. Ανάμεσα στην οντότητα WARD και στην οντότητα HOSPITAL η συσχέτιση has είναι τύπου M:N. Στην Εικόνα 5.6 βλέπουμε το αντίστοιχο ΜΟΣ.



Εικόνα 5.6 Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων. Η οντότητα WARD περιλαμβάνει την κωδικοποίηση των κλινικών που είναι κοινή για όλα τα νοσοκομεία

Ακολουθούν δύο πίνακες που περιλαμβάνουν όλα τα στοιχεία για τις κλινικές των νοσοκομείων. Ο Πίνακας WARD_ABBR περιλαμβάνει την κωδικοποίηση όλων των κλινικών. Κάθε κλινική έχει την ίδια κωδικοποίηση σε όλα τα νοσοκομεία του συστήματος βάσης δεδομένων. Στον πίνακα HOSP_WARD φαίνεται ποιες κλινικές έχει κάθε νοσοκομείο.

Πίνακας WARD_ABBR .

WARD_CODE	WARD_NAME
Πρωτεύον κλειδί	

Πίνακας HOSP_WARD

HOSP_CODE	WARD_CODE	WARD_SUMBEDW

Το πρωτεύον κλειδί είναι (HOSP_CODE, WARD_CODE) και κάθε τμήμα του είναι ξένο κλειδί.

Όπως προαναφέραμε στην υλοποίηση που θα παρατεθεί παρακάτω χρησιμοποιούμε τον πίνακα WARD με πρωτεύον κλειδί (HOSP_CODE, WARD_CODE)

Πίνακας WARD

HOSP_CODE	WARD_CODE	WARD_NAME	WARD_SUMBEDW

Η εισαγωγή ασθενούς σε κλινική θα μπορούσε να περιγραφεί ως εξής:

Πίνακας OCCUPANCY

HOSPITAL_CODE	WARD_CODE	PAT_REGISTRATION	BED_NO	ENTRY_DATE

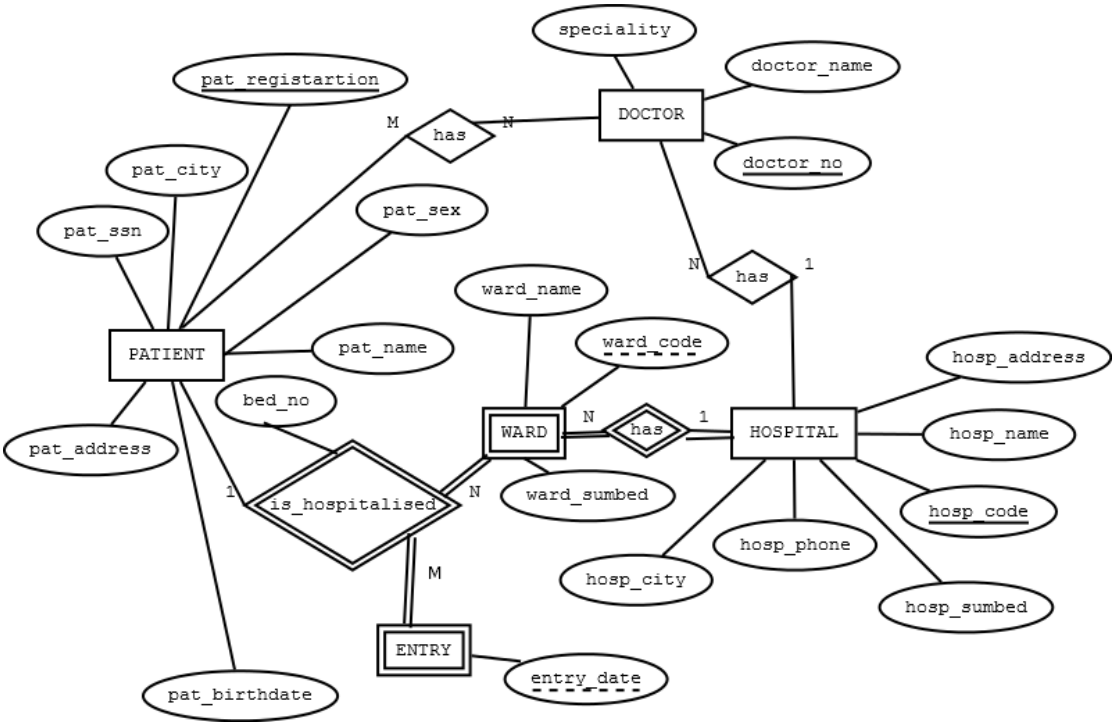
Στην περίπτωση αυτή το πρωτεύον κλειδί του πίνακα είναι (registration, entry_date) και υποτίθεται, επομένως, ότι σε μία συγκεκριμένη ημερομηνία γίνεται μία εισαγωγή ασθενούς σε μία κλινική ενός νοσοκομείου. Ισχύει PAT_REGISTRATION, ENTRY_DATE → BED _ NO, HOSP_CODE, WARD_CODE.

Εναλλακτικά θα μπορούσαμε να εισάγουμε μία στήλη Accno ή Entry_no ως πρωτεύον κλειδί του πίνακα. Ισχύει ENTRY_NO/ACCNO → HOSP_CODE, WARD_CODE, PAT_REGISTRATION, ENTRY_DATE, BED _ NO.

Accno/Entry_no	HOSPITAL_CODE	WARD_CODE	PAT_REGISTRATION	BED_NO	ENTRY_ DATE
Πρωτεύον κλειδί					

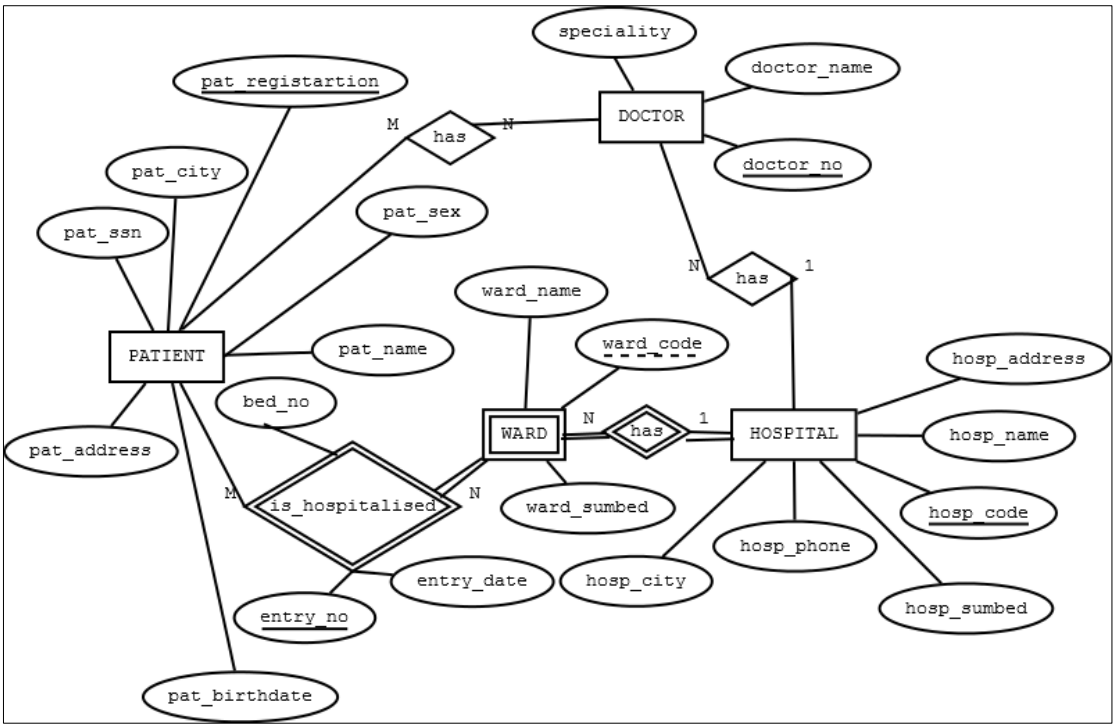
Άσκηση

Στις εικόνες 5.7, 5.8. 5.9, 5.10 παραθέτουμε εναλλακτικές σχεδιάσεις ΜΟΣ. Συζητήστε τις σχεδιάσεις αυτές. Ποιά επιλέγετε και γιατί;



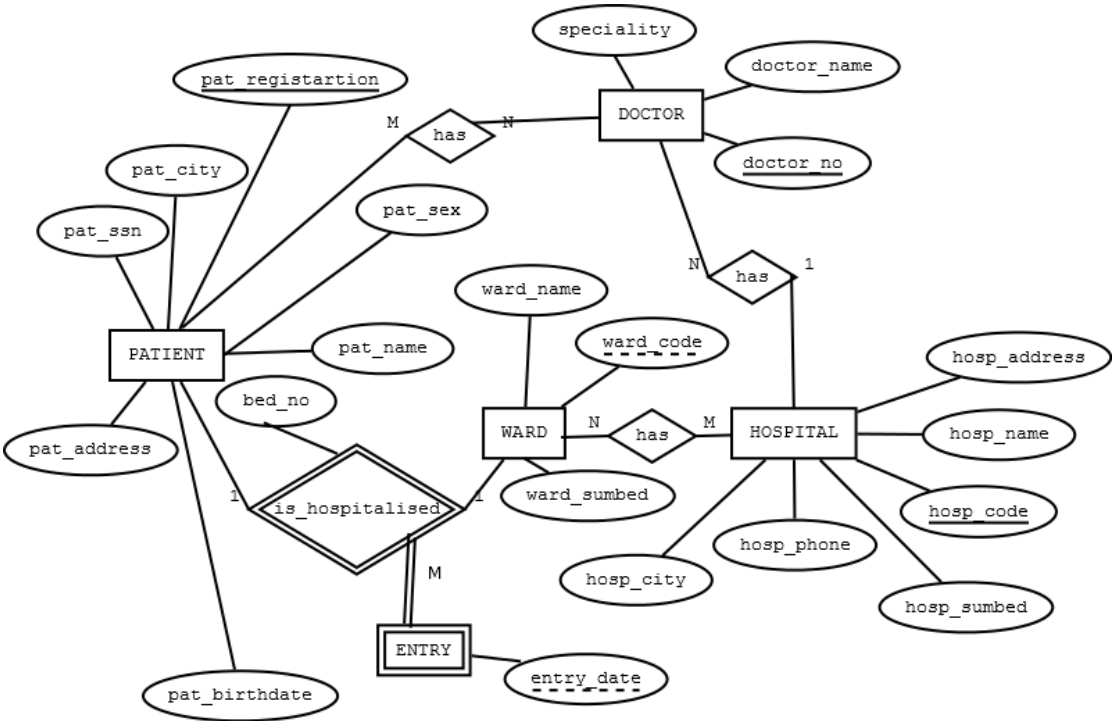
Εικόνα 5.7 Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων. Η εισαγωγή ασθενούς αναπαρίσταται ως εξαρτώμενη οντότητα.

Ακολουθεί δεύτερη εναλλακτική σχεδίαση του ΜΟ (Εικόνα 5.8).



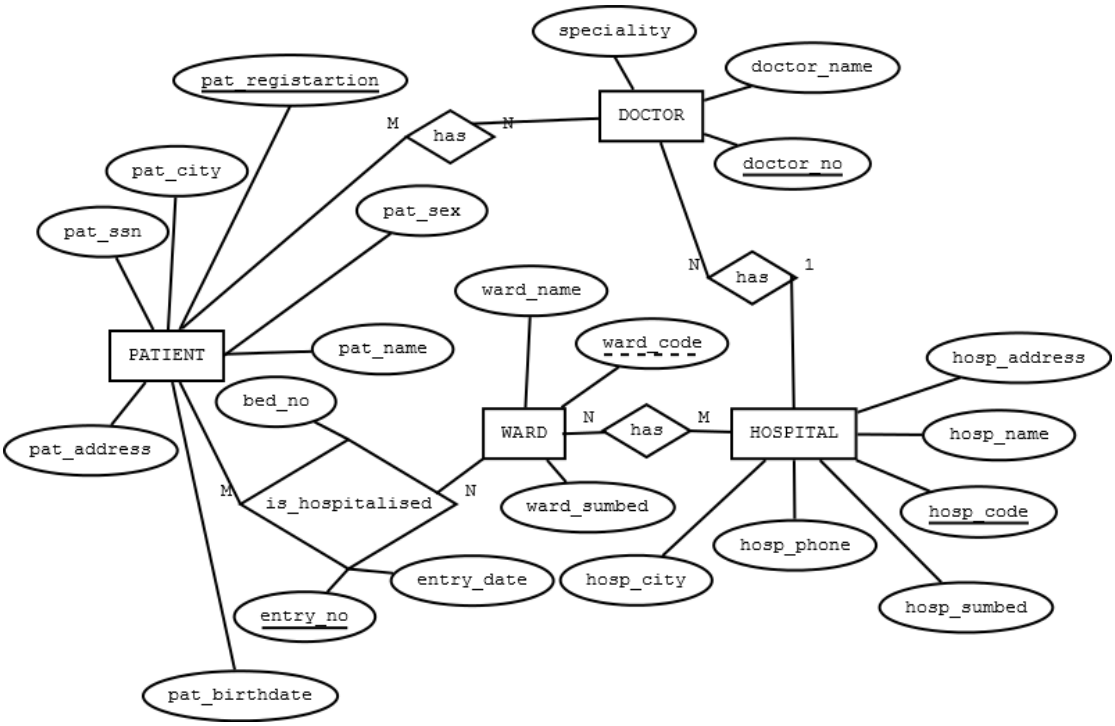
Εικόνα 5.8. Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων. Η εισαγωγή ασθενούς αναπαρίσταται ως συσχέτιση και η κλινική εξαρτώμενη οντότητα.

Ακολουθεί τρίτη εναλλακτική σχεδίαση του ΜΟΣ (Εικόνα 5.9).



Εικόνα 5.9 Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων. Η εισαγωγή ασθενούς είναι εξαρτώμενη οντότητα και η κλινική είναι ισχυρή οντότητα.

Ακολουθεί τέταρτη εναλλακτική σχεδίαση του ΜΟΣ.



Εικόνα 5.10 Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων. Η εισαγωγή ασθενούς είναι συσχέτιση και η κλινική είναι ισχυρή οντότητα.

Σε κάποια από τα μοντέλα υποτίθεται ότι κάθε νοσοκομείο έχει όλες τις κλινικές (παρατηρήστε τη διπλή γραμμή).

Οι εργαστηριακές εξετάσεις του ασθενούς θα μπορούσαν να περιγραφούν ως στήλες ενός πίνακα ή ως στήλες δύο πινάκων. Στην πρώτη περίπτωση έχουμε ένα πίνακα, ως εξής:

Πίνακας TEST

REGISTRATION	LAB_NO	KOD_TEST	RESULTS	TDATE

Θα μπορούσαμε να εισάγουμε πίνακα με περισσότερα στοιχεία εργαστηριακών εξετάσεων, ως εξής:

Πίνακας TEST

TEST_CODE	TEST_NAME	TEST_ABBR
80320	Ethanol, Blood	ALC
82657	Beta-Glucosidase, Leukocytes	BGL
84156	Protein, Total, 24 Hour, Urine	PTU
84311	Alpha-N-Acetylglucosaminidase, Serum	ANAS

Επιπλέον, θα πρέπει να χρησιμοποιήσουμε βέβαια και τον πίνακα των εργαστηριακών εξετάσεων.

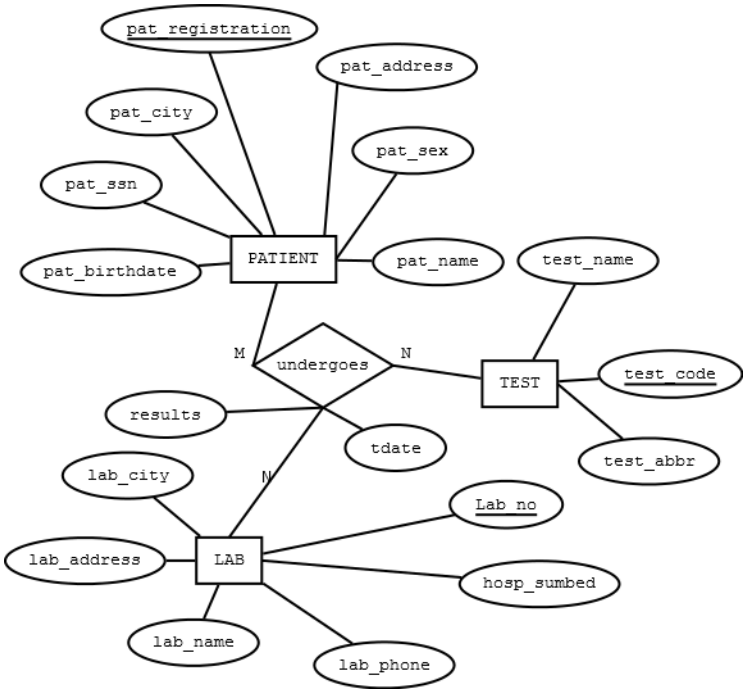
Πίνακας UNDERGOES

REGISTRATION	LAB_NO	KOD_TEST	RESULTS	TDATE

Ερώτηση

Ποιο είναι το πρωτεύον κλειδί;

Στην Εικόνα 5.11 παραθέτουμε το ΜΟΣ



Εικόνα 5.11 Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων.Οι εργαστηριακές εξετάσεις περιγράφονται από τριαδική συσχέτιση τύπου M:N:N

Είναι φανερό ότι ανάλογα με τους περιορισμούς που θα θέσετε επηρεάζονται και το μοντέλο οντοτήτων συσχετίσεων αλλά και οι στήλες των πινάκων σας.

5.2 Θέματα Μοντελοποίησης. Μοντέλο Οντοτήτων-Συσχετίσεων (ΜΟΣ) και Κανονικοποίηση

Με αφετηρία όλα τα στοιχεία που αναφέρθηκαν και συζητήθηκαν παραπάνω θα πρέπει να επιλέξετε κάποια από τα χαρακτηριστικά που συμπεριλάβαμε στη σύντομη περιγραφή του πληροφοριακού συστήματος και να προσθέσετε ενδεχομένως και άλλα χαρακτηριστικά τα οποία κρίνετε απαραίτητα. Στη συνέχεια θα πρέπει να κατασκευάσετε:

- Μοντέλο Οντοτήτων - Συσχετίσεων (Entity - Relationship model)
- Τρίτη Κανονική Μορφή (Third Normal Form)/Κανονική Μορφή Boyce-Codd.

Στην κανονική μορφή είναι προφανές ότι πρέπει να συμπεριλάβετε πίνακες όπως: HOSPITAL, WARD, PATIENT, DOCTOR, STAFF, LAB, TEST

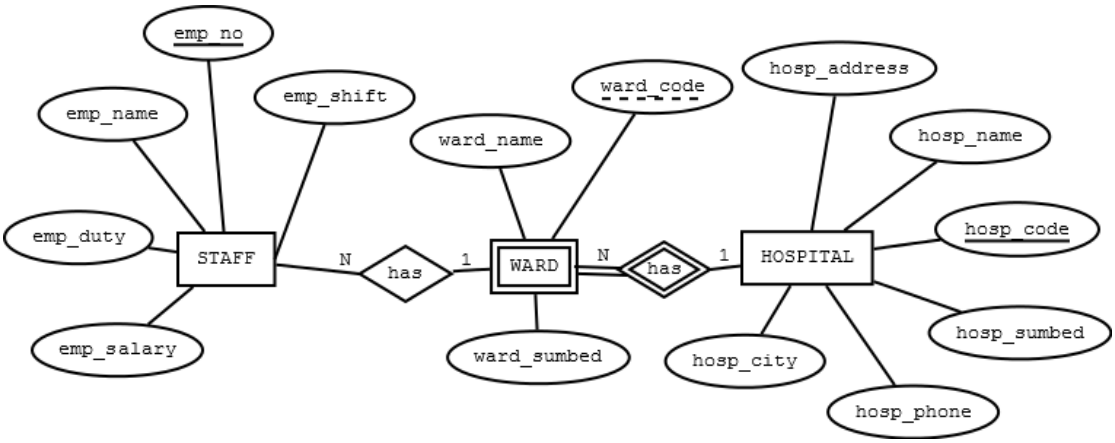
Ακολουθούν θέματα προς επίλυση με απαντήσεις ή/και υποδείξεις για την επίλυσή τους.

Θέμα 1

Υποθέτουμε ότι το νοσηλευτικό και το εν γένει λοιπό προσωπικό (STAFF) των νοσοκομείων ανήκει σε μία μόνο κλινική. Ακολουθεί ΜΟΣ για το προσωπικό. Περιγράψτε τους περιορισμούς που λάβαμε υπόψη για τη σχεδίασή του.

Υπόδειξη

Παρατηρήστε ότι στο μοντέλο της εικόνας 5.12 υποθέτουμε ότι δεν είναι υποχρεωτικό να έχουμε σε κάθε νοσοκομείο όλες τις κλινικές.



Εικόνα 5.12 Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων.Οι εργαστηριακές εξετάσεις περιγράφονται από τριαδική συσχέτιση τύπου M:N:N

Θέμα 2

Δίδεται η γραμμογράφηση των πινάκων WARD, STAFF:

WARD (***HOSP_CODE, WARD_CODE***, ward _ name, ward _ sumbed)

STAFF(***EMP_NO***, HOSP_CODE,WARD_CODE,EMP_NAME,EMP_DUTY,

EMP_SHIFT, EMP_SALARY).

Με ποίους περιορισμούς προέκυψε αυτή η γραμμογράφηση;

Υπόδειξη

Υποτίθεται ότι κάθε νοσοκομείο έχει κλινικές και κάθε κλινική έχει μοναδικό κωδικό στο νοσοκομείο. Επιπλέον, δεν υπάρχει κάποια κωδικοποίηση των κλινικών για όλα τα νοσοκομεία. Δείτε σχετικά και το ΜΟΣ της εικόνας 5.12 στο θέμα 1.

Θέμα 3

Η εταιρεία Hospital Information System H.I.S. υλοποιεί βάση δεδομένων που περιλαμβάνει στοιχεία για: Νοσοκομεία (hospital), κλινικές νοσοκομείων (ward), γιατρούς (doctor), ασθενείς (patient) και συνεργαζόμενα εργαστήρια (lab) για τις εργαστηριακές εξετάσεις (test) των ασθενών. Παραθέτουμε τις στήλες των πινάκων της βάσης δεδομένων που προέκυψαν από την ανάλυση δεδομένων:

Pat_registration = κωδικός ασθενή, Pat_name = ονοματεπώνυμο, Pat_address = διεύθυνση, Pat_birthdate = ημερομηνία γέννησης, Hosp_code = κωδικός Νοσοκομείου, Hosp_name = Ονομασία, Ward_code = κωδικός κλινική, Ward_name = ονομασία, Ward_abbr = συντομογραφία κλινικής, Ward_sumbed = αριθμός κρεβατιών, doctor_no = κωδικός γιατρού, doctor_name = όνομα γιατρού, speciality = ειδικότητα, Lab_no = κωδικός εργαστηρίου, Lab_name = ονομασία εργαστηρίου, Lab_address = διεύθυνση, Test_code = κωδικός εργαστηριακής εξέτασης, test_name = όνομα εξέτασης.

Ακολουθούν οι περιορισμοί.

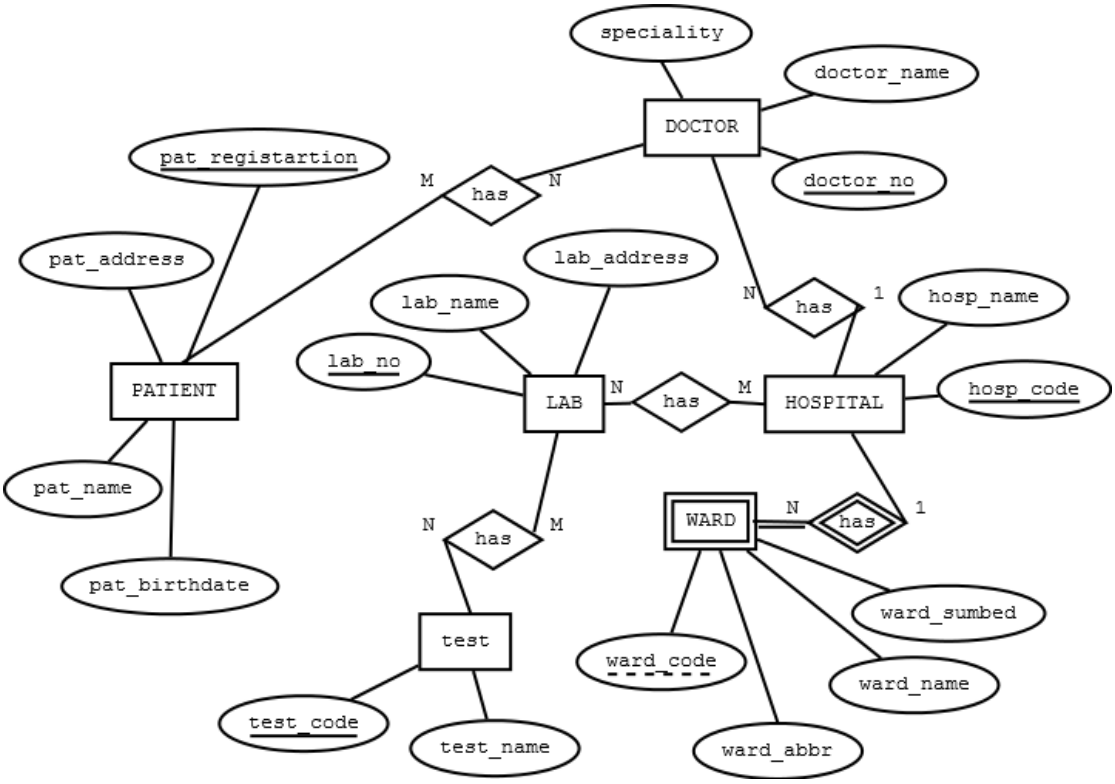
- Κάθε νοσοκομείο έχει κλινικές. Κάθε κλινική έχει έναν μοναδικό κωδικό για όλα τα νοσοκομεία, π.χ., (3, INTENSIVE CARE).
- Κάθε γιατρός ανήκει σε ένα νοσοκομείο και κάθε νοσοκομείο έχει πολλούς γιατρούς. Κάθε ασθενής μπορεί να παρακολουθείται από πολλούς γιατρούς και ένας γιατρός να παρακολουθεί πολλούς ασθενείς.
- Κάθε νοσοκομείο μπορεί να συνεργάζεται με πολλά εργαστήρια και κάθε εργαστήριο με πολλά νοσοκομεία.
- Σε ένα εργαστήριο γίνονται πολλές εξετάσεις και μία εξέταση μπορεί να γίνει σε πολλά εργαστήρια.

Κατασκευάστε Μοντέλο Οντοτήτων Συσχετίσεων για τη βάση δεδομένων της εταιρείας.

Υποδείξεις

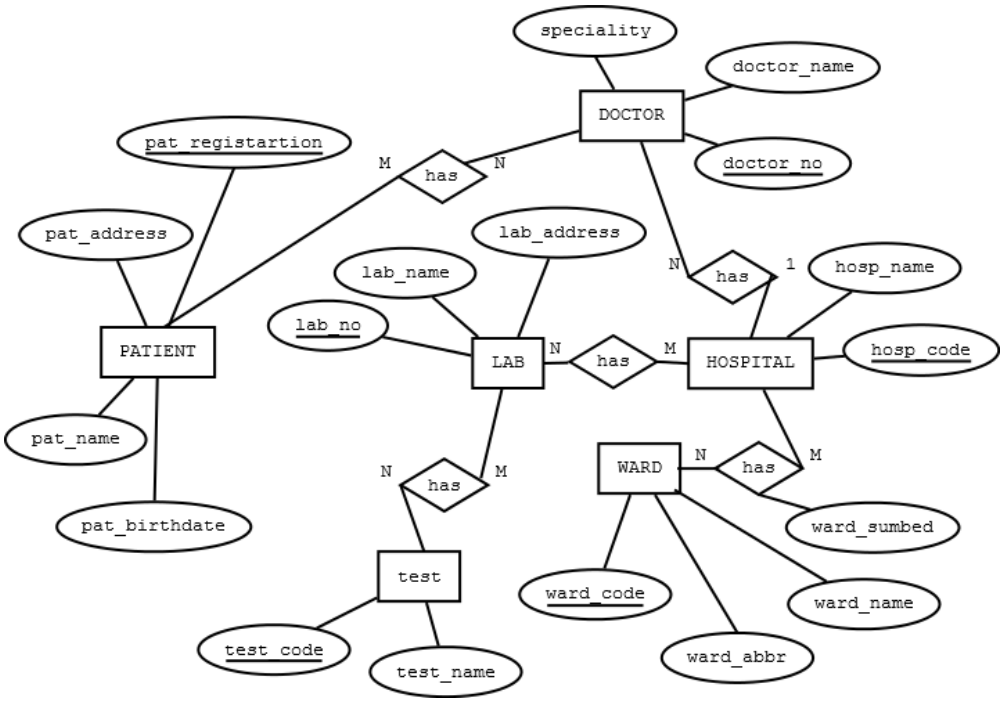
Παραθέτουμε δύο ενδεικτικές λύσεις.

Στην Εικόνα 5.13 βλέπουμε το Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων στο οποίο περιλαμβάνονται οι οντότητες Patient, Doctor, Hospital, Lab, Test και η εξαρτώμενη οντότητα Ward. Περιγράψτε τις συσχετίσεις μεταξύ των οντοτήτων και διατυπώστε τους περιορισμούς που ισχύουν. Το μοντέλο καλύπτει τις διαδικασίες της εισαγωγής των ασθενών στις κλινικές και τις εργαστηριακές τους εξετάσεις;



Εικόνα 5.13 Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων στο οποίο περιλαμβάνονται οι οντότητες patient, doctor, hospital, lab, test και η εξαρτώμενη οντότητα ward.

Στην Εικόνα 5.14 βλέπουμε το Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων στο οποίο περιλαμβάνονται οι οντότητες Patient, Doctor, Hospital, Lab, Test και Ward. Περιγράψτε τις συσχετίσεις μεταξύ των οντοτήτων και διατυπώστε τους περιορισμούς που ισχύουν. Το μοντέλο καλύπτει τις διαδικασίες της εισαγωγής των ασθενών στις κλινικές και τις εργαστηριακές τους εξετάσεις;



Εικόνα 5.14 Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων στο οποίο περιλαμβάνονται οι οντότητες patient, doctor, hospital, lab, test και ward.

Θέμα 4

Για την παραπάνω εταιρεία Hospital Information System H.I.S. θεωρούμε ότι οι ασθενείς μπορούν να κάνουν εξετάσεις σε διάφορα εργαστήρια που μπορεί να τις έχουν γράψει διάφοροι γιατροί. Για κάθε εργαστηριακή εξέταση (Test) που γίνεται σε κάποιο εργαστήριο εισάγουμε και τα εξής στοιχεία: Result=αποτέλεσμα εργαστηριακής εξέτασης, Tdate=ημερομηνία εργαστηριακής εξέτασης.

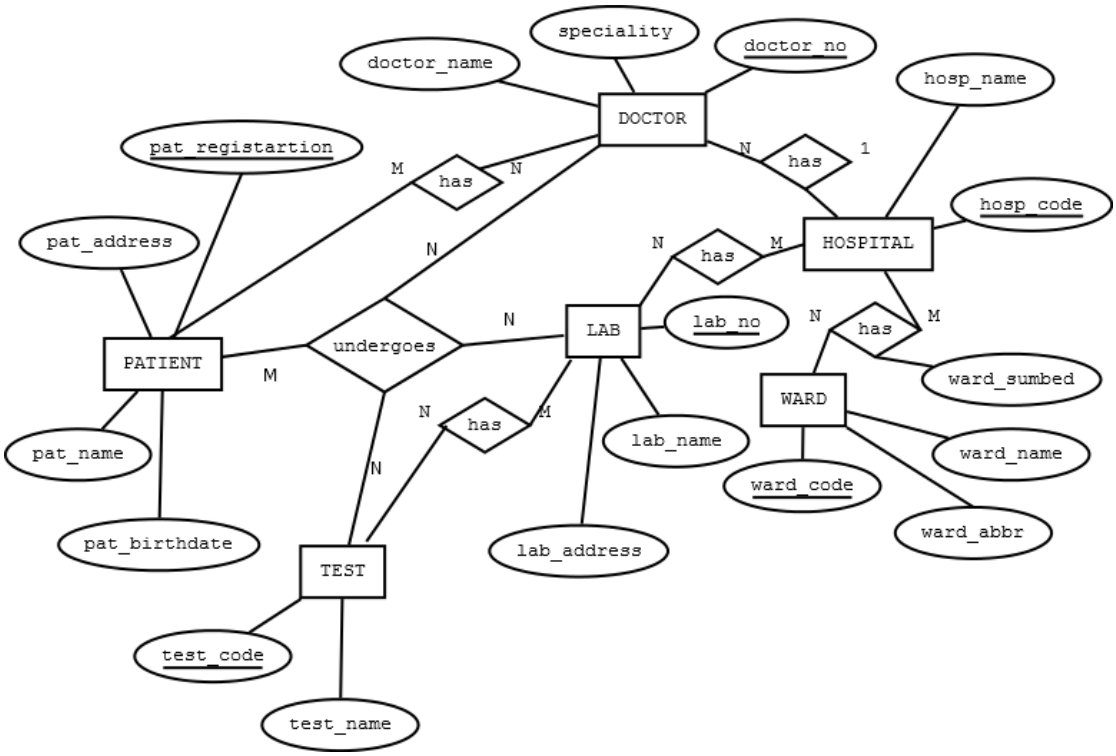
Να σχεδιάσετε ΜΟΣ και τους πίνακες στην Τρίτη κανονική μορφή για τη βάση δεδομένων της εταιρείας.

Υποδείξεις

Στην Εικόνα 5.15 βλέπουμε ενδεικτική λύση.

Παρατηρήστε ότι οι εργαστηριακές εξετάσεις περιγράφονται με τετραδική συσχέτιση μεταξύ των οντοτήτων Patient, Lab, Test, Doctor τύπου M:N:N:N.

Η συσχέτιση αυτή θα υλοποιηθεί με «συνδετικό» πίνακα. Κάποιες από τις επί μέρους συσχετίσεις τύπου M:N θα υλοποιηθούν επίσης ως πίνακες. Για παράδειγμα η συσχέτιση μεταξύ των οντοτήτων Lab, Test.



Εικόνα 5.15 Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων στο οποίο περιλαμβάνεται συσχέτιση μεταξύ τεσσάρων οντοτήτων τύπου M:N:N:N.

Παραθέτουμε ενδεικτικούς πίνακες οι οποίοι προκύπτουν από το ΜΟΣ της εικόνας 5.15.

Πίνακας HOSPITAL

HOSP_CODE	HOSP_NAME
Πρωτεύον κλειδί	

Πίνακας WARD_ABBR

WARD_ABBR	WARD_CODE	WARD_NAME	WARD_SUMBEDW
	Πρωτεύον κλειδί		

Πίνακας HOSP_WARD. Πρωτεύον κλειδί = (HOSP_CODE, WARD_CODE)

HOSP_CODE	WARD_CODE	WARD_SUMBEDW
Ξένο κλειδί	Ξένο κλειδί	

Πίνακας PATIENT

PAT_REGISTRATION	PAT_ADDRESS	PAT_NAME	PAT_BIRTHDATE
Πρωτεύον κλειδί			

Πίνακας DOCTOR

HOSP_CODE	DOCTOR_NO	DOCTOR_NAME	SPECIALITY
Ξένο κλειδί	Πρωτεύον κλειδί		

Πίνακας LAB

LAB_NO	LAB_NAME	LAB_ADDRESS
Πρωτεύον κλειδί		

Πίνακας TEST

TEST_CODE	TEST_NAME
Πρωτεύον κλειδί	

Πίνακας HOSP_LAB. Πρωτεύον κλειδί = (HOSP_CODE, LAB_NO)

HOSP_CODE	LAB_NO
Ξένο κλειδί	Ξένο κλειδί

Πίνακας TEST_LAB. Πρωτεύον κλειδί = (TEST_CODE, LAB_NO)

HOSP_CODE	LAB_NO
Ξένο κλειδί	Ξένο κλειδί

Πίνακας ATT_DOCTOR.

Πρωτεύον κλειδί=(PAT_REGISTRATION, DOCTOR_NO)

PAT_REGISTRATION	DOCTOR_NO
Ξένο κλειδί	Ξένο κλειδί

Πίνακας UNDERGOES.

ACCNO	PAT_REGISTRATION	DOCTOR_NO	LAB_NO	TEST_CODE	RESULTS	TEST_DATE
Πρωτεύον κλειδί	Ξένο κλειδί	Ξένο κλειδί	Ξένο κλειδί	Ξένο κλειδί		

5.3 Δημιουργία βάσης δεδομένων και απλές δηλώσεις (statements) SQL

Στη ενότητα αυτή σας καλούμε να χρησιμοποιήσετε τη γλώσσα SQL για να δημιουργήσετε τη βάση δεδομένων και με αφετηρία τα παρατιθέμενα παραδείγματα να οργανώσετε ένα απλό πληροφοριακό σύστημα.

Αρχικά πρέπει να δημιουργήσετε τους πίνακες της βάσης δεδομένων Νοσοκομείων:

PATIENT, HOSPITAL, WARD, STAFF, LAB, DOCTOR, HOSP_LAB, TEST κ.λπ.

Ακολουθεί ενδεικτική υλοποίηση της βάσης, χωρίς ξένα κλειδιά, την οποία μπορείτε να χρησιμοποιήσετε για να ορίσετε τους δικούς σας πίνακες! Η υλοποίηση χρησιμοποιεί το προϊόν της Oracle.

```
CREATE TABLE PATIENT (PAT_REGISTRATION NUMBER(5) NOT NULL,
 PAT_NAME VARCHAR2(20), PAT_ADDRESS VARCHAR2(15),
 PAT_BIRTHDATE DATE,
 PAT_SEX VARCHAR(1), PAT_SSN NUMBER(10),
 PRIMARY KEY (PAT_REGISTRATION));

CREATE TABLE HOSPITAL(HOSP_CODE NUMBER(2) NOT NULL,
 HOSP_NAME VARCHAR2(15), HOSP_ADDRESS VARCHAR2(15),
 HOSP_PHONE VARCHAR2(10), HOSP_SUMBEDH NUMBER(4),
 PRIMARY KEY (HOSP_CODE));
```

```
CREATE TABLE LAB ( LAB_NO NUMBER(2) NOT NULL,
 LAB_NAME VARCHAR2(10), LAB_ADDRESS VARCHAR2(15),
 LAB_PHONE VARCHAR2(10),
 PRIMARY KEY (LAB_NO));

CREATE TABLE HOSP_LAB(HOSPITAL_CODE NUMBER(2) NOT NULL,
 LAB_NO NUMBER(2) NOT NULL,
 PRIMARY KEY (LAB_NO,HOSPITAL_CODE));
```

```
CREATE TABLE WARD (HOSP_CODE NUMBER(2) NOT NULL,
  WARD_CODE NUMBER(1) NOT NULL,
  WARD_NAME VARCHAR2(20),
  WARD_SUMBEDW NUMBER(3),
  PRIMARY KEY (WARD_CODE, HOSP_CODE));

CREATE TABLE STAFF (HOSP_CODE NUMBER(2) NOT NULL,
  WARD_CODE NUMBER(1) NOT NULL,
  EMP_NO NUMBER(4) NOT NULL,
  EMP_NAME VARCHAR2(20), EMP_DUTY VARCHAR2(10),
  EMP_SHIFT VARCHAR2(1), EMP_SALARY NUMBER(7,2),
  PRIMARY KEY (EMP_NO));

CREATE TABLE TEST (REGISTRATION NUMBER(5) NOT NULL,
  LAB_NO NUMBER(2) NOT NULL,
  KOD_TEST VARCHAR2(3) NOT NULL,
  RESULTS VARCHAR2(4),
  TDATE DATE,
  PRIMARY KEY (REGISTRATION,LAB_NO,KOD_TEST,TDATE));

CREATE TABLE DOCTOR (HOSP_CODE NUMBER(2) NOT NULL,
  DOCTOR_NO NUMBER(3) NOT NULL,
  NAME VARCHAR2(15),
  SPECIALITY VARCHAR2(15),
  PRIMARY KEY (DOCTOR_NO));

CREATE TABLE ATT_DOCTOR( DOCTOR_NO NUMBER(3) NOT NULL,
REGISTRATION NUMBER(5) NOT NULL,
PRIMARY KEY (REGISTRATION,DOCTOR_NO));

CREATE TABLE OCCUPANCY(HOSPITAL_CODE NUMBER(2) NOT NULL,
  WARD_CODE NUMBER(1) NOT NULL,
  REGISTRATION NUMBER(5) NOT NULL,
  BED_NO NUMBER(1),
  ENTRY_DATE DATE NOT NULL,
  PRIMARY KEY (HOSPITAL_CODE,WARD_CODE,REGISTRATION,
   BED_NO, ENTRY_DATE));
```

5.3.1 Τροποποίηση ορισμών πινάκων και διαγραφή πινάκων

Η τροποποίηση του ορισμού (της δομής) πίνακα είναι απαραίτητη όταν αλλάζουν οι απαιτήσεις (οι περιορισμοί) και βεβαίως έχουν κάποιες επιπτώσεις στη βάση δεδομένων. Η διαγραφή πίνακα είναι μία σοβαρή υπόθεση και πρέπει να είμαστε προσεκτικοί γιατί υπάρχουν επιπτώσεις που δεν είναι εύκολο να αναιρεθούν.

Παράδειγμα 1

Προσθέστε στον πίνακα patient τη στήλη height NUMBER (3,2).

```
ALTER TABLE patient ADD(height NUMBER(3,2));
```

Αυτή η δήλωση προσθέτει μια στήλη επιπλέον, τη στήλη Height, στον πίνακα Patient. Αν ο πίνακας περιλαμβάνει στοιχεία θα πρέπει με δηλώσεις update να ενημερώσουμε τη στήλη.

Παράδειγμα 2

Αλλάξτε τον πίνακα patient τροποποιώντας το χαρακτηριστικό pat_name σε CHAR (25)

```
ALTER TABLE patient MODIFY(pat_name VARCHAR2(25));
```

Αυτή η δήλωση αυξάνει το μήκος της στήλης pat_name κατά 5 ψηφία.

Παράδειγμα 3

Προσπαθήστε να προσθέσετε στο πίνακα μια στήλη που είναι "NOT NULL" ενώ ο πίνακας έχει ήδη γραμμές. Τι παρατηρείτε;

```
ALTER TABLE patient ADD(Not_null_col NUMBER(3) NOT NULL);
```

Εμφανίζεται το μήνυμα

```
"TABLE MUST BE EMPTY TO ADD MANDATORY (NOT NULL) COLUMN",
```

δηλαδή ότι ο πίνακας πρέπει να είναι άδειος για να μπορέσουμε να προσθέσουμε μη κενή στήλη.

Αν είναι απαραίτητη η αλλαγή αυτή έχουμε να προτείνουμε μία λύση λίγο παρακάτω (δείτε παράδειγμα δημιουργίας πίνακα male).

Προσπαθήστε να ελαττώσετε το μήκος μιας στήλης. Τι παρατηρείτε;

```
ALTER TABLE patient MODIFY(pat_name CHAR(15));
```

Εμφανίζεται το μήνυμα :

```
 "column to be modified must be empty to decrease column length",
```

δηλαδή πρέπει να είναι άδειος ο πίνακας για να μειώσουμε το μήκος μιας στήλης του.

Παράδειγμα 4 Διαγραφή Πινάκων

Η δήλωση για να διαγράψουμε τον ορισμό πίνακα είναι απλή. Αν θέλουμε να διαγράψουμε τον πίνακα patient μπορούμε να χρησιμοποιήσουμε τη δήλωση:

```
DROP TABLE patient;
```

Προσοχή! Με τη δήλωση DROP TABLE δεν διαγράφουμε μόνο τον ορισμό αλλά και τα δεδομένα του πίνακα. Η δήλωση ROLLBACK δεν αναιρεί την ενέργεια μας γιατί η εντολή που δώσαμε συνεπάγεται αυτόματη εκτέλεση δήλωσης COMMIT.

Αν διαγράψατε τον πίνακα ξαναδημιουργήστε τον χρησιμοποιώντας τον αρχικό ορισμό του.

5.3.2 Εισαγωγή στοιχείων

Κύριος στόχος μας, στη μελέτη περιπτώσεως, είναι η διατύπωση κάποιας "φιλοσοφίας" για το πότε και πως θα χρησιμοποιούμε και ποιές δηλώσεις κάθε φορά της γλώσσας SQL. Ακολουθούν δηλώσεις εισαγωγής στοιχείων.

Με χρήση δηλώσεων INSERT μπορείτε να καταχωρήσετε τα στοιχεία των πινάκων. Οι δηλώσεις μπορούν να γραφτούν με πολλούς τρόπους, π.χ.,

```
INSERT INTO patient (pat_registration, pat_name, pat_address,
 pat_birthrate, pat_sex, pat_ssn)
VALUES ('99999', 'Σπύρου',  'Αγ. Σπυρίδωνα', ' 01/01/1975', ' A', 'Θ695697) ;

INSERT INTO patient
VALUES ('99999', 'Σπύρου',  'Αγ. Σπυρίδωνα', ' 01/01/1975', ' A', 'Θ695697) ;

INSERT INTO patient (pat_registration, pat_name) VALUES ('99999', 'Σπύρου');

INSERT INTO patient (pat_registration, pat_name)  VALUES ('&reg', '&name');
όπου  το σύμβολο & δηλώνει μια μεταβλητή .
```

Ακολουθούν οι δηλώσεις SQL για την εισαγωγή των δεδομένων.

```
INSERT INTO PATIENT VALUES (63827,'RUDIN','60
BERLIN','01/01/75','M',100973253,1.66);
INSERT INTO PATIENT VALUES (36658,'DATE','55
PARIS','08/04/84','M',660657471,1.65);
INSERT INTO PATIENT VALUES (64823,'FRANK','11
MASS','03/05/90','F',985201776,1.67);
INSERT INTO PATIENT VALUES (74835,'BROWN','15
ONTARIO','16/10/63','M',654811767,1.78);
INSERT INTO PATIENT ALUES (18004,'STONE','14
IVORY','22/01/2016','F',914991452,1.56);
INSERT INTO PATIENT VALUES (59076,'MILLER','80
LONDON','04/01/2001','F',611969044,1.69);
INSERT INTO PATIENT VALUES (24024,'FOURIER','40
DENMARK','09/07/96','F',321790059,1.58);
INSERT INTO PATIENT VALUES (10995,'LIST','58
OSLO','07/11/93','M',980862482,1.81);
INSERT INTO PATIENT VALUES (39217,'BATES','51
DALLAS','20/08/88','M',740294390,1.50);
INSERT INTO PATIENT VALUES (38702,'NEAL','65
HALIFAX','03/11/79','F',380010217,1.67);

INSERT INTO HOSPITAL VALUES (22,'DOCTORS', '45 BROWN','2109235411',412);
INSERT INTO HOSPITAL VALUES(13,'CENTRAL','333 SHANNON','2109645411',412);
INSERT INTO HOSPITAL VALUES(45,'CHILDRENS','555 UNIVERSITY','2105971500',845);
INSERT INTO HOSPITAL VALUES(18,'GENERAL','101 COLLEGE','2105953111',987);

INSERT INTO WARD VALUES (13,3,' INTENSIVE CARE',21);
INSERT INTO WARD VALUES (13,6,' PSYCHIATRIC',67);
INSERT INTO WARD VALUES (18,3,' INTENSIVE CARE',10);
INSERT INTO WARD VALUES (18,4,' CARDIAC',53);
INSERT INTO WARD VALUES (22,1,' RECOVERY',10);
```

```
INSERT INTO WARD VALUES (22,6,' PSYCHIATRIC',118);
INSERT INTO WARD VALUES (22,2,' MATERNITY',34);
INSERT INTO WARD VALUES (45,4,' CARDIAC',55);
INSERT INTO WARD VALUES (45,1,' RECOVERY',13);
INSERT INTO WARD VALUES (45,2,' MATERNITY',24);

INSERT INTO STAFF VALUES (22,6,1009,'HARRISON','NURSE','M',18500);
INSERT INTO STAFF VALUES (13,6,3754,'DATE','NURSE','A',17400);
INSERT INTO STAFF VALUES (22,6,8422,'BELL','ORDERLY','M',12600);
INSERT INTO STAFF VALUES (22,2,9901,'NELSON','INTERN','M',17000);
INSERT INTO STAFF VALUES (45,4,1280,'ANDERSON','INTERN','E',17000);
INSERT INTO STAFF VALUES (22,1,6065,'RANDALL','NURSE','E',20200);
INSERT INTO STAFF VALUES (13,6,3106,'HUGO','ORDERLY','A',13500);
INSERT INTO STAFF VALUES (18,4,6357,'KANTOR','INTERN','A',18300);
INSERT INTO STAFF VALUES (22,1,7379,'CAMERON','NURSE','A',16300);
INSERT INTO STAFF VALUES (45,1,8526,'FRANK','NURSE','A',19400);
INSERT INTO OCCUPANCY VALUES (13,3,10995,1,'10/01/97');
INSERT INTO OCCUPANCY VALUES (13,3,18004,2,'10/01/97');

INSERT INTO OCCUPANCY VALUES (13,3,24024,3,'10/01/97');
INSERT INTO OCCUPANCY VALUES (18,4,36658,1,'11/01/97');
INSERT INTO OCCUPANCY VALUES (18,4,38702,2,'11/01/97');
INSERT INTO OCCUPANCY VALUES (22,6,39217,1,'11/01/97');
INSERT INTO OCCUPANCY VALUES (22,6,59076,2,'12/01/2017');
INSERT INTO OCCUPANCY VALUES (22,6,63827,3,'12/01/2017');
INSERT INTO OCCUPANCY VALUES (22,2,64823,1,'12/01/2017');
INSERT INTO OCCUPANCY VALUES (45,1,74835,1,'12/01/2017');

INSERT INTO HOSP_LAB VALUES (13,16);
INSERT INTO HOSP_LAB VALUES (13,42);
INSERT INTO HOSP_LAB VALUES (18,56);
INSERT INTO HOSP_LAB VALUES (18,84);
INSERT INTO HOSP_LAB VALUES (18,16);
INSERT INTO HOSP_LAB VALUES (18,42);
INSERT INTO HOSP_LAB VALUES (22,56);
INSERT INTO HOSP_LAB VALUES (22,84);
INSERT INTO HOSP_LAB VALUES (22,16);

INSERT INTO TEST VALUES (10995,16,'ALC','PASS','12/01/2017');
INSERT INTO TEST VALUES (24024,42,'PTU','FAIL','12/01/2017');
INSERT INTO TEST VALUES (10995,16,'BGL','FAIL','12/01/2017');

INSERT INTO LAB VALUES (16,'ANGERS','14 MAIN','2105324453');
INSERT INTO LAB VALUES (42,'CAENI','55 KING','2104476448');
INSERT INTO LAB VALUES (56,'ALPHA','18 KIPPER','2109299611');
INSERT INTO LAB VALUES (84,'NICE','62 LYON','2103689793');

INSERT INTO DOCTOR VALUES (45,607,'ASHBY','PEDIATRICS');
INSERT INTO DOCTOR VALUES (18,585,'MILLER','GYNECOLOGY');
INSERT INTO DOCTOR VALUES (22,453,'GLASS','PEDIATRICS');
INSERT INTO DOCTOR VALUES (13,435,'LEE','CARDIOLOGY');
INSERT INTO DOCTOR VALUES (45,522,'ADAMS','NEYROLOGY');
INSERT INTO DOCTOR VALUES (22,398,'BEST','UROLOGY');
```

```
INSERT INTO DOCTOR VALUES (18,982,'RAMSEY','CARDIOLOGY');
INSERT INTO DOCTOR VALUES (22,386,'STONE','PSYCHIATRY');

INSERT INTO ATT_DOCTOR VALUES (435,10995);
INSERT INTO ATT_DOCTOR VALUES (435,18004);
INSERT INTO ATT_DOCTOR VALUES (982,36658);
INSERT INTO ATT_DOCTOR VALUES (982,38702);
INSERT INTO ATT_DOCTOR VALUES (398,39217);
INSERT INTO ATT_DOCTOR VALUES (386,39217);
INSERT INTO ATT_DOCTOR VALUES (453,39217);
INSERT INTO ATT_DOCTOR VALUES (386,59076);
INSERT INTO ATT_DOCTOR VALUES (453,64823);
INSERT INTO ATT_DOCTOR VALUES (522,74835);
```

5.4 Περιεχόμενο της βάσης δεδομένων

Το περιεχόμενο των πινάκων μετά την εισαγωγή στοιχείων είναι το εξής:

```
SELECT * FROM PATIENT;
```

PAT_ REGISTRATION	PAT_NAME	PAT_ADDRESS	PAT_ BIRTHDATE	PAT_ SEX	PAT_SSN	HEIGHT
63827	RUDIN	60 BERLIN	01/01/75	M	100973253	1,66
36658	DATE	55 PARIS	08/04/84	M	660657471	1,65
64823	FRANK	11 MASS	03/05/90	F	985201776	1,67
74835	BROWN	15 ONTARIO	16/10/63	M	654811767	1,78
18004	STONE	14 IVORY	22/01/2016	F	914991452	1,56
59076	MILLER	80 LONDON	04/01/2001	F	611969044	1,69
24024	FOURIER	40 DENMARK	09/07/96	F	321790059	1,58
10995	LIST	58 OSLO	07/11/93	M	980862482	1,81
39217	BATES	51 DALLAS	20/08/88	M	740294390	1,5
38702	NEAL	65 HALIFAX	03/11/79	F	380010217	1,67

```
SELECT * FROM HOSPITAL;
```

HOSP_ CODE	HOSP_NAME	HOSP_ADDRESS	HOSP_PHONE	HOSP_ SUMBEDH
22	DOCTORS	45 BROWN	2109235411	412
13	CENTRAL	333 SHANNON	2109645411	412
45	CHILDRENS	555 UNIVERSITY	2105971500	845
18	GENERAL	101 COLLEGE	2105953111	987

```
SELECT * FROM LAB;
```

LAB_NO	LAB_NAME	LAB_ADDRESS	LAB_PHONE
16	ANGERS	14 MAIN	2105324453
42	CAENI	55 KING	2104476448

56	ALPHA	18 KIPPER	2109299611
84	NICE	62 LYON	2103689793

```
SELECT * FROM HOSP_LAB;
```

HOSPITAL_CODE	LAB_NO
13	16
13	42
18	56
18	84
18	16
18	42
22	56
22	84
22	16

```
SELECT * FROM WARD;
```

HOSP_CODE	WARD_CODE	WARD_NAME	WARD_SUMBEDW
13	3	INTENSIVE CARE	21
13	6	PSYCHIATRIC	67
18	3	INTENSIVE CARE	10
18	4	CARDIAC	53
22	1	RECOVERY	10
22	6	PSYCHIATRIC	118
22	2	MATERNITY	34
45	4	CARDIAC	55
45	1	RECOVERY	13
45	2	MATERNITY	24

```
SELECT * FROM STAFF;
```

HOSP_ CODE	WARD_ CODE	EMP_ NO	EMP_NAME	EMP_ DUTY	EMP_ SHIFT	EMP_ SALARY
22	6	1009	HARRISON.	NURSE	M	18500
13	6	3754	DATE.	NURSE	A	17400
22	6	8422	BELL	ORDERLY	M	12600
22	2	9901	NELSON	INTERN	M	17000
45	4	1280	ANDERSON.	INTERN	E	17000
22	1	6065	RANDALL.	NURSE	E	20200

13	6	3106	HUGO.	ORDERLY	A	13500
18	4	6357	KANTOR.	INTERN	A	18300
22	1	7379	CAMERON.	NURSE	A	16300
45	1	8526	FRANK.	NURSE	A	19400

```
SELECT * FROM TEST;
```

REGISTRATION	LAB_NO	KOD_TEST	RESULTS	TDATE
10995	16	ALC	PASS	12/01/2017
24024	42	PTU	FAIL	12/01/2017
10995	16	BGL	FAIL	12/01/2017

```
SELECT * FROM DOCTOR;
```

HOSP_CODE	DOCTOR_NO	NAME	SPECIALITY
45	607	ASHBY	PEDIATRICS
18	585	MILLER.	GYNECOLOGY
22	453	GLASS.	PEDIATRICS
13	435	LEE.	CARDIOLOGY
45	522	ADAMS .	NEYROLOGY
22	398	BEST.	UROLOGY
18	982	RAMSEY.	CARDIOLOGY
22	386	STONE.	PSYCHIATRY

```
SELECT * FROM  ATT_DOCTOR;
```

DOCTOR_NO	REGISTRATION
435	10995
435	18004
982	36658
982	38702
398	39217
386	39217
453	39217
386	59076
453	64823
522	74835

```
SELECT * FROM OCCUPANCY;
```

HOSPITAL_CODE	WARD_CODE	REGISTRA TION	BED_NO	ENTRY_ DATE
13	3	10995	1	10/01/97

13	3	18004	2	10/01/97
13	3	24024	3	10/01/97
18	4	36658	1	11/01/97
18	4	38702	2	11/01/97
22	6	39217	1	11/01/97
22	6	59076	2	12/01/2017
22	6	63827	3	12/01/2017
22	2	64823	1	12/01/2017
45	1	74835	1	12/01/2017

Είναι ουσιαστική βοήθεια να ανατρέχετε σε ένα τέτοιο αντιπροσωπευτικό δείγμα δεδομένων για το σχεδιασμό και τον έλεγχο των αναζητήσεών σας

5.5 Χρήση του πληροφοριακού συστήματος

Ακολουθούν παραδείγματα.

Παράδειγμα 1

Δείξτε κωδικό, όνομα, διεύθυνση και ημερομηνία γέννησης των ασθενών, που γεννήθηκαν από το 1989 έως και το 2005, ταξινομημένα αλφαβητικά.

```
SELECT PAT_REGISTRATION, PAT_NAME, PAT_ADDRESS, PAT_BIRTHDATE
FROM PATIENT
WHERE PAT_BIRTHDATE BETWEEN '01/01/1989' AND '31/12/2005'
ORDER BY PAT_NAME;
```

PAT_ REGISTRATION	PAT_NAME	PAT_ADDRESS	PAT_BIRTHDATE
24024	FOURIER	40 DENMARK	09/07/96
64823	FRANK	11 MASS	03/05/90
10995	LIST	58 OSLO	07/11/93
59076	MILLER	80 LONDON	04/01/2001

Παράδειγμα 2

Δείξτε κωδικό, όνομα, διεύθυνση και ημερομηνία γέννησης των ασθενών, που γεννήθηκαν από το 1989 έως και το 2005, ταξινομημένους ανά ημερομηνία γέννησης και αλφαβητικά.

```
SELECT PAT_REGISTRATION, PAT_NAME, PAT_ADDRESS, PAT_BIRTHDATE
FROM PATIENT
WHERE PAT_BIRTHDATE BETWEEN '01/01/1989' AND '31/12/2005'
ORDER BY PAT_BIRTHDATE, PAT_NAME;
```

5.5.1 Δημιουργία (υπο) πινάκων ή πως θα ορίσουμε χωρίς πολύ κόπο τους πίνακες ενός νέου συστήματος βάσεων δεδομένων που θα διαδεχτεί το παλαιότερο .

Παράδειγμα 1

Δημιουργήστε ένα πίνακα male, υποσύνολο του πίνακα patient, και γράψτε δήλωση INSERT ώστε να καταχωρήσετε όλους τους άνδρες του πίνακα patient στον πίνακα male.

```
CREATE TABLE male
   ( m_pat_registration number(5) not null,
 m_pat_name char(25),
 m_pat_address char(15),
 m_pat_birthdate date,
 m_pat_sex char(1),
 m_pat_ssn char(15));

INSERT INTO male
SELECT pat_registration, pat_name, pat_address, pat_birthdate,
  pat_sex, pat_ssn
FROM patient
WHERE pat_sex = 'M';

SELECT * FROM male;
```

M_PAT_ REGIST RATION	M_PAT_ NAME	M_PAT_ ADDRESS	M_PAT_ BIRTHDATE	M_PAT_ SEX	M_PAT_SSN
63827	RUDIN	60 BERLIN	01/01/75	M	100973253
36658	DATE	55 PARIS	08/04/84	M	660657471
74835	BROWN	15 ONTARIO	16/10/63	M	654811767
10995	LIST	58 OSLO	07/11/93	M	980862482
39217	BATES	51 DALLAS	20/08/88	M	740294390

5.5.2 Απλές δηλώσεις τροποποίησης δεδομένων πινάκων

Παράδειγμα 1

Αλλάξτε την διεύθυνση ενός ασθενούς.

```
update patient
set pat_address = 'THISSEUS'
where pat_registration = 36658;
```

Καλύτερα θα ήταν να βλέπαμε ασθενή, δηλαδή το όνομά του, και μετά να αλλάζαμε στοιχεία του. Επομένως,

```
update patient
set pat_address = 'THISSEUS'
where pat_name = 'RUDIN'
AND pat_registration = 36658;
```

Παράδειγμα 2

Η διεύθυνση και η ημερομηνία γέννησης κάποιου ασθενούς να γίνουν ίδιες με τις αντίστοιχες άλλου ασθενή.

```
Update patient
set (pat_address,pat_birthdate)=(select pat_address,pat_birthdate
 from patient
 where pat_name = 'RUDIN' )
where pat_name = 'FRANK';
```

Παράδειγμα 3

Δείξτε διακεκριμένα καθήκοντα (duties) από τον πίνακα staff.

```
select distinct emp_duty from staff;
```

EMP_DUTY
ORDERLY
NURSE
INTERN

Παράδειγμα 4

Διαλέξτε καθήκοντα (duty) και βάρδιες (shift) από τον πίνακα staff.

```
select emp_duty, emp_shift from staff;
```

EMP_DUTY	EMP_SHIFT
NURSE	M
NURSE	A
ORDERLY	M
INTERN	M
INTERN	E
NURSE	E
ORDERLY	A
INTERN	A
NURSE	A
NURSE	A

Παράδειγμα 5

Διαλέξτε διακεκριμένα καθήκοντα (duty) και βάρδιες (shift) από τον πίνακα staff.

```
select distinct emp_duty, emp_shift from staff;
```

EMP_DUTY	EMP_SHIFT
NURSE	M
NURSE	E
ORDERLY	A

INTERN	A
ORDERLY	M
INTERN	E
INTERN	M
NURSE	A

5.5.3 Χρήση Ψευδωνύμου Πίνακα

Παράδειγμα 1

Δείξτε registration, name, address για ασθενή χρησιμοποιώντας το ψευδώνυμο p για τον πίνακα patient.

```
select p.pat_registration, p.pat_name, p.pat_address
from patient p;
```

PAT_REGISTRATION	PAT_NAME	PAT_ADDRESS
63827	RUDIN	THISSEUS
36658	DATE	THISSEUS
64823	FRANK	THISSEUS
74835	BROWN	15 ONTARIO
18004	STONE	14 IVORY
59076	MILLER	80 LONDON
24024	FOURIER	40 DENMARK
10995	LIST	58 OSLO
39217	BATES	51 DALLAS
38702	NEAL	65 HALIFAX

Η αναζήτηση αυτή με την απαίτηση για χρήση ψευδωνύμου του πίνακα δε φαίνεται να έχει ιδιαίτερο νόημα αλλά μας υπενθυμίζει πόσο χρήσιμη είναι η δυνατότητα χρήσης ψευδωνύμων στις αναζητήσεις .

5.5.4 Ταξινόμηση Στοιχείων

Παράδειγμα 1

Ταξινομήστε τους ασθενείς ως προς ημερομηνία γέννησης (birthdate). Αν δύο ασθενείς έχουν γεννηθεί την ίδια ημέρα ταξινομήστε τους ανά φθίνουσα σειρά κωδικού (registration).

```
select p.pat_registration,p.pat_birthdate, p.pat_name, p.pat_address
from patient p
order by pat_birthdate,pat_registration desc;
```

PAT_REGISTRATION	PAT_BIRTHDATE	PAT_NAME	PAT_ADDRESS
74835	16/10/63	BROWN	15 ONTARIO
64823	01/01/75	FRANK	THISSEUS
63827	01/01/75	RUDIN	THISSEUS

38702	03/11/79	NEAL	65 HALIFAX
36658	08/04/84	DATE	THISSEUS
39217	20/08/88	BATES	51 DALLAS
10995	07/11/93	LIST	58 OSLO
24024	09/07/96	FOURIER	40 DENMARK
59076	04/01/2001	MILLER	80 LONDON
18004	22/01/2016	STONE	14 IVORY

Με τον τρόπο αυτό η αναζήτηση δείχνει όλους τους ασθενείς κατά αύξουσα ημερομηνία γεννήσεως . Όσοι γεννήθηκαν την ίδια ημερομηνία εμφανίζονται κατά φθίνουσα σειρά κωδικού .

5.5.5 Αναζητήσεις που περιλαμβάνουν υπολογισμούς, πράξεις σε strings, χρήση συναρτήσεων

5.5.5.1 Διαχείριση Ημερομηνιών

Παράδειγμα 1

Προσθέστε 10 ημέρες στη σημερινή ημερομηνία

```
select sysdate + 10 from dual;
```

SYSDATE+10
27/01/2017

Παράδειγμα 2

Πότε περίπου ήταν ή θα είναι 30 χρονών οι ασθενείς ;

```
select pat_name,add_months(pat_birthdate, 30*12) from patient;
```

PAT_NAME	ADD_MONTHS(PAT_BIRTHDATE,30*12)
RUDIN	01/01/05
DATE	08/04/14
FRANK	01/01/05
BROWN	16/10/93
STONE	22/01/46
MILLER	04/01/31
FOURIER	09/07/26
LIST	07/11/23
BATES	20/08/18
NEAL	03/11/09

```
select pat_name,pat_birthdate+365*30 from patient;
```

PAT_NAME	PAT_BIRTHDATE+365*30
RUDIN	25/12/04
DATE	31/03/14
FRANK	25/12/04
BROWN	09/10/93
STONE	15/01/46
MILLER	27/12/30
FOURIER	01/07/26
LIST	30/10/23
BATES	12/08/18
NEAL	27/10/09

Παράδειγμα 3

Υπολογίστε την ημερομηνία τελευταίας μέρας του μήνα.

```
select last_day(sysdate) from dual;
```

LAST_DAY(SYSDATE)
31/01/17

5.5.5.2 Συνένωση τιμών στηλών κατα την εμφάνιση των αποτελεσμάτων

Παράδειγμα 1

Δείξτε "μαζί" , χωρίς κενά, όνομα και διεύθυνση ασθενών.

```
select pat_name||pat_address from patient;
```

PAT_NAME||PAT_ADDRESS
RUDIN THISSEUS
DATE THISSEUS
FRANK THISSEUS
BROWN 15 ONTARIO
STONE 14 IVORY
MILLER 80 LONDON
FOURIER 40 DENMARK
LIST 58 OSLO
BATES 51 DALLAS
NEAL 65 HALIFAX

5.5.5.3 Τελεστές Σύγκρισης

Παράδειγμα 1

Δείξτε ασθενείς που το όνομα τους αρχίζει από R ή S.

```
select pat_name from patient
where substr(pat_name,1,1) >'Q' and substr(pat_name,1,1) <'T';
```

PAT_NAME
RUDIN
STONE

```
select pat_name from patient
where substr(pat_name,1,1) between 'Q' and 'T';
```

PAT_NAME
RUDIN
STONE

```
select pat_name from patient
where pat_name like 'R%'
or pat_name like 'S%';
```

PAT_NAME
RUDIN
STONE

```
select pat_name from patient
where substr(pat_name,1,1)='R'
or substr(pat_name,1,1)='S';
```

PAT_NAME
RUDIN
STONE

Παράδειγμα 2

Δείξτε ασθενείς με κωδικό (pat_registration) που δεν ανήκει στο διάστημα [1000, 1500].

```
select pat_name, pat_registration from patient
where pat_registration not between 10000 and 15000;
```

PAT_NAME	PAT_REGISTRATION
RUDIN	63827
DATE	36658
FRANK	64823
BROWN	74835
STONE	18004

MILLER	59076
FOURIER	24024
BATES	39217
NEAL	38702

```
select pat_name, pat_registration from patient
where pat_registration < 1000
or pat_registration > 1500;
```

PAT_NAME
RUDIN
DATE
FRANK
BROWN
STONE
MILLER
FOURIER
LIST
BATES
NEAL

Παράδειγμα 3

Δείξτε υπαλλήλους με κωδικό (emp_no) που δεν ανήκει στο διάστημα [1000, 4500]. Θα δείχνετε κωδικό, όνομα, όνομα νοσοκομείου και όνομα κλινικής.

```
select emp_no, emp_name, hosp_name, ward_name
from hospital, ward, staff
where hospital.hosp_code=ward.hosp_code
AND ward.ward_code= staff.ward_code
AND staff.hosp_code=hospital.hosp_code
AND emp_no not between 10000 and 15000;

select emp_no, emp_name, hosp_name, ward_name
from hospital, ward, staff
where hospital.hosp_code=ward.hosp_code
AND ward.ward_code= staff.ward_code
AND staff.hosp_code=hospital.hosp_code
AND (emp_no < 1000 or emp_no > 1500);
```

Η παρακάτω δήλωση είναι λανθασμένη

```
select emp_no, emp_name, hosp_name, ward_name
from hospital, ward, staff
where hospital.hosp_code=ward.hosp_code
AND ward.ward_code= staff.ward_code
AND emp_no not between 10000 and 15000;
```

5.5.6 Χρήση συναρτήσεων

Παράδειγμα 1

Δείξτε τη θέση του πρώτου χαρακτήρα 'Α' στο όνομα κάθε ασθενούς

```
SELECT pat_name, INSTR (pat_name,'A', 1) from patient;
```

PAT_NAME	INSTR(PAT_NAME,'A',1)
RUDIN	0
DATE	2
FRANK	3
BROWN	0
STONE	0
MILLER	0
FOURIER	0
LIST	0
BATES	2
NEAL	3

Παράδειγμα 2

Δείξτε το μήκος κάθε ονόματος ασθενούς

```
select pat_name, length(rtrim(pat_name))
from patient;
```

PAT_NAME	LENGTH(PAT_NAME)
RUDIN	5
DATE	4
FRANK	5
BROWN	5
STONE	5
MILLER	6
FOURIER	7
LIST	4
BATES	5
NEAL	4

```
select pat_name, DECODE(rtrim(pat_name),
  'RUDIN', 'RASKY',
  'FRANK', 'FRASER',
  'SMITH')"new_name"
from patient;
```

PAT_NAME	new_name

RUDIN	RASKY
DATE	SMITH
FRANK	FRASER
BROWN	SMITH
STONE	SMITH
MILLER	SMITH
FOURIER	SMITH
LIST	SMITH
BATES	SMITH
NEAL	SMITH

Παράδειγμα 3

Δείξτε μέσο μισθό ανά κωδικό νοσοκομείου (hosp_code) στην περίπτωση που ο μέσος μισθός είναι μεγαλύτερος του 15000. Ταξινομήστε τα αποτελέσματα ανά κωδικό νοσοκομείου

```
select hosp_code,avg(emp_salary)
from staff
group by hosp_code
having avg(emp_salary) > 2000
order by hosp_code;

select hosp_code,avg(emp_salary)
from staff
group by hosp_code
having avg(emp_salary) > 15000
order by hosp_code;
```

HOSP_CODE	AVG(EMP_SALARY)
13	15450
18	18300
22	16920
45	18200

Παράδειγμα 4

Δείξτε μέγιστο και ελάχιστο μισθό και διαφορά μέγιστου και ελάχιστου μισθού υπαλλήλων νοσοκομείου.

```
select max(emp_salary), min(emp_salary),
   max(emp_salary)-min(emp_salary)
from staff;
```

MAX(EMP_SALARY)	MIN(EMP_SALARY)	MAX(EMP_SALARY)-MIN(EMP_SALARY)
20200	12600	7600

Παράδειγμα 5

Δείξτε το άθροισμα μισθών των υπαλλήλων διαιρούμενο με τον αριθμό υπαλλήλων που έχουν μισθό.

```
select AVG(emp_salary), SUM(emp_salary) / COUNT(emp_salary) from staff;
```

SUM(EMP_SALARY) / COUNT(EMP_SALARY)
17020

Παράδειγμα 6

Δείξτε μέσο μισθό υπαλλήλων

```
select AVG(emp_salary) from staff;
```

AVG(EMP_SALARY)
17020

Παράδειγμα 7

Δείξτε το άθροισμα μισθών όλων των υπαλλήλων διαιρούμενο με τον αριθμό όλων των υπαλλήλων.

```
select SUM(emp_salary) / COUNT(*) from staff;
```

SUM(EMP_SALARY)/COUNT(*)
17020

Παράδειγμα 8

Γράψτε μία δήλωση SELECT και δείξτε στην περίπτωση που hospital_code ισούται με 22 τα εξής: ward_code, μέσο μισθό για κάθε ward_code όταν ο μέσος μισθός είναι μεγαλύτερος ή ίσος του 17000, και τον αριθμό των ανακτηθέντων γραμμών (rows).

```
select ward_code,avg(emp_salary),count(*)
from staff
where hosp_code = 22
group by ward_code
having avg(emp_salary) >= 17000
order by ward_code ;
```

WARD_CODE	AVG(EMP_SALARY)	COUNT(*)
1	18250	2
2	17000	1

```
SELECT * FROM STAFF WHERE HOSP_CODE=22;
```

HOSP_CODE	WARD_CODE	EMP_NO	EMP_NAME	EMP_ DUTY	EMP_ SHIFT	EMP_ SALARY
22	6	1009	HARRISON.	NURSE	M	18500
22	6	8422	BELL	ORDERLY	M	12600
22	2	9901	NELSON	INTERN	M	17000
22	1	6065	RANDALL.	NURSE	E	20200
22	1	7379	CAMERON.	NURSE	A	16300

5.5.7 Χρήση join, group by

Παράδειγμα 1

Γράψτε μία δήλωση SELECT και δείξτε κωδικό νοσοκομείου (hospital_code), κωδικό κλινικής (ward_code), και τους κωδικούς υπαλλήλων (emp_no) των κλινικών. Στα αποτελέσματα να συμπεριλάβετε και κλινικές που δεν έχουν υπαλλήλους.

```
select ward.hosp_code, ward.ward_code,staff.emp_no from staff,ward
where ward.ward_code = staff.ward_code(+)
order by ward.ward_code;
```

WARD_CODE	EMP_NO	HOSP_CODE
1	8526	22
1	8526	45
1	7379	22
1	7379	45
1	6065	22
1	6065	45
2	9901	45
2	9901	22
3	-	18
3	-	13
4	6357	18
4	1280	18
4	1280	45
4	6357	45
6	1009	13
6	1009	22
6	3754	13
6	3754	22
6	3106	13
6	3106	22
6	8422	22
6	8422	13

Παράδειγμα 2

Γράψτε μία δήλωση SELECT και δείξτε κωδικό νοσοκομείου (hospital_code), κωδικό κλινικής (ward_code) για κάθε κλινική με περισσότερα από ένα μέλη προσωπικού.

```
select hosp_code,ward_code,ward_name
from ward w
where 1 < ( select count(*)
from staff s
```

```
where s.ward_code = w.ward_code
and s.hosp_code = w.hosp_code );
```

HOSP_CODE	WARD_CODE	WARD_NAME
13	6	PSYCHIATRIC
22	1	RECOVERY
22	6	PSYCHIATRIC

Δείτε και τις παρακάτω δηλώσεις:

```
SELECT hosp_code, ward_code, count(*)
FROM staff
GROUP BY hosp_code, ward_code;

SELECT hosp_code, ward_code, count(*)
FROM staff
GROUP BY hosp_code, ward_code
HAVING count(*)>1;

select hosp_code,ward_code,ward_name
from ward
where  (hosp_code, ward_code) IN
  (SELECT hosp_code, ward_code
FROM staff
GROUP BY hosp_code, ward_code
HAVING count(*)>1 );
```

Προσοχή! Ακολουθεί λανθασμένη δήλωση.

```
select hosp_code,ward_code,ward_name
from ward
where 1 < (select count(*)
from staff, ward
where staff.ward_code = ward.ward_code
and staff.hosp_code = ward.hosp_code
);
```

Παράδειγμα 3

Γράψτε μία δήλωση SELECT και δείξτε στοιχεία υπαλλήλων (emp_no, emp_name, emp_salary, hosp_code) που κερδίζουν περισσότερα από το μέσο όρο μισθού του νοσοκομείου τους.

```
select s.emp_no,s.emp_name, s.hosp_code, s.emp_salary
from staff s
where emp_salary > (select avg(emp_salary)
from staff x
where x.hosp_code = s.hosp_code);
```

EMP_NO	EMP_NAME
1009	HARRISON.
-3754	DATE.
9901	NELSON

6065	RANDALL.
8526	FRANK.

Δείτε και την παρακάτω δήλωση:

```
SELECT hosp_code, avg(emp_salary)
FROM staff
GROUP BY hosp_code;
```

5.5.8 Τελεστές UNION-INTERSECT-MINUS

Παράδειγμα 1

```
SELECT emp_name FROM staff WHERE hosp_code = 18
UNION
SELECT name FROM doctor WHERE hosp_code = 18;
```

EMP_NAME
KANTOR.
MILLER.
RAMSEY.

```
SELECT emp_name FROM staff WHERE hosp_code = 18;
```

EMP_NAME
KANTOR.

```
SELECT name FROM doctor WHERE hosp_code = 18;
```

NAME
MILLER.
RAMSEY.

5.5.9 Διαγραφή πινάκων

```
DROP TABLE PATIENT;

DROP TABLE HOSPITAL;

DROP TABLE LAB;

DROP TABLE HOSP_LAB;

DROP TABLE WARD;

DROP TABLE STAFF;

DROP TABLE TEST;

DROP TABLE MALE;
```

```
DROP TABLE DOCTOR;

DROP TABLE ATT_DOCTOR;

DROP TABLE OCCUPANCY;
```

5.5.10 Θέματα μοντελοποίησης και κανονικοποίησης βάσης δεδομένων και υλοποίηση με χρήση δηλώσεων MySQL.

Στην ενότητα αυτή παραθέτουμε θέματα σχεδίασης βάσης δεδομένων και υλοποίησης με χρήση δηλώσεων MySQL.

Θέμα 1

Η εταιρεία Hospital Information System H.I.S. υλοποιεί βάση δεδομένων που περιλαμβάνει στοιχεία για: Νοσοκομεία (hospital), κλινικές νοσοκομείων (ward), γιατρούς (doctor), ασθενείς (patient) και συνεργαζόμενα εργαστήρια (lab) για τις εργαστηριακές εξετάσεις (test) των ασθενών. Παραθέτουμε τις στήλες των πινάκων της βάσης δεδομένων:

Pat_registration = κωδικός ασθενή, Pat_name = ονοματεπώνυμο, Pat_address = διεύθυνση, Pat_birthdate = ημερομηνία γέννησης, Hosp_code = κωδικός Νοσοκομείου, Hosp_name = Ονομασία, Ward_code = κωδικός κλινική, Ward_name = ονομασία, Ward_abbr = συντομογραφία κλινικής, Ward_sumbed = αριθμός κρεβατιών, doctor_no = κωδικός γιατρού, doctor_name = όνομα γιατρού, speciality = ειδικότητα, Lab_no = κωδικός εργαστηρίου, Lab_name = ονομασία εργαστηρίου, Lab_address = διεύθυνση, Test_code = κωδικός εργαστηριακής εξέτασης, test_name = όνομα εξέτασης.

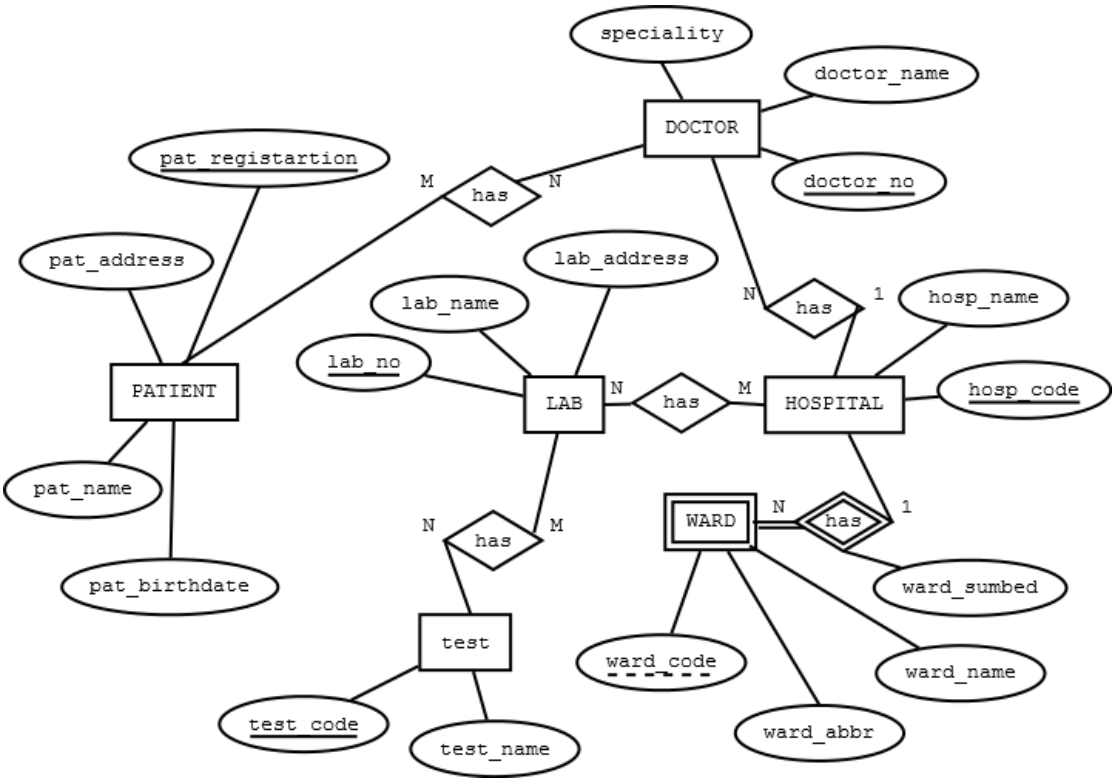
Παραθέτουμε περιορισμούς.

Κάθε νοσοκομείο έχει κλινικές. Κάθε κλινική έχει έναν μοναδικό κωδικό για όλα τα νοσοκομεία, π.χ., (3, INTENSIVE CARE). Κάθε γιατρός ανήκει σε ένα νοσοκομείο και κάθε νοσοκομείο έχει πολλούς γιατρούς. Κάθε ασθενής μπορεί να παρακολουθείται από πολλούς γιατρούς και ένας γιατρός να παρακολουθεί πολλούς ασθενείς. Κάθε νοσοκομείο μπορεί να συνεργάζεται με πολλά εργαστήρια και κάθε εργαστήριο με πολλά νοσοκομεία. Σε ένα εργαστήριο γίνονται πολλές εξετάσεις και μία εξέταση μπορεί να γίνει σε πολλά εργαστήρια.

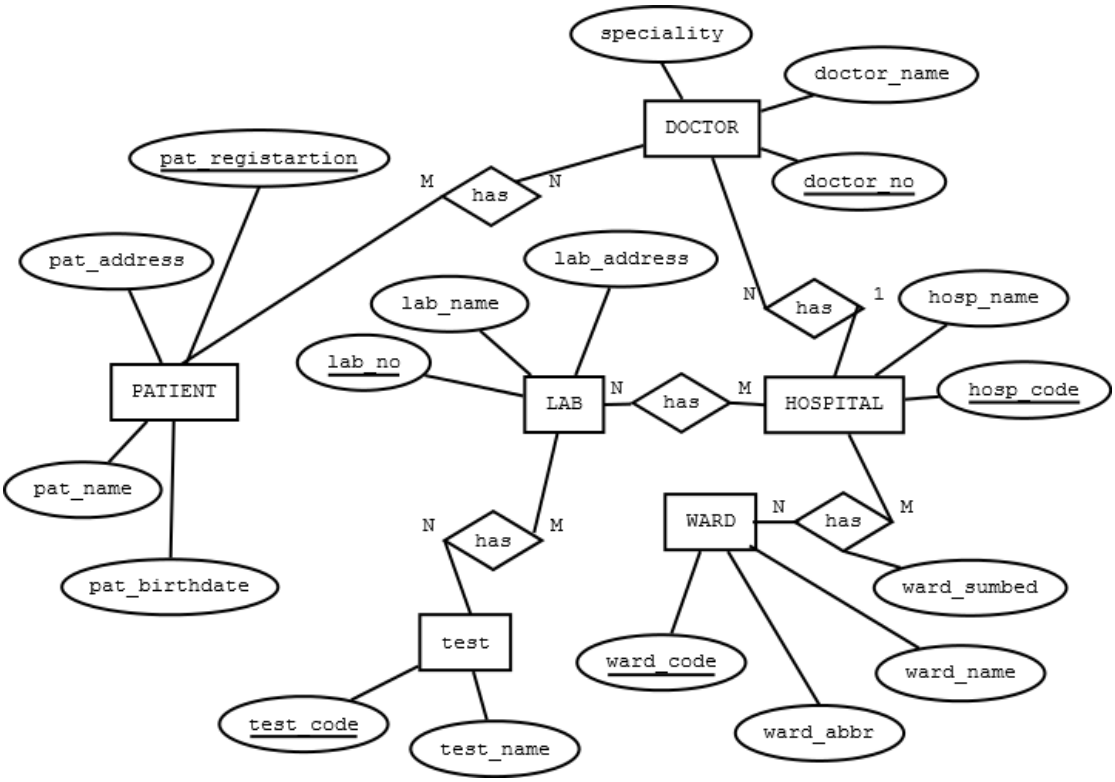
Κατασκευάστε μοντέλο οντοτήτων συσχετίσεων για τη βάση δεδομένων της εταιρείας.

Υποδείξεις

Στην Εικόνα 5.16 βλέπουμε μοντέλο οντοτήτων-συσχετίσεων για το σύστημα νοσοκομείων στο οποίο περιλαμβάνεται εξαρτώμενη οντότητα WARD ενώ στην εικόνα 5.17 βλέπουμε μοντέλο οντοτήτων-συσχετίσεων για το σύστημα νοσοκομείων στο οποίο περιλαμβάνεται ισχυρή οντότητα WARD



Εικόνα 5.16 Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων στο οποίο περιλαμβάνεται εξαρτώμενη οντότητα WARD.



Εικόνα 5.17 Μοντέλο Οντοτήτων-Συσχετίσεων για το σύστημα νοσοκομείων στο οποίο περιλαμβάνεται ισχυρή οντότητα WARD.

Θέμα 2 Διαχείριση παρακολούθησης ασθενούς από ιατρούς

Για την εταιρεία Hospital Information System H.I.S. που είδαμε στο θέμα 1 θεωρούμε ότι οι ασθενείς μπορούν να κάνουν εξετάσεις σε διάφορα εργαστήρια που μπορεί να τις έχουν γράψει διάφοροι γιατροί. Για κάθε εργαστηριακή εξέταση (Test) η οποία γίνεται σε κάποιο εργαστήριο καταχωρούμε και τα εξής στοιχεία: Result=αποτέλεσμα εργαστηριακής εξέτασης, Tdate=ημερομηνία εργαστηριακής εξέτασης. Να γράψετε την Τρίτη κανονική μορφή για τη βάση δεδομένων της εταιρείας.

Παραθέτουμε την τρίτη κανονική μορφή η οποία αντιστοιχεί στην Εικόνα 5.16.

PATIENT(pat_registration, pat_name, pat_birthdate, pat_address)

DOCTOR(doctor_no, doctor_name, speciality, hosp_code)

HOSPITAL(hosp_cod, hosp_name)

LAB(lab_no, lab_name, lab_address)

TEST(test_code, test_name)

WARD(hosp_code, ward_code, ward_name, ward_sumbed)

PAT_DOCTOR(accno, par_registration, doctor_no, from, to)

HOSP_LAB(hosp_code, lab_no)

LAB_TEST(lab_no, test_no)

Γράψτε κύρια και ξένα κλειδία. Γράψτε την τρίτη κανονική μορφή η οποία αντιστοιχεί στην Εικόνα 5.17.

Θέμα 3

Έστω ότι η βάση δεδομένων της εταιρείας H.I.S. περιλαμβάνει στοιχεία για: Νοσοκομεία (hospital), κλινικές (ward), προσωπικό (staff). Ακολουθούν οι στήλες των πινάκων: Emp_no=κωδικός υπαλλήλου, Emp_name=ονοματεπώνυμο, Emp_salary=μισθός, Hosp_code=κωδικός Νοσοκομείου, Hosp_name=Ονομασία, Ward_code=κωδικός κλινική, Ward_name=ονομασία, Ward_abbr=συντομογραφία κλινικής, Ward_sumbed= αριθμός κρεβατιών. Κάθε νοσοκομείο έχει κλινικές. Κάθε κλινική έχει έναν μοναδικό κωδικό, π.χ., (3, INTENSIVE CARE). Κάθε υπάλληλος ανήκει στην κλινική ενός νοσοκομείου.

```
SELECT * FROM STAFF;
```

HOSP_CODE	WARD_CODE	EMP_NO	EMP_NAME	EMP_SALARY
13	3	3754	DATE.	1740
13	6	3106	HUGO	1350
18	3	6357	CODD	1700

```
SELECT * FROM HOSPITAL;
```

HOSP_CODE	HOSP_NAME
13	CENTRAL
18	GENERAL

```
SELECT * FROM WARD_ABBR;
```

WARD_CODE	WARD_NAME	WARD_ABBR
3	INTENSIVE CARE	INTE
6	PSYCHIATRIC	PSYC

```
SELECT * FROM HOSP_WARD;
```

HOSP_CODE	WARD_CODE	WARD_SUMBEDW
13	3	21
13	6	67
18	3	45

1) Γράψτε τους ορισμούς των πινάκων με κύρια και ξένα κλειδιά.

```
DROP DATABASE HIS_THEORY_EXAM;
CREATE DATABASE HIS_THEORY_EXAM;
USE HIS_THEORY_EXAM;

CREATE TABLE WARD_ABBR (WARD_CODE INT(3) NOT NULL,WARD_NAME VARCHAR(20),
 WARD_ABBR CHAR(4),PRIMARY KEY (WARD_CODE));

CREATE TABLE HOSPITAL(HOSP_CODE INT(3) NOT NULL,
 HOSP_NAME VARCHAR(15),PRIMARY KEY (HOSP_CODE));

CREATE TABLE HOSP_WARD (HOSP_CODE INT(3) NOT NULL,
 WARD_CODE INT(3) NOT NULL,
 WARD_SUMBEDW INT(3),PRIMARY KEY (WARD_CODE, HOSP_CODE),
FOREIGN KEY(HOSP_CODE) REFERENCES HOSPITAL(HOSP_CODE),
FOREIGN KEY(WARD_CODE) REFERENCES WARD_ABBR(WARD_CODE));

CREATE TABLE STAFF (HOSP_CODE INT(3) NOT NULL,
 WARD_CODE INT(3) NOT NULL, EMP_NO INT(4) NOT NULL,
 EMP_NAME VARCHAR(20),EMP_SALARY INT(5), PRIMARY KEY (EMP_NO), FOREIGN
KEY(WARD_CODE, HOSP_CODE) REFERENCES HOSP_WARD(WARD_CODE, HOSP_CODE));
```

2) Εισάγετε όλα τα στοιχεία των υπαλλήλων

```
INSERT INTO WARD_ABBR VALUES (1,'RECOVERY', 'REC');
INSERT INTO WARD_ABBR VALUES (2,'MATERNITY', 'MAT');
INSERT INTO WARD_ABBR VALUES (3,'INTENSIVE CARE', 'INT');
INSERT INTO WARD_ABBR VALUES (4,'CARDIAC', 'CAR');
INSERT INTO WARD_ABBR VALUES (5,'RADIOTHERAPY', 'RAD');
INSERT INTO WARD_ABBR VALUES (6,'PSYCHIATRIC', 'PSY');

INSERT INTO HOSPITAL VALUES (22,'DOCTORS');
INSERT INTO HOSPITAL VALUES(13,'CENTRAL');
INSERT INTO HOSPITAL VALUES(45,'CHILDRENS');
INSERT INTO HOSPITAL VALUES(18,'GENERAL');

INSERT INTO HOSP_WARD VALUES (13,3, 21);
INSERT INTO HOSP_WARD VALUES (13,6, 67);
INSERT INTO HOSP_WARD VALUES (18,3, 10);
```

```
INSERT INTO HOSP_WARD VALUES (18,4, 53);
INSERT INTO HOSP_WARD VALUES (22,1, 10);
INSERT INTO HOSP_WARD VALUES (22,6, 118);
INSERT INTO HOSP_WARD VALUES (22,2, 34);
INSERT INTO HOSP_WARD VALUES (45,4, 55);
INSERT INTO HOSP_WARD VALUES (45,1, 13);
INSERT INTO HOSP_WARD VALUES (45,2,24);

INSERT INTO STAFF VALUES (22,6,1009,'HARRISON', 1850);
INSERT INTO STAFF VALUES (13,6,3754,'DATE', 1740);
INSERT INTO STAFF VALUES (22,6,8422,'BELL', 1260);
INSERT INTO STAFF VALUES (22,2,9901,'NELSON', 1700);
INSERT INTO STAFF VALUES (45,4,1280,'ANDERSON', 1700);
INSERT INTO STAFF VALUES (22,1,6065,'RANDALL', 2020);
INSERT INTO STAFF VALUES (13,6,3106,'HUGO', 1350);
INSERT INTO STAFF VALUES (18,4,6357,'KANTOR', 1830);
INSERT INTO STAFF VALUES (22,1,7379,'CAMERON', 1630);
INSERT INTO STAFF VALUES (45,1,8526,'FRANK', 1940);

SELECT * FROM WARD_ABBR;
SELECT * FROM HOSPITAL;
SELECT * FROM HOSP_WARD;
SELECT * FROM STAFF;
```

3) Δείξτε κωδικό, όνομα, κωδικό κλινικής και κλινική των υπαλλήλων που εργάζονται στα νοσοκομεία με κωδικό (hosp_code) 13, 14, 15, 16, 17, 18, ταξινομημένους ανά κωδικό νοσοκομείου και αλφαβητικά.

```
SELECT EMP_NO, EMP_NAME, STAFF.WARD_CODE, WARD_NAME, HOSP_CODE
FROM STAFF, WARD_ABBR
WHERE STAFF.WARD_CODE=WARD_ABBR.WARD_CODE
AND HOSP_CODE BETWEEN 13 AND 18
ORDER BY HOSP_CODE, EMP_NAME;
```

4) Δείξτε υπαλλήλους (emp_no, emp_name, ward_code, ward_name) που το όνομα τους αρχίζει από B ή C (τουλάχιστον με 2 τρόπους).

```
SELECT EMP_NO, EMP_NAME, STAFF.WARD_CODE, WARD_NAME
FROM STAFF, WARD_ABBR
WHERE STAFF.WARD_CODE=WARD_ABBR.WARD_CODE
AND (substr(EMP_NAME,1,1) >'B' and substr(EMP_NAME,1,1) <'C');

SELECT EMP_NO, EMP_NAME, STAFF.WARD_CODE, WARD_NAME
FROM STAFF, WARD_ABBR
WHERE STAFF.WARD_CODE=WARD_ABBR.WARD_CODE
AND substr(EMP_NAME,1,1) between 'B' and 'C';

SELECT EMP_NO, EMP_NAME, STAFF.WARD_CODE, WARD_NAME
FROM STAFF, WARD_ABBR
WHERE STAFF.WARD_CODE=WARD_ABBR.WARD_CODE
AND (EMP_NAME like 'B%'or EMP_NAME like 'C%');

SELECT EMP_NO, EMP_NAME, STAFF.WARD_CODE, WARD_NAME
FROM STAFF, WARD_ABBR
```

```
WHERE STAFF.WARD_CODE=WARD_ABBR.WARD_CODE
AND (substr(EMP_NAME,1,1)='B'or substr(EMP_NAME,1,1)='C');
```

Προσοχή! Οι επόμενες αναζητήσεις είναι λανθασμένες.

```
SELECT EMP_NO, EMP_NAME, STAFF.WARD_CODE, WARD_NAME
FROM STAFF, WARD_ABBR
WHERE STAFF.WARD_CODE=WARD_ABBR.WARD_CODE
AND EMP_NAME like 'B%'or EMP_NAME like 'C%';

SELECT EMP_NO, EMP_NAME, STAFF.WARD_CODE, WARD_NAME
FROM STAFF, WARD_ABBR
WHERE STAFF.WARD_CODE=WARD_ABBR.WARD_CODE
AND substr(EMP_NAME,1,1)='B'or substr(EMP_NAME,1,1)='C';
```

5) Δείξτε κάθε νοσοκομείο με τις κλινικές του

```
select hosp_ward.hosp_code, hosp_name, hosp_ward.ward_code, ward_name
from hospital, hosp_ward, ward_abbr
where hospital.hosp_code=hosp_ward.hosp_code
AND ward_abbr.ward_code=hosp_ward.ward_code;
```

6) Δείξτε υπαλλήλους που ανήκουν στην κλινική INTENSIVE CARE ή PSYCHIATRIC. Θα δείχνετε κωδικό, όνομα, όνομα νοσοκομείου και όνομα κλινικής (emp_no, emp_name, hosp_name, ward_name).

```
select emp_no, emp_name, hosp_name, ward_name
from hospital, hosp_ward, ward_abbr, staff
where hospital.hosp_code=hosp_ward.hosp_code
AND ward_abbr.ward_code=staff.ward_code
AND staff.hosp_code=hospital.hosp_code
AND ward_abbr.ward_code=hosp_ward.ward_code
AND ward_name IN ('INTENSIVE CARE', 'PSYCHIATRIC');
```

7) Δείξτε μέσο μισθό, μέγιστο μισθό, ελάχιστο μισθό, διαφορά μέγιστου και ελάχιστου μισθού ανά κωδικό νοσοκομείου (hosp_code). Ταξινομήστε τα αποτελέσματα ανά κωδικό νοσοκομείου

```
select hosp_code,avg(emp_salary), max(emp_salary), min(emp_salary),
max(emp_salary)-min(emp_salary)
from staff
group by hosp_code
order by hosp_code;
```

8) Δείξτε μέσο μισθό, μέγιστο μισθό, ελάχιστο μισθό, διαφορά μέγιστου και ελάχιστου μισθού ανά κωδικό νοσοκομείου (hosp_code) στην περίπτωση που η διαφορά μέγιστου και ελάχιστου μισθού είναι μεγαλύτερη ή ίση του 500 και μικρότερη ή ίση του 1500. Ταξινομήστε τα αποτελέσματα ανά κωδικό νοσοκομείου

```
select hosp_code,avg(emp_salary), max(emp_salary), min(emp_salary),
max(emp_salary)-min(emp_salary)
from staff
group by hosp_code
having max(emp_salary)-min(emp_salary)BETWEEN 500 AND 1500
order by hosp_code;
```

9) Γράψτε SELECT και δείξτε για τους υπάλληλους που εργάζονται στο νοσοκομείο με κωδικό 13: a) διαφορά μέγιστου και ελάχιστου μισθού υπαλλήλων, b) μέσο μισθό υπαλλήλων, c) άθροισμα μισθών όλων των υπαλλήλων, d) πόσοι υπάλληλοι έχουν μισθό

```
SELECT MAX(emp_salary)-MIN(emp_salary), AVG(emp_salary), SUM(emp_salary),
COUNT(emp_salary)
FROM staff
WHERE hosp_code=13;
```

10) Γράψτε SELECT και δείξτε μόνο για τα νοσοκομεία με κωδικό hospital_code ίσο με 18 ή 22 τα στοιχεία των υπαλλήλων τους (hosp_code, hosp_name, emp_no, emp_name, emp_salary)

```
select hospital.hosp_code, hosp_name, emp_no, emp_name, emp_salary
from hospital, staff
where (staff.hosp_code = 18 OR staff.hosp_code = 22)
AND hospital.hosp_code=staff.hosp_code;
```

11) Γράψτε SELECT και δείξτε μόνο για τα νοσοκομεία με κωδικό hospital_code ίσο με 18 ή 22 τα εξής: ward_code, μέσο μισθό για κάθε ward_code του νοσοκομείου -όταν είναι μεγαλύτερος ή ίσος του 1700-, αριθμό υπαλλήλων

```
select ward_code,avg(emp_salary),count(*)
from staff
where hosp_code = 18 OR hosp_code = 22
group by ward_code
having avg(emp_salary) >= 1700 order by ward_code ;
```

12) Γράψτε δηλώσεις διαγραφής των πινάκων της βάσης και στη συνέχεια δήλωση διαγραφή της βάσης.

```
DROP TABLE STAFF;
DROP TABLE HOSP_WARD;
DROP TABLE HOSPITAL;
DROP TABLE WARD_ABBR;
DROP DATABASE HIS_THEORY_EXAM;
```

Θέμα 4

Η εταιρεία Hospital Information System H.I.S. υλοποιεί βάση δεδομένων που περιλαμβάνει στοιχεία για: Νοσοκομεία (hospital), κλινικές (ward), προσωπικό (staff), ασθενείς (patient). Να οι στήλες των πινάκων (δεν είναι στην 3NF): Pat_registration=κωδικός ασθενή, Pat_name=ονοματεπώνυμο, Pat_address= διεύθυνση, Pat_birthdate=ημερομηνία γέννησης, Emp_no=κωδικός υπαλλήλου, Emp_name=ονοματεπώνυμο, Emp_salary=μισθός, Hosp_code=κωδικός Νοσοκομείου, Hosp_name=Ονομασία, Ward_code=κωδικός κλινική, Ward_name=ονομασία, Ward_sumbed= αριθμός κρεβατιών.

Κάθε νοσοκομείο έχει κλινικές. Κάθε κλινική έχει έναν μοναδικό κωδικό, π.χ., (3, INTENSIVE CARE). Κάθε υπάλληλος ανήκει στην κλινική ενός νοσοκομείου.

```
SELECT * FROM PATIENT;
```

PAT_REGISTRATION	PAT_NAME	PAT_ADDRESS	PAT_BIRTHDATE
64823	FRANK	11 MASS	03/05/90

```
SELECT * FROM STAFF;
```

HOSP_CODE	WARD_CODE	EMP_NO	EMP_NAME	EMP_SALARY
13	3	3754	DATE.	17400
13	6	3106	HUGO	13500

```
SELECT * FROM HOSPITAL;
```

HOSP_CODE	HOSP_NAME
13	CENTRAL

```
SELECT * FROM WARD;
```

HOSP_CODE	WARD_CODE	WARD_NAME	WARD_SUMBEDW
13	3	INTENSIVE CARE	21
13	6	PSYCHIATRIC	67

1) Συμπληρώστε τους ορισμούς των πινάκων με ξένα κλειδιά. Γράψτε ποιά είναι η σωστή σειρά δημιουργίας των πινάκων.

```
CREATE TABLE HOSPITAL(HOSP_CODE INT(2) NOT NULL,
 HOSP_NAME VARCHAR(15),PRIMARY KEY (HOSP_CODE)

CREATE TABLE PATIENT(PAT_REGISTRATION INT(5) NOT NULL,
 PAT_NAME VARCHAR(20), PAT_ADDRESS VARCHAR(15),
 PAT_BIRTHDATE DATE, PRIMARY KEY (PAT_REGISTRATION)

CREATE TABLE WARD (HOSP_CODE INT(2) NOT NULL,
 WARD_CODE INT(1) NOT NULL,WARD_NAME VARCHAR(20),
 WARD_SUMBEDW INT(3),PRIMARY KEY (WARD_CODE, HOSP_CODE)

CREATE TABLE STAFF (HOSP_CODE INT(2) NOT NULL,
 WARD_CODE INT(1) NOT NULL, EMP_NO INT(4) NOT NULL,
 EMP_NAME VARCHAR(20),EMP_SALARY INT(5), PRIMARY KEY (EMP_NO)
```

Ακολουθεί λύση

```
DROP TABLE STAFF;
DROP TABLE WARD;
DROP TABLE HOSPITAL;
DROP TABLE PATIENT;
DROP DATABASE HIS_EXAM;
CREATE DATABASE HIS_EXAM;
USE HIS_EXAM;

CREATE TABLE PATIENT(PAT_REGISTRATION INT(5) NOT NULL,
 PAT_NAME VARCHAR(20), PAT_ADDRESS VARCHAR(15),
 PAT_BIRTHDATE DATE, PRIMARY KEY (PAT_REGISTRATION));

CREATE TABLE HOSPITAL(HOSP_CODE INT(2) NOT NULL,
 HOSP_NAME VARCHAR(15),PRIMARY KEY (HOSP_CODE));

CREATE TABLE WARD (HOSP_CODE INT(2) NOT NULL,
 WARD_CODE INT(1) NOT NULL,WARD_NAME VARCHAR(20),
 WARD_SUMBEDW INT(3),PRIMARY KEY (WARD_CODE, HOSP_CODE),
FOREIGN KEY(HOSP_CODE) REFERENCES HOSPITAL(HOSP_CODE));
```

```
CREATE TABLE STAFF (HOSP_CODE INT(2) NOT NULL,
 WARD_CODE INT(1) NOT NULL, EMP_NO INT(4) NOT NULL,
 EMP_NAME VARCHAR(20),EMP_SALARY INT(5),
 PRIMARY KEY (EMP_NO),
 FOREIGN KEY(WARD_CODE, HOSP_CODE)
REFERENCES WARD(WARD_CODE, HOSP_CODE));
```

2) Εισάγετε τις γραμμές στους πίνακες (άσκηση)

3) Δείξτε κωδικό, όνομα, διεύθυνση και ημερομηνία γέννησης των ασθενών, που γεννήθηκαν από το 1989 έως και το 2005, ταξινομημένους ανά ημερομηνία γέννησης και αλφαβητικά.

```
SELECT PAT_REGISTRATION, PAT_NAME, PAT_ADDRESS, PAT_BIRTHDATE
FROM PATIENT
WHERE PAT_BIRTHDATE BETWEEN '01/01/1989' AND '31/12/2005'
ORDER BY PAT_BIRTHDATE, PAT_NAME;
```

4) Δείξτε ασθενείς που το όνομα τους αρχίζει από R ή S (τουλάχιστον με 2 τρόπους).

```
select pat_name from patient
where substr(pat_name,1,1) >'Q' and substr(pat_name,1,1) <'T';
select pat_name from patient
where substr(pat_name,1,1) between 'Q' and 'T';
select pat_name from patient
where pat_name like 'R%'
or pat_name like 'S%';
select pat_name from patient
where substr(pat_name,1,1)='R'
or substr(pat_name,1,1)='S';
```

5) Δείξτε υπαλλήλους με κωδικό (emp_no) που δεν ανήκει στο διάστημα [1000, 4500]. Θα δείχνετε κωδικό, όνομα, όνομα νοσοκομείου και όνομα κλινικής.

```
select emp_no, emp_name, hosp_name, ward_name
from hospital, ward, staff
where hospital.hosp_code=ward.hosp_code
AND ward.ward_code= staff.ward_code
AND staff.hosp_code=hospital.hosp_code
AND emp_no not between 10000 and 15000;
select emp_no, emp_name, hosp_name, ward_name
from hospital, ward, staff
where hospital.hosp_code=ward.hosp_code
AND ward.ward_code= staff.ward_code
AND staff.hosp_code=hospital.hosp_code
AND (emp_no < 1000 or emp_no > 1500);
```

Η παρακάτω δήλωση είναι λανθασμένη

```
select emp_no, emp_name, hosp_name, ward_name
from hospital, ward, staff
where hospital.hosp_code=ward.hosp_code
AND ward.ward_code= staff.ward_code
AND emp_no not between 10000 and 15000;
```

6) Δείξτε μέσο μισθό ανά κωδικό νοσοκομείου (hosp_code) στην περίπτωση που ο μέσος μισθός είναι μεγαλύτερος του 1500. Ταξινομήστε τα αποτελέσματα ανά κωδικό νοσοκομείου

```
select hosp_code,avg(emp_salary)
from staff
group by hosp_code
having avg(emp_salary) > 1500
order by hosp_code;
```

7) Γράψτε SELECT και δείξτε: a) άθροισμα μισθών όλων των υπαλλήλων διαιρούμενο με τον αριθμό υπαλλήλων που έχουν μισθό, b) μέσο μισθό υπαλλήλων, c) άθροισμα μισθών όλων των υπαλλήλων διαιρούμενο με τον αριθμό όλων των υπαλλήλων. Διαφέρουν οι 3 αριθμοί;

```
select AVG(emp_salary), SUM(emp_salary)/COUNT(emp_salary),
SUM(emp_salary)/COUNT(*) from staff;
```

Ο τρίτος αριθμός μπορεί να διαφέρει.

8) Γράψτε SELECT και δείξτε μόνο για hospital_code=22 τα εξής: ward_code, μέσο μισθό για κάθε ward_code του νοσοκομείου -όταν είναι μεγαλύτερος ή ίσος του 1700-, αριθμό υπαλλήλων

```
select ward_code,avg(emp_salary),count(*)
from staff
where hosp_code = 22
group by ward_code
having avg(emp_salary) >= 1700 order by ward_code ;
```

9) Γράψτε μία δήλωση SELECT και δείξτε κωδικό νοσοκομείου (hospital_code), κωδικό κλινικής (ward_code) για κάθε κλινική με περισσότερα από ένα μέλη προσωπικού.

```
select hosp_code,ward_code,ward_name
from ward w
where 1 < ( select count(*)
from staff s
where s.ward_code = w.ward_code
and s.hosp_code = w.hosp_code );
```

10) Γράψτε μία δήλωση SELECT και δείξτε στοιχεία υπαλλήλων (emp_no, emp_name, emp_salary, hosp_code, ward_code) που κερδίζουν περισσότερα από το μέσο όρο μισθού του νοσοκομείου τους.

```
select s.emp_no,s.emp_name, s.hosp_code, s.emp_salary
from staff s
where emp_salary > (select avg(emp_salary)
from staff x
where x.hosp_code = s.hosp_code);
```

5.6 Σχεδίαση της διεπαφής του χρήστη για το Πληροφοριακό Σύστημα Βάσης Δεδομένων Νοσοκομείων

Στη συνέχεια υποτίθεται ότι μέσω της εφαρμογής που θέλουμε να σχεδιάσουμε θα διεκπεραιώνονται οι βασικές λειτουργίες του συστήματος Νοσοκομείων. Συνοπτικά, ο χρήστης θα μπορεί να βλέπει τα στοιχεία κάθε νοσοκομείου και των κλινικών του, των εργαστηρίων με τα οποία συνεργάζεται κάθε νοσοκομείο, τους

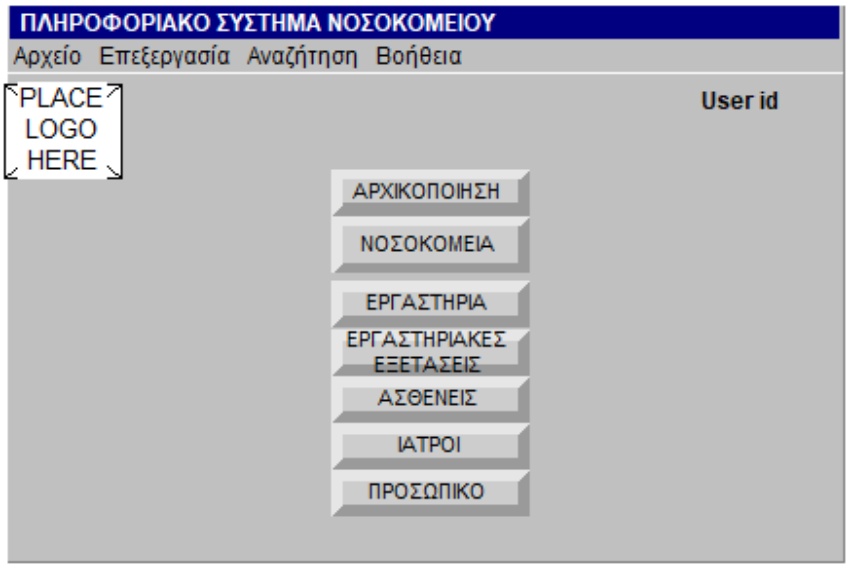
γιατρούς και το νοσηλευτικό προσωπικό, τους ασθενείς, εισαγωγές ασθενών και διεξαγωγή εξετάσεων και βέβαια θα πρέπει να μπορεί να διαχειρίζεται τα στοιχεία αυτά (Αναζήτηση-Εισαγωγή-Διαγραφή-Τροποποίηση στοιχείων).

5.6.1 Σχεδίαση της εφαρμογής

Πριν αρχίσουμε να υλοποιούμε την εφαρμογή μας πρέπει να σχεδιάσουμε προσεκτικά τις φόρμες διαχείρισης της βάσης δεδομένων. Οι φόρμες στην ενότητα αυτή σχεδιάστηκαν στο προϊόν Microsoft Vision. Σε μία πραγματική εφαρμογή οι φόρμες αυτές πρέπει να συζητηθούν αναλυτικά και με τους χρήστες της εφαρμογής. Στο παρόν εξετάστε αναλυτικά τις φόρμες, διαπιστώστε την τυποποίηση της διεπαφής και αντιστοιχίστε κάθε φόρμα με πίνακα ή πίνακες της βάσης.

5.6.2 Κεντρική Φόρμα

Κατά την εισαγωγή μας στην εφαρμογή εμφανίζεται η φόρμα MAIN_FRM με τις κεντρικές λειτουργίες. Παρατηρήστε το «σχηματικό» μενού, που βέβαια στην κανονική φόρμα θα αντικατασταθεί, το λογότυπο, τον τίτλο της εφαρμογής και το πεδίο που θα εμφανίζει τον κωδικό του χρήστη. Στη φόρμα αυτή, και σε όλες τις φόρμες αυτού του τύπου μπορούμε να χρησιμοποιήσουμε και εργαλειοθήκη.



Εικόνα 5.18 Κεντρική φόρμα

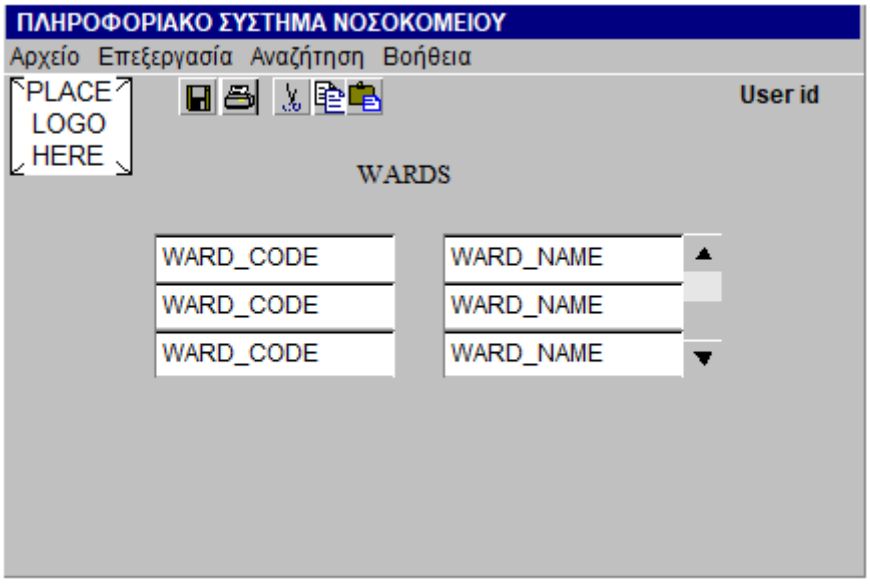
5.6.3 Αρχικοποίηση

Με την επιλογή «**Αρχικοποίηση»,** ο χρήστης οδηγείται σε μια νέα οθόνη (φόρμα) INITIALIZE_FRM στην οποία υπάρχει ένα νέο υπο-μενού με επιλογές αρχικοποίησης κάποιων πινάκων. Εξετάσεων. Οι πίνακες αυτοί ονομάζονται και πίνακες (αρχεία) καθιερωμένου περιεχομένου (authority files) επειδή χρησιμοποιούνται σε όλα τα νοσοκομεία του περιγραφόμενου συστήματος. Για να μπορέσουμε να χρησιμοποιήσουμε την εφαρμογή στο σύνολο της, θα πρέπει αρχικά να δώσουμε τιμές στους πίνακες συντομογραφίας κλινικών και εξετάσεων.

Εικόνα 5.19 Αρχικοποίηση κλινικών και εξετάσεων (πίνακες καθιερωμένου περιεχομένου)

5.6.4 Αρχικοποίηση κλινικών

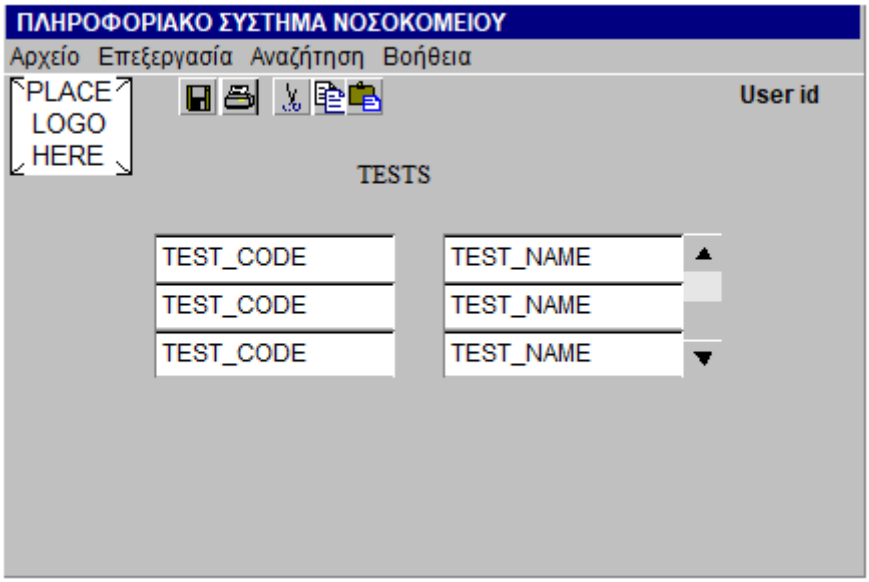
Στην φόρμα WARDABBR_FRM εμφανίζονται οι κωδικοί των κλινικών με το όνομα τους. Είναι μία φόρμα, αρκετά, διαφορετική από τις προηγούμενες. Παρατηρήστε το «σχηματικό» μενού, που βέβαια στην κανονική φόρμα θα αντικατασταθεί, το λογότυπο, τον τίτλο της εφαρμογής και το πεδίο που θα εμφανίζει τον κωδικό του χρήστη. Στη φόρμα αυτή, και σε όλες τις φόρμες αυτού του τύπου μπορούμε να χρησιμοποιήσουμε και εργαλειοθήκη. Η εργαλειοθήκη της εικόνας είναι «σχηματική» και θα πρέπει να αντικατασταθεί στην κανονική φόρμα. Η εισαγωγή και η ενημέρωση στοιχείων του πίνακα πρέπει να γίνεται σε κεντρικό επίπεδο, για όλο το σύστημα.



Εικόνα 5.20 Αρχικοποίηση κλινικών

5.6.5 Αρχικοποίηση εργαστηριακών εξετάσεων

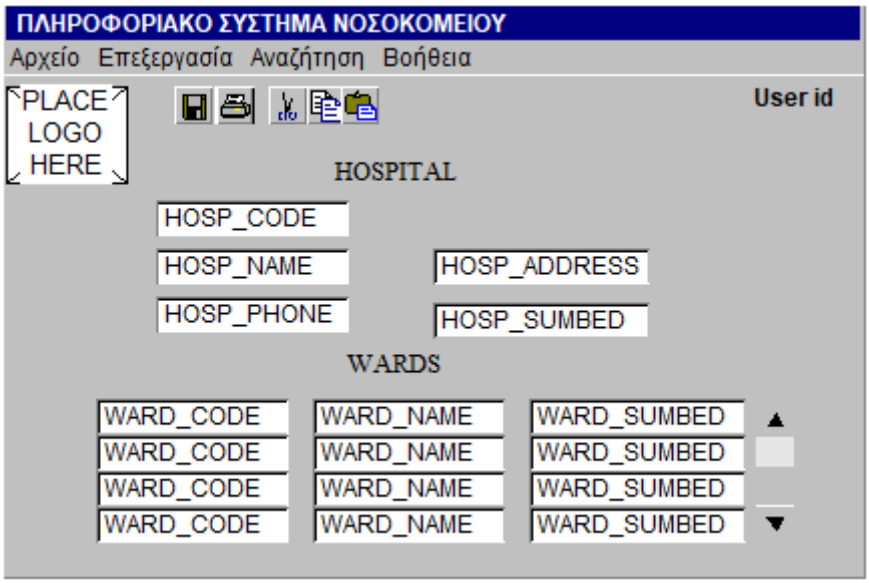
Στην φόρμα TESTABBR_FRM εμφανίζονται οι κωδικοί των εξετάσεων με το όνομα τους. Η εισαγωγή και η ενημέρωση στοιχείων του πίνακα πρέπει να γίνεται σε κεντρικό επίπεδο, για όλο το σύστημα.



Εικόνα 5.21Αρχικοποίηση εργαστηριακών εξετάσεων

5.6.6 Νοσοκομείο και στοιχεία κλινικών του

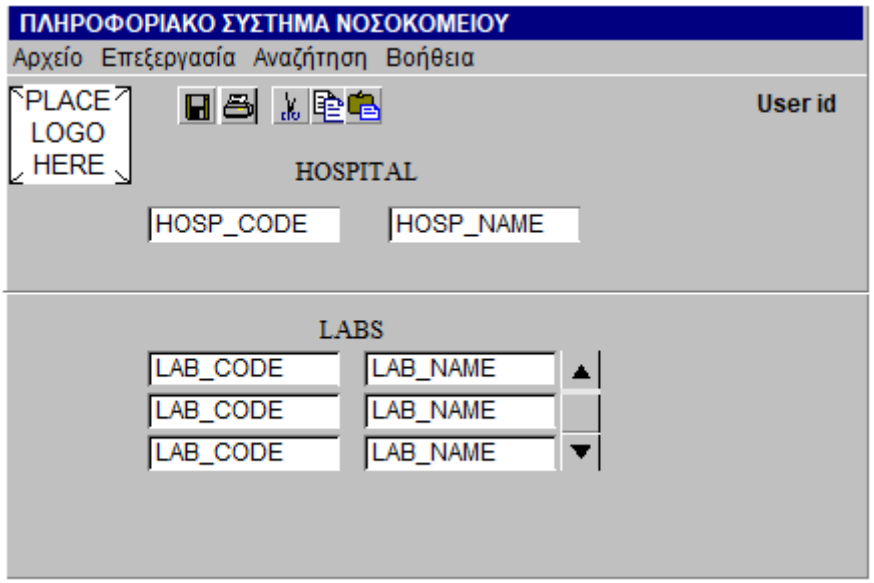
Το νέο στοιχείο στη φόρμα είναι ο τύπος της, τύπος master/detail. Πιο συγκεκριμένα, στην τύπου master/detail φόρμα HOSPWARD_FRM ο χρήστης μπορεί να βλέπει τα στοιχεία (κωδικό, όνομα, διεύθυνση, τηλέφωνο, πλήθος κρεβατιών) του νοσοκομείου και τα στοιχεία (κωδικό, όνομα, πλήθος κρεβατιών) των κλινικών του. Για να χρησιμοποιήσουμε τη φόρμα για εισαγωγή στοιχείων θα πρέπει να καταχωρηθούν (σε κεντρικό επίπεδο), προηγουμένως, στοιχεία συντομογραφίας κλινικών. Κάθε Νοσοκομείο εισάγει και ενημερώνει τα δικά του στοιχεία.



Εικόνα 5.22 Στοιχεία κλινικών για κάθε νοσοκομείο

5.6.7 Νοσοκομείο και εργαστήρια

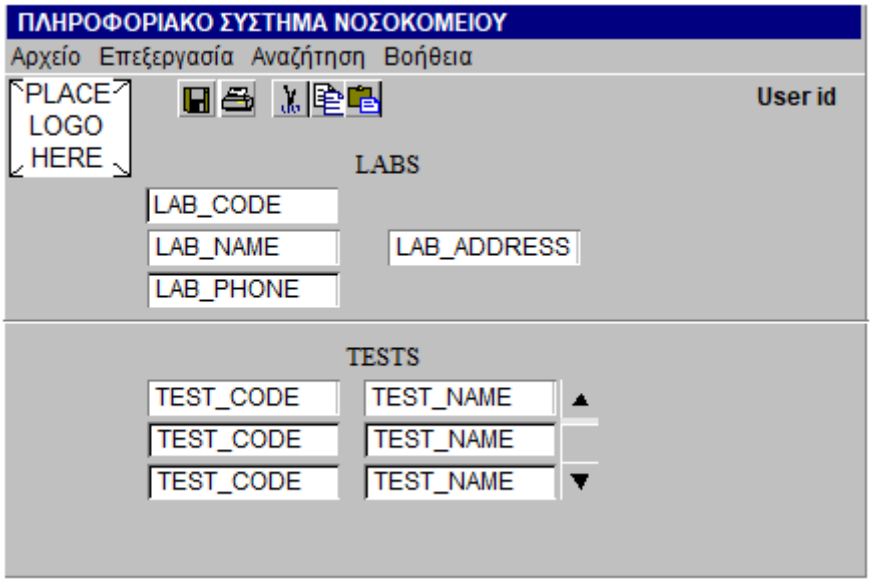
Στην τύπου master detail φόρμα HOSPLAB_FRM ο χρήστης μπορεί να βλέπει τον κωδικό και όνομα του νοσοκομείου και τα στοιχεία (κωδικό, όνομα) των εργαστηρίων του. Για να χρησιμοποιήσουμε τη φόρμα για εισαγωγή στοιχείων θα πρέπει να καταχωρηθούν, προηγουμένως, στοιχεία εργαστηρίων. Κάθε Νοσοκομείο εισάγει και ενημερώνει τα δικά του στοιχεία.



Εικόνα 5.23Νοσοκομεία και εργαστήρια για τις εξετάσεις των ασθενών

5.6.8 Εργαστηριακές Εξετάσεις

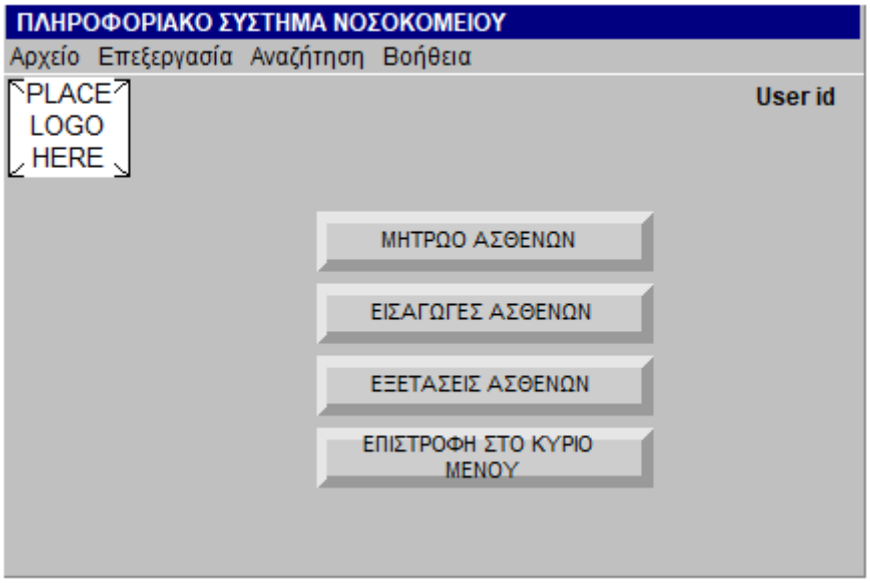
Στη φόρμα LABTEST_FRM εμφανίζονται ο κωδικός, το όνομα, η διεύθυνση και το τηλέφωνο ενός Εργαστηρίου, με τα στοιχεία (κωδικό και όνομα, περιγραφή) των εξετάσεων που μπορούν να γίνουν στο συγκεκριμένο Εργαστήριο. Η εισαγωγή και η ενημέρωση στοιχείων γίνεται από κάθε εργαστήριο.



Εικόνα 5.24 Εργαστηρισκές εξετάσεις για κάθε εργαστήριο

5.6.9 Ασθενείς

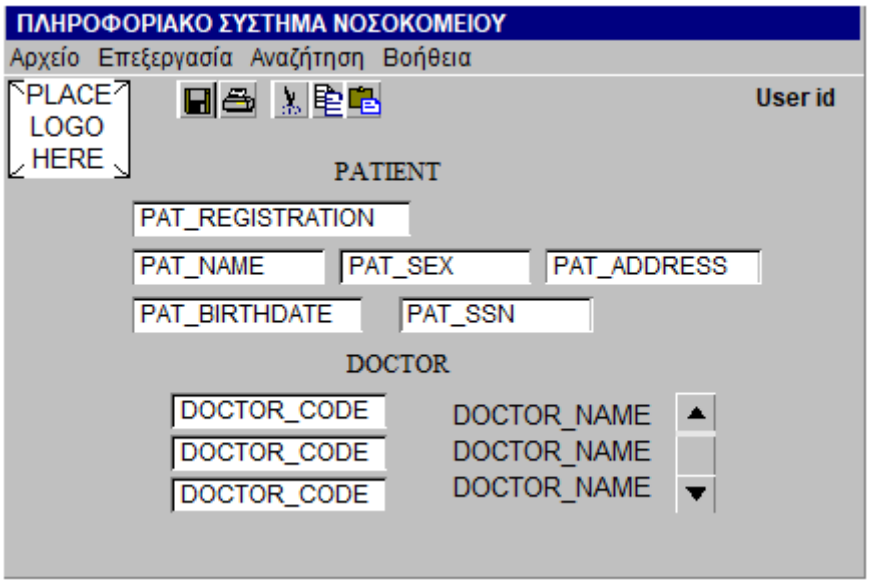
Με την επιλογή «Ασθενείς» στο αρχικό μενού, ο χρήστης οδηγείται στη φόρμα PATMENU_FRM στην οποία υπάρχει ένα νέο υπο-μενού με τις επιλογές που φαίνονται στην παρακάτω φόρμα.



Εικόνα 5.25Μενού για την εισαγωγή και τις εξετάσεις ασθενών

5.6.10 Μητρώο Ασθενών

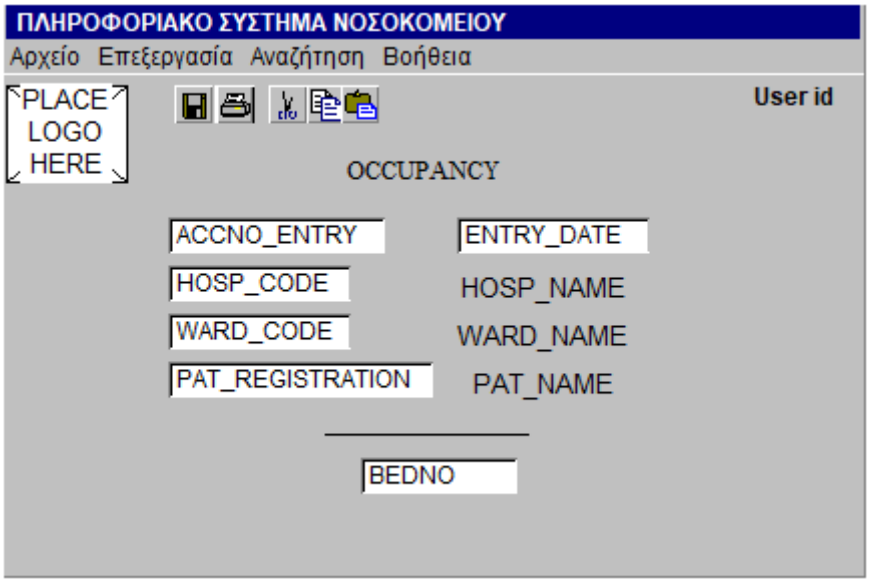
Στην τύπου master detail φόρμα PATIENT_FRM ο χρήστης μπορεί να βλέπει τα στοιχεία ενός ασθενή (κωδικό, όνομα, φύλο, ημερομηνία γέννησης, διεύθυνση, τηλέφωνο, κωδικό ασφάλισης) με τα στοιχεία (κωδικό, όνομα) των ιατρών που τους παρακολουθεί. Η φόρμα αυτή έχει ριζικές διαφορές από τις άλλες φόρμες που είδαμε μέχρι στιγμής. Το όνομα του γιατρού σχεδιάζεται διαφορετικά γιατί ο χειριστής της φόρμας θα πληκτρολογεί τον κωδικό του γιατρού και η εφαρμογή θα εμφανίζει το όνομά του ή ο χειριστής θα εμφανίζει ένα παράθυρο με τα ονόματα των γιατρών πάνω στη φόρμα, θα επιλέγει από το παράθυρο γιατρό και στη φόρμα μας θα εμφανίζεται (μετά το κλείσιμο του παραθύρου) ο κωδικός και το όνομα του γιατρού που επέλεξε ο χειριστής.



Εικόνα 5.26 Στοιχεία ασθενών και των ιατρών που τους παρακολουθούν

5.6.11 Εισαγωγές ασθενών σε κλινικές νοσοκομείου

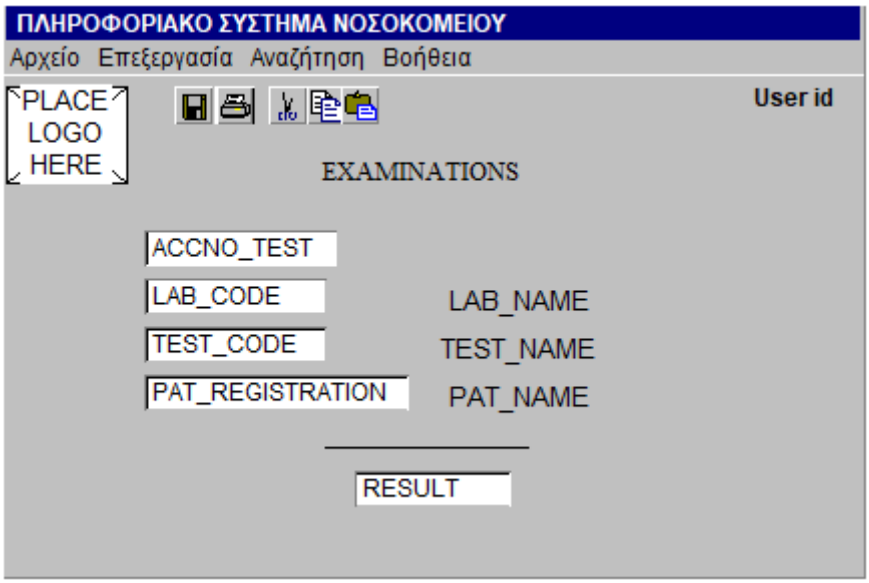
Στη φόρμα ENTRY_FRM εμφανίζεται ο κωδικός του ασθενή, ποια ημερομηνία έκανε εισαγωγή, ο αριθμός του κρεβατιού του, ο κωδικός της κλινικής και του νοσοκομείου που έχει εισαχθεί .



Εικόνα 5.27 Εισαγωγή ασθενών σε κλινική

5.6.12 Εργαστηριακές εξετάσεις ασθενούς

Η φόρμα EXAMINATION_FRM εμφανίζει τις εξετάσεις που έκανε κάποιος ασθενής. Συγκεκριμένα εμφανίζει τον κωδικό του ασθενή, τον κωδικό του εργαστηρίου όπου γίνεται η εξέταση, τον κωδικό της εξέτασης, την ημερομηνία που έγινε η εξέταση και το αποτέλεσμά της. Τη φόρμα μπορούν να δουν μόνο εξουσιοδοτημένος χειριστής και ο ιατρός του ασθενούς.

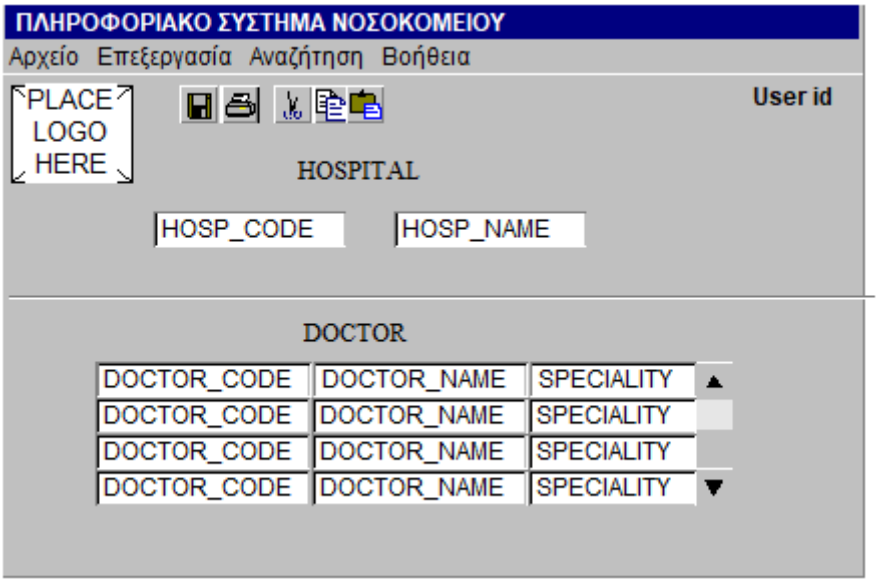


Εικόνα 5.28Εργαστηριακές εξετάσεις ασθενών

5.6.13 Νοσοκομείο και ιατροί του

Η τύπου master detail φόρμα DOCTORS_FRM εμφανίζει τους γιατρούς κάθε νοσοκομείου. Συγκεκριμένα εμφανίζεται ο κωδικός και το όνομα του νοσοκομείου και τα στοιχεία (κωδικός, όνομα, ειδικότητα) του ιατρού.

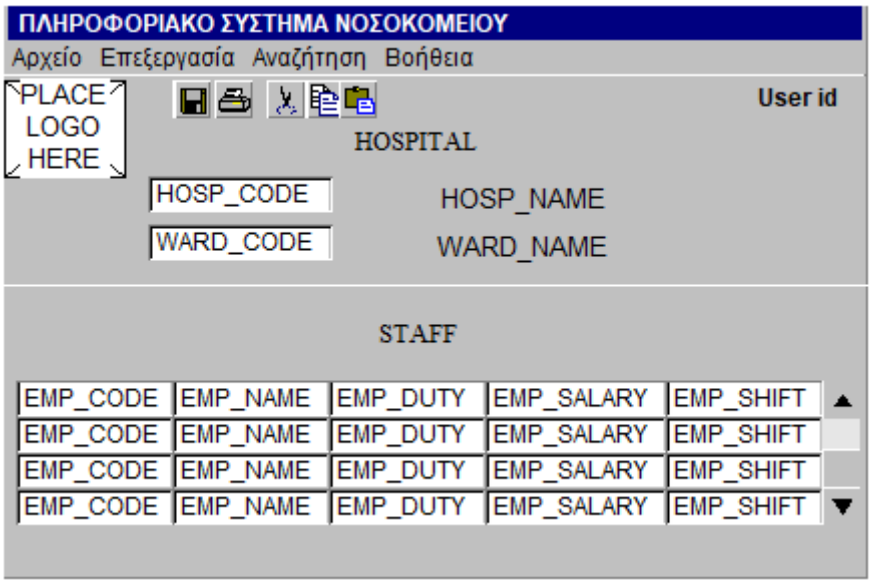
Η εισαγωγή και η ενημέρωση των στοιχείων γίνεται από κάθε Νοσοκομείο.



Εικόνα 5.29 Οι γιατροί ανά νοσοκομείο

5.6.14 Νοσοκομείο και υγειονομικό προσωπικό

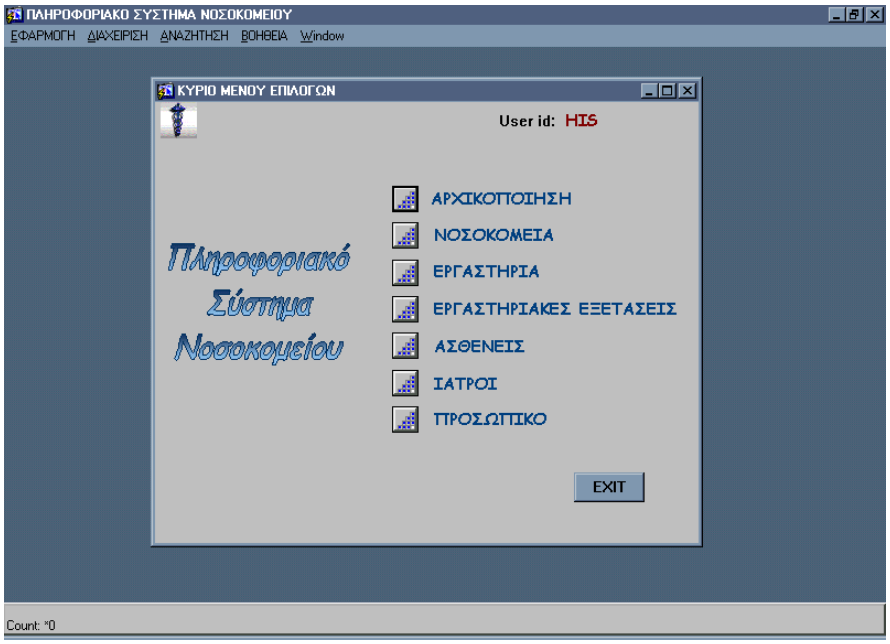
Η φόρμα STAFF_FRM εμφανίζει τα στοιχεία του προσωπικού (κωδικός, όνομα, καθήκον, βάρδια, μισθός) κάθε Κλινικής ενός νοσοκομείου. Έτσι, εμφανίζονται και ο κωδικός του νοσοκομείου, ο κωδικός της κλινικής και το όνομά της. Η εισαγωγή και η ενημέρωση στοιχείων γίνεται σε επίπεδο κλινικής Νοσοκομείου.



Εικόνα 5.30 Προσωπικό και κλινικές

5.6.15 Συζήτηση της τελική διεπαφής (INTERFACE)

Κατά την υλοποίηση της εφαρμογής γίνονται κάποιες μικρές αλλαγές και οριστικοποιούνται κάποια «μορφολογικά» στοιχεία της φόρμας. Κατά κύριο λόγο, οριστικοποιούνται και οι χειρισμοί του τελικού χρήστη για τη διαχείριση στοιχείων με τη φόρμα. Ακολουθούν παράδειγμα της **Κεντρικής Φόρμας.** Είναι η πρώτη φόρμα (MAIN_FRM) που εμφανίζεται κατά την είσοδο στην εφαρμογή μας. Συγκεκριμένα η φόρμα αυτή εμφανίζει οκτώ (8) κουμπιά (Push Buttons). Ο χρήστης επιλέγοντας κάθε ένα από αυτά οδηγείται στις επιμέρους φόρμες της εφαρμογής.



Εικόνα 5.31 Πρώτη σχεδίαση της διεπαφής.

Κεφάλαιο 6
Σχεδίαση βάσεων δεδομένων και συστημάτων βάσεων δεδομένων. Μοντελοποίηση. Κανονικοποίηση. Χρήση τεχνολογίας πληροφοριακών συστημάτων

Σύνοψη

Στο κεφάλαιο αυτό γίνεται μία εκτενής εις βάθος παρουσίαση της σχεδίασης βάσεων δεδομένων. Αρχικά, παραθέτουμε μια περιεκτική συζήτηση των θεμάτων της μοντελοποίησης βάσης δεδομένων. Στη συνέχεια παρουσιάζονται οι σχεσιακές βάσεις δεδομένων και χρήσιμα θέματα κανονικοποίησης της βάσης δεδομένων. Τέλος, παρουσιάζονται και συζητώνται τεχνολογίες πληροφοριακών συστημάτων χρήσιμες για τη σχεδίαση συστημάτων βάσεων δεδομένων. Κύριος στόχος της εργασίας μας είναι η σχεδίαση και ανάπτυξη πληροφοριακού συστήματος (Information System) που βασίζεται σε σχεσιακή βάση δεδομένων (Relational database). Περιλαμβάνονται οι παρακάτω ενότητες:

1) **Μοντελοποίηση. Σημασιολογικά μοντέλα. Μοντέλο Οντοτήτων–Συσχετίσεων. Επεκτάσεις μοντέλου οντοτήτων-συσχετίσεων**. Σκοπός της ενότητας είναι να παρουσιάσει τις απαραίτητες έννοιες ώστε οι αναγνώστες να κατανοήσουν τη μεθοδολογία εννοιολογικής μοντελοποίησης (conceptual modeling) η οποία συνδέεται άρρηκτα με την τεχνολογία των σχεσιακών βάσεων δεδομένων και των συστημάτων βάσεων δεδομένων.
2) **Σχεσιακές βάσεις δεδομένων.** Κανονικοποίηση, μέθοδος συναρτησιακών εξαρτήσεων, πρώτη κανονική μορφή, δεύτερη κανονική μορφή, τρίτη κανονική μορφή, μέθοδος συναρτησιακών εξαρτήσεων και κανονική μορφή Boyce-Codd, άλλες κανονικές μορφές, μετασχηματισμός μοντέλου οντοτήτων συσχετίσεων σε σχεσιακή βάση δεδομένων, εμβάθυνση στη μοντελοποίηση και την κανονικοποίηση, ενοποίηση διαφορετικών συστημάτων βάσεων δεδομένων, ενιαίο παράδειγμα σχεδίασης σχεσιακής βάσης δεδομένων και υλοποίησης με γλώσσα SQL. Παρουσιάζονται δύο σενάρια (use cases):
   (α) Διαχείριση προσωπικού εταιρείας Deep Learning, σχεδίαση, υλοποίηση με γλώσσα SQL,
   (β) διαχείριση παραγγελιών.
3) **Ανάλυση και σχεδιασμός συστήματος βάσης δεδομένων (ΣΒΔ) με χρήση τεχνολογίας πληροφοριακών συστημάτων.** Το ΣΒΔ ως πληροφοριακό σύστημα. Τεχνολογίες πληροφοριακών συστημάτων. Μεθοδολογίες και εργαλεία ΠΣ. Από την προμελέτη, στη μελέτη σκοπιμότητας, στις προδιαγραφές και το διαγωνισμό. Ανάλυση δεδομένων. Δομημένες συνεντεύξεις και ερωτηματολόγια. Καταγραφή αποτελεσμάτων συνεντεύξεων. Συλλογή εντύπων και σχεδιασμός μοντέλων δεδομένων. Σχεδιασμός βάσης δεδομένων. Μελέτη Περίπτωσης. Εταιρεία οργάνωσης και διεξαγωγή σεμιναρίων.

Προαπαιτούμενη Γνώση

Προτείνεται η μελέτη του κεφαλαίου 1 του παρόντος συγγράμματος.

6.1 Μοντελοποίηση δεδομένων

Κύριος στόχος του κεφαλαίου 6 είναι η ανάπτυξη Πληροφοριακού Συστήματος (Information System) που βασίζεται σε Σχεσιακή Βάση Δεδομένων (Relational database). Σκοπός της ενότητας είναι να παρουσιάσει τις απαραίτητες έννοιες ώστε οι φοιτητές να κατανοήσουν τη μεθοδολογία εννοιολογικής μοντελοποίησης (conceptual modeling) που συνδέεται άρρηκτα με την τεχνολογία των Σχεσιακών Βάσεων Δεδομένων και των Συστημάτων Βάσεων Δεδομένων. Έμφαση δίδεται στην παρουσίαση σε βάθος των εννοιών της μεθοδολογίας Μοντελοποίησης Οντοτήτων Συσχετίσεων (Entity-Relationship Modeling methodology) και της βασιζόμενης σε αυτήν Κανονικοποίησης (normalisation). Κύριος στόχος της ενότητας είναι να εφοδιάσει τους φοιτητές με τις απαραίτητες γνώσεις έτσι ώστε να είναι ικανοί να μοντελοποιήσουν βάσεις δεδομένων με χρήση Μοντέλου Οντοτήτων Συσχετίσεων και να σχεδιάσουν τη σχεσιακή βάση δεδομένων.

6.1.1 Εισαγωγή

Η έννοια της **βάσης δεδομένων (ΒΔ)** και η έννοια του πληροφοριακού συστήματος βάσης δεδομένων (ΠΣΒΔ) είναι έννοιες που θα μας απασχολήσουν σε όλο το κεφάλαιο. Προς το παρόν, θα θεωρήσουμε ότι μία βάση δεδομένων είναι ένα είδος ηλεκτρονικής αρχειοθέτησης δεδομένων ή στοιχείων (data). Στην εργασία μας αποδίδουμε τον όρο data ως δεδομένα ή στοιχεία (data).

Σχεσιακή βάση δεδομένων (relational data base) είναι η βάση που έχει τα στοιχεία της οργανωμένα υπό μορφή πινάκων (tables). Οι σχεσιακές βάσεις δεδομένων είναι οι πλέον δημοφιλείς στον κόσμο των Πληροφοριακών Συστημάτων εδώ και πολλά χρόνια. Στην Εικόνα 6.1 παραθέτουμε παράδειγμα Σχεσιακής βάσης δεδομένων προσωπικού εταιρείας

Employee (κύριο κλειδί empno, ξένο κλειδί deptno)

Empno	Name	JobNo	DeptNo
10	ΣΠΥΡΟΥ	100	50
20	ΧΡΗΣΤΟΥ	200	60
30	ΝΙΚΟΥ	300	70

Department (κύριο κλειδί deptno)

DeptNo	Dname
50	ΠΩΛΗΣΕΙΣ
60	ΛΟΓΙΣΤΗΡΙΟ
70	ΜΙΣΘΟΔΟΣΙΑ

Job (κύριο κλειδί jobno)

JobNo	Job	Sal
100	ΠΩΛΗΤΗΣ	4200
200	ΑΝΑΛΥΤΗΣ	3000
300	ΧΕΙΡΙΣΤΗΣ	1000

Εικόνα 6.1 Παράδειγμα Σχεσιακής βάσης δεδομένων προσωπικού εταιρείας

Το **πληροφοριακό σύστημα** (ΠΣ) ενός οργανισμού ή μιας επιχείρησης που εξετάζουμε στην εργασία μας, στηρίζεται σε βάση δεδομένων, και έχει:

1) συγκεκριμένη οργανωτική δομή, αποτυπωμένη στο οργανόγραμμα και στον οργανισμό και τον κανονισμό λειτουργίας, δηλαδή το θεσμικό πλαίσιο λειτουργίας
2) υποστηρίζει λειτουργίες (operations) και διαδικασίες (processes),
3) χρησιμοποιεί για τη λειτουργία του διάφορους πόρους (resources), όπως λογισμικό (software) και υλικό (hardware), και λογισμικό εφαρμογών (application software)
4) βασίζει τη λειτουργία του σε ανθρώπους που έχουν συγκεκριμένους ρόλους

Το ΠΣ παρέχει υπηρεσίες σε χρήστες εντός και εκτός του οργανισμού ή της επιχείρησης. Κύρια χαρακτηριστικά του ΠΣ είναι ότι περιλαμβάνει ένα (υπό)σύστημα διαχείρισης των στοιχείων (data management) του και τα στοιχεία του είναι οργανωμένα σε βάση δεδομένων. Επομένως, με τον όρο διαχείριση των στοιχείων εννοούμε ότι το ΠΣ διαθέτει ένα ειδικό λογισμικό εφαρμογών με το οποίο εξασφαλίζει την εισαγωγή, ενημέρωση, διαγραφή, αναζήτηση και ανάκτηση αυτών των στοιχείων. Πιο συγκεκριμένα, το ΠΣ ανάμεσα στους πόρους του συμπεριλαμβάνει και ένα ειδικό λογισμικό που έχει κεντρικό ρόλο στη διαχείριση της βάσης δεδομένων και ονομάζεται **σύστημα (ή προϊόν) διαχείρισης βάσης δεδομένων (ΣΔΒΔ).** Το λογισμικό εφαρμογών (application software) συνεργάζεται και αξιοποιεί το ΣΔΒΔ για τη διαχείριση των δεδομένων.

Στο πλαίσιο της λειτουργίας του ΠΣ έχει ανατεθεί σε μία ομάδα εξειδικευμένων ειδικών επιστημόνων/υπαλλήλων η οργάνωση των στοιχείων και η αποθήκευση τους σε ΒΔ με τη βοήθεια του ΣΔΒΔ. Σύμφωνα με τον Date ο ειδικός επιστήμονας που αναλαμβάνει το έργο αυτό είναι ο **Data Administrator** που έχει την τεχνική υποστήριξη του **Data Base Administrator**. Σύμφωνα με τον Yourdon το έργο αυτό ανατίθεται σε ειδική ομάδα (Data Base Team). Συχνά, στην πράξη, το έργο αυτό ανατίθεται σε κάποιο έμπειρο ειδικό τον Αναλυτή Δεδομένων (data Analyst). Υπάρχουν πολλές ενδιαφέρουσες προσεγγίσεις στη βιβλιογραφία για το θέμα αυτό.

6.1.2 Μοντελοποίηση στοιχείων

Κεντρική εργασία των ειδικών που αναφέρθηκαν στην ενότητα 6.1.1 είναι η μοντελοποίηση των στοιχείων του οργανισμού/της επιχείρησης. Στη βιβλιογραφία η μοντελοποίηση οντοτήτων συσχετίσεων (Entity Relationship Modeling) αντιμετωπίζεται ως μεθοδολογία (methodology) ή ως διαδικασία (ERM process) ή ως τμήμα μεθοδολογίας και υπάρχουν πάρα πολλά εργαλεία (tools) μοντελοποίησης.

Εννοιολογική Μοντελοποίηση (conceptual modeling) είναι ένα σύνολο διαδικασιών, οι οποίες εφαρμόζονται με συνεπή τρόπο από τους ειδικούς (Data Administrator, Data Base Administrator, Data Base Team, Data Analyst) στα στοιχεία του Οργανισμού ή της Επιχείρησης. Μέσα από το διάλογο των ειδικών με τη Διοίκηση και τους χρήστες του ΠΣ, δηλαδή τους ανθρώπους που θα χρησιμοποιήσουν το σύστημα, η εφαρμογή των διαδικασιών μοντελοποίησης οδηγεί σε/καθορίζει μία σωστή οργάνωση των στοιχείων του Οργανισμού/της Επιχείρησης σε βάση δεδομένων. Σωστή οργάνωση σημαίνει ότι προσδιορίζονται τα «ιδεατά αντικείμενα» που ενδιαφέρουν και καθορίζονται οι απαραίτητες δομές δεδομένων (data structures). Ο όρος «ιδεατά αντικείμενα» αποδίδει στα ελληνικά ξενόγλωσσους όρους όπως conceptual object, entity, class κ.λπ. Τελικά, ανάλογα και με τη μεθοδολογία που θα εφαρμόσει ο ειδικός (ή το εργαλείο που θα χρησιμοποιήσει), τα ιδεατά αντικείμενα έχουν ειδικό νόημα και συνήθως κάθε μεθοδολογία τα αναπαριστά με ειδικά σύμβολα και δίνει οδηγίες κατασκευής ειδικών διαγραμμάτων.

Στη συνέχεια θα αναφερθούμε στις βασικές έννοιες της μοντελοποίησης και θα περιγράψουμε το πολύ γνωστό μοντέλο οντοτήτων συσχετίσεων (Entity Relationship model), που οφείλεται στον Peter Chen. Σε όλα τα γνωστά συγγράμματα (βλέπε για παράδειγμα το σύγγραμμα των Elmasri και Navathe) περιγράφεται και επέκταση του μοντέλου Οντοτήτων - Συσχετίσεων, το Διευρυμένο Μοντέλο Οντοτήτων Συσχετίσεων

(Enhanced Entity Relationship model). Τέλος, θα δοθούν κάποιες "συνταγές" μεταγραφής των μοντέλων αυτών σε σχεσιακές βάσεις δεδομένων .

6.1.3 Βασικές έννοιες της μοντελοποίησης δεδομένων. Οντότητες, συσχετίσεις, χαρακτηριστικά οντοτήτων και συσχετίσεων

Έστω ότι ενδιαφερόμαστε για την ανάπτυξη ενός Πληροφοριακού Συστήματος που θα βασίζεται σε Σχεσιακή (relational) βάση δεδομένων. Η σχεδίαση ενός μοντέλου της βάσης δεδομένων απαιτεί μία ολοκληρωμένη θεώρηση. Ο Engles προτείνει (βλέπε π.χ. το σύγγραμμα του Martin) η θεώρησή μας να προσεγγίσει διαδοχικά τρεις κόσμους («βασίλεια», realms), όπως φαίνεται και στην Εικόνα 1.1.: Πραγματικό κόσμο, κόσμο Πληροφορίας, κόσμο Δεδομένων.

Στη συνέχεια παρουσιάζεται, σύντομα και ελαφρώς τροποποιημένη, η πρόταση του Engles επειδή προσφέρει, κατά την άποψή μας, ένα καλό πλαίσιο για τη μοντελοποίηση των στοιχείων.

Πραγματικός κόσμος 1) Οντότητες (entities) και 2) Συλλογές (όμοιων) οντοτήτων (entity types, entity sets) 3) Συσχετίσεις οντοτήτων (relationships) και 4) Τύποι συσχετίσεων (relationships types, relationships sets) 5) Ιδιότητες (properties) οντοτήτων και 6) Χαρακτηριστικά τύπων οντοτήτων (attributes) 7) Ιδιότητες (properties) συσχετίσεων και 8) Χαρακτηριστικά (attributes) Τύπων συσχετίσεων
Κόσμος Πληροφορίας 1. Χαρακτηριστικά (τύπων) οντοτήτων που μας ενδιαφέρουν 2. Χαρακτηριστικά (τύπων) συσχετίσεων που μας ενδιαφέρουν 3. Καθορισμός συμβολικών ονομάτων χαρακτηριστικών 4. Πεδία τιμών (domains)
Κόσμος Δεδομένων • Τύποι δεδομένων (data types) για τα χαρακτηριστικά • Μήκη (length) • Περιορισμοί (constraints)

Εικόνα 6.2 Η προσέγγιση των τριών κόσμων στη μοντελοποίηση (Engles)

Επομένως, κατά τη μοντελοποίηση πρέπει να προσεγγίζουμε τους τρείς κόσμους:

1) τον **πραγματικό κόσμο**, από όπου πρέπει να ανιχνευθούν:
   - Οι **οντότητες (entities)**, δηλαδή, πρόσωπα, αντικείμενα, γεγονότα, πράξεις ή/και αφηρημένες έννοιες γύρω από τις οποίες θα συγκεντρωθεί και θα αποθηκευθεί πληροφορία.
   - Οι **ιδιότητές τους (properties)**
   - **Συλλογές ομοίων οντοτήτων (entity sets ή entity types).** Συχνά αναφέρονται στη βιβλιογραφία απλά ως οντότητες (entities).
   - **Συσχετίσεις (relationships)** μεταξύ οντοτήτων

- **Τύποι συσχετίσεων μεταξύ συλλογών ομοίων οντοτήτων (relationship types).** Αναφέρονται στη βιβλιογραφία απλά και ως συσχετίσεις (relationships).
- **Τα χαρακτηριστικά (attributes)** τύπων συσχετίσεων.

Τα παραδείγματα 6.1.4.1, 6.1.4.2 και 6.1.4.3, στη συνέχεια, ξεκαθαρίζουν τις διαδικασίες μοντελοποίησης που γίνονται στο πλαίσιο της θεώρησης του πραγματικού κόσμου.

2) τον **κόσμο της πληροφορίας**, όπως διαμορφώνεται μέσα από τις συγκεκριμένες παραδοχές που θα γίνουν από τον αναλυτή δεδομένων ή την ομάδα σχεδιασμού της βάσης δεδομένων. Στο στάδιο αυτό καθορίζεται ποιά χαρακτηριστικά (attributes) ενδιαφέρουν, τελικά, και γίνεται εκχώρηση (assignment) συμβολικών ονομάτων και καθορισμός των πεδίων τιμών για τα χαρακτηριστικά αυτά.

   Το παράδειγμα 6.1.4.4, στη συνέχεια, αναφέρεται στις διαδικασίες μοντελοποίησης που γίνονται στο πλαίσιο της θεώρησης του κόσμου της πληροφορίας.

3) τον **κόσμο των δεδομένων (data)**, όπου θα πρέπει να διατυπωθούν περιορισμοί που ικανοποιούν τα στοιχεία και πρέπει να χρησιμοποιηθούν συμβολοσειρές (strings) χαρακτήρων ή ακολουθίες δυαδικών ψηφίων (bits) για την κωδικοποίηση της πληροφορίας.

   Το παράδειγμα 6.1.4.5, στη συνέχεια, αναφέρεται στις διαδικασίες μοντελοποίησης που γίνονται στο πλαίσιο της θεώρησης του κόσμου των δεδομένων.

Ο Peter Chen θεωρεί τον τύπο οντότητας ως κατηγορία (category) και την οντότητα ως στιγμιότυπο (instance) του τύπου οντότητας. Ο Chen προτείνει να αποδίδουμε τις οντότητες με ουσιαστικά (nouns), π.χ., computer, employee, song. Επιπλέον, προτείνει να αποδίδουμε τις συσχετίσεις με ρήματα (verbs). Τέλος, προτείνει κάποιους κανόνες για την αντιστοίχιση περιγραφών, διατυπωμένων στην αγγλική φυσική γλώσσα, σε ΜΟΣ ("mapping natural language descriptions into ER diagrams"). Ακολουθεί ο πίνακας 6.1 με παραδείγματα.

Πίνακας 6.1 Γλωσσολογική προσέγγιση στη σχεδίαση Μοντέλου Οντοτήτων Συσχετίσεων (Peter Chen).

Grammar	ER model	Examples
Common noun	Entity type	Student Lecturer, Course
Proper noun	Entity	Brown, Chen, Database
Transitive verb	Relationship type	He teaches database courses
Intransitive verb	Attribute type	Codd died.
Adjective (attributive)	Attribute for entity	That's an interesting course.
Adjective (predicative)	Attribute for entity	That course is interesting.
Adjective (postpositive)	Attribute for entity	Tell me something useful.
Adjective (substantive)	Attribute for entity	The expert (student) and the novice (student).

Στη συνέχεια θα χρησιμοποιήσουμε συνήθως χωρίς διάκριση τον όρο οντότητα τόσο για την έννοια οντότητα (entity) όσο και για την έννοια συλλογή όμοιων οντοτήτων (entity type). Επίσης, αφού παρακάτω θα πρέπει να μιλήσουμε για την έννοια συσχέτιση (relationship) μεταξύ οντοτήτων και την έννοια συσχέτιση μεταξύ συλλογών όμοιων οντοτήτων (relationship type) δε θα κάνουμε κάποια διάκριση, αν δεν είναι απαραίτητο, αλλά θα χρησιμοποιήσουμε τον όρο συσχέτιση.

6.1.4 Οντότητες και τα χαρακτηριστικά τους

6.1.4.1 Παράδειγμα οντοτήτων εφαρμογής διαχείρισης προγραμματιστών

Έστω μας ενδιαφέρει να μοντελοποιήσουμε τα στοιχεία των υπαλλήλων του Οίκου Λογισμικού Deep Thought. Μία από τις οντότητες (entities) είναι οποιοσδήποτε από τους προγραμματιστές π.χ., ο προγραμματιστής ΝΙΚΟΥ. Αν μελετήσουμε τους προγραμματιστές μπορούμε να καταλήξουμε ότι ο συγκεκριμένος προγραμματιστής ΝΙΚΟΥ, και κάθε προγραμματιστής, ανήκει σε μία συλλογή όμοιων οντοτήτων (entity type), δηλαδή σε ένα σύνολο προγραμματιστών. Όλοι οι προγραμματιστές μοιράζονται/έχουν κάποια κοινά χαρακτηριστικά (attributes). Τέτοια χαρακτηριστικά της οντότητας ΝΙΚΟΥ αλλά και όλων των όμοιων οντοτήτων θα μπορούσαν να είναι: Κωδικός, μισθός, γλώσσα προγραμματισμού που γνωρίζει ο προγραμματιστής κ.λπ.

6.1.4.2 Παράδειγμα οντοτήτων εφαρμογής διαχείρισης τιμολογίων

Έστω μας ενδιαφέρει να μοντελοποιήσουμε τα στοιχεία των τιμολογίων της εταιρείας MARKET HELLAS. Μία από τις οντότητες είναι ένα συγκεκριμένο τιμολόγιο. Συλλογή όμοιων οντοτήτων είναι τα τιμολόγια της εταιρείας. Χαρακτηριστικά της οντότητας μπορούν να είναι ο αριθμός τιμολογίου, το συνολικό ποσό, η ημερομηνία έκδοσης κ.λπ.

6.1.4.3 Παράδειγμα. Συσχέτιση οντοτήτων και τα χαρακτηριστικά της

Έστω ότι στην εταιρεία του Παραδείγματος 6.1.4.2 προσδιορίσαμε τις οντότητες ΠΡΟΜΗΘΕΥΤΗΣ, ΠΡΟΪΟΝ με χαρακτηριστικά (attributes) αντίστοιχα:

- Αριθμός προμηθευτή, όνομα, έδρα
- Αριθμός προϊόντος, όνομα, βάρος, χρώμα, έδρα αποθήκης

Μπορούμε να ορίσουμε μία συσχέτιση ανάμεσα στις οντότητες αυτές. Για παράδειγμα η συσχέτιση "εφοδιάζει με" συσχετίζει τις δύο οντότητες:

ΠΡΟΜΗΘΕΥΤΗΣ – «εφοδιάζει με» --> ΠΡΟΪΟΝ

Είναι δυνατόν μια συσχέτιση να έχει χαρακτηριστικά (attributes). Έτσι η συσχέτιση "εφοδιάζει με" έχει χαρακτηριστικά την ποσότητα και την ημερομηνία. Προσοχή! Θα μπορούσε, ενδεχομένως, η συσχέτιση (relationship) να θεωρηθεί σαν μια οντότητα με όνομα, π.χ., ΕΦΟΔΙΑΣΜΟΣ. Θα μπορούσε, επίσης, να σκεφτεί κάποιος ότι μπορεί να καταργηθεί η συσχέτιση και τα χαρακτηριστικά της, π.χ., ποσότητα, ημερομηνία, να ενσωματωθούν στις άλλες οντότητες. Για παράδειγμα, η οντότητα του προμηθευτή ίσως θα μπορούσε να έχει τα δύο αυτά χαρακτηριστικά, όπως φαίνεται παρακάτω:

ΠΡΟΜΗΘΕΥΤΗΣ(Αριθμός, όνομα, έδρα, Αριθμός προϊόντος, ποσότητα, ημερομηνία).

Η απόφαση που θα λάβει ο ειδικός για τον καθορισμό οντοτήτων, συσχετίσεων κ.λπ. είναι δύσκολη. Για παράδειγμα, η απόφαση να καταργήσετε τη συσχέτιση ΕΦΟΔΙΑΣΜΟΣ και να ενσωματώσετε τα χαρακτηριστικά της στην οντότητα ΠΡΟΜΗΘΕΥΤΗΣ δεν είναι ορθή. Γενικά, απαιτούνται αρκετή σκέψη και αρκετή συζήτηση με τους χρήστες για να έχουμε μία σωστή οργάνωση της βάσης δεδομένων.

Στη συνέχεια της εργασίας μας θα εξετάσουμε **το σχεσιακό μοντέλο δεδομένων,** το οποίο διαχειρίζεται οντότητες και συσχετίσεις οντοτήτων με τον ίδιο τρόπο, σαν πίνακες.

6.1.4.4 Παράδειγμα. Χαρακτηριστικά οντότητας και πεδία ορισμού.

Για την οντότητα ΠΡΟΓΡΑΜΜΑΤΙΣΤΗΣ καθορίζουμε ότι ενδιαφέρουν τα χαρακτηριστικά:

Κωδικός (empno), Ονομα (ename), Μισθός (sal), κωδικός (langNo) και γλώσσα προγραμματισμού (language), χρόνια χρήσης της γλώσσας (use) και συνολική εμπειρία (expr).

Στα χαρακτηριστικά αυτά δίνουμε συμβολικά ονόματα:

EMPNO, ENAME, SAL, LANG, USE, EXPR

Τα πεδία ορισμού (domain) των χαρακτηριστικών, θα συμβολίζονται με DOM, και είναι τα εξής:

DOM(EMPNO)={x|x 5-ψήφιος ακέραιος}

DOM(SAL) ={x|x δεκαδικός αριθμός με 2 δεκαδικά ψηφία}

DOM(LANGNO) ={x|x 3-ψήφιος ακέραιος}

DOM(LANG)={ADA, C, C++, COBOL, FORTRAN, JAVA, LISP, PASCAL, PL/I, PROLOG, PYTHON, VB}

DOM(USE)=DOM(EXPR)={x|x θετικός ακέραιος} κ.λπ.

6.1.4.5 Παράδειγμα. Χαρακτηριστικά και τύποι δεδομένων.

Για την οντότητα ΠΡΟΓΡΑΜΜΑΤΙΣΤΗΣ καθορίζεται ότι υπάρχουν οι παρακάτω περιορισμοί:

- ο κωδικός είναι μοναδικός, άρα υποψήφιο κύριο κλειδί
- κάθε προγραμματιστής έχει ακριβώς ένα όνομα
- μπορεί να υπάρχουν συνωνυμίες
- έχει ακριβώς ένα μισθό
- μπορεί να γνωρίζει μία ή περισσότερες από μία γλώσσες προγραμματισμού
- τα χρόνια που ο προγραμματιστής χρησιμοποιεί κάθε γλώσσα προγραμματισμού είναι το πολύ διψήφιος θετικός ακέραιος
- η συνολική εμπειρία είναι το πολύ διψήφιος θετικός ακέραιος κ.λπ.

Για τα χαρακτηριστικά της οντότητας ΠΡΟΓΡΑΜΜΑΤΙΣΤΗΣ καθορίζονται τύποι δεδομένων και μήκη:

- ο κωδικός είναι τριψήφιος ακέραιος αριθμός και ο τύπος δεδομένων (data type) του είναι NUMBER(3).
- το μήκος του ονόματος δεν ξεπερνά τους 70 χαρακτήρες και ο τύπος δεδομένων είναι VARCHAR2(70)
- έχει μισθό πραγματικό αριθμό με δύο δεκαδικά ψηφία, πάνω από 4000 (περιορισμός) άρα ο τύπος δεδομένων για το μισθό θα μπορούσε να είναι NUMBER(8,2)
- ο τύπος δεδομένων και το μήκος για τη γλώσσα είναι VARCHAR2(40) κ.λπ.

Χρήσιμο σε αυτές τις περιπτώσεις είναι να σχεδιαστεί ο πίνακας που αντιστοιχεί στην οντότητα με **δείγμα δεδομένων** (sample of data). Στην Εικόνα 6.3 δίδεται δείγμα δεδομένων για τον πίνακα της οντότητας προγραμματιστής, τον πίνακα της γλώσσας προγραμματισμού και τον πίνακα της συσχέτισης «γνωρίζει» (ΠΡΟΓΡΑΜΜΑΤΙΣΤΗΣ ... γνωρίζει ΓΛΩΣΣΑ ...).

PROGRAMMER

EMPNO	ENAME	EXPR	SAL
10010	CODD	4	4000
10101	DATE	5	6000
10301	MARTIN	1	4000

KNOWS

EMPNO	LANGNO	USE
10010	003	3
10010	004	2
10101	001	5
10301	002	1

LANGUAGE

LANGNO	LANG
001	C
002	C++
003	JAVA
004	PYTHON

Εικόνα 6.3 Δείγμα δεδομένων για τους αντίστοιχους πίνακες των οντοτήτων προγραμματιστής και γλώσσα προγραμματισμού και της συσχέτισης «γνωρίζει».

6.1.5 Κύριο κλειδί και υποψήφια κύρια κλειδιά οντότητας

Σε κάθε συλλογή όμοιων οντοτήτων υπάρχουν απλά χαρακτηριστικά ή συνδυασμός χαρακτηριστικών που μπορούν να αποτελέσουν υποψήφια κύρια κλειδιά (candidate primary keys) για τις οντότητες. Κάθε υποψήφιο κύριο κλειδί πρέπει να χαρακτηρίζει μοναδικά κάθε οντότητα της συλλογής. Στο συγκεκριμένο παράδειγμα του πίνακα που αντιστοιχεί στον τύπο οντότητας ΠΡΟΓΡΑΜΜΑΤΙΣΤΗΣ, εκ πρώτης όψεως, υποψήφια κύρια κλειδιά θα μπορούσαν να είναι τα χαρακτηριστικά EMPNO, NAME. Βέβαια, αν εξετάσουμε καλύτερα την περίπτωση, το δείγμα δεν είναι αντιπροσωπευτικό, π.χ., λείπουν οι συνωνυμίες. Επομένως, αν δεχτούμε ότι μπορούμε να έχουμε συνωνυμίες μεταξύ των (οντοτήτων) προγραμματιστών τότε το χαρακτηριστικό NAME δεν μπορεί να είναι υποψήφιο κύριο κλειδί. Στον πίνακα που αντιστοιχεί στον τύπο οντότητας γλώσσα προγραμματισμού υποψήφια κύρια κλειδιά θα μπορούσαν να είναι τα χαρακτηριστικά LANGNO, LANG. Από τα υποψήφια κύρια κλειδιά πρέπει να επιλέξουμε μόνο ένα ως κύριο κλειδί. Τότε όλες οι οντότητες είναι διακριτές γιατί έχουν μία μοναδική τιμή κύριου κλειδιού. Στον πίνακα που αντιστοιχεί στη συσχέτιση κύριο κλειδί είναι ο συνδυασμός (EMPNO, LANGNO).

Ακολουθεί περαιτέρω παρουσίαση και συζήτηση των Συσχετίσεων Οντοτήτων

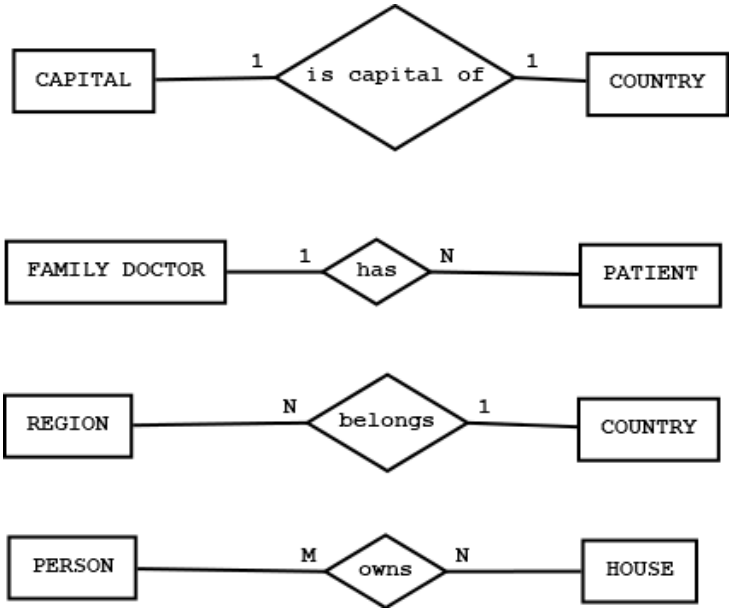
6.1.6 Είδη συσχετίσεων (relationships) μεταξύ δύο οντοτήτων

Συνήθως μία συσχέτιση (relationship) ορίζεται μεταξύ δύο οντοτήτων (entities) και τότε ονομάζεται δυαδική (binary relationship). Μία δυαδική συσχέτιση μπορεί να έχει έναν μόνο από τους παρακάτω τέσσερις τύπους:

- 1 προς 1 (one-to-one)
- 1 προς N (one-to-many)
- N προς 1 (many-to-one)
- M προς N (many-to-many)

Στην Εικόνα 6.4.3 γίνεται φανερή η διάκριση των τεσσάρων δυαδικών τύπων:

1) Ένα προς ένα συσχέτιση μεταξύ πρωτεύουσας και χώρας
2) Ένα προς πολλά συσχέτιση μεταξύ σύμβουλου καθηγητή και φοιτητών-φοιτητριών
3) Πολλά προς ένα συσχέτιση μεταξύ περιφέρειας και χώρας
4) Πολλά προς πολλά συσχέτιση μεταξύ προσώπου και ιδιοκτησίας



Εικόνα 6.4 Τύποι δυαδικών συσχετίσεων (binary relationships) οντοτήτων (examples_of_relationships_new.png)

Ακολουθεί εκτενέστερη συζήτηση με τη βοήθεια παραδειγμάτων.

6.1.6.1 Παραδείγματα δυαδικών συσχετίσεων

Συσχέτιση τύπου 1:1

Έστω οι οντότητες ΑΝΔΡΑΣ, ΓΥΝΑΙΚΑ και η συσχέτιση "είναι παντρεμένος με". Θα εξετάσουμε κάποια “στιγμιότυπα” (instances) της συσχέτισης «Ο ... ¨είναι παντρεμένος με¨ την ...». όπου σε εισαγωγικά ¨ ¨ έχουμε γράψει μία φράση που περιλαμβάνει ένα ρήμα. Η φράση ορίζει τη συσχέτιση ανάμεσα στις δύο οντότητες. Στα παραδείγματά μας, οι οντότητες θα καθοριστούν με χρήση ουσιαστικού (κυρίου ονόματος). Τα στιγμιότυπα των συσχετίσεων των οποίων την ορθότητα θα εξετάσουμε κατασκευάζονται με χρήση τυχαίων κυρίων ονομάτων και είναι τα εξής:

ΝΙΚΟΣ –«είναι παντρεμένος με» --> ΜΑΡΙΑ (είναι λογικό, νόμιμο)

ΝΙΚΟΣ –«είναι παντρεμένος με» --> ΜΑΡΙΑ, ΑΝΤΙΓΟΝΗ (απαράδεκτο)

ΝΙΚΟΣ, ΚΩΣΤΑΣ –«είναι παντρεμένος με» --> ΜΑΡΙΑ (απαράδεκτο)

Άρα η συσχέτιση είναι τύπου 1:1 και οι οντότητες αλλά και οι πίνακες που τις αναπαριστούν, όπως βλέπουμε στη συνέχεια είναι MAN(**MCODE,** NAME, AGE) και WOMAN(**WCODE**, NAME), όπου το κύριο κλειδί φαίνεται υπογραμμισμένο. Στην Εικόνα 6.5 βλέπουμε τους πίνακες με άλλο συμβολισμό!

MAN

MCODE	NAME	AGE
Primary key		

WOMAN

WCODE	NAME
Primary key	

Εικόνα 6.5 Πίνακες που αντιστοιχούν στις οντότητες και στην τύπου 1:1 συσχέτισή τους

Για να αναπαραστήσουμε μία δυαδική συσχέτιση (relationship) τύπου 1:1 οι πίνακες που αντιστοιχούν στις οντότητες θα αλλάξουν και μπορούν να είναι οι εξής:

MAN(**MCODE**, NAME, AGE, WCODE), WOMAN(**WCODE**, NAME) ή

MAN(**MCODE**, NAME, AGE), WOMAN(**WCODE**, NAME, MCODE)

Στην Εικόνα 6.6 βλέπουμε τους πίνακες με άλλο συμβολισμό! Εναλλακτικά βλέπουμε και άλλη σχεδίαση των πινάκων.

MAN

MCODE	NAME	AGE
Primary key		

WOMAN

WCODE	NAME	MCODE
Primary key		Foreign key

Εναλλακτικά

MAN

MCODE	NAME	AGE	WCODE
Primary key			Foreign key

WOMAN

WCODE	NAME
Primary key	

Εικόνα 6.6 Πίνακες που αντιστοιχούν στη συσχέτιση τύπου 1:1 και τους τύπους οντοτήτων της

Αν θέλαμε να έχουμε στη βάση δεδομένων τα στοιχεία όλων των γάμων που έκαναν οι άνδρες και οι γυναίκες τότε η συσχέτιση είναι M:N και μπορεί να παρασταθεί σαν ένας ξεχωριστός πίνακας, ως εξής:

IS_MARRIED_WITH(**MCODE, WCODE**)

Αν θεωρήσουμε ότι η συσχέτιση τύπου M:N έχει την ιδιότητα ANNIVERSARY ο πίνακας μπορεί να οριστεί ως εξής:

IS_MARRIED_WITH(**MCODE, WCODE**, ANNIVERSARY)

Στην Εικόνα 6.7 παρατίθεται με άλλο συμβολισμό η βάση δεδομένων!

MAN

MCODE	NAME	AGE
Primary key		

WOMAN

WCODE	NAME
Primary key	

IS_MARRIED_WITH

MCODE	WCODE	ANNIVERSARY
Foreign key	Foreign key	

Primary key=(MCODE, WCODE)

Εικόνα 6.7 Πίνακες που αντιστοιχούν στη συσχέτιση τύπου M:N και τους τύπους οντοτήτων της

Συσχέτιση τύπου 1:N

Με τελείως ανάλογο τρόπο μπορούμε να εξετάσουμε και την περίπτωση συσχέτισης του τύπου N:1.

Έστω οι οντότητες ΤΜΗΜΑ, ΥΠΑΛΛΗΛΟΣ και η συσχέτιση «έχει (υπάλληλο)». Θα εξετάσουμε κάποια στιγμιότυπα της συσχέτισης:

"ΔΙΕΥΘΥΝΣΗ ΠΡΟΣΩΠΙΚΟΥ" –«έχει (υπάλληλο)»--> ΝΙΚΟ, ΜΑΡΙΑ (νόμιμο)

"ΔΙΕΥΘΥΝΣΗ ΠΡΟΣΩΠΙΚΟΥ","ΛΟΓΙΣΤΗΡΙΟ" –«έχει» -> ΝΙΚΟ (απαράδεκτο)

Άρα η συσχέτιση είναι 1:N και οι δύο οντότητες (όπως θα δούμε) θα παρασταθούν τελικά σαν δύο πίνακες που είναι οι εξής:

DEPT(**DEPTNO,** NAME, BUDGET)

EMPLOYEE(**ENAME**, SAL, HIREDATE, DEPTNO)

Βέβαια, αν η συσχέτιση έχει κάποιο χαρακτηριστικό, π.χ., την ημερομηνία τοποθέτησης του υπαλλήλου στο τμήμα ASSIGNDATE, θα μπορούσαμε να παραστήσουμε οντότητες και συσχέτιση ως εξής:

DEPT(**DEPTNO**, NAME)

EMPLOYEE(**ENAME,** SAL, HIREDATE, DEPTNO, ASSIGNDATE), Foreign key=deptno.

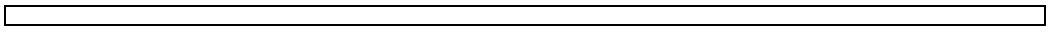
Στην Εικόνα 6.8 παρατίθεται με άλλο συμβολισμό η βάση δεδομένων!

DEPT

DEPTNO	NAME	AGE
Primary key		

EMPLOYEE

ENAME	SAL	HIREDATE	DEPTNO	ASSIGNDATE
Primary key			Foreign key	



Εικόνα 6.8 Πίνακες που αντιστοιχούν στη συσχέτιση τύπου 1:N. Το χαρακτηριστικό της συσχέτισης ASSIGNDATE τοποθετήθηκε στον πίνακα EMPLOYEE.

Συσχετίσεις τύπου M:N

Έστω οι οντότητες ΠΡΟΜΗΘΕΥΤΗΣ, ΠΡΟΪΟΝ, και η συσχέτιση "προμηθεύει με". Θα εξετάσουμε κάποια στιγμιότυπα της συσχέτισης

ΝΙΚΟΣ, ΠΑΝΟΣ –«προμηθεύει με» --> ΚΑΤΣΑΒΙΔΙ (νόμιμο)

ΝΙΚΟΣ –«προμηθεύει με» --> ΚΑΤΣΑΒΙΔΙ, ΒΙΔΑ (νόμιμο)

Άρα, η συσχέτιση είναι τύπου M:N και οι πίνακες για τις οντότητες και τη συσχέτιση είναι οι εξής:

SUPPLIERS(**SNO**, SNAME, STATUS, SCITY), όπου SNO κωδικός προμηθευτή (κύριο κλειδί), SNAME επωνυμία, STATUS χαρακτηρισμός προμηθευτή σχετικός με τη δυνατότητα υπογραφής συμβολαίου, π.χ.10-Approved, 15-Normalized, 20-Identified, 25-Registered, 30-Rejected και SCITY έδρα

PARTS(**PNO**, PNAME, PCITY), όπου PNO κωδικός ανταλλακτικού (κύριο κλειδί), PNAME όνομα, PCITY έδρα αποθήκευσης ανταλλακτικού

SHIPMENT(**SNO, PNO**, QUANTITY, SHIPMENT_DATE), όπου QUANTITY ποσότητα, SHIPMENT_DATE ημερομηνία αποστολής

Αν η συσχέτιση δεν έχει χαρακτηριστικά που ενδιαφέρουν αντί του πίνακα SHIPMENT έχουμε τον πίνακα SHIPMENT_NEW:

SHIPMENT_NEW(**SNO, PNO**)

Προσοχή! Αν θεωρηθεί ως κύριο κλειδί του πίνακα SHIPMENT ο συνδυασμός (SNO, PNO) τότε η παραπάνω σχεδίαση οδηγεί σε προβλήματα. Αν ο πίνακας έχει τη μορφή που φαίνεται στην Εικόνα 6.9 και το κύριο κλειδί είναι (SNO, PNO) τότε δεν μπορούμε να εισάγουμε τη γραμμή (1234, 100, 25, “19/12/2017”).

Στην περίπτωση αυτή για να λυθεί το πρόβλημα προσθέτουμε ως κύριο κλειδί στον πίνακα μία στήλη αύξοντος αριθμού (accno) (βλέπε Εικόνα 6.9). Τα ΣΔΒΔ μπορούν να παράγουν ένα «τεχνητό κύριο κλειδί» (artificial primary key) που ενημερώνεται αυτόματα με trigger ή με άλλους τρόπους (βλέπε για παράδειγμα, αντικείμενο sequence στη γλώσσα PL-SQL). Η λύση αυτή ακολουθείται από πολλούς σχεδιαστές βάσης δεδομένων και σε άλλες περιπτώσεις, Δηλαδή, ορίζουν ως κύριο κλειδί στους πίνακές τους μια στήλη με τον αύξοντα αριθμό των γραμμών.

SHIPMENT

SNO	PNO	QUANTITY	SHIPMENT_DATE
1234	100	12	18/12/2017
1234	200	25	18/12/2017
4567	100	45	18/12/2017
7890	100	23	18/12/2017
7890	300	65	18/12/2017

Εναλλακτική και ορθότερη σχεδίαση με προσθήκη στήλης accno

SHIPMENT

ACCNO	SNO	PNO	QUANTITY	SHIPMENT_DATE
1	1234	100	12	18/12/2017
2	1234	200	25	18/12/2017
3	4567	100	45	18/12/2017
4	7890	100	23	18/12/2017
5	7890	300	65	18/12/2017
6	1234	100	25	19/12/2017

Primary key=ACCNO

Εικόνα 6.9 Πίνακες που αντιστοιχούν στη συσχέτιση τύπου M:N. Σχεδίαση με σύνθετο κύριο κλειδί και εναλλακτική σχεδίαση με κύριο κλειδί έναν αύξοντα αριθμό ή γενικότερα κάποιο ¨τεχνητό κύριο κλειδί»

6.1.7 Άλλοι τύποι συσχετίσεων. Παρατηρήσεις και παραδείγματα

Υπάρχουν και άλλοι τύποι συσχετίσεων:

- Υπάρχουν συσχετίσεις (relationships) μεγαλύτερου βαθμού, π.χ. τριαδικές συσχετίσεις οι οποίες συνδέουν περισσότερους από δύο τύπους οντοτήτων (entities).
- Υπάρχουν συσχετίσεις που συνδέουν την ίδια οντότητα (αυτοπαθείς). Για παράδειγμα, το γεγονός ότι υπάρχουν κάποια εξαρτήματα (ανταλλακτικά, parts) που είναι εξαρτήματα άλλων μας οδηγεί στον ορισμό μιας συσχέτισης αυτού του τύπου.
- Επισημαίνουμε ότι περισσότερες από μια συσχετίσεις μπορούν να συνδέουν δύο οντότητες.
- Μια συσχέτιση, ενδεχομένως, θα μπορούσε να θεωρηθεί σαν οντότητα. Για παράδειγμα, η συσχέτιση «εξάρτημα Pi αποθηκεύεται στην αποθήκη Wλ» ίσως μπορεί να θεωρηθεί ότι είναι μια οντότητα για την οποία πρέπει να κρατάμε ποσότητα κ.λπ.
- Μία συσχέτιση οριζόμενη σε περισσότερες από δύο οντότητες μπορεί να δίνει περισσότερη πληροφορία από το συνδυασμό των επί μέρους συσχετίσεων της. Για παράδειγμα, η τριαδική συσχέτιση τύπου M:N:N «ο προμηθευτής ... προμηθεύει το ... ανταλλακτικό στο ... έργο» δίνει περισσότερη πληροφορία από το συνδυασμό των συσχετίσεων «ο προμηθευτής ... προμηθεύει το ... ανταλλακτικό», «ο προμηθευτής ... προμηθεύει το ... έργο» και «το ... ανταλλακτικό χρησιμοποιείται στο ... έργο». Στις περιπτώσεις αυτές χρησιμοποιείται ο όρος «παγίδευση σύνδεσης» (connection trap).

6.1.7.1 Σενάριο χρήσης (use case)

Έστω ότι μια κατασκευαστική εταιρεία ενδιαφέρεται για ένα σύστημα βάσεων δεδομένων (data base system) που θα κρατάει πληροφορίες γύρω από:

- τα έργα της (projects)
- τα ανταλλακτικά (parts) που χρησιμοποιούνται στα έργα
- τους προμηθευτές (suppliers) των ανταλλακτικών
- τις αποθήκες (warehouses) των ανταλλακτικών
- τους υπαλλήλους (employees) που εργάζονται στα έργα
- τα τμήματα (departments)

- τις έδρες (locations) των αποθηκών κλπ.

Αυτές οι έννοιες είναι οι βασικές Οντότητες (entities) για τις οποίες καταχωρίζουμε στοιχεία (data) στο σύστημα. Οι οντότητες (entities) συνδέονται μεταξύ τους με συσχετίσεις (relationships). Οι συσχετίσεις είναι οι εξής:

R1: SUPPLIER ... supplies PROJECT … (ο προμηθευτής ... προμηθεύει το ... έργο …)

R2: SUPPLIER ... supplies PART … (ο προμηθευτής ... προμηθεύει το ... ανταλλακτικό …)

R3: PROJECT ... uses PART … (το έργο ... χρησιμοποιεί το ... ανταλλακτικό ...)

R4: SUPPLIER ... supplies PART … to the PROJECT … (ο προμηθευτής ... προμηθεύει το ... ανταλλακτικό στο ... έργο …), π.χ., "The supplier Alpha supplies Bolts to the project Rep" (τριαδική συσχέτιση M:N:N)

R5: WAREHOUSE ... has PART … (η αποθήκη ... έχει το ανταλλακτικό ...)

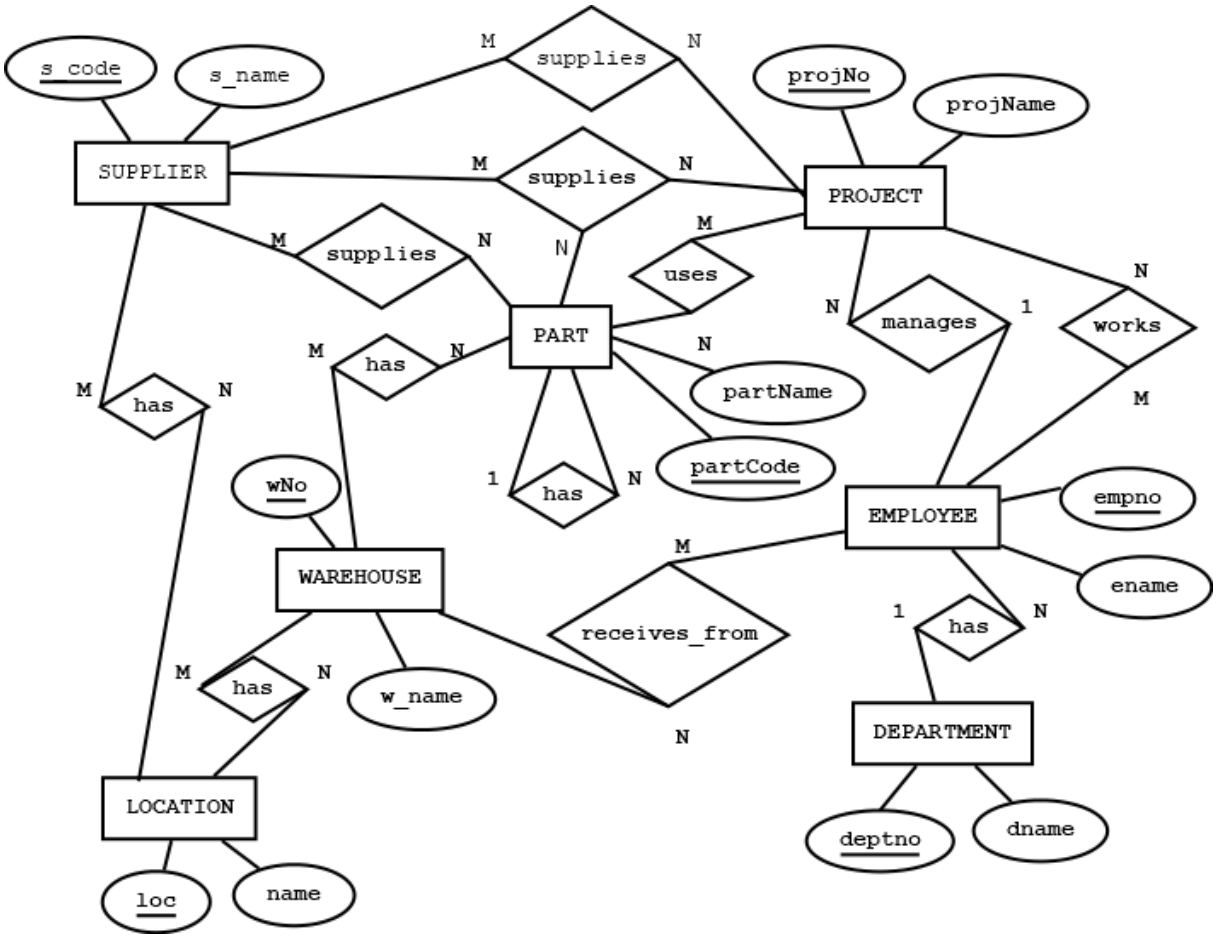
R6: EMPLOYEE ... works_for PROJECT … (ο υπάλληλος ... εργάζεται στο ... έργο …)

R7: EMPLOYEE ... manages PROJECT … (ο υπάλληλος ... είναι επικεφαλής του ... έργου …)

R8: PART … has Part … (το ανταλλακτικό ... είναι τμήμα του ... ανταλλακτικού …) (αυτοπαθής συσχέτιση)

R9: EMPLOYEE … receives_part_from WAREHOUSE … (ο υπάλληλος ... παραλαμβάνει από την ... αποθήκη …) κ.λπ.

Στην Εικόνα 6.10 φαίνονται οι συσχετίσεις των οντοτήτων της κατασκευαστικής εταιρείας. Προσοχή! Η μοντελοποίηση είναι η ίδια αν αντί για ένα σύστημα διαχείρισης βάσης δεδομένων ενδιαφερόμαστε για ένα σύστημα βασιζόμενο σε παραδοσιακά αρχεία.



Εικόνα 6.10 Οντότητες και συσχετίσεις οντοτήτων κατασκευαστικής εταιρείας

6.1.8 Δομικά στοιχεία του Μοντέλου Οντοτήτων Συσχετίσεων

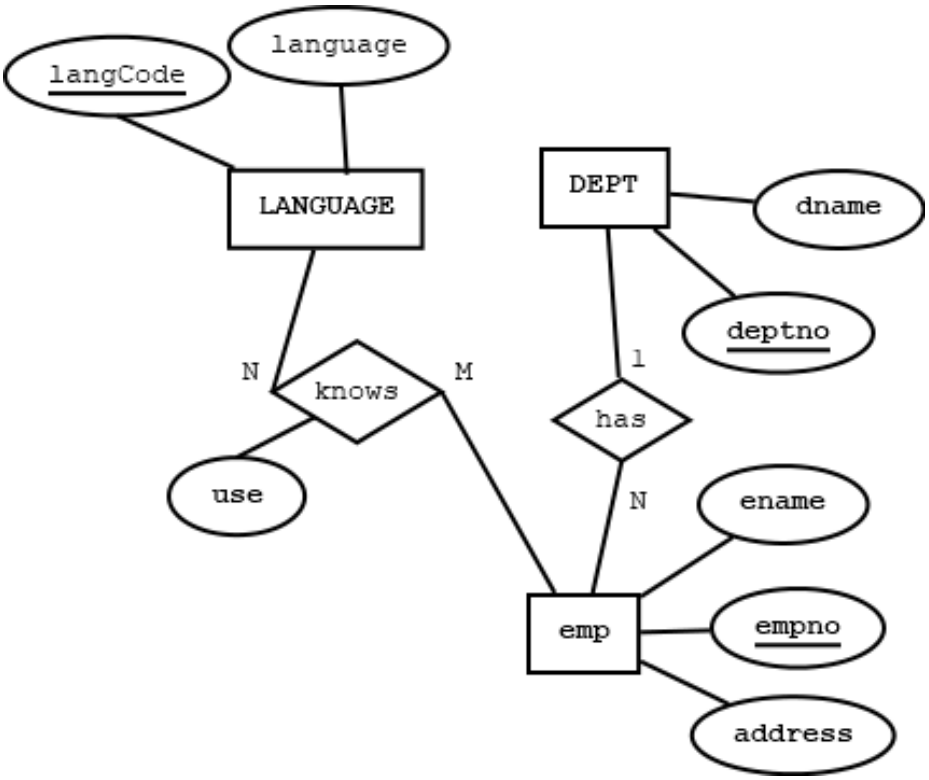
Όπως είδαμε ήδη και στο κεφάλαιο 1, το μοντέλο οντοτήτων συσχετίσεων (Entity Relationship model-ERM) είναι μια διαγραμματική τεχνική αναπαράστασης οντοτήτων, συσχετίσεων και των χαρακτηριστικών τους. Προτάθηκε από τον Peter Chen για τη μοντελοποίηση των στοιχείων-δεδομένων ενός συστήματος και αποτελεί αντικείμενο έρευνας μέχρι σήμερα. Στη συνέχεια δίνονται παραδείγματα μοντέλων. Στην Εικόνα 6.11 βλέπουμε βάση δεδομένων των υπαλλήλων που γνωρίζουν ξένες γλώσσες. Στην Εικόνα 6.12 βλέπουμε επέκταση της βάσης αυτής ώστε να καταχωρίζονται και τα στοιχεία των τέκνων των υπαλλήλων.

Τα δομικά στοιχεία του ΜΟΣ είναι τα εξής:

1) ορθογώνιο παραλληλόγραμμο που συμβολίζει μια οντότητα.
2) ρόμβος που συμβολίζει μια συσχέτιση.
3) έλλειψη που συμβολίζει ένα χαρακτηριστικό μιας οντότητας ή μιας συσχέτισης.
4) έλλειψη με υπογραμμισμένη τη λέξη που συμβολίζει κλειδί οντότητας.
5) ορθογώνιο παραλληλόγραμμο με διπλή γραμμή που συμβολίζει εξαρτώμενη οντότητα (weak entity)
6) ρόμβος με διπλή γραμμή που συμβολίζει μια καθορίζουσα συσχέτιση (identifying relationship)
7) Διπλές γραμμές σύνδεσης οντοτήτων με συσχετίσεις που καθορίζουν τη συμμετοχή οντότητας στη συσχέτιση

Τα σύμβολα φέρουν το όνομα της αντίστοιχης οντότητας, της συσχέτισης κ.λπ. και οι οντότητες και οι συσχετίσεις συνδέονται μεταξύ τους με ευθύγραμμα τμήματα (βλέπε και τις εικόνες 6.11, 6,12, 6,13). Τέλος, στο διάγραμμα αναγράφεται το είδος των σχέσεων:

1:N, M:1, 1:1, M:N, M:N:N



Εικόνα 6.11 Μοντελοποίηση βάσης δεδομένων των υπαλλήλων που γνωρίζουν ξένες γλώσσες

6.1.9 Δυσκολίες στην κατασκευή του μοντέλου

Στις εφαρμογές συχνά είναι δύσκολο να προσδιοριστούν οι σωστές οντότητες και συσχετίσεις. Στο παράδειγμά μας μπορεί ενδεχομένως κάποιος να μπορούσε να κάνει μια από τις παρακάτω επιλογές:

Επιλογή 1 (λανθασμένη)

Υπάρχει μόνο μία οντότητα και καμία συσχέτιση:

EMPLOYEE(EMPNO, ENAME, ADDRESS, DEPTNO, DNAME, LANGCODE, LANGUAGE, USE)

Επιλογή 2 (λανθασμένη)

Μπορούμε να ορίσουμε ως Οντότητες:

EMPL(EMPNO, ENAME, ADDRESS, DEPTNO, LANGCODE, LANGUAGE, USE)

DEPARTMENT(DEPTNO, DNAME)

και τη συσχέτιση:

DEPARTMENT --has--> EMPL, τύπου 1:N

Επιλογή 3

Μπορούμε να ορίσουμε ως Οντότητες:

EMP(EMPNO, ENAME, ADDRESS)

DEPT(DEPTNO, DNAME)

LANGUAGE(LANGCODE, LANGUAGE)

και τις συσχετίσεις:

DEPT --has----> EMP, 1:N

EMP --knows--> LANGUAGE, M:N.

Η συσχέτιση αυτή έχει και το χαρακτηριστικό USE.

Από τις επιλογές αυτές (1)-(3), η τρίτη επιλογή είναι η ορθή και τελικά, η βάση δεδομένων στην περίπτωση αυτή είναι:

EMP(EMPNO, ENAME, ADDRESS, DEPTNO), PK=empno, FK=deptno

DEPT(DEPTNO, DNAME), OK=deptno

LANGUAGE(LANGCODE, LANGUAGE), PK=langcode

KNOWS(EMPNO, LANGCODE, USE), PK=(empno, langcode), FK=empno, FK=langcode

Το πρόβλημα είναι ότι κάθε επιλογή οδηγεί σε διαφορετική οργάνωση της βάσης δεδομένων και πρέπει κάθε φορά να είμαστε σίγουροι για την ορθότητα της επιλογής μας.

Στην Εικόνα 6.12 ενδιαφερόμαστε και για τα παιδιά του υπαλλήλου. Επομένως, οι οντότητες είναι EMP, DEPT, LANGUAGE. Η οντότητα CHILD ονομάζεται εξαρτώμενη ή ασθενής (weak entity) και αποτελεί ειδική περίπτωση. Έχει νόημα η ύπαρξή της πάντα σε συνδυασμό (ταυτόχρονα) με την καθορίζουσα οντότητα

EMP. Δηλαδή, αν δεν κρατάμε στοιχεία πια για έναν υπάλληλο (EMP) δεν έχει νόημα να κρατάμε στοιχεία για τα παιδιά (CHILD) του.

Οι συσχετίσεις είναι:

DEPT -- has----> EMP, 1:N

EMP -- knows--> LANGUAGE, M:N.

Η συσχέτιση αυτή θα μπορούσε να έχει και το χαρακτηριστικό USE.

EMP – has -- > CHILD, 1:N.

Η συσχέτιση αυτή είναι καθορίζουσα συσχέτιση (identifying relationship) και συνδέει την εξαρτώμενη οντότητα CHILD με την οντότητα EMP

Ερώτηση

Τι θα συμβεί αν και οι δύο γονείς του παιδιού εργάζονται στην ίδια εταιρεία;

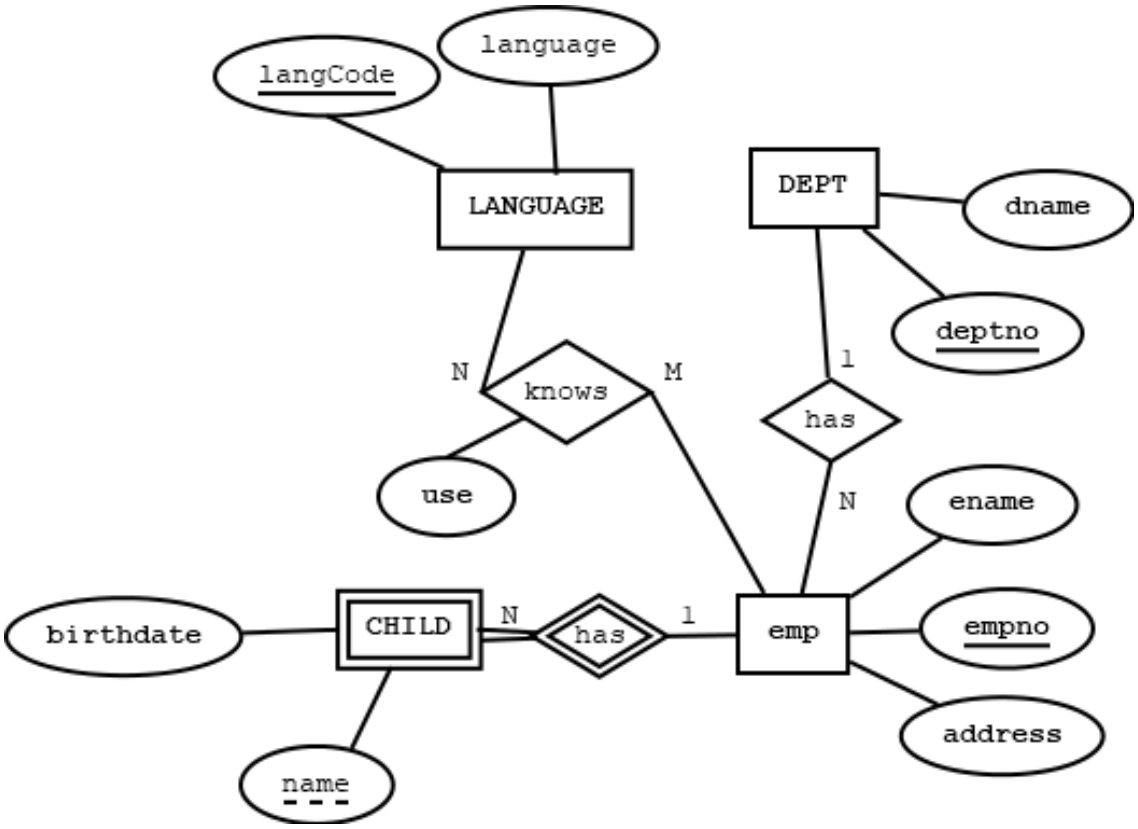
Τα χαρακτηριστικά οντοτήτων και συσχετίσεων είναι:

EMPNO, ENAME, ADDRESS, DEPTNO, DNAME, LANGCODE, LANGUAGE, USE, CHILDNAME, CHILDBIRTHDATE

Η διπλή γραμμή δηλώνει ότι δεν υπάρχει παιδί χωρίς γονέα υπάλληλο από τον οποίο εξαρτάται. Η διατύπωση είναι:

Η εξαρτώμενη οντότητα CHILD συμμετέχει καθολικά στην καθορίζουσα συσχέτιση has.

Η απλή γραμμή δηλώνει μερική συμμετοχή. Στο ΜΟΣ η οντότητα EMP έχει μερική συμμετοχή στην καθορίζουσα συσχέτιση has.



Εικόνα 6.12 Βάση δεδομένων υπαλλήλων, των ξένων γλωσσών που γνωρίζουν, και των τέκνων τους

Τελικά, η βάση δεδομένων στην περίπτωση αυτή είναι:

EMP(EMPNO, ENAME, ADDRESS, DEPTNO), PK=empno, FK=deptno

DEPT(DEPTNO, DNAME), PK=deptno

LANGUAGE(LANGCODE, LANGUAGE), PK=langcode

KNOWS(EMPNO, LANGCODE, USE), PK=(empno, langcode), FK=empno, FK=langcode

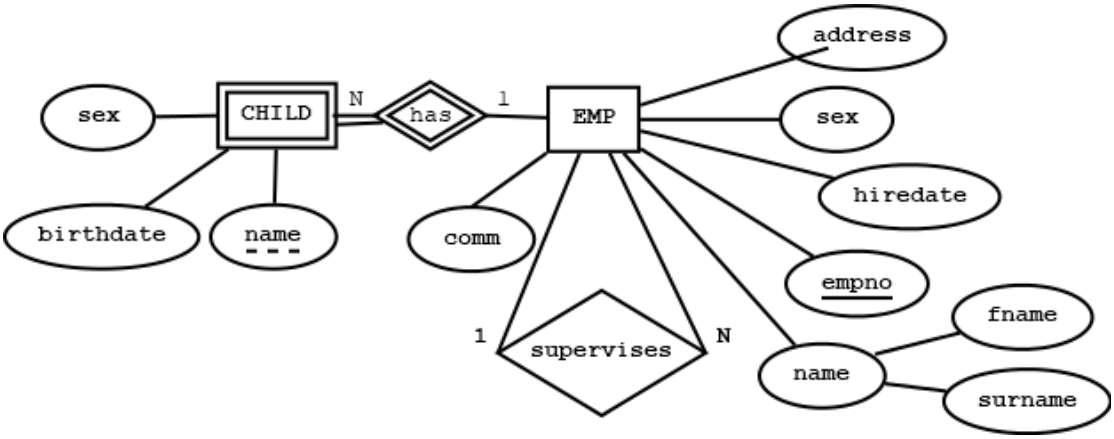
CHILD(EMPNO, CHILDNAME, BIRTHDATE), PK=(empno, childname), FK=empno.

6.1.10 Συνταγή μετάβασης από το μοντέλο Οντοτήτων Συσχετίσεων σε σχεσιακή βάση δεδομένων

Για τη μετάβαση από το μοντέλο οντοτήτων - συσχετίσεων σε πίνακες ακολουθήστε τους εξής κανόνες:

- **Κανόνας 1**: Για κάθε οντότητα κατασκευάζετε έναν πίνακα που θα περιλαμβάνει τουλάχιστον όλα τα χαρακτηριστικά (attributes) της. Το κύριο κλειδί της οντότητας θα είναι και κύριο κλειδί του πίνακα που θα αναπαριστά την οντότητα
- **Κανόνας 2**: Αν η συσχέτιση σ είναι τύπου 1:N και έχει χαρακτηριστικά τότε (για τη συσχέτιση σ) δεν κατασκευάζεις ξεχωριστό πίνακα. Απλά προσθέτεις στον πίνακα της δεύτερης οντότητας το κλειδί της πρώτης οντότητας (ως ξένο κλειδί) και προσθέτεις και τα χαρακτηριστικά της συσχέτισης.
- **Κανόνας 3:** Αν η συσχέτιση σ είναι τύπου 1:1 τότε τη διαχειρίζεστε σαν να ήταν τύπου 1:N, μόνο προς τη μία κατεύθυνση.
- **Κανόνας 4**: Αν η συσχέτιση είναι M:N τότε κατασκευάζεις ξεχωριστό πίνακα που περιλαμβάνει ως σύνθετο κλειδί τα κλειδιά των δύο οντοτήτων. Στον πίνακα περιλαμβάνονται και τα χαρακτηριστικά της συσχέτισης. Τα κλειδιά των δύο οντοτήτων που αποτελούν το σύνθετο κλειδί είναι ξένα κλειδιά στον πίνακα.
- **Κανόνας 5:** Αν μια συσχέτιση σ συνδέει παραπάνω από δύο οντότητες (π.χ. είναι τριαδική τύπου M:N:N) τότε για τη συσχέτιση αυτή συνήθως κατασκευάζουμε ξεχωριστό πίνακα. Ο πίνακας έχει σύνθετο κύριο κλειδί που αποτελείται από τα κλειδιά των οντοτήτων που συμμετέχουν στη συσχέτιση. Τα κλειδιά των άλλων οντοτήτων είναι ξένα κλειδιά για τον πίνακα της συσχέτισης.
- Σε κάποιες περιπτώσεις αντί του σύνθετου κύριου κλειδιού υιοθετούμε ένα κύριο κλειδί που είναι αύξων αριθμός (accno) ή ένα «τεχνητό κύριο κλειδί» από αυτά που κατασκευάζουν και διαχειρίζονται τα ΣΔΒΔ. Στην περίπτωση αυτή τα κλειδιά των άλλων οντοτήτων παραμένουν ως ξένα κλειδιά στον πίνακα της συσχέτισης και ενδεχομένως αποτελούν δευτερεύον κλειδί (secondary key, index). Επιπλέον, τότε προσθέτουμε στο ΜΟΣ στη συσχέτιση και το χαρακτηριστικό accno υπογραμμισμένο.
- **Κανόνας 6:** Αν μια οντότητα είναι εξαρτώμενη κατασκευάζετε έναν πίνακα που θα περιλαμβάνει τουλάχιστον όλα τα χαρακτηριστικά (attributes) της αλλά και το κλειδί της οντότητας από την οποία εξαρτάται. Το κύριο κλειδί του πίνακα της εξαρτώμενης οντότητας θα είναι σύνθετο και θα περιλαμβάνει το μερικό κλειδί της και το κύριο κλειδί της οντότητας από την οποία εξαρτάται.
- **Κανόνας 7:** Αν μια οντότητα είναι αυτοπαθής (δηλαδή, συσχετίζει την οντότητα με τον εαυτό της) τότε στον πίνακα της οντότητας προστίθεται κατάλληλη στήλη που εκφράζει τη συσχέτιση.

Ακολουθεί παράδειγμα (Εικόνα 6.13) που ξεκαθαρίζει τους κανόνες 6, 7! Παρατηρήστε, επίσης, την απλή και τη διπλή γραμμή που συνδέει τις οντότητες EMP, CHILD με την καθορίζουσα συσχέτιση has



Εικόνα 6.13 ΜΟΣ στο οποίο θα εφαρμοστούν οι Κανόνες 6 και 7

Emp(empno, surname, fname, **mgr**, hiredate, address, sex, …), primary key=empno

Οι στήλες empno, mgr έχουν τον ίδιο τύπο δεδομένων και μήκος. Για κάθε υπάλληλο η στήλη empno έχει τον κωδικό του και η στήλη mgr τον κωδικό του manager του.

Child(empno, name, birthdate, sex), primary key=(empno, name), foreign key=empno

Με τους κανόνες αυτούς δεν εξαντλούνται όλες οι περιπτώσεις αλλά καλύπτονται οι πιο συνηθισμένες. Οι κανόνες, αν εφαρμοστούν σωστά, μας οδηγούν σε μία βάση δεδομένων καλά οργανωμένη, που όπως λέμε βρίσκεται στην **Τρίτη Κανονική Μορφή (3NF)**. Βέβαια, η βασική προϋπόθεση, αλλά και το δυσκολότερο έργο, είναι να σχεδιαστεί σωστά το μοντέλο μας. Η διαδικασία αυτή της μεταγραφής του μοντέλου σε πίνακες (σε 3NF) είναι μία εναλλακτική δυνατότητα στη διαδικασία κανονικοποίησης. Η **κανονικοποίηση** θα παρουσιαστεί αναλυτικά και με άλλους τρόπους στην εργασία μας στην επόμενη ενότητα.

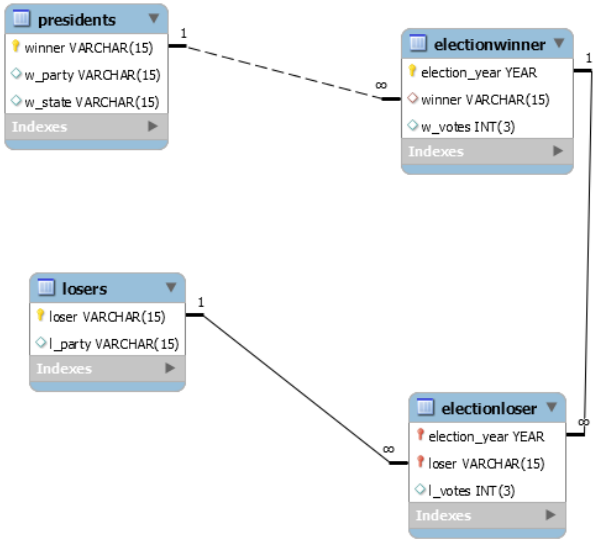
6.1.11 Παράδειγμα Μεταγραφής Μοντέλου Οντοτήτων Συσχετίσεων των αμερικανικών προεδρικών εκλογών σε Τρίτη Κανονική Μορφή

Έστω και πάλι η βάση δεδομένων των αμερικανικών εκλογών που είδαμε και στο κεφάλαιο 1 (δείτε και Εικόνα 1.3.6 με δείγμα δεδομένων των εκλογικών αναμετρήσεων 1952-. Στην Εικόνα 6.14 δίδεται δείγμα εκλογών από το 1952 έως το 1976 και επιπλέον ΜΟΣ σχεδιασμένο στο εργαλείο MySQL Workbench. Το μοντέλο ακολουθεί την προσέγγιση “connect to columns” και είναι λίγο διαφορετικό από αυτό που παρατέθηκε στο προηγούμενο κεφάλαιο. Στην Εικόνα 6.15 παρατίθεται το ίδιο μοντέλο στη γνωστή από την τεχνολογία λογισμικού μεθοδολογία Unified Modeling Language (UML). Παρατηρήστε στο δείγμα ότι σε κάποιες περιπτώσεις επειδή ίσως δε γνωρίζουμε τις τιμές γράψαμε NULL. Στην πραγματικότητα, στην περίπτωση αυτή, δεν αποθηκεύεται κάποια τιμή στη βάση δεδομένων. Απλά με τη λέξη NULL «οπτικοποήσαμε» το γεγονός ότι δεν υπάρχει τιμή αποθηκευμένη στη βάση.

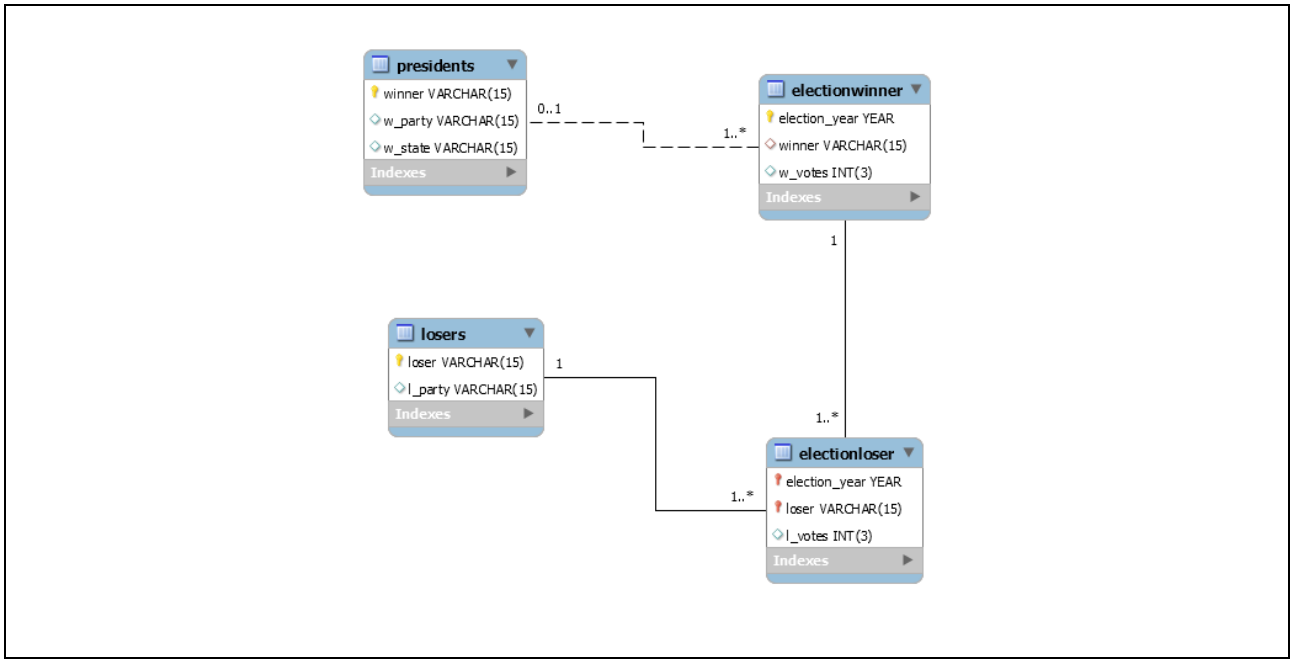
Τιμή NULL (τίποτα) σε έναν πίνακα σημαίνει ότι η αντίστοιχη στήλη δεν έχει τιμή για κάποιο λόγο, π.χ., δε γνωρίζουμε την τιμή ή δεν έχει τιμή ή δεν καταχωρίζουμε τιμή μέχρι στιγμής. Η διαχείριση των τιμών NULL απαιτεί ιδιαίτερη προσοχή στις πραγματικές εφαρμογές, όπως θα δούμε στην ενότητα παρουσίασης της γλώσσας SQL. Επισημαίνουμε, επίσης, ότι οι μηδενικές τιμές δεν είναι τιμές NULL.

Πίνακας αμερικανικών προεδρικών εκλογών

YEAR	WINNER	W_VOTES	W_PARTY	W_STATE	LOSER	L_VOTES	L_PARTY
1952	EISENHOWER	442	REPUBLICAN	TEXAS	STEVENSON	89	DEMOCRAT
1956	EISENHOWER	457	REPUBLICAN	TEXAS	STEVENSON	73	DEMOCRAT
1960	KENNEDY	303	DEMOCRAT	MASS	NIXON	219	REPUBLICAN
1964	JOHNSON	486	DEMOCRAT	TEXAS	GOLDWATER	52	REPUBLICAN
1968	NIXON	301	REPUBLICAN	CALIFORNIA	HUMPHREY	191	DEMOCRAT
1968	NIXON	301	REPUBLICAN	CALIFORNIA	WALLACE	46	INDEPENDENT
1972	NIXON	520	REPUBLICAN	CALIFORNIA	McGOVERN	17	DEMOCRAT
1976	CARTER	297	DEMOCRAT	**NULL**	FORD	240	REPUBLICAN



Εικόνα 6.14 Δείγμα δεδομένων και ΜΟΣ των αμερικανικών προεδρικών εκλογών.



```
DROP DATABASE IF EXIST presidents_elections;
CREATE DATABASE presidents_elections;
USE presidents_elections;
CREATE TABLE presidents(winner VARCHAR(15) NOT NULL,
w_party VARCHAR(15), w_state VARCHAR(15) ,
   PRIMARY KEY(winner));
CREATE TABLE losers(loser VARCHAR(15) NOT NULL,
 l_party VARCHAR(15),
 PRIMARY KEY(loser));
CREATE TABLE electionwinner(election_year  YEAR  NOT NULL,
winner VARCHAR(15),w_votes INT(3),
PRIMARY KEY(election_year),
 FOREIGN KEY(winner) REFERENCES presidents(winner) );
CREATE TABLE electionloser(election_year YEAR NOT NULL,
loser VARCHAR(15) NOT NULL, l_votes INT(3),
   PRIMARY KEY(election_year, loser),
   FOREIGN KEY(election_year) REFERENCES electionwinner(election_year),
   FOREIGN KEY(loser) REFERENCES losers(loser));
```

Εικόνα 6.15 UML μοντέλο, όπως σχεδιάστηκε στο προϊόν Mysql workbench και δηλώσεις δημιουργίας της βάσης

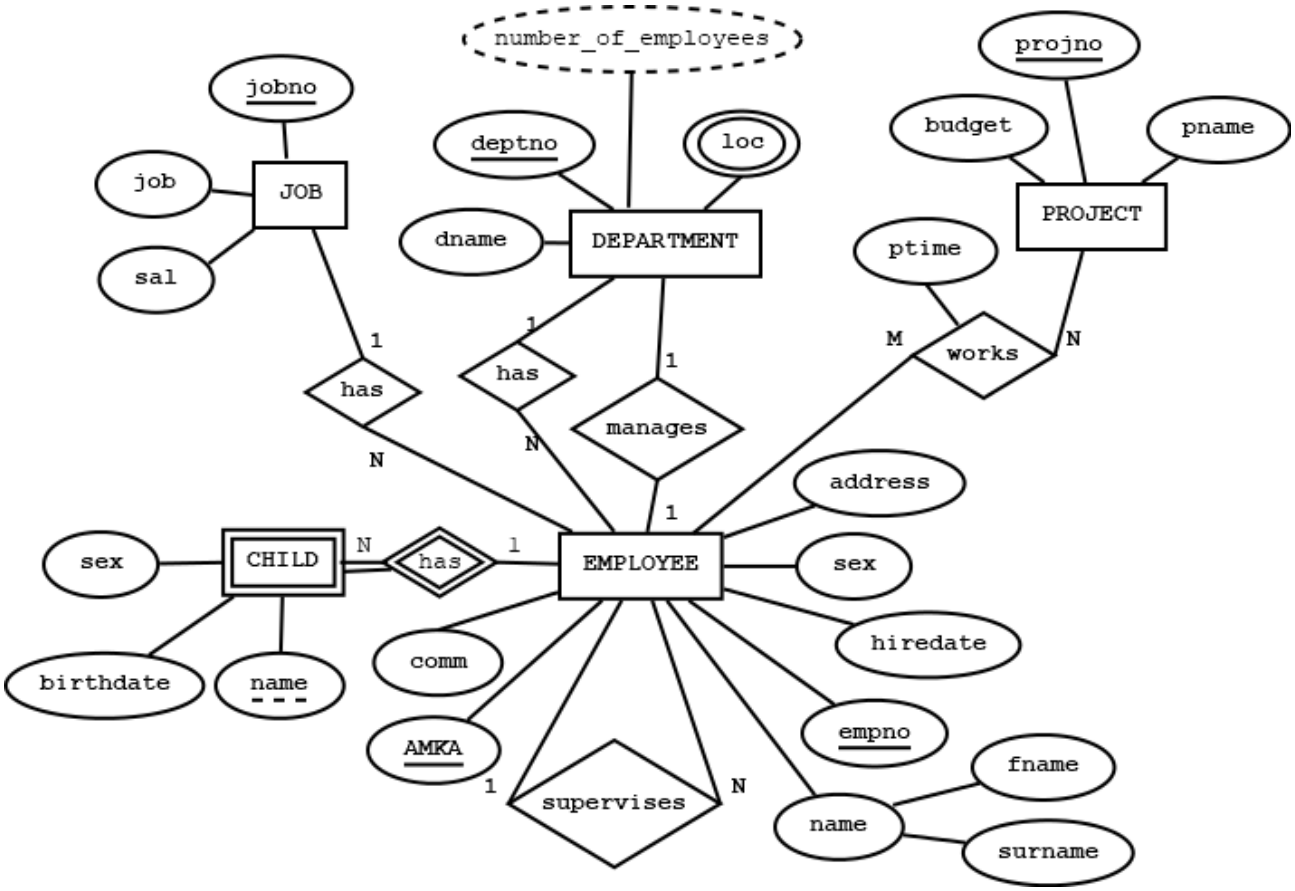
Παρατηρήστε ότι στα μοντέλα των σχημάτων 6.14, 6.15 φαίνεται άμεσα η τρίτη κανονική μορφή των αμερικανικών προεδρικών εκλογών. Στην Εικόνα 6.15 βλέπουμε τον ορισμό της βάσης δεδομένων στο προϊόν MySQL.

6.1.12 Μοντέλο οντοτήτων συσχετίσεων εταιρείας

Στη βιβλιογραφία προτάθηκαν και προτείνονται τροποποιήσεις και επεκτάσεις του μοντέλου οντοτήτων-συσχετίσεων. Το σύγγραμμα των Elmasri και Navathe (υπάρχει και στα ελληνικά) και άλλα γνωστά συγγράμματα είναι μία χρήσιμη πηγή υλικού, μελέτης και αναφοράς για τον ενδιαφερόμενο αναγνώστη στην μοντελοποίηση.

Στην Εικόνα 6.16 παρατίθεται το Μοντέλο Οντοτήτων Συσχετίσεων μιας εταιρείας. Η Εικόνα αυτό, θα αποτελέσει την αφετηρία μίας πληρέστερης παρουσίασης του μοντέλου.

Στη βιβλιογραφία περιγράφονται κάποιες τροποποιήσεις στο μοντέλο του Peter Chen που προσθέτουν στο μοντέλο αντικειμενοστρεφή χαρακτηριστικά. Το μοντέλο οντοτήτων συσχετίσεων με τα ενσωματωμένα αντικειμενοστρεφή χαρακτηριστικά είναι γνωστό στη βιβλιογραφία ως επέκταση μοντέλου οντοτήτων συσχετίσεων (Enhanced Entity - Relationship model) και θα περιγραφεί σε επόμενη ενότητα.



Εικόνα 6.16 Μοντέλο Οντοτήτων Συσχετίσεων Εταιρείας

6.1.12.1 Περιγραφή εννοιών και συμβολισμοί

1) Οντότητες

Η οντότητα αναπαρίσταται με αναγραφή λέξης (ουσιαστικού) σε ορθογώνιο παραλληλόγραμμο. Οντότητες είναι: JOB, DEPARTMENT, PROJECT, EMPLOYEE. Εξαρτώμενη οντότητα είναι: CHILD.

Χαρακτηριστικά – Είδη χαρακτηριστικών

Τα χαρακτηριστικά (attributes) μίας οντότητας μπορούν να υποδιαιρεθούν σε (βλέπε και Εικόνα 6.16):

(Υποψήφια) Κύρια Κλειδιά (candidate key attribute)

Θα εξετάσουμε τα υποψήφια κύρια κλειδιά για την οντότητα EMPLOYEE:

- AMKA αριθμός κοινωνικής ασφάλισης (social security number) που είναι απλό χαρακτηριστικό
- Empno ένας αύξων αριθμός υπαλλήλου που είναι επίσης απλό χαρακτηριστικό
- (Surname, fname) σύνθετο χαρακτηριστικό που αποτελείται από το επώνυμο και το όνομα. Θα μπορούσε να θεωρηθεί ως υποψήφιο κύριο κλειδί στην περίπτωση που δεν έχουμε συνωνυμίες. Όχι και πολύ σωστή υπόθεση!.

Από τα υποψήφια κύρια κλειδιά πρέπει να επιλεγεί ακριβώς ένα ως κύριο κλειδί.

Απλά χαρακτηριστικά (attributes)

Στην Εικόνα 6.16 για την οντότητα EMPLOYEE τα χαρακτηριστικά είναι: ΑΜΚΑ, empno, surname, fname, hiredate, sex, address, comm

Σύνθετα χαρακτηριστικά (composite attribute)

Στην Εικόνα 6.16 σύνθετο χαρακτηριστικό, για την οντότητα EMPLOYEE, είναι το Name. Αναλύεται σε (surname, fname).

Χαρακτηριστικά με πολλαπλές τιμές (multivalued attribute)

Στην Εικόνα 6.16 είναι το χαρακτηριστικό loc για την οντότητα Department.

Υπολογιζόμενα/Παραγώμενα χαρακτηριστικά (derived attribute)

Η τιμή αυτού του χαρακτηριστικού μπορεί να υπολογιστεί από τα στοιχεία που είναι αποθηκευμένα στη βάση δεδομένων.

Στην Εικόνα 6.16 είναι το χαρακτηριστικό Number_of_Employees της οντότητας DEPARTMENT

Στην οντότητα του υπαλλήλου και στην οντότητα του τμήματος μπορούμε να αντιστοιχίσουμε τους εξής πίνακες:

EMPLOYEE**(<u>ΑΜΚΑ/empno</u>**, fname, surname, hiredate, address, sal, comm), PK=ΑΜΚΑ ή empno.

Προσοχή! Ο πίνακας EMPLOYEE θα αλλάξει στη συνέχεια λόγω των συσχετίσεων.

DEPARTMENT(deptno, dname, number_of_employees), PK=deptno

Για το χαρακτηριστικό με πολλαπλές τιμές προσθέτουμε πίνακα:

DEPT_LOC(**deptno, loc**), PK=(deptno, loc), FK=deptno.

Το παραγόμενο χαρακτηριστικό number_of_employees μπορεί να προστεθεί στον πίνακα DEPARTMENT ή να μην προστεθεί σε πίνακα της βάσης δεδομένων. Θεωρητικά είναι προτιμότερο να μην προστεθεί στον πίνακα DEPARTMENT. Όμως στις εφαρμογές καλύτερα να υπάρχει στον πίνακα και να υπολογίζεται αυτόματα η τιμή του με χρήση trigger.

Να και οι πίνακες που αντιστοιχούν στις υπόλοιπες οντότητες.

JOB(**jobno**, job, sal), PK=jobno

PROJECT(**projno**, pname, budget), PK=projno

2) Συσχετίσεις

Ο συνήθης συμβολισμός για τη συσχέτιση είναι ρόμβος με ένα ρήμα.

Συσχετίσεις 1:N

Οι συσχετίσεις DEPARTMENT – has (1:N) – EMPLOYEE και JOB -– has (1:N) – EMPLOYEE αλλάζουν τον πίνακα EMPLOYEE.

EMPLOYEE**(<u>ΑΜΚΑ/empno</u>**, fname, surname, hiredate, address, sal, comm, deptno, jobno), PK=ΑΜΚΑ ή empno, FK=deptno, FK=jobno.

Συσχετίσεις 1:1

Η συσχέτιση EMPLOYEE – manages (1:1) – DEPARTMENT μπορεί να θεωρηθεί ως συσχέτιση τύπου 1:N, μόνο ως προς τη μία κατεύθυνση.

Έστω EMPLOYEE – manages (1:N) – DEPARTMENT. Η συσχέτιση αλλάζει τον πίνακα DEPARTMENT.

DEPARTMENT(deptno, dname, dept_mgr, number_of_employees), PK=deptno

Συσχετίσεις M:N

Η συσχέτιση EMPLOYEE – works (M:N) – PROJECT οδηγεί σε νέο πίνακα. Η συσχέτιση έχει το χαρακτηριστικό ptime.

WORKS(empno, projno, ptime), PK=(empno, projno), FK=empno, FK=projno

Αυτοπαθείς Συσχετίσεις

Η συσχέτιση EMPLOYEE – supervises (1:N) – EMPLOYEE αλλάζει τον πίνακα EMPLOYEE.

EMPLOYEE**(<u>ΑΜΚΑ/empno</u>**, fname, surname, mgr, hiredate, address, sal, comm, deptno, jobno), PK=ΑΜΚΑ ή empno, FK=deptno, FK=jobno.

Συσχετίσεις καθορίζουσες (identifying relationship) κάποιες οντότητες

Για τις συσχετίσεις αυτές χρησιμοποιούμαι το διπλό ρόμβο με αναγραφή φράσης, συνήθως, με κάποιο ρήμα.

πχ. EMP -- 1 – has – N – CHILD

δηλαδή, τα στοιχεία ενός εξαρτώμενου/προστατευόμενου μέλους της οικογένειας (π.χ., όνομα, ημερομηνία γέννησης κ.λπ.) μας ενδιαφέρουν/έχει νόημα να αποθηκεύονται στη βάση δεδομένων μόνο αν έχουμε τα στοιχεία του υπαλλήλου. Για να γίνει σαφέστερο, δεν έχει νόημα να κρατάμε σαν στοιχείο το όνομα και την ηλικία ενός παιδιού (ΜΕΛΙΝΑ, 11) αν δεν έχουμε ως υπάλληλο στην εταιρεία τον γονέα της. Οι συσχετίσεις αυτού του τύπου μας οδηγούν στον ορισμό της έννοιας της ασθενούς οντότητας.

Ασθενείς (ή αδύναμες ή εξηρτημένες, weak) οντότητες και καθορίζουσες συσχετίσεις

Ασθενείς ή εξαρτώμενες είναι οι οντότητες (βλέπε Εικόνα 6.16) οι οποίες έχουν νόημα μόνο όταν υπάρχει μια άλλη οντότητα από την οποία εξαρτώνται.

Η οντότητα CHILD είναι ασθενής οντότητα.

Αν θέλαμε να αναπαραστήσουμε με πίνακες την οντότητα και την εξαρτώμενή της συνήθως προσθέτουμε στον πίνακα της εξαρτώμενης το κλειδί της πρώτης για να σχηματίσουμε ένα σύνθετο κύριο κλειδί της εξαρτώμενης οντότητας. Αν η καθορίζουσα συσχέτιση έχει χαρακτηριστικό τότε το χαρακτηριστικό αυτό θα είναι στήλη στον πίνακα της εξαρτώμενης οντότητας.

Στο παράδειγμά μας ισχύει:

EMPLOYEE(empno, AMKA, fname, surname, ...), PK=AMKA ή empno

CHILD(**<u>AMKA/empno, NAME</u>**, birthdate, sex),

PK=(AMKA/empno, name), FK=AMKA ή empno

Αν θέλουμε να δούμε και παράδειγμα αντίστοιχων γραμμών:

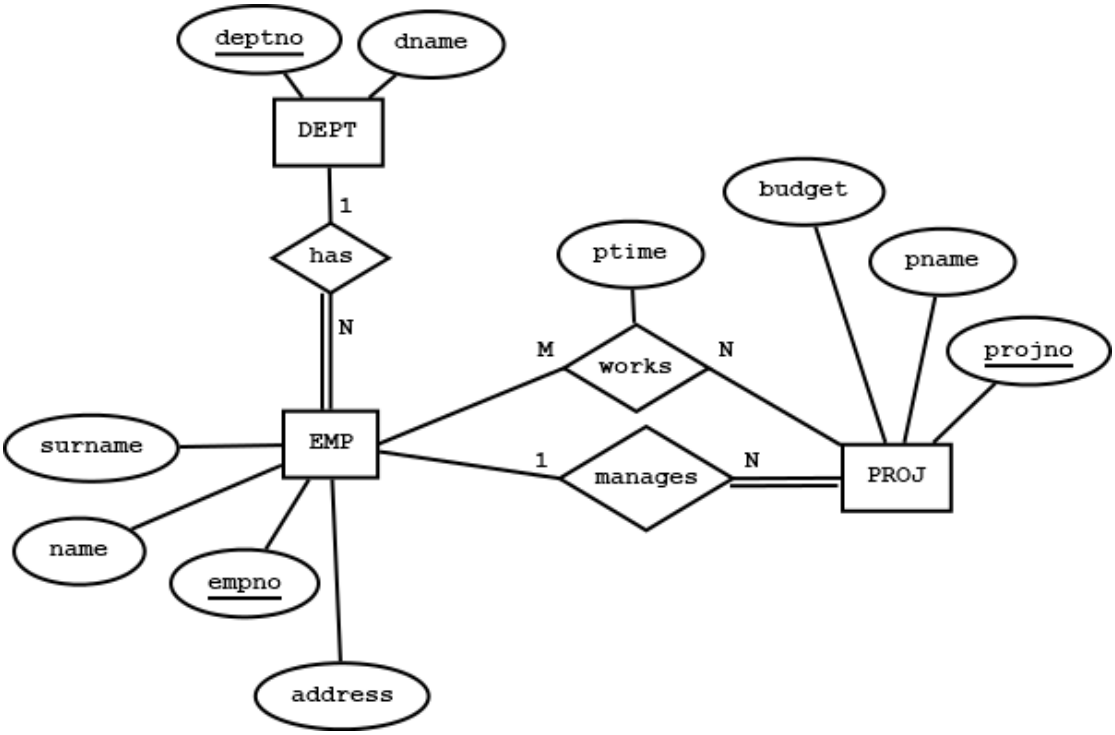
Ο υπάλληλος (1234, CLARKE, ...) έχει ένα παιδί που περιγράφεται ως εξής:

(1234, JOHN, '21/12/2017', M)

Άλλες περιπτώσεις συσχετίσεων

Υπάρχουν ακόμη οι παρακάτω περιπτώσεις, όπως φαίνεται στην Εικόνα 6.17:

Συσχετίσεις και ολική ή μερική συμμετοχή σε αυτές (total participation)



Εικόνα 6.17 Συσχετίσεις και ολική ή μερική συμμετοχή σε αυτές (manages.dia)

Στην Εικόνα 6.17 ισχύει για τις συσχετίσεις:

DEPT – has (1:N) EMP, όπου δηλώνεται η ολική συμμετοχή της οντότητας EMP στη συσχέτιση

Επίσης, EMP – manages (1:N) PROJ.

Στο παράδειγμα δηλώνεται ότι κάθε υπάλληλος εργάζεται σε κάποιο τμήμα και κάθε τμήμα μπορεί να έχει ή να μην έχει υπαλλήλους. Στο παράδειγμα, δηλώνεται επίσης ότι κάθε έργο έχει οπωσδήποτε ένα υπάλληλο επικεφαλής (ολική συμμετοχή της οντότητας PROJ στη συσχέτιση manages) ενώ ένας υπάλληλος δεν είναι οπωσδήποτε επικεφαλής ενός έργου (μερική συμμετοχή της οντότητας EMP στη συσχέτιση manages).

Συσχετίσεις με αναφορά των περιορισμών της συμμετοχής (structural constraint on participation): Μηχανισμός (min, max)

π.χ. **EMP** – (0,1) – **manages** – (1,1) – PROJ

δηλαδή, ένας υπάλληλος μπορεί να είναι επικεφαλής το πολύ σε ένα έργο, ενώ ένα έργο έχει ακριβώς ένα επικεφαλής.

π.χ. **EMP** – (1,N) – **works_on** – (1,N) – PROJ

δηλαδή, ένας υπάλληλος μπορεί να εργάζεται σε ένα ή περισσότερα έργα, ενώ σε ένα έργο εργάζεται ένας ή περισσότεροι υπάλληλοι.

π.χ. **EMP** – (1,1) – **works_on** – (5,N) – DEPT

δηλαδή, ένας υπάλληλος μπορεί να εργάζεται σε ένα ακριβώς τμήμα, ενώ σε κάθε τμήμα εργάζονται 5 ή περισσότεροι υπάλληλοι.

Ενδεικτική βιβλιογραφία και αναφορές

Peter Chen (March 1976) The Entity-Relationship Model-Toward a Unified View of Data. ACM Transactions on Database Systems 1(1), 9–36. doi:10.1145/320434.320440. **Εδώ ξεκίνησαν όλα !**

Peter Chen (2002) Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned in Software pioneers, pp. 296-310, Springer-Verlag, ISBN 3-540-43081-4

Συγγράμματα που γνώρισαν/γνωρίζουν πολλές εκδόσεις

Date, C.J., An introduction to database systems, vol.1, vol.2, Addison-Wesley.

Ullman, J.D., Widom, J., A first course in database systems, Prentice-Hall.

Elmasri, R., Navathe, S. B., Fundamentals of Database Systems, The Benjamin/Cummings Publishing Co.

Martin, J., Computer data-base organization, Prentice-Hall.

Ramakrishnan, R., Gehrke, J., Database management systems, McGraw-Hill

Βιβλία βάσεων δεδομένων στα ελληνικά που χρησιμοποιήθηκαν

Jeffrey D. Ullman, Jennifer Widom, Βασικές Αρχές για τα Συστήματα Βάσεων Δεδομένων, Κλειδάριθμος

Ramakrishnan Raghu, Gehrke Joahannes, Συστήματα Διαχείρισης Βάσεων Δεδομένων, Τζιόλας

Elmasri Ramez, Navathe Shamkant B., Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων, Δίαυλος

Korth Henry, Silberschatz Abraham, Sudarshan S., Συστήματα Βάσεων Δεδομένων- Η Πλήρης Θεωρία των Βάσεων Δεδομένων, Γκιούρδας

Χρήστος Σκουρλάς, Σχεσιακές Βάσεις Δεδομένων, Εκδόσεις Νέων Τεχνολογιών

Χρήστος Σκουρλάς, Υλοποίηση Εφαρμογών με Γλώσσα SQL, Εκδόσεις Νέων Τεχνολογιών

Πρόσθετη βιβλιογραφία

Bagui, S., & Earp, R. (2003). Database design using entity-relationship diagrams (2nd ed.). CRC Press. ISBN 978-1-4398-6176-9

Rolland, F. D. (1998) The essence of databases, Prentice-Hall

Teorey, T. J., Yang, D., & Fry, J. P. (1986). A logical design methodology for relational databases using the extended entity-relationship model. ACM Computing Surveys (CSUR), 18(2), 197-222. dl.acm.org

(A logical design methodology for relational databases using the extended entity-relationship model)

Alhajj, R. (2003). Extracting the extended entity-relationship model from a legacy relational database. Information systems, 28(6), 597-618.

Ordonez, C., Maabout, S., Matusevich, D. S., & Cabrera, W. (2014). Extending ER models to capture database transformations to build data sets for data mining. Data & Knowledge Engineering, 89, 38-54.

Lovison, A., & Rigoni, E. (2010). Extracting optimal datasets for metamodelling and perspectives for incremental samplings. Mathematics and Computers in Simulation, 81(3), 681-692.

Cleve, A., Gobert, M., Meurice, L., Maes, J., & Weber, J. (2015). Understanding database schema evolution: A case study. Science of Computer Programming, 97, 113-121.

Tseng, F. S., & Chen, C. L. (2008). Enriching the class diagram concepts to capture natural language semantics for database access. Data & Knowledge Engineering, 67(1), 1-29.

Belkadi, F., Notin, A., Drémont, N., & Troussier, N. (2012). A meta-model for knowledge representation in engineering design. IFAC Proceedings Volumes, 45(6), 1641-1646.

Morawski, R. Z. (2013). An application-oriented mathematical meta-model of measurement. Measurement, 46(9), 3753-3765.

Βιβλιογραφία της μεθοδολογίας-μοντελοποίησης οντοτήτων συσχετίσεων

Στη βιβλιογραφία (αλλά και σε πλήθος ερευνητικών εργασιών) η μοντελοποίηση οντοτήτων συσχετίσεων (Entity-Relationship Modeling) αντιμετωπίζεται ως μεθοδολογία (methodology) ή ως διαδικασία (ERM process) και υπάρχουν πάρα πολλά εργαλεία (tools) μοντελοποίησης.

Βλέπε για παράδειγμα

Salvaneschi, G., Ghezzi, C., & Pradella, M. (2015). ContextErlang: A language for distributed context-aware self-adaptive applications. Science of Computer Programming, 102, 20-43.

Choi, D., Kim, N., & Hung, D. T. (2012). Conceptual data modeling for realizing context-aware services. Expert Systems with Applications, 39(3), 3022-3030.

Chen, P. P. S. (1976). The entity-relationship model—toward a unified view of data. ACM transactions on database systems (TODS), 1(1), 9-36. doi:10.1145/320434.320440.

Chen, P. (2002). Entity-relationship modeling: historical events, future trends, and lessons learned. In Software pioneers (pp. 296-310). Springer, Berlin, Heidelberg, ISBN 3-540-43081-4

Bagui, S., & Earp, R. (2011). Database Design Using Entity-Relationship Diagrams (2nd ed.). CRC Press. ISBN 978-1-4398-6176- 9. (Εμπεριστατωμένη μονογραφία για τη μοντελοποίηση που βασίζεται σε ER)

6.1.13 Μελέτη περίπτωσης. Πληροφοριακό σύστημα εταιρείας ενοικίασης αυτοκινήτων

Η εταιρία ενοικιάσεως πολυτελών αυτοκινήτων «Mythical Car» εξυπηρετεί τους πελάτες της μέσω πρακτορείων. Τα κεντρικά γραφεία της και ένα πρακτορείο βρίσκονται στην πλατεία Συντάγματος. Λειτουργεί, επίσης, πρακτορείο στο αεροδρόμιο και σύντομα θα λειτουργήσει και νέο πρακτορείο στη Θεσσαλονίκη. Μετά από μελέτη της λειτουργίας της εταιρίας διαπιστώνεται ότι :

- Κάθε πρακτορείο έχει τα δικά του αυτοκίνητα
- Αν απαιτηθεί ένα πρακτορείο μπορεί να δανείσει σε άλλα πρακτορεία αυτοκίνητά του.
- Κάθε πρακτορείο έχει τους δικούς του υπαλλήλους.
- Αν απαιτηθεί ένας υπάλληλος μπορεί να εργαστεί για συγκεκριμένες ώρες και σε άλλα πρακτορεία.
- Κάθε πρακτορείο καταχωρεί τοπικά τους λογαριασμούς του.
- Τα στοιχεία των υπαλλήλων καταχωρούνται σε κεντρικό σύστημα Η/Υ.
- Οι λογιστές των κεντρικών γραφείων έχουν πρόσβαση στους λογαριασμούς των πρακτορείων και στα άλλα οικονομικά στοιχεία που καταχωρούνται τοπικά.
- Οι πελάτες καταχωρούνται στα κεντρικά γραφεία στο κεντρικό σύστημα
- Τα πρακτορεία συνδέονται με τα κεντρικά γραφεία για να προσπελάσουν τα στοιχεία των πελατών.
- Τα πρακτορεία χρησιμοποιούν στοιχεία από την τοπική βάση δεδομένων και στοιχεία από τον κεντρικό Η/Υ για να “τρέξουν” τις παρακάτω εφαρμογές:

    - RESERVATIONS: Οι πελάτες τηλεφωνικά/αυτοπροσώπως κάνουν κράτηση. Προγραμματισμός της εκχώρησης αυτοκινήτου.
    - COLLECTION & RETURN: Τα αυτοκίνητα ενοικιάζονται όταν ο πελάτης παραλαμβάνει. Σημειώνεται ένδειξη Km (κοντέρ) κ.λπ. Όταν επιστραφεί το αυτοκίνητο προετοιμάζεται ο λογαριασμός.
    - INVOICING: Έκδοση Τιμολογίων, Καταστάσεων λογαριασμών κ.λπ.
    - VEHICLE MAINTENANCE: Προγραμματισμός της κανονικής συντήρησης, παρακολούθηση βλαβών.
- Στα κεντρικά γραφεία «τρέχουν» εφαρμογές όπως:
    - PAYROLL : Μισθοδοσία υπαλλήλων
    - CUSTOMERS : Διαχείριση στοιχείων πελατών
    - EMPLOYEE : Διαχείριση στοιχείων υπαλλήλων
    - MARKETING : Παραγωγή υλικού προώθησης και αποστολή σε πελάτες.
    - STATISTICS : Στατιστικά στοιχεία.

Στη συνέχεια, θα εστιάσουμε μόνο στο πρώτο από τα παρακάτω υποσυστήματα:

- Εφαρμογές πρακτορείων («ΠΡΑΚΤΟΡΕΙΑ»)
- Κεντρικό υποσύστημα («ΚΕΝΤΡΙΚΟ»)

Πιο συγκεκριμένα για τα ΠΡΑΚΤΟΡΕΙΑ θα αναφερθούν με κάποια λεπτομέρεια τα αποτελέσματα της Ανάλυσης Δεδομένων: Χαρακτηριστικά κ.λπ.

6.1.13.1 Εφαρμογή πρακτορείων - ανάλυση δεδομένων

Μετά από την ανάλυση δεδομένων προκύπτει ότι το τοπικό σύστημα περιλαμβάνει τα εξής χαρακτηριστικά (attributes):

- Registration=αριθμός κυκλοφορίας αυτοκινήτου
- Model name=κωδικός μοντέλου
- Description=σύντομη περιγραφή
- Maint _int=Km ανάμεσα σε διαδοχικές κανονικές (προγραμματισμένες) συντηρήσεις
- Car_Group_Name=κωδικός ομάδας που ορίζει τον τύπο του αυτοκινήτου και το κόστος ενοικίασης
- Rate_per_Km=Χρέωση ανά Km για τα αυτοκίνητα της ομάδας
- Rate_per_day=Χρέωση ανά ημέρα για τα αυτοκίνητα της ομάδας
- Date_bought=Ημερομηνία αγοράς του αυτοκινήτου
- Cost=Ποσό αγοράς του αυτοκινήτου
- Km_to_date=ένδειξη κοντέρ κατά την επιστροφή του αυτοκινήτου
- Km_last_service=κοντέρ αυτοκινήτου μετά το τελευταίο service
- Status=κατάσταση αυτοκινήτου
  (A=διαθέσιμο, H=νοικιασμένο, S=σε_συντήρηση, X=ανάγκη για συντήρηση)
- Cust_No=Κωδικός πελάτη
- Cust_Surname=Επώνυμο

- Cust_Name=Όνομα
- Cust_Address=Διεύθυνση
- Cust_AFM=Αριθμός Φορολογικού Μητρώου
- Cust_Town=Πόλη
- Cust_Country=Χώρα
- Cust_Post_Code=Ταχυδρομικός Κώδικας
- Cust_Contact_Person=Πρόσωπο για επαφή σε περίπτωση συναλλαγής με εταιρεία
- Cust_Pay_Method=Τρόπος πληρωμής, π.χ. C=Cash,V=VisaCard, M=Mastercard
- Booking_No=Κωδικός ενοικίασης
- Date_Reserved=Ημερομηνία Κράτησης
- Reserved_by=Όνομα υπαλλήλου που έκανε την κράτηση
- Date_Rent_Start=Ημερομηνία έναρξης ενοικιάσεως
- Rental_Period=Ημέρες ενοικίασης
- Amount_Due=Κόστος ενοικίασης υπολογισμένο κατά την επιστροφή
- Km_in=Κοντέρ κατά την επιστροφή από ενοικίαση
- Km_out=Κοντέρ κατά την έναρξη ενοικίασης
- Paid=YES/NO

Αν επιθυμείτε, μπορείτε να προσθέσατε και άλλα χαρακτηριστικά.

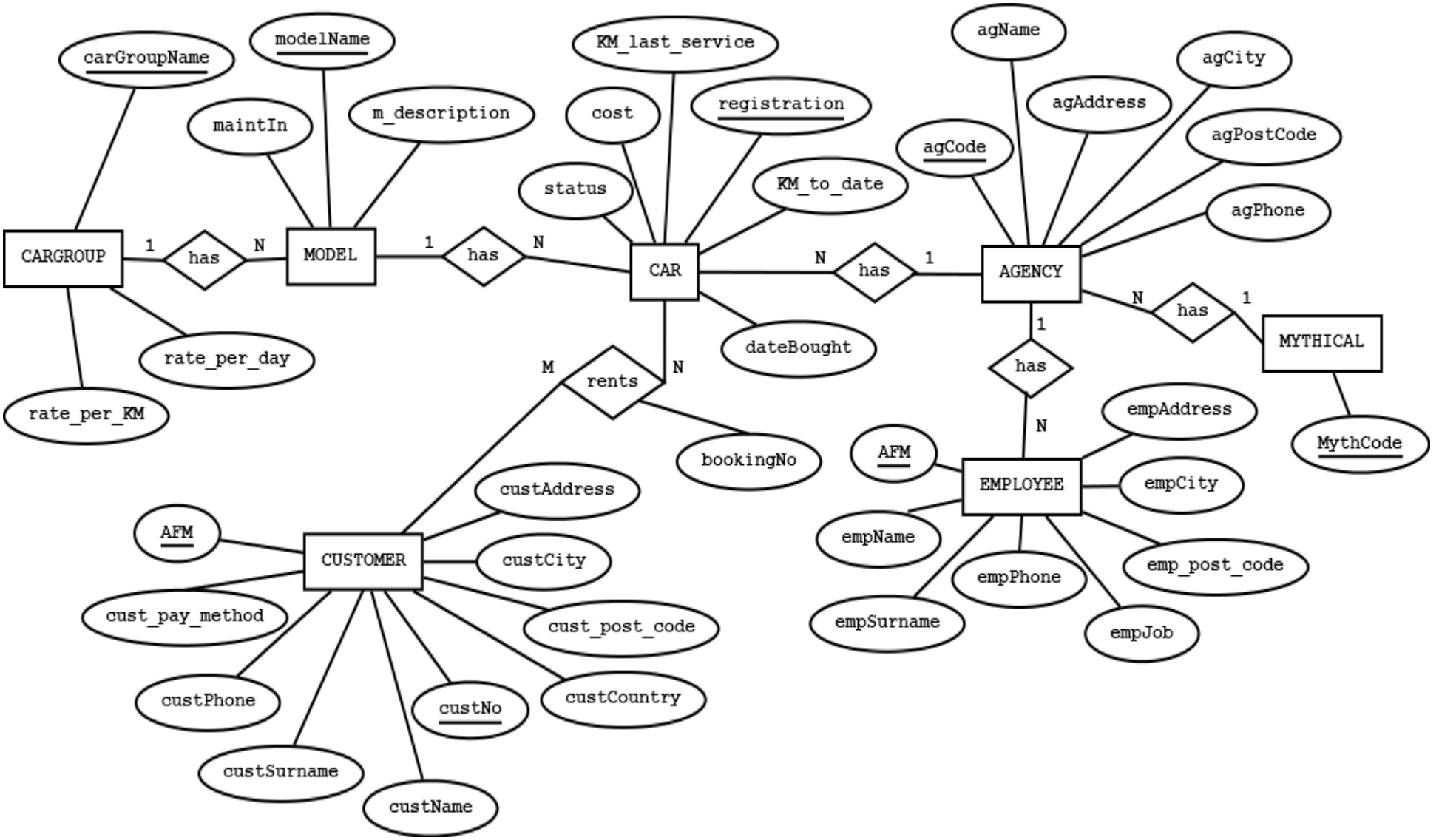
Μια βάση δεδομένων, όπως η ζητούμενη, πρέπει να αντιμετωπιστεί σαν μια βάση πολλαπλών μέσων (multimedia data base). Άρα, θα προτείναμε να προσθέσετε σχετικά στοιχεία, π.χ. εικόνες, βίντεο

Θα πρέπει να γίνουν τα εξής:

- Αναλυτική περιγραφή περιορισμών (constraints)
- Καταγραφή αντιπροσωπευτικών δειγμάτων στοιχείων (sample of data).

Σχεδιασμός μοντέλου Οντοτήτων Συσχετίσεων

Στην Εικόνα 6.18 δίδεται το μοντέλο Οντοτήτων συσχετίσεων του συστήματος βάσεως δεδομένων.



Εικόνα 6.18 ΜΟΣ εταιρίας ενοικιάσεως αυτοκινήτων

Ακολουθεί η τρίτη κανονική μορφή.

1) MYTHICAL(MythCode),

   Κύριο κλειδί=MythCode

2) AGENCY(agCode, agName, agAddress, agCity, agPostCode, agPhone, MythCode),

   Κύριο κλειδί=agCode, ξένο κλειδί=MythCode

3) CARGROUP(carGroupName, rate_per_km, rate_per_day),

   Κύριο κλειδί=carGroupName

4) MODEL(modelName, maintIn,, m_description, carGroupName),

   Κύριο κλειδί=modelName, ξένο κλειδί=carGroupName

5) CUSTOMER(AFM, custNo, custSurname, custName, custAddress, custCity, custCountry, cust_post_code, cust_pay_method, custPhone),

   Κύριο κλειδί=AFM

6) EMPLOYEE(AFM, empSurname, empName, empAddress, empCity, emp_post_code, empJob, empPhone, agCode),

   Κύριο κλειδί=AFM, ξένο κλειδί=agCode

7) CAR(registration, dateBought, status, cost, KM_last_service, KM_to_date, modelName, agCode),

   Κύριο κλειδί=registration, ξένο κλειδί=modelName, ξένο κλειδί=agCode

8) RENTS(custAFM, registration, bookingNo),

   Κύριο κλειδί=bookingNo, ξένο κλειδί=custAFM, ξένο κλειδί=registration)

6.2 Σχεσιακές Βάσεις Δεδομένων (Relational Database) και Κανονικοποίηση (normalization)

6.2.1 Εισαγωγή

Η σχεδίαση ενός **συστήματος βάσεως δεδομένων (ΣΒΔ)** συνδέεται άμεσα με την απάντηση στο εξής ερώτημα:

«Ποιες **δομές δεδομένων** (**data structures)** και ποιες λειτουργίες συνδεόμενες με αυτές θα μπορεί να υποστηρίζει το Σύστημα Βάσης Δεδομένων ;».

Στη βιβλιογραφία χρησιμοποιείται και ο όρος Μοντέλα Δεδομένων (Data Models) αντί του όρου data structure. Η απάντηση στο ερώτημα δόθηκε σε διάφορες κατευθύνσεις ανάλογα με τις υιοθετούμενες/χρησιμοποιούμενες δομές δεδομένων (data structures). Ανάλογα με την απάντηση στο ερώτημα αυτό επιλέγεται και το λογισμικό που υποστηρίζει τις υιοθετούμενες δομές (ή μοντέλα δεδομένων) και τις λειτουργίες σε αυτές. Έτσι αν υποτεθεί ότι επιλέγουμε να οργανώσουμε τα στοιχεία μας σε σχεσιακό ΣΒΔ τότε επιλέγουμε ένα σχεσιακό σύστημα διαχείρισης βάσης δεδομένων και εργαζόμαστε με αυτό.

Δομές Δεδομένων και λειτουργίες

Σχεσιακό ΣΒΔ. Στη σχεσιακή προσέγγιση (Relational approach) για την οργάνωση και την περιγραφή του συστήματος και των λειτουργιών του γίνεται χρήση κανονικοποιημένων δομών (normalized structures) που ονομάζονται σχέσεις (relations) ή πίνακες (tables).

Ιεραρχικό ΣΒΔ. Στην ιεραρχική προσέγγιση (hierarchical approach) για την οργάνωση και την περιγραφή του συστήματος και των λειτουργιών του γίνεται χρήση δομών δέντρου (tree structures).

Δικτυωτό ΣΒΔ. Στη δικτυωτή προσέγγιση (network approach) για την οργάνωση και την περιγραφή του συστήματος και των λειτουργιών του γίνεται χρήση δικτυωτών δομών (network or plex structures).

Συστήματα Ανάκτησης πληροφοριών (Information Retrieval system). Στα συστήματα αυτά, συνήθως βάσεις κειμένου και βάσεις εγγράφων (documents), για την οργάνωση και την περιγραφή του συστήματος και των λειτουργιών του γίνεται χρήση δομών βασιζόμενων σε **αντεστραμμένα αρχεία** (inverted files) ή αρχεία υπογραφών (signature files)

Συστήματα Υπερκειμένου (Hypertext based systems).

Στα συστήματα αυτά (π.χ. βάσεις σελίδων στο διαδίκτυο που περιλαμβάνουν δεσμούς - links σε άλλες σελίδες κτλ.) για την οργάνωση και την περιγραφή του συστήματος και των λειτουργιών του γίνεται χρήση δομών που διαχειρίζονται υπερκείμενο (Hypertext)

Αντικειμενοστρεφή ΣΒΔ.

Στην αντικειμενοστρεφή προσέγγιση (object-oriented approach) για την οργάνωση και την περιγραφή του συστήματος και των λειτουργιών του γίνεται χρήση ειδικών δομών και αντικειμενοστρεφών τεχνικών. Οι έννοιες του αντικειμένου (object), της κλάσης (class), της ιεραρχίας (hierarchy) κτλ. του αντικειμενοστρεφούς μοντέλου επιτρέπουν την αναπαράσταση "μη παραδοσιακών" δεδομένων που δεν μπορούμε να διαχειριστούμε στις σχεσιακές βάσεις (που βασίζονται σε "επίπεδες" (flat) δομές). Τα αντικειμενοστρεφή ΣΔΒΔ έχουν ειδικές γλώσσες για τη διαχείριση των στοιχείων της βάσης: Object Definition Language, Object Query Language.

Αντικειμενοστρεφή ΣΔΒΔ (object-relational DBMS)

Υβριδικό μοντέλο που συνδυάζει σχεσιακό και αντικειμενοστρεφές μοντέλο. Βλέπε και Stonebraker M., D.Moore (1996) Object - Relational DBMSs. The next great wave, Morgan Kaufmann

Συστήματα Βάσεων Δεδομένων πολλαπλών μέσων (multimedia).

Στην προσέγγιση πολλαπλών μέσων η αποθήκευση στοιχείων σε βάση δεδομένων περιλαμβάνει διάφορες μορφές πληροφορίας όπως κείμενο, έγγραφα, video, εικόνα, αρχεία ήχου κτλ. και όλα αυτά σε διάφορους μορφότυπους (format), π.χ., για τις εικόνες μπορούμε να έχουμε διάφορους μορφότυπους όπως TIFF, GIFF, BMP, ενώ για τα κείμενα μπορούμε να έχουμε απλά κείμενα (plain text), .doc files, RTF, Postscript, PDF, για τα έγγραφα μπορούμε να χρησιμοποιούμαι HTML, SGML, XML κτλ. Επομένως, για την οργάνωση και την περιγραφή του συστήματος και των λειτουργιών του γίνεται χρήση δομών πολλών διαστάσεων (multidimensional data structures) (π.χ. k-d trees, point quadtrees, MX-quadtrees, R-trees) που εξαρτώνται από το είδος της αποθηκευμένης πληροφορίας αλλά και από το είδος της εφαρμογής και από το χρησιμοποιούμενο λογισμικό για τη δημιουργία και τη διαχείριση της βάσης. πχ. αν έχουμε μία βάση χαρτών (που είναι μια ειδική περίπτωση βάσης πολλαπλών μέσων) ανάλογα με το χρησιμοποιούμενο λογισμικό, π.χ., GIS χρησιμοποιούμαι και άλλες δομές. Έτσι η Oracle προσφέρει ένα HHCODE (Hyperspatial) τύπο

δεδομένων (data type) για την αποθήκευση γεωγραφικών δεδομένων σε βάση που βασίζεται σε τεχνολογία (technology) quadtree. Για περισσότερα στοιχεία υπάρχουν πολλές αναφορές. Βλέπε για παράδειγμα

Subrahmanian, V. S. (1998). Principles of multimedia database systems. Morgan Kaufmann Publishers Inc.

Οι λειτουργίες που ενδιαφέρουν για κάθε δομή είναι οι παρακάτω:

- εισαγωγή στοιχείων - insert
- διαγραφή στοιχείων - delete
- ενημέρωση στοιχείων - update (ή modify)
- ανάκτηση στοιχείων - retrieve

6.2.2 Βασικές έννοιες της σχεσιακής (Relational) προσέγγισης, σχέση, πλειάδα, χαρακτηριστικά.

Η σχεσιακή θεωρία και η ορολογία της οφείλεται στον Codd. Πρώτη αναφορά στη θεωρία του γίνεται στα 1970 στο άρθρο:

Codd, E. F., (1970) A relational model for large shared data banks, Comm. ACM, 13(6), 377-387.

Ο Codd διατυπώνει τις απόψεις του για το σχεσιακό και το αντικειμενοστρεφές μοντέλο και για το μέλλον του σχεσιακού μοντέλου στο παραπάνω άρθρο και σε πολλά άλλα πασίγνωστα άρθρα του αλλά και σε επίπεδο συγγράμματος:

Codd, E. F. (1990). The relational model for database management: version 2. Addison-Wesley Longman Publishing Co., Inc.

Ορισμός σχέσης

Η σχέση (relation) ορίζεται με τη μαθηματική έννοια δηλαδή είναι ένα υποσύνολο καρτεσιανού γινομένου (συμβολισμός "x") πεδίων ορισμού (domains).

- Μπορούμε να θεωρήσουμε ότι μια σχέση (relation) είναι ένα σύνολο από όμοιες πλειάδες (tuples).
- Κάθε πλειάδα αποτελείται από Ν συνιστώσες (components) διαφορετικού (πιθανόν) τύπου (Ν-άδα).
- Το Ν λέγεται πολλαπλότητα (arity) της σχέσης.

Μπορείτε να θεωρήστε ότι μια σχέση είναι ένας πίνακας από πλειάδες και κάθε στήλη του πίνακα αντιστοιχεί σε μια συνιστώσα. Συνήθως δίνουμε ονόματα στις στήλες που λέγονται χαρακτηριστικά (attributes).

Παράδειγμα

Έστω μια κατασκευαστική εταιρεία που ενδιαφέρεται για την ενημέρωση στοιχείων γύρω από τον εφοδιασμό (αποστολές) ανταλλακτικών. Οι οντότητες, οι σχέσεις οντοτήτων και τα χαρακτηριστικά (ή ιδιότητες, attributes) τους είναι οι εξής:

(οντότητες, entity) **SUPPLIERS** (SNO,SNAME,STATUS,SCITY)

PARTS (PNO,PNAME,COLOR,WEIGHT,PCITY)

(σχέση ή συσχέτιση, relationship) **SHIPMENTS** (SNO,PNO,QTY)

Για τις ανάγκες του παραδείγματος υποτίθεται ότι ανά χρονική στιγμή και ανά συνδυασμό προμηθευτή-ανταλλακτικού έχουμε μόνο ένα εφοδιασμό (μια φόρτωση).

Οι σχέσεις για το παράδειγμα μας είναι οι εξής:

SUPPLIERS $\subseteq$ DOM(SNO) x DOM(SNAME) x DOM(STATUS) x DOM(SCITY)

PARTS $\subseteq$ DOM(PNO) x DOM(PNAME) x DOM(COLOR) x DOM(WEIGHT) x DOM(PCITY)

SHIPMENTS $\subseteq$ DOM(SNO) x DOM(PNO) x DOM(QTY)

Δηλαδή, η σχέση αποτελείται από όμοιες πλειάδες (tuples) τιμών ενώ κάθε τιμή ανήκει σε μία ιδιότητα (στο πεδίο ορισμού της ιδιότητας ακριβέστερα). Με λίγο διαφορετικό μαθηματικό συμβολισμό:

SUPPLIERS $\subseteq \{ (x_1 , x_2 , x_3 , x_4) \mid x_1 \in DOM(SNO),$

$x_2 \in DOM(SNAME),$

$x_3 \in DOM(STATUS),$

$x_4 \in DOM(SCITY) \}$

- Τα σχεσιακά Προϊόντα Διαχείρισης Βάσης Δεδομένων (ΠΔΒΔ) συνήθως υποστηρίζουν/αντιμετωπίζουν για λόγους κατανόησης μια σχέση σαν πίνακα.
- Οι γραμμές των πινάκων αντιστοιχούν σε πλειάδες (tuples), ενώ οι στήλες σε ιδιότητες (attributes).
- Κάθε ιδιότητα (attribute) παίρνει τιμές από ένα συγκεκριμένο σύνολο τιμών.

6.2.2.1 Πως μεταγράφεται το μοντέλο οντοτήτων συσχετίσεων στο σχεσιακό μοντέλο. Σύνοψη

Στη σχεσιακή προσέγγιση οι οντότητες και πολλοί τύποι συσχετίσεων οντοτήτων όπως οι συσχετίσεις τύπου M:N (που ενδεχομένως έχουν και κάποια χαρακτηριστικά (attributes)), αλλά και συσχετίσεις οριζόμενες σε πάνω από δύο οντότητες, π.χ., τριαδικές τύπου M:N:N μεταγράφονται με τον ίδιο τρόπο, δηλαδή παριστάνονται σαν σχέσεις (ή πίνακες).

Οι συσχετίσεις τύπου 1:N με ή χωρίς χαρακτηριστικά (attribute) αναπαρίστανται σαν πεδία της σχέσης (ή πίνακα).

Άρα, τα δεδομένα που καταγράφουμε για οντότητες και συσχετίσεις οντοτήτων είναι οργανωμένα σε σχέσεις (ή πίνακες) στο σχεσιακό μοντέλο.

Παρ' όλο που κάτι τέτοιο είναι κάπως περιοριστικό, μπορείτε να δειτε μια αντιστοιχία ανάμεσα στη σχέση και στο αρχείο. Κάθε πλειάδα αντιστοιχεί σε μια εγγραφή. Κάθε συνιστώσα της πλειάδας αντιστοιχεί σε κάποιο πεδίο.

Ορισμός Σχήματος Σχέσης

Σχήμα Σχέσης (relation scheme) ονομάζεται μια παράθεση των χαρακτηριστικών (ή ιδιοτήτων) της.

Συνοψίζοντας:

Κάθε χαρακτηριστικό (ή ιδιότητα, attribute) παίρνει τιμές από ένα συγκεκριμένο σύνολο τιμών. Τα σύνολα αυτά για ένα σχήμα σχέσης (ή πίνακα) είναι τα πεδία ορισμού (domain) της. Ο αριθμός των ιδιοτήτων δίνει την πολλαπλότητα (arity) του σχήματος σχέσης.

Παραδείγματα

Παραθέτουμε μερικά σχήματα σχέσεων:

FOITHTES(EPWNYMO, ONOMA, ARITMHT, EXAMHNO)

MATHIMATA(LEKTIKO, KWD_MAT)

EGGARFES(ARITMHT, KWD_MAT)

YPALLHLOI(ONOM_EPWN, GLWSSA, XROXRH)

Στην Εικόνα 6.19 παραθέτουμε τις σχέσεις που αντιστοιχούν στα τρία πρώτα παραδείγματα. Το τέταρτο παράδειγμα αφήνεται ως άσκηση. Στη συνέχεια θα χρησιμοποιήσουμε τον όρο σχέση και για το σχήμα σχέσης.

Σχήμα σχέσης FOITHTES και πλειάδες

EPWNYMO	ONOMA	ARITMHT	EXAMHNO
Κυριακόπουλος	Νικηφόρος	213	Δ
Αποστόλου	Ζωή	816	Α
Παπαπέτρου	Νικόλαος	450	Β
Ζευγαρίδης	Ορέστης	346	Γ
Κοταμανίδου	Ειρήνη	610	Α

Σχήμα σχέσης MATHIMATA και πλειάδες

MATHIMA	KWDIKOS_MATHIMATOS
Αρχές Οικονομικής Ι	Α1
Προγραμματισμός Η/Υ Ι	Α5
Ανθρώπινες Σχέσεις στην εργασία	Α8
Προγραμματισμός Η/Υ ΙΙ	Β5
Χρήμα - Πίστη – Τράπεζες	Γ1
Εισαγωγή στο Αστικό Δίκαιο	Α4
Στατιστική Επιχειρήσεων	Β2
Οικονομική της Διοίκησης	Γ3
Ιστορία και Αρχές Συνεργατισμού	Γ7
Συστήματα Πληροφοριών Διοίκησης	Γ6
Γενική Λογιστική Ι	Α3

Σχήμα σχέσης EGGRAFES και πλειάδες

ARITMHT	KWDIKOS_MATHIMATOS
213	Γ1
213	Β5
450	Β5
816	Α5
816	Α8
450	Β2
346	Γ1

346	Γ3
610	A1
610	A3

Εικόνα 6.19 Πλειάδες των σχημάτων σχέσεων

Παρατηρήσεις-Κανόνες για τη σχεσιακή προσέγγιση

Υπάρχει τουλάχιστον μία ιδιότητα ή ένας συνδυασμός ιδιοτήτων με τιμές που είναι μοναδικές για κάθε σχέση. Αυτές οι ιδιότητες ή αυτοί οι συνδυασμοί μπορούν να θεωρηθούν κλειδιά δηλαδή μπορούν να χρησιμοποιηθούν για το καθορισμό (identification) των πλειάδων της σχέσης. Αν έχουμε πολλούς τέτοιους συνδυασμούς τους ονομάζουμε υποψήφια κύρια κλειδιά (candidate keys) και ένας από τους συνδυασμούς επιλέγεται σαν Κύριο κλειδί (primary key). Για παράδειγμα, στη σχέση SUPPLIERS υποψήφια κλειδιά μπορούν να είναι τα SNO, SNAME (αν υποτεθεί ότι δεν υπάρχουν συνωνυμίες). Από αυτά επιλέγεται σαν κύριο (primary) το SNO.

Ξένα Κλειδιά

Ένα συνηθισμένο χαρακτηριστικό των σχέσεων είναι ότι μια σχέση μπορεί να περιέχει ένα ή περισσότερα ξένα (foreign) κλειδιά. Ένα ξένο κλειδί (foreign key) σε μια σχέση συνήθως αναπαρίσταται με μια ιδιότητα (attribute) που είναι κύριο κλειδί σε κάποια άλλη σχέση.

Για παράδειγμα η σχέση,

COURSE (Course-Identification, Course-Title, Course-Description, Teacher-Identification)

περιλαμβάνει την ιδιότητα Teacher-Identification που είναι κύριο κλειδί στη σχέση:

TEACHER(Teacher-Identification, Teacher-LastName, Teacher-FirstName, Phone).

Η ιδιότητα Teacher-Identification είναι ξένο κλειδί στη σχέση COURSE.

Κανόνας πρώτος ή κανόνας ακεραιότητας οντότητας
(Integrity Rule 1 or constraint Rule or Entity Integrity).

Όλες οι τιμές ενός κύριου κλειδιού είναι διακεκριμένες και όλες οι συνιστώσες ενός σύνθετου κύριου κλειδιού πρέπει να έχουν τιμή σε κάθε πλειάδα (γραμμή) της σχέσης (του πίνακα). Δηλαδή ένα (σύνθετο) κύριο κλειδί δεν μπορεί να έχει τιμή NULL (σε οποιαδήποτε από τις συνιστώσες του) για κάποια γραμμή του πίνακα. Ο κανόνας εξασφαλίζει ότι ανά δύο οι οντότητες είναι διακεκριμένες.

Κανόνας δεύτερος ή κανόνας αναφερόμενης ακεραιότητας
(Iintegrity Rule 2 or Referential Integrity).

Έστω D ένα πρώτο πεδίο ορισμού (primary domain), δηλαδή ένα πεδίο ορισμού επάνω στο οποίο ορίζεται ένα απλό κύριο κλειδί (single attribute primary key). Έστω σχέση (relation) R_1 με ιδιότητα A που ορίζεται στο D και έστω σχέση R_2 με κύριο κλειδί οριζόμενο στο D. Τότε η τιμή της ιδιότητας A στο R_1 πρέπει να είναι είτε "τίποτα" (null) ή ίση με τιμή του κύριου κλειδιού κάποιας πλειάδας της σχέσης R_2.

Παράδειγμα

Έστω σύστημα διαχείρισης παραγγελιών. Οι σχέσεις (πίνακες) του συστήματος είναι οι εξής:

ORDER(ORDERNO,CUSTOMER,TOTAL,DATE), με κλειδί ORDERNO.

ORDERLINE(ORDERNO,LINENO,PART,QYANTITY),

με κλειδί (ORDERNO, LINENO).

Για τις σχέσεις αυτές πρέπει να ισχύει ο δεύτερος κανόνας ακεραιότητας.

Για ένα παραπέρα ξεκαθάρισμα των εννοιών παραθέτουμε την πλήρη περιγραφή της βάσης δεδομένων ΠΡΟΜΗΘΕΥΤΕΣ-ΑΝΤΑΛΛΑΚΤΙΚΑ με τη μορφή σχεσιακού σχήματος (relational schema) γραμμένου σε γλώσσα SQL του προϊόντος της Oracle (Εικόνα 6.20). Το σχήμα αυτό επιτρέπει παραβίαση των δύο κανόνων ακεραιότητας. Για να διασφαλίσουμε την ισχύ των κανόνων θα πρέπει να ορίσουμε κατάλληλα κύρια και ξένα κλειδιά.

```
CREATE TABLE SUPPLIERS (SNO CHAR(3) NOT NULL,
  SNAME CHAR(20),
  STATUS NUMBER(2),
  SCITY CHAR(20));
CREATE TABLE PARTS (PNO NUMBER(4) NOT NULL,
  PNAME CHAR(25),
  PWEIGHT NUMBER (7,2),
  PCOLOR CHAR(15),
  PCITY CHAR(20));
 CREATE TABLE SHIPMENT (SNO CHAR(3) NOT NULL,
 PNO CHAR(4) NOT NULL,
 QTY NUMBER(7,2));
```

Εικόνα 6.20 Ορισμός του σχήματος της βάσης δεδομένων ΠΡΟΜΗΘΕΥΤΕΣ-ΑΝΤΑΛΛΑΚΤΙΚΑ με γλώσσα SQL.

Στην Εικόνα 6.21 παραθέτουμε ένα δείγμα δεδομένων (sample of data) για τη σχεσιακή βάση ΠΡΟΜΗΘΕΥΤΕΣ-ΑΝΤΑΛΛΑΚΤΙΚΑ.

SUPPLIERS

SNO	SNAME	SCITY	STATUS
SO1	ΣΑΚΗΣ	ΛΟΝΔΙΝΟ	20
SO2	ΤΑΚΗΣ	ΠΑΡΙΣΙ	10
S03	ΒΑΣΟΣ	ΠΑΡΙΣΙ	30

PARTS

PNO	PNAME	PCOLOR	PCITY
P010	ΒΙΔΑ Α	ΚΟΚΚΙΝΗ	ΛΟΝΔΙΝΟ
P020	ΒΙΔΑ Β	ΚΟΚΚΙΝΗ	ΠΑΡΙΣΙ
P030	ΒΙΔΑ C	ΚΟΚΚΙΝΗ	ΡΩΜΗ
P040	ΒΙΔΑ C	ΚΙΤΡΙΝΗ	ΛΟΝΔΙΝΟ

SHIPMENT

SNO	PNO	QTY
S01	P010	300
S01	P020	200
S01	P030	400
S02	P010	300

S02	P020	400
S03	P020	200

Εικόνα 6.21 Δείγμα της βάσης δεδομένων ΠΡΟΜΗΘΕΥΤΕΣ-ΑΝΤΑΛΛΑΚΤΙΚΑ

Κρίνουμε σκόπιμο να επαναλάβουμε ότι στη σχεσιακή οπτική γωνία (relational view) οι οντότητες (entities) και οι συσχετίσεις (relationships) έχουν ταυτόσημη αναπαράσταση.

6.2.3 Πράξεις στις σχέσεις (σχεσιακή άλγεβρα)

Στις σχέσεις μπορούμε να κάνουμε όλες τις πράξεις που μπορούμε να κάνουμε σε σύνολα, π.χ., ένωση, τομή, διαφορά, καρτεσιανό γινόμενο. Εδώ θα σταθούμε σε αυτές τις πράξεις που μας ενδιαφέρουν περισσότερο:

1. Επιλογή (στη βιβλιογραφία χρησιμοποιούνται οι όροι Selection ή restriction):

$$R_1 = \sigma_E (R)$$

Η πράξη της «επιλογής» συμβολίζεται με σ και εφαρμόζεται σε μία σχέση R. Το αποτέλεσμά της είναι μια σχέση R_1 που έχει την ίδια πολλαπλότητα με την R και περιλαμβάνει τις πλειάδες της R για τις οποίες ισχύει η συνθήκη E.

Παράδειγμα

Στην Εικόνα 6.19 είδαμε το σχήμα σχέσης FOITHTHS και τις πλειάδες του.

Η σχέση FOITHTES1 = $\sigma_{\text{EXAMHNO=A}}$ (FOITHTES) περιέχει μόνο εκείνες τις πλειάδες της σχέσης R όπου το EXAMHNO έχει την τιμή A (πίνακας 6.2).

Πίνακας 6.2 Σχήμα σχέσης FOITHTES1 και πλειάδες

EPWNYMO	ONOMA	ARITMHT	EXAMHNO
Αποστόλου	Ζωή	816	A
Κοταμανίδου	Ειρήνη	610	A

Σύνθετη συνθήκη της πράξης της επιλογής

Δείτε τι ισχύει στην περίπτωση σύνθετης συνθήκης

$$\sigma_{\text{COMPANY = ABC \& CITY = ATHENS}} (\text{EMPLOYEE}) =$$

$$\sigma_{\text{COMPANY = ABC}} (\sigma_{\text{CITY = ATHENS}} (\text{EMPLOYEE}))$$

Στο δεύτερο μέλος της ισότητας επιλέγουμε αρχικά τις πλειάδες που ικανοποιούν τη συνθήκη CITY = ATHENS και στη συνέχεια εφαρμόζουμε την πράξη της επιλογής στην προκύπτουσα συνθήκη.

Στη συνθήκη μπορούν να χρησιμοποιηθούν οι τελεστές = , < ,> , <= , >= .

Σημαντική παρατήρηση

Όταν γράφουμε ένα ερώτημα (αναζήτηση, query) στην άλγεβρα καθορίζουμε τη σειρά εκτέλεσης των επιμέρους περιλαμβανόμενων λειτουργιών (πράξεων)

2. Προβολή (projection): $R_1 = \pi_{a1,a2,a3,...,ak}$ (R)

Έστω ότι R είναι μια σχέση πολλαπλότητας N με χαρακτηριστικά a_1 , a_2 , ... a_N και ότι εφαρμόζουμε σε αυτήν την πράξη της προβολής. Τότε το αποτέλεσμα της πράξης είναι μια σχέση R_1 η οποία είναι μια σχέση πολλαπλότητας $k < N$ με χαρακτηριστικά a_1 , a_2 , ... a_k , όπου τα a_m είναι χαρακτηριστικά της R και το m μπορεί να έχει τιμές από 1 μέχρι k. Δηλαδή, η προβολή εφαρμόζεται σε μία σχέση και δείχνει το χαρακτηριστικό ή τα χαρακτηριστικά της σχέσης (με άλλη ορολογία τη στήλη ή τις στήλες του πίνακα) που θέλουμε.

Παράδειγμα

Στον πίνακα 6.3 βλέπουμε το αποτέλεσμα, δηλαδή τη σχέση FOITHTES2, της εφαρμογής της προβολής στη σχέση FOITHTES της εικόνας 6.19. Ισχύει ότι:

$$\text{FOITHTES2} = \pi_{\text{ARITMHT, EXAMHNO , EPWNYMO}} (\text{FOITHTES})$$

Πίνακας 6.3 Σχήμα σχέσης FOITHTES2 και πλειάδες

ARITMHT	EXAMHNO	EPWNYMO
213	Δ	Κυριακόπουλος
816	Α	Αποστόλου
450	Β	Παπαπέτρου
346	Γ	Ζευγαρίδης
610	Α	Κοταμανίδου

Παρατήρηση

Το αποτέλεσμα μιας προβολής μπορεί να έχει λιγότερες πλειάδες από την αρχική σχέση.

Παράδειγμα

Στον πίνακα 6.4 βλέπουμε το αποτέλεσμα, δηλαδή τη σχέση EGGRAFES1, της εφαρμογής της προβολής στη σχέση EGGRAFES της εικόνας 6.19. Ισχύει ότι:

$$\text{EGGRAFES1} = \pi_{\text{ARITMHT}} (\text{EGGRAFES})$$

Πίνακας 6.4 Σχήμα σχέσης EGGRAFES1 και πλειάδες

ARITMHT
213
450
816
346
610

Θυμηθείτε ότι η σχέση είναι σύνολο. Και σε ένα σύνολο, ένα στοιχείο δεν εμφανίζεται περισσότερες από μία φορές.

3. Καρτεσιανό γινόμενο (Cartesian Product or Times)

Έννοια γνωστή από τη θεωρία συνόλων.

Αν η σχέση R_1 έχει σχήμα A_1 , A_2 .... A_v και η σχέση R_2 έχει σχήμα B_1 ,B_2 .... B_μ

τότε το καρτεσιανό γινόμενο των δύο σχέσεων, συμβολικά R_1 x R_2, ορίζεται σαν μία σχέση με σχήμα (A_1 , A_2 .... A_v , B_1 ,B_2 .... B_μ), δηλαδή,

$$R_1 \text{ x } R_2 = \{ (\alpha_1 , ... , \alpha_v , \beta_1 , ... \beta_\mu) \mid (\alpha_1 , ...) \in R_1 ,$$

$$(\beta_1 , ...) \in R_2 \}$$

Αν card(R) είναι ο αριθμός των πλειάδων (tuples) μιας σχέσης R τότε

$$\text{card} (R_1 \text{ x } R_2) <= \text{card } (R_1) \text{ x card } (R_2)$$

Όπως είναι προφανές η έννοια γενικεύεται άμεσα για περισσότερες σχέσεις.

4. Σύνδεση (join) σχέσεων: R = R1 ⋈ R2

Αν R_1 , R_2 είναι σχέσεις με πολλαπλότητες N1 και N2 και με M κοινά χαρακτηριστικά των δύο σχέσεων κατασκευάζουμε την R που είναι μια σχέση με πολλαπλότητα

$$N = N1 + N2 - M.$$

Η R δημιουργείται με την παράθεση (concatenation) όλων των πλειάδων $t_1 \in R_1$και $t_1 \in R_2$ που ικανοποιούν τη συνθήκη της σύνδεσης .

Σχηματικά, για όλες τις πλειάδες των δύο σχέσεων κάνουμε τα εξής:

- Συνδέουμε κάθε πλειάδα της R_1 με όλες τις πλειάδες της R2 και δημιουργούμε έτσι πλειάδες με πολλαπλότητα N1 + N2.
- Από αυτές κρατάμε μόνον αυτές που έχουν σε όλα τα κοινά χαρακτηριστικά τις ίδιες τιμές.
- Σε αυτές που κρατήσαμε, κρατάμε τα (κοινά) χαρακτηριστικά μια φορά μόνον.

Παράδειγμα

Στον πίνακα 6.5 βλέπουμε το αποτέλεσμα, δηλαδή τη σχέση EGGRFOIT, της εφαρμογής της σύνδεσης της σχέσης FOITHTES με τη σχέση EGGRAFES της εικόνας 6.19. Ισχύει ότι:

EGGRFOIT = FOITHTES ⋈ EGGRAFES

Πίνακας 6.5 Σχήμα σχέσης EGGRFOIT και πλειάδες

EPWNYMO	ONOMA	ARITMHT	EXAMHNO	KWDIKOS_MATHIMATOS
Κυριακόπουλος	Νικηφόρος	213	Δ	Γ1
Κυριακόπουλος	Νικηφόρος	213	Δ	Β5
Αποστόλου	Ζωή	816	Α	Α5
Αποστόλου	Ζωή	816	Α	Α8
Παπαπέτρου	Νικόλαος	450	Β	Β5
Παπαπέτρου	Νικόλαος	450	Β	Β2
Ζευγαρίδης	Ορέστης	346	Γ	Γ1
Ζευγαρίδης	Ορέστης	346	Γ	Γ3
Κοταμανίδου	Ειρήνη	610	Α	Α1
Κοταμανίδου	Ειρήνη	610	Α	Α3

Αν οι R_1 , R_2 δεν έχουν κοινά χαρακτηριστικά (attributes) ισχύει ότι η σύνδεση τους είναι ίση με σ_C $(R_1 \times R_2)$.

Ακολουθούν και άλλες πράξεις πολύ γνωστές από τη θεωρία συνόλων .

5. Ένωση (union), $R_1 \cup R_2$

Δύο σχέσεις R_1 , R_2 είναι συμβατές ως προς την ένωση αν έχουν το ίδιο σχήμα.

6. Διαφορά (difference), $R_1 - R_2$.

Και εδώ πρέπει να έχουμε συμβατότητα των δύο σχέσεων .

7. Τομή (intersection), $R_1 \cap R_2$.

Ισχύει $R_1 \cap R_2 = R_1 - (R_1 - R_2)$

Άρα και εδώ πρέπει να έχουμε συμβατότητα των δύο σχέσεων.

Οι παραπάνω πράξεις υλοποιούνται συνήθως στις γλώσσες διαχείρισης δεδομένων (Data Manipulation Languages, DML) που βασίζονται στη σχεσιακή άλγεβρα και χρησιμοποιούνται για τη διατύπωση ερωτημάτων προς το ΣΔΒΔ δηλαδή για την ανάκτηση πληροφοριών.

6.2.4 Σχεσιακό ΣΔΒΔ

Ένα ΣΔΒΔ θεωρείται ότι είναι σχεσιακό [βλέπε π.χ., Page] αν υποστηρίζει τουλάχιστον:

- Αποθήκευση της πληροφορίας (των δεδομένων) σε σχεσιακές δομές δεδομένων ή με άλλα λόγια την αποθήκευση της πληροφορίας σε πίνακες
- Ανάκτηση των πληροφοριών με χρησιμοποίηση των λειτουργιών SELECT, JOIN, PROJECT
- Αναπαράσταση των συσχετίσεων μεταξύ δεδομένων κατά τρόπο που εξασφαλίζει ότι οι συσχετίσεις αναπαρίστανται ρητά μέσα στα δεδομένα αυτά ("...explicitly represented in that data itself..."), δηλαδή είναι μέρος των δεδομένων της βάσης.

Κατά τον Date ένα προϊόν είναι σχεσιακό αν υποστηρίζει:

- τις σχεσιακές δομές δεδομένων
- σχεσιακή διαχείριση δεδομένων
- ακεραιότητα δεδομένων (1ος, 2ος κανόνες ακεραιότητας)
- τις οκτώ (8) πράξεις της σχεσιακής άλγεβρας που καθόρισε ο Codd: Select, Project, Join, Union, Intersect,
- Minus (difference), Times (product), Divide.

Η τελευταία πράξη δε συζητήθηκε έως τώρα αλλά θα δοθεί παράδειγμα στη συνέχεια.

6.2.4.1 Οι «12 κανόνες» (που είναι 13) ή πότε ένα προϊόν είναι σχεσιακό.

Ακολουθεί μία «παλιά» συζήτηση.

"The list is definitive and largely undisputed in the industry. The 12 rules can be used as the best relational design criteria ..." (Η λίστα είναι οριστική και σε μεγάλο βαθμό πέραν κάθε αμφισβήτησης για τον κλάδο. Οι 12 κανόνες μπορούν να χρησιμοποιηθούν ως τα καλύτερα σχεσιακά κριτήρια σχεδιασμού) [Page, p.94]

Στις μέρες μας οι κανόνες που ακολουθούν δεν αναφέρονται τόσο συχνά. Η συζήτηση γίνεται σε άλλα θέματα όπως η απόδοση (performance) του ΣΔΒΔ κτλ.

1) *"Any truly relational database must be manageable entirely through its own relational capabilities".* Μία πραγματικά σχεσιακή βάση είναι εξ' ολοκλήρου διαχειρίσιμη μέσω των σχεσιακών δυνατοτήτων (ενός πραγματικά σχεσιακού ΣΔΒΔ). Κανόνας που καλύπτει και τους άλλους αλλά ακούγεται και λίγο σαν ταυτολογία
2) *The Information Rule*-Αποθήκευση όλων των πληροφοριών ως τιμών σε πίνακες
3) *The rule of guaranteed access*-Κάθε στοιχείο (data item) είναι προσπελάσιμο με όνομα πίνακα, όνομα στήλης και τιμή κύριου κλειδιού
4) *The systematic treatment of null values*-Η υποστήριξη των τιμών αυτών πρέπει να γίνεται με συνέπεια στο ΣΔΒΔ, να είναι ανεξάρτητη από τον τύπο δεδομένων (πχ. η τιμή null σημαίνει το ίδιο σε πεδίο τύπου char και σε πεδίο τύπου integer) κτλ.
5) *The database description rule*-απλοποιώντας λίγο σημαίνει ότι υπάρχει κατάλογος/λεξικό δεδομένων με την περιγραφή της βάσης
6) T*he comprehensive sub-language rule* - απλοποιώντας σημαίνει ότι υπάρχει γλώσσα (ή γλώσσες) ορισμού δεδομένων, διαχείρισης, διατύπωσης περιορισμών ακεραιότητας κτλ.
7) *The view updating rule* - όλες οι όψεις που ορίζουμε (δηλαδή οι "ιδεατοί"/μη υλοποιήσιμοι στην πραγματικότητα πίνακες των δεδομένων που ορίζονται με επιλογή κάποιων στοιχείων από άλλους πίνακες ή και στήλες παραγόμενες από στοιχεία άλλων πινάκων) και που είναι θεωρητικά δυνατόν να ενημερώνονται πρέπει να ενημερώνονται από το ΣΔΒΔ (δύσκολο!).
8) *The insert and update rule*-απλοποιώντας σημαίνει υποστήριξη εντολών insert, update για σύνολα γραμμών κάθε μιας σχέσης (πίνακα) της βάσης.
9) *The physical independence rule*-η πρόσβαση του χρήστη στη βάση δεν επηρεάζεται από αλλαγές στην αναπαράσταση των αποθηκευμένων δεδομένων
10) *The logical data independence rule*-όταν αλλάζουν οι πίνακες της βάσης δεν αλλάζουν τα προγράμματα που χρησιμοποιούν οι χρήστες
11) *Integrity independence rule*-υποστήριξη των δύο κανόνων ακεραιότητας
12) *Distribution rule*-προγράμματα εφαρμογών που εκτελούνται σε μη κατανεμημένη βάση δεδομένων δεν επηρεάζονται αν τα δεδομένα της βάσης γίνουν κατανεμημένα στα πλαίσια ενός κατανεμημένου σχεσιακού ΣΔΒΔ.
13) *No subversion rule*-αφορά προϊόντα που δεν ήταν σχεσιακά και χρησιμοποιούσαν ένα "σχεσιακό" ανώτερο επίπεδο (front-end) για να δώσουν τη δυνατότητα χρήσης σύμφωνα με τα ισχύοντα στα σχεσιακά ΣΔΒΔ. Για παράδειγμα το πολύ γνωστό παλαιότερα Cullinet's IDMS/R παραβίαζε τον κανόνα αυτό.

Στη βιβλιογραφία [π.χ. Rolland, 1998] δίνονται διάφοροι ορισμοί για το πόσο σχεσιακό είναι ένα προϊόν:

- **Minimally relational system** - κατ' ελάχιστο σχεσιακό (πρέπει να υποστηρίζει την αποθήκευση των δεδομένων σε πίνακες χρησιμοποιώντας απλά πεδία ορισμού-domains, να υποστηρίζει εντολές SELECT, PROJECT, JOIN)
- **Relationally complete system**- σχεσιακά πλήρες (Minimally relational και να υποστηρίζει δηλώσεις UNION, MINUS)
- **Fully relational system** - πλήρως σχεσιακό (relationally complete) (υποστήριξη 1ου και 2ου κανόνων ακεραιότητας, υποστήριξη σε πεδία ορισμού οριζόμενα από το χρήστη)

6.2.5 Παραδείγματα πράξεων της σχεσιακής άλγεβρας και αντίστοιχων δηλώσεων σε γλώσσα SQL

Στην ενότητα αυτή θα δοθούν παραδείγματα εφαρμογής των πράξεων της σχεσιακής άλγεβρας σε πίνακες. Επιπλέον, θα παρατεθούν οι αντίστοιχες δηλώσεις σε γλώσσα SQL που οδηγούν στα ίδια αποτελέσματα. Πολλά από τα παραδείγματα βασίζονται σε παραπέρα επεξεργασία και εμπλουτισμό παραδειγμάτων του συγγράμματος:

Winfried Karl Grassmann and Jean-Paul Tremblay (1996) Logic and discrete mathematics. A computer science perspective, Prentice-Hall, chapter 12, section 12.3 Relational Algebra, pp. 605-620

Στην Εικόνα 6.22 παρουσιάζουμε τις σχέσεις P και Q και παραδείγματα εφαρμογής των πράξεων union (ένωση), intersection (τομή), difference (διαφορά), selection (επιλογή), projection (προβολή).

Σχέση P

Empno	Name	Dept	School
1015	Smith	Hist	Arts
1711	Carter	Math	Science
2560	Andrews	Chem	Science
3673	Angel	Acct	Commerce

Σχέση P

Empno	Name	Dept	School
651	Kaplan	Hist	Science
1711	Carter	Math	Science
2735	Jones	Ped	Education
3673	Angel	Acct	Commerce

(a) Σχέση P U Q

Empno	Name	Dept	School
651	Kaplan	Hist	Science
1015	Smith	Hist	Arts
1711	Carter	Math	Science
2560	Andrews	Chem	Science
2735	Jones	Ped	Education
3673	Angel	Acct	Commerce

(b)Σχέση P ∩ Q

Empno	Name	Dept	School
1711	Carter	Math	Science
3673	Angel	Acct	Commerce
(c) Σχέση	P - Q		
Empno	Name	Dept	School
1015	Smith	Hist	Arts
2560	Andrews	Chem	Science

(d) σ SCHOOL=SCIENCE (P)

Empno	Name	Dept	School
1711	Carter	Math	Science
2560	Andrews	Chem	Science

(e) π EMPNO, NAME , SCHOOL (Q)

Empno	Name	School
651	Kaplan	Science
1711	Carter	Science
2735	Jones	Education
3673	Angel	Commerce

Εικόνα 6.22 Οι σχέσεις P και Q και παραδείγματα εφαρμογής των πράξεων union, intersection, difference, selection, projection.σε αυτές.

Στη συνέχεια παραθέτουμε τις αντίστοιχες δηλώσεις σε γλώσσα SQL οι οποίες αντιστοιχούν στα παραδείγματα της εικόνας 6.22:

```
(a) SELECT empno, name, dept, school FROM P
  UNION
  SELECT empno, name, dept, school FROM Q;

(b) SELECT empno, name, dept, school FROM P
 INTERSECT
 SELECT empno, name, dept, school FROM Q;

(c) SELECT empno, name, dept, school FROM P
 MINUS
 SELECT empno, name, dept, school FROM Q;

(d) SELECT empno, name, dept, school
  FROM P
  WHERE SCHOOL='Science';

(e) SELECT empno, name, school
  FROM P;
```

Στην Εικόνα 6.23 παρουσιάζουμε τις σχέσεις R και S και παραδείγματα εφαρμογής των πράξεων product (καρτεσιανό γινόμενο), join (σύνδεση), natural join (φυσική σύνδεση) σε αυτές.

Σχέση R

Empno	Name
1015	Smith
1711	Carter
2560	Andrews

Σχέση S

Empno	Comm
1015	1050
1711	775
2560	1125

R x S(b)

Empno	Name	Empno	Comm
1015	Smith	1015	1050
1015	Smith	1711	775
1015	Smith	2560	1125
1711	Carter	1015	1050
1711	Carter	1711	775
1711	Carter	2560	1125
2560	Andrews	1015	1050
2560	Andrews	1711	775
2560	Andrews	2560	1125

R ⋈ S [R.empno=S.empno]

Empno	Name	Empno	Comm
1015	Smith	1015	1050
1711	Carter	1711	775
2560	Andrews	2560	1125

Natural Join

Empno	Name	Comm	
1015	Smith	1050	
1711	Carter	775	
2560	Andrews	1125	

Εικόνα 6.23 Οι σχέσεις R και S και παραδείγματα εφαρμογής των πράξεων product, join, natural join

Στη συνέχεια παραθέτουμε τις αντίστοιχες δηλώσεις σε γλώσσα SQL οι οποίες αντιστοιχούν στα παραδείγματα της εικόνας 6.22:

```
(a) SELECT R.empno, name, S.empno, comm FROM R,S;

(b) SELECT R.empno, name, S.empno, comm FROM R
```

```
  JOIN S ON R.empno=S.empno;

(c) SELECT R.empno, name, comm FROM R
    NATURAL JOIN S ON R.empno=S.empno;
```

Στην Εικόνα 6.24 παρουσιάζουμε τις σχέσεις R και S και τις σχέσεις P και Q και παραδείγματα εφαρμογής της πράξης division (διαίρεση).

Σχέσεις P Q P / Q

x	y
x_1	c
x_1	y_2
x_1	y_3
x_2	y_1
x_2	y_2
x_3	y_1
x_3	y_2
x_3	y_3
x_4	y_2
x_5	y_3

y
y_1
y_2
y_3

x
x_1
x_3

Σχέσεις R S R / S

A	B
1	2
2	f
x	e
x	q
2	q

A
e
q

A
x

Εικόνα 6.24 Δύο παραδείγματα εφαρμογής της πράξης Division

Η γλώσσα SQL δεν υποστηρίζει άμεσα την πράξη Division. Η υλοποίηση της με δήλωση της γλώσσας SQL μπορεί να γίνει με διάφορους τρόπους (βλέπε, π.χ., Rolland,1998, pp. 60, 108-110). Ακολουθεί παράδειγμα με χρήση της υποπρότασης EXISTS (Εικόνα 6.25). Η υλοποίηση έγινε στο προϊόν MySQL.

Supply Schema

Suppliers

Sid	Pid	Sname
101	1	Adams
102	1	Smith
101	3	Adams

Parts

Pid	Pname
1	Bolts
2	Screw Drivers
3	Nuts

103	2	Jones	4	Small Screws
102	2	Smith	5	Red Screws
102	3	Smith		
102	4	Smith		
102	5	Smith		

```
DROP DATABASE division_example;
CREATE DATABASE division_example;
USE division_example;
CREATE TABLE Suppliers(sid int, pid int, sname char(30));
CREATE TABLE Parts(pid int, pname char(30));
INSERT INTO suppliers VALUES
 (101, 1, "ADAMS"),
 (102, 1, "SMITH"),
 (101, 3, "ADAMS"),
 (103, 2, "JONES"),
 (102, 2, "SMITH"),
 (102, 3, "SMITH"),
 (102, 4, "SMITH"),
 (102, 5, "SMITH");
INSERT INTO Parts VALUES
  (1, "Bolts"),
  (2, "Screw drivers"),
  (3, "nuts"),
  (4, "small screws"),
  (5, "red screws");
SELECT * FROM suppliers;
SELECT * FROM parts;

# Δείξτε τους προμηθευτές που προμηθεύουν όλα τα ανταλλακτικά

SELECT DISTINCT x.Sid
FROM suppliers AS x
WHERE NOT EXISTS (
 SELECT *
 FROM parts AS y
 WHERE NOT EXISTS (
  SELECT *
   FROM suppliers AS z
   WHERE (z.Sid=x.Sid)
AND (z.pid=y.pid)));
```

Εικόνα 6.25 Υλοποίηση της πράξης Division με χρήση EXISTS στο προϊόν της MySQL

Ερωτήσεις, απαντήσεις και λυμένες ασκήσεις

Αναφέρατε τις βασικές πράξεις της σχεσιακής άλγεβρας

Σύμφωνα με τον Codd έχουμε οκτώ πράξεις της σχεσιακής άλγεβρας

Επιλογή (Selection): $R_1 = \sigma_E (R)$

Προβολή (projection): $R_1 = \pi_{a1,a2,a3,...,ak} (R)$

Καρτεσιανό γινόμενο (Cartesian Product or Times): $R_1 \times R_2$

Σύνδεση (join) σχέσεων : $R = R_1 \bowtie R_2$

Ένωση (Union)

Τομή (Intersect)

Διαφορά (Minus)

Διαίρεση (Division)

Γράψτε ορισμό των πράξεων Selection, Projection, Cartesian Product, join και δώστε παραδείγματα εφαρμογής στις σχέσεις της εικόνας 6.26.

STUDENTS

Studno	Student	Name	Semester
10	ΣΠΥΡΟΥ	ΣΠΥΡΟΣ	Γ
20	ΝΙΚΟΥ	ΝΙΚΟΣ	Δ

ENROL

Studno	Course	Enrol_date
10	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ Ι	10/3/2022
10	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΙΙ	9/3/2022
20	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ Ι	10/3/2022
20	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΙΙ	9/3/2022

Εικόνα 6.26 Σχέσεις STUDENTS (φοιτητές), ENROL (εγγραφές φοιτητών σε μαθήματα)

Επιλογή (Selection): $R_1 = \sigma_E (R)$

Το αποτέλεσμα της πράξης είναι μια σχέση R_1 που έχει την ίδια πολλαπλότητα με την R και περιλαμβάνει τις πλειάδες της R για τις οποίες ισχύει η συνθήκη E.

Παράδειγμα

Η σχέση $STUDENTS\_1 = \sigma_{semester='Γ' \text{ OR } semester='Δ'} (STUDENTS)$ περιέχει μόνο εκείνες τις πλειάδες της σχέσης R όπου το εξάμηνο έχει τιμή Γ ή Δ.

Προβολή (projection): $R_1 = \pi_{a1,a2,a3,...,ak} (R)$

Η σχέση R_1 έχει πολλαπλότητα k < N με χαρακτηριστικά a_1, a_2, ... a_k, όπου N είναι η πολλαπλότητα της σχέσης R, και τα χαρακτηριστικά περιλαμβάνονται στη σχέση R.

a_1, a_2, ... a_k

Παράδειγμα

Η σχέση $STUDENTS\_2 = \pi_{empno,\ surname,\ semester} (STUDENTS)$ δείχνει το χαρακτηριστικό (στήλη) ή τα χαρακτηριστικά (στήλες) της σχέσης (ή πίνακα) STUDENT που θέλουμε.

Καρτεσιανό γινόμενο (Cartesian Product or Times) R1 x R2

Για τον υπολογισμό του καρτεσιανού γινομένου συνδέουμε κάθε πλειάδα της R_1 με όλες τις πλειάδες της R_2 και δημιουργούμε έτσι πλειάδες με πολλαπλότητα N1 + N2, όπου N1, N2 είναι οι πολλαπλότητες των σχέσεων R_1 και R_2. Με μαθηματικό φορμαλισμό, αν η σχέση R_1 έχει σχήμα A_1 , A_2 .... A_N και η σχέση R2 έχει σχήμα B_1, B_2 .... B_M τότε το καρτεσιανό γινόμενο των δύο σχέσεων, συμβολικά R1 x R2, ορίζεται σαν μία σχέση με σχήμα (A_1 , A_2 .... A_N , B_1, B_2 .... B_M), δηλαδή,

$$R1 \times R2 = \{ (\alpha_1 , \alpha_2 .... \alpha_v , \beta_1, \beta_2 .... \beta_\mu) \mid (\alpha_1 , ...) \in R1 , (\beta_1 , ...) \in R2 \}$$

Παράδειγμα

Η σχέση STUDENTS_3 = STUDENT **x** ENROL προκύπτει αν συνδέσουμε κάθε πλειάδα της R_1 με όλες τις πλειάδες της R2.

Σύνδεση (join) σχέσεων : $R = R_1 \bowtie R_2$

Αν R_1, R_2 είναι σχέσεις με πολλαπλότητες N1 και N2 και με M κοινά χαρακτηριστικά των δύο σχέσεων κατασκευάζουμε την R που είναι μια σχέση με πολλαπλότητα N = N1 + N2 - M. Η σχέση R δημιουργείται με την παράθεση (concatenation) όλων των πλειάδων $t_1 \in R_1$ και $t_1 \in R_1$ που ικανοποιούν τη συνθήκη της σύνδεσης.

Παράδειγμα

Η σχέση STUDENTS_4 = STUDENT $\bowtie$ ENROL προκύπτει αν για όλες τις πλειάδες των δύο σχέσεων κάνουμε τα εξής: (1) Συνδέουμε κάθε πλειάδα της R_1 με όλες τις πλειάδες της R_2 (2) κρατάμε μόνον αυτές τις πλειάδες που έχουν σε όλα τα κοινά χαρακτηριστικά τις ίδιες τιμές (3) Στις πλειάδες που κρατήσαμε, κρατάμε τα κοινά χαρακτηριστικά μια φορά μόνον.

Στην Εικόνα 6.27 βλέπουμε τις σχέσεις emp (στοιχεία υπαλλήλων) και dept (στοιχεία τμημάτων). Δείξτε όνομα και διεύθυνση όλων των υπαλλήλων οι οποίοι εργάζονται στο τμήμα “Development”

Emp

EMPNO	ENAME	ADDRESS	DEPTNO
10	CLARKE	ATHENS	10
20	ADAMS	NEW YORK	20

Dept

DEPTNO	DNAME
10	DEVELOPMENT
20	RESEARCH

Εικόνα 6.27 Σχέσεις emp (υπάλληλοι), dept τμήματα)

Ακολουθεί η απάντηση.

DEVELOPMENT_DEPT <---- $\sigma_{dname="DEVELOPMENT"}$ (DEPT)

DEVELOPMENT_EMP <---- (DEVELOPMENT_DEPT $\bowtie_{DNO=DEPTNO}$ EMP)

RESULT <---- $\pi_{ENAME, ADDRESS}$(DEVELOPMENT_EMP)

6.2.6 Συναρτησιακές εξαρτήσεις και κανονικοποίηση

Στην ενότητα αυτή ορίζουμε τις συναρτησιακές εξαρτήσεις και παρουσιάζουμε τις ιδιότητές τους.

Αν $R(A_1, A_2, ..., A_v)$ ένα σχήμα σχέσης και X, Y υποσύνολα του $\{A_1, A_2, ..., A_v\}$ τότε λέμε ότι:

"το X καθορίζει συναρτησιακά το Y" ή

"το Y εξαρτάται συναρτησιακά από το X"

και γράφουμε:

X --> Y

αν και μόνον αν οποιοδήποτε και αν είναι το περιεχόμενο της R είναι αδύνατο να βρούμε δύο πλειάδες που να έχουν τις ίδιες τιμές για τα χαρακτηριστικά που περιλαμβάνονται στο X και διαφορετικές για αυτά που περιλαμβάνονται στο Y.

Οι συναρτησιακές εξαρτήσεις που ισχύουν σε ένα σύνολο χαρακτηριστικών καθορίζονται από το φυσικό νόημα των μεγεθών που παριστάνονται.

Παραδείγματα

Στο σύνολο χαρακτηριστικών:

{EPWNYMO, ONOMA, ARITMHT, EXAMHNO, LEKTIKO, KWD_MAT}

διακρίνουμε μπορούμε να δούμε τις παρακάτω συναρτησιακές εξαρτήσεις:

ARITMHT ---> EPWNYMO, ONOMA, EXAMHNO

φυσικό νόημα: ο αριθμός μητρώου καθορίζει μονοσήμαντα το φοιτητή ή δεν υπάρχουν δύο φοιτητές με τον ίδιο αριθμό μητρώου.

LEKTIKO --> KWD_MAT

KWD_MAT --> LEKTIKO

φυσικό νόημα: αμφιμονοσήμαντη αντιστοιχία κωδικού μαθήματος και λεκτικού (τίτλου μαθήματος).

EPWNYMO, ONOMA --> ARITMHT

φυσικό νόημα: δεν υπάρχουν δύο φοιτητές με το ίδιο επώνυμο και το ίδιο όνομα. Μια τέτοια υπόθεση δεν είναι και πολύ σωστή. Το ίδιο ισχύει και για τη συναρτησιακή εξάρτηση.

6.2.6.1 Ιδιότητες για τις συναρτησιακές εξαρτήσεις.

Αν U ένα σύνολο χαρακτηριστικών και W, X, Y, Z $\subseteq$ U (συχνά αναφέρεται και σαν παγκόσμια σχέση - universal relation) τότε ισχύουν:

Τα Αξιώματα του Armstrong

Αν Y $\subseteq$ X τότε X --> Y

Αν X --> Y τότε XZ --> YZ

Αν X --> Y και Y --> Z τότε X --> Z

και αποδεικνύονται, επίσης, τα

Θεωρήματα

Αν X --> Y και X --> Z τότε X --> YZ

Αν X --> Y και WY --> Z τότε WX --> Z

Αν X --> Y και Z $\subseteq$ Y τότε X --> Z

Σημειώστε ότι αν και μιλάμε για τα αξιώματα του Armstrong στην πραγματικότητα οι προτάσεις 1-6 αποδεικνύονται όλες από τον ορισμό της έννοιας της συναρτησιακής εξάρτησης είτε με ευθεία απόδειξη είτε με εις άτοπον απαγωγή. Δείτε για παράδειγμα και το κεφάλαιο 12 στην ελληνική έκδοση του συγγράμματος:

Elmasri, R., Navathe, S.B. (2016) Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων, 7η έκδοση, ISBN 978-960-531-343-2, εκδόσεις Δίαυλος.

6.2.6.2 Κλειδιά - Ευρετήρια

Αν έχουμε ένα σχήμα σχέσης τότε ονομάζουμε κλειδί (key) της σχέσης κάθε ελάχιστο υποσύνολο των χαρακτηριστικών της που οι τιμές τους καθορίζουν μονοσήμαντα κάθε πλειάδα της σχέσης.

Χρησιμοποιώντας την έννοια της συναρτησιακής εξάρτησης, μπορούμε να πούμε ότι:

Αν έχουμε το σχήμα σχέσης R(A_1,...., A_n) τότε το X $\subseteq$ { A_1,...., A_n } είναι κλειδί αν και μόνον αν:

- ισχύει η X --> A_1,...., A_n
- δεν υπάρχει Y $\subseteq$ X τέτοιο ώστε Y --> A_1,...., A_n

Παράδειγμα

Για το σχήμα σχέσης:

FOITHTES(EPWNYMO, ONOMA, ARITMHT, EXAMHNO)

το ARITMHT είναι κλειδί, γιατί:

ARITMHT --> EPWNYMO, ONOMA, ARITMHT, EXAMHNO

Αν δεχτούμε την:

EPWNYMO, ONOMA --> ARITMHT

τότε και το EPWNYMO, ONOMA είναι κλειδί.

Το {EPWNYMO, ONOMA, ARITMHT} δεν είναι κλειδί γιατί το γνήσιο υποσύνολό του ARITMHT είναι κλειδί.

Για το σχήμα σχέσης:

MATHIMATA(LEKTIKO, KWD_MAT)

και τα δύο χαρακτηριστικά είναι κλειδιά.

Επειδή για ένα σχήμα σχέσης μπορεί να υπάρχουν πολλά κλειδιά, τα ονομάζουμε συνήθως υποψήφια κλειδιά (candidate keys). Από αυτά επιλέγουμε κάποιο ως πρωτεύον (κύριο) κλειδί (primary key).

Ο όρος δευτερεύον κλειδί (secondary key) αναφέρεται συνήθως για κάθε σύνολο χαρακτηριστικών που έχει κάποια ιδιαίτερη φυσική σημασία, χωρίς απαραίτητα να είναι κλειδί.

Πολύ συχνά, για την διευκόλυνση του εντοπισμού των πλειάδων που πληρούν κάποια συνθήκη, δημιουργούμε κάποιο ευρετήριο (index).

Μπορείς να σκέφτεστε το ευρετήριο σαν ένα πίνακα με καταχωρίσεις της μορφής:

τιμή κλειδιού ---- αριθμός πλειάδας

Παράδειγμα

Για το σχήμα σχέσης:

FOITHTES (EPWNYMO, ONOMA, ARITMHT, EXAMHNO)

και για το κλειδί ARITMHT, το ευρετήριο θα είναι:

213 -- 1

346 -- 4

450 -- 3

610 -- 5

816 -- 2

Μπορούμε να δημιουργούμε ευρετήρια για το πρωτεύον και οποιοδήποτε δευτερεύον κλειδί.

6.2.6.3 Κανονικοποίηση - Σχεδίαση

Όταν σχεδιάζουμε μια βάση δεδομένων πρέπει να προφυλαχτούμε από μερικές κακοτοπιές:

- δεν πρέπει να υπάρχει επανάληψη αποθήκευσης πληροφοριών,
- η διαχείριση της ΒΔ θα πρέπει να είναι απαλλαγμένη από ανωμαλίες ενημέρωσης, εισαγωγής, διαγραφής.

Η κανονικοποίηση (normalization) μας επιτρέπει να κάνουμε μια σωστή σχεδίαση.

Ένα σχήμα σχέσης είναι στην:

1η Κανονική Μορφή (First Normal Form)

Όταν δεν υπάρχουν σε αυτήν επαναλήψιμα χαρακτηριστικά.

Οι σχέσεις όπως τις είδαμε μέχρι τώρα, είναι στην 1η κανονική μορφή.

2η Κανονική Μορφή (Second Normal Form)

Όταν είναι στην 1η Κανονική Μορφή και όλα τα χαρακτηριστικά εξαρτώνται συναρτησιακά από ολόκληρο το κλειδί.

3η Κανονική Μορφή (Third Normal Form)

Όταν είναι στη 2η Κανονική Μορφή και όλα τα χαρακτηριστικά εξαρτώνται άμεσα από το κλειδί και όχι μέσω τρίτου χαρακτηριστικού.

Κανονική Μορφή Boyce-Codd (BCNF)

Ένα σχήμα σχέσης R είναι στην BCNF αν για κάθε συναρτησιακή εξάρτηση X --- > A που εμφανίζεται στο R , το X είναι ένα υπερκλειδί του R (Navathe ,τομ.2 , σελ. 57) .

Ο όρος **υπερκλειδί (superkey)** δηλώνει ένα υποσύνολο χαρακτηριστικών της σχέσης R. Ένα υπερκλειδί είναι υποψήφιο κύριο κλειδί (candidate key) για τη σχέση μόνο αν είναι ταυτόχρονα ελάχιστος συνδυασμός χαρακτηριστικών για τον οποίο ισχύει η συναρτησιακή εξάρτηση.

Ακολουθεί μία απλή συνταγή για να φέρουμε ένα σύνολο χαρακτηριστικών στην Κανονική Μορφή BCNF, άρα και στην 3NF:

1) Γράφουμε όλες τις συναρτησιακές εξαρτήσεις.
2) Πρέπει να σημειώσουμε ότι υπάρχει περίπτωση η αντιστοίχιση κάποιων χαρακτηριστικών (attributes) να είναι απαραίτητη χωρίς οποιοδήποτε από αυτά να είναι εξηρτημένο από το άλλο. Τότε τα χαρακτηριστικά αυτά αποτελούν ξεχωριστό σχήμα σχέσης (relation scheme).
3) Διαγράφουμε όσες από αυτές έπονται από τις άλλες (με τις ιδιότητες των συναρτησιακών εξαρτήσεων).
4) Συμπτύσσουμε σε μια, όλες τις συναρτησιακές εξαρτήσεις που έχουν το ίδιο αριστερό μέρος (με την ιδιότητα
5) Αν έχουμε ζεύγη της μορφής X --> Y, Y --> X κρατούμε το ένα από αυτά.
6) Κάθε μια σχέση εξάρτησης μας δίνει μια σχέση. Τα χαρακτηριστικά που βρίσκονται στο αριστερό μέρος μας δίνουν το κλειδί.

Η συνταγή αυτή κατασκευάζει την κανονική μορφή Boyce-Codd που είναι ισχυρότερη (συνεπάγεται την τρίτη κανονική μορφή).

Παράδειγμα 1

Δίδεται το σύνολο χαρακτηριστικών:

{EPWNYMO, ONOMA, ARITMHT, EXAMHNO, KWD_MAT, LEKTIKO}

Όπως είδαμε πιο πριν, ισχύουν οι παρακάτω συναρτησιακές εξαρτήσεις:

ARITMHT --> EPWNYMO

ARITMHT --> ONOMA

ARITMHT --> EXAMHNO

LEKTIKO --> KWD_MAT

KWD_MAT --> LEKTIKO

Θα πρέπει ακόμα να περιγράψουμε κάπως την αντιστοίχιση φοιτητών (ή των αριθμών μητρώου που έχουν) με τα μαθήματα στα οποία γράφονται.

Για την περίπτωση αυτή χρησιμοποιούμε ένα "ψευδοχαρακτηριστικό" και γράφουμε:

ARITMHT, KWD_MAT --> θ

Το χαρακτηριστικό αυτό (θ) θα το "ξεχάσουμε" στη συνέχεια. Αυτές οι συναρτησιακές εξαρτήσεις είναι ανεξάρτητες μεταξύ τους. Ετσι, στο βήμα 2 δεν έχουμε να κάνουμε τίποτε. Στο βήμα 3 οι τρεις πρώτες συμπτύσσονται στην:

ARITMHT --> EPWNYMO, ONOMA, EXAMHNO

Στο βήμα 4 απορρίπτουμε την LEKTIKO --> KWD_MAT.

Έτσι, καταλήγουμε με τις:

ARITMHT --> EPWNYMO, ONOMA, EXAMHNO

KWD_MAT --> LEKTIKO

ARITMHT, KWD_MAT --> Θ

που μας οδηγούν στα σχήματα σχέσης:

FOITHTES(ARITMHT, EPWNYMO, ONOMA, EXAMHNO)

(κλειδί: ARITMHT)

MATHIMATA(KWD_MAT, LEKTIKO)

(κλειδί: KWD_MAT ή LEKTIKO)

EGGRAFES(ARITMHT, KWD_MAT)

(κλειδί: ARITMHT, KWD_MAT)

Παράδειγμα 2

Στη βάση δεδομένων του τμήματος μηχανογράφησης μιας εταιρείας έχουμε τα εξής χαρακτηριστικά:

ONOMA: ονοματεπώνυμο εργαζόμενου

GLWSSA: γλώσσα προγραμματισμού που γνωρίζει

XROXRH: χρόνια που χρησιμοποιεί μια γλώσσα προγραμματισμού

THESH: θέση που έχει στο τμήμα

HMRMIS: ημερομίσθιο

XRYP: χρόνια υπηρεσίας στην εταιρεία

ERGO: μηχανογραφικό έργο σε εξέλιξη

YPERG: υπεύθυνος έργου

1) **Συναρτησιακές Εξαρτήσεις**

   ONOMA --> THESH (μια θέση για κάθε εργαζόμενο)

   ONOMA --> HMRMIS (και ένα ημερομίσθιο)

   ONOMA --> XRYP(και χρόνια υπηρεσίας)

   ONOMA, GLWSSA --> XROXRH (ένας εργαζόμενος χρησιμοποιεί μια γλώσσα για ορισμένο διάστημα)

   ONOMA, ERGO --> θ (εκχώρηση εργαζομένων στα έργα)

   ERGO --> YPERG (κάθε έργο έχει έναν υπεύθυνο)

2) **Οι συναρτησιακές εξαρτήσεις είναι ανεξάρτητες**
3) ONOMA --> THESH, HMRMIS, XRYP
4) Σχήμα:

   PROSWPIKO(ONOMA, THESH, HMRMIS, XRYP)

   (κλειδί: ONOMA)

PROSONTA(ONOMA, GLWSSA, XROXRH)

(κλειδί: ONOMA, GLWSSA)

EKXWRHSEIS(ONOMA, ERGO)

(κλειδί: ONOMA, ERGO)

ERGA(ERGO, YPERG)

(κλειδί: ERGO)

Παράδειγμα 3 Βάση δεδομένων Προσωπικού - Μισθοδοσίας

ONOMA : ονοματεπώνυμο εργαζόμενου

XRYP : χρόνια υπηρεσίας στην εταιρεία

THESH : θέση που έχει στο τμήμα

BASMIS : βασικός μισθός

TITSP : τίτλοι σπουδών

ANEPID : ανθυγιεινό επίδομα

ARPAID : αριθμός παιδιών

ONPAID : όνομα παιδιού

HLPAID : ηλικία παιδιού

1) **Συναρτησιακές Εξαρτήσεις**

ONOMA --> THESH(μια θέση για κάθε εργαζόμενο)

ONOMA --> XRYP (και χρόνια υπηρεσίας)

ONOMA --> ARPAID (και αριθμός παιδιών)

ONOMA --> TITSP (και σπουδές)

ONOMA --> ANEPID (και ανθ. επίδομα)

THESH --> ANEPID(ανθ. επίδομα ανάλογα με τη θέση)

ONOMA, ONPAID --> HLPAID (το κάθε παιδί ενός εργαζόμενου έχει την ηλικία του)

ONOMA → BASMIS (βασικός μισθός σε προσωπική συμφωνία)

2) Από τις ONOMA --> THESH και THESH --> ANEPID μπορούμε να πάρουμε την

ONOMA --> ANEPID (ιδιότητα 3).

Γι' αυτό και διαγράφουμε την τελευταία.

3) ONOMA --> THESH, XRYP, ARPAID, TITSP, BASMIS

THESH --> ANEPID

ONOMA, ONPAID --> HLPAID

4) **Σχήμα:**

PROSWPIKO(ONOMA, THESH, XRYP, ARPAID, TITSP, BASMIS)

(κλειδί: ONOMA)

EPIDOMA(THESH, ANEPID)

(κλειδί: THESH)

OIKOG_KAT(ONOMA, ONPAID, HLPAID)

(κλειδί: ONOMA, ONPAID)

Το παραπάνω σχήμα βασίζεται στην υπόθεση ότι ο βασικός μισθός καθορίζεται με προσωπική συμφωνία με τον εργαζόμενο. Αν κάνουμε την υπόθεση ότι ο βασικός μισθός εξαρτάται από τη θέση (αρκετά συνηθισμένη περίπτωση) τότε έχουμε την εξάρτηση:

THESH --> BASMIS

Από αυτήν και την ONOMA --> THESH συνάγεται η ONOMA --> BASMIS η οποία και διαγράφεται. Το σχήμα τώρα γίνεται:

PROSWPIKO(ONOMA, THESH, XRYP, ARPAID, TITSP)

(κλειδί: ONOMA)

MISTHOS(THESH, BASMIS, ANEPID) (κλειδί: THESH)

OIKOG_KAT(ONOMA, ONPAID, HLPAID)

(κλειδί: ONOMA, ONPAID)

Μια άλλη συνηθισμένη περίπτωση είναι να καθορίζεται ο βασικός μισθός από τη θέση και τις σπουδές οπότε έχουμε την εξάρτηση:

THESH, TITSP --> BASMIS

Και σ' αυτήν την περίπτωση η ONOMA --> BASMIS διαγράφεται, γιατί συνάγεται από τις άλλες:

Από τις ONOMA --> THESH και ONOMA --> TITSP

με την ιδιότητα 4 παίρνουμε

ONOMA --> THESH, TITSP

Από αυτήν και την THESH, TITSP --> BASMIS με την ιδιότητα 3 παίρνουμε την

ONOMA --> BASMIS.

Το σχήμα τώρα γίνεται:

PROSWPIKO(ONOMA, THESH, XRYP, ARPAID, TITSP)

(κλειδί: ONOMA)

MISTHOS(THESH, TITSP, BASMIS)

(κλειδί: THESH, TITSP)

EPIDOMA(THESH, ANEPID) (κλειδί: THESH)

OIKOG_KAT(ONOMA, ONPAID, HLPAID)

(κλειδί: ONOMA, ONPAID)

Παράδειγμα 4 Ακολουθεί άλλο ένα "σχολικό" παράδειγμα:

ON_FOITH : ονοματεπώνυμο φοιτητή

DIDASKWN : διδάσκων

MATHIMA : όνομα μαθήματος

BATHMOS : βαθμός φοιτητή στο μάθημα

WRA : ώρα (προγράμματος) για το μάθημα

AITHOYSA : αίθουσα που γίνεται το μάθημα

1) **Συναρτησιακές Εξαρτήσεις:**

MATHIMA --> DIDASKWN

(ένας διδάσκων ανά μάθημα)

MATHIMA, ON_FOITH --> BATHMOS

(ένας βαθμός ανά φοιτητή ανά μάθημα)

WRA, AITHOYSA --> MATHIMA

(κάποια ώρα σε μια αίθουσα ένα μάθημα)

WRA, ON_FOITH --> AITHOYSA

(κάποια ώρα ένας φοιτητής σε μια αίθουσα)

WRA, DIDASKWN --> AITHOYSA

(κάποια ώρα ένας διδάσκων σε μια αίθουσα)

WRA, ON_FOITH --> MATHIMA

(κάποια ώρα ένας φοιτητής σε ένα μάθημα)

WRA, DIDASKWN --> MATHIMA

(κάποια ώρα ένας διδάσκων σε ένα μάθημα)

WRA, MATHIMA --> AITHOYSA

(κάποια ώρα ένα μάθημα σε μια αίθουσα)

2) **Οι συναρτησιακές εξαρτήσεις δεν είναι ανεξάρτητες:**

α) Από την (i) και την (v) συνάγεται η (viii):

Από την (i) MATHIMA --> DIDASKWN με την ιδιότητα 2 συνάγουμε την

WRA, MATHIMA --> WRA, DIDASKWN

Από αυτήν και την (v) WRA, DIDASKWN --> AITHOYSA με την ιδιότητα 3 συνάγουμε την (viii) WRA, MATHIMA --> AITHOYSA

β) Από τις (v) και (iii) συνάγεται η (vii):

Με την ιδιότητα 1 παίρνουμε WRA, DIDASKWN --> WRA

Από αυτήν και την (v) WRA, DIDASKWN --> AITHOYSA με την ιδιότητα 4 συνάγουμε την WRA,

DIDASKWN --> WRA, AITHOYSA

Από αυτήν και την (iii) WRA, AITHOYSA --> MATHIMA με την ιδιότητα 3 συνάγουμε την (vii) WRA,

DIDASKWN --> MATHIMA

Όπως ακριβώς στη β) μπορείς να δεις και ότι:

γ) Από τις (iv) και (iii) συνάγεται η (vi)

δ) Από τις (vi) και (viii) συνάγεται η (iv)

Μπορούμε λοιπόν να αγνοήσουμε τις (vi), (vii) και (viii).

3) **Από τις (i) και (iii) συνάγεται η:**

WRA, AITHOYSA --> WRA, DIDASKWN(iii')

WRA, DIDASKWN --> WRA, AITHOYSA (v')

Αν αντικαταστήσουμε στην (iii) τα WRA, AITHOYSA με τα WRA, DIDASKWN

παίρνουμε την (iii') WRA, DIDASKWN --> MATHIMA

Αυτή συμπτύσσεται με την (v) στην:

WRA, DIDASKWN --> AITHOYSA, MATHIMA

4) **Σχήμα:**

EKXWRHSEIS(MATHIMA, DIDASKWN)(κλειδί: MATHIMA)

APOT_MATH(MATHIMA, ON_FOITH, BATHMOS)

(κλειδί: MATHIMA, ON_FOITH)

PROG_DID(WRA, DIDASKWN, AITHOYSA, MATHIMA)

(κλειδί: WRA, AITHOYSA ή WRA, DIDASKWN)

PROG_FOIT(WRA, ON_FOITH, AITHOYSA) (κλειδί: WRA, ON_FOIT)

6.2.6.4 Σενάριο χρήσης

Θέλετε να σχεδιάστε τη βάση δεδομένων για τις παραγγελίες που εκκρεμούν σε μια βιομηχανία. Κάθε παραγγελία έχει τα εξής χαρακτηριστικά:

- ΑΡΙΘΜΟΣ : 5-ψήφιος ακέραιος που χαρακτηρίζει μονοσήμαντα κάθε παραγγελία
- ΕΠΩΝΥΜΙΑ : επωνυμία της επιχείρησης ή ονοματεπώνυμο του προσώπου που έδωσε την παραγγελία
- ΑΦΜ : 13-ψήφιος ακέραιος αριθμός φορολογικού μητρώου αυτού που έδωσε την παραγγελία
- ΔΙΕΥΘΥΝ : ταχυδρομική διεύθυνση αυτού που έδωσε την παραγγελία
- ΗΜΕΡΟΜ : ημερομηνία που δόθηκε η παραγγελία
- ΕΙΔΟΣ : 12-ψήφιος κωδικός που χαρακτηρίζει μονοσήμαντα το είδος που παραγγέλλεται
- ΠΟΣΟΤΗΤΑ : για κάθε είδος που παραγγέλλεται

Κάθε παραγγελία έχει κατά μέσο όρο 5 διαφορετικά είδη. Οι πελάτες της βιομηχανίας δεν είναι περισσότεροι από 500. Οι παραγγελίες που εκκρεμούν ανά πάσα στιγμή δεν είναι περισσότερες από 1000.

Θα πρέπει να σχεδιάσετε τα αρχεία της βάσης και να δώσετε μια εκτίμηση των απαιτήσεων σε βοηθητική μνήμη. Η υλοποίηση να σχεδιασθεί χωρίς ευρετήρια. Το ΣΔΒΔ που θα χρησιμοποιήσετε δουλεύει στη βάση "1 σχέση --> 1 αρχείο". Οι επιβαρύνσεις των αρχείων από το ΣΔΒΔ θεωρούνται αμελητέες. Ακολουθεί η σχεδίαση.

Συναρτησιακές Εξαρτήσεις:

1) ΑΡΙΘΜΟΣ --> ΕΠΩΝΥΜΙΑ
2) ΑΡΙΘΜΟΣ --> ΑΦΜ
3) ΑΡΙΘΜΟΣ --> ΔΙΕΥΘΥΝ
4) ΑΡΙΘΜΟΣ --> ΗΜΕΡΟΜ
5) ΑΡΙΘΜΟΣ, ΕΙΔΟΣ --> ΠΟΣΟΤΗΤΑ
6) ΕΠΩΝΥΜΙΑ --> ΑΦΜ
7) ΑΦΜ --> ΕΠΩΝΥΜΙΑ
8) ΕΠΩΝΥΜΙΑ --> ΔΙΕΥΘΥΝ

Η (2) συνάγεται από τις (1) και (6).

Η (3) συνάγεται από τις (1) και (8).

Από τις (6) και (7) κρατάμε την (6).

Τελικά:

1. ΑΡΙΘΜΟΣ --> ΕΠΩΝΥΜΙΑ
4. ΑΡΙΘΜΟΣ --> ΗΜΕΡΟΜ
5. ΑΡΙΘΜΟΣ, ΕΙΔΟΣ --> ΠΟΣΟΤΗΤΑ
6. ΕΠΩΝΥΜΙΑ --> ΑΦΜ
8. ΕΠΩΝΥΜΙΑ --> ΔΙΕΥΘΥΝ

Σχήμα:

ΠΑΡΑΓΓΕΛΙΕΣ(ΑΡΙΘΜΟΣ, ΕΠΩΝΥΜΙΑ, ΗΜΕΡΟΜ)

κλ.: ΑΡΙΘΜΟΣ

ΠΑΡ_ΕΙΔΟΣ(ΑΡΙΘΜΟΣ, ΕΙΔΟΣ, ΠΟΣΟΤΗΤΑ)

κλ.: ΑΡΙΘΜΟΣ, ΕΙΔΟΣ

ΠΕΛΑΤΕΣ(ΕΠΩΝΥΜΙΑ, ΑΦΜ, ΔΙΕΥΘΥΝ)

κλ.: ΕΠΩΝΥΜΙΑ ή ΑΦΜ

Υλοποίηση: (μήκη σε χαρακτήρες)

Αρχείο : ΠΑΡΑΓΓΕΛΙΕΣ

Εγγραφή:

ΑΡΙΘΜΟΣ: Numeric, μήκος: 5

ΕΠΩΝΥΜΙΑ: Char/Text, μήκος: 40

ΗΜΕΡΟΜ: Date, μήκος: 8

Μήκος Εγγραφής: 53

Πλήθος Εγγραφών: 1000

Απαίτηση σε βοηθ. μνήμη: 53000 χαρακτήρες περίπου

Αρχείο: ΠΑΡ_ΕΙΔΟΣ

Εγγραφή:

ΑΡΙΘΜΟΣ : Numeric, μήκος: 5

ΕΙΔΟΣ : Numeric, μήκος: 12

ΠΟΣΟΤΗΤΑ : Numeric, μήκος: 6

Μήκος Εγγραφής: 23

Πλήθος Εγγραφών: 1000 παραγγ. Χ 5 είδη/παραγγ. = 5000

Απαίτηση σε βοηθ. μνήμη: 5000 Χ 23 = 115000 χαρακτήρες περίπου

Αρχείο: ΠΕΛΑΤΕΣ

Εγγραφή:

ΕΠΩΝΥΜΙΑ : Char/Text, μήκος: 40

ΑΦΜ : Numeric, μήκος: 13

ΔΙΕΥΘΥΝ : Char/Text, μήκος: 70

Μήκος Εγγραφής : 123

Πλήθος Εγγραφών : 500

Απαίτηση σε βοηθ. μνήμη : 500 Χ 123 = 61500 χαρακτήρες περίπου

Σύνολο απαιτήσεων σε βοηθητική μνήμη:

53000 + 115000 + 61500 = 229500 χαρακτήρες περίπου

Άσκηση 1

Σε ένα σύγχρονο γραφείο όλα τα έγγραφα γράφονται σε Ηλεκτρονικό Υπολογιστή με τη χρήση λεξικού επεξεργαστή (word processor) και κρατιούνται σε μαγνητική μορφή. Για να διευκολύνεται η αναζήτηση στο "ηλεκτρονικό αρχείο" χρειαζόμαστε μια βάση στοιχείων όπου θα υπάρχουν τα στοιχεία που περιγράφουν το κάθε έγγραφο.

Ας δούμε τα στοιχεία που περιγράφουν ένα έγγραφο:

- Αριθμός Εξερχόμενου Εγγράφου (ΑΕΕ), κάτι σαν αριθμός πρωτοκόλλου,
- Χρόνος Εξόδου. Συνήθως σαν χρόνος εξόδου μπαίνει η ημερομηνία, αλλά σε πιο σύγχρονα μέσα μετάδοσης (ηλεκτρονικό ταχυδρομείο) έχει νόημα και η ώρα.

Ο ΑΕΕ και ο Χρόνος Εξόδου χαρακτηρίζουν μονοσήμαντα κάθε έγγραφο.

- Παραλήπτης
- Διεύθυνση Παραλήπτη

- Σχετικό Έγγραφο
- Συντάκτης Εγγράφου
- Δακτυλογράφος Εγγράφου
- Λέξη-Κλειδί. Για κάθε έγγραφο αποθηκεύονται μια ή περισσότερες λέξεις-κλειδιά που "περιγράφουν" το περιεχόμενο του εγγράφου.
- Όνομα Αρχείου όπου είναι αποθηκευμένο το έγγραφο σε μαγνητική μορφή.

Να υποθέσουμε ακόμη τα εξής:

- Ο αριθμός εξερχόμενου εγγράφου (ΑΕΕ) είναι 5-ψήφιος.
- Ο χρόνος εξόδου είναι μόνον η ημερομηνία.
- Για το όνομα του παραλήπτη αρκούν 30 χαρακτήρες και
- για τη διεύθυνση παραλήπτη 50 χαρακτήρες.
- Κάθε σχετικό έγγραφο περιγράφεται από κάποιο κωδικό με 40 χαρακτήρες.
- Το όνομα του συντάκτη εγγράφου περιγράφεται με 4 χαρακτήρες.
- Παρόμοια και το όνομα δακτυλογράφου εγγράφου.
- Για κάθε λέξη-κλειδί αρκούν 20 χαρακτήρες.
- Το όνομα αρχείου αποτελείται από 24 χαρακτήρες.

Κάθε έγγραφο έχει κατά μέσον όρο:

- Δύο (2) παραλήπτες,
- Δύο (2) σχετικά έγγραφα,
- Πέντε (5) λέξεις-κλειδιά.

Η Βάση Στοιχείων που θα σχεδιάσετε θα πρέπει να κρατάει τα στοιχεία εγγράφων ολόκληρου έτους (250 εργάσιμες). Κάθε εργάσιμη ημέρα φεύγουν από το γραφείο δέκα (10) έγγραφα κατά μέσον όρο. Θα πρέπει να σχεδιάσετε τα αρχεία της βάσης και να δώσετε μια εκτίμηση των απαιτήσεων σε βοηθητική μνήμη. Η υλοποίηση να σχεδιασθεί χωρίς ευρετήρια. Το ΣΔΒΔ που θα χρησιμοποιήσετε δουλεύει στη βάση "1 σχέση - 1 αρχείο". Οι επιβαρύνσεις των αρχείων από το ΣΔΒΔ θεωρούνται αμελητέες.

Άσκηση 2

Για ένα απλοποιημένο σύστημα αναζήτησης και δανεισμού βιβλίων σε μια βιβλιοθήκη μπορούμε να χρησιμοποιήσουμε τα εξής χαρακτηριστικά:

- Αριθμός εισαγωγής: Αριθμός που χαρακτηρίζει μονοσήμαντα κάθε αντικείμενο που εισέρχεται στη βιβλιοθήκη.
- ISBN (International Standard Book Number): Αριθμός που χαρακτηρίζει μονοσήμαντα κάθε βιβλίο που εκδίδεται στον κόσμο.
- Τίτλος: Τίτλος βιβλίου.
- Συγγραφέας: Επώνυμο και αρχικά του συγγραφέα (ή επιμελητή έκδοσης) ενός βιβλίου.
- Γλώσσα: Γλώσσα που είναι γραμμένο το κείμενο του βιβλίου.
- Λέξη-Κλειδί: Λέξη (ή φράση) που χαρακτηρίζει το περιεχόμενο του βιβλίου.
- Κωδικός χρήστη: Αριθμός που χαρακτηρίζει μονοσήμαντα κάθε χρήστη της βιβλιοθήκης.
- Ονοματεπώνυμο χρήστη

- Διεύθυνση χρήστη
- Τηλέφωνο χρήστη

Παρατήρηση:

Στο παρακάτω παράδειγμα βλέπετε τη σχέση του αριθμού εισαγωγής με το ISBN.

Το βιβλίο: Principles of Database Design του S. Bing Yao (Ed.) έχει ISBN 0-13-708876-0. Μια βιβλιοθήκη προμηθεύεται τρία (3) αντίτυπα του βιβλίου και τους δίνει αριθμούς εισαγωγής: 45516/001.64, 45517/001.64 και 45518/001.64. Δηλαδή, τα τρία αντίτυπα έχουν διαφορετικούς αριθμούς εισαγωγής που δίνονται από τη βιβλιοθήκη, ενώ έχουν το ίδιο ISBN, που δίνεται από ειδική υπηρεσία, με βάση διεθνείς κανόνες.

Η βάση στοιχείων που εξετάζουμε θα πρέπει, εκτός από την αρχειοθέτηση βιβλίων και χρηστών, να μπορεί να χειριστεί και τον δανεισμό βιβλίων, δηλαδή να κάνει αντιστοίχιση βιβλίων και χρηστών που τα έχουν δανειστεί.

Για τη σχεδίαση υπόθεσε ακόμα τα εξής:

- Ο αριθμός εισαγωγής καταλαμβάνει 16 θέσεις.
- Ο ISBN καταλαμβάνει 13 θέσεις.
- Ο τίτλος καταλαμβάνει μέχρι 200 χαρακτήρες.
- Ο συγγραφέας καταχωρίζεται με 20 το πολύ χαρακτήρες.
- Η γλώσσα παριστάνεται με κωδικό 3 γραμμάτων.
- Για κάθε λέξη-κλειδί αρκούν 20 χαρακτήρες.
- Ο κωδικός χρήστη είναι πενταψήφιος ακέραιος.
- Το ονοματεπώνυμο χρήστη καταλαμβάνει 32 χαρακτήρες το πολύ.
- Η διεύθυνση χρήστη με 50 χαρακτήρες το πολύ.
- Το τηλέφωνο χρήστη με 10 χαρακτήρες το πολύ.

Κάθε βιβλίο έχει κατά μέσον όρο:

- 1.2 συγγραφείς,
- Τρεις (3) λέξεις-κλειδιά
- Κάθε χρήστης έχει κατά μέσον όρο 4 δανεισμένα βιβλία.
- Η βιβλιοθήκη έχει συνολικά περί τις 20000 βιβλία και περίπου 1000 χρήστες.

Θα πρέπει να σχεδιάσετε τα αρχεία της βάσης και να δώσετε μια εκτίμηση των απαιτήσεων σε βοηθητική μνήμη. Η υλοποίηση να σχεδιασθεί χωρίς ευρετήρια. Το ΣΔΒΔ που θα χρησιμοποιήσεις δουλεύει στη βάση "1 σχέση - 1 αρχείο". Οι επιβαρύνσεις των αρχείων από το ΣΔΒΔ θεωρούνται αμελητέες.

Ενδεικτική Βιβλιογραφία στην οποία βασίστηκε η ενότητα των συναρτησιακών εξαρτήσεων

Date, C.J. (2004), An introduction to database systems, Addison-Wesley, ISBN:978-0-321-19784-9

Ullman, J.D. (1988) Principles of database and knowledge-base systems, Computer Science Press

Ullman, J.D. (1982) Principles of database systems, Prentice-Hall, 2nd edition

Ullman, J.D., Widom, J. (2014) A first course in database systems, Prentice-Hall

Elmasri, R., Navathe, S.B. (2016) Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων, 7η έκδοση, ISBN 978-960-531-343-2, εκδόσεις Δίαυλος.

Page, A. J. (1991) Relational databases, Concepts, Selection and Implementation, Sigma Press

Rolland, F. D. (1998) The essence of databases , Prentice-Hall

Yao, S. Bing, (Ed.) (1985) Principles of database design, Prentice-Hall. Ειδικότερα χρησιμοποιήθηκε το άρθρο, Y.E. Lien, Relational database design, pp. 211-252.

6.2.7 Κανονικοποίηση. Παράδειγμα εφαρμογής απλών κανόνων κατασκευής της τρίτης κανονικής μορφής

Παραθέτουμε τρεις απλούς κανόνες.

Κανόνας 1

Σε κάθε πίνακα δεν πρέπει να εμφανίζονται σύνθετα πεδία ορισμού ανά στήλη. Σε περιγραφική προσέγγιση, ο πίνακας δεν έχει σύνθετες στήλες και σε κάθε γραμμή του πίνακα κάθε «θέση» της πρέπει να περιέχει ακριβώς μια τιμή ή «τιμή» NULL

Κανόνας 2

Αν το κύριο κλειδί του πίνακα, δηλαδή το κλειδί που ορίζει μονοσήμαντα όλες τις στήλες του, είναι σύνθετο (αποτελείται από περισσότερες από μία στήλες) και ένα τμήμα του ορίζει μονοσήμαντα κάποιες στήλες πρέπει το τμήμα αυτό και οι στήλες που ορίζει να αποτελέσουν ξεχωριστό πίνακα.

Κανόνας 3

Σε κάθε πίνακα, όλες οι στήλες πρέπει να αντιστοιχούν απ' ευθείας στο κλειδί χωρίς μεταβατικές εξαρτήσεις διαμέσου άλλων στηλών.

Στη βιβλιογραφία, συχνά οι κανόνες διατυπώνονται με τους όρους **σχέση-relation** (αντί του όρου πίνακας) και **χαρακτηριστικό-attribute** (αντί στήλη).

Παράδειγμα κανονικοποίησης (normalization)

Έστω ότι η ανάλυση δεδομένων της εταιρείας Deep Learning οδήγησε στα παρακάτω στοιχεία:

Empno=κωδικός υπαλλήλου, Fname=όνομα, Surname=επώνυμο, JobNo=κωδικός παρούσης θέσεως υπαλλήλου, Jobname=θέση υπαλλήλου π.χ., ΠΩΛΗΤΗΣ, DeptNo=κωδικός τμήματος υπαλλήλου, Dname=τμήμα υπαλλήλου, Dloc=έδρα τμήματος υπαλλήλου, Sal=μισθός υπαλλήλου, Comm=Προμηθεια υπαλλήλου, Projno=κωδικός έργου στο οποίο εργάζεται ο υπάλληλος, Pdescr=έργο, Ploc=έδρα έργου, Ptime=ποσοστό χρόνου που ο υπάλληλος απασχολείται σε ένα έργο.

Περιορισμοί

1) Ένας υπάλληλος έχει μία θέση (jobno), εργάζεται σε ένα τμήμα (deptno), μπορεί να έχει προμήθεια (comm) ή να μην έχει, και απασχολείται σε ένα ή περισσότερα έργα (projno) για κάποιο ποσοστό του χρόνου του (ptime).
2) Ένα τμήμα έχει μοναδικό κωδικό (deptno), μία επωνυμία, μία έδρα και μπορεί να έχει πολλούς υπαλλήλους.
3) Σε κάποια θέση π.χ., ΠΩΛΗΤΗΣ μπορεί να απασχολούνται πολλοί υπάλληλοι.
4) Σε κάθε έργο απασχολούνται πολλοί υπάλληλοι.
5) Ο μισθός (sal) εξαρτάται από τη θέση (job).

Employee

Empno	Fname	Surname	JobNo	Jobname	DeptNo	Dname	dloc	Sal	comm	Projno	Pdescr	ploc	ptime
10	ΣΠΥΡΟΣ	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	ΒΙΛΙΑ	2200	250	1000	PAYROLL	ΘΗΒΑ	60
10	ΣΠΥΡΟΣ	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	ΒΙΛΙΑ	2200	250	2000	MEDIA	ΒΙΛΙΑ	40
20	ΝΙΚΟΣ	DATE	200	ΑΝΑΛΥΤΗΣ	60	ΠΡΟΣΩΠΙΚΟ	ΘΗΒΑ	2000	100	1000	PAYROLL	ΘΗΒΑ	100
30	ΝΙΚΟΣ	ΜΑΡΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΛΟΓΙΣΤΗΡΙΟ	ΘΗΒΑ	1000		2000	MEDIA	ΒΙΛΙΑ	100
40	ΜΑΡΚΟΣ	ΜΑΡΚΟΥ	100	ΠΩΛΗΤΗΣ	80	MARKETING	ΘΗΒΑ	2200	500	2000	MEDIA	ΒΙΛΙΑ	100

Εναλλακτικά θα μπορούσαμε να είχαμε περισσότερους πίνακες στην πρώτη κανονική μορφή.

Empl

Empno	Fname	Surname	JobNo	Jobname	DeptNo	Sal	comm	Projno	Pdescr	ploc	ptime
10	ΣΠΥΡΟΣ	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	2200	250	1000	PAYROLL	ΘΗΒΑ	60
10	ΣΠΥΡΟΣ	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	2200	250	2000	MEDIA	ΒΙΛΙΑ	40
20	ΝΙΚΟΣ	DATE	200	ΑΝΑΛΥΤΗΣ	60	2000	100	1000	PAYROLL	ΘΗΒΑ	100
30	ΝΙΚΟΣ	ΜΑΡΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	1000		2000	MEDIA	ΒΙΛΙΑ	100
40	ΜΑΡΚΟΣ	ΜΑΡΚΟΥ	100	ΠΩΛΗΤΗΣ	80	2200	500	2000	MEDIA	ΒΙΛΙΑ	100

Dept

DeptNo	Dname	dloc
50	ΠΩΛΗΣΕΙΣ	ΒΙΛΙΑ
50	ΠΩΛΗΣΕΙΣ	ΒΙΛΙΑ
60	ΠΡΟΣΩΠΙΚΟ	ΘΗΒΑ
70	ΛΟΓΙΣΤΗΡΙΟ	ΘΗΒΑ
80	MARKETING	ΘΗΒΑ

Εικόνα 6.28 Πρώτη κανονική μορφή ως ένας πίνακας με κύριο κλειδί σύνθετο που αποτελείται από τις στήλες (empno, projno). Εναλλακτική σχεδίαση που αποτελείται από 2 πίνακες.

Βήμα 1: Τοποθετούμε όλα τα χαρακτηριστικά σε μία σχέση (έναν πίνακα). Υπάρχει βέβαια η δυνατότητα να έχουμε περισσότερες σχέσεις στην πρώτη κανονική μορφή (Εικόνα 6.26).

Βήμα 2: Εξετάζουμε ποιο χαρακτηριστικό ή χαρακτηριστικά (στήλη ή στήλες) του κλειδιού της πρώτης κανονικής μορφής «καθορίζει» (καθορίζουν) από μόνο του κάθε τμήμα του κύριου κλειδιού της πρώτης κανονικής μορφής. «Χωρίζουμε» ανάλογα σε πίνακες τον πίνακα της πρώτης κανονικής μορφής**.** Επομένως, θα εξετάσουμε αν οι ακόλουθες στήλες και οι συνδυασμοί αυτών μπορούν να αποτελέσουν κύριο κλειδί για κάποιες από τις υπόλοιπες στήλες: Empno, Projno, (empno, projno).

Τελικά προκύπτει η δεύτερη κανονική μορφή της εικόνας 6.29.

Employ

Empno	Fname	Surname	JobNo	Jobname	DeptNo	Dname	dloc	Sal	Comm
10	ΣΠΥΡΟΣ	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	ΒΙΛΙΑ	2200	250
20	ΝΙΚΟΣ	DATE	200	ΑΝΑΛΥΤΗΣ	60	ΠΡΟΣΩΠΙΚΟ	ΘΗΒΑ	2000	100
30	ΝΙΚΟΣ	ΜΑΡΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΛΟΓΙΣΤΗΡΙΟ	ΘΗΒΑ	1000	
40	ΜΑΡΚΟΣ	ΜΑΡΚΟΥ	100	ΠΩΛΗΤΗΣ	80	MARKETING	ΘΗΒΑ	2200	500

Κύριο κλειδί=(empno)

Proj

Projno	Pdescr	ploc
1000	PAYROLL	ΘΗΒΑ
2000	MEDIA	ΒΙΛΙΑ

Κύριο κλειδί=(projno)

Works

Empno	Projno	ptime
10	1000	60
10	2000	40
20	1000	100
30	2000	100
40	2000	100

Κύριο κλειδί=(empno, projno)

Εικόνα 6.29 Οι πίνακες της δεύτερης κανονικής μορφής

Βήμα 3: Ελέγχουμε σε κάθε πίνακα της δεύτερης κανονικής μορφής μήπως κάποια στήλη (ή κάποιος συνδυασμός στηλών) που δεν ανήκει στο κύριο κλειδί του πίνακα μπορεί να «καθορίζει» από μόνη της άλλες στήλες του πίνακα. «Χωρίζουμε» ανάλογα τον πίνακα αυτόν.

Επομένως, στο παράδειγμά μας, ο πίνακας employ χωρίζεται σε τρεις (3) πίνακες: emp, job, dept. Οι υπόλοιποι πίνακες της δεύτερης κανονικής μορφής παραμένουν όπως είναι και στην Τρίτη κανονική μορφή (Εικόνα 6.30).

Emp

Empno	Fname	Surname	JobNo	DeptNo	Comm
10	ΣΠΥΡΟΣ	ΣΠΥΡΟΥ	100	50	250
20	ΝΙΚΟΣ	DATE	200	60	100
30	ΝΙΚΟΣ	ΜΑΡΚΟΥ	300	70	
40	ΜΑΡΚΟΣ	ΜΑΡΚΟΥ	100	80	500

Κύριο κλειδί=(empno)

Job

JobNo	Jobname	Sal
100	ΠΩΛΗΤΗΣ	2200
200	ΑΝΑΛΥΤΗΣ	2000
300	ΧΕΙΡΙΣΤΗΣ	1000

Κύριο κλειδί=(jobno)

Dept

DeptNo	Dname	dloc
50	ΠΩΛΗΣΕΙΣ	ΒΙΛΙΑ
60	ΠΡΟΣΩΠΙΚΟ	ΘΗΒΑ
70	ΛΟΓΙΣΤΗΡΙΟ	ΘΗΒΑ
50	MARKETING	ΘΗΒΑ

Κύριο κλειδί=(deptno)

Proj

Projno	Pdescr	ploc
1000	PAYROLL	ΘΗΒΑ
2000	MEDIA	ΒΙΛΙΑ

Κύριο κλειδί=(projno)

Works

Empno	Projno	ptime
10	1000	60
10	2000	40
20	1000	100
30	2000	100
40	2000	100

Κύριο κλειδί=(empno, projno)

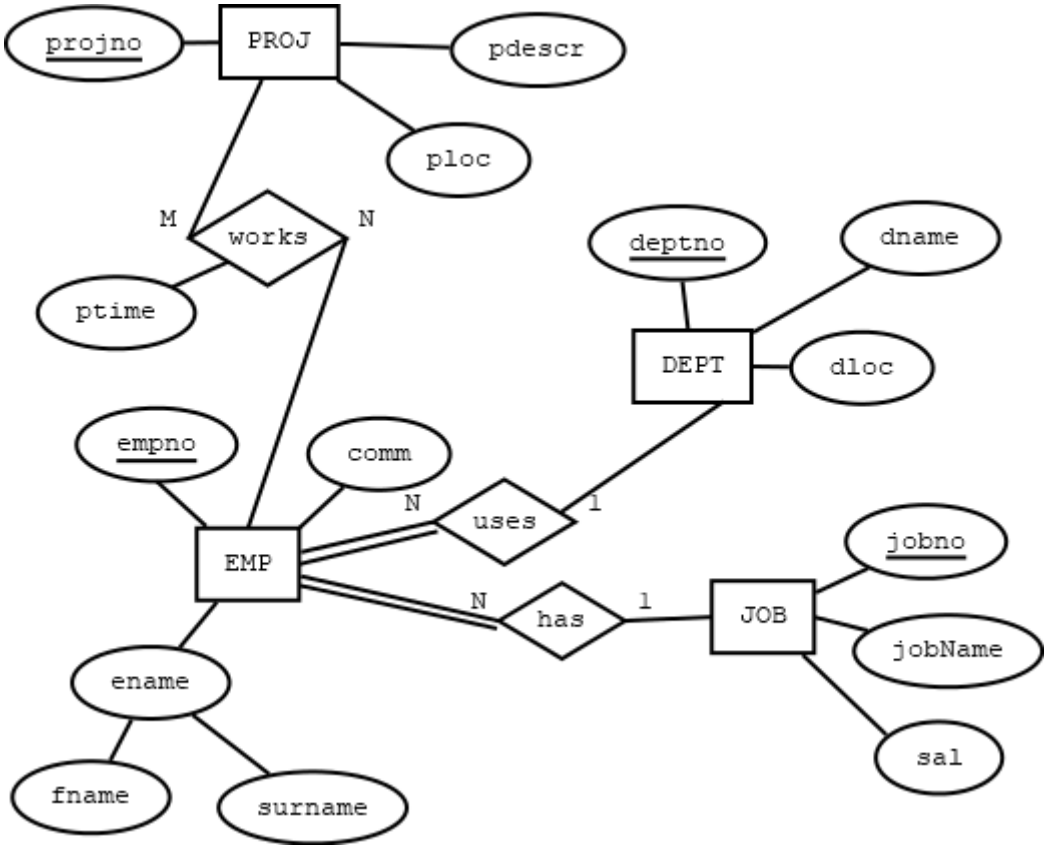
Εικόνα 6.30 Τρίτη κανονική μορφή

6.2.8 Μοντέλο οντοτήτων συσχετίσεων. Καθολική συμμετοχή οντότητας σε συσχέτιση.

Στην Εικόνα 6.31 βλέπουμε οντότητες, συσχετίσεις και τα χαρακτηριστικά του μοντέλου οντοτήτων συσχετίσεων. Η διπλή γραμμή που συνδέει την οντότητα EMP με τη συσχέτιση has(DEPT,EMP) σημαίνει ότι η οντότητα EMP μετέχει καθολικά στη συσχέτιση has. Δηλαδή, κάθε υπάλληλος έχει τοποθετηθεί υποχρεωτικά σε μία θέση (deptno). Η απλή γραμμή που συνδέει την οντότητα DEPT με τη συσχέτιση has σημαίνει ότι η οντότητα DEPT δεν μετέχει καθολικά στη συσχέτιση has. Δηλαδή, ένα τμήμα μπορεί να έχει ή να μην έχει υπάλληλους.

Οντότητες	Κλειδί	Άλλα χαρακτηριστικά
EMP	Empno	Surname, name, comm
DEPT	Deptno	Dname, dloc
JOB	Jobno	Jobname, sal
PROJ	Projno	Pdescr, ploc

Συσχετίσεις	Οντότητες	Τύπος	Χαρακτηριστικά
Works(EMP,PROJ)	EMP -- PROJ	M:N	Ptime
Has(DEPT,EMP)	DEPT -- EMP	1:N	
Has(JOB,EMP)	JOB -- EMP	1:N	



Εικόνα 6.31 Μοντέλο οντοτήτων – συσχετίσεων. Καθολική συμμετοχή οντότητας σε συσχέτιση.

6.2.9 Κανονικοποίηση βάσης δεδομένων τμήματος πληροφορικής (IT department)

Έστω ότι η ανάλυση δεδομένων της εταιρείας DATA CENTER οδήγησε στα παρακάτω στοιχεία:

Empno=κωδικός υπαλλήλου-προγραμματιστή, Fname=όνομα, Expr=συνολική προϋπηρεσία, Sal=μισθός υπαλλήλου, Lang_code=κωδικός γλώσσας προγραμματισμού που χρησιμοποιεί ο υπάλληλος, Language=γλώσσα προγραμματισμού, Use=χρόνια που χρησιμοποιεί τη γλώσσα προγραμματισμού ο υπάλληλος, P_code=κωδικός έργου στο οποίο εργάζεται ο υπάλληλος, Pname=ονομασία έργο, Budget=προϋπολογισμός έργου, Ptime=ποσοστό χρόνου που ο υπάλληλος απασχολείται σε ένα έργο.

Περιορισμοί

Ένας υπάλληλος έχει έναν μοναδικό κωδικό υπαλλήλου (Empno), όνομα (ename), προϋπηρεσία (expr), μισθό (sal). Ένας υπάλληλος γνωρίζει και χρησιμοποιεί τουλάχιστον μία γλώσσα προγραμματισμού. Μία γλώσσα προγραμματισμού μπορεί να τη χρησιμοποιούν πολλοί προγραμματιστές. Σε κάθε έργο της εταιρείας απασχολούνται πολλοί υπάλληλοι. Κάθε υπάλληλος μπορεί να μοιράζει τον χρόνο του σε πολλά έργα.

Βήμα 1: Τοποθετούμε όλα τα χαρακτηριστικά σε έναν πίνακα

1NF

ENAME	LANGUAGE	LANG_CODE	USE	EMP NO	EXPR	SAL	P_CODE	PNAME	BUDGET	P_TIME
CODD	C	100	3	100	4	3200	P_01	PAYROLL	100000	20
CODD	C++	200	2	100	4	3200	P_01	PAYROLL	100000	20
CODD	C	100	3	100	4	3200	W_02	WAREHOUSE	100000	80
CODD	C++	200	2	100	4	3200	W_02	WAREHOUSE	100000	80
DATE	C	100	4	500	5	4500	W_02	WAREHOUSE	200000	90
DATE	JAVA	300	1	500	5	4500	W_02	WAREHOUSE	200000	90
DATE	PYTHON	400	1	500	5	4500	P_01	PAYROLL	100000	10

σύνθετο κύριο κλειδί = (EMPNO, LANG_CODE, P_CODE)

Εικόνα 6.32 Πρώτη κανονική μορφή βάσης δεδομένων προσωπικού

Βήμα 2: Κοιτάμε ποιες στήλες «καθορίζει» από μόνο του κάθε τμήμα του κύριου κλειδιού της πρώτης κανονικής μορφής. «Χωρίζουμε» ανάλογα σε πίνακες τον πίνακα της πρώτης κανονικής μορφής**.** Εξετάζουμε τους παρακάτω συνδυασμούς:

Empno, Lang_code, P_code, (empno, lang_code), (empno, p_code), (lang_code, p_code),

(empno, lang_code, p_code).

Η δεύτερη κανονική μορφή συμπίπτει με την Τρίτη. Συμβολικά γράφεται 2NF ≡ 3NF Στην Εικόνα 6.33 γράφουμε την Τρίτη κανονική μορφή.

Ο πίνακας **Proj_Lang** έχει νόημα να υπάρχει στη βάση δεδομένων αν θέλουμε να γνωρίζουμε ποιές γλώσσες προγραμματισμού χρησιμοποιούνται σε κάθε έργο και το αντίστροφο.

Ο πίνακας **Emp_Proj_Lang** έχει νόημα να υπάρχει στη βάση δεδομένων αν θέλουμε να γνωρίζουμε για κάθε υπάλληλο ποιές γλώσσες προγραμματισμού χρησιμοποιεί σε κάθε έργο στο οποίο εργάζεται.

Προσοχή!

Από τα δεδομένα των πινάκων emp_lang, emp_proj, proj_lang δεν προκύπτει η πληροφορία αυτή. Στην πραγματικότητα, ο πίνακας **Emp_Proj_Lang** προκύπτει από μία συσχέτιση τύπου M:N:N

Languages (PK=lang_code)

LANG_CODE	LANGUAGE
100	C
200	C++
300	JAVA
400	PYTHON

Projects (PK=p_code)

P_CODE	PNAME	BUDGET
P_01	PAYROLL	100000
W_02	WAREHOUSE	200000

Employees (PK=empno)

EMPNO	ENAME	EXPR	SAL
100	CODD	4	3200
500	DATE	5	4500

Emp_Lang (PK=(empno, lang_code))

EMPNO	LANG_CODE	USE
100	100	3
100	200	2
500	100	4
500	300	1
500	400	1

Emp_Proj (PK=(empno, p_code))

EMPNO	P_CODE	P_TIME
100	P_01	20
100	W_02	80
500	P_01	10
500	W_02	90

Proj_Lang (PK=(lang_code, p_code))

LANG_CODE	P_CODE
100	P_01
200	P_01
100	W_02
200	W_02

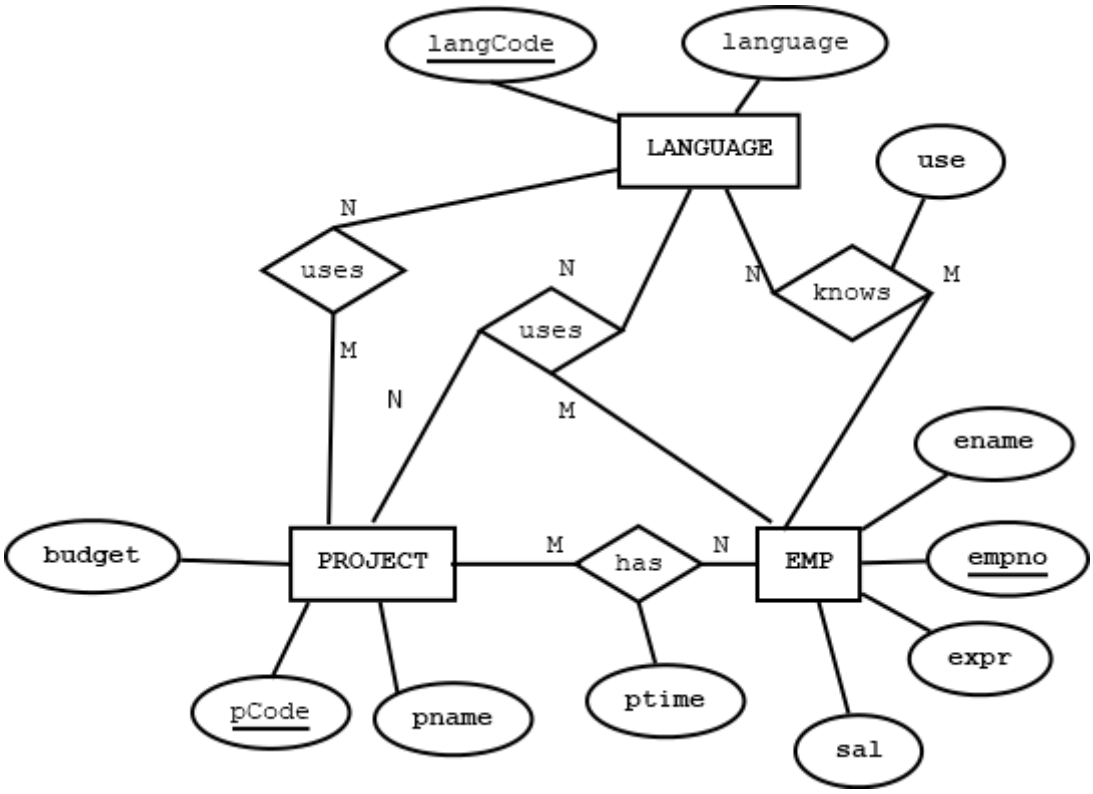
300	W_02
400	P_01

Emp_Proj_Lang (PK=(empno, lang_code, p_code))

EMPNO	LANG_CODE	P_CODE
100	100	P_01
100	200	P_01
100	100	W_02
100	200	W_02
500	100	W_02
500	300	W_02
500	400	P_01

Εικόνα 6.33 Τρίτη κανονική μορφή

Στην Εικόνα 6.34 βλέπουμε το αντίστοιχο μοντέλο της τρίτης κανονικής μορφής της εικόνας 6.33.



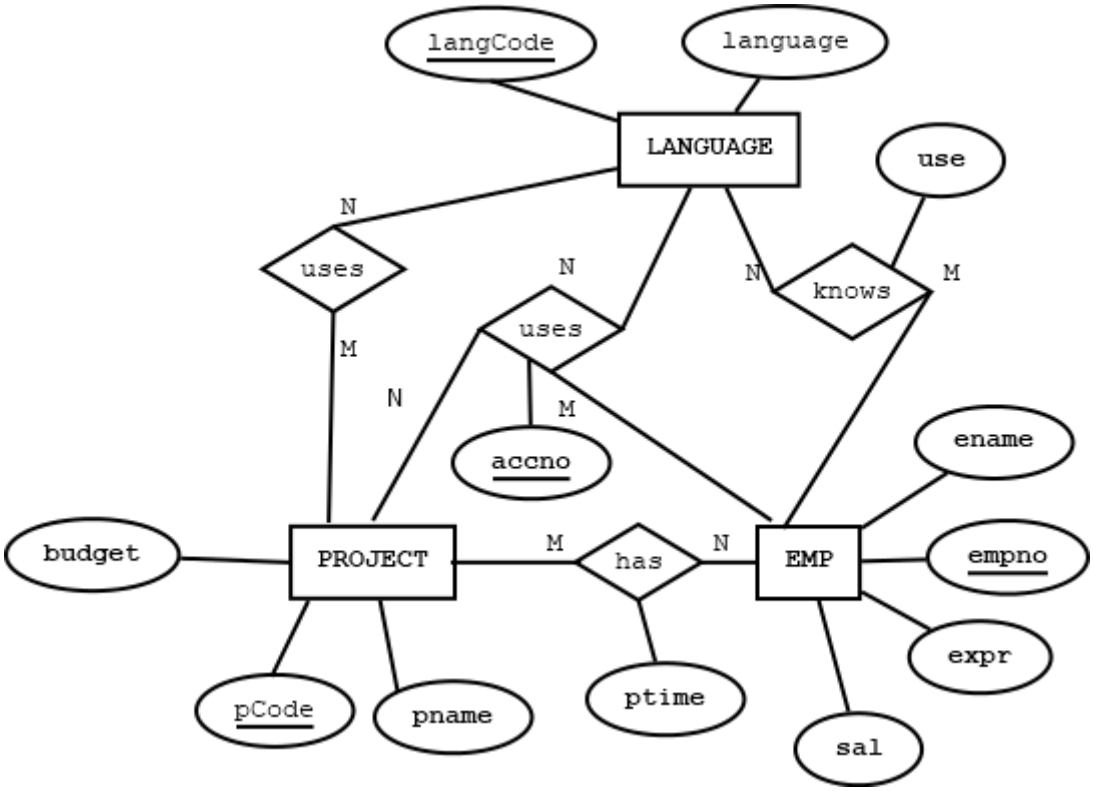
Εικόνα 6.34 Μοντέλο οντοτήτων συσχετίσεων αντίστοιχο της τρίτης κανονικής μορφής της εικόνας 6.31.

Στην Εικόνα 6.35 βλέπουμε την εναλλακτική σχεδίαση του τελευταίου πίνακα της εικόνας 6.32 και την αντίστοιχη εναλλακτική σχεδίαση του μοντέλου οντοτήτων συσχετίσεων.

Emp_Proj_Lang (PK=accno)

EMPNO	LANG_CODE	P_CODE	ACCNO
100	100	P_01	1
100	200	P_01	2

100	100	W_02	3
100	200	W_02	4
500	100	W_02	5
500	300	W_02	6
500	400	P_01	7



Εικόνα 6.35 Εναλλακτική σχεδίαση μοντέλου οντοτήτων συσχετίσεων της εικόνας 6.32 και πίνακας Emp_Proj_Lang με κύριο κλειδί accno.

6.2.10 Κανονικοποίηση βάσης δεδομένων συστήματος διαχείρισης παραγγελιών (orders)

Στην Εικόνα 6.36 παραθέτουμε την παγκόσμια σχέση ή με διαφορετική ορολογία έναν πίνακα με όλα τα χαρακτηριστικά της βάσης δεδομένων

Order_no	C_code	C_name	Address	O_date	Total	Item_no	I_name	List_price	qty	ptotal
123	1235	ΠΑΠΑΔΟΠΟΥΛΟΣ	ΘΗΣΕΩΣ 130	22/1/2018	900	101	BOLT	2	200	400
123	1235	ΠΑΠΑΔΟΠΟΥΛΟΣ	ΘΗΣΕΩΣ 130	22/1/2018	900	201	SCREW DRIVER	5	100	500

Εικόνα 6.36 Παγκόσμια σχέση ή όλα τα χαρακτηριστικά σε έναν πίνακα.

Ο πίνακας είναι σε πρώτη κανονική μορφή και το κύριο κλειδί είναι (order_no, item_no).

Στην Εικόνα 6.37 παραθέτουμε ηλεκτρονική φόρμα παραγγελίας. Σε παρενθέσεις γράφονται οι στήλες της βάσης δεδομένων.

Κωδικός παραγγελίας (order_no) □□□□□ **ημερομηνία** (o_date) □□/□□/□□□□

Στοιχεία πελάτη

Κωδικός (c_code) □□□□□

Επωνυμία (c_name) □□□□□□□□□□□□□□□□□□□□

Διεύθυνση (address) □□□□□□□□□□□□□□□□□□□□

Γραμμές παραγγελίας

Α/Α	Κωδικός είδους (Item_no)	είδος (item)	τιμή καταλόγου (list_price)	ποσότητα (qty)	μερικό σύνολο (ptotal)
□□	□□□	□□□□□□□□□□□□□□	□□□	□□□□	
□□	□□□	□□□□□□□□□□□□□□	□□□	□□□□	
□□	□□□	□□□□□□□□□□□□□□	□□□	□□□□	
□□	□□□	□□□□□□□□□□□□□□	□□□	□□□□	

Σύνολο (total) □□□□□

Εικόνα 6.37 Ηλεκτρονική φόρμα παραγγελίας

Στην Εικόνα 6.38 παραθέτουμε την ηλεκτρονική φόρμα παραγγελίας συμπληρωμένη με παράδειγμα.

Κωδικός παραγγελίας (order_no) 123 **ημερομηνία** (o_date) 22/01/2018

Στοιχεία πελάτη

Κωδικός (c_code) 1235

Επωνυμία (c_name) ΠΑΠΑΔΟΠΟΥΛΟΣ

Διεύθυνση (address) ΘΗΣΕΩΣ 130

Γραμμές παραγγελίας

Α/Α	Κωδικός είδους Item_no	είδος item	τιμή καταλόγου list_price	ποσότητα qty	μερικό σύνολο ptotal
1.	101	BOLT	2	200	400
2.	201	SCREW DRIVER	5	100	500

Σύνολο (total) 900

Εικόνα 6.38 Παράδειγμα συμπλήρωσης της ηλεκτρονικής φόρμας παραγγελίας

Θα μπορούσαμε να ανασχεδιάσουμε την ίδια φόρμα εξετάζοντας τα δεδομένα τα οποία εισάγονται σε αυτήν. Ακολουθεί, στην Εικόνα 6.39, η ίδια ηλεκτρονική φόρμα παραγγελίας σχεδιασμένη με δύο μπλοκς: ΜΠΛΟΚ βασικών στοιχείων παραγγελίας και ΜΠΛΟΚ γραμμών παραγγελίας. Οι φόρμες αυτού του τύπου είναι γνωστές και ως HEADING–DETAILS FORMS (ή MASTER-SLAVE ή MASTER-DETAILS).

(ΜΠΛΟΚ βασικών στοιχείων παραγγελίας)

Κωδικός παραγγελίας (order_no) □□□□□ **ημερομηνία** (o_date) □□/□□/□□□□

Στοιχεία πελάτη

Κωδικός (c_code) □□□□□

Επωνυμία (c_name) □□□□□□□□□□□□□□□□□□□□

Διεύθυνση (address) □□□□□□□□□□□□□□□□□□□□

Σύνολο (total) □□□□□

(ΜΠΛΟΚ γραμμών παραγγελίας)

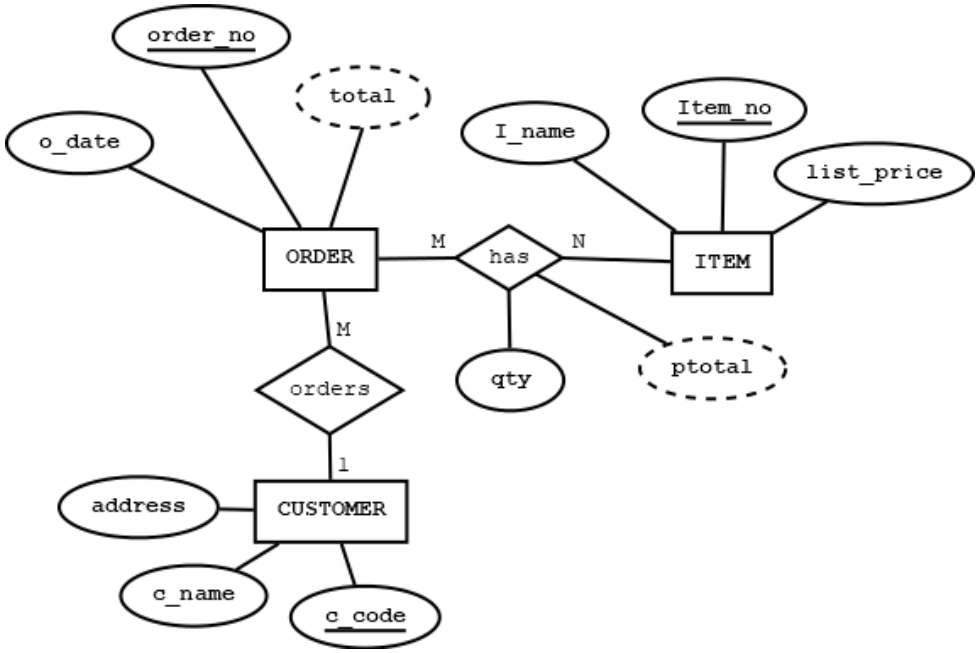
Γραμμές παραγγελίας

Invisible field (Κωδικός παραγγελίας) (order_no)

Α/Α	Κωδικός είδους	είδος	τιμή καταλόγου	ποσότητα	μερικό σύνολο
	Item_no	item	list_price	qty	ptotal
□□	□□□	□□□□□□□□□□	□□□	□□□	□□□□
□□	□□□	□□□□□□□□□□	□□□	□□□	□□□□
□□	□□□	□□□□□□□□□□	□□□	□□□	□□□□
□□	□□□	□□□□□□□□□□	□□□	□□□	□□□□

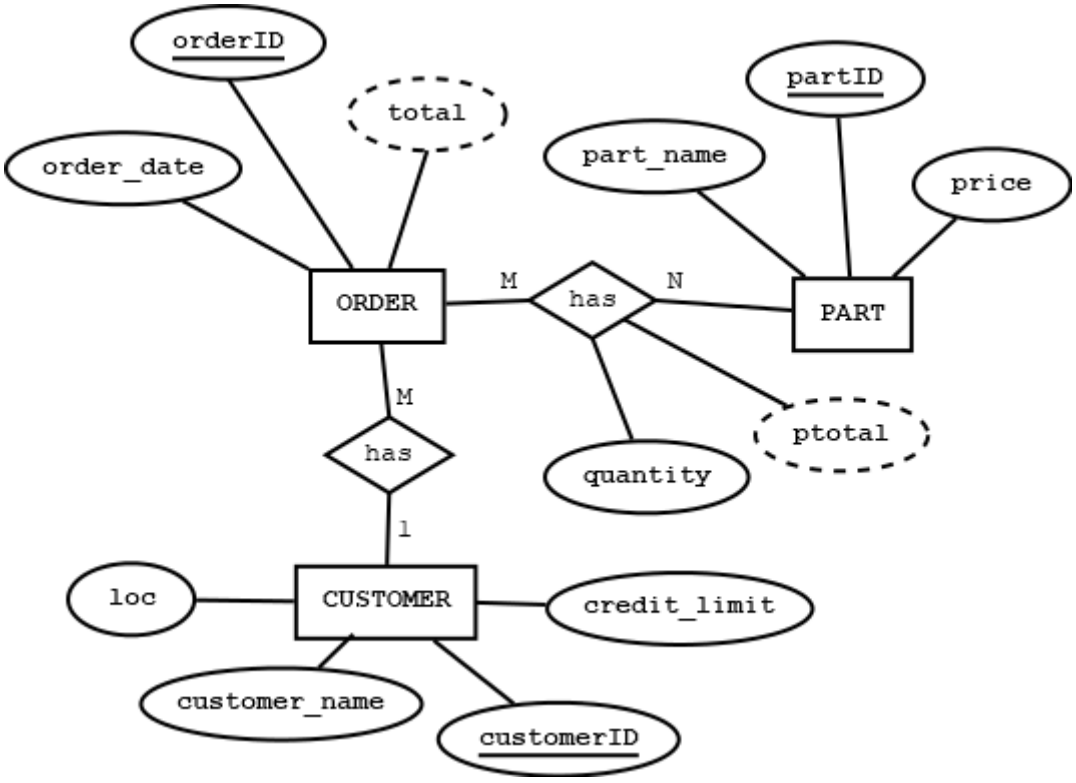
Εικόνα 6.39 Ηλεκτρονική φόρμα παραγγελίας με δύο μπλοκ, μπλοκ βασικών στοιχείων και μπλοκ γραμμών παραγγελίας

Ακολουθεί, στην Εικόνα 6.40, το μοντέλο οντοτήτων συσχετίσεων. Στο μοντέλο μας δεν απεικονίζεται κάποιο χαρακτηριστικό αριθμού γραμμής παραγγελίας. Στη συνέχεια αυτό θα αλλάξει.

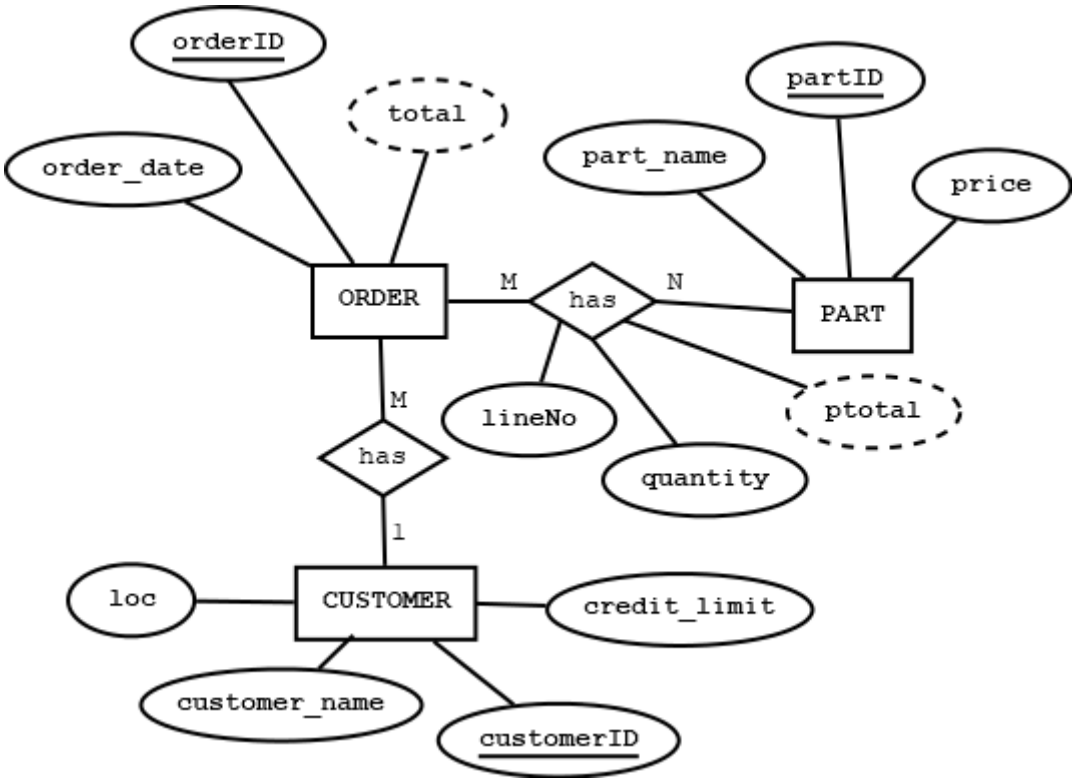


Εικόνα 6.40 ΜΟΣ του συστήματος παραγγελιών

Ακολουθεί εξειδίκευση με διάφορους τρόπους του ίδιου μοντέλου για παραγγελίες ανταλλακτικών (βλέπε εικόνες 6.41, 6.42).



Εικόνα 6.41 ΜΟΣ του συστήματος παραγγελιών ανταλλακτικών



Εικόνα 6.42 ΜΟΣ του συστήματος παραγγελιών ανταλλακτικών το οποίο περιλαμβάνει και το χαρακτηριστικό γραμμή παραγγελίας (Lineno)

Στην Εικόνα 6.43 βλέπουμε τους πίνακες του συστήματος παραγγελιών με δείγμα δεδομένων.

SELECT * FROM customers;

Customer_id	Customer_name	Loc	Credit_limit
100	CODD	NEW YORK	10000
200	DATE	DALLAS	20000
300	CHEN	CHICAGO	15000
400	ELMASRI	BOSTON	35000

SELECT * FROM orders;

Order_id	Order_date	total	Customer_id
10	1/1/2017	3000	100
11	1/1/2017	2900	100
12	2/5/2017	1200	100
15	2/5/2017	1200	100
20	7/7/2017	2000	200
21	7/7/2017	2000	100
22	7/7/2017	1200	200
23	7/7/2017	2000	200
30	4/8/2017	1800	100
77	7/12/2017	2000	NULL

SELECT * FROM parts;

part_id	Part_name	Price
12345	BOLT	1
56789	SCREWDRIVER	4

SELECT * FROM orderlines;

Order_id	Part_id	Quantity	Ptotal
10	12345	1800	1800
10	56789	300	1200

Εικόνα 6.43 Πίνακες του συστήματος παραγγελιών με δείγμα δεδομένων.

Ακολουθεί Μοντέλο οντοτήτων συσχετίσεων.

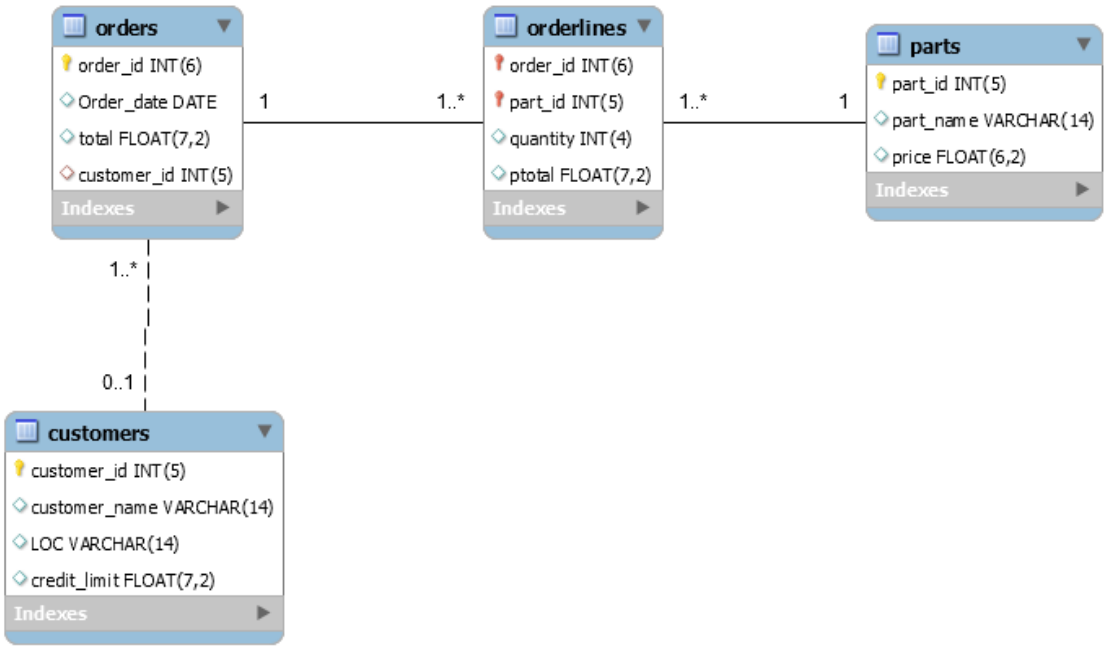
Στην Εικόνα 6.44 για το σχεδιασμό του ΜΟΣ του συστήματος παραγγελιών χρησιμοποιείται συμβολισμός UML.

Στην εικόνα 6.45 χρησιμοποιείται συμβολισμός CLASSIC.

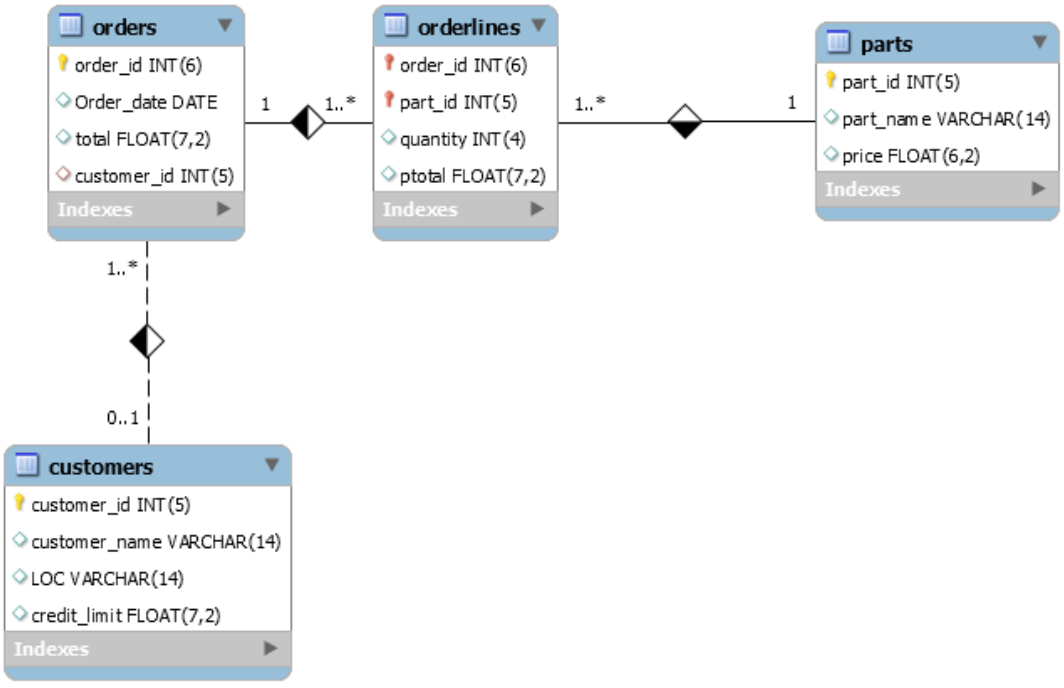
Στην εικόνα 6.46 χρησιμοποιείται συμβολισμός Crow’s foot.

Στην εικόνα 6.47 χρησιμοποιείται συμβολισμός IDEFIX.

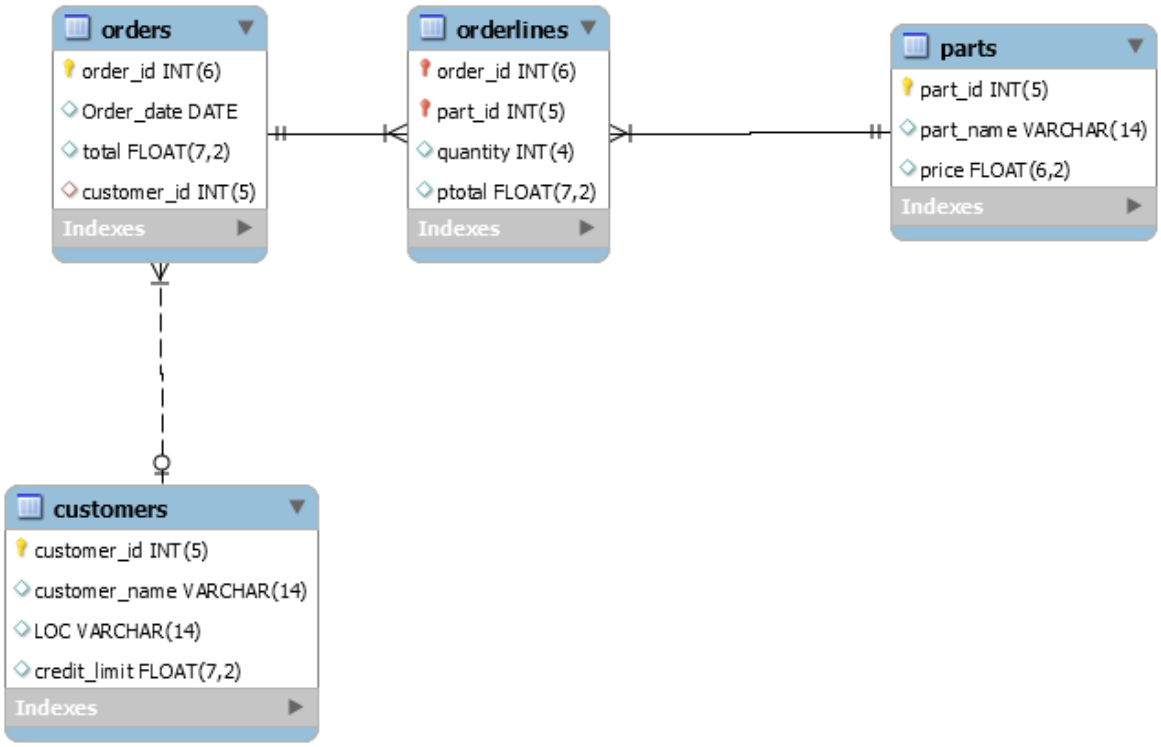
Στην εικόνα 6.48 χρησιμοποιείται συμβολισμός Connect-to-columns.



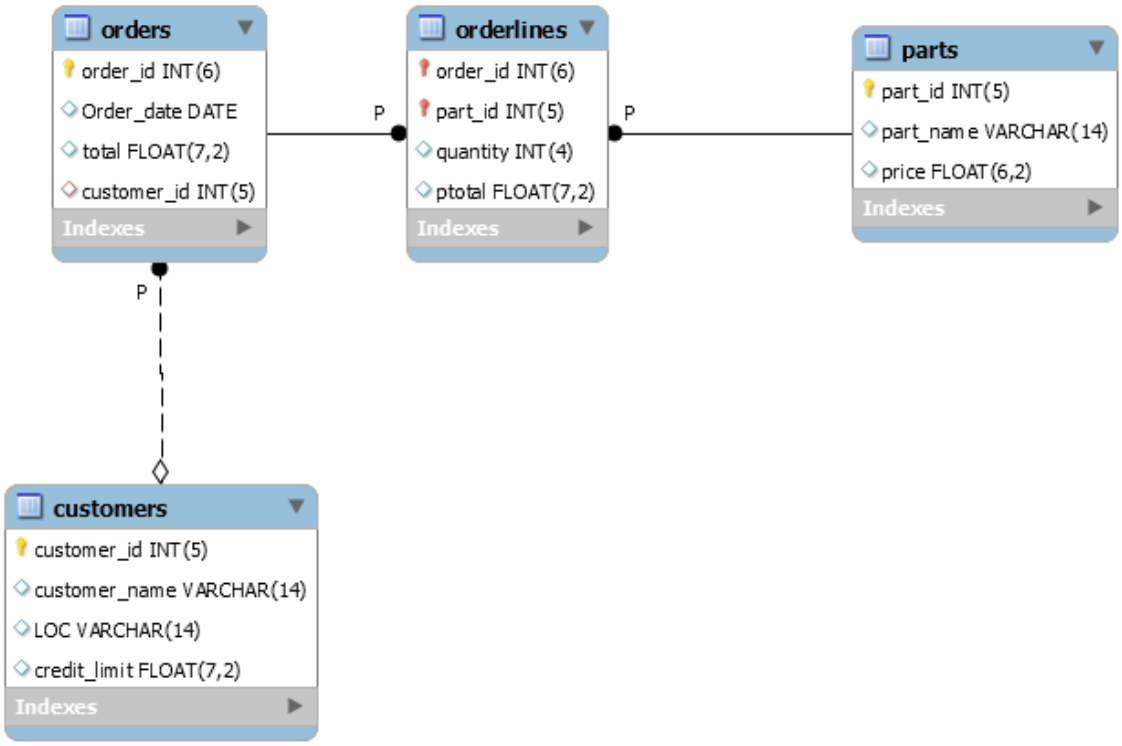
Εικόνα 6.44 Για το σχεδιασμό του ΜΟΣ του συστήματος παραγγελιών χρησιμοποιείται συμβολισμός UML.



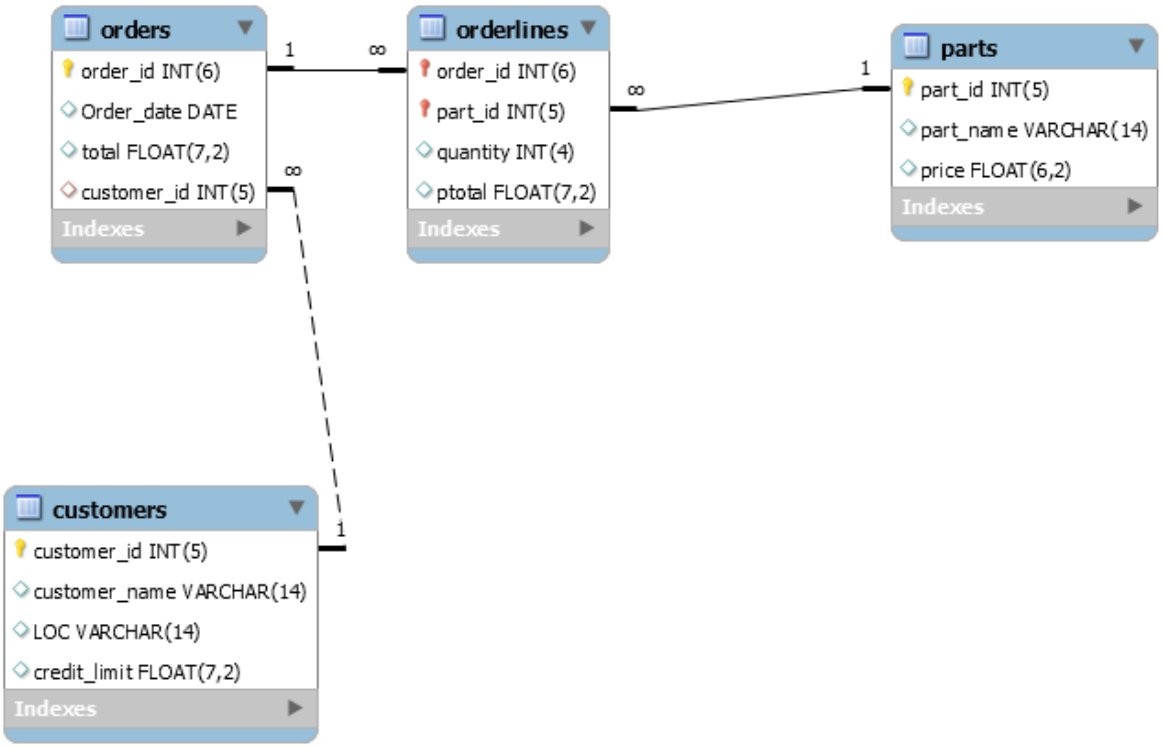
Εικόνα 6.45 Για το σχεδιασμό του ΜΟΣ του συστήματος παραγγελιών χρησιμοποιείται συμβολισμός μοντέλου Classic



Εικόνα 6.46 Για το σχεδιασμό του ΜΟΣ του συστήματος παραγγελιών χρησιμοποιείται συμβολισμός μοντέλου Crow's foot

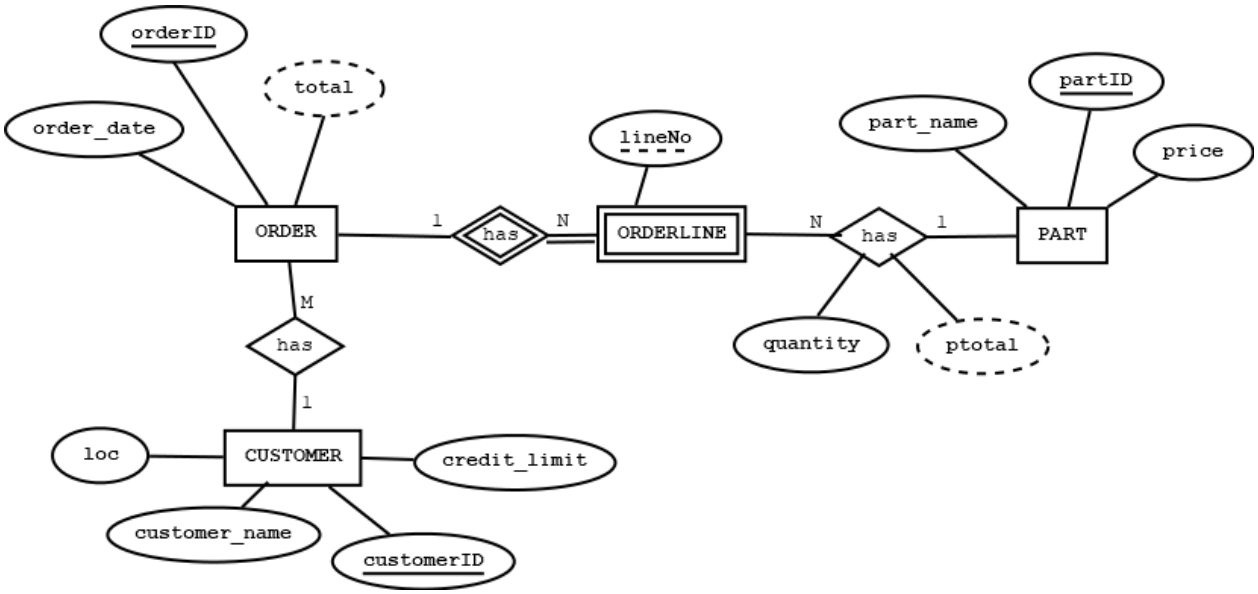


Εικόνα 6.47 Για το σχεδιασμό του ΜΟΣ του συστήματος παραγγελιών χρησιμοποιείται συμβολισμός μοντέλου IDEFIX



Εικόνα 6.48 Σχεδιασμός του ΜΟΣ του συστήματος παραγγελιών με χρήση συμβολισμού μοντέλου connect-to-columns

Στην εικόνα 6.49 βλέπουμε μία ενδιαφέρουσα παραλλαγή στη σχεδίαση του μοντέλου η οποία περιγράφει τις γραμμές της παραγγελίας ως εξαρτώμενες οντότητες.



Εικόνα 6.49 Σχεδιασμός του ΜΟΣ του συστήματος παραγγελιών με τις γραμμές της παραγγελίας ως εξαρτώμενες οντότητες από την οντότητα με τα βασικά στοιχεία της παραγγελίας

6.3 Μία εισαγωγή στα πληροφοριακά συστήματα. Θέματα ανάλυσης και σχεδιασμού συστήματος βάσης δεδομένων με χρήση τεχνολογίας πληροφοριακών συστημάτων

Στην ενότητα αυτή διατυπώνουμε τέσσερις ερωτήσεις και προσπαθούμε να απαντήσουμε στα πλαίσια της «κλασσικής» ανάλυσης συστημάτων:

1) Τι είναι ένα πληροφοριακό σύστημα (ΠΣ) και τι είναι ο κύκλος ζωής του;

Αν υποτεθεί ότι παίζουμε το ρόλο του αναλυτή συστημάτων (ενδεχομένως και του προγραμματιστή, ταυτόχρονα) και εργαζόμαστε για την ανάλυση και το σχεδιασμό ενός πληροφοριακού συστήματος (ΠΣ) ενός οργανισμού ή μιας επιχείρησης τότε είναι χρήσιμο να θεωρούμε ότι:

Το ΠΣ μπορεί να μοντελοποιηθεί σαν ένα σύστημα κοινωνικών συσχετίσεων μεταξύ ανθρώπων που πρέπει να επιτελέσουν/υποστηρίξουν τις λειτουργίες του οργανισμού ή της επιχείρησης χρησιμοποιώντας ή παρέχοντας, ανάλογα με το ρόλο τους, τα απαραίτητα μέσα. Το πλαίσιο των δραστηριοτήτων των ανθρώπων αυτών καθορίζεται από τους περιορισμούς που επιβάλει η οργανωτική δομή του οργανισμού.

Επομένως, στην κλασική ανάλυση το ΠΣ αντιμετωπίζεται σαν ένα κοινωνικό σύστημα αποτελούμενο από τέσσερις (4) αλληλοσχετιζόμενες συνιστώσες:

Α) Αρχιτεκτονική και Οργανωτική δομή

Συνήθως, εκφράζεται με οργανόγραμμα που αποτυπώνει τη συσχέτιση Διευθύνσεων, Τμημάτων κ.τ.λ. και συνοδεύεται από επεξηγηματικό κείμενο που περιγράφει λειτουργίες υπο-τμημάτων, προσωπικό κ.α. Ενσωματώνει τα παρακάτω:

- συσχετίσεις μεταξύ διευθύνσεων, τμημάτων κ.τ.λ.
- κανόνες, κανονισμούς και τιμολόγια σύμφωνα με τα οποία παρέχονται υπηρεσίες
- κατανομή εργασιών, συντονισμός υπο-τμημάτων κ.τ.λ.
- ρόλος ατόμων
- δομή επικοινωνιών που απαιτείται για το συντονισμό και την ολοκλήρωση των δραστηριοτήτων

Β) Λειτουργίες

Περιλαμβάνει όλες τις δραστηριότητες που απαιτούνται για την κάλυψη των αναγκών του οργανισμού/επιχείρησης και των αναγκών των πελατών του

Γ) Μέσα

Όταν αναφερόμαστε σε μέσα, ιδιαίτερη έμφαση δίδεται στα στοιχεία (data) και στις πληροφορίες που ενδιαφέρουν τον οργανισμό/επιχείρηση. Η πληροφορία αποτελεί αντικείμενο μελέτης σε όλες τις συνιστώσες του ΠΣ. Σύμφωνα με πολλές γνωστές μεθοδολογίες, ενδιαφέρει ιδιαίτερα η αποτύπωση της ροής της πληροφορίας στον οργανισμό και η μοντελοποίηση δεδομένων. Ενδιαφέρουν, ακόμη, ο εξοπλισμός, το λογισμικό, τα μέσα επικοινωνίας κ.τ.λ.

Το κεντρικό λογισμικό είναι το Σύστημα Διαχείρισης Βάσεων Δεδομένων και οι συνοδευτικές εφαρμογές λογισμικού.

Δ) Προσωπικό

Συμπεριλαμβάνει αριθμό και είδος προσωπικού και απαιτούμενες δεξιότητες, επίπεδο εκπαίδευσης, περιοχή ή περιοχές εξειδίκευσης κ.λπ.

2) Ποιες είναι οι συνηθέστερες (κύριες) δραστηριότητες του αναλυτή συστημάτων;

Κατά την άποψή μας δύο είναι οι κύριες εργασίες:

- Αποτύπωση του υπάρχοντος συστήματος
- Προτάσεις ή και αποτύπωση για το νέο (μελλοντικό) σύστημα

Επιπλέον, όπως είναι προφανές, ενδιαφέρει ιδιαίτερα ο προσεκτικός σχεδιασμός και η πραγματοποίηση της μετάπτωσης από το υπάρχον στο νέο σύστημα.

3) Ποια τα κύρια εργαλεία του αναλυτή;

Η δομημένη συνέντευξη, τα ερωτηματολόγια, η συλλογή και ανάλυση υλικού (εντύπων, εκτυπώσεων κ.τ.λ.), οι διαγραμματικές τεχνικές, π.χ. μεθοδολογίες Data Flow Diagram, Entity-Relationship Model, Jackson. Αυτά τα εργαλεία, συχνά, συνδέονται με συγκεκριμένες μεθοδολογίες ανάλυσης, σχεδίασης και υλοποίησης συστημάτων, π.χ., UML, που μπορεί να χρησιμοποιεί/υιοθετεί ο χώρος εργασίας σας. Θα δούμε παρακάτω συγκεκριμένα παραδείγματα διαγραμματικών τεχνικών, π.χ., τα διαγράμματα χρήσης δεδομένων (Data Usage Diagrams) της μεθοδολογίας Jackson το οποίο μπορεί να χρησιμοποιηθεί και για την ανάλυση των στοιχείων (δεδομένων) ενός εντύπου.

4) Ποια τα «τυπικά» στάδια της ανάλυσης και του σχεδιασμού του ΠΣ;

Υπάρχει μία μεγάλη συζήτηση και βιβλιογραφία γύρω από τις μεθοδολογίες και τα εργαλεία στο θέμα αυτό. Συνήθως, κάθε πρόταση μεθοδολογίας αναφέρεται στον κύκλο ζωής του πληροφοριακού συστήματος, προσπαθεί να «τοποθετήσει» την ανάλυση απαιτήσεων αναφορικά με τον κύκλο ζωής του συστήματος, προδιαγράφει τις εργασίες της, καθορίζει τις επόμενες φάσεις κ.τ.λ. Έμφαση δίδεται, συνήθως, στη μελέτη σκοπιμότητας, στην επιλογή και αξιολόγηση εξοπλισμού και στην μετάπτωση στο νέο σύστημα.

Πηγές περαιτέρω μελέτης

Υπάρχουν πολλά αξιόλογα ανοικτά συγγράμματα των εκδόσεων Κάλλιπος τα οποία καλύπτουν πάρα πολλές και ενδιαφέρουσες πτυχές των πληροφοριακών συστημάτων:

Μητάκος, Θ. (2015) Πληροφοριακά Συστήματα Διοίκησης. Μελέτη, ανάλυση και διαχείριση, Κάλλιπος

Μητρόπουλος, Σ., Δουληγέρης, Χ. (2015) Πληροφοριακά Συστήματα στο Διαδίκτυο. Εφαρμογές, Ανάπτυξη, Υποδομές, Κάλλιπος

Φιτσιλής, Π. (2015) Σύγχρονα Πληροφοριακά Συστήματα Επιχειρήσεων. ERP-CRM-BPR, Κάλλιπος

Αν ενδιαφέρεστε για μία επισκόπηση της βιβλιογραφίας στο θέμα των ΠΣ και της κλασσικής ανάλυσης συστημάτων μπορείτε να ανατρέξετε σε πολλά ξενόγλωσσα και ελληνικά εγχειρίδια. Ενδεικτικά αναφέρω, το συνοπτικό και καλογραμμένο:

Avizon and Fitzgerald (2006) Information systems development: Methodologies, Techniques and Tools, McGraw-Hill, 2006

Αν ενδιαφέρεστε για μία ολοκληρωμένη, γενικότερη, θεώρηση, που να περιλαμβάνει και την Ανάλυση Συστημάτων, μπορείτε να μελετήσετε και να αξιοποιήσετε ιστοτόπους (sites) που αναφέρονται στο θέμα αυτό, π.χ., το δικτυακό τόπο http://www.rspa.com, R.S. Pressman & Associates, Inc.) ή το γνωστό σύγγραμμα:

Roger Pressman, Software Engineering – A practitioner's approach, McGraw-Hill, 7th Ed.

Τέλος, υπάρχουν και γνωστά ελληνικά συγγράμματα π.χ.,

Ε. Κιουντούζη, Μεθοδολογίες Ανάλυσης και σχεδιασμού Πληροφοριακών Συστημάτων, Εκδόσεις Μπένου.

6.3.1 Το πρόβλημα της σύγκρισης και της επιλογής της μεθοδολογίας σχεδιασμού και ανάπτυξης πληροφοριακού συστήματος

Υπάρχουν πάρα πολλές προσεγγίσεις (μεθοδολογίες, τεχνικές, εργαλεία κτλ.) για την ανάλυση, το σχεδιασμό και την ανάπτυξη ενός ΠΣ. Στη βιβλιογραφία υπάρχει ο όρος "ζούγκλα μεθοδολογιών ανάπτυξης ΠΣ" (π.χ., Avizon – Fitzgerald, 2006) επειδή οι μεθοδολογίες είναι χιλιάδες. Σχεδόν όλες οι μεθοδολογίες περιλαμβάνουν προτάσεις για τον κύκλο ανάπτυξης του εγχειρήματος της αναδιοργάνωσης μιας υπηρεσίας ή του πληροφοριακού της συστήματος ή του απαραίτητου συστήματος λογισμικού οργανισμού/επιχείρησης

Η μέχρι σήμερα πρακτική είναι να συγκροτούνται μεθοδολογίες υπό τη μορφή μίας θεωρίας ανάπτυξης πληροφοριακών συστημάτων και συλλογών τεχνικών, εργαλείων και περιγραφικών γλωσσών επιμέρους όψεων του συστήματος. Οι μεθοδολογίες έχουν ένα συγκεκριμένο προσανατολισμό σε ότι αφορά τους στόχους τους:

- Προσπαθούν να καλύψουν συνολικά την οργάνωση και λειτουργία της επιχείρησης και του πληροφοριακού συστήματος, παρέχουν Enterprise Modelling tools κ.τ.λ.
- Προσφέρουν ένα πλαίσιο μοντελοποίησης δεδομένων (data modelling).
- Καλύπτουν την ανάλυση απαιτήσεων και την αρχιτεκτονική του συστήματος λογισμικού.

Ευρέως χρησιμοποιούνται εδώ και πολλά χρόνια οι δομημένες τεχνικές ανάλυσης και σχεδιασμού πληροφοριακών συστημάτων (structured analysis and design techniques), οι αντικειμενοστρεφείς μεθοδολογίες κ.λπ. Γνωστές μεθοδολογίες είναι: Information Engineering (IE) του Martin, Yourdon Systems Method, Jackson Systems Development, Structured Systems Analysis and Design Methodology (SSADM), και πολλές άλλες. Σημαντικός είναι, στις μέρες μας, ο ρόλος της μεθοδολογίας Unified Modeling Language (UML), η οποία αποτελεί μία τυπική γλώσσα βιομηχανικό πρότυπο (industry-standard language) για τις εφαρμογές λογισμικού συστημάτων. Η μεθοδολογία UML διασφαλίζει τον προσδιορισμό, την απεικόνιση, την κατασκευή και την τεκμηρίωση των τεχνουργημάτων των συστημάτων λογισμικού ("the artifacts of software systems").

Το χαρακτηριστικό των περισσοτέρων μεθοδολογιών είναι ότι είναι γενικές και ότι απευθύνονται σε ειδικούς προϋποθέτοντας ένα σχετικά υψηλό βαθμό ωρίμανσης (maturity) του οργανισμού/της επιχείρησης.

Είναι αρκετά συνηθισμένο στην πράξη το γεγονός ότι υπάρχουν περιπτώσεις που οι μεθοδολογίες αυτές προκειμένου να αξιοποιηθούν αποτελεσματικά χρειάζονται είτε προσαρμογές ή απλοποιήσεις. Κυρίως αυτό συναντάται όταν έχουμε:

- Εξειδικευμένα συστήματα ή υποσυστήματα πληροφοριακών συστημάτων.
- Δοκιμασμένες αρχιτεκτονικές ΠΣ ή τυποποιημένες διαδικασίες από εξωτερικούς παράγοντες.
- Αδυναμία, λόγω περιορισμένης ωριμότητας του οργανισμού, αποτελεσματικής αξιοποίησης μεθόδων.

Ακολουθεί μία διατύπωση του προβλήματος της υιοθέτησης και της προσαρμογής μεθοδολογίας για την ανάπτυξη πληροφοριακών συστημάτων:

Το πρόβλημα της υλοποίησης και σχεδίασης εξειδικευμένων πληροφοριακών συστημάτων ή υποσυστημάτων σε χαμηλής, μεσαίας ή ακόμα και υψηλής, ωριμότητας οργανισμούς σε συνδυασμό με την απαίτηση για

χαμηλό κόστος, υψηλή ταχύτητα εφαρμογής με περιορισμένους ανθρώπινους και υλικούς πόρους ανάγεται στην απλοποίηση των μεθοδολογικών προσεγγίσεων και των εργαλείων και στην επιλογή των κατάλληλων.

6.3.2 Ένα «πρακτικό» πλαίσιο μεθοδολογίας ανάλυσης, σχεδιασμού & υλοποίησης πληροφοριακών συστημάτων με χρήση βάσεων δεδομένων

Το Πληροφοριακό σύστημα (Π.Σ.) μπορεί να αντιμετωπίζεται (system view) σαν ένα σύνολο που πρέπει να μελετηθεί σε κατευθύνσεις-άξονες, όπως:

1) οργανωτική δομή (οργανόγραμμα) και αρχιτεκτονική συστήματος
2) πιθανοί χρήστες - ομάδες χρηστών
3) λειτουργίες του πς και παρεχόμενες υπηρεσίες
4) μέσα (resources) απαιτούμενα
5) συμμετοχή σε σχήματα συνεργασίας
6) προσωπικό - ρόλοι εμπλεκομένων προσώπων
7) πρότυπα υιοθετούμενα / χρησιμοποιούμενα στις λειτουργίες/ υπηρεσίες
8) ωριμότητα (maturity) τεχνολογίας και χρηστών
9) οικονομικά στοιχεία
10) προβολή του έργου και των υπηρεσιών (exploitation)
11) τυποποίηση (και φιλικότητα) της διεπαφής (interface) του χρήστη
12) μετάπτωση στο νέο σύστημα.

Παραθέτουμε κύριες δραστηριότητες και παραδοτέα της ανάλυσης των πληροφοριακών συστημάτων.

6.3.2.1 Καθορισμός Πιθανών χρηστών (και ομάδων χρηστών) του ΠΣ

Αποτελεί κύριο τμήμα ή και αντικείμενο ξεχωριστής μελέτης σε κάθε περίπτωση η έρευνα αγοράς και ο καθορισμός των πιθανών χρηστών του ΠΣ.

Συνοπτικά:

- Ένα ΠΣ σύστημα βασίζεται στους χρήστες του και υπάρχει για αυτούς.
- Σε αρκετές περιπτώσεις τα ΠΣ ή πολλά υποσυστήματά τους, οικοδομούνται χωρίς να υπάρχει πλήρως αποτυπωμένη η σχετική ανάγκη.
- Σε αρκετές περιπτώσεις δεν υπάρχουν ικανοποιητικά ποσοτικά στοιχεία από τη χρήση του ΠΣ
- Σε αρκετές περιπτώσεις δεν έχει προηγηθεί έρευνα αγοράς που να περιλαμβάνει πιθανούς χρήστες.

6.3.2.2 Καταγραφή της υπάρχουσας κατάστασης (ή αποτύπωση του ΠΣ)

Γίνεται με τη μέθοδο δομημένων ή ελεύθερων συνεντεύξεων (συζητήσεων που περιλαμβάνουν διατύπωση κρίσιμων ερωτήσεων) με ανθρώπους του φορέα του ΠΣ αλλά και άλλων φορέων και ειδικών. Συνοδεύεται από ερωτηματολόγια, συλλογή εντύπων, παρατήρηση λειτουργίας σε πραγματικές συνθήκες κ.τ.λ.

Για την έγκυρη καταγραφή της υπάρχουσας κατάστασης του ΠΣ αλλά και των μελλοντικών απαιτήσεων από τις λειτουργίες, τις υπηρεσίες κτλ. (αλλά και σε πολλές άλλες από τις περιπτώσεις του παραπάνω πλαισίου) είναι πολύ χρήσιμο να ακολουθείται μία απλή μεθοδολογία:

Να διατυπώνονται κύρια σενάρια (Use Cases) χρησιμοποίησης του συστήματος, περιγραφής της λειτουργίας του, περιγραφής των ρόλων των εμπλεκομένων χρηστών στο ΠΣ κτλ.

Για την διατύπωση κρίσιμων ερωτήσεων κατά τις συνεντεύξεις, αν εφαρμόζετε συγκεκριμένη μεθοδολογία υπάρχουν συγκεκριμένες υποδείξεις διαφορετικά ακολουθείται το πλαίσιο που καθορίζεται από γνωστές λίστες ελέγχου (checklists). Πολλές φορές κρίνεται σκόπιμο να γίνονται συνεντεύξεις για επιμέρους θέματα όχι μόνο με ανθρώπους του φορέα του ΠΣ αλλά και άλλων φορέων και ειδικών.

6.3.2.3 Προκαταρκτική μελέτη σκοπιμότητας (ή προμελέτη)

Συνήθως, γίνεται από τον ίδιο τον οργανισμό, έχει σύντομη διάρκεια (1/2 έως 1 ½ μήνα) και καθορίζει τις προδιαγραφές για τη μελέτη σκοπιμότητας που θα ακολουθήσει. Πιο συγκεκριμένα καθορίζει:

- Προϋπολογισμό μελέτης σκοπιμότητας
- Χρόνο εκπόνησης
- Ζητούμενα, εύρος έργου, πιθανές λύσεις που πρέπει να εξεταστούν κ.τ.λ.

6.3.2.4 Μελέτη σκοπιμότητας

Περιγράφει εναλλακτικές λύσεις (σενάρια) και επιλέγει και προτείνει την επικρατέστερη λύση παραθέτοντας επιχειρήματα σχετικά με:

- Οικονομική σκοπιμότητα (cost–benefit analysis)
- Τεχνική σκοπιμότητα

ενώ παράλληλα καλύπτει/διερευνά άλλες πλευρές εξαρτώμενες από το συγκεκριμένο έργο, π.χ., Νομικά θέματα, πιθανές εμπλοκές, διαχείριση απορρήτων πληροφοριών κ.τ.λ. Χρησιμοποιεί την προμελέτη και ως ένα βαθμό προδιαγράφει το νέο, προτεινόμενο σύστημα.

Συνήθως, υπάρχουν τρία τουλάχιστον σενάρια:

- «φτωχό», συντηρητικό
- ενδιάμεσο
- προχωρημένο

Αν οι συνθήκες το επιτρέπουν αποφεύγουμε το πρώτο σενάριο.

Περιεχόμενα μελέτης σκοπιμότητας

1) Εισαγωγή
    - Μια σύντομη διατύπωση του προβλήματος
    - το περιβάλλον υλοποίησης του συστήματος (τεχνολογικό πλαίσιο κ.τ.λ.)
    - περιορισμοί που επιδρούν στο έργο (project)
2) Σύνοψη και προτάσεις για Διοίκηση (management)
3) Σενάρια – Εναλλακτικές λύσεις
    - Παρουσίαση εναλλακτικών λύσεων
    - Κριτήρια επιλογής λύσης
    - Προτεινόμενη λύση

4) Περιγραφή συστήματος (σύμφωνα με το επικρατέστερο σενάριο / λύση)

   Συντομευμένη μορφή του υλικού που περιλαμβάνεται (ή θα περιληφθεί) στις Προδιαγραφές του νέου συστήματος (system specification)

5) Ανάλυση κόστους–οφέλους (cost–benefit analysis)

   Το πρόβλημα στην περίπτωση αυτή είναι ότι συνήθως το κόστος είναι χειροπιαστό, όχι όμως και το όφελος.

6) Υπολογισμός τεχνικού ρίσκου
7) Νομικά θέματα πα-εμπλοκές (legal ramifications)
8) Άλλα κρίσιμα σημεία (topics) εξαρτώμενα από το έργο

6.3.2.5 Εκπόνηση προδιαγραφών και πρόταση νέου συστήματος

Βασίζεται στη μελέτη σκοπιμότητας και προτείνει ένα από τα εκεί περιγραφόμενα σενάρια συμπεριλαμβάνοντας αναλυτικότερα τις προδιαγραφές. Είναι ενδιαφέρον να επισημανθεί ότι, συχνά, υπάρχει ένα είδος «ωρίμανσης» προδιαγραφών:

- Στην προμελέτη προτείνεται η αγορά / δημιουργία και λειτουργία βάσης δεδομένων
- Στη μελέτη σκοπιμότητας προτείνεται η αγορά εφαρμογής (προγραμμάτων) που είναι (proprietary) ή δεν είναι δεμένη με συγκεκριμένη πλατφόρμα εξοπλισμού (open).
- Στο στάδιο εκπόνησης αναλυτικών, λεπτομερών προδιαγραφών είναι καλό να επιλέγεται συγκεκριμένη εφαρμογή ή εφαρμογές που εξασφαλίζουν ανεξαρτησία από συγκεκριμένη πλατφόρμα.

6.3.2.6 Διακήρυξη, αξιολόγηση και συμβάσεις

Η διακήρυξη περιλαμβάνει όρους και προδιαγραφές (μειοδοτικού) διαγωνισμού για την προμήθεια πληροφοριακού συστήματος, εξοπλισμού, συστήματος λογισμικού κ.λπ. Η Ενιαία Ανεξάρτητη Αρχή Δημοσίων Συμβάσεων (ΕΑΑΔΗΣΥ) εκδίδει (αναρτά) πρότυπα τεύχη τα οποία και αποτελούν πλαίσιο για διακηρύξεις κ.λπ. του δημόσιου τομέα. Ο νόμος 4903/2022 (Α46/5-3-2022) με τίτλο "Πρότυπες προτάσεις για έργα υποδομής και λοιπές επείγουσες διατάξεις" περιλαμβάνει διατάξεις που αφορούν σε θέματα δημοσίων συμβάσεων. Ειδικότερα:

- τα άρθρα 1 - 14 ρυθμίζουν τις ΠΡΟΤΥΠΕΣ ΠΡΟΤΑΣΕΙΣ έργων υποδομών
- τα άρθρα 18 - 26 αφορούν λοιπές ρυθμίσεις δημοσίων συμβάσεων που τροποποιούν επιμέρους άρθρα των ν. 4412/2016 και 4782/2021.

6.3.3 Το κρίσιμο στοιχείο της Ωριμότητας (maturity) (τεχνολογίας αλλά και των χρηστών)

Με τον όρο ωριμότητα (maturity) αναφερόμαστε:

1) στην ωριμότητα της τεχνολογίας που χρησιμοποιείται στο ΠΣ
2) στην ωριμότητα του οργανισμού στη χρήση της τεχνολογίας αυτής, και
3) στην ωριμότητα των χρηστών (user groups) του ΠΣ.

Είναι απαραίτητη η σύγκριση του επιπέδου των χρηστών του ΠΣ με το επίπεδο των χρηστών διεθνώς. Σε κάθε περίπτωση, είναι απαραίτητο να εξεταστεί η παράμετρος της ωριμότητας της τεχνολογίας αλλά και των

ανθρώπων που θα κάνουν ανάπτυξη του συστήματος, αλλά και των χρηστών. Κατά συνέπεια η ωριμότητα θα είναι κύριο κριτήριο για την απάντηση σε πιθανές "τεχνικές" ερωτήσεις, όπως οι παρακάτω που έχουν επιλεγεί για να "φωτίσουν" την όλη τεχνολογική προσέγγιση μας:

- Ο οργανισμός είναι έτοιμος για μια αρχιτεκτονική ολοκληρωμένου ΠΣ που θα περιλαμβάνει Web και βάση;
- Υπάρχει δυνατότητα να γραφτούν εφαρμογές για τους χρήστες των υποσυστημάτων έτσι ώστε η πρόσβαση στο Κεντρικό σύστημα να γίνεται με βάση της αρχές της Java ή αντίστοιχης τεχνολογίας;
- Υπάρχει εμπειρία σε θέματα ασφάλειας ώστε να επιτρέψουμε στα επιμέρους υποσυστήματα να μπορούν να προσφέρουν συνδέσμους (links) σε κατάλληλες πληροφορίες μέσα στο ΠΣ αλλά και σε τρίτους ενδιαφερόμενους;

6.3.4 Τα εργαλεία που θα χρησιμοποιήσουμε. Σενάρια ή περιπτώσεις χρήσης (use cases)

Το σενάριο είναι ένα πολύ σημαντικό εργαλείο το οποίο μπορεί να μας βοηθήσει να κατανοήσουμε τις λειτουργίες του πληροφοριακού συστήματος, να ανιχνεύσουμε προβλήματα του υπάρχοντος πληροφοριακού συστήματος και ουσιαστικές λειτουργίες που απουσιάζουν και επιπλέον μας επιτρέπει να στοιχειοθετήσομε τις απαιτήσεις από το σύστημα στο μέλλον. Στη συνέχεια δίνουμε συγκεκριμένο παράδειγμα σεναρίου (Μητάκος, 2015). Η διαχείριση ακολουθεί τις βασικές αρχές της περίφημης μεθόδου του Jacobson. Ο ενδιαφερόμενος αναγνώστης μπορεί να ανατρέξει και στο κλασικό σύγγραμμα,

Ivar Jacobson et. al., Object–Oriented Software Engineering. A Use Case driven approach, Addison-Wesley

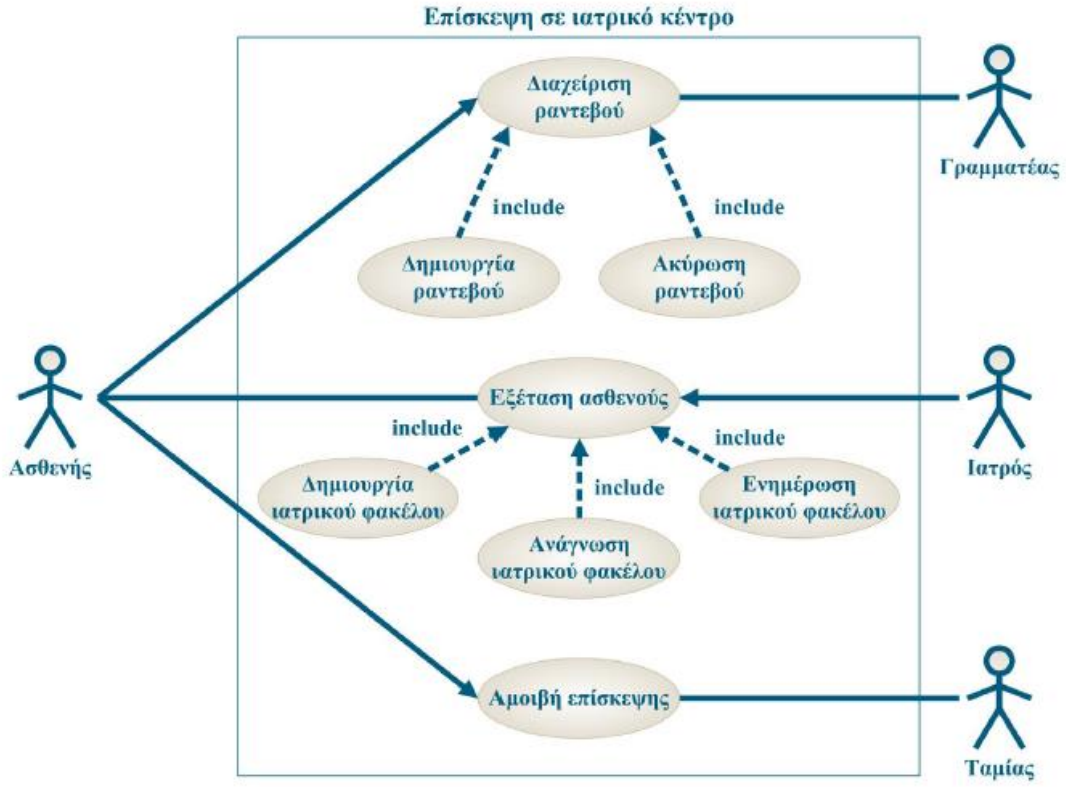
ή σε γνωστά συγγράμματα ανάλυσης ή σε εγχειρίδια της UML.

6.3.4.1 Παράδειγμα σεναρίου λειτουργίας επίσκεψης σε ιατρικό κέντρο (Μητάκος, 2015).

Αντιγράφουμε από το σύγγραμμα του Μητάκου (2015):

«Δημιουργήστε διάγραμμα περιπτώσεων χρήσης το οποίο να μοντελοποιεί την επίσκεψη ασθενούς σε ιατρικό κέντρο. Ο ασθενής, προκειμένου να επισκεφτεί το ιατρικό κέντρο, πρέπει πρώτα να κλείσει ραντεβού. Υπεύθυνη για τη διαχείριση των ραντεβού από την πλευρά του ιατρικού κέντρου είναι η γραμματέας του κέντρου. Λάβετε υπόψη ότι ένα ραντεβού δεν μπορεί μόνο να προγραμματιστεί, αλλά και να ακυρωθεί. Στη συνέχεια, ο ασθενής εξετάζεται από κάποιον γιατρό. Ο γιατρός δημιουργεί έναν ιατρικό φάκελο για τον ασθενή, σε περίπτωση που δεν υπάρχει ήδη. Αν υπάρχει ιατρικός φάκελος, τότε ο γιατρός τον διαβάζει, όταν ο ασθενής τον επισκεφτεί. Επίσης, μετά την εξέταση ο γιατρός ενημερώνει τον ιατρικό φάκελο του ασθενούς. Όταν ολοκληρωθεί η εξέταση, ο ασθενής πληρώνει στο ταμείο του ιατρικού κέντρου το κόστος της εξέτασής του από τον γιατρό».

Ακολουθεί ενδεικτική λύση στην εικόνα 6.50.



Εικόνα 6.50 Διάγραμμα περιπτώσεων χρήσης: επίσκεψη ασθενούς σε ιατρικό κέντρο (Μητάκος, 2015)

6.3.5 Συνέντευξη–Ερωτηματολόγιο

Στην ενότητα αυτή θα εξετάσουμε το πως προετοιμάζουμε μία συνέντευξη ή/και ένα ερωτηματολόγιο. Υπάρχουν πέντε (5) απαραίτητα βήματα:

- Διάβασε σχετικό υλικό
- Καθόρισε στόχους
- Αποφάσισε από ποιους θα πάρεις συνέντευξη
- Προετοίμασέ τους
- Αποφάσισε για τύπο ερωτήσεων και δομή.

Το ερωτηματολόγιο, συνήθως, ακολουθεί, συμπληρώνει και επιβεβαιώνει μία συνέντευξη.

Ποιοι οι θεμελιώδεις τύποι ερωτήσεων

- Ανοικτές (open – ended), δηλαδή ερωτήσεις που επιτρέπουν κάθε απάντηση
- Κλειστές (closed), δηλαδή ερωτήσεις που περιορίζουν/δεσμεύουν τις απαντήσεις κατά προκαθορισμένο τρόπο

Ποια η δομή της συνέντευξης

Υπάρχουν τρεις βασικές δομές συνέντευξης:

- Δομή πυραμίδας (pyramid), όπου αρχίζουμε με λεπτομερείς , κλειστές ερωτήσεις και συνεχίζουμε με πιο γενικές.

- Δομή χωνιού (funnel), όπου αρχίζουμε με ανοικτές γενικές ερωτήσεις και προχωρούμε με ειδικές κλειστές ερωτήσεις
- Δομή διαμαντιού (diamond), όπου συνδυάζονται οι δύο προηγούμενες με αποτέλεσμα, συνήθως, τη χρονική επιμήκυνση της συνέντευξης.

Ποια η δομή του ερωτηματολογίου

Περιλαμβάνει, τουλάχιστον, τις παρακάτω υποενότητες:

- Προκαταρκτικά, όπου πληροφορούμε τον ερωτώμενο για το σκοπό της έρευνας, τη σημασία της κ.τ.λ.
- Ενότητα ερωτήσεων, όπου διατυπώνονται με σαφή, σύντομο και απλό τρόπο οι ερωτήσεις μας. Ανάλογα με το αντικείμενο η ενότητα αυτή, συνήθως, υποδιαιρείται σε υποενότητες.
- Επίλογος – Ευχαριστίες

Τέλος, πρέπει να αναφερθεί ότι η συνέντευξη πρέπει να εξασφαλίζει ειδική άδεια για τη συλλογή εντύπων που χρησιμοποιεί η εταιρεία.

6.4 Μελέτη Περιπτώσεως Μηχανοργάνωση Εταιρείας σχεδιασμού και υλοποίησης εκπαιδευτικών σεμιναρίων

Ως βασικό θέμα προς διεκπεραίωση από τον αναγνώστη παραθέτουμε μία ενδιαφέρουσα μελέτη περιπτώσεως. Παράλληλα με τη διεκπεραίωση του θέματος, ο αναγνώστης καλείται να εφαρμόσει τις έννοιες και τις τεχνικές που παρουσιάστηκαν στην ενότητα 6.3 για να εμπεδώσει τον τρόπο εφαρμογής της τεχνολογίας των πληροφοριακών συστημάτων. Αρχίζουμε με την περιγραφή της εταιρίας.

Το Πανεπιστήμιο Αττικής έχει μία μονάδα, Information Technology training (ITt) η οποία οργανώνει σεμινάρια επιμόρφωσης-εκμάθησης σε θέματα πληροφορικής τεχνολογίας για τις παρακάτω ομάδες–στόχους (target groups):

- Προσωπικό εταιρειών, επ΄αμοιβή
- Άνεργους ηλικίας άνω των 25 και νέους απόφοιτους δευτεροβάθμιας εκπαίδευσης.
- Άτομα με ειδικές ανάγκες.

Πολλά από τα σεμινάρια που απευθύνονται στις δύο τελευταίες ομάδες στόχους γίνονταν σε συνεργασία με (ή με ανάθεση από) άλλους φορείς, π.χ. τοπική αυτοδιοίκηση, και συνήθως είναι επιδοτούμενα.

Η μονάδα ITt διαθέτει το τμήμα (υπο-μονάδα) ACTIVE BASE, το οποίο και αναλαμβάνει την υποστήριξη των εργασιών της ITt, π.χ. εργαστηριακό εξοπλισμό διεξαγωγής των σεμιναρίων, συγγραφή λογισμικού διαχείρισης των σεμιναρίων, αλλά και μελετητικό ή συμβουλευτικό έργο, για training, σε τρίτους. Στη μονάδα είναι σε λειτουργία ένα λογισμικό που καλύπτει κάποιες από τις ανάγκες της εταιρείας.

Ζητούμενο

Θα υποθέσουμε ότι αναλαμβάνετε, ως νέος συνεργάτης της μονάδας ITt, το έργο της βελτίωσης της μηχανοργάνωσής της με στόχο τη σχεδίαση και υλοποίηση ενός σύγχρονου συστήματος βάσης δεδομένων. Μη γνωρίζοντας, επακριβώς, το θέμα της λειτουργίας μονάδων, όπως η ITt,, και των δραστηριοτήτων και των λειτουργιών της της, αποφασίζετε να οργανώσετε:

1) Συνεντεύξεις με υπευθύνους της μονάδας
2) Συμπλήρωση ερωτηματολογίου από καθηγητές (εισηγητές) και εκπαιδευόμενους.

Θέλετε, τέλος, να συλλέξετε και να αναλύσετε έντυπα που χρησιμοποίησε ή χρησιμοποιεί η εταιρεία, εκτυπώσεις από το παλαιότερο σύστημα, γραμμογράφηση αρχείων κ.τ.λ.

Για όλα αυτά, δίδονται κάποια στοιχεία παραδείγματα στη συνέχεια ώστε να μπορέσετε να κατανοήσετε κάποιες διαδικασίες και κυρίως για να έχετε ένα παράδειγμα του τρόπου εργασίας.

Έτσι αφού μελετήσετε στοιχεία σχετικά με την αποτύπωση-τεκμηρίωση της δραστηριότητας της εταιρείας, μπορείτε, επίσης, να μελετήσετε και την καθημερινή λειτουργία της (ο όρος που χρησιμοποιούμε είναι «παρατήρηση»). Στη συνέχεια, μπορείτε να προχωρήσετε σε αποτύπωση του νέου συστήματος και σε περιγραφή της μετάπτωσης από το παλαιό στο νέο σύστημα.

6.4.1 Επισκόπηση της ανάλυσης και της σχεδίασης της ITt

Υπόβαθρο Πολλές φορές, υπάρχει ελάχιστη διαθέσιμη καταγεγραμμένη πληροφορία/τεκμηρίωση (εκθέσεις, ανασκοπήσεις κ.τ.λ.). Έτσι, σε πρώτη προσέγγιση, αφού μελετηθούν τα στοιχεία αυτά για να βοηθηθείτε στην κατανόηση της λειτουργίας της εταιρείας η ανάλυση ξεκινά ουσιαστικά με τις συνεντεύξεις.

Συνεντεύξεις

Στη μελέτη περιπτώσεως θα μπορούσατε να ακολουθήσετε την εξής πορεία:

- Σχεδίαση συνέντευξης ή συνεντεύξεων
- Αρχική (preliminary) συνέντευξη με εκπαιδευτικό διευθυντή της μονάδας. Προφανώς, μία μονάδα αυτού του τύπου μπορεί να έχει καθηγητή υπεύθυνο-διευθυντή εκπαίδευσης (Education Director) ή/και γενικό διευθυντή (General Manager). Όλες οι συνεντεύξεις πρέπει να ολοκληρώνονται με την καταγραφή τους.
- Η πρώτη συνέντευξη επειδή συνήθως δεν υπάρχει πλούσιο υλικό πρέπει να έχει δομή χωνιού:
- Ανοικτές ερωτήσεις που προοδευτικά οδηγούν σε κλειστές.
- Μετά από αυτή τη συνέντευξη μπορούν να καθοριστούν οντότητες και συσχετίσεις και να γίνει μία πρώτη προσπάθεια σχεδιασμού μοντέλου οντοτήτων–συσχετίσεων (ΜΟΣ). Στο σημείο αυτό πρέπει να επισημάνουμε ότι μία πληρέστερη προσέγγιση θα έπρεπε να αποτυπώσει τη ροή της πληροφορίας, π.χ., με χρήση διαγραμμάτων ροής δεδομένων.
- Επόμενη (follow-up) συνέντευξη, π.χ., με εκπαιδευτικό διευθυντή (Education Director) ή γενικό διευθυντή (General Manager) – Καταγραφή
- Η επόμενη συνέντευξη πρέπει να προετοιμαστεί πολύ προσεκτικά. Οι ερωτήσεις πρέπει να σχεδιάζονται και να ομαδοποιούνται κατά αντικείμενο. Είναι σκόπιμο οι ερωτήσεις να δομηθούν με τη μέθοδο της πυραμίδας ώστε να οριστεί, αρχικά, το αντικείμενο με κλειστές σύντομες ερωτήσεις πριν προχωρήσουμε σε ανοικτές που θα διευρύνουν την κατανόηση. Η συνέντευξη μπορεί να ολοκληρωθεί με σύντομη ανασκόπηση για να επιβεβαιωθεί η κατανόηση των απαντήσεων.
- Τελική (final interview) συνέντευξη πχ. με γενικό διευθυντή (General Director) – Καταγραφή
- Η συνέντευξη θα προσπαθήσει να επαληθεύσει το μοντέλο πληροφορίας, να απαντήσει σε κάποιες σημαντικές ερωτήσεις και να βελτιώσει την κατανόηση του συστήματος. Η τελική κατά κανόνα βασίζεται σε κλειστές ερωτήσεις.
- Λίστα με Συμπεράσματα που αποτυπώσατε και τελική σχεδίαση ΜΟΣ.

Ακολουθούν παραδείγματα ερωτήσεων για τη δεύτερη συνέντευξη

Γενικές ερωτήσεις

1) Οργανώνετε αίθουσες διεξαγωγής και αναλαμβάνετε και τον χρονοπρογραμματισμό των σεμιναρίων ή απλά καθορίζετε εισηγητές και εξοπλισμό-μηχανήματα;
2) Ποιος καθορίζει τους εκπαιδευόμενους;
3) Τα σεμινάρια έχουν ως επικεφαλής έναν από τους εισηγητές ή έχουν οργανωτή που είναι επικεφαλής χωρίς να διδάσκει;
4) Τι σεμινάρια προσφέρετε σε Δήμους;
5) Πως κατηγοριοποιείτε τον εξοπλισμό που δανείζετε (στους δήμους κ.τ.λ.) για τη διεξαγωγή των σεμιναρίων;

Ερωτήσεις για Δήμους

1) Πως κωδικοποιείτε τους Δήμους με τους οποίους συνεργάζεστε;
2) Πως είναι οργανωμένα τα σεμινάρια; Για παράδειγμα, κάθε σεμινάριο μπορεί να έχει πολλά θέματα–subjects και κάθε θέμα διδάσκεται τουλάχιστον μία φορά την εβδομάδα);
3) Ποια η διάρκεια των σεμιναρίων;
4) Μπορείτε να αναλύσετε τον τρόπο επιλογής των εκπαιδευομένων;

Ερωτήσεις για επιχειρήσεις/εταιρείες

1) Τα σεμινάρια που παρέχεται είναι ταυτόσημα για εταιρείες και Δήμους;
2) Πως κωδικοποιείτε τις εταιρείες;
3) Ποιος παίρνει παρουσίες και ρυθμίζει γενικά τα θέματα φοίτησης του προσωπικού εταιρειών;

Ερωτήσεις για το μέλλον

1) Πως βλέπετε την ανάπτυξη της εταιρείας στο εγγύς και στο απώτερο μέλλον;

6.4.2 Έντυπα αποτύπωσης λειτουργιών σύμφωνα με τις συνεντεύξεις

Στον πίνακα 6.6 βλέπουμε την αποτύπωση της συνέντευξης.

Πίνακας 6.6 Αποτύπωση συνέντευξης

Ερωτώμενος (Interviewee): **κος Αστέρης**	Ημερομηνία (Date): 22/2/2022
Θέση (Position): **Διευθυντής εκπαίδευσης**	Θέμα (Subject): Μονάδα ITt
Στόχοι συνέντευξης **(Objectives of the interview).**	1) Κατανόηση σε γενικές γραμμές της μονάδας 2) Καθορισμός δραστηριοτήτων 3) Καθορισμός στόχων για τις επόμενες συνεντεύξεις
Αξιολόγηση συνέντευξης **(Were the objectives meet?)**	Ναι, πιστεύω ότι έχω κατανοήσει την εταιρεία, πως εργάζεται (λειτουργεί), τις θεματικές περιοχές που δραστηριοποιείται και μπορώ να διατυπώσω ερωτήσεις για περαιτέρω συνεντεύξεις. Έχω, επιπλέον, μία βάση για τη μοντελοποίηση της εταιρείας.

Main points (Κύρια σημεία)	
1)	Η μονάδα ITt οργανώνει σεμινάρια παρέχοντας καθηγητές και εξοπλισμό.
2)	Έχει τρεις περιοχές δραστηριότητας: Εταιρείες, δήμους, άτομα με ειδικές ανάγκες. Η συνεργασία με Δήμους σχεδόν μονοπωλεί τη δραστηριότητά της και συνίσταται κατά κύριο λόγο στην εκπαίδευση/επιμόρφωση νέων αποφοίτων λυκείου και ανέργων ηλικίας μεγαλύτερης των 25.
3)	Προς το παρόν οργανώνονται περίπου 100 σεμινάρια για 15 Δήμους ετησίως. Οι αριθμοί αυτοί αναμένεται να αυξηθούν στο μέλλον.
4)	Οι εκπαιδευόμενοι παρακολουθούν ένα μόνο σεμινάριο κάθε φορά και επιδοτούνται.
5)	Κάθε σεμινάριο, έχει ένα τίτλο, υποδιαιρείται σε θέματα (subjects) και διεξάγεται καθημερινά επί 5ωρο, συνήθως επί 4μηνο.
6)	Κάθε θέμα έχει ένα εκπαιδευτή (“εισηγητή”) ενώ ένας εκπαιδευτής μπορεί να διδάσκει πολλά θέματα σε πολλά σεμινάρια.
7)	Ο πελάτης είναι υπεύθυνος για τις αίθουσες διδασκαλίας.
8)	Οι επιχειρήσεις και τα άτομα με ειδικές ανάγκες απαιτούν σεμινάρια προσαρμοσμένα στις ανάγκες τους.
9)	Οι δήμοι επιλέγουν σεμινάρια από μία λίστα 15 διαφορετικών τύπων σεμιναρίων.
10)	Η εταιρεία αναλαμβάνει παρουσιολόγια σπουδαστών/καθηγητών για κάθε σεμινάριο και χρησιμοποιεί μία γραμματέα ανά δήμο.
11)	Σε περίπτωση απουσίας ένας εκπαιδευόμενος ή εισηγητής δεν πληρώνεται κτλ.

Παραδείγματα ερωτήσεων στην τελική συνέντευξη

1) Για τα σεμινάρια που συγχρηματοδοτούνται από κράτος/ΕΕ τα χρήματα πληρώνονται στον Ειδικό Λογαριασμό Κονδυλίων Έρευνας (ΕΛΚΕ) του Πανεπιστημίου ή απ’ ευθείας στη μονάδα;
2) Θα μπορούσα να συλλέξω τις απαραίτητες πληροφορίες/στοιχεία (documents) που κρατάτε για τους καθηγητές και τους σπουδαστές;
3) Για τα χρηματοδοτούμενα σεμινάρια εμπλέκεστε σε θέματα πληρωμών των εκπαιδευομένων ή απλά και μόνο στις λεπτομέρειες της παρακολούθησης των σεμιναρίων (απουσίες κ.τ.λ.);

6.4.3 Ερωτηματολόγια για τη μελέτη περιπτώσεως

Στη μελέτη περιπτώσεως θα μπορούσατε να ακολουθήσετε την εξής πορεία:

Πρώτα να σχεδιάσετε τα ερωτηματολόγια σας.

Μία συνηθισμένη δομή ερωτηματολογίου είναι:

- Εισαγωγή (ποιος είστε, πως και γιατί ρωτάτε κ.τ.λ.)
- Γενικές ερωτήσεις
- Ερωτήσεις για τα σεμινάρια
- Ερωτήσεις γύρω από Ρόλους εμπλεκομένων
- Ερωτήσεις για οικονομικά στοιχεία
- Επίλογος (ευχαριστίες κ.τ.λ.)

Μετά θα επεξεργαστείτε μία λίστα με συμπεράσματα που αποτυπώσατε και θα προχωρήσετε στη σχεδίαση ‘η την επανασχεδίαση του ΜΟΣ.

6.4.3.1 Παραδείγματα ερωτήσεων των ερωτηματολογίων για συνεργαζόμενους εισηγητές με τη μονάδα ITt

1) Πως βλέπετε τη λειτουργία της μονάδας;
2) Θα μπορούσατε να περιγράψετε την οργανωτική δομή της;
3) Πως πρωτοακούσατε για την εταιρεία; Διαφήμιση, φίλος/συνάδελφος, επικοινωνία με πρωτοβουλία της μονάδας, αλλιώς
4) Τι περιλάμβανε η διαδικασία επιλογής; (Συνέντευξη κ.τ.λ.)
5) Σε ποιους φορείς διδάσκετε; (Συγκεκριμένη λίστα φορέων)
6) Σε ποια σεμινάρια διδάσκετε; (Συγκεκριμένες λίστες ομαδοποιημένες κατά αντικείμενο)
7) Ποια η διάρκεια αυτών των σεμιναρίων;
8) Ποια θέματα (subjects) διδάσκετε στα σεμινάρια και πόσες ώρες την εβδομάδα;
9) Εμπλέκεστε σε δημοσιοποίηση/διαφήμιση της μονάδας και των σεμιναρίων;
10) Συμμετέχετε στην επιλογή σπουδαστών και σπουδαστριών; Αν ναι ποια διαδικασία ακολουθείτε; (συνέντευξη, γραπτή εξέταση, προσόντα, συστάσεις, υπόδειξη, αλλιώς–πως)
11) Πως αξιολογείτε τους σπουδαστές;
12) Πως πληρώνεστε; (Ανά ενότητα, Μηνιαίως κ.τ.λ.)
13) Πληρώνεστε σε περίπτωση ασθένειας;
14) Ποιος ελέγχει την παρουσία σας; (γραμματέας, ...)
15) Πληρώνεστε έξοδα μετακίνησης; Αν ναι τότε αυτά είναι καθορισμένα (fixed to your status) ή τα ζητάτε υποβάλλοντας δικαιολογητικά (claimed by you)
16) Πληρώνεστε το ίδιο ποσό ανεξάρτητα από το είδος του σεμιναρίου; Αν όχι πως διαφοροποιείται η πληρωμή; (διαφορετικοί τύποι σεμιναρίων έχουν διαφορετική κλίμακα αμοιβής–rate, κάθε σεμινάριο έχει δική του κλίμακα, οι συνεργαζόμενοι δήμοι έχουν μία δική τους κλίμακα, αλλιώς–πως)
17) Πόσες φόρμες συμπληρώνετε και κάθε πότε;
18) Πιστεύετε ότι όλες αυτές οι φόρμες είναι ουσιώδεις;

6.5 Συλλογή εντύπων για την εταιρεία ITt

Στην εικόνα 6.51 βλέπουμε παράδειγμα εντύπου παρουσιολογίου για τους εισηγητές.

ITt Doc15

Teacher Monthly Registration Form

Business: .....
Month : .....
Name: .....

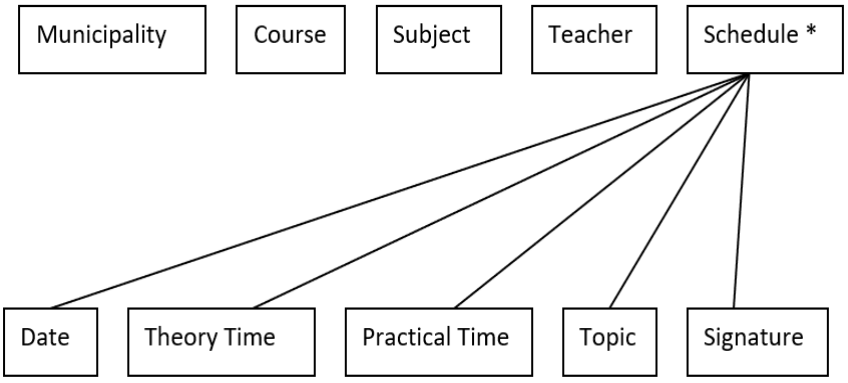
Date	Course	Start time	Finish time	Total hours	Signature	Actual hours

Total Hours per Month

Secretary

Εικόνα 6.51 Παρουσιολόγιο για τους εισηγητές.

Στην εικόνα 6.52 δίδεται ένα συγκεκριμένο παράδειγμα διαγραμματικής τεχνικής, τα διαγράμματα χρήσης δεδομένων (Data Usage Diagrams) (βλέπε μεθοδολογία Jackson). Είναι ένα απλό παράδειγμα χρήσης διαγραμματικής τεχνικής για την ανάλυση των στοιχείων ενός εντύπου. Πιο συγκεκριμένα, τα διαγράμματα χρήσης δεδομένων περιγράφουν τα στοιχεία που βλέπει κάποιος στη φόρμα και τις συσχετίσεις αυτών των στοιχείων. Για μία φόρμα χρονοπρογραμματισμού ωρών διδάσκοντος (Hourly Teacher Schedule) το διάγραμμα (data usage diagram) μπορεί να είναι αυτό της εικόνας 6.52.



Εικόνα 6.52 Διάγραμμα Jackson για το παρουσιολόγιο των εισηγητών.

6.6 Ανάλυση δεδομένων των εντύπων και σχεδίαση ΜΟΣ

Μετά την ολοκλήρωση των συνεντεύξεων και της παραπάνω εργασίας (ή και κατά τη διάρκεια τους) αρχίζει η διαδικασία σχεδίασης του Εννοιολογικού μοντέλου (Conceptual model) και στη συνέχεια η σχεδίαση του σχεσιακού μοντέλου (Relational model, Normalization) και της διεπαφής του χρήστη.