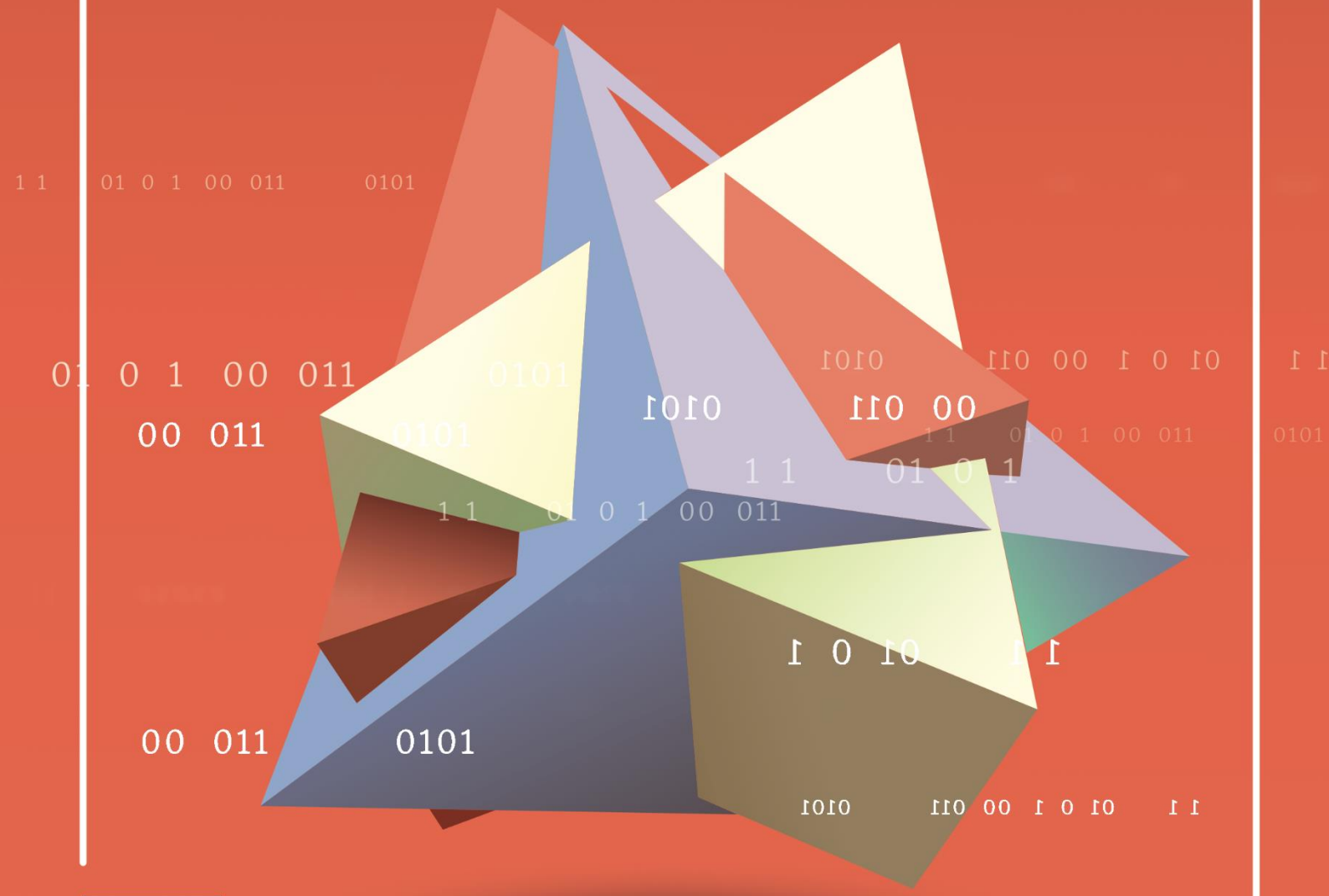
Συστήματα Βάσεων Δεδομένων

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 Β

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 Β

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

Σύνοψη
Στο κεφάλαιο αυτό περιγράφονται σημαντικές έννοιες ελέγχου δεδομένων, ο μηχανισμός της όψης, και η σύνταξη και η σημασιολογία των δηλώσεων της γλώσσας ελέγχου δεδομένων-DCL (Data Control Language). Γίνεται, ακόμη, αναφορά στο ρόλο του διαχειριστή βάσης δεδομένων (Data Base Administrator-DBA) και στη σημασία του DBA για τη συνέπεια (consistency), την ασφάλεια (security) και την ακεραιότητα (integrity) των δεδομένων. Παρατίθεται αναλυτική περιγραφή των δικαιωμάτων πρόσβασης. Γίνεται μια εισαγωγή στις συναλλαγές (transactions). Η διεκπεραίωση του θέματος βασίζεται σε παραδείγματα κυρίως στην (υπο)γλώσσα ελέγχου δεδομένων της SQL, όπως υποστηρίζεται στο προϊόν της Oracle αλλά και στο προϊόν MySQL.

Στο κεφάλαιο περιλαμβάνονται οι παρακάτω ενότητες:

1) *Ο **Ρόλος** και τα **καθήκοντα** του **διαχειριστή** βάσεων δεδομένων. Γλώσσα ελέγχου δεδομένων. Συναλλαγές ή δοσοληψίες (transactions). Δηλώσεις COMMIT, ROLLBACK της γλώσσας SQL. Όψεις (Views). Ενημερωσιμότητα όψεων. Προγραμματισμός όψεων στο προϊόν της Oracle.*

2) ***Περιήγηση.** Διαχείριση της ασφάλειας ενός συστήματος βάσης δεδομένων προσωπικού με τη χρήση της υπογλώσσας ελέγχου δεδομένων της ORACLE SQL.*

   Στο πλαίσιο της περιήγησης περιγράφεται και ο ρόλος του διαχειριστή βάσης δεδομένων (DBA) και πώς συμβάλλει στην ασφάλεια του συστήματος. Η περιγραφή σημαντικών καθηκόντων του DBA γίνεται μέσα από παραδείγματα.

3) ***Περιήγηση.** Ασφάλεια βάσης δεδομένων και προϊόν MySQL.*

   Στην περιήγηση παρουσιάζουμε θέματα ασφάλειας βάσεων δεδομένων στο προϊόν MySQL. Η συζήτηση γίνεται στο πλαίσιο της μελέτης ενός συστήματος βάσης δεδομένων προσωπικού και της υλοποίησης της ασφάλειας του συστήματος και των δεδομένων με χρήση του προϊόντος MySQL. Παρουσιάζονται θέματα της ακεραιότητας των δεδομένων, του ορισμού μηχανισμού κύριου (primary key) και ξένου κλειδιού (foreign key) που διασφαλίζει την τήρηση των δύο κανόνων ακεραιότητας (Integrity Rule 1, Referential Integrity Rule), του ορισμού περιορισμών δεδομένων (Constraints, π.χ. CHECK το οποίο δεν υποστηρίζεται προς το παρόν από το προϊόν) και της τήρησης επιχειρηματικών κανόνων (Business Rules) κ.λπ.

Προαπαιτούμενη γνώση
Προτείνεται η μελέτη του κεφαλαίου 3 του παρόντος συγγράμματος

7.1 Εισαγωγή. Περιήγηση στις υπο-γλώσσες της γλώσσας SQL

Η γλώσσα διαχείρισης βάσεων δεδομένων SQL περιλαμβάνει τις παρακάτω υπογλώσσες:

- Τη γλώσσα ορισμού δεδομένων DDL (Data Definition Language)
- Τη γλώσσα ελέγχου δεδομένων DCL (Data Control Language)
- Τη γλώσσα χειρισμού (διαχείρισης) δεδομένων DML (Data Manipulation Language)

Επιπλέον, είναι αρκετά συχνή στη βιβλιογραφία η αναφορά σε μία ακόμη υπογλώσσα:

- Τη γλώσσα ελέγχου δοσοληψιών TCL (Transaction Control Language).

Τέλος, κάποιοι κατηγοριοποιούν τις SQL δηλώσεις κάπως διαφορετικά σε τέσσερις κατηγορίες:

- DDL – Data Definition Language
- DQL– Data Query Language
- DML – Data Manipulation Language
- DCL – Data Control Language

7.1.1 DDL (Data Definition Language)

Η γλώσσα ορισμού δεδομένων αποτελείται από τις δηλώσεις SQL που χρησιμοποιούνται για τον ορισμό του σχήματος της βάσης δεδομένων και τη δημιουργία και την τροποποίηση της δομής των αντικειμένων της βάσης δεδομένων. Δεν χρησιμοποιούνται για την εισαγωγή σε πίνακες, την τροποποίηση και τη διαγραφή των δεδομένων. Προσοχή! Οι δηλώσεις της γλώσσας δεν χρησιμοποιούνται από έναν τελικό χρήστη, ο οποίος έχει πρόσβαση στη βάση δεδομένων μέσω μιας εφαρμογής.

Ακολουθεί λίστα δηλώσεων της DDL:

CREATE: Χρησιμοποιείται για τη δημιουργία της βάσης δεδομένων και των αντικειμένων της, π.χ., πίνακας (table), ευρετήριο (index), όψη (view), συνάρτηση (function), αποθηκευμένη διαδικασία (stored procedure) και ενεργοποιητές-εναύσματα (trigger).

DROP: Χρησιμοποιείται για τη διαγραφή αντικειμένων της βάσης δεδομένων.

ALTER: Χρησιμοποιείται για την αλλαγή της δομής της βάσης δεδομένων.

TRUNCATE: Χρησιμοποιείται για την κατάργηση όλων των εγγραφών (record) από έναν πίνακα, συμπεριλαμβανομένου και του χώρου που έχουν διατεθεί για τις εγγραφές.

COMMENT: Χρησιμοποιείται για την προσθήκη σχολίων στο λεξικό δεδομένων.

RENAME: Χρησιμοποιείται για τη μετονομασία ενός αντικειμένου της βάσης δεδομένων.

Στο κεφάλαιο 3 εστιάσαμε στις δηλώσεις της γλώσσας DDL για να ορίσουμε τη βάση και τα αντικείμενά της.

7.1.2 DCL (Γλώσσα ελέγχου δεδομένων)

Η γλώσσας περιλαμβάνει δηλώσεις SQL που ελέγχουν την πρόσβαση στα δεδομένα και στη βάση δεδομένων. Πιο συγκεκριμένα, περιλαμβάνει τις δηλώσεις GRANT και REVOKE οι οποίες διαχειρίζονται τα δικαιώματα των χρηστών, τα δικαιώματα πρόσβασης στα δεδομένα και άλλους ελέγχους του συστήματος βάσης δεδομένων:

- Η δήλωση GRANT δίνει στους χρήστες δικαιώματα πρόσβασης στη βάση δεδομένων.

- Η δήλωση REVOKE αφαιρεί τα δικαιώματα πρόσβασης του χρήστη που έχουν εκχωρηθεί χρησιμοποιώντας τη δήλωση GRANT.

7.1.3 DML (Data Manipulation Language)

Στη γλώσσα DML περιλαμβάνονται οι δηλώσεις SQL οι οποίες διαχειρίζονται τα δεδομένα που υπάρχουν στη βάση δεδομένων. Στο κεφάλαιο 3 εξετάσαμε αναλυτικά τις δηλώσεις οι οποίες περιλαμβάνονται στην DML: δήλωση SELECT, δηλώσεις INSERT, UPDATE, DELETE. Η δήλωση SELECT σε κάποιες αναφορές περιλαμβάνεται στη γλώσσα DQL.

Ακολουθεί λίστα δηλώσεων της DML:

INSERT: Χρησιμοποιείται για την εισαγωγή δεδομένων σε έναν πίνακα.

UPDATE: Χρησιμοποιείται για την ενημέρωση υπαρχόντων δεδομένων σε έναν πίνακα.

DELETE: Χρησιμοποιείται για τη διαγραφή εγγραφών από έναν πίνακα βάσης δεδομένων.

LOCK: Χρησιμοποιείται για τη διαχείριση της ταυτόχρονης πρόσβασης και επεξεργασίας των δεδομένων (concurrency control).

CALL: Κλήση ενός υποπρογράμματος PL/SQL ή JAVA.

EXPLAIN PLAN: Περιγράφει τη διαδρομή πρόσβασης στα δεδομένα (the access path to data).

Παρατήρηση! Θα μπορούσαμε να θεωρήσουμε ότι στην παραπάνω λίστα των δηλώσεων της DML κάποιες δηλώσεις είναι (και) δηλώσεις της DCL.

7.1.4 DQL (Data Query Language)

Οι δηλώσεις DQL χρησιμοποιούνται για την εκτέλεση ερωτημάτων (queries) στα δεδομένα των αντικειμένων του σχήματος της βάσης δεδομένων. Μία δήλωση της DQL ανακτά (λαμβάνει) δεδομένα από τη βάση δεδομένων και μάλιστα μπορεί να τα επεξεργαστεί, να τα διατάξει (order by), να τα ομαδοποιήσει κ.λπ. Στη γλώσσα περιλαμβάνεται μόνο η δήλωση SELECT η οποία χρησιμοποιείται για την ανάκτηση δεδομένων από τη βάση δεδομένων. Μία δήλωση SELECT εκτελείται στα δεδομένα ενός ή περισσοτέρων πινάκων και το αποτέλεσμα τοποθετείται σε έναν προσωρινό πίνακα, ο οποίος εμφανίζεται άμεσα ΄στο χρήστη ή λαμβάνεται από το front-end πρόγραμμα για περαιτέρω ενέργειες.

7.1.5 TCL (Transaction Control Language)

Οι δηλώσεις TCL ασχολούνται με τις συναλλαγές στη βάση δεδομένων.

Ακολουθεί λίστα δηλώσεων της TCL:

COMMIT: Οριστικοποιεί τις μεταβολές που επιφέρει στα δεδομένα μια συναλλαγή (Transaction).

ROLLBACK: Αναιρεί τις μεταβολές που επιφέρει στα δεδομένα μια συναλλαγή σε περίπτωση οποιουδήποτε σφάλματος.

SAVEPOINT: Ορίζει ένα σημείο αποθήκευσης σε μια συναλλαγή.

SET TRANSACTION: Καθορίζει κάποια χαρακτηριστικά για τη συναλλαγή.

Κύρια αποστολή της Γλώσσας Ελέγχου Δεδομένων DCL (Data Control Language) είναι η προστασία της βάσης δεδομένων από ανεπιθύμητες μεταβολές.

- Η προστασία συνίσταται στο γεγονός ότι έχει καθοριστεί διαβάθμιση **δικαιωμάτων πρόσβασης** με την οποία κάθε συγκεκριμένη ομάδα χρηστών περιορίζεται σε συγκεκριμένες διαδικασίες ορισμού και χειρισμού δεδομένων. Τα δικαιώματα πρόσβασης εξασφαλίζουν την ασφάλεια των δεδομένων.
- Επίσης, συνίσταται στη διασφάλιση της συνέπειας των στοιχείων της βάσης δεδομένων. Η συνέπεια πολλές φορές εξαρτάται από συγκεκριμένα σύνολα ενεργειών (δηλώσεων SQL) που προκαλούν τη μεταβολή των δεδομένων. Τα σύνολα αυτά ονομάζονται συναλλαγές (**transactions**) και αποτελούν μία ενότητα. Δηλαδή, μία συναλλαγή αποτελείται από δηλώσεις που είναι αλληλεξαρτώμενες και εκτελούνται με τη λογική «όλα ή τίποτα». Για παράδειγμα, δύο δηλώσεις UPDATE αποτελούν μια τραπεζική συναλλαγή και εκτελούν τη μεταφορά χρημάτων από έναν τραπεζικό λογαριασμό σε έναν άλλο. Οι δηλώσεις αποτελούν ενότητα, και είτε ολοκληρώνονται και οι δύο είτε αναιρούνται και οι δύο. Δεν είναι δυνατόν να αφαιρέσει η πρώτη δήλωση UPDATE ένα ποσό από ένα λογαριασμό και στη συνέχεια να αποτύχει η δεύτερη δήλωση και να μη μεταφερθούν τα χρήματα.
- Τέλος, συνίσταται στον ορισμό περιορισμών που διασφαλίζουν την ακεραιότητα των δεδομένων.

7.2 Η σημασία της γλώσσας DCL. Δικαιώματα Πρόσβασης

Στη συνέχεια θα μιλήσουμε αναλυτικά για τα δικαιώματα πρόσβασης και τις συναλλαγές. Η συζήτηση θα βασιστεί στην (υπο)γλώσσα ελέγχου δεδομένων της SQL, όπως υποστηρίζεται στο προϊόν της Oracle. Επιπλέον, στο πλαίσιο μελέτης περίπτωσης θα συζητήσουμε τα θέματα της ακεραιότητας των δεδομένων, του ορισμού μηχανισμού κύριου (primary key) και ξένου κλειδιού (foreign key), που διασφαλίζει την τήρηση των δύο κανόνων ακεραιότητας (Integrity Rule 1 or Entity Integrity, Referential Integrity Rule), του ορισμού περιορισμών (Constraints, π.χ. CHECK) δεδομένων και της τήρησης επιχειρηματικών κανόνων (business rules). Στα κεφάλαια 8 και 9 θα συζητήσουμε περαιτέρω τη χρήση αποθηκευμένων διαδικασιών, triggers, procedures κ.λπ., για τη διασφάλιση της τήρησης περιορισμών και επιχειρηματικών κανόνων.

Υπάρχουν δύο διαφορετικά είδη δικαιωμάτων πρόσβασης:

- Δικαιώματα που καθορίζουν το είδος της πρόσβασης των χρηστών.
- Δικαιώματα που καθορίζουν τον τύπο των ενεργειών, που επιτρέπονται σε πίνακες.

7.2.1 Δικαιώματα πρόσβασης χρηστών CONNECT, RESOURCE, DBA

Υπάρχουν τριών ειδών δικαιώματα: CONNECT, RESOURCE, DBA.

Δικαίωμα CONNECT

Οι χρήστες με CONNECT δικαίωμα έχουν τη δυνατότητα:

- πρόσβασης στο περιβάλλον της βάσης, π.χ., ORACLE,
- να δουν τα περιεχόμενα πινάκων που ανήκουν σε άλλους χρήστες με την προϋπόθεση ότι έχουν το SELECT δικαίωμα για αυτούς τους πίνακες,
- να εκτελέσουν ενέργειες χειρισμού δεδομένων (INSERT, UPDATE, DELETE) σε πίνακες άλλων χρηστών εάν τους έχει δοθεί το αντίστοιχο δικαίωμα,

Οι χρήστες με το CONNECT δικαίωμα δεν μπορούν:

- να δημιουργήσουν ή να διαγράψουν πίνακες, δείκτες (index) ή συστάδες πινάκων (cluster),
- να μεταβάλλουν τη δομή πινάκων.

Δικαίωμα RESOURCE

Η ύπαρξη αυτού του δικαιώματος προϋποθέτει την ύπαρξη του δικαιώματος CONNECT. Ο χρήστης με αυτό το δικαίωμα έχει τη δυνατότητα να:

- δημιουργήσει πίνακες, ευρετήρια (index) και συστάδες (cluster) και να χειριστεί αυτά τα αντικείμενα χωρίς κανένα περιορισμό.
- παραχωρήσει και να αφαιρέσει δικαιώματα πρόσβασης σε άλλους χρήστες για τα αντικείμενα που του ανήκουν.
- χρησιμοποιήσει τη δήλωση AUDIT για τον έλεγχο της πρόσβασης στα αντικείμενα που του ανήκουν.

Ο χρήστης με RESOURCE δικαίωμα δεν έχει τη δυνατότητα να:

- χειριστεί πίνακες που έχουν δημιουργηθεί από άλλο χρήστη, εκτός αν του έχει παραχωρηθεί το ανάλογο δικαίωμα πρόσβασης,
- παραχωρήσει ή να αφαιρέσει CONNECT ή RESOURCE δικαιώματα σε άλλους χρήστες.

Δικαίωμα DBA

Το δικαίωμα DBA είναι το πιο ισχυρό δικαίωμα πρόσβασης. Ένας χρήστης με αυτό το δικαίωμα έχει όλα τα δικαιώματα που συνεπάγονται το CONNECT δικαίωμα και το RESOURCE δικαίωμα και έχει τη δυνατότητα να:

- προσπελάσει τους πίνακες όλων των χρηστών και να εκτελέσει οποιαδήποτε ενέργεια πάνω σε αυτούς.
- παραχωρήσει και να αφαιρέσει δικαιώματα πρόσβασης χρηστών στη βάση (CONNECT, RESOURCE, DBA),
- δημιουργήσει γενικά συνώνυμα (PUBLIC SYNONYM),
- δημιουργήσει και να τροποποιήσει διαμερίσεις (partitions),
- παρακολουθήσει την πρόσβαση στο σύστημα της βάσης δεδομένων καθώς και την πρόσβαση σε πίνακες χρηστών,
- αποθηκεύσει τη βάση δεδομένων σαν απλό αρχείο λειτουργικού συστήματος για λόγους δημιουργίας εφεδρικού αντιγράφου ή μεταφοράς.

Σημείωση.

Οι χρήστες για να δημιουργήσουν όψη (προβολή δεδομένων, view) σε δικό τους σχήμα (database schema) πρέπει να έχουν το δικαίωμα CREATE VIEW. Για να δημιουργήσουν μια όψη σε σχήμα άλλου χρήστη, πρέπει να έχουν το δικαίωμα CREATE ANY VIEW. Για να δημιουργήσουν μια δευτερεύουσα όψη, πρέπει να έχουν το δικαίωμα UNDER ALL VIEW.

7.2.2 Παραχώρηση δικαιωμάτων πρόσβασης σε χρήστες

Η παραχώρηση σε ένα χρήστη με συγκεκριμένο όνομα και συνθηματικό ενός ή περισσοτέρων δικαιωμάτων πρόσβασης στο σύστημα της ORACLE γίνεται με την ακόλουθη δήλωση:

```
GRANT {CONNECT | RESOURCE | DBA}
TO   όνομα_χρήστη
 [IDENTIFIED BY  συνθηματικό];
```

Η παραπάνω δήλωση μπορεί να χρησιμοποιηθεί για να δημιουργήσει νέο χρήστη, να δώσει επιπλέον δικαιώματα πρόσβασης σε χρήστες ή για την αλλαγή συνθηματικών.

Η αφαίρεση δικαιωμάτων πρόσβασης από χρήστες γίνεται με τη δήλωση:

```
REVOKE {CONNECT | RESOURCE | DBA}
FROM όνομα_χρήστη;
```

7.2.3 Δικαιώματα πρόσβασης χρηστών σε αντικείμενα

Κάθε πίνακας και κάθε όψη (VIEW) έχει έναν μοναδικό ιδιοκτήτη. Μόνο ο ιδιοκτήτης ενός αντικειμένου μπορεί στην αρχή να προσπελάσει το συγκεκριμένο αντικείμενο. Αν θελήσει να επιτρέψει και σε άλλους χρήστες να δουν ή να τροποποιήσουν κάποιον από τους πίνακές του πρέπει να το κάνει εκτελώντας την ακόλουθη δήλωση:

```
GRANT {δικαίωμα, ... | ALL }
ON πίνακας
TO {  όνομα_χρήστη | PUBLIC }
[WITH GRANT OPTION];
```

όπου δικαίωμα είναι ένα από τα ακόλουθα:

```
SELECT, INSERT, UPDATE, DELETE, ALTER, INDEX.
```

Με το ALL παρέχονται όλα τα προηγούμενα δικαιώματα.

Με το PUBLIC τα καθορισμένα δικαιώματα παρέχονται σε όλους τους χρήστες που έχουν το δικαίωμα πρόσβασης CONNECT.

Η υποπρόταση WITH GRANT OPTION επιτρέπει στον εξουσιοδοτημένο, με τα δικαιώματα που εμφανίζονται στην δήλωση, χρήστη να μεταβιβάσει αυτά τα δικαιώματα σε άλλους χρήστες.

Η αφαίρεση δικαιωμάτων πρόσβασης σε αντικείμενα από χρήστες γίνεται με τη δήλωση REVOKE, η οποία συντάσσεται ως εξής:

```
REVOKE  {δικαίωμα, ... | ALL }
ON  πίνακας
FROM { χρήστης  | PUBLIC };
```

7.3 Συναλλαγές (Transactions)

Ο όρος transaction έχει μεταφραστεί στα ελληνικά σαν συναλλαγή ή δοσοληψία ή κίνηση. Στο βιβλίο αυτό θα χρησιμοποιήσουμε συνήθως τον όρο συναλλαγή που αποδίδει την έννοια σε τραπεζικό περιβάλλον, π.χ. τραπεζικές συναλλαγές.

Μια συναλλαγή (transaction) μπορεί να ορισθεί περιγραφικά ως σύνολο μεταβολών που γίνονται στη βάση δεδομένων μεταξύ δηλώσεων COMMIT και που αποτελούν μια λογική ενότητα τέτοια ώστε να διασφαλίζεται η συνέπεια της βάσης δεδομένων.

Στην βιβλιογραφία υπάρχουν αυστηρότεροι ορισμοί της έννοιας αυτής. Περισσότερα είδαμε στο κεφάλαιο 2 του συγγράμματος. Ακολουθούν δηλώσεις χρήσιμες για τη διαχείριση συναλλαγών.

7.3.1 COMMIT

Πριν την εκτέλεση της δήλωσης COMMIT οι μεταβολές που έχουν γίνει από το προηγούμενο COMMIT και έπειτα είναι ορατές μόνο στον χρήστη που τις πραγματοποίησε. Αφού εκτελεσθεί η δήλωση COMMIT οι μεταβολές αυτές έχουν οριστικοποιηθεί και γίνονται ορατές σε όλους τους χρήστες. Η δήλωση COMMIT συντάσσεται ως εξής:

```
COMMIT  [WORK];
```

Ο όρος WORK είναι προαιρετικός και δε συνεπάγεται κάποιο ειδικό αποτέλεσμα.

Μετά από μία πτώση του συστήματος μόνο οι μεταβολές των δεδομένων για τις οποίες έχει εκτελεσθεί δήλωση COMMIT, έχουν οριστικοποιηθεί.

Με την εντολή

```
SET AUTOCOMMIT ON;
```

κάθε μεταβολή των δεδομένων από μία SQL δήλωση οριστικοποιείται άμεσα. Αρχικά στην Oracle το AUTOCOMMIT τίθεται από το σύστημα σε κατάσταση OFF ενώ στην περίπτωση του προϊόντος MySQL τίθεται από το σύστημα σε κατάσταση ON.

Προσοχή! Η εντολή SET AUTOCOMMIT ON; είναι εντολή της συνιστώσας Oracle SQL*PLUS. Στο προϊόν MySQL η εντολή είναι:

```
SET AUTOCOMMIT=0;
```

Οι παρακάτω δηλώσεις πραγματοποιούν αυτόματα COMMIT:

```
ALTER, AUDIT, CREATE, DISCONNECT, DROP, EXIT, GRANT, NOAUDIT, QUIT, REVOKE.
```

7.3.2 ROLLBACK

Πριν την εκτέλεση της δήλωσης COMMIT οι μεταβολές δεδομένων που έχουν γίνει μπορούν να ακυρωθούν. Έτσι η βάση δεδομένων παραμένει στην κατάσταση που ήταν πριν γίνουν οι μεταβολές. Αυτή η διαδικασία ακύρωσης πραγματοποιείται με τη δήλωση

```
ROLLBACK [WORK];
```

Η δήλωση ROLLBACK εκτελείται αυτόματα για όλους τους χρήστες όταν το σύστημα επανέρχεται από μία πτώση. Ακολουθούν θέματα-παραδείγματα που εντάσσονται σε CASE STUDY.

7.4 Περιήγηση. Διαχείριση της ασφάλειας ενός συστήματος βάσης δεδομένων με τη χρήση της υπογλώσσας ελέγχου δεδομένων της ORACLE SQL. Ο ρόλος του Διαχειριστή Βάσης Δεδομένων

Έστω ότι έχουμε υλοποιήσει ένα απλό σύστημα βάσης δεδομένων διαχείρισης στοιχείων υπαλλήλων και των τμημάτων στα οποία ανήκουν. Υποτίθεται ότι έχουμε δύο βασικούς πίνακες τον πίνακα με τα στοιχεία των υπαλλήλων (EMPLOYEE) και τον πίνακα με τα στοιχεία των τμημάτων (DEPARTMENT). Επειδή η διεκπεραίωση του θέματος θα γίνει με παραδείγματα σε περιβάλλον SQL*PLUS της ORACLE κρίνεται σκόπιμο να αναφερθούμε σύντομα σε ρόλους κάποιων χρηστών του συστήματος διαχείρισης της βάσης δεδομένων:

- Ο παντοδύναμος χρήστης SYS. Πρώτη σας μέριμνα με την εγκατάσταση του προϊόντος η αλλαγή του συνθηματικού.
- Ο δεύτερος πανίσχυρος χρήστης SYSTEM. Και αυτός έχει όλα τα δικαιώματα, δηλαδή δικαιώματα CONNECT, RESOURCE, DBA. Ο χρήστης αυτός είναι διαχειριστής της βάσης δεδομένων (DBA). Επομένως, πρώτη σας εργασία μετά την εγκατάσταση του προϊόντος η αλλαγή του συνθηματικού του.
- Ο ισχυρός χρήστης SCOTT με συνθηματικό TIGER που είναι και ο δημιουργός και κάτοχος πινάκων με τους οποίους μπορούμε να πειραματιστούμε όταν ξεκινάμε να μαθαίνουμε το προϊόν της ORACLE. Προσοχή! Αν και ο αρχάριος συνήθως χρησιμοποιεί τον κωδικό αυτού του χρήστη για να μάθει τα βασικά του προϊόντος και νομίζει ότι ο SCOTT είναι ένα είδος διαχειριστή της απλοποιημένης βάσης με την οποία εργάζεται στην πραγματικότητα ο SCOTT έχει δικαιώματα μόνο CONNECT, RESOURCE! Η επίδειξη (demo) του scott με συνθηματικό tiger δεν εγκαθίσταται αυτόματα όπως συνέβαινε με παλαιότερες εκδόσεις της Oracle. Για να εγκαταστήσετε το σχήμα SCOTT στη βάση δεδομένων σας, εκτελέστε ένα από τα ακόλουθα σενάρια από έναν λογαριασμό DBA:

```
SQL> @ ?/rdbms/admin/scott.sql
SQL> @ ?/sqlplus/demo/demobld.sql
SQL> @ ?/rdbms/admin/utlsampl.sql
```

Δείτε και το παρακάτω απόσπασμα: “Scott is a database user used for demonstration purposes containing the famous EMP, DEPT, BONUS and SALGRADE tables. According to legend, this account was named after Bruce Scott (co-author and co-architect of Oracle v1 to v3) and the password was the name of his daughter’s cat, Tiger”.

https://www.orafaq.com/wiki/SCOTT

Ο ενδιαφερόμενος αναγνώστης μπορεί να μελετήσει τη διαχείριση βάσης της Oracle στο εγχειρίδιο,

Bert Rich, Roopesh Ashok Kumar (2020) Oracle Database 2 Day DBA, 18c, E83770-04, Oracle and/or its affiliates. 2 Day DBA (oracle.com)

Στη συνέχεια όταν θέλουμε να αναφερθούμε στο όνομα του χρήστη και το συνθηματικό του θα το γράφουμε USERID/PASSWORD , π.χ. SCOTT/TIGER

Εκτός από τους χρήστες αυτούς μπορούμε να κατασκευάσουμε και άλλους με οποιαδήποτε δικαιώματα χρησιμοποιώντας τη δήλωση GRANT. Έτσι αν συνδεθούμε στη βάση σαν SYSTEM/my_password, δηλαδή σαν DBA, μπορούμε να φτιάξουμε και άλλους χρήστες ακόμη και χρήστες με δικαιώματα DBA. Σαν παράδειγμα μπορούμε να κατασκευάσουμε το DBA codd/ted.

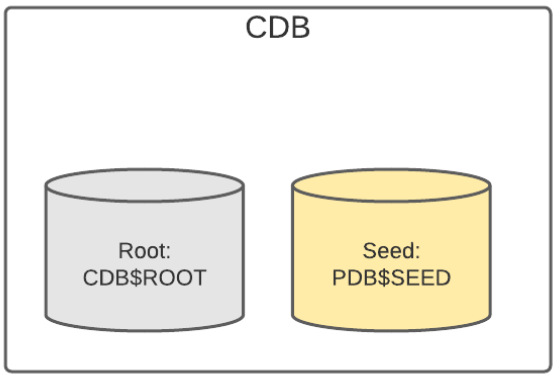
Προσοχή! Το όνομα και το επώνυμο είναι τελείως ακατάλληλα για USERID και PASSWORD, για προφανείς λόγους, και αποτελούν μεγάλο κίνδυνο για την ασφάλεια του συστήματος**.**

Oracle Multitenant Architecture

Ο ρόλος του DBA περιλαμβάνει πάρα πολλά καθήκοντα πέρα από την εκχώρηση και την αφαίρεση δικαιωμάτων χρηστών, τη διασφάλιση της ακεραιότητας και της ασφαλούς χρήσης της βάσης δεδομένων, τη λήψη εφεδρικών αντιγράφων της βάσης (back up), την ανάκαμψη της βάσης (recovery) μετά από προβλήματα. Για παράδειγμα, η αρχιτεκτονική των συστημάτων βάσεων δεδομένων αλλάζει, υπάρχουν νέες εκδόσεις λογισμικού (release) κ.λπ., και όλα αυτά προσθέτουν νέες αρμοδιότητες και υποχρεώσεις στον DBA. Παραθέτουμε κάποια στοιχεία για έννοιες και ορολογία της νεότερης αρχιτεκτονικής του προϊόντος της Oracle:

- Βάση δεδομένων με δυνατότητα προσθήκης (Pluggable Database, PDB): «Η βάση δεδομένων με δυνατότητα προσθήκης (PDB) είναι μια συλλογή σχημάτων και αντικειμένων που λειτουργούν σαν μια «κανονική» βάση δεδομένων για εφαρμογές και IDE» (“A PDB is a collection of schemas and objects that act like a “regular” database to applications and IDEs”).
- Κοντέινερ βάσης δεδομένων (Container Database, CDB): «Η βάση δεδομένων Oracle μπορεί να λειτουργήσει ως κοντέινερ «πολυμίσθωσης» βάσης δεδομένων. Αυτό το κοντέινερ (CDB) μπορεί να μην περιλαμβάνει βάση δεδομένων ή να περιλαμβάνει μία ή περισσότερες βάσεις δεδομένων με δυνατότητα προσθήκης (PDB) (“The Oracle database can function as a “multitenant container database”, otherwise known as a CDB. This CDB can include zero or more “pluggable databases”, or PDBs”).

Ένα κοντέινερ Oracle CDB μπορεί να περιέχει πολλά κοντέινερ. Ένα κοντέινερ μπορεί να είναι είτε PDB είτε ρίζα. Στην Εικόνα 7.1 βλέπουμε ότι «Η βάση δεδομένων περιέχει το CDB. Μέσα στο CDB υπάρχουν δύο κοντέινερ: α) κοντέινερ Root (που ονομάζεται CDB$ROOT), και β) Seed PDB (που ονομάζεται PDB$SEED), το οποίο είναι ένα πρότυπο που παρέχεται από το σύστημα και το οποίο το CDB μπορεί να χρησιμοποιήσει για τη δημιουργία νέων PDB» (” The database contains the CDB. Inside the CDB are two containers: Root, named CDB$ROOT. Seed PDB, named PDB$SEED… The seed PDB is a system-supplied template that the CDB can use to create new PDBs”)



Εικόνα 7.1 Η βάση δεδομένων περιλαμβάνει CDB που έχει δύο κοντέινερ (containers): Root με όνομα CDB$ROOT και Seed PDB με όνομα PDB$SEED.(Oracle, Introduction to the Multitenant Architecture)

Οι ενδιαφερόμενοι αναγνώστες μπορούν να μελετήσουν και να εξοικειωθούν με την αρχιτεκτονική πολυμίσθωσης της Oracle στο εγχειρίδιο:

Oracle E96136-16, Oracle® Multitenant-Administrator’s Guide, 19c, June 2022

Oracle Multitenant Administrator’s Guide, 19c

Ειδικότερα μπορούν να μελετήσουν την ενότητα Introduction to the Multitenant Architecture

Introduction to the Multitenant Architecture (oracle.com)

7.4.1 Πώς οριστικοποιούμε/ακυρώνουμε μεταβολές στα στοιχεία της βάσης δεδομένων

Ακολουθούν θέματα με τα οποία ξεκαθαρίζουμε τις έννοιες της οριστικοποίησης (επικύρωσης) των μεταβολών του περιεχομένου πινάκων και της ακύρωσης των μεταβολών.

Θέμα 1

Αυτόματη ενημέρωση της βάσης από την ORACLE μετά από κάθε μεταβολή.

```
SET AUTOCOMMIT IMMEDIATE
```

Θέμα 2

Πώς διακόπτουμε την αυτόματη ενημέρωση

```
SET AUTOCOMMIT OFF
```

Τώρα οι άλλοι χρήστες δεν βλέπουν τις μεταβολές που επιφέρεις στα στοιχεία με δηλώσεις INSERT, UPDATE, DELETE.

Με τη δήλωση COMMIT WORK οι αλλαγές είναι ορατές στους εξουσιοδοτημένους χρήστες.

Θέμα 3

Πώς ακυρώνεις τις μεταβολές.

Ενώ είσαι σε AUTOCOMMIT OFF (και μόνο τότε) μπορείς να ακυρώσεις τις μεταβολές με τη δήλωση:

```
ROLLBACK WORK
```

Προσοχή! Δεν μπορείς να ακυρώσεις μεταβολές αν δώσεις προηγούμενα δήλωση COMMIT WORK

Σημείωση

Όταν ολοκληρώσεις την εργασία σου και βγεις με EXIT από το περιβάλλον SQL*PLUS, η ORACLE αυτόματα εκτελεί τη δήλωση COMMIT WORK και οριστικοποιεί τις μεταβολές.

Σε περίπτωση βλάβης (program failure, system failure) η ORACLE αυτόματα εκτελεί δήλωση ROLLBACK WORK (Automatic Recovery on System Failure).

7.4.2 Πώς διαχειριζόμαστε την ασφάλεια του συστήματος βάσης δεδομένων

Έστω ο ισχυρός χρήστης codd (κάτι σαν το χρήστη scott) που έχει δικαιώματα:

```
CONNECT  - RESOURCE
```

δηλαδή μπορεί να συνδεθεί (log on), να δημιουργήσει συνώνυμα πινάκων, πίνακες, δείκτες, συστάδες (clusters), να εισάγει ή να τροποποιήσει στοιχεία, να διαγράψει και να αναζητήσει στοιχεία και να εκχωρήσει δικαιώματα εργασίας στους πίνακες που δημιουργεί.

Σύνδεση του χρήστη codd που έχει συνθηματικό ted

```
CONNECT codd/ted;
```

Εκχώρησε δικαιώματα SELECT στο χρήστη ADAMS για το πίνακα DEPT.

```
GRANT SELECT
ON DEPT
TO adams;
```

Δικαιώματα στους adams, jones για INSERT,UPDATE.

```
GRANT INSERT, UPDATE
ON DEPT
TO adams, jones;
```

Ο adams έχει δικαίωμα μόνο για CONNECT, άρα μπορεί να συνδεθεί και να εργαστεί με τους πίνακες για τους οποίους του δόθηκε εξουσιοδότηση, αν έχει δικαίωμα CREATE VIEW μπορεί να δημιουργήσει όψεις και συνώνυμα αλλά δεν μπορεί να δημιουργήσει πίνακες, δείκτες και συστάδες.

Σύνδεση σαν adams και επιλογή δεδομένων.

```
 CONNECT adams/jim
 SELECT *
 FROM codd.DEPT;
```

Δημιουργία του συνώνυμου DEPART για τον πίνακα codd.DEPT.

Ο χρήστης codd δίνει συνώνυμο depart για να διευκολύνει τον adams.

```
CREATE SYNONYM DEPART
FOR codd.DEPT;
Τώρα ο adams επιλέγει δεδομένα ως εξής:
 SELECT *
 FROM DEPART;
```

Ο adams δεν μπορεί να διαγράψει! Δηλαδή απαγορεύεται η δήλωση:

```
  DELETE FROM codd.DEPT WHERE DEPTNO =10;
```

Ο codd μπορεί βέβαια να του δώσει όλα τα δικαιώματα:

```
CONNECT codd/ted;
GRANT ALL
ON DEPT
  TO adams;
```

7.4.3 Πώς χρησιμοποιούμε τις όψεις σαν μηχανισμό εξασφάλισης της ασφάλειας της εφαρμογής και του συστήματος βάσης δεδομένων. Παραδείγματα

Θέμα 1

Δημιούργησε όψη του πίνακα EMP που να παραλείπει τα χαρακτηριστικά SAL, COMM έτσι ώστε να δίνει σε κάποιους χρήστες της βάσης τη ψευδαίσθηση ότι η βάση (ο πίνακας στο παράδειγμά μας) είναι διαφορετική από την πραγματική .

```
 CREATE VIEW EMPLOYEE AS
 SELECT EMPNO, ENAME, JOB, MGR, DEPTNO
 FROM   EMP;
```

Θέμα 2

Δημιούργησε όψη με τους υπάλληλους της Διεύθυνσης 20 περιορίζοντας κάποιους χρήστες μόνο στα στοιχεία αυτά.

```
 CREATE VIEW EMP20 AS
 SELECT *
```

```
  FROM   EMP
  WHERE  DEPTNO = 20;
```

Θέμα 3

Πώς ένας υπάλληλος βλέπει μόνο τα στοιχεία που τον αφορούν.

Δημιουργούμε την αντίστοιχη όψη:

```
  CREATE VIEW MYSELF AS
  SELECT *
  FROM   EMP
  WHERE  ENAME = USER;
```

και δίνουμε δικαίωμα αναζήτησης σε όλους (GRANT ... TO PUBLIC).

```
 GRANT SELECT
 ON MYSELF
 TO PUBLIC;
```

τότε ο οποιοσδήποτε (π.χ. ο adams) μπορεί να βλέπει τα στοιχεία του.

```
   CONNECT adams/jim
   SELECT * FROM codd.MYSELF;
```

Θέμα 4

Πώς ο επικεφαλής θα βλέπει μόνο τα στοιχεία των υπαλλήλων του.

Συνδέεται ο codd:

```
connect codd/ted
```

Δημιουργεί την όψη:

```
CREATE VIEW MY_EMPS AS
 SELECT *
 FROM   EMP
 WHERE  DEPTNO IN
   (SELECT DEPTNO
    FROM   EMP
    WHERE  JOB = 'MANAGER'
    AND ENAME = USER);
```

Τι βλέπει ο codd αν δώσει την παρακάτω δήλωση;

```
SELECT * FROM MY_EMPS;
GRANT SELECT ON MY_EMPS TO PUBLIC;
```

Θέμα 5

Πώς ο επικεφαλής βλέπει μόνο τους υπαλλήλους του που κερδίζουν λιγότερα από αυτόν.

Συνδέεται ο codd:

```
connect codd/ted
```

Δημιουργεί την όψη:

```
CREATE VIEW WORKERS (NAME,JOB,SAL,COMM,DEPTNO) AS
SELECT WORKER.ENAME,WORKER.JOB,WORKER.SAL,WORKER.COMM,
   WORKER.DEPTNO
FROM EMP WORKER,EMP MGR
WHERE WORKER.DEPTNO = MGR.DEPTNO
  AND (MGR.JOB = 'MANAGER'  AND MGR.SAL >= WORKER.SAL)
  AND MGR.ENAME=USER;
```

Εκχωρεί δικαίωμα αναζήτησης:

```
GRANT SELECT ON WORKERS TO PUBLIC;
```

Έτσι ο CLARKE συνδέεται:

```
CONNECT CLARKE/ARTHUR
```

και βλέπει αυτά που πρέπει να δει:

```
SELECT * FROM codd.WORKERS;
```

Θέμα 6

Δώσε στον Adams το δικαίωμα να δίνει δικαίωμα SELECT στο πίνακα DEPT.

```
CONNECT codd/ted
GRANT SELECT
ON DEPT
TO ADAMS
WITH GRANT OPTION;
```

Θέμα 7

Αφαίρεσε από τον ADAMS το δικαίωμα εισαγωγής στοιχείων (INSERT) στον πίνακα DEPT.

```
REVOKE INSERT
ON DEPT
FROM ADAMS;
```

7.4.4 Ένας διάλογος (session) με το σύστημα. Δημιουργία χρήστη και εκχώρηση δικαιωμάτων

Οι πίνακες της βάσης δεδομένων στην Oracle δημιουργούνται σε λογική περιοχή η οποία ονομάζεται tablespace. Συνδεόμαστε ως SYSTEM και βλέπουμε τις στήλες για τα tablespaces.

```
SQL> describe dba_tablespaces;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 TABLESPACE_NAME                           NOT NULL VARCHAR2(30)
 BLOCK_SIZE                                NOT NULL NUMBER
 INITIAL_EXTENT                                     NUMBER
 NEXT_EXTENT                                        NUMBER
 MIN_EXTENTS                               NOT NULL NUMBER
 MAX_EXTENTS                                        NUMBER
 MAX_SIZE                                           NUMBER
 PCT_INCREASE                                       NUMBER
 MIN_EXTLEN                                         NUMBER
 STATUS                                             VARCHAR2(9)
 CONTENTS                                           VARCHAR2(21)
 LOGGING                                            VARCHAR2(9)
 FORCE_LOGGING                                      VARCHAR2(3)
 EXTENT_MANAGEMENT                                  VARCHAR2(10)
 ALLOCATION_TYPE                                    VARCHAR2(9)
 PLUGGED_IN                                         VARCHAR2(3)
 SEGMENT_SPACE_MANAGEMENT                           VARCHAR2(6)
 DEF_TAB_COMPRESSION                                VARCHAR2(8)
 RETENTION                                          VARCHAR2(11)
 BIGFILE                                            VARCHAR2(3)
 PREDICATE_EVALUATION                               VARCHAR2(7)
 ENCRYPTED                                          VARCHAR2(3)
```

Εικόνα 7.2 Περιγραφή του tablespace

Βλέπουμε τα ονόματά τους.

```
SELECT TABLESPACE_NAME FROM dba_tablespaces;
```

```
SQL> SELECT TABLESPACE_NAME FROM dba_tablespaces;

TABLESPACE_NAME
------------------------------
SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS
```

Εικόνα 7.3 Oracle express tablespaces

Θα δημιουργήσουμε ένα χρήστη και θα του δώσουμε αρχικά τα δικαιώματα CONNECT, RESOURCE και αργότερα και το δικαίωμα DBA.

```
/* create user c##scott */
CREATE USER c##scott IDENTIFIED BY tiger;
```

Βλέπουμε τις στήλες για τους χρήστες

```
SQL> describe dba_users;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USERNAME                                  NOT NULL VARCHAR2(128)
 USER_ID                                   NOT NULL NUMBER
 PASSWORD                                           VARCHAR2(4000)
 ACCOUNT_STATUS                            NOT NULL VARCHAR2(32)
 LOCK_DATE                                          DATE
 EXPIRY_DATE                                        DATE
 DEFAULT_TABLESPACE                        NOT NULL VARCHAR2(30)
 TEMPORARY_TABLESPACE                      NOT NULL VARCHAR2(30)
 LOCAL_TEMP_TABLESPACE                              VARCHAR2(30)
 CREATED                                   NOT NULL DATE
 PROFILE                                   NOT NULL VARCHAR2(128)
 INITIAL_RSRC_CONSUMER_GROUP                        VARCHAR2(128)
 EXTERNAL_NAME                                      VARCHAR2(4000)
 PASSWORD_VERSIONS                                  VARCHAR2(17)
 EDITIONS_ENABLED                                   VARCHAR2(1)
 AUTHENTICATION_TYPE                                VARCHAR2(8)
 PROXY_ONLY_CONNECT                                 VARCHAR2(1)
 COMMON                                             VARCHAR2(3)
 LAST_LOGIN                                         TIMESTAMP(9) WITH TIME ZONE
 ORACLE_MAINTAINED                                  VARCHAR2(1)
 INHERITED                                          VARCHAR2(3)
 DEFAULT_COLLATION                                  VARCHAR2(100)
 IMPLICIT                                           VARCHAR2(3)
 ALL_SHARD                                          VARCHAR2(3)
 EXTERNAL_SHARD                                     VARCHAR2(3)
 PASSWORD_CHANGE_DATE                               DATE
 MANDATORY_PROFILE_VIOLATION                        VARCHAR2(3)
```

Εικόνα 7.4 Περιγραφή των χρηστών

Βλέπουμε τους χρήστες

```
FORMAT USERNAME FORMAT A10
SELECT USERNAME
FROM DBA_USERS;
```

```
USERNAME
--------------------------
SYSBACKUP
REMOTE_SCHEDULER_AGENT
GSMUSER
SYSRAC
GSMROOTUSER
SI_INFORMTN_SCHEMA
DIP
ORDPLUGINS
SYSKM
DGPDB_INT
ORDDATA

USERNAME
--------------------------
ORACLE_OCM
SYS$UMF
C##SCOTT
SYSDG
ORDSYS

38 rows selected.
```

Εικόνα 7.5 Χρήστες

```
SQL> GRANT CONNECT TO C##SCOTT;

Grant succeeded.

SQL> GRANT RESOURCE TO C##SCOTT;

Grant succeeded.

SQL> CONNECT C##SCOTT/tiger;
Connected.
```

Εικόνα 7.6 Εκχώρηση δικαιωμάτων CONNECT, RESOURCE

```
DROP TABLE EMPLOYEE;

DROP TABLE DEPARTMENT;

CREATE TABLE DEPARTMENT(DEPTNO NUMBER(2) NOT NULL,
  DNAME CHAR(14),
  PRIMARY KEY(DEPTNO)) TABLESPACE USERS;

CREATE TABLE EMPLOYEE(EMPNO NUMBER(4) NOT NULL,
   ENAME CHAR(10), JOB CHAR(9), SAL NUMBER(7,2),
   DEPTNO NUMBER(2),
   PRIMARY KEY(EMPNO),
   FOREIGN KEY(DEPTNO) REFERENCES DEPARTMENT(DEPTNO))

TABLESPACE USERS;

DESCRIBE EMPLOYEE;

DESCRIBE DEPARTMENT;
```

```
SQL> DESCRIBE EMPLOYEE;
Name                                      Null?    Type
----------------------------------------- -------- ------------
EMPNO                                     NOT NULL NUMBER(4)
ENAME                                              CHAR(10)
JOB                                                CHAR(9)
SAL                                                NUMBER(7,2)
DEPTNO                                             NUMBER(2)

SQL> DESCRIBE DEPARTMENT;
Name                                      Null?    Type
----------------------------------------- -------- ------------
DEPTNO                                    NOT NULL NUMBER(2)
DNAME                                              CHAR(14)
```

Εικόνα 7.7 Στήλες πινάκων

```
GRANT DBA TO c##scott;

/* data entry */
INSERT INTO DEPARTMENT(DEPTNO, DNAME)
  VALUES (10, 'ACCOUNTING');
INSERT INTO DEPARTMENT(DEPTNO, DNAME)
  VALUES (20, 'RESEARCH');
INSERT INTO DEPARTMENT(DEPTNO, DNAME)
  VALUES (30, 'SALES');
INSERT INTO DEPARTMENT(DEPTNO, DNAME)
  VALUES (40, 'OPERATIONS');
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
   VALUES (7369, 'SMITH', 'CLERK', 800, 20);
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
   VALUES (7499, 'ALLEN', 'SALESMAN', 1600, 30);
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
   VALUES (7521, 'WARD', 'SALESMAN', 1250, 30);
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
   VALUES (7566, 'JONES', 'MANAGER', 2975, 20);
```

```
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
   VALUES (7654, 'MARTIN', 'SALESMAN', 1250, 30);
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
   VALUES (7698, 'BLAKE', 'MANAGER', 2850, 30);
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
   VALUES (7782, 'CLARK', 'MANAGER', 2450, 10);
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
VALUES (7788, 'SCOTT', 'ANALYST', 3000, 20);
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
VALUES (7839, 'KING', 'PRESIDENT', 5000, 10);
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
VALUES (7844, 'TURNER', 'SALESMAN', 1500, 30);
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
VALUES (7876, 'ADAMS', 'CLERK', 1100, 20);
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
VALUES (7900, 'JAMES', 'CLERK', 950, 30);
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
VALUES (7902, 'FORD', 'ANALYST', 3000, 20);
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
VALUES (7934, 'MILLER', 'CLERK', 1300, 10);
INSERT INTO EMPLOYEE(EMPNO, ENAME, JOB, SAL, DEPTNO)
VALUES (7999, 'BATES', 'ANALYST', 1300, NULL);

SELECT * FROM DEPARTMENT;
SELECT * FROM EMPLOYEE;
```

```
SQL> SELECT * FROM DEPARTMENT;

    DEPTNO DNAME
---------- --------------
        10 ACCOUNTING
        20 RESEARCH
        30 SALES
        40 OPERATIONS

SQL> SELECT * FROM EMPLOYEE;

     EMPNO ENAME      JOB              SAL     DEPTNO
---------- ---------- --------- ---------- ----------
      7369 SMITH      CLERK            800         20
      7499 ALLEN      SALESMAN        1600         30
      7521 WARD       SALESMAN        1250         30
      7566 JONES      MANAGER         2975         20
      7654 MARTIN     SALESMAN        1250         30
      7698 BLAKE      MANAGER         2850         30
      7782 CLARK      MANAGER         2450         10
      7788 SCOTT      ANALYST         3000         20
      7839 KING       PRESIDENT       5000         10
      7844 TURNER     SALESMAN        1500         30
      7876 ADAMS      CLERK           1100         20

     EMPNO ENAME      JOB              SAL     DEPTNO
---------- ---------- --------- ---------- ----------
      7900 JAMES      CLERK            950         30
      7902 FORD       ANALYST         3000         20
      7934 MILLER     CLERK           1300         10
      7999 BATES      ANALYST         1300

15 rows selected.
```

Εικόνα 7.8 Δεδομένα πινάκων

Ο ιδιοκτήτης του πίνακα (c##scott/tiger) EMPLOYEE δίνει τα δικαιώματα χρήσης για τον πίνακά του.

```
SQL> connect c##scott/tiger
Connected.
```

```
SQL> grant all on emp
  2  to system
  3  with grant option;
Grant succeeded.
SQL> CONNECT SYSTEM/MANAGER
Connected.
SQL> SELECT * FROM SCOTT.EMP;
```

Βλέπει όλα τα στοιχεία του πίνακα.

```
15 rows selected.
```

```
SQL> CONNECT C##SCOTT/tiger;
Connected.
SQL> DESCRIBE TAB;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 TNAME                                     NOT NULL VARCHAR2(128)
 TABTYPE                                            VARCHAR2(13)
 CLUSTERID                                          NUMBER

SQL> SELECT TNAME FROM TAB;

TNAME
--------------------------------------------------------------------------------
EMPLOYEE
BIN$wqYlv4nsS6mw0rNLRGitWA==$0
BIN$X21LrdFKQ82AkR26AEzrlw==$0
```

Εικόνα 7.9 Πίνακες που βλέπει ο χρήστης

Ακολουθούν κάποιες ενέργειες που είδαμε και σε προηγούμενες παραγράφους.

```
SQL> CREATE VIEW MYSELF
  2  AS SELECT * FROM EMPLOYEE
  3  WHERE ENAME = USER;
View created.
SQL> GRANT CONNECT TO SMITH IDENTIFIED BY SMITH;
Grant succeeded.
SQL> CONNECT SMITH/SMITH
Connected.
```

Ο Smith συνδέθηκε αλλά αυτό δε σημαίνει ότι μπορεί να κάνει και αναζητήσεις.

```
SQL> SELECT * FROM MYSELF;
SELECT * FROM MYSELF
 *

SQL> SELECT * FROM MYSELF;
SELECT * FROM MYSELF
 *

ERROR at line 1:
ORA-00942: table or view does not exist
SQL> CONNECT c##scott/tiger;

Connected.

SQL> GRANT SELECT ON MYSELF TO c##smith;
Grant succeeded.
```

```
SQL> CONNECT SMITH/SMITH
Connected.
```

Τώρα μπορεί να δει τα στοιχεία του (εικόνα 7.10)

```
SQL> SELECT * FROM SYSTEM.MYSELF;
 EMPNO ENAME  JOB MGR HIREDATE  SAL  COMM   DEPTNO

---------- ---------- --------- ---------- --------- ---------- ---------- ----------

7369 SMITH  CLERK  7902 17-DEC-20   800 20
```

Εικόνα 7.10 Ο χρήστης βλέπει μόνο τα στοιχεία του «διαβάζοντας» την όψη

7.4.5 Πώς συμβάλλει στην ασφάλεια του συστήματος ο DBA. Μία περιγραφή σημαντικών καθηκόντων του μέσα από παραδείγματα.

Συνδέσου σαν DBA.

```
CONNECT SYSTEM/(password)
```

Θυμήσου ότι πρέπει να αλλάξεις το αρχικό συνθηματικό MANAGER!

Δημιούργησε τον χρήστη FORD με συνθηματικό CAR.

```
GRANT CONNECT TO FORD IDENTIFIED BY CAR;
```

Πώς θα συνδέεται ο Ford .

```
CONNECT FORD/CAR
```

Πως ο Ford θα μπορεί να δημιουργεί πίνακες.

Μπαίνουμε σαν DBA

```
CONNECT SYSTEM/(password)
```

και του εκχωρούμε το δικαίωμα:

```
GRANT RESOURCE TO FORD;
```

Έτσι ο FORD συνδέεται

```
CONNECT FORD/CAR
```

και δημιουργεί πίνακα:

```
CREATE TABLE PARTS
   (PARTNO NUMBER,
    DESCRIPTION CHAR(20),
    QUANTITY NUMBER);
```

και εκχωρεί δικαιώματα αναζήτησης σε όλους

```
   GRANT SELECT ON PARTS TO PUBLIC;
```

7.4.5.1 Δημιούργησε νέο DBA.

Συνδέσου σαν DBA

```
CONNECT SYSTEM/(password)
```

και δημιούργησε τον νέο DBA (KING/ALEXANDER).

```
GRANT DBA TO KING IDENTIFIED BY ALEXANDER;
```

7.4.5.2 Πώς δημιουργούμε ευρετήρια (δείκτες, index)

Δημιούργησε δείκτη βασιζόμενο στη στήλη ENAME του πίνακα EMP.

```
CREATE INDEX EMP_ENAME ON EMP(ENAME);
```

Δημιούργησε μοναδικό δείκτη βασιζόμενο στη στήλη EMPNO του πίνακα EMP.

```
CREATE UNIQUE INDEX EMP_EMPNO ON EMP(EMPNO);
```

Κατάργησε το δείκτη EMP_EMPNO .

```
DROP INDEX EMP_EMPNO;
```

7.4.5.3 Πώς δημιουργούμε συστάδες πινάκων

Θέμα 1

Έστω ότι θέλεις να δημιουργήσεις μια συστάδα που να περιλαμβάνει τους πίνακες EMP, DEPT (βλέπε στήλη DEPTNO στους δύο πίνακες) ώστε να επιταχύνεις την αναζήτηση όταν χρησιμοποιείς δηλώσεις SELECT που περιλαμβάνουν JOIN των πινάκων αυτών. Η δημιουργία συστάδας διασφαλίζει ότι τα δεδομένα των πινάκων αποθηκεύονται σε φυσικό επίπεδο (π.χ., στους σκληρούς δίσκους) «πολύ κοντά».

```
CREATE CLUSTER DEPT_EMP
(DEPTNO NUMBER);
```

Πρόσθεσε τον πίνακα DEPT στη συστάδα.

```
CREATE TABLE T1 CLUSTER DEPT_EMP(DEPTNO) SELECT * FROM DEPT;

DROP TABLE DEPT;

RENAME T1 TO DEPT;
```

Πρόσθεσε τον πίνακα EMP στη συστάδα.

```
CREATE TABLE T1 CLUSTER DEPT_EMP(DEPTNO) SELECT * FROM EMP;

DROP TABLE EMP;

RENAME T1 TO EMP;
```

Μπορείς να διαπιστώσεις ότι η δημιουργία της συστάδας δε θα αλλάξει τον τρόπο που γράφεις τις αναζητήσεις σου. Για παράδειγμα, δείξε τους υπάλληλους της διεύθυνσης 10. Βέβαια ο χρόνος αναζήτησης πρέπει να έχει μεταβληθεί (βελτιωθεί) .

```
SELECT DEPT.*,EMP.*
  FROM DEPT,EMP
  WHERE DEPT.DEPTNO = EMP.DEPTNO
  AND DEPT.DEPTNO = 10;
```

7.4.6 Πώς βλέπεις τους πίνακες και άλλα στοιχεία του λεξικού δεδομένων

Υπάρχει ένας βασικός κανόνας:

Ο DBA βλέπει τα πάντα και οι υπόλοιποι ότι τους επιτρέπει ο DBA.

Έστω ότι συνδέεσαι σαν SCOTT/TIGER.

Δες (εικόνα 7.11) τους πίνακες που δημιούργησες (εργαζόμενος σαν SCOTT/TIGER).

```
SELECT * FROM TAB;

  TNAME           TABTYPE     CLUSTERID
  ------------------------------------
  CARDS           TABLE
  CUSTOMER        TABLE
  DEPT            TABLE
  DEPT_EMP        CLUSTER
  DUMMY           TABLE
  EMP             TABLE
  FARES           TABLE
  FLIGHTS         TABLE
  IAPXTB          TABLE
  MASTER          TABLE
  ORDERLINES      TABLE
  ORDERS          TABLE
  ORDERSNOTES     VIEW
  ORDERS_LINES    CLUSTER
  PARTS           TABLE
  RESERVATIONS    TABLE

  16 records selected.
```

Εικόνα 7.11 Πίνακες που δημιούργησε ο χρήστης

Επειδή θα αλλάξεις συχνά ρόλο στη συνέχεια σημειώνουμε πως μπορείς να δεις ποιος χρήστης είσαι κάθε χρονική στιγμή.

```
SQL> SELECT USER FROM DUAL;
```

Συνδέσου σαν DBA και διαπίστωσε ότι τώρα βλέπεις και άλλους πίνακες από το λεξικό δεδομένων.

```
SQL> CONNECT SYSTEM/MANAGER
Connected.

SQL> SELECT * FROM TAB;
```

```
SQL> SELECT USER FROM DUAL;

USER
------------------------------------------------------------------------
SYSTEM

SQL> DESCRIBE TAB;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 TNAME                                     NOT NULL VARCHAR2(128)
 TABTYPE                                            VARCHAR2(13)
 CLUSTERID                                          NUMBER

SQL> SELECT TNAME FROM TAB;

TNAME
------------------------------------------------------------------------
LOGMNR_SESSION_EVOLVE$
LOGMNR_GLOBAL$
LOGMNR_GT_TAB_INCLUDE$
LOGMNR_GT_USER_INCLUDE$
LOGMNR_GT_XID_INCLUDE$
LOGMNR_PDB_INFO$
LOGMNR_DID$
LOGMNR_UID$
LOGMNRGGC_GTLO
LOGMNRGGC_GTCS
LOGMNRC_DBNAME_UID_MAP
```

Εικόνα 7.12 Πίνακες που βλέπει ο SYSTEM

```
SQL> DESCRIBE DBA_USERS;
```

```
SQL> DESCRIBE DBA_USERS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USERNAME                                  NOT NULL VARCHAR2(128)
 USER_ID                                   NOT NULL NUMBER
 PASSWORD                                           VARCHAR2(4000)
 ACCOUNT_STATUS                            NOT NULL VARCHAR2(32)
 LOCK_DATE                                          DATE
 EXPIRY_DATE                                        DATE
 DEFAULT_TABLESPACE                        NOT NULL VARCHAR2(30)
 TEMPORARY_TABLESPACE                      NOT NULL VARCHAR2(30)
 LOCAL_TEMP_TABLESPACE                              VARCHAR2(30)
 CREATED                                   NOT NULL DATE
 PROFILE                                   NOT NULL VARCHAR2(128)
 INITIAL_RSRC_CONSUMER_GROUP                        VARCHAR2(128)
 EXTERNAL_NAME                                      VARCHAR2(4000)
 PASSWORD_VERSIONS                                  VARCHAR2(17)
 EDITIONS_ENABLED                                   VARCHAR2(1)
 AUTHENTICATION_TYPE                                VARCHAR2(8)
 PROXY_ONLY_CONNECT                                 VARCHAR2(1)
 COMMON                                             VARCHAR2(3)
 LAST_LOGIN                                         TIMESTAMP(9) WITH TIME ZONE
 ORACLE_MAINTAINED                                  VARCHAR2(1)
 INHERITED                                          VARCHAR2(3)
 DEFAULT_COLLATION                                  VARCHAR2(100)
 IMPLICIT                                           VARCHAR2(3)
 ALL_SHARD                                          VARCHAR2(3)
 EXTERNAL_SHARD                                     VARCHAR2(3)
 PASSWORD_CHANGE_DATE                               DATE
 MANDATORY_PROFILE_VIOLATION                        VARCHAR2(3)
```

Εικόνα 7.13 Περιγραφή των χρηστών

```
SQL> SELECT USERNAME FROM DBA_USERS
USERNAME
------------------
SYS
SYSTEM
```

```
SQL> DESCRIBE DBA_ROLE_PRIVS;

SQL> SELECT * FROM DBA_ROLE_PRIVS WHERE GRANTEE = 'C##SCOTT';
```

```
SQL> DESCRIBE DBA_ROLE_PRIVS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 GRANTEE                                            VARCHAR2(128)
 GRANTED_ROLE                                       VARCHAR2(128)
 ADMIN_OPTION                                       VARCHAR2(3)
 DELEGATE_OPTION                                    VARCHAR2(3)
 DEFAULT_ROLE                                       VARCHAR2(3)
 COMMON                                             VARCHAR2(3)
 INHERITED                                          VARCHAR2(3)

SQL> SELECT GRANTEE, GRANTED_ROLE FROM DBA_ROLE_PRIVS WHERE GRANTEE = 'C##SCOTT';

GRANTEE
--------------------------------------------------------------------------------
GRANTED_ROLE
--------------------------------------------------------------------------------
C##SCOTT
CONNECT

C##SCOTT
DBA

C##SCOTT
RESOURCE
```

Εικόνα 7.14 Περιγραφή δικαιωμάτων του ρόλου DBA

```
SQL> DESCRIBE DBA_SYS_PRIVS;
```

```
SQL> DESCRIBE DBA_SYS_PRIVS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 GRANTEE                                            VARCHAR2(128)
 PRIVILEGE                                          VARCHAR2(40)
 ADMIN_OPTION                                       VARCHAR2(3)
 COMMON                                             VARCHAR2(3)
 INHERITED                                          VARCHAR2(3)
```

Εικόνα 7.15 Περιγραφή δικαιωμάτων του SYS

```
 SQL> SELECT * FROM DBA_SYS_PRIVS WHERE GRANTEE = 'C##SCOTT';
```

```
SQL> SELECT * FROM DBA_SYS_PRIVS WHERE GRANTEE = 'C##SCOTT';

GRANTEE  PRIVILEGE                                ADM COM INH
-------- ---------------------------------------- --- --- ---
C##SCOTT UNLIMITED TABLESPACE                     NO  NO  NO
```

Εικόνα 7.16 Δικαιώματα χρήστη C##SCOTT

```
SQL> DESCRIBE CATALOG;
```

```
SQL> DESCRIBE CATALOG;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 TNAME                                     NOT NULL VARCHAR2(128)
 CREATOR                                   NOT NULL VARCHAR2(128)
 TABLETYPE                                          VARCHAR2(8)
 REMARKS                                            VARCHAR2(4000)
```

Εικόνα 7.17 Περιγραφή καταλόγου

```
SQL> SELECT * FROM CATALOG WHERE TNAME='EMPLOYEE';
```

```
SQL> SELECT * FROM CATALOG WHERE TNAME='EMPLOYEE';

TNAME
--------------------------------------------------
CREATOR
--------------------------------------------------
TABLETYP
--------
REMARKS
--------------------------------------------------
EMPLOYEE
C##SCOTT
TABLE
```

Εικόνα 7.18 Περιγραφή καταλόγου για τον πίνακα EMPLOYEE

```
SQL> DESCRIBE SYSCATALOG;
SQL> DESCRIBE COL;
```

```
SQL> DESCRIBE SYSCATALOG;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 TNAME                                     NOT NULL VARCHAR2(128)
 CREATOR                                   NOT NULL VARCHAR2(128)
 TABLETYPE                                          VARCHAR2(8)
 REMARKS                                            VARCHAR2(4000)

SQL> DESCRIBE COL;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 TNAME                                     NOT NULL VARCHAR2(128)
 COLNO                                     NOT NULL NUMBER
 CNAME                                     NOT NULL VARCHAR2(128)
 COLTYPE                                            VARCHAR2(265)
 WIDTH                                     NOT NULL NUMBER
 SCALE                                              NUMBER
 PRECISION                                          NUMBER
 NULLS                                              VARCHAR2(19)
 DEFAULTVAL                                         LONG
 CHARACTER_SET_NAME                                 VARCHAR2(44)
```

Εικόνα 7.19 Περιγραφή SYSCATALOG

```
SQL> SELECT * FROM COL WHERE TNAME = 'EMPLOYEE';
```

```
SQL> DESCRIBE CATALOG;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 TNAME                                     NOT NULL VARCHAR2(128)
 CREATOR                                   NOT NULL VARCHAR2(128)
 TABLETYPE                                          VARCHAR2(8)
 REMARKS                                            VARCHAR2(4000)

SQL> SELECT CREATOR FROM CATALOG WHERE TNAME='EMPLOYEE';
```

Εικόνα 7.20 Ποιος δημιούργησε τον πίνακα EMPLOYEE

```
SQL> CONNECT SCOTT/TIGER
Connected.
COLUMN COLNO HEADING 'CNO';
COLUMN CNAME HEADING 'NAME'
COLUMN COLNO FORMAT 999;
COLUMN CNAME FORMAT A20;
```

```
SQL> SELECT COLNO,CNAME FROM COL WHERE TNAME='EMPLOYEE';
```

```
SQL> COLUMN COLNO HEADING 'CNO';
SQL> COLUMN CNAME HEADING 'NAME'
SQL> COLUMN COLNO FORMAT 999;
SQL> COLUMN CNAME FORMAT A20;
SQL>
SQL>
SQL> SELECT COLNO,CNAME FROM COL WHERE TNAME='EMPLOYEE';

 CNO NAME
---- --------------------
   1 EMPNO
   2 ENAME
   3 JOB
   4 SAL
   5 DEPTNO
```

Εικόνα 7.21 Στήλες πίνακα

7.5 Περιήγηση στην ασφάλεια συστημάτων βάσης δεδομένων στο προϊόν MySQL

Θα αρχίσουμε από τον ορισμό της βάσης δεδομένων και των πινάκων της.

7.5.1 Πώς ορίζουμε τη βάση δεδομένων και τους πίνακες. Δήλωση CREATE DATABASE. Δήλωση CREATE TABLE

```
DROP DATABASE IF EXISTS personnel;

CREATE DATABASE personnel;

USE personnel;

CREATE TABLE DEPT (DEPTNO INT(2) NOT NULL,
  DNAME CHAR(14),
  LOC CHAR(13));

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

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

CREATE TABLE ASSIGN(EMPNO INT(4) NOT NULL,
   PROJNO INT(3) NOT NULL,
   PTIME INT(3));
SHOW TABLES;
```

Στην εικόνα 7.22 βλέπουμε τους πίνακες της βάσης δεδομένων.

```
mysql> SHOW TABLES;
+---------------------+
| Tables_in_personnel |
+---------------------+
| assign              |
| dept                |
| emp                 |
| proj                |
+---------------------+
4 rows in set (0.00 sec)
```

Εικόνα 7.22 Οι πίνακες της βάσης δεδομένων.

Προσοχή! Στους παραπάνω ορισμούς δεν συμπεριλάβαμε τον κύρια και ξένα κλειδιά. Επιπλέον, μπορούμε να έχουμε ονόματα υπαλλήλων μόνο μέχρι 10 χαρακτήρες κ.λπ.

Ακολουθούν δηλώσεις SQL που τροποποιούν τους ορισμούς αυτούς.

7.5.2 Πώς τροποποιούμε ορισμό πίνακα – Δήλωση ALTER TABLE

Για να προσθέσουμε στο πίνακα EMP τη στήλη EMPNO σαν κύριο κλειδί:

```
ALTER TABLE DEPT ADD
   CONSTRAINT PK_DEPT_DEPTNO PRIMARY KEY(DEPTNO);

ALTER TABLE EMP ADD
   CONSTRAINT PK_EMP_EMPNO PRIMARY KEY(EMPNO);

ALTER TABLE EMP ADD
   CONSTRAINT FK_EMP_DEPTNO
FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO);

DESCRIBE DEPT;
DESCRIBE EMP;
```

```
mysql> DESCRIBE DEPT;
+--------+----------+------+-----+---------+-------+
| Field  | Type     | Null | Key | Default | Extra |
+--------+----------+------+-----+---------+-------+
| DEPTNO | int      | NO   | PRI | NULL    |       |
| DNAME  | char(14) | YES  |     | NULL    |       |
| LOC    | char(13) | YES  |     | NULL    |       |
+--------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> DESCRIBE EMP;
+----------+------------+------+-----+---------+-------+
| Field    | Type       | Null | Key | Default | Extra |
+----------+------------+------+-----+---------+-------+
| EMPNO    | int        | NO   | PRI | NULL    |       |
| ENAME    | char(10)   | YES  |     | NULL    |       |
| JOB      | char(9)    | YES  |     | NULL    |       |
| MGR      | int        | YES  |     | NULL    |       |
| HIREDATE | date       | YES  |     | NULL    |       |
| SAL      | float(7,2) | YES  |     | NULL    |       |
| COMM     | float(7,2) | YES  |     | NULL    |       |
| DEPTNO   | int        | YES  | MUL | NULL    |       |
+----------+------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

Εικόνα 7.23 Στήλες πινάκων

```
SHOW CREATE TABLE DEPT;
```

```
mysql> SHOW CREATE TABLE DEPT;
+-------+-----------------------------------------------------------------------
-------------------------------------------------------------------------------
| Table | Create Table

+-------+-----------------------------------------------------------------------
-------------------------------------------------------------------------------
| DEPT  | CREATE TABLE `dept` (
  `DEPTNO` int NOT NULL,
  `DNAME` char(14) DEFAULT NULL,
  `LOC` char(13) DEFAULT NULL,
  PRIMARY KEY (`DEPTNO`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-------+-----------------------------------------------------------------------
-------------------------------------------------------------------------------
1 row in set (0.01 sec)
```

Εικόνα 7.24 Βλέπουμε τη δήλωση CREATE TABLE DEPT.

```
SHOW CREATE TABLE EMP;
```

```
mysql> SHOW CREATE TABLE EMP;
+-------+-----------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------+
| Table | Create Table

                                                                          |
+-------+-----------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------+
| EMP   | CREATE TABLE `emp` (
  `EMPNO` int NOT NULL,
  `ENAME` char(10) DEFAULT NULL,
  `JOB` char(9) DEFAULT NULL,
  `MGR` int DEFAULT NULL,
  `HIREDATE` date DEFAULT NULL,
  `SAL` float(7,2) DEFAULT NULL,
  `COMM` float(7,2) DEFAULT NULL,
  `DEPTNO` int DEFAULT NULL,
  PRIMARY KEY (`EMPNO`),
  KEY `FK_EMP_DEPTNO` (`DEPTNO`),
  CONSTRAINT `FK_EMP_DEPTNO` FOREIGN KEY (`DEPTNO`) REFERENCES `dept` (`DEPTNO`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-------+-----------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Εικόνα 7.25 Βλέπουμε τη δήλωση CREATE TABLE EMP

7.5.3 Προσθήκη στον ορισμό πινάκων κύριων και ξένων κλειδιών χωρίς ρητή δήλωση constraint

Εναλλακτικά μπορούμε να ορίσουμε κύρια και ξένα κλειδιά χωρίς ρητή δήλωση constraint Διαγράψτε και ξαναδημιουργήστε τους πίνακες EMP, DEPT.

```
DROP TABLE EMP;
DROP TABLE DEPT;

CREATE TABLE DEPT (DEPTNO INT(2) NOT NULL,
  DNAME CHAR(14),
  LOC CHAR(13));

CREATE TABLE EMP (EMPNO INT (4) NOT NULL,
 ENAME CHAR(10),
 JOB CHAR(9),
 MGR INT (4),
 HIREDATE DATE,
 SAL FLOAT(7,2),
```

```
 COMM FLOAT(7,2),
 DEPTNO INT(2));
```

7.5.3.1 Προσθέστε κύρια κλειδιά. Δήλωση ALTER TABLE … ADD PRIMARY KEY

```
ALTER TABLE EMP ADD PRIMARY KEY(EMPNO);
ALTER TABLE DEPT ADD PRIMARY KEY(DEPTNO);
ALTER TABLE PROJ ADD PRIMARY KEY(PROJNO);
ALTER TABLE ASSIGN ADD PRIMARY KEY(EMPNO, PROJNO);
```

7.5.3.2 Αλλαγές στις στήλες πίνακα

Θα αλλάξουμε στον πίνακα EMP τις στήλες ENAME και JOB.ALTER TABLE EMP MODIFY ENAME CHAR(20);

```
ALTER TABLE EMP MODIFY JOB CHAR(20);

DESCRIBE EMP;
```

```
mysql> DESCRIBE EMP;
+----------+------------+------+-----+---------+-------+
| Field    | Type       | Null | Key | Default | Extra |
+----------+------------+------+-----+---------+-------+
| EMPNO    | int        | NO   | PRI | NULL    |       |
| ENAME    | char(20)   | YES  |     | NULL    |       |
| JOB      | char(20)   | YES  |     | NULL    |       |
| MGR      | int        | YES  |     | NULL    |       |
| HIREDATE | date       | YES  |     | NULL    |       |
| SAL      | float(7,2) | YES  |     | NULL    |       |
| COMM     | float(7,2) | YES  |     | NULL    |       |
| DEPTNO   | int        | YES  |     | NULL    |       |
+----------+------------+------+-----+---------+-------+
8 rows in set (0.01 sec)
```

Εικόνα 7.26 Περιγραφή στηλών πίνακα EMP μετά τις αλλαγές

7.5.3.3 Προσθήκη ξένων κλειδιών

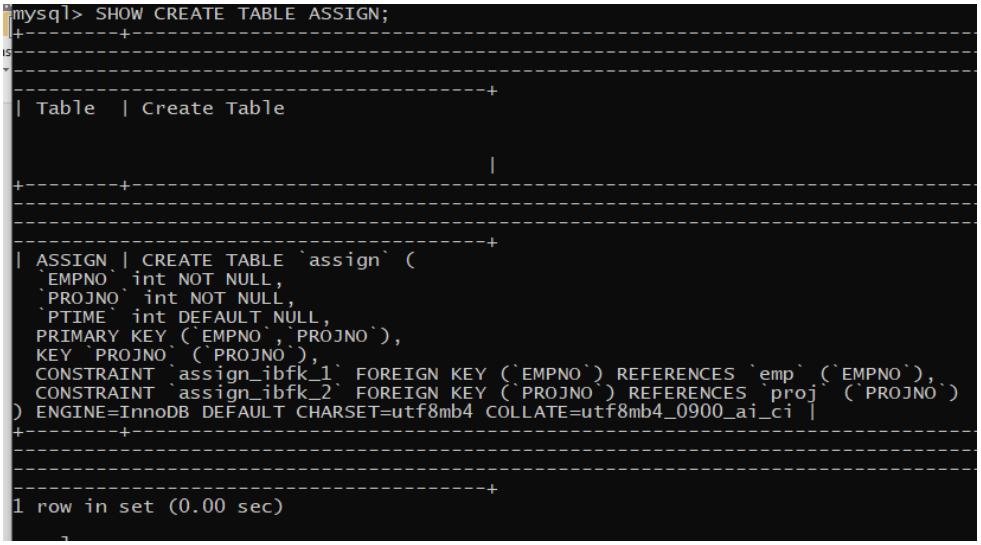
```
ALTER TABLE EMP ADD
   FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO);
```

```
mysql>
mysql> ALTER TABLE EMP ADD FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESCRIBE EMP;
+----------+------------+------+-----+---------+-------+
| Field    | Type       | Null | Key | Default | Extra |
+----------+------------+------+-----+---------+-------+
| EMPNO    | int        | NO   | PRI | NULL    |       |
| ENAME    | char(20)   | YES  |     | NULL    |       |
| JOB      | char(20)   | YES  |     | NULL    |       |
| MGR      | int        | YES  |     | NULL    |       |
| HIREDATE | date       | YES  |     | NULL    |       |
| SAL      | float(7,2) | YES  |     | NULL    |       |
| COMM     | float(7,2) | YES  |     | NULL    |       |
| DEPTNO   | int        | YES  | MUL | NULL    |       |
+----------+------------+------+-----+---------+-------+
8 rows in set (0.01 sec)
```

Εικόνα 7.27Προσθήκη ξένου κλειδιού σε πίνακα

```
ALTER TABLE ASSIGN ADD
 FOREIGN KEY(EMPNO) REFERENCES EMP(EMPNO);
```

```
ALTER TABLE ASSIGN ADD
   FOREIGN KEY(PROJNO) REFERENCES PROJ(PROJNO);
DESCRIBE ASSIGN;
```



Εικόνα 7.28 Πως ορίσαμε τον πίνακα ASSIGN

7.5.4 Πώς διαχειριζόμαστε δείκτες (ευρετήρια, index). Δήλωση CREATE INDEX

Όταν γράφετε 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 ερωτήσεων.

7.5.4.1 Πώς ορίζουμε δείκτες για απλές στήλες πίνακα

Για να ορίσουμε το όνομα (ENAME) σαν δείκτη INAME στον πίνακα EMP

```
CREATE INDEX INAME ON EMP (ENAME);
```

7.5.4.2 Πώς ορίζουμε δείκτες για συνδυασμό στηλών πίνακα

Για να ορίσουμε δείκτη το συνδυασμό Μισθού (SAL), προμήθειας (COMM)

```
CREATE INDEX SALCOM ON EMP (SAL,COMM);
```

Εναλλακτικά μπορούμε να ορίσουμε αλλιώς το ευρετήριο SALCOM

```
DROP INDEX SALCOM ON EMP;
CREATE INDEX SALCOM ON EMP (COMM, SAL);
```

Προσοχή! Οι δύο ορισμοί δεν είναι ισοδύναμοι.

Αν οι αναζητήσεις συνήθως περιέχουν υποπροτάσεις WHERE όπως:

```
 WHERE SAL< 1180 AND SAL> 1160
```

τότε πρέπει να χρησιμοποιήσετε τον πρώτο ορισμό.

7.5.4.3 Πως βλέπουμε τα ευρετήρια

SHOW INDEXES FROM EMP;

Και πως βλέπουμε τον ορισμό που γράψαμε για τη δημιουργία πίνακα με τα ευρετήρια του.

```
mysql>
mysql> show create table emp;
+-------+-----------------------
```

```
mysql> SELECT * FROM EMP;
+-------+------------------+------------------+------+------------+---------+---------+--------+
| EMPNO | ENAME            | JOB              | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+------------------+------------------+------+------------+---------+---------+--------+
|  6956 | ΣΚΟΥΡΑΣ          | ΠΩΛΗΤΗΣ          | 7890 | 2019-01-01 | 1000.00 |    NULL |     30 |
|  7512 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ          | 7890 | 2019-01-01 | 1000.00 |    NULL |     30 |
|  7522 | ΑΝΔΡΕΟΥ Ν.       | ΚΛΗΤΗΡΑΣ         | 7890 | 2017-01-01 | 1000.00 |    NULL |     30 |
|  7612 | ΑΝΔΡΕΟΥ          | ΠΩΛΗΤΗΣ          | 7890 | 2009-01-01 | 1000.00 |    NULL |     30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       | MANAGER          | 7890 | 2007-01-01 | 3000.00 |  500.00 |     30 |
+-------+------------------+------------------+------+------------+---------+---------+--------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM DEPT;
+--------+----------------------+------------+
| DEPTNO | DNAME                | LOC        |
+--------+----------------------+------------+
|     10 | ΛΟΓΙΣΤΗΡΙΟ           | ΑΘΗΝΑ      |
|     30 | ΠΩΛΗΣΕΙΣ             | ΑΘΗΝΑ      |
+--------+----------------------+------------+
2 rows in set (0.00 sec)
```

Εικόνα 7.30 Δεδομένα πινάκων

Δώστε σε όλους τους πωλητές προμήθεια 400 ευρώ.

```
UPDATE EMP
SET COMM=400
WHERE JOB="ΠΩΛΗΤΗΣ"
AND EMPNO NOT IN (7890);

SELECT * FROM EMP;
```

```
mysql>
mysql> # Δώστε σε όλους τους πωλητές προμήθεια 400 ευρώ.
mysql> UPDATE EMP
    -> SET COMM=400
    -> WHERE JOB="ΠΩΛΗΤΗΣ"
    -> AND EMPNO NOT IN (7890);
Query OK, 3 rows affected (0.01 sec)
Rows matched: 3  Changed: 3  Warnings: 0

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)
```

Εικόνα 7.31 Δίδεται προμήθεια σε όλους

7.6 Πως ορίζουμε και διαχειριζόμαστε όψεις (views) σε MySQL

Αρχίζουμε με τον ορισμό της όψης (view)

Μια όψη (view) είναι ένας ιδεατός (virtual) πίνακας που περιλαμβάνει στοιχεία από ένα ή περισσότερους 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). Η χρήση τους συνιστάται σε περιπτώσεις όπως:

- απλοποίηση της προσπέλασης στοιχείων
- ακεραιότητα στοιχείων
- ανεξαρτησία των στοιχείων
- ασφάλεια των στοιχείων

- προστασία του ιδιωτικού απόρρητου.

7.6.1 Πως ορίζουμε και πως καταργούμε όψη

Θέμα 1

Θα ορίσουμε την όψη των υπαλλήλων που έχουν επικεφαλής τον ΑΝΔΡΕΟΥ Ν.

```
CREATE VIEW PROJMGR (EMPLOYEE, SALARY)
  AS SELECT  ENAME,SAL
  FROM   EMP
  WHERE  EMP.MGR= 7890;
```

Θέμα 2

Για να δούμε το περιεχόμενο της όψης.

```
SELECT  EMPLOYEE, SALARY FROM PROJMGR;
```

```
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>
mysql>
mysql>
mysql> CREATE VIEW PROJMGR (EMPLOYEE, SALARY)
    ->           AS SELECT  ENAME,SAL
    ->           FROM   EMP
    ->           WHERE  EMP.MGR= 7890;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> # Για να δούμε το περιεχόμενο της όψης.
mysql>
mysql> SELECT  EMPLOYEE, SALARY FROM PROJMGR;
+-------------------+---------+
| EMPLOYEE          | SALARY  |
+-------------------+---------+
| ΣΚΟΥΡΑΣ           | 1000.00 |
| ΝΙΚΟΥ Ν.          | 1000.00 |
| ΑΝΔΡΕΟΥ Ν.        | 1000.00 |
| ΑΝΔΡΕΟΥ           | 1000.00 |
| ΑΝΔΡΕΟΥ Ν.        | 3000.00 |
+-------------------+---------+
5 rows in set (0.00 sec)
```

Εικόνα 7.32 Η δημιουργία όψης και το περιεχόμενό της

Θέμα 3

Για να καταργήσουμε την όψη

```
DROP VIEW projmgr;
```

7.6.2 Πώς ορίζουμε όψη με χρήση join

Παράδειγμα 1

Να σχεδιάσετε μια αναφορά που να παρουσιάζει τους υπάλληλους του έργου 15.

Πρώτα δημιουργείτε μια όψη βασιζόμενη στους πίνακες EMP και PROJ, ASSIGN

Θα χρειαστεί πρώτα να εισάγουμε στοιχεία στους πίνακες.

```
ALTER TABLE PROJ MODIFY PNAME CHAR(20);

INSERT INTO PROJ VALUES(12, "PAYROLL", 30000);
INSERT INTO PROJ VALUES(15, "DATA ANALYSIS", 90000);

SELECT * FROM PROJ;
SELECT * FROM EMP;
```

```
mysql> SELECT * FROM PROJ;
+--------+---------------+----------+
| PROJNO | PNAME         | BUDGET   |
+--------+---------------+----------+
|     12 | PAYROLL       | 30000.00 |
|     15 | DATA ANALYSIS | 90000.00 |
+--------+---------------+----------+
2 rows in set (0.00 sec)

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)
```

Εικόνα 7.33 Στοιχεία πινάκων

```
INSERT INTO ASSIGN VALUES(7890, 15, 100), (6956, 12, 30), (6956, 15, 70), (7512,
15, 100);

SELECT * FROM ASSIGN;
```

```
mysql>
mysql> INSERT INTO ASSIGN VALUES(7890, 15, 100), (6956, 12, 30), (6956, 15, 70), (7512, 15, 100);
Query OK, 4 rows affected (0.02 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM ASSIGN;
+-------+--------+-------+
| EMPNO | PROJNO | PTIME |
+-------+--------+-------+
|  6956 |     12 |    30 |
|  6956 |     15 |    70 |
|  7512 |     15 |   100 |
|  7890 |     15 |   100 |
+-------+--------+-------+
4 rows in set (0.00 sec)
```

Εικόνα 7.34 Προσθήκη στοιχείων στον πίνακα assign

Τώρα είμαστε έτοιμοι.

```
SELECT ASSIGN.EMPNO, ENAME, ASSIGN.PROJNO, PNAME, PTIME
 FROM   EMP, ASSIGN, PROJ
 WHERE  EMP.EMPNO =ASSIGN.EMPNO
 AND ASSIGN.PROJNO=PROJ.PROJNO
 AND ASSIGN.PROJNO=15;

CREATE VIEW PROJ_15
  (EMP_CODE, ENAME, PROJ_CODE, PROJECT, PTIME)
 AS SELECT ASSIGN.EMPNO, ENAME,
   ASSIGN.PROJNO, PNAME, PTIME
 FROM   EMP, ASSIGN, PROJ
 WHERE  EMP.EMPNO =ASSIGN.EMPNO
 AND ASSIGN.PROJNO=PROJ.PROJNO
 AND ASSIGN.PROJNO=15;
```

```
SELECT * FROM PROJ_15;
```

```
mysql> SELECT ASSIGN.EMPNO, ENAME, ASSIGN.PROJNO, PNAME, PTIME
    ->              FROM   EMP, ASSIGN, PROJ
    ->              WHERE  EMP.EMPNO =ASSIGN.EMPNO
    ->              AND ASSIGN.PROJNO=PROJ.PROJNO
    ->              AND ASSIGN.PROJNO=15;
+-------+------------------+--------+--------------+-------+
| EMPNO | ENAME            | PROJNO | PNAME        | PTIME |
+-------+------------------+--------+--------------+-------+
|  7512 | ΝΙΚΟΥ Ν.         |     15 | DATA ANALYSIS |   100 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       |     15 | DATA ANALYSIS |   100 |
|  6956 | ΣΚΟΥΡΑΣ          |     15 | DATA ANALYSIS |    70 |
+-------+------------------+--------+--------------+-------+
3 rows in set (0.00 sec)

mysql>
mysql> CREATE VIEW PROJ_15
    ->              (EMP_CODE, ENAME, PROJ_CODE, PROJECT, PTIME)
    ->  AS SELECT ASSIGN.EMPNO, ENAME,
    ->                    ASSIGN.PROJNO, PNAME, PTIME
    ->              FROM   EMP, ASSIGN, PROJ
    ->              WHERE  EMP.EMPNO =ASSIGN.EMPNO
    ->              AND ASSIGN.PROJNO=PROJ.PROJNO
    ->              AND ASSIGN.PROJNO=15;
ERROR 1050 (42S01): Table 'PROJ_15' already exists
mysql> SELECT * FROM PROJ_15;
+----------+------------------+-----------+--------------+-------
| EMP_CODE | ENAME            | PROJ_CODE | PROJECT      | PTIME
+----------+------------------+-----------+--------------+-------
|     7512 | ΝΙΚΟΥ Ν.         |        15 | DATA ANALYSIS |   100
|     7890 | ΑΝΔΡΕΟΥ Ν.       |        15 | DATA ANALYSIS |   100
|     6956 | ΣΚΟΥΡΑΣ          |        15 | DATA ANALYSIS |    70
+----------+------------------+-----------+--------------+-------
3 rows in set (0.00 sec)

mysql>
```

Εικόνα 7.35 Δημιουργία όψης η οποία βασίζεται σε σύνδεση πινάκων

Παράδειγμα 2

Ποιοι υπάλληλοι εργάζονται σε κάποια από τα έργα Ποιοι υπάλληλοι εργάζονται σε κάποια από τα έργα και ποια είναι τα έργα αυτά

```
SELECT ASSIGN.EMPNO, ENAME, ASSIGN.PROJNO, PNAME, PTIME
 FROM   EMP, ASSIGN, PROJ
 WHERE  EMP.EMPNO =ASSIGN.EMPNO
 AND ASSIGN.PROJNO=PROJ.PROJNO;
```

```
mysql>
mysql> SELECT ASSIGN.EMPNO, ENAME, ASSIGN.PROJNO, PNAME, PTIME
    ->              FROM   EMP, ASSIGN, PROJ
    ->              WHERE  EMP.EMPNO =ASSIGN.EMPNO
    ->              AND ASSIGN.PROJNO=PROJ.PROJNO;
+-------+------------------+--------+--------------+-------+
| EMPNO | ENAME            | PROJNO | PNAME        | PTIME |
+-------+------------------+--------+--------------+-------+
|  6956 | ΣΚΟΥΡΑΣ          |     12 | PAYROLL      |    30 |
|  6956 | ΣΚΟΥΡΑΣ          |     15 | DATA ANALYSIS |    70 |
|  7512 | ΝΙΚΟΥ Ν.         |     15 | DATA ANALYSIS |   100 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       |     15 | DATA ANALYSIS |   100 |
+-------+------------------+--------+--------------+-------+
4 rows in set (0.00 sec)
```

Εικόνα 7.36 Ποιοι υπάλληλοι εργάζονται σε έργα

```
CREATE VIEW PROJSTAFF
  (EMP_CODE, ENAME, PROJ_CODE, PROJECT, PTIME)
 AS SELECT ASSIGN.EMPNO, ENAME,
```

```
   ASSIGN.PROJNO, PNAME, PTIME
 FROM   EMP, ASSIGN, PROJ
 WHERE  EMP.EMPNO =ASSIGN.EMPNO
 AND ASSIGN.PROJNO=PROJ.PROJNO;

SELECT * FROM PROJSTAFF;
```

```
mysql>
mysql> CREATE VIEW PROJSTAFF
    ->             (EMP_CODE, ENAME, PROJ_CODE, PROJECT, PTIME)
    ->  AS SELECT ASSIGN.EMPNO, ENAME,
    ->                 ASSIGN.PROJNO, PNAME, PTIME
    ->           FROM   EMP, ASSIGN, PROJ
    ->           WHERE  EMP.EMPNO =ASSIGN.EMPNO
    ->           AND ASSIGN.PROJNO=PROJ.PROJNO;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM PROJSTAFF;
+----------+------------------+-----------+---------------+-------+
| EMP_CODE | ENAME            | PROJ_CODE | PROJECT       | PTIME |
+----------+------------------+-----------+---------------+-------+
|     6956 | ΣΚΟΥΡΑΣ          |        12 | PAYROLL       |    30 |
|     6956 | ΣΚΟΥΡΑΣ          |        15 | DATA ANALYSIS |    70 |
|     7512 | ΝΙΚΟΥ Ν.         |        15 | DATA ANALYSIS |   100 |
|     7890 | ΑΝΔΡΕΟΥ Ν.       |        15 | DATA ANALYSIS |   100 |
+----------+------------------+-----------+---------------+-------+
4 rows in set (0.00 sec)
```

Εικόνα 7.37 Δημιουργία όψης

Παράδειγμα 3

Ποιοι υπάλληλοι εργάζονται σε έργα με έδρα ΑΘΗΝΑ και ποια είναι τα έργα αυτά;

```
SELECT ASSIGN.EMPNO, ENAME, ASSIGN.PROJNO, PNAME, PTIME,
 DNAME, LOC
 FROM   DEPT, EMP, ASSIGN, PROJ
 WHERE DEPT.DEPTNO=EMP.DEPTNO
 AND EMP.EMPNO =ASSIGN.EMPNO
 AND ASSIGN.PROJNO=PROJ.PROJNO;
```

```
mysql>
mysql> SELECT ASSIGN.EMPNO, ENAME, ASSIGN.PROJNO, PNAME, PTIME,
    ->                DNAME, LOC
    ->           FROM   DEPT, EMP, ASSIGN, PROJ
    ->           WHERE DEPT.DEPTNO=EMP.DEPTNO
    ->           AND EMP.EMPNO =ASSIGN.EMPNO
    ->           AND ASSIGN.PROJNO=PROJ.PROJNO;
+-------+------------------+--------+---------------+-------+------------------+-------------+
| EMPNO | ENAME            | PROJNO | PNAME         | PTIME | DNAME            | LOC         |
+-------+------------------+--------+---------------+-------+------------------+-------------+
|  6956 | ΣΚΟΥΡΑΣ          |     12 | PAYROLL       |    30 | ΠΩΛΗΣΕΙΣ         | ΑΘΗΝΑ       |
|  7890 | ΑΝΔΡΕΟΥ Ν.       |     15 | DATA ANALYSIS |   100 | ΠΩΛΗΣΕΙΣ         | ΑΘΗΝΑ       |
|  7512 | ΝΙΚΟΥ Ν.         |     15 | DATA ANALYSIS |   100 | ΠΩΛΗΣΕΙΣ         | ΑΘΗΝΑ       |
|  6956 | ΣΚΟΥΡΑΣ          |     15 | DATA ANALYSIS |    70 | ΠΩΛΗΣΕΙΣ         | ΑΘΗΝΑ       |
+-------+------------------+--------+---------------+-------+------------------+-------------+
4 rows in set (0.00 sec)
```

Εικόνα 7.38 Ποιοι υπάλληλοι εργάζονται σε έργα και ποια είναι τα έργα αυτά

Πρώτα δημιουργείτε όψη που βασίζεται στους πίνακες ASSIGN, PROJ, EMP, DEPT:

```
CREATE VIEW ATHENS_PROJECTS
  (EMP_CODE, NAME, PROJ_CODE, PROJ, PTIME, DNAME, LOC)
  AS
  SELECT ASSIGN.EMPNO, ENAME, ASSIGN.PROJNO,
 PNAME, PTIME,
 DNAME, LOC
 FROM   DEPT, EMP, ASSIGN, PROJ
 WHERE DEPT.DEPTNO=EMP.DEPTNO
```

```
 AND EMP.EMPNO =ASSIGN.EMPNO
 AND ASSIGN.PROJNO=PROJ.PROJNO;

SELECT * FROM ATHENS_PROJECTS;
```

```
mysql>
mysql> CREATE VIEW ATHENS_PROJECTS
    ->           (EMP_CODE, NAME, PROJ_CODE, PROJ, PTIME, DNAME, LOC)
    ->           AS
    ->           SELECT ASSIGN.EMPNO, ENAME, ASSIGN.PROJNO,
    ->                      PNAME, PTIME,
    ->                      DNAME, LOC
    ->              FROM   DEPT, EMP, ASSIGN, PROJ
    ->              WHERE DEPT.DEPTNO=EMP.DEPTNO
    ->              AND EMP.EMPNO =ASSIGN.EMPNO
    ->              AND ASSIGN.PROJNO=PROJ.PROJNO;
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql>
mysql>
mysql> SELECT * FROM ATHENS_PROJECTS;
+----------+------------------+-----------+---------------+-------+------------------+------------+
| EMP_CODE | NAME             | PROJ_CODE | PROJ          | PTIME | DNAME            | LOC        |
+----------+------------------+-----------+---------------+-------+------------------+------------+
|     6956 | ΣΚΟΥΡΑΣ          |        12 | PAYROLL       |    30 | ΠΩΛΗΣΕΙΣ         | ΑΘΗΝΑ      |
|     7890 | ΑΝΔΡΕΟΥ Ν.       |        15 | DATA ANALYSIS |   100 | ΠΩΛΗΣΕΙΣ         | ΑΘΗΝΑ      |
|     7512 | ΝΙΚΟΥ Ν.         |        15 | DATA ANALYSIS |   100 | ΠΩΛΗΣΕΙΣ         | ΑΘΗΝΑ      |
|     6956 | ΣΚΟΥΡΑΣ          |        15 | DATA ANALYSIS |    70 | ΠΩΛΗΣΕΙΣ         | ΑΘΗΝΑ      |
+----------+------------------+-----------+---------------+-------+------------------+------------+
4 rows in set (0.00 sec)
```

Εικόνα 7.39 Δεδομένα που βλέπουμε στην όψη

Έπειτα γράφετε την αναζήτηση:

```
SELECT *
FROM ATHENS_PROJECTS
  WHERE LOC = "ΑΘΗΝΑ";
```

7.6.3 Ορισμός όψης με υποπρόταση check option

Η χρήση της υποπρότασης WITH CHECK OPTION επιτρέπει τον έλεγχο των ενεργειών του χρήστη κατά την ενημέρωση στοιχείων της βάσης, δηλαδή εξασφαλίζει την ακεραιότητα των στοιχείων, κ.λπ.

Αρχίζουμε παραθέτοντας απλά παραδείγματα της χρήσης της βάσης δεδομένων με και χωρίς όψεις.

Παράδειγμα 1

Δείτε το μισθό των υπαλλήλων

```
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)
```

Εικόνα 7.40 Στοιχειά υπαλλήλων

Παράδειγμα 2

Δώστε αύξηση 15% σε όλους

```
UPDATE EMP
SET SAL=SAL*1.15;
SELECT EMPNO, ENAME, SAL FROM EMP;
```

```
mysql> UPDATE EMP
    -> SET SAL=SAL*1.15;
Query OK, 5 rows affected (0.01 sec)
Rows matched: 5  Changed: 5  Warnings: 0

mysql> SELECT EMPNO, ENAME, SAL FROM EMP;
+-------+-------------------+---------+
| EMPNO | ENAME             | SAL     |
+-------+-------------------+---------+
|  6956 | ΣΚΟΥΡΑΣ           | 1150.00 |
|  7512 | ΝΙΚΟΥ Ν.          | 1150.00 |
|  7522 | ΑΝΔΡΕΟΥ Ν.        | 1150.00 |
|  7612 | ΑΝΔΡΕΟΥ           | 1150.00 |
|  7890 | ΑΝΔΡΕΟΥ Ν.        | 3450.00 |
+-------+-------------------+---------+
5 rows in set (0.00 sec)
```

Εικόνα 7.41 Στοιχεία υπαλλήλων μετά από τις μεταβολές

Παράδειγμα 3

Δώστε νέα αύξηση 30% στους υπάλληλους 6956 και 7512.

```
UPDATE EMP
SET SAL=SAL*1.30
WHERE EMPNO IN (6956, 7512);

SELECT EMPNO, ENAME, SAL FROM EMP;
```

```
mysql> UPDATE EMP
    -> SET SAL=SAL*1.30
    -> WHERE EMPNO IN (6956, 7512);
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql>
mysql> SELECT EMPNO, ENAME, SAL FROM EMP;
+-------+-------------------+---------+
| EMPNO | ENAME             | SAL     |
+-------+-------------------+---------+
|  6956 | ΣΚΟΥΡΑΣ           | 1495.00 |
|  7512 | ΝΙΚΟΥ Ν.          | 1495.00 |
|  7522 | ΑΝΔΡΕΟΥ Ν.        | 1150.00 |
|  7612 | ΑΝΔΡΕΟΥ           | 1150.00 |
|  7890 | ΑΝΔΡΕΟΥ Ν.        | 3450.00 |
+-------+-------------------+---------+
5 rows in set (0.00 sec)

mysql>
```

Εικόνα 7.42 Στοιχεία υπαλλήλων μετά τις νέες μεταβολές

Παράδειγμα 4

Έστω ότι δημιουργείτε την όψη των υπαλλήλων με μισθό πάνω από 1250.

```
CREATE VIEW CHECKSAL AS
 SELECT  *  FROM emp
 WHERE sal > 1250 ;
SELECT * FROM CHECKSAL;
```

```
mysql>
mysql> SELECT EMPNO, ENAME, SAL FROM EMP;
+-------+------------------+---------+
| EMPNO | ENAME            | SAL     |
+-------+------------------+---------+
|  6956 | ΣΚΟΥΡΑΣ          | 1495.00 |
|  7512 | ΝΙΚΟΥ Ν.         | 1495.00 |
|  7522 | ΑΝΔΡΕΟΥ Ν.       | 1150.00 |
|  7612 | ΑΝΔΡΕΟΥ          | 1150.00 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       | 3450.00 |
+-------+------------------+---------+
5 rows in set (0.00 sec)

mysql>
mysql>
mysql>
mysql> CREATE VIEW CHECKSAL AS
    ->       SELECT  *  FROM emp
    ->       WHERE sal > 1250 ;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> SELECT * FROM CHECKSAL;
+-------+------------------+----------------+------+------------+---------+--------+--------+
| EMPNO | ENAME            | JOB            | MGR  | HIREDATE   | SAL     | COMM   | DEPTNO |
+-------+------------------+----------------+------+------------+---------+--------+--------+
|  6956 | ΣΚΟΥΡΑΣ          | ΠΩΛΗΤΗΣ        | 7890 | 2019-01-01 | 1495.00 | 400.00 |     30 |
|  7512 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ        | 7890 | 2019-01-01 | 1495.00 | 400.00 |     30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       | MANAGER        | 7890 | 2007-01-01 | 3450.00 | 500.00 |     30 |
+-------+------------------+----------------+------+------------+---------+--------+--------+
3 rows in set (0.00 sec)
```

Εικόνα 7.43 Δημιουργία όψης

7.6.3.1 Προβλήματα σε όψεις οι οποίες ορίστηκαν χωρίς χρήση της υποπρότασης with check option

Εισάγουμε στοιχεία στη view και αυτά εισάγονται στον πίνακα emp. Θα διαπιστώσουμε ότι υπάρχει πρόβλημα.

Εισάγουμε χρησιμοποιώντας την όψη CHECKSAL νέο υπάλληλο με μισθό μικρότερο των 1250.

```
INSERT INTO CHECKSAL
VALUES(7777, "NIKOY N.", "ΠΩΛΗΤΗΣ",7890,
  "2021-01-01", 2000, NULL, 30);
INSERT INTO CHECKSAL
VALUES(8888, "NIKOY N.", "ΠΩΛΗΤΗΣ",7890,
   "2009-01-01",
   1000,NULL,30);

SELECT * FROM EMP;
SELECT * FROM CHECKSAL;
```

Να τι θα δούμε.

```
mysql>
mysql> INSERT INTO CHECKSAL
    ->         VALUES(7777, "ΝΙΚΟΥ Ν.", "ΠΩΛΗΤΗΣ",7890,
    ->                "2021-01-01", 2000, NULL, 30);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO CHECKSAL
    ->         VALUES(8888, "ΝΙΚΟΥ Ν.", "ΠΩΛΗΤΗΣ",7890,
    ->                "2009-01-01",
    ->               1000,NULL,30);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM EMP;
+-------+------------------+-----------------+------+------------+---------+---------+---------+
| EMPNO | ENAME            | JOB             | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+------------------+-----------------+------+------------+---------+---------+---------+
|  6956 | ΣΚΟΥΡΑΣ          | ΠΩΛΗΤΗΣ         | 7890 | 2019-01-01 | 1495.00 | 400.00 |      30 |
|  7512 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ         | 7890 | 2019-01-01 | 1495.00 | 400.00 |      30 |
|  7522 | ΑΝΔΡΕΟΥ Ν.       | ΚΛΗΤΗΡΑΣ        | 7890 | 2017-01-01 | 1150.00 |   NULL |      30 |
|  7612 | ΑΝΔΡΕΟΥ          | ΠΩΛΗΤΗΣ         | 7890 | 2009-01-01 | 1150.00 | 400.00 |      30 |
|  7777 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ         | 7890 | 2021-01-01 | 2000.00 |   NULL |      30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       | MANAGER         | 7890 | 2007-01-01 | 3450.00 | 500.00 |      30 |
|  8888 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ         | 7890 | 2009-01-01 | 1000.00 |   NULL |      30 |
+-------+------------------+-----------------+------+------------+---------+---------+---------+
7 rows in set (0.00 sec)

mysql> SELECT * FROM CHECKSAL;
+-------+------------------+-----------------+------+------------+---------+---------+---------+
| EMPNO | ENAME            | JOB             | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+------------------+-----------------+------+------------+---------+---------+---------+
|  6956 | ΣΚΟΥΡΑΣ          | ΠΩΛΗΤΗΣ         | 7890 | 2019-01-01 | 1495.00 | 400.00 |      30 |
|  7512 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ         | 7890 | 2019-01-01 | 1495.00 | 400.00 |      30 |
|  7777 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ         | 7890 | 2021-01-01 | 2000.00 |   NULL |      30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       | MANAGER         | 7890 | 2007-01-01 | 3450.00 | 500.00 |      30 |
+-------+------------------+-----------------+------+------------+---------+---------+---------+
4 rows in set (0.00 sec)
```

Εικόνα 7.44 Εισαγωγή δεδομένων

Διαπιστώστε ότι υπάρχει πρόβλημα παραβίασης του ορισμού της view.

```
SELECT * FROM EMP;
SELECT * FROM CHECKSAL;
```

```
mysql> SELECT * FROM EMP;
+-------+------------------+-----------------+------+------------+---------+---------+---------+
| EMPNO | ENAME            | JOB             | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+------------------+-----------------+------+------------+---------+---------+---------+
|  6956 | ΣΚΟΥΡΑΣ          | ΠΩΛΗΤΗΣ         | 7890 | 2019-01-01 | 1495.00 | 400.00 |      30 |
|  7512 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ         | 7890 | 2019-01-01 | 1495.00 | 400.00 |      30 |
|  7522 | ΑΝΔΡΕΟΥ Ν.       | ΚΛΗΤΗΡΑΣ        | 7890 | 2017-01-01 | 1150.00 |   NULL |      30 |
|  7612 | ΑΝΔΡΕΟΥ          | ΠΩΛΗΤΗΣ         | 7890 | 2009-01-01 | 1150.00 | 400.00 |      30 |
|  7777 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ         | 7890 | 2021-01-01 | 2000.00 |   NULL |      30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       | MANAGER         | 7890 | 2007-01-01 | 3450.00 | 500.00 |      30 |
|  8888 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ         | 7890 | 2009-01-01 | 1000.00 |   NULL |      30 |
+-------+------------------+-----------------+------+------------+---------+---------+---------+
7 rows in set (0.00 sec)

mysql> SELECT * FROM CHECKSAL;
+-------+------------------+-----------------+------+------------+---------+---------+---------+
| EMPNO | ENAME            | JOB             | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+------------------+-----------------+------+------------+---------+---------+---------+
|  6956 | ΣΚΟΥΡΑΣ          | ΠΩΛΗΤΗΣ         | 7890 | 2019-01-01 | 1495.00 | 400.00 |      30 |
|  7512 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ         | 7890 | 2019-01-01 | 1495.00 | 400.00 |      30 |
|  7777 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ         | 7890 | 2021-01-01 | 2000.00 |   NULL |      30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       | MANAGER         | 7890 | 2007-01-01 | 3450.00 | 500.00 |      30 |
+-------+------------------+-----------------+------+------------+---------+---------+---------+
4 rows in set (0.00 sec)
```

Εικόνα 7.45 Περιεχόμενο πίνακα και τι βλέπουμε στην όψη

Προσπαθήστε να διαγράψτε τους υπάλληλους που μόλις γράψατε στη βάση χρησιμοποιώντας τη view

```
DELETE FROM CHECKSAL WHERE EMPNO IN(7777, 8888);
SELECT * FROM EMP;
SELECT * FROM CHECKSAL;
```

```
mysql>
mysql> DELETE FROM CHECKSAL WHERE EMPNO IN(7777, 8888);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM EMP;
+-------+------------------+------------------+------+------------+---------+---------+--------+
| EMPNO | ENAME            | JOB              | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+------------------+------------------+------+------------+---------+---------+--------+
|  6956 | ΣΚΟΥΡΑΣ          | ΠΩΛΗΤΗΣ          | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7512 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ          | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7522 | ΑΝΔΡΕΟΥ Ν.       | ΚΛΗΤΗΡΑΣ         | 7890 | 2017-01-01 | 1150.00 |    NULL |     30 |
|  7612 | ΑΝΔΡΕΟΥ          | ΠΩΛΗΤΗΣ          | 7890 | 2009-01-01 | 1150.00 | 400.00  |     30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       | MANAGER          | 7890 | 2007-01-01 | 3450.00 | 500.00  |     30 |
|  8888 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ          | 7890 | 2009-01-01 | 1000.00 |    NULL |     30 |
+-------+------------------+------------------+------+------------+---------+---------+--------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM CHECKSAL;
+-------+------------------+------------------+------+------------+---------+---------+--------+
| EMPNO | ENAME            | JOB              | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+------------------+------------------+------+------------+---------+---------+--------+
|  6956 | ΣΚΟΥΡΑΣ          | ΠΩΛΗΤΗΣ          | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7512 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ          | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       | MANAGER          | 7890 | 2007-01-01 | 3450.00 | 500.00  |     30 |
+-------+------------------+------------------+------+------------+---------+---------+--------+
3 rows in set (0.00 sec)
```

Εικόνα 7.46 Τι βλέπουμε σε πίνακα και όψη μετά τη διαγραφή υπαλλήλου

Τι παρατηρήσατε;

Διαγράψτε και τον νέο υπάλληλο 8888.

```
DELETE FROM EMP WHERE EMPNO=8888;
SELECT * FROM EMP;
```

```
mysql>
mysql> DELETE FROM EMP WHERE EMPNO=8888;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM EMP;
+-------+------------------+------------------+------+------------+---------+---------+--------+
| EMPNO | ENAME            | JOB              | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+------------------+------------------+------+------------+---------+---------+--------+
|  6956 | ΣΚΟΥΡΑΣ          | ΠΩΛΗΤΗΣ          | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7512 | ΝΙΚΟΥ Ν.         | ΠΩΛΗΤΗΣ          | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7522 | ΑΝΔΡΕΟΥ Ν.       | ΚΛΗΤΗΡΑΣ         | 7890 | 2017-01-01 | 1150.00 |    NULL |     30 |
|  7612 | ΑΝΔΡΕΟΥ          | ΠΩΛΗΤΗΣ          | 7890 | 2009-01-01 | 1150.00 | 400.00  |     30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.       | MANAGER          | 7890 | 2007-01-01 | 3450.00 | 500.00  |     30 |
+-------+------------------+------------------+------+------------+---------+---------+--------+
5 rows in set (0.00 sec)
```

Εικόνα 7.47 Νέο περιεχόμενο πίνακα

7.6.3.2 Επίλυση προβλημάτων σε όψεις με την προσθήκη της υποπρότασης with check option στον ορισμό τους

Παράδειγμα 1

Να πως θα διορθώσουμε το πρόβλημα που διαπιστώσαμε στην ενότητα 7.6.3.2.

```
DROP VIEW CHECKSAL;
```

Υπενθυμίζουμε ότι η διαγραφή της όψης είναι σχετικά ανώδυνη επειδή ακριβώς διαγράφουμε ουσιαστικά μόνο έναν ορισμό χωρίς να επηρεάζουμε τα δεδομένα. Βέβαια, αν δεν προσέξουμε, για ένα χρήστη ο οποίος διαχειρίζεται κάποια από τα δεδομένα του πίνακα emp χρησιμοποιώντας την όψη CHECKSAL υπάρχει πρόβλημα. Πρέπει να ειδοποιηθεί έγκαιρα για την προσωρινή διακοπή της λειτουργίας της όψης και η όψη πρέπει να επαναδημιουργηθεί κατάλληλα ώστε ο χρήστης να τη χρησιμοποιεί με τον τρόπο που χρησιμοποιούσε την παλιά όψη CHECKSAL. Ακολουθεί η επαναδημιουργία της όψης.

```
CREATE VIEW CHECKSAL AS
 SELECT  *  FROM emp
```

```
 WHERE sal > 1800
 WITH  CHECK OPTION ;
```

Προσπαθούμε τώρα να εισάγουμε τα ίδια στοιχεία στη view και θα εξετάσουμε αν αυτά εισάγονται στον πίνακα emp.

```
INSERT INTO CHECKSAL
VALUES(7777, "NIKOY N.", "ΠΩΛΗΤΗΣ",7890,
  "2021-01-01", 2000, NULL, 30);

INSERT INTO CHECKSAL
VALUES(8888, "NIKOY N.", "ΠΩΛΗΤΗΣ",7890,
   "2009-01-01",
   1000,NULL,30);

SELECT * FROM EMP;
SELECT * FROM CHECKSAL;
```

```
mysql>
mysql> INSERT INTO CHECKSAL
    ->         VALUES(7777, "NIKOY N.", "ΠΩΛΗΤΗΣ",7890,
    ->                "2021-01-01", 2000, NULL, 30);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO CHECKSAL
    ->         VALUES(8888, "NIKOY N.", "ΠΩΛΗΤΗΣ",7890,
    ->                "2009-01-01",
    ->                1000,NULL,30);
ERROR 1369 (HY000): CHECK OPTION failed 'personnel.checksal'
mysql> SELECT * FROM EMP;
+-------+------------------+-----------------+------+------------+---------+---------+--------+
| EMPNO | ENAME            | JOB             | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+------------------+-----------------+------+------------+---------+---------+--------+
|  6956 | ΣΚΟΥΡΑΣ          | ΠΩΛΗΤΗΣ         | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7512 | NIKOY N.         | ΠΩΛΗΤΗΣ         | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7522 | ΑΝΔΡΕΟΥ N.       | ΚΛΗΤΗΡΑΣ        | 7890 | 2017-01-01 | 1150.00 |    NULL |     30 |
|  7612 | ΑΝΔΡΕΟΥ          | ΠΩΛΗΤΗΣ         | 7890 | 2009-01-01 | 1150.00 | 400.00  |     30 |
|  7777 | NIKOY N.         | ΠΩΛΗΤΗΣ         | 7890 | 2021-01-01 | 2000.00 |    NULL |     30 |
|  7890 | ΑΝΔΡΕΟΥ N.       | MANAGER         | 7890 | 2007-01-01 | 3450.00 | 500.00  |     30 |
+-------+------------------+-----------------+------+------------+---------+---------+--------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM CHECKSAL;
+-------+------------------+-----------------+------+------------+---------+---------+--------+
| EMPNO | ENAME            | JOB             | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+------------------+-----------------+------+------------+---------+---------+--------+
|  7777 | NIKOY N.         | ΠΩΛΗΤΗΣ         | 7890 | 2021-01-01 | 2000.00 |    NULL |     30 |
|  7890 | ΑΝΔΡΕΟΥ N.       | MANAGER         | 7890 | 2007-01-01 | 3450.00 | 500.00  |     30 |
+-------+------------------+-----------------+------+------------+---------+---------+--------+
2 rows in set (0.00 sec)
```

Εικόνα 7.48 Η υποπρόταση WITH CHECK OPTION «περιφρουρεί» την όψη

Βλέπουμε ότι δεν επιτρέπεται να εισάγουμε τον 8888 επειδή προσπαθούμε να του βάλουμε μισθό μικρότερο από 1250 (ενώ με την όψη λόγω, του ορισμού της, πρέπει να διαχειριζόμαστε μισθούς μικρότερους των 1250).

7.6.4 Συναλλαγές

Βλέπουμε τα δεδομένα του πίνακα EMP. Αρχίζουμε Transaction.

```
SELECT * FROM EMP;
SET AUTOCOMMIT=0;
```

Διαγράψτε τον υπάλληλο 7777.

```
DELETE FROM EMP WHERE EMPNO=7777;

SELECT * FROM EMP;
```

```
mysql>
mysql> SELECT * FROM EMP;
+-------+-------------------+-------------------+------+------------+---------+---------+--------+
| EMPNO | ENAME             | JOB               | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+-------------------+-------------------+------+------------+---------+---------+--------+
|  6956 | ΣΚΟΥΡΑΣ           | ΠΩΛΗΤΗΣ           | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7512 | ΝΙΚΟΥ Ν.          | ΠΩΛΗΤΗΣ           | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7522 | ΑΝΔΡΕΟΥ Ν.        | ΚΛΗΤΗΡΑΣ          | 7890 | 2017-01-01 | 1150.00 |    NULL |     30 |
|  7612 | ΑΝΔΡΕΟΥ           | ΠΩΛΗΤΗΣ           | 7890 | 2009-01-01 | 1150.00 | 400.00  |     30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.        | MANAGER           | 7890 | 2007-01-01 | 3450.00 | 500.00  |     30 |
+-------+-------------------+-------------------+------+------------+---------+---------+--------+
5 rows in set (0.00 sec)

mysql> SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)

mysql> # Διαγράψτε τον υπάλληλο 7777.
mysql> DELETE FROM EMP WHERE EMPNO=7777;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM EMP;
+-------+-------------------+-------------------+------+------------+---------+---------+--------+
| EMPNO | ENAME             | JOB               | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+-------------------+-------------------+------+------------+---------+---------+--------+
|  6956 | ΣΚΟΥΡΑΣ           | ΠΩΛΗΤΗΣ           | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7512 | ΝΙΚΟΥ Ν.          | ΠΩΛΗΤΗΣ           | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7522 | ΑΝΔΡΕΟΥ Ν.        | ΚΛΗΤΗΡΑΣ          | 7890 | 2017-01-01 | 1150.00 |    NULL |     30 |
|  7612 | ΑΝΔΡΕΟΥ           | ΠΩΛΗΤΗΣ           | 7890 | 2009-01-01 | 1150.00 | 400.00  |     30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.        | MANAGER           | 7890 | 2007-01-01 | 3450.00 | 500.00  |     30 |
+-------+-------------------+-------------------+------+------------+---------+---------+--------+
5 rows in set (0.00 sec)
```

Εικόνα 7.49 Μεταβολές στο περιεχόμενο του πίνακα

7.6.4.1 Ακύρωση συναλλαγής

Ακυρώστε τη συναλλαγή (Transaction).

```
ROLLBACK;
SELECT * FROM EMP;
```

```
mysql>
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM EMP;
+-------+-------------------+-------------------+------+------------+---------+---------+--------+
| EMPNO | ENAME             | JOB               | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+-------------------+-------------------+------+------------+---------+---------+--------+
|  6956 | ΣΚΟΥΡΑΣ           | ΠΩΛΗΤΗΣ           | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7512 | ΝΙΚΟΥ Ν.          | ΠΩΛΗΤΗΣ           | 7890 | 2019-01-01 | 1495.00 | 400.00  |     30 |
|  7522 | ΑΝΔΡΕΟΥ Ν.        | ΚΛΗΤΗΡΑΣ          | 7890 | 2017-01-01 | 1150.00 |    NULL |     30 |
|  7612 | ΑΝΔΡΕΟΥ           | ΠΩΛΗΤΗΣ           | 7890 | 2009-01-01 | 1150.00 | 400.00  |     30 |
|  7890 | ΑΝΔΡΕΟΥ Ν.        | MANAGER           | 7890 | 2007-01-01 | 3450.00 | 500.00  |     30 |
+-------+-------------------+-------------------+------+------------+---------+---------+--------+
5 rows in set (0.00 sec)
```

Παράδειγμα 1

Η δήλωση

```
CREATE VIEW checkemp
 AS SELECT * FROM emp
 WHERE deptno IN (SELECT deptno FROM dept)
 WITH CHECK OPTION ;
```

επιτρέπει εισαγωγή και ενημέρωση στοιχείων μόνο αν ο κωδικός του τμήματος (deptno) υπάρχει ήδη στον πίνακα DEPT.

Επομένως, η παρακάτω δήλωση δεν εκτελείται.

```
INSERT INTO CHECKSAL
VALUES(6666, "NIKOY N.", "ΠΩΛΗΤΗΣ",7890,
  "2021-01-01", 2000, NULL, 50);
```

```
mysql>
mysql> CREATE  VIEW  checkemp
    ->         AS  SELECT  *  FROM emp
    ->         WHERE  deptno  IN (SELECT  deptno  FROM  dept)
    ->         WITH CHECK OPTION ;
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO CHECKSAL
    ->         VALUES(6666, "NIKOY N.", "ΠΩΛΗΤΗΣ",7890,
    ->              "2021-01-01", 2000, NULL, 50);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`personnel`.`emp`, CONSTRAINT `FK_DEPTNO` FOREIG
N KEY (`DEPTNO`) REFERENCES `dept` (`DEPTNO`))
mysql>
```

Εικόνα 7.50 Δημιουργία ασφαλούς όψης

7.7 Δημιουργία και κατάργηση χρηστών στο προϊόν MySQL. Εκχώρηση και αφαίρεση δικαιωμάτων.

Δημιουργία χρήστη

```
CREATE USER 'jim'@'localhost' IDENTIFIED BY 'adams';
```

Κατάργηση χρήστη

DROP USER 'jim'@'localhost';

Δημιουργία χρήστη και εκχώρηση δικαιωμάτων

```
CREATE USER 'jim'@'localhost' IDENTIFIED BY 'adams';
GRANT ALL ON personnel.* TO 'jim'@'localhost';
SHOW GRANTS FOR 'jim'@'localhost';
```

```
mysql>
mysql> SHOW GRANTS FOR 'jim'@'localhost';
+-------------------------------------------------------------+
| Grants for jim@localhost                                    |
+-------------------------------------------------------------+
| GRANT USAGE ON *.* TO `jim`@`localhost`                     |
| GRANT ALL PRIVILEGES ON `personnel`.* TO `jim`@`localhost`  |
+-------------------------------------------------------------+
2 rows in set (0.00 sec)
```

Εικόνα 7.51 Εκχώρηση δικαιωμάτων

Αλλαγή δικαιωμάτων

```
ALTER USER 'jim'@'localhost' WITH MAX_QUERIES_PER_HOUR 90;
```

Πίνακας χρηστών

```
DESCRIBE mysql.user;


SELECT HOST, USER, SELECT_PRIV, INSERT_PRIV, UPDATE_PRIV, DELETE_PRIV FROM
mysql.user;
```

```
mysql> SELECT HOST, USER, SELECT_PRIV, INSERT_PRIV, UPDATE_PRIV, DELETE_PRIV FROM mysql.user;
+-----------+------------------+-------------+-------------+-------------+-------------+
| HOST      | USER             | SELECT_PRIV | INSERT_PRIV | UPDATE_PRIV | DELETE_PRIV |
+-----------+------------------+-------------+-------------+-------------+-------------+
| %         | christos         | Y           | Y           | Y           | Y           |
| localhost | jim              | N           | N           | N           | N           |
| localhost | mysql.infoschema | Y           | N           | N           | N           |
| localhost | mysql.session    | N           | N           | N           | N           |
| localhost | mysql.sys        | N           | N           | N           | N           |
| localhost | root             | Y           | Y           | Y           | Y           |
+-----------+------------------+-------------+-------------+-------------+-------------+
6 rows in set (0.00 sec)
```

Εικόνα 7.52 Χρήστες και δικαιώματα

7.7.1 Δηλώσεις GRANT και REVOKE. Εκχώρηση και αφαίρεση δικαιωμάτων

```
CREATE USER 'jim'@'localhost' IDENTIFIED BY 'adams';
GRANT ALL ON personnel.* TO 'jim'@'localhost';
SHOW GRANTS FOR 'jim'@'localhost';
```

```
mysql>
mysql> SHOW GRANTS FOR 'jim'@'localhost';
+-----------------------------------------------------------------+
| Grants for jim@localhost                                        |
+-----------------------------------------------------------------+
| GRANT USAGE ON *.* TO `jim`@`localhost`                         |
| GRANT ALL PRIVILEGES ON `personnel`.* TO `jim`@`localhost`      |
+-----------------------------------------------------------------+
2 rows in set (0.00 sec)
```

Εικόνα 7.53 Εκχώρηση δικαιωμάτων σε χρήστη

```
ALTER USER 'jim'@'localhost' WITH MAX_QUERIES_PER_HOUR 90;
DESCRIBE mysql.user;
SELECT HOST, USER, SELECT_PRIV, INSERT_PRIV, UPDATE_PRIV, DELETE_PRIV FROM
mysql.user;
```

```
mysql> SELECT HOST, USER, SELECT_PRIV, INSERT_PRIV, UPDATE_PRIV, DELETE_PRIV FROM mysql.user;
+-----------+------------------+-------------+-------------+-------------+-------------+
| HOST      | USER             | SELECT_PRIV | INSERT_PRIV | UPDATE_PRIV | DELETE_PRIV |
+-----------+------------------+-------------+-------------+-------------+-------------+
| %         | christos         | Y           | Y           | Y           | Y           |
| localhost | jim              | N           | N           | N           | N           |
| localhost | mysql.infoschema | Y           | N           | N           | N           |
| localhost | mysql.session    | N           | N           | N           | N           |
| localhost | mysql.sys        | N           | N           | N           | N           |
| localhost | root             | Y           | Y           | Y           | Y           |
+-----------+------------------+-------------+-------------+-------------+-------------+
6 rows in set (0.00 sec)
```

Εικόνα 7.54 Δικαιώματα μετά τις μεταβολές

Κεφάλαιο 8
Προγραμματισμός εφαρμογών βάσεων δεδομένων. Χρήση περιορισμών (constraint), όψεων (view) και εναυσμάτων (trigger)

Σύνοψη
Στο κεφάλαιο αυτό περιγράφονται σημαντικές έννοιες και εργαλεία του προγραμματισμού εφαρμογών βάσεων δεδομένων. Αρχικά περιγράφονται οι περιορισμοί (constraints) και οι όψεις (views) ως βασικά εργαλεία για τον προγραμματιστή και αποτελούν δομικά στοιχεία για μια στρατηγική υλοποίησης που μας επιτρέπει να διασφαλίσουμε τη συνέπεια (consistency), την ασφάλεια (security) και την ακεραιότητα (integrity) της βάσης δεδομένων. Στη συνέχεια γίνεται αναφορά στη συγγραφή και χρήση εναυσμάτων (triggers). Ο προγραμματισμός εναυσμάτων (triggers) και οι ενεργές βάσεις δεδομένων (Active databases) παρουσιάζονται σε περιβάλλον MySQL και Oracle PL/SQL.

Το κεφάλαιο περιλαμβάνει τις παρακάτω ενότητες που είναι οργανωμένες σαν «περιηγήσεις» (tours) και είναι καλό να μελετηθούν και να δοκιμαστούν και σε υπολογιστή:

1) *Περιήγηση στους περιορισμούς (a guided tour on SQL Constraints)*
2) *Περιήγηση στις όψεις (a guided tour on SQL View)*
3) *Triggers by example–Η περίπτωση του προϊόντος Oracle. Χρήση τεχνολογίας PL/SQL. Εισαγωγή στον προγραμματισμό με χρήση triggers.*
4) *Triggers by example–Η περίπτωση του προϊόντος MySQL*
5) *Περιήγηση. Προγραμματισμός εφαρμογής διαχείρισης παραγγελιών με χρήση triggers σε περιβάλλον Oracle PL/SQL*
6) *Προγραμματισμός εφαρμογής διαχείρισης παραγγελιών με χρήση triggers στο προϊόν MySQL*

Προαπαιτούμενη γνώση
Προτείνεται η μελέτη των κεφαλαίων 3, 4 και 6 του παρόντος συγγράμματος.

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

Σκοπός της ενότητας είναι η επισκόπηση δηλώσεων της γλώσσας SQL έτσι ώστε ο ενδιαφερόμενος να εστιάσει απρόσκοπτα σε θέματα υλοποίησης Βάσεων Δεδομένων με χρήση SQL που περιλαμβάνουν τα εξής:

- Δημιουργία βάσης δεδομένων, δημιουργία πινάκων, ορισμός ευρετηρίων, αλλαγή ορισμού (προσθήκη στηλών, τροποποίηση στηλών) πινάκων, κ.λπ.
- Διαχείριση Περιορισμών (Constraints)

Στόχος της ενότητας είναι η εμπέδωση της χρήσης περιορισμών:

- Ο περιορισμός Not Null
- Ο περιορισμός Unique σε στήλη ή στήλες του πίνακα

- Ο περιορισμός Primary Key
- Ο περιορισμός Foreign Key
- Ο περιορισμός DEFAULT
- Ο περιορισμός AUTO_INCREMENT (mySQL)
- Ο περιορισμός Check

Αφετηρία μας η υπενθύμιση κάποιων εννοιών απαραίτητων για τη διαχείριση βάσης δεδομένων.

8.1.1 Οι (υπο)γλώσσες της γλώσσας SQL

Η γλώσσα SQL περιλαμβάνει (υπο)γλώσσες και κάθε γλώσσα περιλαμβάνει δηλώσεις (statements). Παραθέτουμε γλώσσες και δηλώσεις:

1) Η **γλώσσα ορισμού δεδομένων, Data Definition Language (DDL)**, παρέχει δηλώσεις για τη διαχείριση του ορισμού αντικειμένων της βάσης δεδομένων.

```
CREATE DATABASE

ALTER DATABASE

CREATE TABLE

ALTER TABLE

DROP TABLE

CREATE INDEX

CREATE UNIQUE INDEX

DROP INDEX

CREATE TRIGGER

DROP TRIGGER

REPLACE TRIGGER

-- διαθέσιμη σε ORACLE και όχι σε MySQL

CREATE PROCEDURE

DROP PROCEDURE
CREATE FUNCTION
DROP FUNCTION
```

2) Η **γλώσσα διαχείρισης δεδομένων, Data Manipulation Language (DML)**, παρέχει δηλώσεις για τη διαχείριση αντικειμένων της βάσης δεδομένων.

```
SELECT

UPDATE

DELETE

INSERT INTO
```

3) Η **γλώσσα ελέγχου δεδομένων, Data Control Language (DCL)**, παρέχει δηλώσεις που εκχωρούν και αφαιρούν δικαιώματα

```
GRANT

REVOKE
```

Στις δηλώσεις της γλώσσας SQL περιλαμβάνονται και δηλώσεις που σχετίζονται με τη διαχείριση συναλλαγών (transaction)

```
COMMIT

ROLLBACK
```

```
DROP DATABASE IF EXISTS pers;

CREATE DATABASE pers;

USE pers;
```

Οι (υπο)γλώσσες της SQL και οι δηλώσεις θα χρησιμοποιηθούν στο παρόν κεφάλαιο. Στη συνέχεια θα εστιάσουμε στη δημιουργία βάσης δεδομένων, τη δημιουργία πινάκων, τον ορισμό ευρετηρίων, την αλλαγή ορισμού (προσθήκη στηλών, τροποποίηση στηλών) πινάκων. Στα παραδείγματα θα χρησιμοποιηθεί το προϊόν MySQL.

8.1.2 Δημιουργία βάσης δεδομένων και πινάκων

Έστω η βάση δεδομένων διεύθυνσης προσωπικού pers_constraint. Η βάση αποτελείται από τους πίνακες τμημάτων (DEPT) και των υπαλλήλων (EMP) τους.

Dept (στοιχεία τμήματος)

Deptno	Dname	Loc
κωδικός	Όνομα	Έδρα

Emp (στοιχεία υπαλλήλου)

Empno	Name	Job	Deptno	Sal	Comm
κωδικός	όνομα	θέση	Κωδικός τμήματος	μισθός	προμήθεια

Παραθέτουμε τις δηλώσεις δημιουργίας πινάκων και εισαγωγής δεδομένων.

```
CREATE TABLE DEPT (DEPTNO NUMERIC(2),
  DNAME VARCHAR(24),
  LOC CHAR(23));

CREATE TABLE EMP (EMPNO NUMERIC (4) NOT NULL,
 NAME VARCHAR (20), JOB VARCHAR(30), SAL DECIMAL(8,2),
 COMM DECIMAL(8,2), DEPTNO NUMERIC (2));

INSERT INTO DEPT(DEPTNO, DNAME, LOC)
 VALUES (10, 'SALES', 'ATHENS'), (20, 'ACCOUNTING', 'ATHENS');
```

```
INSERT INTO EMP(EMPNO, NAME, JOB, DEPTNO, SAL, COMM)
 VALUES (100, 'CODD', 'SALESMAN', 10, 2500, NULL),
(200, 'NAVATHE', 'SALESMAN', 10, 3500, 450),
(300, 'ELMASRI', 'ANALYST', 20, 2800, NULL),
(400, 'DATE', 'ANALYST', 10, 2400, NULL);
SELECT * FROM emp;
SELECT * FROM dept;
```

8.1.3 Δημιουργία ευρετηρίων (Create Index)

Τα ευρετήρια (indexes) δημιουργούνται για τα δεδομένα πινάκων προκειμένου να ανακτούμε τα δεδομένα των πινάκων γρηγορότερα. Τα ευρετήρια δεν είναι διάφανα στον χρήστη και χρησιμοποιούνται από το ΣΔΒΔ για να επιταχύνουν τις αναζητήσεις του χρήστη (users' queries). Παραθέτουμε τη σύνταξη της δήλωσης.

```
CREATE INDEX index_name ON table_name (column_name);
```

Θα παραθέσουμε είδη ευρετηρίων χρησιμοποιώντας παραδείγματα.

Ευρετήρια για απλές στήλες πίνακα

Για να ορίσουμε το ευρετήριο INAME που θα βασίζεται στη στήλη ENAME στον πίνακα EMP γράφουμε τη δήλωση: CREATE INDEX INAME ON EMP (ENAME);

Ευρετήρια για συνδυασμό στηλών πίνακα

Για να ορίσουμε το ευρετήριο SALCOM που θα βασίζεται στο συνδυασμό στηλών Μισθού (SAL) και προμήθειας (COMM) γράφουμε τη δήλωση: CREATE INDEX SALCOM ON EMP (SAL, COMM);

Προσοχή! Τα παρακάτω ευρετήρια είναι διαφορετικά και χρησιμοποιούνται σε διαφορετικές αναζητήσεις, δηλαδή βοηθούν στη γρηγορότερη εκτέλεση διαφορετικών SELECT δηλώσεων:

```
CREATE INDEX SALCOM ON EMP (SAL, COMM);
CREATE INDEX COMSAL ON EMP (COMM, SAL);
```

Κατάργηση ευρετηρίου – Δήλωση DROP INDEX

```
DROP INDEX salcom ON emp;
```

8.1.4 Διαχείριση Περιορισμών (Constraints)

Θα κατανοήσουμε τις περιπτώσεις που χρειαζόμαστε περιορισμούς με παράδειγμα.

Δημιουργήσαμε τον πίνακα dept

```
CREATE TABLE DEPT (DEPTNO NUMERIC(2),
  DNAME VARCHAR(24),
  LOC CHAR(23));
```

Παρατηρήστε ότι για τις στήλες του πίνακα δεν ορίσαμε περιορισμούς.

Εισάγετε δεδομένα.

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
 VALUES (10, 'SALES', 'ATHENS'), (NULL, 'ACCOUNTING', 'ATHENS');
```

Δείτε τα δεδομένα.

```
SELECT* FROM dept;
```

Deptno	Dname	Loc
10	SALES	ATHENS
NULL	ACCOUNTING	ATHENS

Δηλαδή, εισάγουμε ένα τμήμα χωρίς κωδικό.

Διαγράφουμε τα δεδομένα για να προχωρήσουμε στη συνέχεια σε παραδείγματα διαχείρισης διαφόρων ειδών περιορισμών.

```
DELETE FROM dept;
```

Ο περιορισμός NOT NULL

Όταν μια στήλη ενός πίνακα οριστεί ως NOT NULL δεν μπορεί να δεχτεί NULL τιμές (values). Παρατηρήστε ότι μπορούμε να προσθέσουμε εκ των υστέρων περιορισμούς NOT NULL σε στήλη του πίνακα αρκεί να μην υπάρχουν ήδη NULL τιμές στη στήλη.

Προσθήκη περιορισμού:

```
ALTER TABLE dept CHANGE deptno deptno NUMERIC(2) NOT NULL;
```

Εισαγωγή δεδομένων στον πίνακα για δοκιμές.

```
INSERT INTO dept(deptno, dname, loc) VALUES (10, 'SALES', 'ATHENS');
SELECT * FROM dept;
```

Deptno	Dname	loc
10	SALES	ATHENS

```
INSERT INTO dept(deptno, dname, loc) VALUES (NULL, 'ACCOUNTING', 'ATHENS');

ERROR 1048 (23000): column 'deptno' cannot be null
```

Προσοχή! Λύσαμε το πρόβλημα αλλά με τη δήλωση,

```
INSERT INTO dept(deptno, dname, loc) VALUES (10, 'ACCOUNTING', 'ATHENS');
```

μπορούμε να εισάγουμε δύο τμήματα με τον ίδιο κωδικό. Αυτό συμβαίνει γιατί δεν ορίσαμε τη στήλη deptno σαν κύριο κλειδί. Να τι θα δούμε.

```
SELECT * FROM dept;
```

Deptno	Dname	loc
10	SALES	ATHENS
10	ACCOUNTING	ATHENS

Διαγραφή των λανθασμένων δεδομένων

```
DELETE FROM dept WHERE deptno=10;
```

Ο περιορισμός Unique σε στήλη του πίνακα

Ο περιορισμός Unique (UNIQUE constraint) εφαρμόζεται σε στήλη (ή σε συνδυασμό στηλών) του πίνακα και σημαίνει ότι κάθε τιμή στη στήλη (ή το συνδυασμό στηλών) πρέπει να υπάρχει μόνο μια φορά. Δηλαδή, κάθε συγκεκριμένη τιμή αυτής της στήλης είναι μοναδική όπως θα συνέβαινε και αν είχαμε ορίσει ένα κύριο κλειδί για αυτή τη στήλη. Οι περιορισμοί UNIQUE και PRIMARY KEY (constraints) παρέχουν εγγύηση για τη μοναδικότητα μιας τιμής για μια στήλη ή ένα συνδυασμό στηλών.

Προσοχή! Μπορεί να έχετε πολλούς περιορισμούς UNIQUE constraints για κάθε πίνακα, αλλά μόνο ένα PRIMARY KEY constraint.

Ακολουθεί παράδειγμα. Έστω ότι στον πίνακα EMP έχετε εισάγει δεδομένα.

```
SELECT * FROM emp;
```

Empno	Name	Job	Sal	Comm	Deptno
100	CODD	SALESMAN	2500	NULL	10
200	NAVATHE	SALESMAN	3500	450	10
300	ELMASRI	ANALYST	2800	NULL	20
400	DATE	ANALYST	2400	NULL	10

Προσθέστε τον περιορισμό.

```
ALTER TABLE emp ADD UNIQUE (name);
```

Δοκιμάστε τη δήλωση,

```
INSERT INTO EMP(EMPNO, NAME, JOB, DEPTNO, SAL, COMM)
VALUES (100, 'CODD', 'SALESMAN', 10, 2500, NULL);
ERROR 1062 (23000): Duplicate entry 'CODD' for key 'NAME'
```

Δηλαδή ο περιορισμός εμπόδισε τη λανθασμένη εισαγωγή, όπως λέμε, διατήρησε την ακεραιότητα των δεδομένων μας.

Μπορείτε να δημιουργήστε τον περιορισμό δηλώνοντας και το όνομά του.

```
ALTER TABLE emp ADD CONSTRAINT uc_name UNIQUE (name);
```

8.1.5 Μία συστηματική περιήγηση στον ορισμό περιορισμών

Έστω η βάση δεδομένων διεύθυνσης προσωπικού pers_constraint. Η βάση αποτελείται από τους πίνακες DEPT, JOB, EMP. Υποτίθεται ότι ισχύουν οι επιχειρηματικός κανόνες (περιορισμοί) (business rules/constraints):

- Κάθε υπάλληλος ανήκει σε ένα μόνο τμήμα και κάθε τμήμα μπορεί να έχει πολλούς υπαλλήλους
- Κάθε υπάλληλος έχει ακριβώς μία θέση στο τμήμα του και για κάθε θέση έχουμε πολλούς υπαλλήλους να την κατέχουν
- Ο μισθός εξαρτάται από τη θέση, π.χ., όλοι οι αναλυτές έχουν τον ίδιο μισθό.

Dept (στοιχεία τμήματος)

Deptno	Dname	Loc
κωδικός	όνομα	έδρα

Job (στοιχεία θέσης)

Jobcode	Job_descr	Sal
Κωδικός	Περιγραφή	Μισθός

Emp

Empno	Name	Jobno	Deptno	Comm
κωδικός	όνομα	Κωδικός θέσης	Κωδικός τμήματος	προμήθεια

Δημιουργήστε τη βάση δεδομένων pers_constraint και τους πίνακες χρησιμοποιώντας τις παρακάτω δηλώσεις.

```
DROP DATABASE IF EXISTS pers_constraint;
CREATE DATABASE pers_constraint;
USE pers_constraint;
CREATE TABLE DEPT (DEPTNO NUMERIC(2),
  DNAME VARCHAR(24),
  LOC CHAR(23));

CREATE TABLE JOB(JOBCODE NUMERIC(3),
  JOB_DESCR VARCHAR(24),
  SAL NUMERIC (10,2));
CREATE TABLE EMP (EMPNO NUMERIC (4) NOT NULL,
 NAME VARCHAR (20),
 JOBNO NUMERIC(3),
 DEPTNO NUMERIC (2),
 COMM NUMERIC (10,2));
```

Δείτε τους πίνακες.

```
SHOW TABLES;
```

Εισάγετε δεδομένα στους πίνακες. Για παράδειγμα θα εισάγουμε τα δεδομένα που εμφανίζονται στους παρακάτω πίνακες.

DEPT

Deptno	Dname	Loc
50	SALES	ATHENS
60	ACCOUNTING	ATHENS
70	PAYROL	VOLOS

JOB

Jobcode	Job_descr	Sal
100	SALESMAN	2000
200	ANALYST	2000
300	DBA	3000

EMP

Empno	Name	Jobno	Deptno	Comm
10	CODD	100	50	NULL
20	NAVATHE	200	50	450
30	ELMASRI	300	60	NULL
40	DATE	100	50	NULL

Παραθέτουμε τις δηλώσεις.

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
 VALUES (50, 'SALES', 'ATHENS'),
(60, 'ACCOUNTING', 'ATHENS'),
(70, 'PAYROL', 'VOLOS');
INSERT INTO JOB(JOBCODE, JOB_DESCR, SAL)
 VALUES (100, 'SALESMAN', 2000),
(200, 'ANALYST', 2000),
(300, 'DBA', 3000);
INSERT INTO EMP(EMPNO, NAME, JOBNO, DEPTNO, COMM)
 VALUES (10, 'CODD', 100, 50, NULL),
(20, 'NAVATHE', 200, 50, 450),
(30, 'ELMASRI', 300, 60, NULL),
(40, 'DATE', 100, 50, NULL);
```

Δείτε τα στοιχεία

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

Προσθέστε τη στήλη HIREDATE στον πίνακα. Γράψτε δηλώσεις UPDATE για να εισάγετε στοιχεία στη νέα στήλη.

EMP

Empno	Name	Jobno	Deptno	Comm	Hiredate
10	CODD	100	50	NULL	10/01/2001
20	NAVATHE	200	50	450	25/02/1999
30	ELMASRI	300	60	NULL	17/03/2000
40	DATE	100	50	NULL	07/06/1989

```
# ALTER TABLE ADD COLUMN
ALTER TABLE EMP ADD HIREDATE DATE;
DESCRIBE EMP;
```

Με χρήση # στο προϊόν MySQL δηλώνουμε γραμμή σχολίων.

Τροποποιήστε το μήκος της στήλης JOB_DESCR

```
# ALTER TABLE MODIFY COLUMN
ALTER TABLE JOB MODIFY JOB_DESCR VARCHAR(30);
DESCRIBE JOB;
```

Καταργήστε τη στήλη loc

```
# ALTER TABLE DROP COLUMN
ALTER TABLE DEPT DROP loc;
DESCRIBE DEPT;
```

Αντικαταστήστε τη στήλη DNAME με τη στήλη DEPT_NAME

```
# ALTER TABLE CHANGE COLUMN
ALTER TABLE DEPT CHANGE DNAME DEPT_NAME VARCHAR(25);
DESCRIBE DEPT;
```

Δοκιμάστε να δημιουργήσετε πίνακα βασιζόμενοι σε υπάρχοντα πίνακα.

Αρχικά υποθέστε ότι τα δεδομένα του πίνακα EMP είναι:

EMP

Empno	Name	Jobno	Deptno	Comm	Hiredate
κωδικός	Όνομα	Θέση	τμήμα	προμήθεια	Ημερομηνία πρόσληψης
10	CODD	100	50	NULL	10/01/2001
20	NAVATHE	200	50	450	25/02/1999
30	ELMASRI	300	60	NULL	17/03/2000
40	DATE	100	50	NULL	07/06/1989

Δημιουργήστε τον πίνακα MY_EMP με δήλωση CREATE TABLE η οποία χρησιμοποιεί τους ίδιους τύπους δεδομένων για τις στήλες του που αντιστοιχούν σε στήλες του πίνακα EMP (αφήνεται ως άσκηση).

MY_EMP

Eno	Ename	Jobno	Deptno	Comm	Height
κωδικός	όνομα	Κωδικός θέσης	Κωδικός τμήματος	προμήθεια	ύψος

Μεταφέρετε στοιχεία από τον πίνακα EMP στον πίνακα MY_EMP.

```
INSERT INTO my_emp(eno, ename, deptno, comm)
  SELECT empno, ename, jobno, deptno, comm
FROM emp;
```

Ο πίνακας MY_EMP έχει τα παρακάτω δεδομένα.

MY_EMP

Eno	Ename	Jobno	Deptno	Comm	Height
10	CODD	100	50	NULL	
20	NAVATHE	200	50	450	
30	ELMASRI	300	60	NULL	
40	DATE	100	50	NULL	

Συμπληρώστε τα δεδομένα χρησιμοποιώντας δηλώσεις UPDATE.

```
UPDATE my_emp
SET height-1.80
WHERE eno=10;
```

Γράψτε και τις υπόλοιπες δηλώσεις UPDATE ώστε να συμπληρώσουμε τα δεδομένα στον πίνακα.

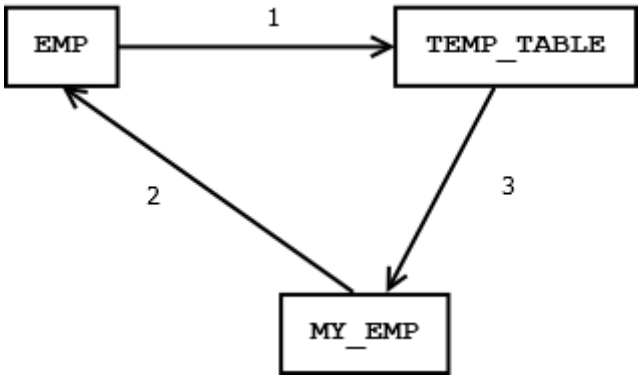
MY_EMP

Eno	Ename	Jobno	Deptno	Comm	Height
10	CODD	100	50	NULL	1.80
20	NAVATHE	200	50	450	2.10
30	ELMASRI	300	60	NULL	2.05
40	DATE	100	50	NULL	1.90

Στη συνέχεια παραθέτουμε θέματα προς επίλυση. Παραθέτουμε ενδεικτικές λύσεις.

Θέμα 1. Ανταλλάξτε τα ονόματα δύο πινάκων.

Στην εικόνα 8.1 βλέπουμε τον τρόπο σκέψης για την ανταλλαγή των ονομάτων.



Εικόνα 8.1 Ανταλλαγή των ονομάτων πινάκων

Υποθέστε ότι ο πίνακας EMP υπάρχει ήδη. Θα δημιουργήσουμε τον πίνακα MY_EMP και θα ανταλλάξουμε τα ονόματα των δύο πινάκων.

```
# RENAME TABLES
CREATE TABLE MY_EMP (ENO NUMERIC (4) NOT NULL,
 ENAME VARCHAR (20),
 JOBNO NUMERIC(3),
 DEPTNO NUMERIC (2),
 COMM NUMERIC (10,2),
 HEIGHT NUMERIC(5,2));

SHOW TABLES;
INSERT INTO MY_EMP(ENO, ENAME, JOBNO, DEPTNO, COMM, HEIGHT)
 VALUES (10, 'CODD', 100, 50, NULL, 1.80),
(20, 'NAVATHE', 200, 50, 450, 2.10),
(30, 'ELMASRI', 300, 60, NULL, 2.05),
(40, 'DATE', 100, 50, NULL, 1.90);

SELECT * FROM MY_EMP;
SELECT * FROM EMP;
```

```
RENAME TABLE EMP TO TEMP_TABLE,
 MY_EMP TO EMP,
 TEMP_TABLE TO MY_EMP;

SHOW TABLES;

SELECT * FROM MY_EMP;
SELECT * FROM EMP;
```

Θέμα 2.

Αλλάξτε το όνομα πίνακα χρησιμοποιώντας δήλωση ALTER TABLE ... RENAME ... ;

```
# RENAME TABLE USING ALTER TABLE
ALTER TABLE EMP RENAME TO NEW_EMP;
SHOW TABLES;
```

Θέμα 3.

Δημιουργήστε πίνακα (με δήλωση CREATE TABLE) βασιζόμενοι σε υπάρχοντα πίνακα. Τα ονόματα στηλών του νέου πίνακα διατηρούν τα ονόματα των αντίστοιχων στηλών του υπάρχοντος. Στον νέο πίνακα εισάγονται αυτόματα τα αντίστοιχα στοιχεία του υπάρχοντος.

```
# CREATE A NEW TABLE BASED ON AN EXISTING TABLE
DROP TABLE IF EXISTS TEMP_TABLE;
CREATE TABLE TEMP_TABLE(eno NUMERIC(4), ename VARCHAR(20),
 jobno NUMERIC(3), deptno NUMERIC(2))
AS SELECT  eno, ename, jobno, deptno
FROM NEW_EMP;
Δείτε τα δεομένα.
SELECT * FROM TEMP_TABLE;
SELECT * FROM NEW_EMP;
```

Θέμα 4.

Δημιουργήστε πίνακα βασιζόμενοι σε υπάρχοντα ή υπάρχοντες πίνακες. Στον νέο πίνακα εισάγονται στοιχεία του πίνακα ή των πινάκων με χρήση δήλωσης INSERT INTO ... SELECT ...;

```
# CREATE NEW TABLE. INSERT INTO ... SELECT ...;
DROP TABLE IF EXISTS TEMP_TABLE;
CREATE TABLE TEMP_TABLE(empno NUMERIC(4), name VARCHAR(20),
 jobno NUMERIC(3), deptno NUMERIC(2));
INSERT INTO TEMP_TABLE
SELECT  eno, ename, jobno, deptno
 FROM NEW_EMP;
SELECT * FROM TEMP_TABLE;
SELECT * FROM NEW_EMP;
```

Ακολουθούν θέματα τα οποία μας επιτρέπουν να κατανοήσουμε περισσότερο τον τρόπο χρήσης περιορισμών. Προηγουμένως ξαναδημιουργήστε τη βάση δεδομένων και τους πίνακες.

DEPT

Deptno	Dname	Loc
50	SALES	ATHENS

60	ACCOUNTING	ATHENS
70	PAYROL	VOLOS

JOB

Jobcode	Job_descr	Sal
100	SALESMAN	2000
200	ANALYST	2000
300	DBA	3000

EMP

Empno	Name	Jobno	Deptno	Comm
10	CODD	100	50	NULL
20	NAVATHE	200	50	450
30	ELMASRI	300	60	NULL
40	DATE	100	50	NULL

```
# DROP DATABASE. CREATE DATABASE
DROP DATABASE IF EXISTS pers_constraint;
CREATE DATABASE pers_constraint;
USE pers_constraint;

CREATE TABLE DEPT (DEPTNO NUMERIC(2),
  DNAME VARCHAR(24),
  LOC CHAR(23));

CREATE TABLE JOB(JOBCODE NUMERIC(3),
  JOB_DESCR VARCHAR(24),
  SAL NUMERIC (10,2));

CREATE TABLE EMP (EMPNO NUMERIC (4) NOT NULL,
 NAME VARCHAR (20),
 JOBNO NUMERIC(3),
 DEPTNO NUMERIC (2),
 COMM NUMERIC (10,2));

SHOW TABLES;

INSERT INTO DEPT(DEPTNO, DNAME, LOC)
 VALUES (50, 'SALES', 'ATHENS'),
(60, 'ACCOUNTING', 'ATHENS'),
(70, 'PAYROL', 'VOLOS');

INSERT INTO JOB(JOBCODE, JOB_DESCR, SAL)
 VALUES (100, 'SALESMAN', 2000),
(200, 'ANALYST', 2000),
(300, 'DBA', 3000);

INSERT INTO EMP(EMPNO, NAME, JOBNO, DEPTNO, COMM)
 VALUES (10, 'CODD', 100, 50, NULL),
(20, 'NAVATHE', 200, 50, 450),
```

```
(30, 'ELMASRI', 300, 60, NULL),
(40, 'DATE', 100, 50, NULL);

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

Θέμα 5.

Προσθέστε περιορισμό ορίζοντας κύριο κλειδί στον πίνακα DEPT

> Ο **πρώτος κανόνας ακεραιότητας**, σε μία περιγραφική προσέγγιση, ορίζει ότι το κύριο κλειδί ενός πίνακα δεν έχει τιμή NULL και δεν έχουμε δύο γραμμές στον πίνακα με την ίδια τιμή κλειδιού.
>
> Προσοχή! Αν ο πίνακας έχει ήδη γραμμές που παραβιάζουν τον πρώτο κανόνα ακεραιότητας τότε το κύριο κλειδί δεν δημιουργείται. Για παράδειγμα αν έχουμε γραμμές που έχουν την ίδια τιμή σε στήλη (ή σε συνδυασμό στηλών) δεν μπορούμε να ορίσουμε τη στήλη (ή το συνδυασμό στηλών) ως κύριο κλειδί. Πρέπει πρώτα να διορθώσουμε τα δεδομένα μας.

```
# ADD PRIMARY KEY CONSTRAINT
ALTER TABLE DEPT
 ADD CONSTRAINT pk_Deptno PRIMARY KEY (Deptno);
DESCRIBE DEPT;
```

Θέμα 6

Καταργήστε τον περιορισμό που ορίσατε για κύριο κλειδί στον πίνακα DEPT

```
# DROP CONSTRAINT PRIMARY KEY
ALTER TABLE DEPT DROP PRIMARY KEY;
DESCRIBE DEPT;
```

Θέμα 7

Καταργήστε και δημιουργήστε εκ νέου τον πίνακα DEPT ώστε η στήλη του department κατά την εισαγωγή στοιχείων αν δεν δώσουμε άλλη τιμή να έχει την προκαθορισμένη τιμή 'Development'.

Dept

D_id	department
10	Development
20	Development
30	Sales

```
# DEFAULT CONSTRAINT
DROP TABLE IF EXISTS DEPT;
CREATE TABLE dept
(D_Id INT NOT NULL,
department VARCHAR(90) DEFAULT 'Development',
PRIMARY KEY(d_id));
INSERT INTO dept(D_Id) VALUES (10), (20);
SELECT * FROM DEPT;
INSERT INTO DEPT VALUES (30, 'Sales');
SELECT * FROM DEPT;
```

Για τις ανάγκες του επόμενου θέματος που είναι πολύ σημαντικό θα ξαναδημιουργήσετε τη βάση δεδομένων και τους παρακάτω πίνακες:

DEPT

Deptno	Dname	Loc
50	SALES	ATHENS
60	ACCOUNTING	ATHENS
70	PAYROL	VOLOS

JOB

Jobcode	Job_descr	Sal
100	SALESMAN	2000
200	ANALYST	2000
300	DBA	3000

EMP

Empno	Name	Jobno	Deptno	Comm
10	CODD	100	50	NULL
20	NAVATHE	200	50	450
30	ELMASRI	300	60	NULL
40	DATE	100	50	NULL

Θέμα 8.

Προσθέστε κύρια και ξένα κλειδιά σε πίνακες

- ADD CONSTRAINT .... PRIMARY KEY .... ;
- ADD CONSTRAINT ... FOREIGN KEY ... REFERENCES ....;

Ο πρώτος κανόνας ακεραιότητας (1st Integrity rule), σε μία περιγραφική προσέγγιση, ορίζει ότι το κύριο κλειδί ενός πίνακα δεν έχει τιμή NULL και, επιπλέον, δεν έχουμε δύο γραμμές στον πίνακα με την ίδια τιμή κλειδιού. Αν το κύριο κλειδί είναι σύνθετο τότε όλες οι στήλες που το αποτελούν πρέπει να έχουν τιμή και επιπλέον δεν έχουμε δύο γραμμές στον πίνακα με την ίδια ακριβώς τιμή κλειδιού.

Ο δεύτερος κανόνας ακεραιότητας ή κανόνας αναφερόμενης ακεραιότητας (referential integrity) σε μία περιγραφική προσέγγιση, συνδέει-δεσμεύει τα δεδομένα ενός πίνακα με τα δεδομένα ενός άλλου πίνακα από τον οποίο λέμε ότι εξαρτάται. Οι πίνακες πρέπει να έχουν κύρια κλειδιά και το κύριο κλειδί του ενός να υπάρχει ως στήλη (ξένο κλειδί) και στον δεύτερο, π.χ., το κύριο κλειδί του πίνακα του τμήματος υπάρχει ως ξένο κλειδί και στον πίνακα του υπαλλήλου.

Ο δεύτερος κανόνας ορίζει ότι το ξένο κλειδί ενός πίνακα (π.χ. του πίνακα του υπαλλήλου) μπορεί να πάρει ως τιμή μόνο μία τιμή του κύριου κλειδιού του πίνακα με τον οποίο συνδέεται (π.χ., το ξένο κλειδί μπορεί

να πάρει ως τιμή μία τιμή του κύριου κλειδιού του πίνακα του τμήματος). Για να γίνει σαφέστερο, με παράδειγμα, ο υπάλληλος μπορεί να τοποθετηθεί σε ένα τμήμα που υπάρχει ήδη.

Λέμε ότι ο πίνακας με το ξένο κλειδί εξαρτάται από (αναφέρεται στον, referenced) τον άλλο πίνακα, π.χ. ο πίνακας του υπαλλήλου εξαρτάται από τον πίνακα του τμήματος.

Σημείωση. Το ξένο κλειδί που ορίζεται σε μία στήλη θα μπορούσε να πάρει τιμή NULL αν κατά τον ορισμό του πίνακα δεν το δηλώσαμε ως NOT NULL.

Προσοχή! Αν οι πίνακες έχουν ήδη γραμμές που παραβιάζουν τον πρώτο κανόνα ακεραιότητας τότε το κύριο κλειδί στους πίνακες δεν μπορεί να προστεθεί. Αν οι πίνακες παραβιάζουν τον δεύτερο κανόνα ακεραιότητας τότε το ξένο κλειδί δε δημιουργείται. Πρέπει πρώτα να διορθώσουμε τα δεδομένα μας.

```
# ADD PRIMARY KEYS-FOREIGN KEYS
DROP DATABASE IF EXISTS pers_constraint;
CREATE DATABASE pers_constraint;
USE pers_constraint;

CREATE TABLE DEPT (DEPTNO NUMERIC(2),
  DNAME VARCHAR(24),
  LOC CHAR(23));

CREATE TABLE JOB(JOBCODE NUMERIC(3),
  JOB_DESCR VARCHAR(24),
  SAL NUMERIC (10,2));

CREATE TABLE EMP (EMPNO NUMERIC (4) NOT NULL,
 NAME VARCHAR (20),
 JOBNO NUMERIC(3),
 DEPTNO NUMERIC (2),
 COMM NUMERIC (10,2));

ALTER TABLE DEPT
 ADD CONSTRAINT pk_Deptno PRIMARY KEY (Deptno);

DESCRIBE DEPT;

ALTER TABLE JOB
 ADD CONSTRAINT pk_Jobcode PRIMARY KEY (Jobcode);

DESCRIBE JOB;

ALTER TABLE EMP
 ADD CONSTRAINT pk_Empno PRIMARY KEY (Empno);

DESCRIBE EMP;

ALTER TABLE EMP
  ADD CONSTRAINT fk_Deptno FOREIGN KEY(Deptno) REFERENCES DEPT(Deptno);

DESCRIBE EMP;

ALTER TABLE EMP
ADD CONSTRAINT fk_Jobno FOREIGN KEY(Jobno) REFERENCES JOB(Jobcode);

DESCRIBE EMP;
```

Οι εντολές DESCRIBE χρησιμοποιούνται για να δούμε τη νέα δομή του πίνακα.

Θέμα 9 Διαγράψτε περιορισμούς κύριου, ξένου κλειδιού

Παραθέτουμε δηλώσεις για τη δημιουργία νέων πινάκων για τις ανάγκες του θέματος.

```
# DROP CONSTRAINT
DROP TABLE IF EXISTS Persons;
CREATE TABLE Persons (
PersonID int NOT NULL,
Person_Name char(20),
   PRIMARY KEY (PersonID));

DROP TABLE IF EXISTS Orders;
CREATE TABLE Orders (
OrderID int NOT NULL,
OrderNumber int NOT NULL,
PersonID int,
PRIMARY KEY (OrderID),
CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
REFERENCES Persons(PersonID)
);

INSERT INTO Persons VALUES (10, 'SMITH');
INSERT INTO Orders VALUES (1, 100, 10);

SELECT * FROM PERSONS;
SELECT * FROM ORDERS;

ALTER TABLE Orders
DROP FOREIGN KEY FK_PersonOrder;

INSERT INTO Orders VALUES (2, 100, 20);
SELECT * FROM PERSONS;
SELECT * FROM ORDERS;
```

Προσοχή! Αν έχουμε δύο πίνακες συνδεδεμένους με ξένο κλειδί θα πρέπει πρώτα να καταργηθεί το ξένο κλειδί και μετά να καταργηθούν τα κύρια κλειδιά.

Θέμα 10. Περιορισμοί στη σειρά εισαγωγής ή διαγραφής γραμμών δεδομένων λόγω των κανόνων ακεραιότητας. Ενεργοί και ανενεργοί περιορισμοί

Παραθέτουμε παράδειγμα. Οι πίνακες emp, job, dept συνδέονται με κύρια και ξένα κλειδιά. Ξαναδείτε το θέμα 8.

Αρχικά διαγράφουμε τα δεδομένα των πινάκων με δήλωση DELETE. Παραθέτουμε παράδειγμα ορθής σειράς εκτέλεσης των δηλώσεων.

```
DELETE FROM emp;
DELETE FROM job;
DELETE FROM dept;



# EXAMPLE OF DROP CONSTRAINT
# εισάγουμε δεδομένα, παραθέτουμε παράδειγμα ορθής σειράς εκτέλεσης των
δηλώσεων.
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
 VALUES (50, 'SALES', 'ATHENS'),
```

```
(60, 'ACCOUNTING', 'ATHENS'),
(70, 'PAYROL', 'VOLOS');

INSERT INTO JOB(JOBCODE, JOB_DESCR, SAL)
 VALUES (100, 'SALESMAN', 2000),
(200, 'ANALYST', 2000),
(300, 'DBA', 3000);

INSERT INTO EMP(EMPNO, NAME, JOBNO, DEPTNO, COMM)
 VALUES (10, 'CODD', 100, 50, NULL),
(20, 'NAVATHE', 200, 50, 450),
(30, 'ELMASRI', 300, 60, NULL),
(40, 'DATE', 100, 50, NULL);

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

Ζητούμενα.

- - Δείτε αν οι περιορισμοί FOREIGN KEY είναι ενεργοί (τιμή 1) ή ανενεργοί (τιμή 0).
- - Αλλάξτε την κατάσταση των περιορισμών τύπου Foreign Key.

Στο προϊόν MySQL έχουμε την εντολή:

```
SET FOREIGN_KEY_CHECKS -- Check or Not Foreign Key Constraints
```

Παραθέτουμε παραδείγματα.

```
-- Specify to check foreign key constraints (this is the default)
SET FOREIGN_KEY_CHECKS = 1;
SELECT @@FOREIGN_KEY_CHECKS;

-- Do not check foreign key constraints
SET FOREIGN_KEY_CHECKS = 0;
SELECT @@FOREIGN_KEY_CHECKS;
```

Θέμα 11. Δείτε όλους τους περιορισμούς σε μία βάση δεδομένων.

Στο προϊόν MySQL η βάση δεδομένων του συστήματος (με άλλη ορολογία, data dictionary) ονομάζεται INFORMATION_SCHEMA και περιλαμβάνει όλους τους ορισμούς των αντικείμενων της βάσης δεδομένων και πολλά άλλα στοιχεία.

```
SELECT *
FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
WHERE CONSTRAINT_SCHEMA='PERS_CONSTRAINT';
```

Θέμα 12. Δημιουργήστε πίνακα χρησιμοποιώντας περιορισμό auto_increment

Ο περιορισμός AUTO_INCREMENT εφαρμόζεται σε μία στήλη του πίνακα και διαχειρίζεται αυτόματα σαν αύξοντα αριθμό την τιμή της στήλης. Η δήλωση INSERT INTO δεν μπορεί να αλλάξει τηντιμή αυτή.

Ακολουθεί παράδειγμα.

```
DROP TABLE IF EXISTS Project;
```

```
CREATE TABLE Project(P_Id int NOT NULL AUTO_INCREMENT, Project varchar(255),
 PRIMARY KEY (P_Id));

INSERT INTO project(p_id, Project) VALUES(1,' Apollo 11');
INSERT INTO project(Project) VALUES(' Gemini 12');

SELECT * FROM Project;

ALTER TABLE Project AUTO_INCREMENT=100;

INSERT INTO project(Project) VALUES(' Soyuz TM-33');

SELECT * FROM Project;
```

Θέμα13. Ο περιορισμός Check

```
DROP TABLE Persons;
CREATE TABLE Persons(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CHECK (P_Id>0));

INSERT INTO Persons(P_ID, LastName) VALUES(-10, 'ADAMS' );

SELECT * FROM Persons;

DROP TABLE Persons;

CREATE TABLE Persons(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT chk_Person CHECK (P_Id>0 AND City='Sandnes')
);

INSERT INTO Persons(P_ID, LastName) VALUES(-10, 'ADAMS' );

SELECT * FROM Persons;
```

Οι δηλώσεις CREATE TABLE που γράψατε είναι σωστές. Δοκιμάσατε και δηλώσεις INSERT INTO. Τελικά ο περιορισμός CHECK δουλεύει στην περίπτωση της mySQL;

Η απάντηση είναι αρνητική μέχρι σήμερα για τις τρέχουσες versions του προϊόντος! Το αποδείξατε με παραδείγματα!

Άσκηση

Έστω ένα σύστημα βάσης δεδομένων διαχείρισης των έργων (projects) εταιρίας. Παραθέτουμε δηλώσεις για τον ορισμό της βάσης, των πινάκων και την εισαγωγή γραμμών στους πίνακες.

Ζητούμενα

1) Δείξτε ότι υπάρχει κίνδυνος εμφάνισης διπλο-γραμμών (γραμμών που είναι πανομοιότυπες, δηλαδή αντιστοιχούν σε στοιχεία που από λάθος εισάγονται δύο ή περισσότερες φορές)
2) Αν απαιτείται εισάγετε πρόσθετα δεδομένα και γράψτε ερωτήματα (query, SELECT) που εμφανίζει διπλό-γραμμές.
3) Προσθέστε primary key, foreign key πραγματοποιώντας τις απαραίτητες αλλαγές στα δεδομένα

```
SHOW DATABASES;

DROP DATABASE IF EXISTS 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),
  DNO INT(2));

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

CREATE TABLE ASSIGN(
 EMPNO INT(4) NOT NULL, PROJNO INT(3) NOT NULL,
 PTIME INT(3));

describe emp;

INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, '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 (11, 'CODD', 'ANALYST', '1989/01/01', 15, 2900, NULL, 15);
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, 15);
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 (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);

SELECT * FROM DEPT;
SELECT * FROM EMP;
SELECT * FROM PROJ;
SELECT * FROM ASSIGN;
```

8.2 Περιήγηση στις όψεις (a guided tour on SQL view)

Σκοπός της ενότητας είναι η επισκόπηση δηλώσεων της γλώσσας SQL έτσι ώστε ο ενδιαφερόμενος να εστιάσει απρόσκοπτα σε θέματα υλοποίησης βάσεων δεδομένων με χρήση SQL που περιλαμβάνουν δημιουργία και χρήση όψεων (views). Στόχος της ενότητας είναι η κατανόηση της ορθής χρησιμοποίησης των όψεων (view). Στην εικόνα 8.2 βλέπουμε τους πίνακες της βάσης δεδομένων με δείγμα στοιχείων.

Δημιουργήστε βάση δεδομένων και πίνακες.

```
DROP DATABASE IF EXISTS pers_view;
CREATE DATABASE pers_view;
USE pers_view;
CREATE TABLE DEPT (DEPTNO NUMERIC(2),
  DNAME VARCHAR(24),
  LOC CHAR(23));
CREATE TABLE JOB(JOBCODE NUMERIC(3),
  JOB_DESCR VARCHAR(24),
  SAL NUMERIC (10,2));
CREATE TABLE EMP (EMPNO NUMERIC (4) NOT NULL,
 NAME VARCHAR (20),
 JOBNO NUMERIC(3),
 DEPTNO NUMERIC (2),
 COMM NUMERIC (10,2));
```

Δείτε τους πίνακες.

```
SHOW TABLES;
```

Εισάγετε δεδομένα στους πίνακες.

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
 VALUES (50, 'SALES', 'ATHENS'),
(60, 'ACCOUNTING', 'ATHENS'),
(70, 'PAYROL', 'VOLOS');

INSERT INTO JOB(JOBCODE, JOB_DESCR, SAL)
 VALUES (100, 'SALESMAN', 2000),
(200, 'ANALYST', 2000),
(300, 'DBA', 3000);

INSERT INTO EMP(EMPNO, NAME, JOBNO, DEPTNO, COMM)
 VALUES (10, 'CODD', 100, 50, NULL),
(20, 'NAVATHE', 200, 50, 450),
(30, 'ELMASRI', 300, 60, NULL),
(40, 'DATE', 100, 50, NULL);

INSERT INTO EMP(EMPNO, NAME, JOBNO, DEPTNO, COMM)
 VALUES (50, 'CODD', 100, 50, NULL),
(60, 'CODD', 200, 50, 450),
(70, 'CODD', 200, 60, 500),
(80, 'CODD', 100, 60, NULL);
```

Δείτε τα στοιχεία

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

Στην εικόνα 8.2 βλέπουμε τους πίνακες της βάσης δεδομένων με δείγμα στοιχείων.

```
mysql> SELECT * FROM DEPT;
+--------+------------+--------+
| DEPTNO | DNAME      | LOC    |
+--------+------------+--------+
|     50 | SALES      | ATHENS |
|     60 | ACCOUNTING | ATHENS |
|     70 | PAYROL     | VOLOS  |
+--------+------------+--------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM JOB;
+---------+-----------+---------+
| JOBCODE | JOB_DESCR | SAL     |
+---------+-----------+---------+
|     100 | SALESMAN  | 2000.00 |
|     200 | ANALYST   | 2000.00 |
|     300 | DBA       | 3000.00 |
+---------+-----------+---------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM EMP;
+-------+---------+-------+--------+--------+
| EMPNO | NAME    | JOBNO | DEPTNO | COMM   |
+-------+---------+-------+--------+--------+
|    10 | CODD    |   100 |     50 |   NULL |
|    20 | NAVATHE |   200 |     50 | 450.00 |
|    30 | ELMASRI |   300 |     60 |   NULL |
|    40 | DATE    |   100 |     50 |   NULL |
|    50 | CODD    |   100 |     50 |   NULL |
|    60 | CODD    |   200 |     50 | 450.00 |
|    70 | CODD    |   200 |     60 | 500.00 |
|    80 | CODD    |   100 |     60 |   NULL |
+-------+---------+-------+--------+--------+
8 rows in set (0.00 sec)
```

Εικόνα 8.2 Οι πίνακες της βάσης δεδομένων με δείγμα στοιχείων. Χρησιμοποιούνται στα θέματα της ενότητας 8.2.

Θέμα 1.

Δημιουργία view με όνομα emp_view. Δείτε τα στοιχεία της. Κάθε φορά που ενημερώνω τον πίνακα ενημερώνεται και η view. Δοκιμάστε, στη συνέχεια, δηλώσεις INSERT, UPDATE, DELETE στη view. Διαπιστώστε ότι ενημερώνεται ο πίνακας EMP.

```
CREATE VIEW emp_view(e_ID, e_Name, e_Job, e_Dept, e_Comm)
AS SELECT EMPNO, NAME, JOBNO, DEPTNO, COMM FROM EMP;
SELECT * FROM emp_view;
```

Δείτε τα αποτελέσματα στην εικόνα 8.3.

```
mysql> CREATE VIEW emp_view(e_ID, e_Name, e_Job, e_Dept, e_Comm)
    -> AS SELECT EMPNO, NAME, JOBNO, DEPTNO, COMM FROM EMP;
Query OK, 0 rows affected (0.09 sec)

mysql> SELECT * FROM emp_view;
+------+---------+-------+--------+--------+
| e_ID | e_Name  | e_Job | e_Dept | e_Comm |
+------+---------+-------+--------+--------+
|   10 | CODD    |   100 |     50 |   NULL |
|   20 | NAVATHE |   200 |     50 | 450.00 |
|   30 | ELMASRI |   300 |     60 |   NULL |
|   40 | DATE    |   100 |     50 |   NULL |
|   50 | CODD    |   100 |     50 |   NULL |
|   60 | CODD    |   200 |     50 | 450.00 |
|   70 | CODD    |   200 |     60 | 500.00 |
|   80 | CODD    |   100 |     60 |   NULL |
+------+---------+-------+--------+--------+
8 rows in set (0.00 sec)
```

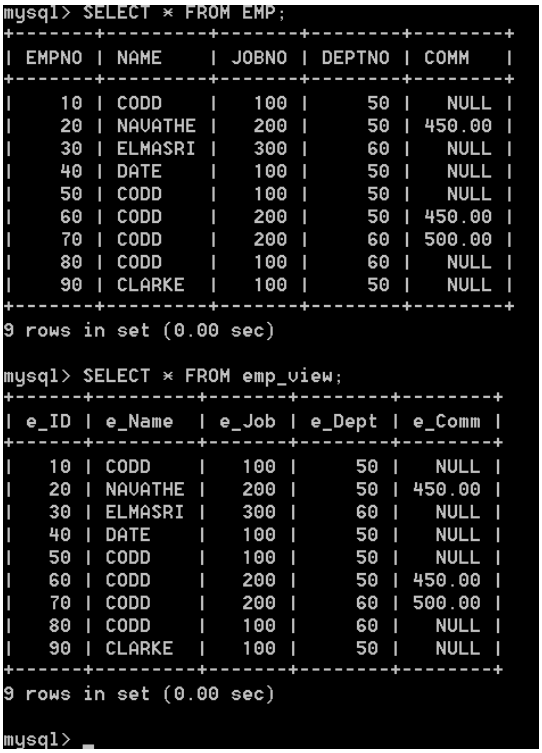
Εικόνα 8.3 Δημιουργία όψης και εμφάνιση δεδομένων

Εισάγουμε δεδομένα στον πίνακα.

```
INSERT INTO EMP(EMPNO, NAME, JOBNO, DEPTNO, COMM) VALUES (90, 'CLARKE', 100, 50, NULL);
```

Δείτε στην όψη τα αποτελέσματα της εισαγωγής δεδομένων στον βασικό πίνακα (εικόνα 8.4).

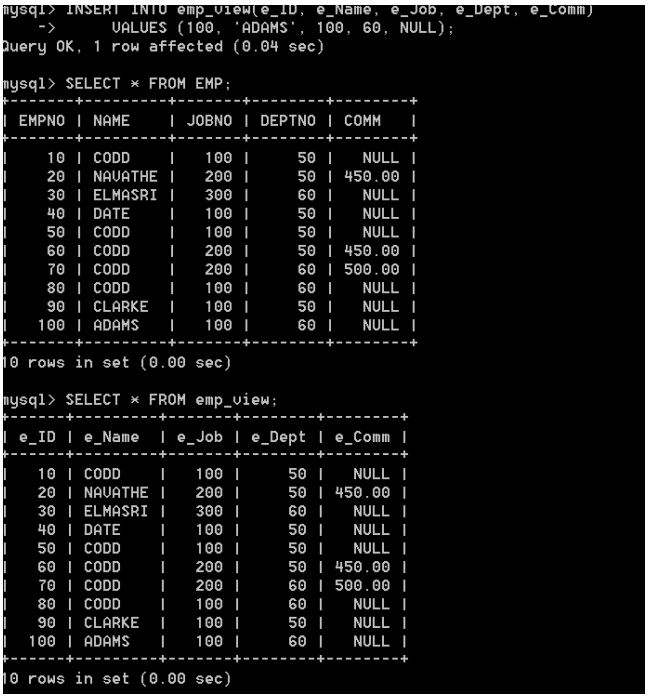
```
SELECT * FROM EMP;
SELECT * FROM emp_view;
```



Εικόνα 8.4 Εισάγουμε στοιχεία στον πίνακα και τα στοιχεία φαίνονται και στην όψη.

Εισάγουμε στοιχεία στην όψη και βλέπουμε (εικόνα 8.5) ότι τα στοιχεία εισάγονται στον πίνακα,

```
INSERT INTO emp_view(e_ID, e_Name, e_Job, e_Dept, e_Comm) VALUES (100, 'ADAMS', 100, 60, NULL);

SELECT * FROM EMP;
SELECT * FROM emp_view;
```

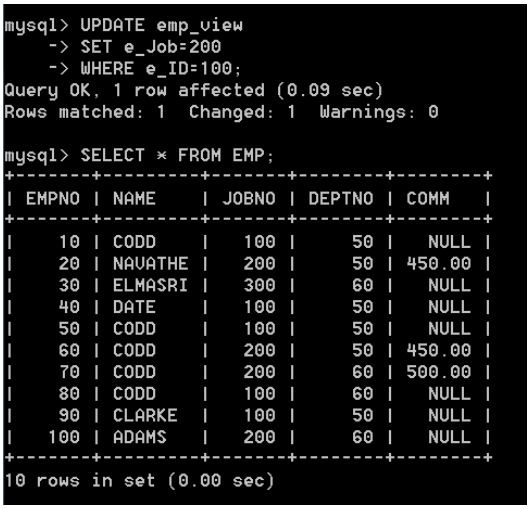


Εικόνα 8.5 Εισάγουμε στοιχεία στην όψη και αυτά εισάγονται στον πίνακα.

Αλλάζουμε (ενημερώνουμε) δεδομένα χρησιμοποιώντας την όψη.

```
UPDATE emp_view
SET e_Job=200
WHERE e_ID=100;

SELECT * FROM EMP;
SELECT * FROM emp_view;
```

Στην εικόνα 8.6 βλέπουμε ότι οι αλλαγές που κάναμε χρησιμοποιώντας την όψη έγιναν στον πίνακα.



Εικόνα 8.6 Η μεταβολή δεδομένων με χρήση της όψης συνεπάγεται τη μεταβολή των δεδομένων αυτών στο βασικό πίνακα.

8.2.1 Όψεις (view) και συναλλαγές (transaction)

Θα εξετάσουμε τις μεταβολές των δεδομένων της βάσης που γίνονται με συναλλαγές (transaction). Θα εξετάσουμε, πιο συγκεκριμένα, την επικύρωση (οριστικοποίηση) και την ακύρωση των μεταβολών.

Θέμα 1

Έστω ότι αρχίζουμε μια συναλλαγή.

```
START TRANSACTION;
DELETE FROM emp_view
WHERE e_Name='ADAMS';
```

Δείτε ενδιάμεσα αποτελέσματα.

```
SELECT * FROM EMP;
SELECT * FROM emp_view;
```

Διαπιστώστε ότι οι μεταβολές φαίνονται σε πίνακα και όψη (view). Στη συνέχεια θα τερματίσουμε τη συναλλαγή (transaction).

Με δήλωση ROLLBACK ακυρώνοντας όλες τις μεταβολές. Αν χρησιμοποιήσουμε δήλωση COMMIT; τότε όλες οι μεταβολές γράφονται μόνιμα στη βάση δεδομένων.

Αν δεν ξεκινήσουμε συναλλαγή (με δήλωση START TRANSACTION;) τότε το προϊόν mySQL είναι σε AUTOCOMMIT mode, δηλαδή κάθε δήλωση (INSERT, κ.λπ.) προκαλεί μεταβολές που γράφονται μόνιμα (COMMIT) στη βάση δεδομένων.

```
ROLLBACK;
```

Δείτε εκ νέου τα δεδομένα.

```
SELECT * FROM EMP;
SELECT * FROM emp_view;
```

Θέμα 2 «Αφύλακτη» όψη

Παραθέτουμε μία «αφύλακτη» όψη (view), δηλαδή μία όψη χωρίς την υποπρόταση with check option στον ορισμό της. Όπως θα διαπιστώσουμε, ενώ υποτίθεται από τον τρόπο ορισμού της όψης ότι μπορούμε να κάνουμε μεταβολές σε υπαλλήλους μόνο του τμήματος 50 η «αφύλακτη» όψη επιτρέπει παράνομες μεταβολές του περιεχομένου της βάσης δεδομένων.

Στην εικόνα 8.7 βλέπουμε τα δεδομένα του πίνακα και της όψης.

```
CREATE VIEW emp_on_SALES(e_ID, e_Name, e_Job, e_Dept, e_Comm)
AS SELECT EMPNO, NAME, JOBNO, DEPTNO, COMM FROM EMP
  WHERE deptno IN (SELECT deptno FROM dept WHERE dname='SALES');

SELECT * FROM EMP;
SELECT * FROM emp_on_SALES;
```

```
mysql> SELECT * FROM EMP;
+-------+---------+-------+--------+--------+
| EMPNO | NAME    | JOBNO | DEPTNO | COMM   |
+-------+---------+-------+--------+--------+
|    10 | CODD    |   100 |     50 |   NULL |
|    20 | NAVATHE |   200 |     50 | 450.00 |
|    30 | ELMASRI |   300 |     60 |   NULL |
|    40 | DATE    |   100 |     50 |   NULL |
|    50 | CODD    |   100 |     50 |   NULL |
|    60 | CODD    |   200 |     50 | 450.00 |
|    70 | CODD    |   200 |     60 | 500.00 |
|    80 | CODD    |   100 |     60 |   NULL |
|    90 | CLARKE  |   100 |     50 |   NULL |
|   100 | ADAMS   |   200 |     60 |   NULL |
+-------+---------+-------+--------+--------+
10 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM emp_on_SALES;
+------+---------+-------+--------+---------+
| e_ID | e_Name  | e_Job | e_Dept | e_Comm  |
+------+---------+-------+--------+---------+
|   10 | CODD    |   100 |     50 |    NULL |
|   20 | NAVATHE |   200 |     50 |  450.00 |
|   40 | DATE    |   100 |     50 |    NULL |
|   50 | CODD    |   100 |     50 |    NULL |
|   60 | CODD    |   200 |     50 |  450.00 |
|   90 | CLARKE  |   100 |     50 |    NULL |
+------+---------+-------+--------+---------+
6 rows in set (0.01 sec)
```

Εικόνα 8.7 Δεδομένα πίνακα και δεδομένα όψης,

Εισάγουμε στοιχεία στον πίνακα που σωστά δεν φαίνονται στην όψη

```
INSERT INTO EMP(EMPNO, NAME, JOBNO, DEPTNO, COMM)
VALUES (110, 'NAVATHE', 100, 60, NULL);
```

Η εισαγωγή δεδομένων υπαλλήλου στον πίνακα ο οποίος δεν τοποθετείται σε τμήμα δεν εμφανίζεται στην όψη λόγω του ορισμού της (εικόνα 8.8).

```
SELECT * FROM EMP;
SELECT * FROM emp_on_SALES;
```

```
mysql> INSERT INTO EMP(EMPNO, NAME, JOBNO, DEPTNO, COMM)
    ->      VALUES (110, 'NAVATHE', 100, 60, NULL);
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM EMP;
+-------+---------+-------+--------+--------+
| EMPNO | NAME    | JOBNO | DEPTNO | COMM   |
+-------+---------+-------+--------+--------+
|    10 | CODD    |   100 |     50 |   NULL |
|    20 | NAVATHE |   200 |     50 | 450.00 |
|    30 | ELMASRI |   300 |     60 |   NULL |
|    40 | DATE    |   100 |     50 |   NULL |
|    50 | CODD    |   100 |     50 |   NULL |
|    60 | CODD    |   200 |     50 | 450.00 |
|    70 | CODD    |   200 |     60 | 500.00 |
|    80 | CODD    |   100 |     60 |   NULL |
|    90 | CLARKE  |   100 |     50 |   NULL |
|   100 | ADAMS   |   200 |     60 |   NULL |
|   110 | NAVATHE |   100 |     60 |   NULL |
+-------+---------+-------+--------+--------+
11 rows in set (0.00 sec)

mysql> SELECT * FROM emp_on_SALES;
+------+---------+-------+--------+--------+
| e_ID | e_Name  | e_Job | e_Dept | e_Comm |
+------+---------+-------+--------+--------+
|   10 | CODD    |   100 |     50 |   NULL |
|   20 | NAVATHE |   200 |     50 | 450.00 |
|   40 | DATE    |   100 |     50 |   NULL |
|   50 | CODD    |   100 |     50 |   NULL |
|   60 | CODD    |   200 |     50 | 450.00 |
|   90 | CLARKE  |   100 |     50 |   NULL |
+------+---------+-------+--------+--------+
6 rows in set (0.00 sec)
```

Εικόνα 8.8 Εισάγουμε τα στοιχεία του υπαλλήλου 110 (Navathe) τα οποία δεν εμφανίζονται στην όψη

Τώρα θα «παρανομήσουμε».

Εισάγουμε στην όψη δεδομένα υπαλλήλου παραβιάζοντας τη συνθήκη WHERE που υπάρχει στον ορισμό της (εικόνα 8.9).

```
INSERT INTO emp_on_SALES(e_ID, e_Name, e_Job, e_Dept, e_Comm)
 VALUES (120, 'ELMASRI', 100, 60, NULL);

SELECT * FROM EMP;
SELECT * FROM emp_on_SALES;
```

```
mysql> SELECT * FROM EMP;
+-------+---------+-------+--------+--------+
| EMPNO | NAME    | JOBNO | DEPTNO | COMM   |
+-------+---------+-------+--------+--------+
|    10 | CODD    |   100 |     50 |   NULL |
|    20 | NAVATHE |   200 |     50 | 450.00 |
|    30 | ELMASRI |   300 |     60 |   NULL |
|    40 | DATE    |   100 |     50 |   NULL |
|    50 | CODD    |   100 |     50 |   NULL |
|    60 | CODD    |   200 |     50 | 450.00 |
|    70 | CODD    |   200 |     60 | 500.00 |
|    80 | CODD    |   100 |     60 |   NULL |
|    90 | CLARKE  |   100 |     50 |   NULL |
|   100 | ADAMS   |   200 |     60 |   NULL |
|   110 | NAVATHE |   100 |     60 |   NULL |
|   120 | ELMASRI |   100 |     60 |   NULL |
+-------+---------+-------+--------+--------+
12 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM emp_on_SALES;
+------+---------+-------+--------+--------+
| e_ID | e_Name  | e_Job | e_Dept | e_Comm |
+------+---------+-------+--------+--------+
|   10 | CODD    |   100 |     50 |   NULL |
|   20 | NAVATHE |   200 |     50 | 450.00 |
|   40 | DATE    |   100 |     50 |   NULL |
|   50 | CODD    |   100 |     50 |   NULL |
|   60 | CODD    |   200 |     50 | 450.00 |
|   90 | CLARKE  |   100 |     50 |   NULL |
+------+---------+-------+--------+--------+
6 rows in set (0.00 sec)
```

Εικόνα 8.9 Παραβιάζουμε τη συνθήκη ορισμού της όψης και εισάγουμε παράνομα τα στοιχεία του υπαλλήλου 120 (Elmasri)

8.2.2 Πως μια ενημερώσιμη όψη θα είναι ασφαλής ή να πως πρέπει να ορίζω τις (ενημερώσιμες) όψεις.

Παραθέτουμε παράδειγμα.

```
CREATE VIEW emp_on_SALES_safe(e_ID, e_Name, e_Job, e_Dept, e_Comm)
AS SELECT EMPNO, NAME, JOBNO, DEPTNO, COMM
FROM EMP
WHERE deptno IN (SELECT deptno FROM dept WHERE dname='SALES')
WITH CHECK OPTION;

INSERT INTO emp_on_SALES_safe(e_ID, e_Name, e_Job, e_Dept, e_Comm) VALUES (130,
'DATE', 100, 60, NULL);

SELECT * FROM EMP;
SELECT * FROM emp_on_SALES_safe;
```

Στην εικόνα 8.10 βλέπουμε ότι η απόπειρα εισαγωγής στοιχείων τα οποία παραβιάζουν τη συνθήκη WHERE της όψης αποτυγχάνει.

```
mysql> INSERT INTO emp_on_SALES_safe(e_ID, e_Name, e_Job, e_Dept, e_Comm)
    ->      VALUES (130, 'DATE', 100, 60, NULL);
ERROR 1369 (HY000): CHECK OPTION failed 'pers_view.emp_on_sales_safe'
mysql> SELECT * FROM EMP;
+-------+---------+-------+--------+--------+
| EMPNO | NAME    | JOBNO | DEPTNO | COMM   |
+-------+---------+-------+--------+--------+
|    10 | CODD    |   100 |     50 |   NULL |
|    20 | NAVATHE |   200 |     50 | 450.00 |
|    30 | ELMASRI |   300 |     60 |   NULL |
|    40 | DATE    |   100 |     50 |   NULL |
|    50 | CODD    |   100 |     50 |   NULL |
|    60 | CODD    |   200 |     50 | 450.00 |
|    70 | CODD    |   200 |     60 | 500.00 |
|    80 | CODD    |   100 |     60 |   NULL |
|    90 | CLARKE  |   100 |     50 |   NULL |
|   100 | ADAMS   |   200 |     60 |   NULL |
|   110 | NAVATHE |   100 |     60 |   NULL |
|   120 | ELMASRI |   100 |     60 |   NULL |
+-------+---------+-------+--------+--------+
12 rows in set (0.00 sec)

mysql> SELECT * FROM emp_on_SALES_safe;
+------+---------+-------+--------+--------+
| e_ID | e_Name  | e_Job | e_Dept | e_Comm |
+------+---------+-------+--------+--------+
|   10 | CODD    |   100 |     50 |   NULL |
|   20 | NAVATHE |   200 |     50 | 450.00 |
|   40 | DATE    |   100 |     50 |   NULL |
|   50 | CODD    |   100 |     50 |   NULL |
|   60 | CODD    |   200 |     50 | 450.00 |
|   90 | CLARKE  |   100 |     50 |   NULL |
+------+---------+-------+--------+--------+
6 rows in set (0.00 sec)
```

Εικόνα 8.10 Αποτυχημένη απόπειρα εισαγωγής δεδομένων τα οποία παραβιάζουν τον ορισμό της όψης

8.2.3 Μη ενημερώσιμες όψεις.

Μία όψη είναι μη ενημερώσιμη όταν απαγορεύεται να εκτελέσουμε σε αυτήν δηλώσεις INSERT, UPDATE, DELETE. Προφανώς, όλες οι όψεις επιτρέπουν δήλωση SELECT. Παραθέτουμε παράδειγμα μη ενημερώσιμης όψης.

Παράδειγμα 1.

Στον ορισμό χρησιμοποιούμε τελεστή DISTINCT.

```
DROP VIEW IF EXISTS EMP_DISTINCT_NAMES;

CREATE VIEW EMP_DISTINCT_NAMES (NAME)
 AS SELECT DISTINCT NAME FROM EMP ORDER BY NAME;

SELECT * FROM EMP_DISTINCT_NAMES;

INSERT INTO EMP_DISTINCT_NAMES VALUES('GREEN');
```

Στην εικόνα 8.11 βλέπουμε τη μη ενημερώσιμη όψη στον ορισμό της οποίας χρησιμοποιήθηκε DISTICT.

```
mysql> CREATE VIEW EMP_DISTINCT_NAMES (NAME)
    ->  AS SELECT DISTINCT NAME FROM EMP ORDER BY NAME;
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT * FROM EMP_DISTINCT_NAMES;
+---------+
| NAME    |
+---------+
| ADAMS   |
| CLARKE  |
| CODD    |
| DATE    |
| ELMASRI |
| NAVATHE |
+---------+
6 rows in set (0.01 sec)

mysql> INSERT INTO EMP_DISTINCT_NAMES VALUES('GREEN');
ERROR 1471 (HY000): The target table EMP_DISTINCT_NAMES of the INSERT is not ins
ertable-into
mysql>
```

Εικόνα 8.11 Μη ενημερώσιμη όψη.

Παράδειγμα 2

Παραθέτουμε τον ορισμό όψης η οποία περιλαμβάνει GROUP BY. Η όψη είναι μη ενημερώσιμη.

```
DROP VIEW IF EXISTS GROUP_EMP;

CREATE VIEW GROUP_EMP (dept, count_emp, avg_comm)
 AS SELECT deptno, COUNT(*), AVG(comm) FROM EMP GROUP BY deptno;

SELECT * FROM GROUP_EMP;
```

Δοκιμάστε δήλωση INSERT για να το διαπιστώσετε, π.χ.,

```
INSERT INTO GROUP_EMP VALUES('GREEN');
```

8.2.3.1 Τι γίνεται όταν η όψη βασίζεται σε συνδέσεις πινάκων.

Όταν η όψη βασίζεται σε συνδέσεις πινάκων μπορεί να είναι ή να μην είναι ενημερώσιμη.

Παραθέτουμε τον ορισμό μη ενημερώσιμης όψης βασιζόμενης σε σύνδεση (join) πινάκων.

```
CREATE VIEW emp_dept_view(EMPNO, NAME, JOBNO, DEPTNO, DNAME)
AS SELECT empno, name, jobno, emp.deptno, dnameFROM emp INNER JOIN dept ON
emp.deptno=dept.deptno;

SELECT * FROM emp_dept_view;

INSERT INTO emp_dept_view(EMPNO, NAME, JOBNO, DEPTNO, DNAME)
VALUES (140, 'DATE', 100, 50, NULL);

SELECT * FROM emp_dept_view;
```

Στην εικόνα 8.12 βλέπουμε παράδειγμα μη ενημερώσιμης όψης της οποίας ο ορισμός περιλαμβάνει σύνδεση πινάκων.

```
mysql> CREATE VIEW emp_dept_view(EMPNO, NAME, JOBNO, DEPTNO, DNAME)
    -> AS SELECT empno, name, jobno, emp.deptno, dname
    -> FROM emp INNER JOIN dept ON emp.deptno=dept.deptno;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT * FROM emp_dept_view;
+-------+---------+-------+--------+------------+
| EMPNO | NAME    | JOBNO | DEPTNO | DNAME      |
+-------+---------+-------+--------+------------+
|    10 | CODD    |   100 |     50 | SALES      |
|    20 | NAVATHE |   200 |     50 | SALES      |
|    30 | ELMASRI |   300 |     60 | ACCOUNTING |
|    40 | DATE    |   100 |     50 | SALES      |
|    50 | CODD    |   100 |     50 | SALES      |
|    60 | CODD    |   200 |     50 | SALES      |
|    70 | CODD    |   200 |     60 | ACCOUNTING |
|    80 | CODD    |   100 |     60 | ACCOUNTING |
|    90 | CLARKE  |   100 |     50 | SALES      |
|   100 | ADAMS   |   200 |     60 | ACCOUNTING |
|   110 | NAVATHE |   100 |     60 | ACCOUNTING |
|   120 | ELMASRI |   100 |     60 | ACCOUNTING |
+-------+---------+-------+--------+------------+
12 rows in set (0.00 sec)

mysql> INSERT INTO emp_dept_view(EMPNO, NAME, JOBNO, DEPTNO, DNAME)
    -> VALUES (140, 'DATE', 100, 50, NULL);
ERROR 1393 (HY000): Can not modify more than one base table through a join view
'pers_view.emp_dept_view'
mysql>
```

Εικόνα 8.12 Μη ενημερώσιμη όψη της οποίας ο ορισμός περιλαμβάνει σύνδεση πινάκων.

Παραθέτουμε τον ορισμό δημιουργίας ενημερώσιμης όψης βασιζόμενης σε σύνδεση (join) πινάκων.

```
CREATE VIEW NEW_emp_dept_view(EMPNO, NAME, JOBNO, DEPTNO)
AS SELECT empno, name, jobno, emp.deptno
FROM emp INNER JOIN dept ON emp.deptno=dept.deptno;

SELECT * FROM NEW_emp_dept_view;

INSERT INTO NEW_emp_dept_view(EMPNO, NAME, JOBNO, DEPTNO)
VALUES (140, 'DATE', 100, 50);

SELECT * FROM NEW_emp_dept_view;
```

Στην εικόνα 8.13 βλέπουμε ενημερώσιμη όψη βασιζόμενης σε σύνδεση (join) πινάκων

```
mysql> INSERT INTO NEW_emp_dept_view(EMPNO, NAME, JOBNO, DEPTNO)
    -> VALUES (140, 'DATE', 100, 50);
Query OK, 1 row affected (0.04 sec)

mysql> SELECT * FROM NEW_emp_dept_view;
+-------+---------+-------+--------+
| EMPNO | NAME    | JOBNO | DEPTNO |
+-------+---------+-------+--------+
|    10 | CODD    |   100 |     50 |
|    20 | NAVATHE |   200 |     50 |
|    30 | ELMASRI |   300 |     60 |
|    40 | DATE    |   100 |     50 |
|    50 | CODD    |   100 |     50 |
|    60 | CODD    |   200 |     50 |
|    70 | CODD    |   200 |     60 |
|    80 | CODD    |   100 |     60 |
|    90 | CLARKE  |   100 |     50 |
|   100 | ADAMS   |   200 |     60 |
|   110 | NAVATHE |   100 |     60 |
|   120 | ELMASRI |   100 |     60 |
|   140 | DATE    |   100 |     50 |
+-------+---------+-------+--------+
13 rows in set (0.00 sec)
```

Εικόνα 8.13 Ενημερώσιμη όψη βασιζόμενη σε σύνδεση (join) πινάκων

8.2.3.2 Πότε μια όψη (view) δεν είναι ενημερώσιμη

Μια όψη (view) δεν είναι ενημερώσιμη όταν ο ορισμός της όψης περιλαμβάνει:

1) Δήλωση Select με πράξεις, π.χ.

```
  CREATE VIEW …
 AS SELECT sal+IFNULL(comm,0) …
```

2) Συναρτήσεις ομαδοποίησης-αθροιστικές (Aggregate functions), όπως SUM(), MIN(), MAX(), COUNT(), … Ακολουθεί παράδειγμα.

```
  CREATE VIEW …
 AS SELECT avg(sal+IFNULL(comm,0)) …
```

3) Τελεστή DISTINCT. Ακολουθεί παράδειγμα.

```
  CREATE VIEW …
 AS SELECT DISTINCT job …
```

4) Τελεστή GROUP BY . Ακολουθεί παράδειγμα.

```
  CREATE VIEW …
 AS SELECT …. GROUP BY …
```

5) Τελεστή GROUP BY …. HAVING … Ακολουθεί παράδειγμα.

```
  CREATE VIEW …
 AS SELECT … GROUP BY …. HAVING …
```

6) Τελεστή UNION. Ακολουθεί παράδειγμα.

```
  CREATE VIEW …
 AS SELECT ….
UNION
SELECT ….
```

7) Υποαναζήτηση (Subquery) η οποία περιλαμβάνει SELECT με πράξεις κ.λπ. στη λίστα της (in the select list). Ακολουθεί παράδειγμα.

```
  CREATE VIEW …
 AS SELECT ….
  (SELECT sal+IFNULL(comm,0) ….) ….
```

8) Σε κάποιες περιπτώσεις συνδέσεων (Certain joins). Αναφέραμε ήδη παραδείγματα
9) Μη ενημερώσιμη όψη (Non updatable view) στην υποπρόταση FROM (FROM clause).
10) Έστω δήλωση SELECT της μορφής,

```
SELECT ….
 FROM TABLE_1, …
 WHERE ….( φωλιασμένη δήλωση SELECT … ) ….
```

Η φωλιασμένη δήλωση SELECT έχει τη μορφή,

```
SELECT ….
 FROM TABLE_1, …
 WHERE ….(
```

δηλαδή η φωλιασμένη δήλωση αναφέρεται στον ίδιο πίνακα TABLE_1.

Τότε μία όψη που θα οριστεί με βάση τη δήλωση SELECT που περιλαμβάνει τη φωλιασμένη δήλωση δεν είναι ενημερώσιμη.

8.3 Triggers by example–Η περίπτωση του προϊόντος Oracle. Χρήση τεχνολογίας PL/SQL. Εισαγωγή στον προγραμματισμό με χρήση εναυσμάτων (trigger).

Στόχος της ενότητας είναι βοηθήσει τους αναγνώστες να κατανοήσουν και να εμπεδώσουν κρίσιμα σημεία της τεχνολογίας των εναυσμάτων (trigger) και να μάθουν να κατασκευάζουν και να χρησιμοποιούν εναύσματα σύμφωνα με τις ανάγκες των εφαρμογών βάσεων δεδομένων στο περιβάλλον Oracle PL/SQL.

Έστω ότι η βάση προσωπικού εταιρείας περιλαμβάνει πίνακα με στοιχεία υπαλλήλων (emp) και πίνακα στοιχεία τμημάτων (dept). Οι πίνακες emp, dept παρατίθενται στην εικόνα 8.14 με ενδεικτικό δείγμα δεδομένων.

Παραθέτουμε τις στήλες του πίνακα υπαλλήλων EMP.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
κωδικός	Όνομα	Θέση	Επικεφαλής	Ημερομηνία πρόσληψης	Μισθός	Προμήθεια	Κωδικός τμήματος

Παραθέτουμε τις στήλες του πίνακα τμημάτων DEPT.

DEPTNO	DNAME	LOC
Κωδικός τμήματος	Όνομα	Έδρα

Ο πίνακας DEPT έχει κύριο κλειδί τη στήλη deptno.

Ο πίνακας EMP έχει κύριο κλειδί τη στήλη empno και ξένο κλειδί τη στήλη deptno.

Δημιουργία των πινάκων emp, dept.

Για να δημιουργήσετε τους πίνακες μπορείτε να χρησιμοποιήσετε τις παρακάτω δηλώσεις.

```
/* Κατασκευάστε τους πίνακες  Emp, Dept */
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(10), JOB VARCHAR2(9), 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));
```

Στην εικόνα 8.14 παραθέτουμε τους πίνακες DEPT, EMP με δείγμα δεδομένων που θα χρησιμοποιήσουμε στα παραδείγματα της ενότητας.

Πίνακας τμημάτων DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK

20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

πίνακας υπαλλήλων EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800	-	20
7499	ALLEN	SALESMAN	7698	20/02/81	1600	300	30
7521	WARD	SALESMAN	7698	22/02/81	1250	500	30
7566	JONES	MANAGER	7839	02/04/81	2975	-	20
7654	MARTIN	SALESMAN	7698	28/10/81	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/81	2850	-	30
7782	CLARK	MANAGER	7839	09/06/81	2450	-	10
7788	SCOTT	ANALYST	7566	19/04/87	3000	-	20
7839	KING	PRESIDENT	-	17/11/81	5000	-	10
7844	TURNER	SALESMAN	7698	08/10/81	1500	0	30
7876	ADAMS	CLERK	7788	23/05/87	1100	-	20
7900	JAMES	CLERK	7698	03/12/81	950	-	30
7902	FORD	ANALYST	7566	03/12/81	3000	-	20
7934	MILLER	CLERK	7782	23/01/82	1300	-	10
7999	BATES	ANALYST	7566	23/01/04	1300	-	-

Εικόνα 8.14 Πίνακες της βάσης δεδομένων προσωπικού με δείγμα δεδομένων

Εισαγωγή στοιχείων στους πίνακες

Για να εισάγετε δεδομένα στους πίνακες μπορείτε να χρησιμοποιήσετε 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 (7369, 'SMITH', 'CLERK', 7902, '17/12/1980', 800, NULL, 20);
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7499, 'ALLEN', 'SALESMAN', 7698, '20/02/1981', 1600, 300, 30);
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7521, 'WARD', 'SALESMAN', 7698, '22/02/1981', 1250, 500, 30);
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7566, 'JONES', 'MANAGER', 7839, '02/04/1981', 2975, NULL, 20);
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7654, 'MARTIN', 'SALESMAN', 7698, '28/10/1981', 1250, 1400, 30);
```

```
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7698, 'BLAKE', 'MANAGER', 7839, '01/05/1981', 2850, NULL, 30);
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7782, 'CLARK', 'MANAGER', 7839, '09/06/1981', 2450, NULL, 10);
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7788, 'SCOTT', 'ANALYST', 7566, '19/04/1987', 3000, NULL, 20);
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7839, 'KING', 'PRESIDENT', NULL, '17/11/1981', 5000, NULL, 10);
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7844, 'TURNER', 'SALESMAN', 7698, '08/10/1981', 1500, 0, 30);
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7876, 'ADAMS', 'CLERK', 7788, '23/05/1987', 1100, NULL, 20);
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7900, 'JAMES', 'CLERK', 7698, '03/12/1981', 950, NULL, 30);
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7902, 'FORD', 'ANALYST', 7566, '03/12/1981', 3000, NULL, 20);
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7934, 'MILLER', 'CLERK', 7782, '23/01/1982', 1300, NULL, 10);
INSERT INTO Emp(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7999, 'BATES', 'ANALYST', 7566, '23/01/2004', 1300, NULL, NULL);

/* τέλος εντολών  */
```

8.3.1 Δημιουργία νέων πινάκων οι οποίοι βασίζονται σε υπάρχοντες πίνακες και αντιγραφή των δεδομένων στους νέους πίνακες

Κατασκευάστε τους πίνακες department, employee που έχουν ανάλογη γραμμογράφηση με τους πίνακες emp, dept και αντιγραφή των δεδομένων στους νέους πίνακες.

```
CREATE TABLE department(deptno NUMBER(2) NOT NULL,
 dname VARCHAR2(14));
```

Με την επόμενη δήλωση INSERT INTO μπορείτε να «φορτώσετε» από τον πίνακα dept δεδομένα στον νέο πίνακα.

```
INSERT INTO department SELECT deptno, dname FROM dept;

SELECT * FROM department;
```

DEPTNO	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS

```
CREATE TABLE employee(empno NUMBER(4) NOT NULL,
ename VARCHAR2(10), deptno NUMBER(2));
```

Εισαγωγή στοιχείων στον πίνακα employee. Η χρήση της συνθήκης (WHERE) γίνεται για να δούμε τις δυνατότητες που έχουμε σε δήλωση INSERT … SELECT.

Με την επόμενη δήλωση μπορείτε να «φορτώσετε» από τον πίνακα emp δεδομένα στον νέο πίνακα.

```
INSERT INTO employee
 SELECT empno, ename, deptno
 FROM emp
 WHERE empno>7700;
SELECT empno, ename, deptno FROM employee;
```

EMPNO	ENAME	DEPTNO
7782	CLARK	10
7788	SCOTT	20
7839	KING	10
7844	TURNER	30
7876	ADAMS	20
7900	JAMES	30
7902	FORD	20
7934	MILLER	10
7999	BATES	

8.3.2 Έναυσμα (trigger) για την εισαγωγή ή την ενημέρωση των ονομάτων με κεφαλαία γράμματα στους πίνακες της βάσης δεδομένων.

Ακολουθεί παράδειγμα εναύσματος (trigger) που αναλαμβάνει την εισαγωγή ή την ενημέρωση των ονομάτων των τμημάτων με κεφαλαία γράμματα στον πίνακα dept της βάσης δεδομένων όπως και να τα πληκτρολογεί ο χρήστης σε δήλωση INSERT ή UPDATE . Το έναυσμα trigger dept_insert_update είναι αποθηκευμένο (stored) και εκτελείται πριν από την εισαγωγή ή την ενημέρωση των στοιχείων του πίνακα department και μεταγράφει τα στοιχεία αυτά με κεφαλαία γράμματα.

```
CREATE OR REPLACE TRIGGER dept_insert_update
BEFORE INSERT OR UPDATE ON department
FOR EACH ROW
BEGIN
DBMS_OUTPUT.PUT_LINE('Εκτέλεση trigger dept_insert_update');
:NEW.dname := UPPER(:NEW.dname);
/* :NEW.dname contains the new value */
END;
/
```

8.3.3 Δοκιμές και επεξήγηση της λειτουργίας εναύσματος

Παραθέτουμε δοκιμές του εναύσματος trigger dept_insert_update κατά την εισαγωγή και κατά την ενημέρωση στοιχείων τμημάτων.

```
/* testing */
INSERT INTO department VALUES(70, 'Learn');
1 row created.
```

DEPTNO	DNAME
10	ACCOUNTING

20	RESEARCH
30	SALES
40	OPERATIONS
70	LEARN

8.3.3.1 Επεξήγηση της λειτουργίας εναύσματος (trigger) που ενεργοποιείται από δήλωση INSERT

Παραθέτουμε επεξήγηση του εναύσματος (trigger) dept_insert_update:

1) Δίδεται η δήλωση INSERT INTO department VALUES(70, 'Learn');
2) Λόγω του γεγονότος "BEFORE INSERT OR UPDATE ON department" που υπάρχει στον ορισμό του «αφυπνίζεται» ο trigger dept_insert_update (ο ορισμός του υπάρχει στο λεξικό δεδομένων).
3) Επειδή δόθηκε δήλωση INSERT και ο trigger είναι row type (δες και υποπρόταση FOR EACH ROW) τα ζεύγη των global μεταβλητών που αντιστοιχούν στις στήλες του πίνακα Department έχουν τις παρακάτω τιμές:

   :OLD.deptno < - - NULL, :OLD.dname < -- NULL

   :NEW.deptno < - - 70, :NEW.dname < -- 'Learn'
4) Λόγω της εντολής :NEW.dname := UPPER(:NEW.dname);

   Ο trigger μεταγράφει σε κεφαλαία την τιμή της μεταβλητής :NEW.dname.
5) Ο trigger τερματίζεται και εκτελείται η δήλωση:

   INSERT INTO department VALUES(70, 'LEARN');

```
/* testing */
UPDATE department
SET dname = 'Payroll
WHERE deptno=70;
```

DEPTNO	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS
70	PAYROLL

8.3.3.2 Επεξήγηση της λειτουργίας εναύσματος (trigger) που ενεργοποιείται από δήλωση UPDATE

Παραθέτουμε επεξήγηση του εναύσματος (trigger) trigger dept_insert_update:

1) Δίδεται η δήλωση,

```
UPDATE department SET dname = 'Payroll'
WHERE deptno=70;
```

2) Λόγω του γεγονότος "BEFORE INSERT OR UPDATE ON department" που υπάρχει στον ορισμό του «αφυπνίζεται» ο trigger dept_insert_update (υπάρχει στο λεξικό δεδομένων).

3) Επειδή δόθηκε εντολή UPDATE και ο trigger είναι row type (δες και υποπρόταση FOR EACH ROW) τα ζεύγη των global μεταβλητών που αντιστοιχούν στις στήλες του πίνακα Department έχουν τις παρακάτω τιμές:

   :OLD.deptno < - - 70, :OLD.dname < -- 'Learn'

   :NEW.deptno < - - 70, :NEW.dname < -- 'Payroll'

4) Λόγω της εντολής :NEW.dname := UPPER(:NEW.dname);

   Ο trigger μεταγράφει σε κεφαλαία την τιμή της μεταβλητής :NEW.dname.

5) Ο trigger τερματίζεται και εκτελείται η εντολή:

```
UPDATE department SET dname = 'PAYROLL'
WHERE deptno=70;
```

8.3.4 Προσθήκη στήλης αριθμού υπαλλήλων σε πίνακα τμήματος και ενημέρωση των τιμών της στήλης με δήλωση UPDATE.

Στο παράδειγμα μας γίνεται προσθήκη της στήλης no_of_employees (αριθμός υπαλλήλων που έχει το τμήμα) στον πίνακα department. Στη συνέχεια υπολογίζουμε την τιμή της νέας στήλης για κάθε τμήμα μετρώντας πόσους υπάλληλους έχει το τμήμα. Η πληροφορία αυτή υπάρχει στον πίνακα των υπαλλήλων. Ο υπολογισμός γίνεται με κατάλληλη δήλωση UPDATE.

```
/* Change TABLE department */
ALTER TABLE department ADD (no_of_employees NUMBER(3));
Table altered.

SELECT * FROM department;
```

DEPTNO	DNAME		NO_OF_EMPLOYEES
10	ACCOUNTING		
20	RESEARCH		
30	SALES		
40	OPERATIONS		
70	PAYROLL		

Αρχικοποίηση της νέας στήλης με δήλωση UPDATE.

```
/* Initialization of the new column */
UPDATE department
SET no_of_employees =
  (SELECT COUNT(*)
   FROM employee
   WHERE employee.deptno = department.deptno);
```

DEPTNO	DNAME	NO_OF_EMPLOYEES
10	ACCOUNTING	3
20	RESEARCH	3
30	SALES	2

40	OPERATIONS	-
70	PAYROLL	-
-	-	1

8.3.5 Αυτοματοποίηση της διαχείρισης του περιεχομένου της στήλης αριθμός υπαλλήλων με εναύσματα

Θα μελετήσουμε την αυτοματοποίηση της διαχείρισης του περιεχομένου της στήλης αριθμός υπαλλήλων του τμήματος (No_of_Employees) με τρία εναύσματα (trigger).

8.3.5.1 Έναυσμα (Trigger) που αφυπνίζεται από το γεγονός AFTER INSERT INTO

Το έναυσμα (Trigger) emp_insert αφυπνίζεται από το γεγονός AFTER INSERT INTO employee.

```
CREATE OR REPLACE TRIGGER emp_insert
AFTER INSERT ON employee
FOR EACH ROW
BEGIN
UPDATE department
SET no_of_employees = NVL(no_of_employees, 0) + 1
WHERE  deptno= :NEW.deptno;
/* :NEW.deptno < - - νέα τιμή, :OLD.deptno < - -  NULL */
END;
/
Trigger created.
```

Δοκιμή

```
INSERT INTO employee VALUES(7985, 'NAVATHE', 10);
```

Η δήλωση ενημερώνει τον πίνακα employee

EMPNO	ENAME	DEPTNO
7782	CLARK	10
7788	SCOTT	20
7839	KING	10
7844	TURNER	30
7876	ADAMS	20
7900	JAMES	30
7902	FORD	20
7934	MILLER	10
7999	BATES	-
7985	NAVATHE	10

Επεξήγηση

Το έναυσμα τύπου γραμμής (row type trigger), βλέπε FOR EACH ROW, «αφυπνίζεται» λόγω της συνθήκης AFTER INSERT ON employee.

Λόγω της δήλωσης INSERT INTO employee VALUES(7985, 'NAVATHE', 10); έχουμε

- :OLD.empno<--NULL, :OLD.ename<--NULL, :OLD.deptno<--NULL
- :NEW.empno<-7985, :NEW.ename<-'NAVATHE', :NEW.deptno<-10

Άρα η δήλωση

```
UPDATE department
SET no_of_employees = NVL(no_of_employees, 0) + 1
WHERE  deptno= :NEW.deptno;
```

Είναι ισοδύναμη με τη δήλωση

```
UPDATE department
SET no_of_employees = NVL(no_of_employees, 0) + 1
WHERE  deptno= 10;
```

Και ο πίνακας department γίνεται

DEPTNO	DNAME	NO_OF_EMPLOYEES
10	ACCOUNTING	4
20	RESEARCH	3
30	SALES	2
40	OPERATIONS	-
70	PAYROLL	-
-	-	1

8.3.5.2 Έναυσμα (Trigger) που αφυπνίζεται από το γεγονός AFTER DELETE ON

Το έναυσμα (Trigger) emp_delete αφυπνίζεται από το γεγονός AFTER DELETE ON employee

```
CREATE OR REPLACE TRIGGER emp_delete
AFTER DELETE ON employee
FOR EACH ROW
BEGIN
UPDATE department
SET no_of_employees = NVL(no_of_employees, 0) - 1
WHERE  deptno= :OLD.deptno;
END;
/
```

Δοκιμή

Με τη δήλωση DELETE employee WHERE empno = 7985; ο πίνακας department γίνεται:

DEPTNO	DNAME	NO_OF_EMPLOYEES
10	ACCOUNTING	3

20	RESEARCH	3
30	SALES	2
40	OPERATIONS	-
70	PAYROLL	-
-	-	1

Παρατηρήστε ότι μία υπάλληλος, η Bates, δεν έχει τοποθετηθεί σε τμήμα.

8.3.5.3 Έναυσμα (Trigger) που αφυπνίζεται από το γεγονός AFTER UPDATE ON

Το έναυσμα (Trigger) emp_update αφυπνίζεται από το γεγονός AFTER UPDATE ON employee

```
CREATE OR REPLACE TRIGGER emp_update
AFTER UPDATE ON employee
FOR EACH ROW
BEGIN
/* Increments the count for the employee's new department */
UPDATE department
SET no_of_employees = NVL(no_of_employees, 0) + 1
WHERE  deptno= :NEW.deptno;

/* Decreases the count for the employee's old department */

UPDATE department
SET no_of_employees = no_of_employees - 1
WHERE  deptno= :OLD.deptno;
END;
/
```

Δοκιμή

```
UPDATE employee
SET deptno= 10
WHERE empno=20;
```

DEPTNO	DNAME	NO_OF_EMPLOYEES
10	ACCOUNTING	6
20	RESEARCH	-
30	SALES	2
40	OPERATIONS	-
70	PAYROLL	-
-	-	1

Ακολουθούν θέματα χρήσιμα για την καλύτερη διαχείριση των εναυσμάτων.

Θέμα 1 Περιγραφή εναυσμάτων στο λεξικό δεδομένων

Ακολουθεί η περιγραφή των εναυσμάτων στο λεξικό δεδομένων και λίστα εναυσμάτων που ενεργοποιούνται από ενέργειες (εκτέλεση δηλώσεων) στον πίνακα των υπαλλήλων.

```
/* see triggers */
DESCRIBE USER_TRIGGERS;

Name   Type
 ----------------------------------------------------
 TRIGGER_NAME VARCHAR2(30)
 TRIGGER_TYPE   VARCHAR2(16)
 TRIGGERING_EVENT VARCHAR2(75)
 TABLE_OWNER   VARCHAR2(30)
 BASE_OBJECT_TYPE  VARCHAR2(16)
 TABLE_NAME   VARCHAR2(30)
 COLUMN_NAME  VARCHAR2(4000)
 REFERENCING_NAMES  VARCHAR2(128)
 WHEN_CLAUSEVARCHAR2(4000)
 STATUS   VARCHAR2(8)
 DESCRIPTIONVARCHAR2(4000)
 ACTION_TYPE   VARCHAR2(11)
 TRIGGER_BODYLONG

SELECT TRIGGER_NAME, TRIGGERING_EVENT, TRIGGER_TYPE
FROM USER_TRIGGERS
WHERE TABLE_NAME= 'EMPLOYEE'
ORDER BY TRIGGER_NAME;
```

```
TRIGGER_NAMETRIGGERING_EVENT  TRIGGER_TYPE
-------------------------------------------------------------------------------
-
EMP_DELETE DELETE  AFTER EACH ROW
EMP_INSERT  INSERT   AFTER EACH ROW
EMP_UPDATEUPDATE  AFTER EACH ROW
```

Πως διαγράφουμε εναύσματα

```
/* Drop all the database objects */

DROP TRIGGER dept_insert_update;
Trigger dropped.
DROP TRIGGER emp_insert;
Trigger dropped.
DROP TRIGGER emp_delete;
Trigger dropped.
DROP TRIGGER emp_update;
Trigger dropped.
DROP TABLE employee;
Table dropped.
DROP TABLE department;
Table dropped.
```

Πως βλέπουμε τα λάθη όταν προγραμματίζουμε ένα έναυσμα

```
SHOW ERRORS TRIGGER emp_insert;
```

Ενεργοποίηση/απενεργοποίηση ενός εναύσματος (trigger_

```
ALTER TRIGGER emp_insert DISABLE;
ALTER TRIGGER emp_insert ENABLE;
ALTER TABLE my_emp DISABLE ALL TRIGGERS;
ALTER TRIGGER emp_insert COMPILE;
DROP TRIGGER emp_insert;
```

8.3.6 Κάποια ενδιαφέροντα θέματα διαχείρισης των εναυσμάτων στο προϊόν της Oracle

Ακολουθούν θέματα.

Θέμα 1.

Πως θα δούμε τις εντολές στο «σώμα» του εναύσματος (trigger_body).

```
SELECT trigger_name,trigger_body
FROM USER_TRIGGERS
WHERE table_name= 'MY_EMP'
ORDER BY trigger_name;
/
```

Θέμα 2

Πως θα περιορίσουμε τη χρήση της βάσης για συγκεκριμένες ώρες.

CREATE OR REPLACE TRIGGER only_during_my_hours

```
BEFORE INSERT OR UPDATE OR DELETE ON my_emp
BEGIN
  DBMS_OUTPUT.PUT_LINE('only_during_my_hours trigger');
  IF   TO_NUMBER(TO_CHAR(SYSDATE,'hh24')) < 8
/* nothing before 8:00 am */
OR TO_NUMBER(TO_CHAR(SYSDATE,'hh24')) >= 5
/* changes must be made before 5:00 pm */
OR TO_CHAR(SYSDATE,'dy') IN ('sun')  THEN
/* nothing on Sunday */
   RAISE_APPLICATION_ERROR(-20000,
 'Αλλαγές μόνο τις ώρες που θέλουμε');
  END IF;
END;
/
```

8.4 Triggers by example–Η περίπτωση του προϊόντος MySQL

Η συζήτηση του θέματος θα γίνει με τη μορφή περιήγησης στην περίπτωση του προϊόντος MySQL (Tour on triggers. The case of MySQL).

Έστω η βάση προσωπικού my_first_triggers_db που περιλαμβάνει τους πίνακες emp, dept.

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

Σύντομη περιγραφή

Ακολουθεί μια περιήγηση σε αφυπνιζόμενα αποθηκευμένα προγράμματα (εναύσματα, triggers), δηλαδή προγράμματα που είναι αποθηκευμένα στη βάση δεδομένων του προϊόντος MySQL και ενεργοποιούνται και εκτελούνται πριν ή μετά από δηλώσεις INSERT, UPDATE, DELETE στη βάση δεδομένων.

Ο όρος trigger έχει αποδοθεί στα ελληνικά με πολλούς τρόπους: Έναυσμα, ενεργοποίηση, σκανδάλη, σκανδαλισμος, αφυπνιζόμενο (αποθηκευμένο) πρόγραμμα, ενεργοποιούμενο (αποθηκευμένο) πρόγραμμα.

Στόχος της ενότητας είναι να βοηθήσει τους αναγνώστες να κατανοήσουν και να εμπεδώσουν κρίσιμα σημεία της τεχνολογίας των εναυσμάτων (trigger) και να μάθουν να κατασκευάζουν και να χρησιμοποιούν εναύσματα σύμφωνα με τις ανάγκες των εφαρμογών βάσεων δεδομένων σε mySQL.

Δημιουργία της βάσης δεδομένων my_first_triggers_db και των πινάκων της Dept, Emp

```
DROP DATABASE my_first_triggers_db;

CREATE DATABASE my_first_triggers_db;

USE my_first_triggers_db;

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', 'ATHENS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC) VALUES (20, 'SALES', 'LONDON');
INSERT INTO DEPT(DEPTNO, DNAME, LOC) VALUES (30, 'RESEARCH', 'ATHENS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC) VALUES (40, 'PAYROLL', 'LONDON');
```

```
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;
```

8.4.1 Δημιουργία πίνακα που βασίζεται σε υπάρχοντες πίνακες

Αντί να εργαστούμε απευθείας στους πίνακες emp, dept ορίζοντας triggers κατασκευάζουμε πρώτα τους πίνακες Employee, Department βασιζόμενοι στη δομή των πινάκων emp, dept και εισάγουμε τα δεδομένα στους νέους πίνακες με δηλώσεις:

```
SELECT … INSERT ….;
```

Στη συνέχεια, δημιουργούμε στον πίνακα Department τα εναύσματα dept_insert, dept_update για να συζητήσουμε βασικές έννοιες των row type triggers και να κατανοήσουμε τη χρησιμότητά τους.

Στη συνέχεια, προσθέτουμε στον πίνακα Department τη στήλη no_of_employees (αριθμός υπαλλήλων που έχει το τμήμα). Θα «γεμίσουμε» τη στήλη no_of_employees γράφοντας κατάλληλη δήλωση UPDATE. Όποτε εκτελέσουμε τη δήλωση UPDATE ενημερώνεται το περιεχόμενο του πίνακα Department.

Στη συνέχεια, δημιουργούμε στον πίνακα Employee τα εναύσματα emp_insert, emp_delete, emp_update για να διαχειριστούμε αυτόματα το περιεχόμενο της στήλης no_of_employees του πίνακα Department και να κατανοήσουμε τη χρησιμότητά τους στην αυτοματοποίηση διαχείρισης δεδομένων. Για παράδειγμα, θα κατασκευάσουμε το έναυσμα emp_insert που ενεργοποιείται όταν εισάγουμε γραμμές με στοιχεία υπαλλήλων στον πίνακα Employee. Το έναυσμα emp_insert αναλαμβάνει την αυτόματη ενημέρωση της τιμής της στήλης no_of_employees του πίνακα Department που αντιστοιχεί στο τμήμα στο οποίο εισάγεται ο κάθε υπάλληλος.

Τέλος, σαν άσκηση αφήνεται η δημιουργία αντίστοιχων εναυσμάτων στους πίνακες Dept, Emp.

Δημιουργία πινάκων Employee, Department που βασίζονται στους υπάρχοντες πίνακες Dept, Emp

```
CREATE TABLE employee(empno INT(4) NOT NULL, ename VARCHAR(10),
  deptno INT(2));
CREATE TABLE department(deptno INT(2) NOT NULL, dname VARCHAR(14));
INSERT INTO department SELECT deptno, dname FROM   dept;
INSERT INTO employee SELECT empno, ename, deptno FROM   emp;
SELECT * FROM department;
SELECT * FROM employee;
```

Employee (νέος πίνακας υπαλλήλων)

Empno	Ename	Deptno
10	CODD	10
15	ELMASRI	10
20	NAVATHE	20

30	DATE	10

Departmnet (νέος πίνακας τμημάτων)

Deptno	Dname
10	ACCOUNTING
20	SALES
30	RESEARCH
40	PAYROLL

8.4.2 Δημιουργία εναυσμάτων (trigger) στο προϊόν MySQL για την εισαγωγή ονομάτων με κεφαλαία γράμματα

Θα περιγράψουμε πως θα εισάγουμε στη βάση το όνομα του τμήματος με κεφαλαία γράμματα, όπως και να το πληκτρολογεί ο χρήστης, με τον ορισμό του εναύσματος (trigger) dept_insert.

8.4.2.1 Χρήση Delimiter για τον ορισμό εναύσματος

Η επικοινωνία client και server στο προϊόν MySQL βασίζεται σε εντολές που τελειώνουν με τον χαρακτήρα ερωτηματικό (**;**). Για παράδειγμα, γράφουμε τη δήλωση SELECT * FROM department; και ο server καταλαβαίνει ότι ολοκληρώθηκε η δήλωση, ελέγχει αν είναι σωστή, την εκτελεί και επιστρέφει τα αποτελέσματα.

Αν θέλουμε να ορίσουμε trigger/procedure/function δηλώνουμε κάποιο delimiter έτσι ώστε να καταλάβει ο server ότι θα ακολουθήσει ο ορισμός του αντικειμένου (trigger/procedure/function). Η ολοκλήρωση του ορισμού γίνεται όταν (ξανα)γράψουμε το delimiter που ορίσαμε. Τότε ο server ελέγχει τον ορισμό του αντικειμένου και αν είναι ορθός δημιουργεί το αντικείμενο και το «αποθηκεύει» στη βάση δεδομένων του συστήματος. Στον πίνακα 8.1 βλέπουμε παράδειγμα χρήσης Delimiter,

Πίνακας 8.1 Παράδειγμα χρήσης Delimiter

delimiter //	Όρισα ως delimiter τους χαρακτήρες //

CREATE TRIGGER dept_insert
BEFORE INSERT ON department
FOR EACH ROW
BEGIN
SET NEW.dname = UPPER(NEW.dname);
END;
	Ορίζω το αντικείμενο (trigger) dept_insert

//	Ο ορισμός ολοκληρώθηκε, είναι σωστός, και ο server δημιουργεί τον trigger
delimiter ;	Αλλάζω delimiter ώστε η επικοινωνία client και server να γίνεται όπως πριν. Ο server θα περιμένει πάλι τον χαρακτήρα ; για να καταλάβει ότι ολοκληρώθηκε μία εντολή.

Ως delimiter θα μπορούσαμε να χρησιμοποιήσουμε χαρακτήρες ή ακολουθίες χαρακτήρων όπως $, $$, !, /, // κ.λπ. Θυμηθείτε να αφήνετε κενό ανάμεσα στη λέξη delimiter και τον χαρακτήρα. Για παράδειγμα, ποτέ μην γράφετε delimiter//

Ακολουθεί ο ορισμός του εναύσματος.

```
delimiter //
CREATE TRIGGER dept_insert
BEFORE INSERT ON department
FOR EACH ROW
BEGIN
SET NEW.dname = UPPER(NEW.dname);
END;
//
delimiter ;
```

8.4.2.2 Δοκιμή και επεξήγηση εναύσματος (trigger)

Ακολουθεί δοκιμή και επεξήγηση του εναύσματος dept_insert:

1) Δίδεται η δήλωση INSERT INTO department VALUES(70, 'Learn');
2) Λόγω του γεγονότος "BEFORE INSERT ON department" που υπάρχει στον ορισμό του «αφυπνίζεται» ο trigger dept_insert (ο ορισμός του υπάρχει στη βάση δεδομένων του συστήματος, δηλαδή στη βάση Information_schema).
3) Επειδή δόθηκε δήλωση INSERT και ο trigger είναι row type (δες και υποπρόταση FOR EACH ROW) τα ζεύγη των global μεταβλητών που αντιστοιχούν στις στήλες του πίνακα Department έχουν τις παρακάτω τιμές:

   OLD.deptno < - - NULL, OLD.dname < -- NULL

   NEW.deptno < - - 70, NEW.dname < -- 'Learn'
4) Λόγω της εντολής SET NEW.dname = UPPER(NEW.dname); ο trigger μεταγράφει σε κεφαλαία την τιμή της μεταβλητής NEW.dname.
5) Ο trigger τερματίζεται και εκτελείται η δήλωση:

   INSERT INTO department VALUES(70, 'LEARN');
6) SELECT * FROM department;

Department (πίνακας τμημάτων)

Deptno	Dname
10	ACCOUNTING
20	SALES
30	RESEARCH
40	PAYROLL
70	**LEARN**

8.4.3 Δημιουργία εναύσματος για την καταχώρηση των ονομάτων στη βάση με κεφαλαία γράμματα

Θα εξηγήσουμε πως θα εισάγουμε στη βάση το όνομα του τμήματος με κεφαλαία γράμματα, όπως και να το εισάγει ο χρήστης σε δήλωση UPDATE (trigger dept_update).

Το έναυσμα που θα γράψουμε λέγεται dept_update.

```
delimiter //
CREATE TRIGGER dept_update
```

```
BEFORE UPDATE ON department
FOR EACH ROW
BEGIN
SET NEW.dname = UPPER(NEW.dname);
END;
//
delimiter ;
```

Ακολουθεί δοκιμή του εναύσματος dept_update

```
/* testing */
UPDATE department SET dname = 'Operations' WHERE deptno=70;
SELECT * FROM department;
```

Department (πίνακας τμημάτων)

Deptno	Dname
10	ACCOUNTING
20	SALES
30	RESEARCH
40	PAYROLL
70	**OPERATIONS**

Επεξήγηση trigger dept_update:

Δίδεται η δήλωση

```
UPDATE department SET dname = 'Operations'
WHERE deptno=70;
```

Λόγω του γεγονότος "BEFORE UPDATE ON department" που υπάρχει στον ορισμό του «αφυπνίζεται» ο trigger dept_update (υπάρχει στη βάση δεδομένων του συστήματος).

Επειδή δόθηκε δήλωση UPDATE και ο trigger είναι row type (δες και υποπρόταση FOR EACH ROW) τα ζεύγη των global μεταβλητών που αντιστοιχούν στις στήλες του πίνακα Department έχουν τις παρακάτω τιμές:

```
OLD.deptno < - -  70, OLD.dname < -- 'LEARN'
NEW.deptno < - - 70, NEW.dname < -- 'Operations'
```

Λόγω της εντολής SET NEW.dname=UPPER(NEW.dname);

Ο trigger μεταγράφει σε κεφαλαία την τιμή της μεταβλητής NEW.dname.

Ο trigger τερματίζεται και εκτελείται η δήλωση:

```
UPDATE department SET dname = 'OPERATIONS'
WHERE deptno=70;
```

8.4.4 Προσθήκη στήλης αριθμού υπαλλήλων στο τμήμα και συμπλήρωση των τιμών της με δήλωση UPDATE

Στον πίνακα Department θα προσθέσουμε τη στήλη no_of_employees (αριθμός υπαλλήλων που έχει το τμήμα). Στη συνέχεια, θα «γεμίσουμε» τη στήλη no_of_employees γράφοντας κατάλληλη δήλωση UPDATE.

```
ALTER TABLE department ADD (no_of_employees INT(3));
/* Initialization of the new column */

UPDATE department
SET no_of_employees =
  (SELECT COUNT(*)
   FROM employee
   WHERE employee.deptno = department.deptno);
SELECT * FROM department;
```

Department (πίνακας τμημάτων)

Deptno	Dname	No_of_employees
10	ACCOUNTING	3
20	SALES	1
30	RESEARCH	0
40	PAYROLL	0
70	OPERATIONS	0

8.4.5 Διαχείριση με εναύσματα της στήλης αριθμού υπαλλήλων στον πίνακα του τμήματος

Στη συνέχεια θα γράψουμε τρεις triggers για την αυτοματοποίηση της διαχείρισης του περιεχομένου της στήλης No_of_employees.

8.4.5.1 Έναυσμα (Trigger) που αφυπνίζεται από το γεγονός AFTER INSERT ON

Θα συζητήσουμε το έναυσμα (trigger) emp_insert που αφυπνίζεται από το γεγονός AFTER INSERT ON employee.

```
delimiter //
CREATE TRIGGER emp_insert
AFTER INSERT ON employee
FOR EACH ROW
BEGIN
UPDATE department
SET no_of_employees = IFNULL(no_of_employees, 0) + 1
WHERE  deptno= NEW.deptno;
END;
//
delimiter ;
```

Δοκιμή trigger emp_insert

```
INSERT INTO employee VALUES(7985, 'CLARKE', 10);
SELECT * FROM employee;
SELECT * FROM department;
```

Employee (πίνακας υπαλλήλων)

Empno	Ename	Deptno
10	CODD	10
15	ELMASRI	10
20	NAVATHE	20
30	DATE	10
7985	**CLARKE**	**10**

Department (πίνακας τμημάτων)

Deptno	Dname	No_of_employees
10	ACCOUNTING	**4**
20	SALES	1
30	RESEARCH	0
40	PAYROLL	0
70	OPERATIONS	0

Επεξήγηση trigger emp_insert

Ο row type trigger (βλέπε FOR EACH ROW) «αφυπνίζεται» λόγω της συνθήκης AFTER INSERT ON employee.

Λόγω της δήλωσης INSERT INTO employee VALUES(7985, 'CLARKE', 10); έχω:

OLD.empno<--NULL, OLD.ename<--NULL, OLD.deptno<--NULL

NEW.empno<-7985, NEW.ename<-'CLARKE', NEW.deptno<-10

Άρα η δήλωση

```
UPDATE department
SET no_of_employees = IFNULL(no_of_employees,0)+1
WHERE  deptno= NEW.deptno;
```

είναι ισοδύναμη με τη δήλωση

```
UPDATE department
SET no_of_employees = IFNULL(no_of_employees,0)+1
WHERE  deptno=10;
```

8.4.5.2 Έναυσμα (Trigger) που αφυπνίζεται από το γεγονός AFTER DELETE ON

Θα συζητήσουμε το έναυσμα (trigger) emp_delete που αφυπνίζεται από το γεγονός AFTER DELETE ON employee.

```
delimiter //
CREATE TRIGGER emp_delete
AFTER DELETE ON employee
FOR EACH ROW
BEGIN
UPDATE department
SET no_of_employees = IFNULL(no_of_employees, 0) - 1
WHERE  deptno= OLD.deptno;
END;
//
delimiter ;

DELETE FROM employee WHERE empno = 7985;
SELECT * FROM department;
SELECT * FROM employee;
```

Employee (πίνακας υπαλλήλων)

Empno	Ename	Deptno
10	CODD	10
15	ELMASRI	10
20	NAVATHE	20
30	DATE	10
~~7985~~	~~CLARKE~~	~~10~~

Department (πίνακας τμημάτων)

Deptno	Dname	No_of_employees
10	ACCOUNTING	3
20	SALES	1
30	RESEARCH	0
40	PAYROLL	0
70	OPERATIONS	0

Ο row type trigger (βλέπε FOR EACH ROW) «αφυπνίζεται» λόγω της συνθήκης AFTER DELETE ON employee.

Λόγω της δήλωσης DELETE FROM employee WHERE empno = 7985; έχω:

OLD.empno<--7985, OLD.ename<--CLARKE, OLD.deptno<--10

NEW.empno<- NULL, NEW.ename<- NULL, NEW.deptno<- NULL

Άρα η δήλωση

```
UPDATE department
SET no_of_employees = IFNULL(no_of_employees, 0) - 1
WHERE  deptno= OLD.deptno;
END;
```

είναι ισοδύναμη με τη δήλωση

```
UPDATE department
SET no_of_employees = IFNULL(no_of_employees, 0) - 1
WHERE  deptno= 10;
END;
```

8.4.5.3 Έναυσμα (Trigger) που αφυπνίζεται από το γεγονός AFTER UPDATE ON

Θα περιγράψουμε το έναυσμα (Trigger) emp_update που αφυπνίζεται από το γεγονός AFTER UPDATE ON employee.

```
delimiter //
CREATE TRIGGER emp_update
AFTER UPDATE ON employee
FOR EACH ROW
BEGIN
UPDATE department
SET no_of_employees = IFNULL(no_of_employees, 0) + 1
WHERE  deptno= NEW.deptno;
UPDATE department
SET no_of_employees = IFNULL(no_of_employees, 0) - 1
WHERE  deptno= OLD.deptno;
END;
//
delimiter ;
```

Δοκιμή

```
UPDATE employee
SET deptno= 10
WHERE empno=20;
SELECT * FROM department;
SELECT * FROM employee;
```

Employee (πίνακας υπαλλήλων)

Empno	Ename	Deptno
10	CODD	10
15	ELMASRI	**10**
20	NAVATHE	10
30	DATE	10

Department (πίνακας τμημάτων)

Deptno	Dname	No_of_employees
10	ACCOUNTING	**4**
20	SALES	**0**
30	RESEARCH	0
40	PAYROLL	0
70	OPERATIONS	0

8.4.5.4 Πως βλέπουμε τα στοιχεία για τα εναύσματα (trigger) στο προϊόν MySQL

Παραθέτουμε δηλώσεις για το πως βλέπουμε τα στοιχεία για τα εναύσματα (trigger) στο προϊόν MySQL και το πως καταργούμε εναύσματα.

```
/* see triggers */
DESCRIBE Information_schema.TRIGGERS;
SELECT TRIGGER_NAME, EVENT_MANIPULATION, TRIGGER_SCHEMA
FROM INFORMATION_SCHEMA.TRIGGERS
WHERE TRIGGER_SCHEMA  = 'my_first_triggers_db'
ORDER BY TRIGGER_NAME;
```

Πως καταργούμε trigger

```
DROP TRIGGER dept_insert;
DROP TRIGGER dept_update;
DROP TRIGGER emp_insert;
DROP TRIGGER emp_delete;
DROP TRIGGER emp_update;
DROP TABLE employee;
DROP TABLE department;
```

8.5 Περιήγηση. Προγραμματισμός εφαρμογής διαχείρισης παραγγελιών με χρήση triggers σε περιβάλλον Oracle PL/SQL

Στόχοι της ενότητας είναι να βοηθήσει τους αναγνώστες να εμπεδώσουν κρίσιμα σημεία της τεχνολογίας των εναυσμάτων (trigger) και να μάθουν να κατασκευάζουν και να χρησιμοποιούν εναύσματα σε περιβάλλον PL/SQL σύμφωνα με τις ανάγκες των εφαρμογών βάσεων δεδομένων.

Έστω σύστημα διαχείρισης παραγγελιών.

Customers (στοιχεία πελάτη)

CUSTNO	CNAME	LOC
κωδικός	Επωνυμία	Έδρα
1	SMITH	ATHENS
2	JONES	VOLOS
3	BATES	NEW YORK

Stocks (στοιχεία είδους, αποθέματα φρούτων)

Stockno	Description	List_Price
κωδικός	περιγραφή	Τιμή
1	APPLE	1
2	ORANGE	1.5
3	LEMON	1,7

Orders (βασικά στοιχεία παραγγελίας)

Orderno	Custno	Odate	Total
κωδικός	Κωδικός πελάτη	ημερομηνία	Συνολικό ποσό
1	1	2022-5-21 22:37:36	17.50

Orderlines (στοιχεία γραμμών παραγγελίας)

Orderno	Stockno	Qty	Ptotal
Κωδικός παραγγελίας	Κωδικός είδους	Ποσότητα	Μερικό σύνολο
1	1	10	10.00
1	2	5	7.50

```
INSERT INTO customers(custno, cname, loc) VALUES(1, 'SMITH', 'ATHENS');
INSERT INTO customers(custno, cname, loc) VALUES(2, 'JONES', 'VOLOS');
INSERT INTO customers(custno, cname, loc) VALUES(3, 'BATES', 'NEW YORK');
INSERT INTO stocks(stockno, description, listjprice) VALUES(1, 'APPLE', 1.0);
INSERT INTO stocks(stockno, description, listjprice) VALUES(2, ORANGE', 1.5);
INSERT INTO stocks(stockno, description, listjprice) VALUES(3, 'LEMON', 1.7);
INSERT INTO orders(ordemo, custno, odate) VALUES (1,1, sysdate);

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(stockno) REFERENCES stocks(stockno));
```

8.5.1 Δημιουργία εναύσματος για τη διαχείριση του περιεχομένου στήλης ως αύξοντος αριθμού

Παραθέτουμε παράδειγμα για τη δημιουργία, δοκιμή και επεξήγηση του εναύσματος (Trigger) CustSequenceNumber για τη διαχείριση του περιεχομένου στήλης ως αύξοντος αριθμού.

Ακολουθεί η δημιουργία και αρχικοποίηση του πίνακα maxseqno που είναι η απαραίτητη δομή για τη δημιουργία του εναύσματος.

```
/* INITIALIZATION */
CREATE TABLE maxseqno
(tablename CHAR(20),
fieldname CHAR(10),
maxaccno NUMBER);
```

```
INSERT INTO maxseqno VALUES('CUSTOMERS','CUSTNO',NULL);
```

Πίνακας maxseqno

tablename	fieldname	maxaccno
Όνομα πίνακα	Όνομα στήλης πίνακα	Τρέχουσα τιμή στήλης που διαχειριζόμαστε με το έναυσμα
CUSTOMERS	CUSTNO	NULL
….	….	….

```
CREATE OR REPLACE TRIGGER CustSequenceNumber
BEFORE INSERT ON customers
FOR EACH ROW
DECLARE accno_var NUMBER;
BEGIN
UPDATE maxseqno
SET maxaccno = NVL(maxaccno, 0) + 1
WHERE TABLENAME = 'CUSTOMERS'
AND FIELDNAME = 'CUSTNO';
DBMS_OUTPUT.PUT_LINE('UPDATE maxseqno');
SELECT maxaccno INTO accno_var
FROM maxseqno
WHERE TABLENAME = 'CUSTOMERS'
AND FIELDNAME = 'CUSTNO';
DBMS_OUTPUT.PUT_LINE('ACCN0_VAR=');
DBMS_OUTPUT.PUT_LINE(accno_var);
:NEW.CUSTNO := accno_var;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('ΜΗ ΠΡΟΣΔΙΟΡΙΣΙΜΟΣ ΑΥΞΩΝ ΑΡΙΘΜΟΣ')
WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM) ;
END;
/
```

Δοκιμή

Καταχωρούμε τρεις πελάτες με δηλώσεις INSERT INTO. Παραθέτουμε το περιεχόμενο των πινάκων όπως διαμορφώνεται από τις δηλώσεις και την κλήση και εκτέλεση του εναύσματος.

Customers

CUSTNO	CNAME	LOC
1	SMITH	ATHENS
2	JONES	VOLOS
3	BATES	NEW YORK

Πίνακας maxseqno

tablename	fieldname	maxaccno
CUSTOMERS	CUSTNO	3
….	….	….

8.5.2 Κατασκευή εναύσματος (trigger) για τον αυτόματο υπολογισμό του μερικού συνόλου των γραμμών παραγγελίας

Κατασκευάζουμε το έναυσμα (trigger) για τον αυτόματο υπολογισμό του μερικού συνόλου ptotal των γραμμών παραγγελίας στον πίνακα orderliness .

```
CREATE OR REPLACE TRIGGER ptotal
BEFORE INSERT ON orderlines
FOR EACH ROW DECLARE
      ptotal_var NUMBER;
      listprice_var NUMBER;
BEGIN
SELECT list_price
INTO listprice_var
FROM stocks
WHERE stockno = :NEW.stockno;
SELECT :NEW.qty * listprice_var
INTO ptotal_var
FROM DUAL
WHERE :NEW.ptotal := ptotal_var;
END;
/
```

Δοκιμή

```
INSERT INTO orderlines(ordemo, stockno. qty) VALUES (1, 1, 10);
INSERT INTO orderlines(ordemo, stockno. qty) VALUES (1, 2, 5);
```

Παραθέτουμε το αποτέλεσμα της δοκιμής στο περιεχόμενο των πινάκων.

Stocks

STOCKNO	DESCRIPTION	LIST_PRICE
1	APPLE	1
2	ORANGE	1.5
3	LEMON	1,7

Orderliness

Orderno	Stockno	Qty	Ptotal
1	1	10	10.00
1	2	5	7.50

8.5.3 Κατασκευή εναύσματος (trigger) για τον αυτόματο υπολογισμό του συνολικού ποσού της παραγγελίας

Κατασκευάζουμε το έναυσμα (trigger) total το οποίο διαβάζει τα μερικά σύνολα ptotal των γραμμών της παραγγελίας στον πίνακα orderlines και υπολογίζει και καταχωρεί το συνολικό ποσό της παραγγελίας στον πίνακα orders.

```
CREATE OR REPLACE TRIGGER total
BEFORE INSERT ON orderlines
FOR EACH ROW
DECLARE total_var NUMBER;
BEGIN
SELECT SUM(ptotal) + :NEW.ptotal
INTO total_var
FROM orderlines
WHERE orderno = :NEW.orderno;
UPDATE orders
SET total = total_var
WHERE orderno = :NEW.orderno;
END;
/
```

Δοκιμή

```
INSERT INTO orderlines(ordemo, stockno, qty) VALUES (1, 1, 10);
INSERT INTO orderlines(ordemo, stockno, qty) VALUES (1, 2, 5);
```

Παραθέτουμε το περιεχόμενο των πινάκων μετά την εκτέλεση των δηλώσεων INSERT INTO και του εναύσματος.

orders

Orderno	Custno	Qdate	Total
1	1	2022-5-21 22:37:36	17.50

orderlines

Orderno	Stockno	Qty	Ptotal
1	1	10	10.00
1	2	5	7.50

8.6 Υλοποίηση βάσης δεδομένων διαχείρισης παραγγελιών με χρήση triggers και με τη χρήση του προϊόντος MySQL

Εργάζεστε για το online shop του φυτωρίου Athens Fruit Stocks και αναλαμβάνετε τη σχεδίαση και υλοποίηση βάσης δεδομένων διαχείρισης παραγγελιών δέντρων και φυτών. Στον πίνακα stock περιλαμβάνονται διάφορες ποικιλίες δέντρων και φυτών. Μετά την ανάλυση δεδομένων καταλήγετε στους τέσσερις παρακάτω πίνακες:

Πίνακας πελατών Customers Οι πελάτες είναι κηπουροί (Gardeners)

Κωδικός	Επωνυμία	Έδρα
Custno	Cname	Loc
INT	VARCHAR(255)	VARCHAR(255)
PRIMARY KEY		
10	SMITH	ATHENS
20	JONES	VOLOS
30	BATES	NEW YORK

Πίνακας φυτών Stocks Περιλαμβάνει αποθέματα φρούτων (ποικιλίες) (Fruit stocks-varieties)

Κωδικός	Ονομασία	Τιμή καταλόγου
Stockno	Description	List_price
INT	VARCHAR(255)	DECIMAL(5,2)
PRIMARY KEY		
101	APPLE TREE	10
102	ORANGE TREE	15
103	LEMON TREE	17

Πίνακας παραγγελίας Orders

Κωδικός	Κωδικός πελάτη	Ημερομηνία	Σύνολο
Orderno	Custno	Odate	List_price
INT	INT	DATETIME	DECIMAL(5,2)
FOREIGN KEY	PRIMARY KEY		
1	10		175

Πίνακας λεπτομερειών (γραμμών) παραγγελίας Orderlines

Κωδικός	Κωδικός φυτού	Ποσότητα	Μερικό σύνολο
Orderno	Stockno	Qty	List_price
INT	INT	INT	DECIMAL(5,2)
FOREIGN KEY	FOREIGN KEY		
1	101	10	100
1	102	5	75

PRIMARY KEY=(Orderno, stockno)

Στη συνέχεια παραθέτουμε την υλοποίηση με χρήση του προϊόντος mySQL. Κατασκευάζουμε αρχικά τη βάση δεδομένων και τους πίνακες.

```
DROP DATABASE IF EXISTS myorders;

DROP DATABASE IF EXISTS myorders;

CREATE DATABASE myorders;

USE myorders;

DROP TABLE IF EXISTS customers;
CREATE TABLE customers(custno INT, cname VARCHAR(255) NOT NULL,
   loc VARCHAR(255), PRIMARY KEY (custno) );

DROP TABLE IF EXISTS stocks;
CREATE TABLE stocks(stockno INT, description VARCHAR(255) NOT NULL,
 list_price DECIMAL(5,2) NOT NULL, PRIMARY KEY (stockno));

DROP TABLE IF EXISTS orders;
CREATE TABLE orders(orderno INT AUTO_INCREMENT, custno INT NOT NULL,
 odate DATETIME NOT NULL, total DECIMAL(5,2),
 PRIMARY KEY (orderno) );
```

```
DROP TABLE IF EXISTS orderlines;
CREATE TABLE orderlines(orderno INT, stockno INT, qty INT NOT NULL,
 ptotal DECIMAL(5,2), PRIMARY KEY (orderno, stockno));
```

Να πως θα δούμε τους πίνακες.

```
SHOW TABLES;
```

Να πως θα δούμε τις στήλες των πινάκων.

```
DESCRIBE customers;
DESCRIBE stocks;
DESCRIBE orders;
DESCRIBE orderlines;
```

Φροντίστε να εισάγονται στη βάση δεδομένων όλα τα ονόματα, πελατών και ειδών με κεφαλαία

```
delimiter //
CREATE TRIGGER insert_customers_upper
BEFORE INSERT ON customers
FOR EACH ROW
BEGIN
SET NEW.cname = UPPER(NEW.cname);
SET NEW.LOC = UPPER(NEW.LOC);
END;
//
delimiter ;
```

```
delimiter //
CREATE TRIGGER update_customers_upper
BEFORE UPDATE ON customers
FOR EACH ROW
BEGIN
SET NEW.cname = UPPER(NEW.cname);
SET NEW.LOC = UPPER(NEW.LOC);
END;
//
delimiter ;
```

```
delimiter //
CREATE TRIGGER insert_stocks_upper
BEFORE INSERT ON stocks
FOR EACH ROW
BEGIN
SET NEW.description = UPPER(NEW.description);
END;
//
delimiter ;
```

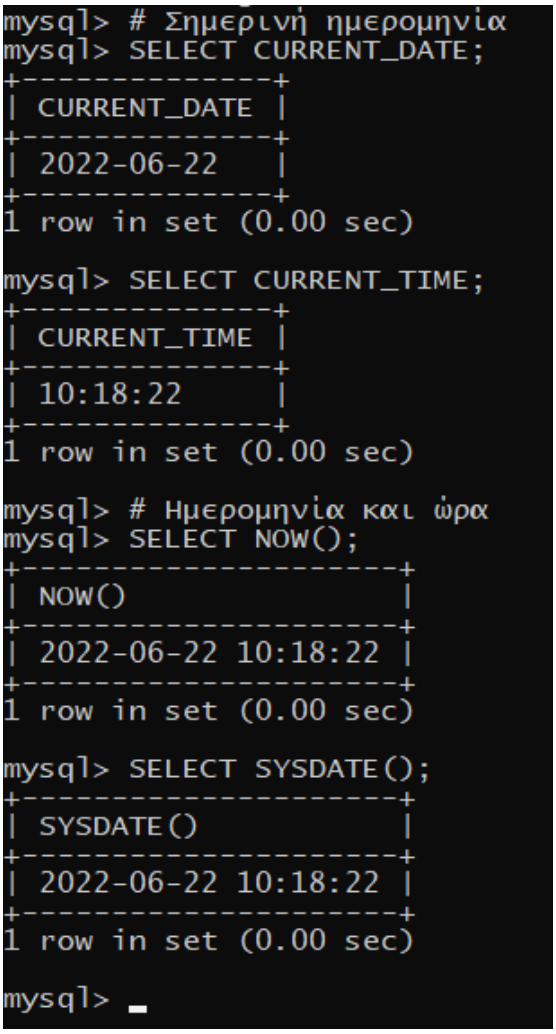
```
delimiter //
CREATE TRIGGER update_stocks_upper
```

```
BEFORE UPDATE ON stocks
FOR EACH ROW
BEGIN
SET NEW.description = UPPER(NEW.description);
END;
//
delimiter ;
```

Ο trigger orders_trig είναι αποθηκευμένος (stored) και εκτελείται πριν από την εισαγωγή των γραμμών του πίνακα orders. Εισάγει αυτόματα την ημερομηνία και την ώρα της νέας παραγγελίας.

Δείτε πρώτα στην εικόνα 8.15 το αποτέλεσμα χρησιμοποίησης των παρακάτω συναρτήσεων στις 21/5/2022.

```
# Σημερινή ημερομηνία
SELECT CURRENT_DATE;
SELECT CURRENT_TIME;
# Ημερομηνία και ώρα
SELECT NOW();
SELECT SYSDATE();
```



Εικόνα 8.15 Χρήση συναρτήσεων για τρέχουσα ημερομηνία και ώρα στο προϊόν MySQL

```
DROP TRIGGER IF EXISTS orders_trig;
DELIMITER //
```

```
CREATE TRIGGER orders_trig
BEFORE INSERT ON orders
FOR EACH ROW
BEGIN
SET NEW.odate = NOW();
END;
//
DELIMITER ;
```

Ο trigger orderlines_trig_ins είναι αποθηκευμένος (stored) και εκτελείται πριν από την **εισαγωγή** των γραμμών του πίνακα orderlines για να υπολογίσει το μερικό σύνολο της παραγγελίας.

```
DROP TRIGGER IF EXISTS orderlines_trig_ins;
DELIMITER //
CREATE TRIGGER orderlines_trig_ins
BEFORE INSERT ON orderlines
FOR EACH ROW
BEGIN
SET NEW.ptotal = (SELECT list_price
 FROM stocks
 WHERE stockno=NEW.stockno) * NEW.qty;
 END;
 //
 DELIMITER ;
```

Ο trigger orderlines_trig_upd είναι αποθηκευμένος (stored) και εκτελείται πριν από την **ενημέρωση** των γραμμών του πίνακα orderlines για να υπολογίσει το μερικό σύνολο της παραγγελίας.

```
DROP TRIGGER IF EXISTS orderlines_trig_upd;
 DELIMITER //
CREATE TRIGGER orderlines_trig_upd
BEFORE UPDATE ON orderlines
FOR EACH ROW
BEGIN
SET NEW.ptotal = (SELECT list_price
 FROM stocks
 WHERE stockno=NEW.stockno) * NEW.qty;
 END;
 //
 DELIMITER ;
```

Ο trigger orderlines_total_ins είναι αποθηκευμένος (stored) και εκτελείται **μετά** από την **εισαγωγή** των γραμμών του πίνακα orderlines για να υπολογίσει το **συνολικό ποσό** της παραγγελίας.

```
DROP TRIGGER IF EXISTS orderlines_total_ins;
DELIMITER //
CREATE TRIGGER orderlines_total_ins
AFTER INSERT ON orderlines
FOR EACH ROW
BEGIN
UPDATE orders SET total = (SELECT SUM(ptotal)
 FROM orderlines
 WHERE orderno=NEW.orderno)
WHERE orderno = NEW.orderno;
```

```
END;
//
DELIMITER ;
```

Ο trigger orderlines_total_upd είναι αποθηκευμένος (stored) και εκτελείται **μετά** από την **ενημέρωση** των γραμμών του πίνακα orderlines για να υπολογίσει το **συνολικό ποσό** της παραγγελίας.

```
DROP TRIGGER IF EXISTS orderlines_total_upd;
DELIMITER //
CREATE TRIGGER orderlines_total_upd
AFTER UPDATE ON orderlines
FOR EACH ROW
BEGIN
UPDATE orders SET total = (SELECT SUM(ptotal)
 FROM orderlines
 WHERE orderno=NEW.orderno)
WHERE orderno = NEW.orderno;
END;
//
DELIMITER ;
```

Ακολουθούν δοκιμές.

Για να δοκιμάσετε τους triggers χρησιμοποιήστε τις δηλώσεις

```
INSERT INTO customers(custno, cname, loc) VALUES(1, 'smith', 'ATHENS');
INSERT INTO customers(custno, cname, loc) VALUES(2, 'JONES', 'volos');
INSERT INTO customers(custno, cname, loc) VALUES(3, 'bates', 'NEW YORK');

SELECT * FROM customers;
```

CUSTNO	CNAME	LOC
1	SMITH	ATHENS
2	JONES	VOLOS
3	BATES	NEW YORK

```
UPDATE customers
SET loc='paris'
WHERE custno=3;

SELECT * FROM customers;
```

CUSTNO	CNAME	LOC
1	SMITH	ATHENS
2	JONES	VOLOS
3	BATES	PARIS

```
INSERT INTO stocks(stockno, description, list_price) VALUES(1, 'APPLE', 1);
INSERT INTO stocks(stockno, description, list_price) VALUES(2, 'ORANGE', 1.5);
INSERT INTO stocks(stockno, description, list_price) VALUES(3, 'lemon', 1.7);
INSERT INTO stocks(stockno, description, list_price) VALUES(4, 'grapes', 2.5);

SELECT * FROM stocks;
```

Stockno	Description	List_price
1	APPLE	1.00
2	ORANGE	1.50
3	LEMON	1.70
4	GRAPES	2.50

```
UPDATE stocks
SET description='strawberries'
WHERE stockno=4;

SELECT * FROM stocks;
```

Stockno	Description	List_price
1	APPLE	1.00
2	ORANGE	1.50
3	LEMON	1.70
4	STRAWBERRIES	2.50

```
INSERT INTO orders(custno, odate) VALUES (1, current_date);

INSERT INTO orderlines(orderno, stockno, qty) VALUES (1, 1, 10);
INSERT INTO orderlines(orderno, stockno, qty) VALUES (1, 2, 5);

INSERT INTO orders(custno, odate) VALUES (1, current_date);
INSERT INTO orderlines(orderno, stockno, qty) VALUES (1, 3, 10);
INSERT INTO orderlines(orderno, stockno, qty) VALUES (2, 3, 8);
INSERT INTO orderlines(orderno, stockno, qty) VALUES (2, 1, 15);

SELECT * FROM orders;
```

Orderno	Custno	Odate	Total
1	1	2022-05-21 08:50:37	34.50
2	1	2022-05-21 08:50:47	28.60

```
SELECT * FROM orderlines;
```

Orderno	Stockno	Qty	Ptotal
1	1	10	10.00
1	2	5	7.50
1	3	10	17.00
2	1	15	15.00
2	3	8	13.60

```
UPDATE ORDERLINES SET QTY=7
WHERE STOCKNO=1 AND ORDERNO=1;

SELECT * FROM orders;
```

Orderno	Custno	Odate	Total
1	1	2022-05-22 08:50:37	31.50
2	1	2022-05-22 08:50:47	28.60

```
SELECT * FROM orderlines;
```

Orderno	Stockno	Qty	Ptotal
1	1	10	7.00
1	2	5	7.50
1	3	10	17.00
2	1	15	15.00
2	3	8	13.60

8.6.1 Μία ενδιαφέρουσα διερεύνηση

Χρησιμοποιήστε δέσμη εντολών (script) ορίζοντας τη βάση λίγο διαφορετικά και εκτελέστε.

```
DROP DATABASE IF EXISTS myorders;

DROP DATABASE IF EXISTS myorders;
CREATE DATABASE myorders;
USE myorders;

DROP TABLE IF EXISTS customers;
CREATE TABLE customers(custno INT, cname VARCHAR(255) NOT NULL,
   loc VARCHAR(255), PRIMARY KEY (custno) );

DROP TABLE IF EXISTS stocks;
CREATE TABLE stocks(stockno INT, description VARCHAR(255) NOT NULL,
 list_price DECIMAL(5,2) NOT NULL, PRIMARY KEY (stockno));

DROP TABLE IF EXISTS orders;
CREATE TABLE orders(orderno INT AUTO_INCREMENT, custno INT NOT NULL,
 odate DATETIME NOT NULL, total DECIMAL(5,2),
 UNIQUE(custno, odate),
 PRIMARY KEY (orderno) );

DROP TABLE IF EXISTS orderlines;
CREATE TABLE orderlines(orderno INT, stockno INT, qty INT NOT NULL,
 ptotal DECIMAL(5,2), PRIMARY KEY (orderno, stockno));

SHOW TABLES;

DESCRIBE customers;
DESCRIBE stocks;
DESCRIBE orders;
DESCRIBE orderlines;
```

```
delimiter //
CREATE TRIGGER insert_customers_upper
BEFORE INSERT ON customers
FOR EACH ROW
BEGIN
SET NEW.cname = UPPER(NEW.cname);
SET NEW.LOC = UPPER(NEW.LOC);
END;
//
delimiter ;
```

```
delimiter //
CREATE TRIGGER update_customers_upper
BEFORE UPDATE ON customers
FOR EACH ROW
BEGIN
SET NEW.cname = UPPER(NEW.cname);
SET NEW.LOC = UPPER(NEW.LOC);
END;
//
delimiter ;
```

```
delimiter //
CREATE TRIGGER insert_stocks_upper
BEFORE INSERT ON stocks
FOR EACH ROW
BEGIN
SET NEW.description = UPPER(NEW.description);
END;
//
delimiter ;
```

```
delimiter //
CREATE TRIGGER update_stocks_upper
BEFORE UPDATE ON stocks
FOR EACH ROW
BEGIN
SET NEW.description = UPPER(NEW.description);
END;
//

delimiter ;
```

```
DROP TRIGGER IF EXISTS orders_trig;
DELIMITER //
CREATE TRIGGER orders_trig
BEFORE INSERT ON orders
FOR EACH ROW
```

```
BEGIN
SET NEW.odate = NOW();
END;
//
DELIMITER ;
```

```
DROP TRIGGER IF EXISTS orderlines_trig_ins;
DELIMITER //
CREATE TRIGGER orderlines_trig_ins
BEFORE INSERT ON orderlines
FOR EACH ROW
BEGIN
SET NEW.ptotal = (SELECT list_price
 FROM stocks
 WHERE stockno=NEW.stockno) * NEW.qty;
 END;
 //
 DELIMITER ;
```

```
DROP TRIGGER IF EXISTS orderlines_trig_upd;
 DELIMITER //
CREATE TRIGGER orderlines_trig_upd
BEFORE UPDATE ON orderlines
FOR EACH ROW
BEGIN
SET NEW.ptotal = (SELECT list_price
 FROM stocks
 WHERE stockno=NEW.stockno) * NEW.qty;
 END;
 //
 DELIMITER ;
```

```
DROP TRIGGER IF EXISTS orderlines_total_ins;
DELIMITER //
CREATE TRIGGER orderlines_total_ins
AFTER INSERT ON orderlines
FOR EACH ROW
BEGIN
UPDATE orders SET total = (SELECT SUM(ptotal)
 FROM orderlines
 WHERE orderno=NEW.orderno)
WHERE orderno = NEW.orderno;
END;
//
DELIMITER ;
```

```
DROP TRIGGER IF EXISTS orderlines_total_upd;
DELIMITER //
```

```
CREATE TRIGGER orderlines_total_upd
AFTER UPDATE ON orderlines
FOR EACH ROW
BEGIN
UPDATE orders SET total = (SELECT SUM(ptotal)
 FROM orderlines
 WHERE orderno=NEW.orderno)
WHERE orderno = NEW.orderno;
END;
//
DELIMITER ;
```

```
INSERT INTO customers(custno, cname, loc) VALUES(1, 'smith', 'ATHENS');
INSERT INTO customers(custno, cname, loc) VALUES(2, 'JONES', 'volos');
INSERT INTO customers(custno, cname, loc) VALUES(3, 'bates', 'NEW YORK');

SELECT * FROM customers;

UPDATE customers
SET loc='paris'
WHERE custno=3;

SELECT * FROM customers;

INSERT INTO stocks(stockno, description, list_price) VALUES(1, 'APPLE', 1);
INSERT INTO stocks(stockno, description, list_price) VALUES(2, 'ORANGE', 1.5);
INSERT INTO stocks(stockno, description, list_price) VALUES(3, 'lemon', 1.7);
INSERT INTO stocks(stockno, description, list_price) VALUES(4, 'grapes', 2.5);

SELECT * FROM stocks;

UPDATE stocks
SET description='strawberries'
WHERE stockno=4;

SELECT * FROM stocks;

INSERT INTO orders(custno, odate) VALUES (1, current_date);

INSERT INTO orderlines(orderno, stockno, qty) VALUES (1, 1, 10);
INSERT INTO orderlines(orderno, stockno, qty) VALUES (1, 2, 5);

INSERT INTO orders(custno, odate) VALUES (1, current_date);
INSERT INTO orderlines(orderno, stockno, qty) VALUES (1, 3, 10);
INSERT INTO orderlines(orderno, stockno, qty) VALUES (2, 3, 8);
INSERT INTO orderlines(orderno, stockno, qty) VALUES (2, 1, 15);

SELECT * FROM orders;
SELECT * FROM orderlines;

UPDATE ORDERLINES SET QTY=7
WHERE STOCKNO=1 AND ORDERNO=1;

SELECT * FROM orders;
SELECT * FROM orderlines;
```

Προσοχή!

Η υποπρόταση UNIQUE(custno, odate) μπορεί να προκαλέσει κάτω από κάποιες χρονικές συμπτώσεις προβλήματα στην εισαγωγή παραγγελιών. Σημαίνει ότι ο ίδιος πελάτης δεν μπορεί να δώσει παραγγελία το ίδιο DATETIME! Δείτε το αποτέλεσμα της εκτέλεσης.

```
SELECT * FROM orders;
```

Orderno	Custno	Odate	Total
1	1	2022-07-22 09:43:16	31.50

```
SELECT * FROM orderlines;
```

Orderno	Stockno	Qty	Ptotal
1	1	10	7.00
1	2	5	7.50
1	3	10	17.00
2	1	15	15.00
2	3	8	13.60

```
# Προσθέστε και άλλες παραγγελίες για να δείτε τι θα συμβεί.

INSERT INTO orders(custno, odate) VALUES (1, current_date);
INSERT INTO orderlines(orderno, stockno, qty) VALUES (2, 2, 10);
INSERT INTO orderlines(orderno, stockno, qty) VALUES (3, 3, 8);
INSERT INTO orderlines(orderno, stockno, qty) VALUES (3, 1, 15);
```

8.7 Πως βλέπουμε πληροφορίες για τη βάση δεδομένων στο προϊόν της mySQL

Θα ενδιαφερθούμε σε μία ενδιαφέρουσα βάση δεδομένων, τη βάση information_schema. Η βάση αυτή χρησιμοποιείται από το προϊόν MySQL και περιλαμβάνει όλες τις πληροφορίες για τη βάση δεδομένων που κατασκευάσαμε και όλες τις άλλες βάσεις. Στην εικόνα 8.16 παραθέτουμε τις βάσεις δεδομένων που χρησιμοποιεί το προϊόν MySQL (και βάσεις που ορίσαμε).

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| myorders           |
| mysql              |
| test               |
| testdb             |
+--------------------+
5 rows in set (0.00 sec)
```

Εικόνα 8.16 Βάσεις δεδομένων που χρησιμοποιεί το προϊόν MySQL. Information_schema, mysql

Στην εικόνα 8.17 χρησιμοποιούμε τη βάση Information_schema και βλέπουμε κάποιους από τους πίνακές της.

Οι πίνακες αυτοί αυξάνονται σε κάθε νέα version του προϊόντος.

```
mysql> USE INFORMATION_SCHEMA;
Database changed
mysql> SHOW TABLES;
+---------------------------------------+
| Tables_in_information_schema          |
+---------------------------------------+
| ADMINISTRABLE_ROLE_AUTHORIZATIONS     |
| APPLICABLE_ROLES                      |
| CHARACTER_SETS                        |
| CHECK_CONSTRAINTS                     |
| COLLATION_CHARACTER_SET_APPLICABILITY |
| COLLATIONS                            |
| COLUMN_PRIVILEGES                     |
| COLUMN_STATISTICS                     |
| COLUMNS                               |
| COLUMNS_EXTENSIONS                    |
| ENABLED_ROLES                         |
| ENGINES                               |
| EVENTS                                |
| FILES                                 |
| INNODB_BUFFER_PAGE                    |
| INNODB_BUFFER_PAGE_LRU                |
| INNODB_BUFFER_POOL_STATS              |
| INNODB_CACHED_INDEXES                 |
| INNODB_CMP                            |
| INNODB_CMP_PER_INDEX                  |
| INNODB_CMP_PER_INDEX_RESET            |
| INNODB_CMP_RESET                      |
| INNODB_CMPMEM                         |
| INNODB_CMPMEM_RESET                   |
| INNODB_COLUMNS                        |
| INNODB_DATAFILES                      |
| INNODB_FIELDS                         |
| INNODB_FOREIGN                        |
| INNODB_FOREIGN_COLS                   |
| INNODB_FT_BEING_DELETED               |
| INNODB_FT_CONFIG                      |
| INNODB_FT_DEFAULT_STOPWORD            |
| INNODB_FT_DELETED                     |
| INNODB_FT_INDEX_CACHE                 |
| INNODB_FT_INDEX_TABLE                 |
| INNODB_INDEXES                        |
| INNODB_METRICS                        |
| INNODB_TABLESPACES                    |
| INNODB_TABLESPACES_BRIEF              |
| INNODB_TABLESTATS                     |
| INNODB_TEMP_TABLE_INFO                |
| INNODB_TRX                            |
| INNODB_VIRTUAL                        |
| KEY_COLUMN_USAGE                      |
| KEYWORDS                              |
| OPTIMIZER_TRACE                       |
| PARAMETERS                            |
| PARTITIONS                            |
| PLUGINS                               |
| PROCESSLIST                           |
| PROFILING                             |
| REFERENTIAL_CONSTRAINTS               |
| RESOURCE_GROUPS                       |
| ROLE_COLUMN_GRANTS                    |
| ROLE_ROUTINE_GRANTS                   |
| ROLE_TABLE_GRANTS                     |
| ROUTINES                              |
| SCHEMA_PRIVILEGES                     |
| SCHEMATA                              |
| SCHEMATA_EXTENSIONS                   |
| ST_GEOMETRY_COLUMNS                   |
| ST_SPATIAL_REFERENCE_SYSTEMS          |
| ST_UNITS_OF_MEASURE                   |
| STATISTICS                            |
| TABLE_CONSTRAINTS                     |
| TABLE_CONSTRAINTS_EXTENSIONS          |
| TABLE_PRIVILEGES                      |
| TABLES                                |
| TABLES_EXTENSIONS                     |
| TABLESPACES                           |
| TABLESPACES_EXTENSIONS                |
| TRIGGERS                              |
| USER_ATTRIBUTES                       |
| USER_PRIVILEGES                       |
| VIEW_ROUTINE_USAGE                    |
| VIEW_TABLE_USAGE                      |
| VIEWS                                 |
+---------------------------------------+
79 rows in set (0.10 sec)
```

Εικόνα 8.17 Πίνακες της βάσης δεδομένων Information_schema

Στην εικόνα 8.18 βλέπουμε τις στήλες του πίνακα των εναυσμάτων (trigger).

```
mysql>
mysql> DESCRIBE TRIGGERS;
+---------------------------+---------------+------+-----+---------+-------+
| Field                     | Type          | Null | Key | Default | Extra |
+---------------------------+---------------+------+-----+---------+-------+
| TRIGGER_CATALOG           | varchar(512)  | YES  |     | NULL    |       |
| TRIGGER_SCHEMA            | varchar(64)   | NO   |     |         |       |
| TRIGGER_NAME              | varchar(64)   | NO   |     |         |       |
| EVENT_MANIPULATION        | varchar(6)    | NO   |     |         |       |
| EVENT_OBJECT_CATALOG      | varchar(512)  | YES  |     | NULL    |       |
| EVENT_OBJECT_SCHEMA       | varchar(64)   | NO   |     |         |       |
| EVENT_OBJECT_TABLE        | varchar(64)   | NO   |     |         |       |
| ACTION_ORDER              | bigint(4)     | NO   |     | 0       |       |
| ACTION_CONDITION          | longtext      | YES  |     | NULL    |       |
| ACTION_STATEMENT          | longtext      | NO   |     | NULL    |       |
| ACTION_ORIENTATION        | varchar(9)    | NO   |     |         |       |
| ACTION_TIMING             | varchar(6)    | NO   |     |         |       |
| ACTION_REFERENCE_OLD_TABLE| varchar(64)   | YES  |     | NULL    |       |
| ACTION_REFERENCE_NEW_TABLE| varchar(64)   | YES  |     | NULL    |       |
| ACTION_REFERENCE_OLD_ROW  | varchar(3)    | NO   |     |         |       |
| ACTION_REFERENCE_NEW_ROW  | varchar(3)    | NO   |     |         |       |
| CREATED                   | datetime      | YES  |     | NULL    |       |
| SQL_MODE                  | varchar(8192) | NO   |     |         |       |
| DEFINER                   | varchar(77)   | NO   |     |         |       |
| CHARACTER_SET_CLIENT      | varchar(32)   | NO   |     |         |       |
| COLLATION_CONNECTION      | varchar(32)   | NO   |     |         |       |
| DATABASE_COLLATION        | varchar(32)   | NO   |     |         |       |
+---------------------------+---------------+------+-----+---------+-------+
22 rows in set (0.02 sec)
```

Εικόνα 8.18 Στήλες πίνακα triggers

Στην εικόνα 8.19 βλέπουμε κάποια στοιχεία για τα εναύσματα που ορίσαμε:

```
select trigger_name, event_manipulation, event_object_table from triggers;
```

```
mysql>
mysql> select trigger_name, event_manipulation,event_object_table from triggers;
+----------------------+--------------------+--------------------+
| trigger_name         | event_manipulation | event_object_table |
+----------------------+--------------------+--------------------+
| orderlines_trig_ins  | INSERT             | orderlines         |
| orderlines_total_ins | INSERT             | orderlines         |
| orderlines_trig_upd  | UPDATE             | orderlines         |
| orderlines_total_upd | UPDATE             | orderlines         |
| orders_trig          | INSERT             | orders             |
+----------------------+--------------------+--------------------+
5 rows in set (0.06 sec)
```

Εικόνα 8.19 Στοιχεία για τα εναύσματα που ορίσαμε

Στην εικόνα 8.20 βλέπουμε κάποια στοιχεία για συγκεκριμένο έναυσμα που ορίσαμε:

```
mysql> select trigger_name, event_manipulation,event_object_table from triggers
    -> where trigger_name='orders_trig';
+--------------+--------------------+--------------------+
| trigger_name | event_manipulation | event_object_table |
+--------------+--------------------+--------------------+
| orders_trig  | INSERT             | orders             |
+--------------+--------------------+--------------------+
1 row in set (0.03 sec)
```

Εικόνα 8.20 Στοιχεία για συγκεκριμένο έναυσμα

Στην εικόνα 8.21 βλέπουμε στοιχεία για τα εναύσματα που δημιουργήσαμε για συγκεκριμένη βάση δεδομένων.

```
Use myorders;
Select * from information_schema.triggers\G
```

```
mysql>
mysql> use myorders;
Database changed
mysql>
mysql>
mysql> SELECT * FROM INFORMATION_SCHEMA.TRIGGERS\G
*************************** 1. row ***************************
           TRIGGER_CATALOG: NULL
            TRIGGER_SCHEMA: myorders
              TRIGGER_NAME: orderlines_trig_ins
        EVENT_MANIPULATION: INSERT
      EVENT_OBJECT_CATALOG: NULL
       EVENT_OBJECT_SCHEMA: myorders
        EVENT_OBJECT_TABLE: orderlines
              ACTION_ORDER: 0
          ACTION_CONDITION: NULL
          ACTION_STATEMENT: SET NEW.ptotal = (SELECT list_price
                                              FROM stocks
                                              WHERE stockno=NEW.stockno) * NEW.qty
        ACTION_ORIENTATION: ROW
             ACTION_TIMING: BEFORE
ACTION_REFERENCE_OLD_TABLE: NULL
ACTION_REFERENCE_NEW_TABLE: NULL
  ACTION_REFERENCE_OLD_ROW: OLD
  ACTION_REFERENCE_NEW_ROW: NEW
                   CREATED: NULL
                  SQL_MODE: STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SU
BSTITUTION
                   DEFINER: root@localhost
      CHARACTER_SET_CLIENT: latin1
```

```
mysql> SHOW TRIGGERS LIKE 'orders%'\G
*************************** 1. row ***************************
             Trigger: orders_trig
               Event: INSERT
               Table: orders
           Statement: SET NEW.odate = NOW()
              Timing: BEFORE
             Created: NULL
            sql_mode: STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITU
TION
             Definer: root@localhost
character_set_client: latin1
collation_connection: latin1_swedish_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Κεφάλαιο 9
Προγραμματισμός εφαρμογών βάσεων δεδομένων. Δημιουργία και χρήση stored procedures: procedures, functions, triggers, cursors σύμφωνα με το πρότυπο SQL PSM

Σύνοψη
Στο κεφάλαιο αυτό 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 συμμορφώνεται με το πρότυπο Persistent Stored Modules (SQL PSM standard). Στη συνέχεια, συζητάμε τα θέματα του προγραμματισμού και της χρήσης διαδικασιών (procedures), συναρτήσεων (functions) και cursors με παραδείγματα. Στα παραδείγματα γίνεται χρήση του προϊόντος MySQL/MariaDB's του οποίου η διαδικαστική γλώσσα (procedural language) συμμορφώνεται σε μεγάλο βαθμό με το πρότυπο SQL/PSM. Επισημαίνεται ότι η τεχνολογία PL/SQL που υποστηρίζει και εξελίσσει η εταιρεία Oracle δεν ακολουθεί το πρότυπο SQL PSM.

Το κεφάλαιο περιλαμβάνει τις παρακάτω ενότητες που είναι οργανωμένες σαν «περιηγήσεις» (tours) και είναι καλό να μελετηθούν και να δοκιμαστούν σε υπολογιστή:

- *Περιήγηση στον προγραμματισμό και τη χρήση Triggers για την υλοποίηση περιορισμών-επιχειρησιακών κανόνων (constraints-business rules) στο προϊόν διαχείρισης βάσεων δεδομένων MySQL*
- *Περιηγήσεις στον προγραμματισμό και τη χρήση Triggers, Functions, Procedures, Cursors σε διάφορα προϊόντα διαχείρισης βάσεων δεδομένων*
- *Περιήγηση στη διαχείριση προβλημάτων από πλεονάζοντα δεδομένα. Περιήγηση στη διαχείριση ακεραιότητας και συνέπειας δεδομένων με χρήση triggers. Καταγραφή ενεργειών χρήστη με trigger*

Ο ενδιαφερόμενος παραπέμπεται και σε παραδείγματα στο κεφάλαιο «περί συναλλαγών (transactions)» και ειδικότερα στην ενότητα «Ασφάλεια συναλλαγών (transactions' security). SQL διαγνωστικά σφαλμάτων (SQL error diagnostics). Δήλωση Get Diagnostics. Το πρόβλημα των ασφαλών συναλλαγών. Παραδείγματα αντιμετώπισης».

Προαπαιτούμενη γνώση
Προτείνεται η μελέτη των κεφαλαίων 3, 7 και 8 του παρόντος συγγράμματος.

9.1 Περιήγηση στον προγραμματισμό και τη χρήση Triggers για την υλοποίηση περιορισμών-επιχειρησιακών κανόνων (constraints-business rules) στο προϊόν διαχείρισης βάσεων δεδομένων MySQL

Σκοπός της ενότητας είναι η κατανόηση του προγραμματισμού εναυσμάτων (triggers) και η εμβάθυνση στη χρήση εναυσμάτων (triggers) που αναλαμβάνουν την υλοποίηση περιορισμών-επιχειρησιακών κανόνων (constraints-business rules). Ειδικότερα, ενδιαφέρει η κατανόηση της διαχείρισης σφαλμάτων σύμφωνα με τα SQL πρότυπα (χρήση SQLSTATE) αλλά και τεχνικές εξαρτώμενες από το προϊόν (proprietary solutions), π.χ., χρήση MySQL SQLCODE.

Στόχος είναι η εκμάθηση του προγραμματισμού και της χρησιμοποίησης trigger ώστε ο ενδιαφερόμενος να προσεγγίσει το επίπεδο ενός ικανού developer στο προϊόν διαχείρισης βάσεων δεδομένων MySQL. Επιπλέον, στόχος είναι μία εισαγωγή στις συναλλαγές (transactions) με ανάλυση των δηλώσεων COMMIT, ROLLBACK, SET AUTOCOMMIT=0. Τέλος, στόχος είναι η εκμάθηση του προγραμματισμού με χρήση δηλώσεων Signal και Resignal και η εισαγωγή στον προγραμματισμό διαχείρισης σφαλμάτων-εξαιρέσεων που χρησιμοποιεί χειριστές (handlers), δήλωση GET DIAGNOSTICS

9.1.1 Εισαγωγή στις συναλλαγές (transactions). Δηλώσεις Commit, Rollback

Κάθε συναλλαγή (transaction) ολοκληρώνεται με δήλωση Commit (γράφονται όλες οι μεταβολές στη βάση δεδομένων) ή δήλωση Rollback (ακυρώνονται όλες οι μεταβολές).

Παράδειγμα συναλλαγής

Δημιουργία του πίνακα τραπεζικών λογαριασμών accounts στη βάση δεδομένων my_accounts.

```
DROP DATABASE IF EXISTS my_accounts;
CREATE DATABASE my_accounts;
USE my_accounts;
CREATE TABLE Accounts (
 acctID INTEGER NOT NULL PRIMARY KEY,
 balance INTEGER NOT NULL);
```

9.1.2 Έναρξη συναλλαγών στο προϊόν MySQL.

```
SET AUTOCOMMIT=0; -- start transactions
```

Ακολουθεί συναλλαγή εισαγωγής δύο νέων τραπεζικών λογαριασμών. Όλες οι μεταβολές επικυρώνονται με δήλωση Commit.

```
INSERT INTO Accounts (acctID,balance) VALUES (101,1000);
INSERT INTO Accounts (acctID,balance) VALUES (202,2000);
COMMIT; -- end of the first transaction
   -- all the changes are committed
SELECT * FROM accounts;
```

acctID	balance
101	1000
202	2000

Ακολουθεί συναλλαγή μεταφοράς ποσού μεταξύ λογαριασμών. Όλες οι μεταβολές αναιρούνται με δήλωση ROLLBACK. Εισαγωγή δύο νέων λογαριασμών.

```
UPDATE ACCOUNTS
SET BALANCE=BALANCE-100
WHERE ACCTID=101;
```

```
UPDATE ACCOUNTS
SET BALANCE=BALANCE+100
WHERE ACCTID=202;
```

```
SELECT * FROM accounts;
```

acctID	balance
101	900
202	2100

```
ROLLBACK; -- end of the second transaction
-- all the changes are rolled back
SELECT * FROM accounts;
```

acctID	balance
101	1000
202	2000

9.1.3 Πληροφορίες για τα σφάλματα στα προγράμματά μας που μας επιστρέφει ο server

Ο εξυπηρετητής (server) του προϊόντος της MySQL επιστρέφει στον client πληροφορίες για τα σφάλματα (error information) που υπάρχουν στα προγράμματά μας. Για παράδειγμα, δείτε την παρακάτω δήλωση:

```
MySQL> SELECT * FROM bonus;
```

Αν ο πίνακας bonus δεν υπάρχει τότε ο server μας επιστρέφει (επιστρέφει στον client) τις εξής πληροφορίες για το σφάλμα:

```
ERROR 1146 (42S02): Table 'check_validation.bonus' doesn't exist
```

Το μήνυμα περιλαμβάνει τιμή σφάλματος 1146 (σε κώδικα sqlcode), τιμή σφάλματος 42S02 (σε κώδικα SQLSTATE) και μήνυμα που περιγράφει το σφάλμα. Ακολουθεί σύντομη συζήτηση:

- Στο προϊόν της MySQL υπάρχει ένας κώδικας γνωστός ως SQLCODE που αποτελείται από τετραψήφιες τιμές. Ο κώδικας SQLCODE καθορίζει για κάθε σφάλμα μία τιμή σφάλματος που βέβαια ισχύει μόνο στο προϊόν της MySQL. Στην περίπτωσή μας η τιμή σφάλματος είναι ίση με 1146.
- Κάθε ΣΔΒΔ έχει έναν δικό του κώδικα SQLCODE με τιμές για τα σφάλματα.
- Υπάρχει και ο κώδικας SQLSTATE. Οι τιμές του κώδικα SQLSTATE ορίζονται στο πρότυπο ANSI SQL και οι τιμές σφαλμάτων είναι 5-χαρακτήρες. Στην περίπτωσή μας η τιμή σφάλματος είναι ίση με '42S02'.
- Δεν υπάρχει αντιστοιχία όλων των σφαλμάτων (αριθμών) του κώδικα sqlcode με τιμές κώδικα SQLSTATE. Στις περιπτώσεις αυτές χρησιμοποιείται η τιμή (SQLSTATE value) 'HY000' που συμβολίζει ένα «γενικό» σφάλμα (general error).
- Οι developers χρησιμοποιούν όλο και περισσότερο τη λίστα σφαλμάτων του κώδικα SQLSTATE.

Η παρακάτω δήλωση δείχνει τους δύο κώδικες για το σφάλμα.

```
SELECT @sqlstate, @sqlcode;
```

@sqlstate	@sqlcode
42S02	1146

Ο πίνακας 9.1 περιλαμβάνει παραδείγματα τιμών των δύο κωδίκων MySQL SQLCODE, SQLSTATE.

Για πλήρη λίστα των σφαλμάτων βλέπε και το έγγραφο Server Error Codes and Messages.

https://dev.mysql.com/doc/refman/8.0/en/error-messages-server.html

Πίνακας 9.1 Παραδείγματα τιμών των κωδίκων MySQL SQLCODE, SQLSTATE

sqlstate	sqlcode	ERROR	Message
3D000	1046	ER_NO_DB_ERROR	No database selected
42000	1055	ER_WRONG_FIELD_WITH_GROUP	'%s' isn't in GROUP BY
42000	1057	ER_WRONG_SUM_SELECT	Statement has sum functions and columns in same statement
21S01	1058	ER_WRONG_VALUE_COUNT	Column count doesn't match value count
42000	1059	ER_TOO_LONG_IDENT	Identifier name '%s' is too long
42S21	1060	ER_DUP_FIELDNAME	Duplicate column name '%s'
42S02	1146	ER_NO_SUCH_TABLE	Table '%s.%s' doesn't exist
23000	1216	ER_NO_REFERENCED_ROW	Cannot add or update a child row: a foreign key constraint fails. InnoDB reports this error when you try to add a row but there is no parent row, and a foreign key constraint fails. Add the parent row first.
21000	1242	ER_SUBQUERY_NO_1_ROW	Subquery returns more than 1 row
23000	1452	ER_NO_REFERENCED_ROW_2	Cannot add or update a child row: a foreign key constraint fails (%s)
02000	1643	ER_SIGNAL_NOT_FOUND	Unhandled user-defined not found condition
HY000	1644	ER_SIGNAL_EXCEPTION	Unhandled user-defined exception condition
0K000	1645	ER_RESIGNAL_WITHOUT_ ACTIVE_HANDLER	RESIGNAL when handler not active

9.1.4 Σφάλμα που προκύπτει από τιμή που παραβιάζει την ακεραιότητα της βάσης δεδομένων. Χρήση δήλωσης GET DIAGNOSTICS

Ακολουθεί παράδειγμα σφάλματος που προκύπτει από τιμή που παραβιάζει την ακεραιότητα της βάσης δεδομένων. Χρήση δήλωσης GET DIAGNOSTICSΔημιουργία της βάσης δεδομένων και των πινάκων με PRIMARY KEY και FOREIGN KEY.

```
DROP DATABASE IF EXISTS check_validation;

CREATE DATABASE check_validation;
USE check_validation;

DROP TABLE IF EXISTS Customers;

CREATE TABLE Customers(Custno INTEGER NOT NULL PRIMARY KEY,
 Cust_name VARCHAR(30));

DROP TABLE IF EXISTS Accounts;

CREATE TABLE Accounts ( acctID INTEGER NOT NULL PRIMARY KEY,
balance INTEGER NOT NULL, custno INTEGER,
```

```
FOREIGN KEY(Custno) REFERENCES Customers(Custno),
CONSTRAINT unloanable_account CHECK (balance >= 0));

INSERT INTO Customers VALUES(10, 'CODD');
INSERT INTO Customers VALUES(20, 'DATE');
INSERT INTO Accounts (acctID, balance, custno) VALUES (101, 1000, 10);
INSERT INTO Accounts (acctID, balance, custno) VALUES (202, 2000, 20);

SELECT * FROM customers;
```

Custno	Cust_name
10	CODD
20	DATE

```
SELECT * FROM accounts;
```

acctID	balance	Custno
101	1000	10
202	2000	20

Δοκιμή δημιουργίας (εισαγωγής) λογαριασμού ανύπαρκτου πελάτη

```
INSERT INTO Accounts (acctID, balance, custno) VALUES (303, 1000, 30);
-- a foreign key constraint fails
SELECT @sqlstate, @sqlcode;
```

@sqlstate	@sqlcode
23000	1452

Στις νεότερες κυκλοφορίες (version) του προϊόντος MySQL υποστηρίζεται η δήλωση GET DIAGNOSTICS. Ακολουθεί το παράδειγμα χρησιμοποίησής της.

```
INSERT INTO Accounts (acctID, balance, custno) VALUES (303, 1000, 30);
GET DIAGNOSTICS @rowcount=ROW_COUNT;
GET DIAGNOSTICS CONDITION 1 @sqlstate=RETURNED_SQLSTATE, @sqlcode=MYSQL_ERRNO ;
SELECT @sqlstate, @sqlcode, @rowcount;
```

Να οι τιμές των μεταβλητών:

@sqlstate	@sqlcode	@rowcount
23000	1452	-1

Η τιμή -1 της μεταβλητής @rowcount σημαίνει ότι έχει αποτύχει η δήλωση INSERT INTO Accounts (acctID, balance, custno) VALUES (303, 1000, 30);

Περισσότερα στοιχεία μπορείτε να μελετήσετε στο εγχειρίδιο MySQL Internals Manual

(https://dev.mysql.com/doc/internals/en/)

9.1.5 Η σύνταξη δήλωσης CREATE TRIGGER στο προϊόν mySQL

```
CREATE
[DEFINER = { user | CURRENT_USER }]
TRIGGER trigger_name
trigger_time trigger_event
ON tbl_name FOR EACH ROW
[trigger_order]
trigger_body
trigger_time: { BEFORE | AFTER }
trigger_event: { INSERT | UPDATE | DELETE }
trigger_order: { FOLLOWS | PRECEDES }
other_trigger_name;
```

9.1.6 Trigger που διασφαλίζει ότι το υπόλοιπο τραπεζικού λογαριασμού είναι μη αρνητικό και χρησιμοποιεί δήλωση SIGNAL

Ακολουθεί παράδειγμα trigger που διασφαλίζει ότι το υπόλοιπο λογαριασμού είναι μη αρνητικό και χρησιμοποιεί δήλωση SIGNAL

```
# Δημιουργία εκ νέου της βάσης δεδομένων
DROP DATABASE IF EXISTS check_validation;
CREATE DATABASE check_validation;
USE check_validation;

# Δημιουργία των πινάκων

DROP TABLE IF EXISTS Customers;
CREATE TABLE Customers(Custno INTEGER NOT NULL PRIMARY KEY,
 Cust_name VARCHAR(30));

DROP TABLE IF EXISTS Accounts;


CREATE TABLE Accounts ( acctID INTEGER NOT NULL PRIMARY KEY,
balance INTEGER NOT NULL, custno INTEGER,
FOREIGN KEY(Custno) REFERENCES Customers(Custno),
CONSTRAINT unloanable_account CHECK (balance >= 0));

INSERT INTO Customers VALUES(10, 'CODD');
INSERT INTO Customers VALUES(20, 'DATE');
INSERT INTO Accounts (acctID, balance, custno) VALUES (101, 1000, 10);
INSERT INTO Accounts (acctID, balance, custno) VALUES (202, 2000, 20);

SELECT * FROM customers;
```

Custno	Cust_name
10	CODD
20	DATE

```
SELECT * FROM accounts;
```

acctID	balance	Custno
101	1000	10
202	2000	20

 # Απόπειρα εισαγωγής λανθασμένων στοιχείων

```
INSERT INTO Accounts VALUES (1, -1, 10);
INSERT INTO Accounts VALUES (2, 100, 10); -- valid value

SELECT * FROM accounts;
```

Ιδού το πρόβλημα.

acctID	balance	Custno
101	1000	10
202	2000	20
1	-1	10
2	100	10

 # Πρόβλημα με δήλωση UPDATE

```
UPDATE Accounts SET balance = -100
WHERE acctID = 2;
SELECT * FROM accounts;
```

acctID	balance	Custno
101	1000	10
202	2000	20
1	-1	10
2	-100	10

Θα επιλύσουμε το πρόβλημα κατασκευάζοντας trigger στον οποίο και θα χρησιμοποιήσουμε δήλωση SIGNAL (βλέπε και 9.1.6.1)

 # Αναδημιουργία του πίνακα Accounts και δημιουργία των triggers

```
DROP TABLE IF EXISTS Accounts;

CREATE TABLE Accounts ( acctID INTEGER NOT NULL PRIMARY KEY,
 balance INTEGER NOT NULL,
CONSTRAINT unloanable_account CHECK (balance >= 0));

INSERT INTO Accounts (acctID, balance) VALUES (101, 1000);
INSERT INTO Accounts (acctID, balance) VALUES (202, 2000);

COMMIT;

SELECT * FROM accounts;
```

acctID	balance
101	1000
202	2000

9.1.6.1 Σύνταξη δήλωσης SIGNAL

Η δήλωση SIGNAL επιστρέφει ένα σφάλμα (error) ή μία προειδοποίηση (warning) όταν καλείται από stored procedure, stored function, trigger ή event. Πιο συγκεκριμένα επιστρέφει τιμή σφάλματος και μήνυμα του κώδικα SQLSTATE.

```
SIGNAL condition_value
[SET signal_information_item
[, signal_information_item] ...];

condition_value:
SQLSTATE [VALUE] sqlstate_value
  | condition_name

signal_information_item:
condition_information_item_name = simple_value_specification

condition_information_item_name:
CLASS_ORIGIN
  | SUBCLASS_ORIGIN
  | MESSAGE_TEXT
  | MYSQL_ERRNO
  | CONSTRAINT_CATALOG
  | CONSTRAINT_SCHEMA
  | CONSTRAINT_NAME
  | CATALOG_NAME
  | SCHEMA_NAME
  | TABLE_NAME
  | COLUMN_NAME
  | CURSOR_NAME

condition_name, simple_value_specification:
(δείτε τα παραδείγματα)
```

Μια συνήθης μορφή,

```
SIGNAL SQLSTATE | condition_name;
SET condition_information_item_name_1 = value_1,
condition_information_item_name_1 = value_2, etc;
```

Για παράδειγμα,

```
SIGNAL SQLSTATE '23513'
SET MESSAGE_TEXT = 'Negative balance not allowed';
```

9.1.6.2 Χρήση δήλωσης SIGNAL για την εμφάνιση τιμής σφάλματος SQLSTATE και μηνύματος MESSAGE_TEXT

Οι παρακάτω triggers Accounts_upd_trg και Accounts_ins_trg χρησιμοποιούν δήλωση SIGNAL για να εμφανίσουν την τιμή σφάλματος ('23513') του κώδικα SQLSTATE και το μήνυμα σφάλματος MESSAGE_TEXT ('Negative balance not allowed').

```
delimiter !
CREATE TRIGGER Accounts_upd_trg
BEFORE UPDATE ON Accounts
FOR EACH ROW
BEGIN
IF NEW.balance < 0 THEN
SIGNAL SQLSTATE '23513'
SET MESSAGE_TEXT = 'Negative balance not allowed';
END IF;
END; !
delimiter ;
```

```
delimiter !
CREATE TRIGGER Accounts_ins_trg
BEFORE INSERT ON Accounts
FOR EACH ROW
BEGIN
IF NEW.balance < 0 THEN
SIGNAL SQLSTATE '23513'
SET MESSAGE_TEXT = 'Negative balance not allowed';
END IF;
END; !
delimiter ;
```

Οι δύο trigger μας προστατεύουν από σφάλματα αρνητικού υπολοίπου λογαριασμού κατά την απόπειρα εκτέλεσης δηλώσεων UPDATE, INSERT οι οποίες παραβιάζουν την ακεραιότητα της βάσης δεδομένων/

--testing the triggers

Απόπειρα δημιουργίας (εισαγωγής) λογαριασμού με αρνητικό υπόλοιπο.

```
INSERT INTO Accounts VALUES (1, -1);
GET DIAGNOSTICS @rowcount = ROW_COUNT;
GET DIAGNOSTICS CONDITION 1 @sqlstate = RETURNED_SQLSTATE,
```

```
@sqlcode = MYSQL_ERRNO ;
SELECT @sqlstate, @sqlcode, @rowcount;
```

@sqlstate	@sqlcode	@rowcount
23513	1644	-1

Ο πίνακας Accounts δεν άλλαξε.

AcctID	balance
101	1000
202	2000

Δημιουργία λογαριασμού με θετικό υπόλοιπο και απόπειρα μεταβολής του υπολοίπου σε αρνητικό.

```
INSERT INTO Accounts VALUES (2, 100);
UPDATE Accounts SET balance = -100
WHERE acctID = 2;
GET DIAGNOSTICS @rowcount = ROW_COUNT;
GET DIAGNOSTICS CONDITION 1 @sqlstate =RETURNED_SQLSTATE,
 @sqlcode = MYSQL_ERRNO ;
SELECT @sqlstate, @sqlcode, @rowcount;
```

@sqlstate	@sqlcode	@rowcount
23513	1644	-1

```
SELECT * FROM accounts;
```

AcctID	balance
101	1000
202	2000
2	100

9.1.6.3 Πως θα δούμε τους triggers στο περιβάλλον της mySQL

Ακολουθούν γενικής μορφής δηλώσεις SELECT που δείχνουν:

Πως ορίστηκε ένας Trigger

```
SELECT * FROM Information_Schema.Triggers
WHERE Trigger_schema = 'database_name' AND
  Trigger_name = 'trigger_name';
```

Ποιοί Triggers ορίστηκαν στη βάση δεδομένων

```
SELECT * FROM Information_Schema.Triggers
WHERE Trigger_schema = 'database_name';
```

Ποιοί Triggers ορίστηκαν σε πίνακα

```
SELECT * FROM Information_Schema.Triggers
WHERE Trigger_schema = 'database_name' AND
  Event_object_table = 'table_name';
```

Πως διαγράφεται ένας TRIGGER

```
DROP TRIGGER trigger_name;
```

Να πως συγκεκριμενοποιούνται οι γενικής μορφής δηλώσεις στο παράδειγμά μας.

```
SELECT * FROM Information_Schema.Triggers
WHERE Trigger_schema = 'check_validation' AND
  Trigger_name = 'Accounts_ins_trg' \G
SELECT * FROM Information_Schema.Triggers
WHERE Trigger_schema = 'check_validation' \G
```

```
SELECT * FROM Information_Schema.Triggers
WHERE Trigger_schema = 'check_validation' AND
  Event_object_table = 'accounts' \G
```

Ακολουθούν δηλώσεις κατάργησης trigger.

```
DROP TRIGGER Accounts_ins_trg ;
DROP TRIGGER Accounts_upd_trg;
```

Παράδειγμα 1

Πως ο κωδικός τμήματος θα έχει θετικές τιμές με χρήση trigger

```
USE MY_FIRST_TRIGGERS_DB;
DROP TABLE IF EXISTS NEW_DEPT;
CREATE TABLE NEW_DEPT(DEPTNO INT NOT NULL, DNAME CHAR(30),
PRIMARY KEY(DEPTNO));
```

Παραθέτουμε τον ορισμό του trigger.

Αν ήδη υπάρχει και θέλουμε να τον αναδημιουργήσουμε αρχικά τον καταργούμε.

Σημείωση. Στην περίπτωση της Oracle παρέχεται η δήλωση CREATE OR REPLACE TRIGGER. Επομένως, αν ο developer κατά την κατασκευή rigger σε Oracle κάνει κάποιο λογικό λάθος ή αν απλά θέλει να αλλάξει τη λειτουργία του μπορεί άμεσα να χρησιμοποιήσει τη δήλωση.

```
drop trigger if exists trg_signals_raising_error;
delimiter //
create trigger trg_signals_raising_error
before insert on new_dept
for each row
begin
declare msg varchar(255);
if new.deptno < 0 then
set msg = concat('Trying to insert a negative value in deptno: ',
cast(new.deptno as char));
signal sqlstate '45000' set message_text = msg;
end if;
end
//
delimiter ;
```

```
-- run the following statements as separate:
insert into new_dept values (1, 'SALES'), (-1, 'ACCOUNTING'), (2, 'RESEARCH'); -
- everything fails as one row is wrong
select * from new_dept;
insert into new_dept values (1, 'SALES'); -- succeeds
insert into new_dept values (-1, 'ACCOUNTING'); -- fails
select * from new_dept;
```

Παράδειγμα 2

Όταν χρησιμοποιείτε δήλωση INSERT IGNORE πρέπει να επισημανθεί ότι οι insert triggers εκτελούνται ακόμη και αν η νέα γραμμή δεν εισάγεται στον πίνακα λόγω παραβίασης του πρώτου κανόνα ακεραιότητας (duplicate key constraint).

```
Create table temp(c_date DATE, c_user CHAR(30), Msg CHAR(40));
delimiter //
create trigger trg_duplicate_key
before insert on new_dept
for each row
begin
INSERT INTO temp VALUES(current_date, current_user, 'Trigger execution');
end
//
delimiter ;
select * from new_dept;
INSERT IGNORE into new_dept values (1, 'SALES'); -- duplicate key
```

Δεν γράφει τη γραμμή στον πίνακα αλλά το μήνυμα είναι:

```
QUERY OK, 0 rows affected.
```

```
select * from new_dept;
```

Deptno	dname
1	SALES

```
select * from temp;
```

C_date	C_user	Msg
2018-05-29	root @localhost	Trigger execution

9.1.7 Μία αναφορά στους Temporary tables που μπορούμε να χρησιμοποιήσουμε στα παραδείγματά μας

Ακολουθεί δημιουργία πίνακα παραγγελιών και χρήση προσωρινού πίνακα (Temporary table) στον οποίο και μεταφέρουμε τμήμα του περιεχομένου του πίνακα παραγγελιών.

```
drop database if exists orders_db;
create database orders_db;

use orders_db;
create table orders (order_num int not null, custno int not null, odate datetime
```

```
DEFAULT now(), total int(10), primary key(order_num));
SET AUTOCOMMIT=0;

insert into orders values (1,1,current_timestamp,1000);
insert into orders values (2,1,current_timestamp,5000);
insert into orders values (3,2,current_timestamp,6000);

commit;
```

Εκτελέστε δήλωση SELECT. Διαπιστώστε ότι κάθε παραγγελία εισάγεται στον πίνακα το τρέχον στιγμιότυπο (current_timestamp) κατά την εκτέλεση δήλωσης insert into orders.

```
select * from orders;
CREATE TEMPORARY TABLE my_temp(t_num int NOT NULL auto_increment,
  orderno INT, total INT, PRIMARY KEY(t_num));
INSERT INTO my_temp(orderno, total) SELECT order_num, total FROM orders;

SELECT * FROM my_temp;
```

T_num	Orderno	Total
1	1	1000
2	2	5000
3	3	6000

Προσοχή στο παρακάτω!

```
SHOW TABLES;
```

Tables_in_orders_db
Orders

Δεν βλέπουμε τον προσωρινό πίνακα.

9.1.8 Μία συζήτηση για τη διαφορά των τύπων δεδομένων int, float, decimal (difference between float, decimal and integer datatypes) και τις εφαρμογές

Για τη στήλη total της παραγγελίας μπορούμε να χρησιμοποιήσουμε τύπους δεδομένων, π.χ., int, float, decimal(10,2).

Καλό είναι να έχουμε στο μυαλό μας το παρακάτω παράδειγμα:

```
DROP TABLE IF EXISTS num_types;
CREATE TABLE num_types(a1 DECIMAL(10,2), a2 DECIMAL(10,2),
b1 FLOAT, b2 FLOAT, c1 INTEGER, c2 INTEGER);
INSERT INTO num_types VALUES(100.50,100.50, 100.50, 100.50, 100.50, 100.50);
SELECT * FROM num_types;
```

a1	a2	b1	b2	c1	c2
100.50	100.50	100.5	100.5	101	101

```
SELECT a1+a2 'decimal', b1+b2 'float' , c1+c2 'integer'
FROM num_types;
```

decimal	Float	integer
201.00	201	202

```
SELECT (a1+a2)/6.55 'decimal', (b1+b2)/6.55 'float' , (c1+c2)/6.55 'integer'
FROM num_types;
```

decimal	float	integer
30.687023	30.68702290076336	30.8397

Σε γενικές γραμμές float αριθμούς χρησιμοποιούμε σε επιστημονικούς υπολογισμούς και decimal αριθμούς σε εμπορικές εφαρμογές.

9.2 Περιηγήσεις στον προγραμματισμό και τη χρήση Triggers, Functions, Procedures, Cursors σε διάφορα προϊόντα διαχείρισης βάσεων δεδομένων

Σκοπός της ενότητας είναι η κατανόηση σε βάθος της τεχνολογίας διαχείρισης βάσης δεδομένων με χρήση Stored Procedures. Ειδικότερα εξετάζονται θέματα προγραμματισμού Triggers, Functions, Procedures, Cursors. Στόχος είναι η εκμάθηση του προγραμματισμού και της χρησιμοποίησης Stored Procedures (Triggers, Functions, Procedures, Cursors) ώστε ο φοιτητής να προσεγγίσει το επίπεδο ενός ικανού developer.

Στη συνέχεια αναφέρουμε πλεονεκτήματα της χρήσης Stored procedures

Η χρησιμοποίηση Stored Procedures εξασφαλίζει:

- Αυξημένη παραγωγικότητα
- Ευέλικτη διαχείριση της ανάπτυξης εφαρμογών
- Οι Stored Procedures επιτρέπουν σε τμήματα της εφαρμογής (parts of application logic) να αποθηκευθούν στη βάση για λόγους ασφαλείας και επίδοσης (for security and performance Reasons).
- Οι ίδιες Stored Procedures διατίθενται για πολλαπλή χρήση (multi-use), σε διαφορετικές συναλλαγές (transactions) ή από πολλαπλά προγράμματα.

Στην ενότητα αυτή παρουσιάζουμε και συζητάμε πολλά παραδείγματα. Στο e-book,

Martti Laiho, Matti Kurki, Malcolm Crowe, Fritz Laux, Dimitris Dervos and Kari Hirvonen (2018), Introduction to Transaction Programming, Draft 2018-12-03, DBTechNet.org, p. 654

το οποίο διατίθεται με άδεια χρήσης creative commons παρατίθενται πάρα πολλά παραδείγματα σε Oracle, MySQL, PostgreSQL, MS SQLServer, Python, Ruby κ.λπ. Πολλά από τα παραδείγματά τους αποτέλεσαν έναυσμα για τα παραδείγματα που χρησιμοποιούνται στο παρόν σύγγραμμα.

(20) (PDF) Introduction toTransaction Programming (researchgate.net)

Οι ενδιαφερόμενοι αναγνώστες μπορούν ακόμη να δοκιμάσουν και πολλά άλλα παραδείγματα από τα εγχειρίδια των προμηθευτών, bloggers, κ.λπ.

9.2.1 Μεταβλητές οριζόμενες από τον προγραμματιστή (User-Defined Variables)

Μπορείτε να ορίσετε την τιμή μιας μεταβλητής σε μία δήλωση (statement) και να αναφερθείτε σε αυτήν αργότερα σε μία άλλη δήλωση. Με τον τρόπο αυτό «περνάτε» μεταβλητές από μία δήλωση σε άλλη. Τα

ονόματα μεταβλητών δεν εξαρτώνται από κεφαλαία-πεζά (not case-sensitive) και στο προϊόν MySQL είναι μέχρι 64 χαρακτήρες.

Πως δίνουμε τιμές σε μεταβλητές

```
SET @var_name = expr [, @var_name = expr] ...
```

Παράδειγμα

```
SET @alpha=10, @beta=20;
SELECT @alpha, @beta, @chi := @alpha+@beta;
```

9.2.2 Πως απαριθμούμε τις γραμμές ενός πίνακα

Ακολουθεί παράδειγμα με απαρίθμηση των γραμμών του πίνακα τραπεζικών λογαριασμών.

```
DROP DATABASE IF EXISTS my_accounts;

CREATE DATABASE my_accounts;

USE my_accounts;

CREATE TABLE Accounts ( acctID INTEGER NOT NULL PRIMARY KEY,
balance INTEGER NOT NULL);

INSERT INTO Accounts (acctID, balance) VALUES (101, 1000);
INSERT INTO Accounts (acctID, balance) VALUES (202, 2000);
INSERT INTO Accounts (acctID, balance) VALUES (303, 2500);
INSERT INTO Accounts (acctID, balance) VALUES (404, 3000);

SELECT * FROM accounts;
```

acctID	balance
101	1000
202	2000
303	2500
404	3000

```
SET @rownum=0;
SELECT @rownum:=@rownum+1, acctID, balance FROM accounts ORDER BY acctID;
```

@rownum:=@rownum+1	acctID	balance
1	101	1000
2	202	2000
3	303	2500
4	404	3000

9.2.2.1 Πως υλοποιούμε πίνακα που έχει στήλες άλλου πίνακα και επιπλέον στήλη αρίθμησης γραμμών rownum

```
CREATE TABLE New_Accounts (rownum INTEGER,
   acctID INTEGER NOT NULL PRIMARY KEY, balance INTEGER NOT NULL);

SET @rownum=0;

INSERT INTO New_Accounts

SELECT @rownum:=@rownum+1, acctID, balance FROM accounts ORDER BY acctID;

SELECT * FROM New_Accounts;
```

rownum	acctID	Balance
1	101	1000
2	202	2000
3	303	2500
4	404	3000

9.2.3 Πως δημιουργούμε και εκτελούμε («τρέχουμε») παραμετρικές δηλώσεις SQL χρησιμοποιώντας μεταβλητές

Δημιουργούμε πίνακες πελατών και λογαριασμών

```
DROP DATABASE IF EXISTS check_validation;

CREATE DATABASE check_validation;

USE check_validation;

DROP TABLE IF EXISTS Customers;

CREATE TABLE Customers(Custno INTEGER NOT NULL PRIMARY KEY,
 Cust_name VARCHAR(30));

DROP TABLE IF EXISTS Accounts;

CREATE TABLE Accounts ( acctID INTEGER NOT NULL PRIMARY KEY,
balance INTEGER NOT NULL, custno INTEGER,
FOREIGN KEY(Custno) REFERENCES Customers(Custno),
CONSTRAINT unloanable_account CHECK (balance >= 0));

INSERT INTO Customers VALUES(10, 'CODD');
INSERT INTO Customers VALUES(20, 'DATE');
INSERT INTO Customers VALUES(30, 'NAVATHE');
INSERT INTO Accounts (acctID, balance, custno) VALUES (101, 1000, 10);
INSERT INTO Accounts (acctID, balance, custno) VALUES (202, 2000, 20);
INSERT INTO Accounts (acctID, balance, custno) VALUES (303, 2500, 10);
INSERT INTO Accounts (acctID, balance, custno) VALUES (404, 3000, 10);
INSERT INTO Accounts (acctID, balance, custno) VALUES (505, 1000, 30);
INSERT INTO Accounts (acctID, balance, custno) VALUES (506, 2000, 30);

SELECT * FROM customers;
```

custno	cust_name
10	CODD
20	DATE
30	NAVATHE

```
SELECT * FROM accounts;
```

AcctID	Balance	Custno
101	1000	10
202	2000	20
303	2500	10
404	3000	10
505	1000	30
506	2000	30

9.2.3.1 Παραμετρική δήλωση SELECT η οποία περιλαμβάνει υποπρόταση Group by

Παραθέτουμε παράδειγμα.

Δείξτε πόσους λογαριασμούς έχει κάθε πελάτης και το σύνολο των καταθέσεων του, εξαιρουμένου του πελάτη 20.

```
SELECT custno, COUNT(*), SUM(balance)
FROM accounts
WHERE custno NOT IN (20)
GROUP BY custno;
```

Custno	COUNT(*)	SUM(balance)
10	3	6500
30	2	3000

Αντίστοιχη παραμετρική ώστε αντί του πελάτη 20 να εξαίρουμε όποιον πελάτη επιλέξουμε δίδοντας τιμή στη μεταβλητή cust_no.

```
SET @cust_no=20;

SELECT custno, COUNT(*), SUM(balance)
FROM accounts
WHERE custno NOT IN (@cust_no)
GROUP BY custno;
```

Custno	COUNT(*)	SUM(balance)
10	3	6500
30	2	3000

```
SELECT @cust_no;
```

@cust_no
20

9.2.3.2 Παραμετρική δήλωση SELECT η οποία περιλαμβάνει συνθήκη WHERE

Δείξτε πόσους λογαριασμούς έχει και το σύνολο των καταθέσεων του πελάτη 10.

```
SELECT COUNT(*), SUM(balance)
FROM accounts
WHERE custno=10;
```

COUNT(*)	SUM(balance)
3	6500

Αντίστοιχη παραμετρική ώστε να δίνουμε τον κωδικό πελάτη ως τιμή σε μεταβλητή.

```
SET @cust_no=10;
SELECT @cust_no;

SELECT COUNT(*), SUM(balance)
FROM accounts
WHERE custno=@cust_no;
```

@cust_no
10

COUNT(*)	SUM(balance)
3	6500

9.2.3.3 Παραμετρική δήλωση SELECT η οποία περιλαμβάνει αθροιστικές συναρτήσεις

Δείξτε για την τράπεζα πόσους λογαριασμούς έχει, το σύνολο των καταθέσεων και το μέσο ποσό κατάθεσης,

```
SELECT COUNT(*), SUM(balance), AVG(balance)
FROM accounts;
```

COUNT(*)	SUM(balance)	AVG(balance)
6	11500	1916.6667

Αντίστοιχη παραμετρική

```
SET @count_acctID=0, @sum_acctID=0, @avg_acctID=0;
SELECT COUNT(*), SUM(balance), AVG(balance)
INTO @count_acctID, @sum_acctID, @avg_acctID
FROM accounts;
SELECT @count_acctID, @sum_acctID, @avg_acctID, @my_avg :=
@sum_acctID/@count_acctID;
```

@count_acctID	@sum_acctID	@avg_acctID	@sum_acctID/@count_acctID
6	11500	1916.666666666	1916.666666666

Στον πίνακα 9.2 βλέπουμε τη σύνοψη των παραπάνω, δηλαδή δηλώσεις SQL SELECT και αντίστοιχες παραμετρικές.

Πίνακας 9.2 Δηλώσεις SQL SELECT και αντίστοιχες παραμετρικές.

Δήλωση	Αντίστοιχη παραμετρική
SELECT custno, COUNT(*), SUM(balance) FROM accounts WHERE custno NOT IN (20) GROUP BY custno;	SET @cust_no=20; SELECT custno, COUNT(*), SUM(balance) FROM accounts WHERE custno NOT IN (@cust_no) GROUP BY custno;
SELECT COUNT(*), SUM(balance) FROM accounts WHERE custno=10;	Αντίστοιχη παραμετρική SET @cust_no=10; SELECT COUNT(*), SUM(balance) FROM accounts WHERE custno=@cust_no; SELECT @cust_no;
SELECT COUNT(*), SUM(balance), AVG(balance) FROM accounts;	SET @count_acctID=0, @sum_acctID=0, @avg_acctID=0; SELECT COUNT(*), SUM(balance), AVG(balance) INTO @count_acctID, @sum_acctID, @avg_acctID FROM accounts; SELECT @count_acctID, @sum_acctID, @avg_acctID, @my_avg := @sum_acctID/@count_acctID;

Προσοχή! Το παρακάτω πρόγραμμα είναι λανθασμένο!

```
SET @cust=0, @count_acctID=0, @sum_acctID=0;


SELECT custno, COUNT(*), SUM(balance)
INTO @cust, @count_acctID, @sum_acctID
FROM accounts
GROUP BY custno;


ERROR 1172(42000): Result consisted of more than one row
```

Σημείωση

Λόγω της σημασίας που έχει να μπορούμε να γράψουμε και να εκτελέσουμε procedure που βασίζεται σε δήλωση SELECT ακολουθούν παραδείγματα. Αργότερα στην ίδια ενότητα θα παρουσιάσουμε διεξοδικά το θέμα procedure.

9.3 Πως κατασκευάζουμε από παραμετρικές δηλώσεις SQL τις αντίστοιχες procedure.

Ακολουθούν παραδείγματα. Θα αρχίσουμε με τον ορισμό της βάσης δεδομένων psm, των πινάκων της customers, accounts, και την εισαγωγή των δεδομένων με τρόπο ανάλογο αυτού που είδαμε στην ενότητα 9.2.3.

Παραθέτουμε, αρχικά, τους πίνακες πελατών και τραπεζικών λογαριασμών.

```
SELECT * FROM customers;
```

Custno	cust_name	Loc
10	CODD	ATHENS
20	DATE	ATHENS
30	NAVATHE	NEW YORK

```
SELECT * FROM accounts;
```

AcctID	Balance	Custno
101	1000	10
202	2000	20
303	2500	10
404	3000	10
505	2500	30
506	3500	30

Ακολουθεί ο ορισμός βάσης δεδομένων, πινάκων και δηλώσεις εισαγωγής δεδομένων.

```
DROP DATABASE IF EXISTS psm;

CREATE DATABASE psm;
USE psm;

DROP TABLE IF EXISTS Customers;

CREATE TABLE Customers(Custno INTEGER NOT NULL PRIMARY KEY,
 Cust_name VARCHAR(30), loc CHAR(30));

DROP TABLE IF EXISTS Accounts;

CREATE TABLE Accounts ( acctID INTEGER NOT NULL PRIMARY KEY,
balance INTEGER NOT NULL, custno INTEGER,
FOREIGN KEY(Custno) REFERENCES Customers(Custno),
CONSTRAINT unloanable_account CHECK (balance >= 0));

INSERT INTO Customers VALUES(10, 'CODD', 'ATHENS');
INSERT INTO Customers VALUES(20, 'DATE', 'ATHENS');
INSERT INTO Customers VALUES(30, 'NAVATHE', 'NEW YORK');
INSERT INTO Accounts (acctID, balance, custno) VALUES (101, 1000, 10);
INSERT INTO Accounts (acctID, balance, custno) VALUES (202, 2000, 20);
INSERT INTO Accounts (acctID, balance, custno) VALUES (303, 2500, 10);
INSERT INTO Accounts (acctID, balance, custno) VALUES (404, 3000, 10);
```

```
INSERT INTO Accounts (acctID, balance, custno) VALUES (505, 2500, 30);
INSERT INTO Accounts (acctID, balance, custno) VALUES (506, 3500, 30);
```

9.3.1 Κατασκευή procedure η οποία βασίζεται σε δήλωση SELECT με υποπρόταση Group by

Παραθέτουμε παράδειγμα procedure που βασίζεται σε δήλωση SELECT με υποπρόταση Group by.

```
DELIMITER //
CREATE PROCEDURE GetAccountsByCustno()
BEGIN
  SELECT custno, COUNT(*), SUM(balance)
  FROM accounts
  GROUP BY custno; END //
 DELIMITER ;
```

Δοκιμή και αποτελέσματα

```
CALL GetAccountsByCustno();
```

Custno	COUNT(*)	SUM(balance)
10	3	6500
20	1	2000
30	2	6000

9.3.2 Κατασκευή procedure η οποία βασίζεται σε παραμετρική δήλωση SELECT με σύνδεση πινάκων (join) και υποπρόταση group by.

Παραθέτουμε παράδειγμα procedure που βασίζεται σε παραμετρική δήλωση SELECT με σύνδεση πινάκων (join) και υποπρόταση group by.

```
SET @CustomerLocation='ATHENS';
SELECT cust_name, COUNT(*), SUM(balance)
  FROM accounts JOIN customers ON accounts.custno=customers.custno
  WHERE loc= @CustomerLocation
  GROUP BY cust_name;
```

Cust_name	COUNT(*)	SUM(balance)
CODD	3	6500
DATE	1	2000

Ακολουθεί η procedure.

```
DROP PROCEDURE IF EXISTS CountCustomersByLocation;
DELIMITER $$
CREATE PROCEDURE CountCustomersByLocation(
  IN CustomerLocation VARCHAR(45))
BEGIN
  SELECT cust_name, COUNT(*), SUM(balance)
  FROM accounts JOIN customers ON accounts.custno=customers.custno
```

```
  WHERE loc= CustomerLocation
  GROUP BY cust_name;
END;
 $$
 DELIMITER ;
```

Δοκιμή

```
CALL CountCustomersByLocation('ATHENS');
```

Cust_name	COUNT(*)	SUM(balance)
CODD	3	6500
DATE	1	2000

9.3.3 Δηλώσεις SELECT για την επεξεργασία των γραμμών πίνακα

Θα παραθέσουμε περιπτώσεις χρήσιμων δηλώσεων SELECT οι οποίες επεξεργάζονται τα στοιχεία του πίνακα των λογαριασμών. Αφήνεται ως άσκηση να γράψετε τις αντίστοιχες procedures.

9.3.3.1 Πως βλέπουμε μόνο τις 5 πρώτες γραμμές του πίνακα

```
INSERT INTO Accounts (acctID, balance, custno) VALUES (501, 1000, 10);
INSERT INTO Accounts (acctID, balance, custno) VALUES (502, 2000, 20);
INSERT INTO Accounts (acctID, balance, custno) VALUES (503, 2500, 10);
INSERT INTO Accounts (acctID, balance, custno) VALUES (504, 3000, 10);
SELECT * FROM accounts;
```

AcctID	Balance	Custno
101	1000	10
202	2000	20
303	2500	10
404	3000	10
501	1000	10
502	2000	20
503	2500	10
504	3000	10

```
SET @rownum = 0;
SET @startRow = 0;
SET @maxRows = 5;

SELECT * FROM (
SELECT @rownum:=@rownum+1 as rownum, t.*
FROM (SELECT @rownum:=0) r, accounts t) t
WHERE rownum BETWEEN @startRow and @startRow + @maxRows;
```

Rownum	AcctID	Balance	Custno
1	101	1000	10

2	202	2000	20
3	303	2500	10
4	404	3000	10
5	501	1000	10

9.3.3.2 Πως βλέπουμε μόνο τις άρτιες (ζυγές) γραμμές του πίνακα

```
set @counter=0;

select *, @counter rownum from accounts having (@counter:=@counter+1)%2=0 order
by acctid;
```

AcctID	Balance	Custno	Rownum
202	2000	20	2
404	3000	10	4
502	2000	20	6
504	3000	10	8

9.3.3.3 Πως «γεμίζουμε» στήλη με τυχαίες τιμές (random values)

Αρχικά προσθέτουμε στον πίνακα των λογαριασμών τη στήλη temp_rand.

```
ALTER TABLE accounts ADD column temp_rand INT;
set @a:=0;

update accounts set temp_rand=@a:=@a+1 order by rand();
SELECT * FROM accounts;
```

AcctID	Balance	Custno	temp_rand
101	1000	10	6
202	2000	20	8
303	2500	10	3
404	3000	10	5
501	1000	10	1
502	2000	20	4
503	2500	10	2
504	3000	10	7

Να πως καταργούμε τη στήλη temp_rand.

ALTER TABLE accounts DROP column temp_rand;

Άσκηση 1: Να κάνετε όλα τα παραπάνω προγράμματα procedures.

Άσκηση 2: Να γράψετε τις παρακάτω procedures:

1) Καλείται (η procedure) με κωδικό πελάτη και δείχνει το πλήθος των λογαριασμών του και το σύνολο των καταθέσεων του

2) Καλείται με όνομα πελάτη και δείχνει το πλήθος των λογαριασμών του και το σύνολο των καταθέσεων του
3) Καλείται με κωδικό πελάτη και δείχνει τους κωδικούς των υπολοίπων πελατών με το πλήθος των λογαριασμών τους και το σύνολο των καταθέσεών τους
4) Δείχνει τους κωδικούς των πελατών με το πλήθος των λογαριασμών τους και το σύνολο των καταθέσεών τους
5) Δείχνει το πλήθος των λογαριασμών και το σύνολο των καταθέσεών

9.4 Εναύσματα (Triggers) στο προϊόν MySQL

Η υποστήριξη της τεχνολογίας των εναυσμάτων στο προϊόν MySQL έχει βελτιωθεί σημαντικά σε σχέση με παλαιότερες εκδόσεις. Επισημαίνουμε ότι το προϊόν MySQL υποστηρίζει πλέον την πυροδότηση εναυσμάτων από το ίδιο γεγονός, π.χ., BEFORE INSERT ON accounts. Επιπλέον, μας επιτρέπει να καλούμε τα εναύσματα, που συνδέονται με το ίδιο γεγονός, με τη σειρά που επιθυμούμε.

9.4.1 Σύνταξη CREATE TRIGGER σε mySQL

```
CREATE
[DEFINER = { user | CURRENT_USER }]
TRIGGER trigger_name
trigger_time trigger_event
ON tbl_name FOR EACH ROW
[trigger_order]
trigger_body
trigger_time: { BEFORE | AFTER }
trigger_event: { INSERT | UPDATE | DELETE }
trigger_order: { FOLLOWS | PRECEDES } other_trigger_name;
```

9.4.2 Έναυσμα (trigger) που προσθέτει τα ποσά καταθέσεων που εισάγονται με δηλώσεις INSERT σε τραπεζικούς λογαριασμούς.

Θα κατασκευάσουμε το έναυσμα (trigger) calc_sum που προσθέτει τα ποσά καταθέσεων που εισάγονται με δηλώσεις INSERT σε τραπεζικούς λογαριασμούς.

Αρχικά ορίζουμε εκ νέου τον πίνακα τραπεζικών λογαριασμών. Το ποσό (amount) έχει τύπο δεδομένων DECIMAL(12,2).

```
DROP TABLE IF EXISTS accounts;

CREATE TABLE accounts (acct_num INT, amount DECIMAL(12,2));
```

```
CREATE TRIGGER calc_sum
BEFORE INSERT ON accounts
FOR EACH ROW
  SET @sum = @sum + NEW.amount;
```

Δοκιμή

```
SET @sum = 0;

INSERT INTO accounts VALUES(100,1000.50),(101, 2000.50),(102, 1500.00);

SELECT @sum AS 'Total amount';
```

Total amount
4501.00

```
SELECT * FROM accounts;
```

AcctID	balance
100	1000.50
101	2000.50
102	1500.00

Προσοχή στο παρακάτω!

Ορίζουμε ότι το ποσό (amount) αντί να έχει τύπο δεδομένων DECIMAL(12,2) έχει τύπο INT.

```
DROP TABLE IF EXISTS accounts;

CREATE TABLE accounts (acct_num INT, amount int);

CREATE TRIGGER calc_sum
BEFORE INSERT ON accounts
FOR EACH ROW
  SET @sum = @sum + NEW.amount;
```

Δοκιμή

```
SET @sum = 0;

INSERT INTO accounts VALUES(100,1000.50),(101, 2000.50),(102, 1500.00);

SELECT @sum AS 'Total amount';
```

Total amount
4502

```
SELECT * FROM accounts;
```

AcctID	balance
100	1001
101	2001
102	1500

9.4.3 Έναυσμα (trigger) που διαβάζει από πίνακα το σύνολο των καταθέσεων, προσθέτει τα ποσά που εισάγονται (με δηλώσεις INSERT) σε νέους λογαριασμούς και εισάγει το νέο υπόλοιπο στον πίνακα.

Θα κατασκευάσουμε το έναυσμα (trigger) calc_sum που διαβάζει από τον πίνακα total_bal το σύνολο των καταθέσεων, προσθέτει τα ποσά που εισάγονται (με δηλώσεις INSERT) σε νέους λογαριασμούς και εισάγει το νέο υπόλοιπο στον πίνακα.

Παραθέτουμε δύο λύσεις και προτείνουμε την υιοθέτηση της δεύτερης γιατί η πρώτη μπορεί να οδηγήσει σε προβλήματα σε περιβάλλον ταυτόχρονης πρόσβασης πολλών χρηστών στη βάση δεδομένων.

Πρώτη λύση

Δημιουργούμε εκ νέου τη βάση δεδομένων, τον πίνακα των λογαριασμών πελατών και τον πίνακα total_bal στον οποίο καταγράφουμε το σύνολο των καταθέσεων των πελατών μας.

```
DROP DATABASE if exists my_accounts;
CREATE DATABASE my_accounts;
USE my_accounts;
DROP TABLE IF EXISTS accounts;
CREATE TABLE accounts (acct_num INT, amount DECIMAL(12,2));
CREATE TABLE total_bal(total_sum DECIMAL(12,2));
-- η στήλη total_sum έχει το σύνολο καταθέσεων για όλους τους λογαριασμούς
Insert into total_bal VALUES(0);
DROP TRIGGER IF EXISTS calc_sum;
DELIMITER #
CREATE TRIGGER calc_sum
BEFORE INSERT ON accounts
FOR EACH ROW
BEGIN
   DECLARE new_sum DECIMAL(12,2);
   SELECT total_sum
   INTO new_sum
   FROM total_bal;
  SET new_sum=new_sum+ NEW.amount;
  Update total_bal
  SET total_sum= new_sum;
END;
#
```

Παρατηρήστε στον ορισμό του εναύσματος ότι διαβάζουμε την τιμή της στήλης total_sum και την εκχωρούμε στη μεταβλητή new_sum:

```
SELECT total_sum INTO new_sum …
```

Επομένως πολλοί χρήστες διαβάζουν ταυτόχρονα την ίδια τιμή για τη μεταβλητή total_sum. Ο πρώτος που θα εκτελέσει (βλέπε τρίτο βήμα του εναύσματος) δήλωση UPDATE θα προκαλέσει πρόβλημα στους υπόλοιπους που διάβασαν την ίδια τιμή της στήλης total_sum.

```
DELIMITER ;
# TESTING
INSERT INTO accounts VALUES(100,1000.50),(101, 2000.50),(102, 1500.00);
SELECT * FROM accounts;
```

```
SELECT * FROM total_bal;
INSERT INTO accounts VALUES(103,1000.50),(104, 2000.50),(105, 1500.00);
SELECT * FROM accounts;
SELECT * FROM total_bal;
Δεύτερη λύση
DROP DATABASE if exists my_accounts;
CREATE DATABASE my_accounts;
USE my_accounts;
DROP TABLE IF EXISTS accounts;
CREATE TABLE accounts (acct_num INT, amount DECIMAL(12,2));
CREATE TABLE total_bal(total_sum DECIMAL(12,2));
INSERT INTO total_bal VALUES(0);
DROP TRIGGER IF EXISTS calc_sum;
```

```
DELIMITER #
CREATE TRIGGER calc_sum
BEFORE INSERT ON accounts
FOR EACH ROW
BEGIN
  Update total_bal
  SET total_sum= total_sum + NEW.amount;
END;
#
DELIMITER ;
# TESTING
INSERT INTO accounts VALUES(100,1000.50),(101, 2000.50),(102, 1500.00);
SELECT * FROM accounts;
SELECT * FROM total_bal;
INSERT INTO accounts VALUES(103,1000.50),(104, 2000.50),(105, 1500.00);
SELECT * FROM accounts;
SELECT * FROM total_bal;
```

9.5 Οριζόμενες από το χρήστη συναρτήσεις (User-defined functions, “stored functions”) σε περιβάλλον mySQL

Περιγράφουμε σύντομα το πλαίσιο:

- Οι διάλεκτοι SQL (dialects) των προϊόντων DBMS περιλαμβάνουν πολλές ενσωματωμένες (built-in) functions, π.χ., substr, lower, upper.
- Στο πρότυπο SQL/PSM (SQL/Persistent Stored Modules) ορίζονται: aggregate, arithmetic, temporal, string functions. Παραδείγματα περιλαμβάνουν υπολογισμούς, μετασχηματισμούς δεδομένων κ.λπ.
- Στο πρότυπο SQL/PSM παρέχεται το πλαίσιο για user-defined functions (Stored Functions): Οι οριζόμενες από το χρήστη συναρτήσεις μπορούν να δημιουργηθούν και να χρησιμοποιηθούν για να επεκτείνουν τη γλώσσα SQL.
- Οι Stored functions καλούνται από (are invoked by) δηλώσεις SQL.

9.5.1 Σύνταξη δηλώσεων CREATE PROCEDURE και CREATE FUNCTION

CREATE
[DEFINER = { user | CURRENT_USER }]
PROCEDURE sp_name ([proc_parameter[,...]])
[characteristic ...] routine_body
CREATE
[DEFINER = { user | CURRENT_USER }]
FUNCTION sp_name ([func_parameter[,...]])
RETURNS type
[characteristic ...] routine_body
proc_parameter:
[IN | OUT | INOUT] param_name type
func_parameter:
param_name type
type:
Any valid MySQL data type
characteristic:
COMMENT 'string'
| LANGUAGE SQL
| [NOT] DETERMINISTIC
| { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
| SQL SECURITY { DEFINER | INVOKER }
routine_body:
Valid SQL routine statement

Παραθέτουμε επεξήγηση των συμβόλων { } και / που χρησιμοποιούνται στη σύνταξη των δηλώσεων:

{ user | CURRENT_USER } – σημαίνει μόνο ένα από τα user, CURRENT_USER

[IN | OUT | INOUT] – σημαίνει προαιρετική χρήση (optional use)

9.5.2 Πως καλούνται οι συναρτήσεις γενικά

Ακολουθούν απλά παραδείγματα.

Εφαρμογή συνάρτησης length σε συμβολοσειρά.

```
SELECT length('Hello'), lower('Hello'), upper('Hello');
```

length('Hello')	lower('Hello')	upper('Hello')
5	hello	HELLO

Εφαρμογή συναρτήσεων length, lower, upper σε στήλη πίνακα.

```
SELECT length('Hello'), lower('Hello'), upper('Hello') FROM DUAL;
```

length('Hello')	lower('Hello')	upper('Hello')
5	hello	HELLO

```
SELECT custno, cust_name, length(cust_name), lower(cust_name), upper(cust_name)
FROM customers;
```

custno	cust_name	length(cust_name)	lower(cust_name)	upper(cust_name)
10	Codd	4	codd	CODD
20	Date	4	date	DATE
30	Navathe	7	navathe	NAVATHE

9.5.3 Αρχή με παράδειγμα ορισμού συνάρτησης που υπολογίζει το n!=1*2*…*n

Στον πίνακα 9.3 βλέπουμε την κατασκευή συνάρτησης για τον υπολογισμό του γινομένου του παραγοντικού (n!) για n τύπου INT και n τύπου BIGINT.

Πίνακας 9.3 Υπολογισμός παραγοντικού (n!) για n τύπου INT και n τύπου BIGINT.

DROP FUNCTION IF EXISTS factorial;
DELIMITER !
CREATE FUNCTION factorial(n INT)
RETURNS INT
DETERMINISTIC
BEGIN
DECLARE f INT DEFAULT 1;
WHILE n > 0 DO
SET f = n * f;
SET n = n - 1;
END WHILE;
RETURN f;
END !
DELIMITER ;
	DROP FUNCTION IF EXISTS factorial;
	DELIMITER !
	CREATE FUNCTION factorial(n BIGINT)
	RETURNS BIGINT
	DETERMINISTIC
	BEGIN
	DECLARE f BIGINT DEFAULT 1;
	WHILE n > 0 DO
	SET f = n * f;
	SET n = n - 1;
	END WHILE;
	RETURN f;
	END !
	DELIMITER ;

Δοκιμή (τιμές nє [0,12]) `SELECT factorial(4);`	**Δοκιμή (τιμές nє [0,20]** `SELECT factorial(15);`

SELECT factorial(4);

Factorial(4)
24

SELECT factorial(15);

Factorial(15)
1307674368000

Ακολουθεί ο πίνακας 9.4 τύπων δεδομένων για ακέραιους αριθμούς.

Πίνακας 9.4 Πίνακας τύπων δεδομένων για ακέραιους αριθμούς

Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	Maximum Value Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^63	0	2^63-1	2^64-1

9.5.4 Χρήση συναρτήσεων (functions)

Θα ξεκινήσουμε με τη συγγραφή συνάρτησης που θα εμφανίζει μήνυμα. Η συνάρτηση hello_message καλείται και δείχνει τη λέξη Hello και μήνυμα. Για παράδειγμα:

```
Select hello_message('GREECE');
 Hello GREECE
DROP FUNCTION IF EXISTS hello_message;
DELIMITER $$
CREATE FUNCTION hello_message(message TEXT)
  RETURNS TEXT
BEGIN
  RETURN CONCAT('Hello ', message);
END;
$$
DELIMITER ;
Select hello_message('GREECE');
```

Άσκηση 1

Γράψτε συνάρτηση μετατροπής βαθμών Celsius σε βαθμούς Fahrenheit.

[T(°F) – 32]*5= [T(°C) * 9}

Άσκηση 2

Υπολογισμός περιφέρειας και εμβαδού κύκλου δοθείσης ακτίνας r.

Εμβαδόν κύκλου $E = \pi \times r^2$

Περιφέρεια κύκλου $E = 2 \times \pi \times r$

Άσκηση 3

Υπολογισμός εμβαδού τριγώνου από τις πλευρές του με τον τύπο του Ήρωνα.

E=SQRT(τ× (τ−a) × (τ−b) × (τ−c)), τ=(a+b+c)/2

9.5.5 Χρήση τοπικής μεταβλητής για υπολογισμούς μέσα στη συνάρτηση

Θα χρησιμοποιήσουμε μια τοπική μεταβλητή για να κάνουμε κάποιους υπολογισμούς μέσα στη συνάρτηση.

```
DROP FUNCTION IF EXISTS hello_message;
DELIMITER $$
CREATE FUNCTION hello_message(message TEXT)
  RETURNS TEXT
BEGIN
  DECLARE string_length INT;
  SET string_length = LENGTH(message);
  RETURN CONCAT('Hello ', message, ' - your message has ', string_length, '
characters');
END;
$$
DELIMITER ;
Select hello_message('GREECE');
```

9.5.5.1 Υπολογισμός τόκου (interest) τραπεζικού λογαριασμού με συνάρτηση

Στον πίνακα 9.5 βλέπουμε τον ορισμό συνάρτησης με δύο τρόπους για τον υπολογισμό τόκου (interest) τραπεζικού λογαριασμού.

Πίνακας 9.5 Υπολογισμός τόκου (interest) τραπεζικού λογαριασμού με συνάρτηση

DROP FUNCTION IF EXISTS calculate_balance;
DELIMITER !
CREATE FUNCTION calculate_balance
(InterestRate DECIMAL(10,2),
balance DECIMAL(10,2))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
RETURN (interestRate * balance) / 100;
END !
DELIMITER ;
 	
# testing 	
`SELECT calculate_balance(5, 2110) AS interest;`
	DROP FUNCTION IF EXISTS calculate_balance;
	DELIMITER !
	CREATE FUNCTION calculate_balance
	(InterestRate DECIMAL(10,2),
	balance DECIMAL(10,2))
	RETURNS DECIMAL(10,2)
	DETERMINISTIC
	BEGIN
	DECLARE interest DECIMAL(10,2);
	SET interest = (interestRate * balance) / 100;
	RETURN interest;
	END !
	DELIMITER ;
	# testing
	`SELECT calculate_balance(5, 2110);`

Δοκιμή συνάρτησης

```
SELECT calculate_balance(5, 2110) AS INTEREST;
```

interest
105.50

9.6 Δημιουργία Procedures

Στην ενότητα 9.5.1 είδαμε τη σύνταξη της δήλωσης CREATE PROCEDURE.

9.6.1 Δήλωση μεταβλητών (Variables' declaration) και εκχώρηση τιμών σε αυτές (values' assignment)

Παραθέτουμε παραδείγματα στα οποία φαίνεται πως δηλώνουμε μεταβλητές (Variables' declaration) και εκχωρούμε τιμές σε αυτές (values' assignment), Η γενική μορφή της δήλωσης είναι:

```
DECLARE variable_name datatype(size) DEFAULT default_value;
```

Παράδειγμα 1

```
DECLARE total_sales INT DEFAULT 0
```

Παράδειγμα 2

```
DECLARE total_count INT DEFAULT 0
SET total_count = 10;
```

Παράδειγμα 3

```
DECLARE total_products INT DEFAULT 0
SELECT COUNT(*) INTO total_products FROM products;
```

9.6.2 Κλήση (εκτέλεση) Procedure

Η σύνταξη είναι απλή.

```
CALL STORED_PROCEDURE_NAME();
```

Παράδειγμα κλήσης

```
CALL GetAllPapers(); -- execution
```

9.6.2.1 Υπολογισμός τόκου (interest) και κεφαλαίου (balance) με PROCEDURE

```
DROP PROCEDURE IF EXISTS balanceCalc;
DELIMITER !
CREATE PROCEDURE balanceCalc ( IN interestRate DECIMAL(10,2),
  INOUT balance DECIMAL(10,2),
  OUT interest DECIMAL(10,2))
DETERMINISTIC
BEGIN
```

```
SET interest = (interestRate * balance) / 100;
SET balance = balance + interest;
END !
DELIMITER ;
```

Δοκιμή

```
SET @balance=2000;
SET @interestRate=5;
Select @balance;
```

@balance
2000

```
CALL balanceCalc(@interestRate, @balance, @interest);
Select @interestRate, @balance, @interest;
```

@interestRate	@balance	@interest
5	2100.00	100.00

9.6.3 Τοπικές μεταβλητές για υπολογισμούς μέσα σε procedure

Δημιουργία εκ νέου της procedure.

```
DROP PROCEDURE my_procedure_Local_Variables;
DELIMITER $$
CREATE PROCEDURE my_procedure_Local_Variables()
BEGIN
# ορισμός τοπικών μεταβλητών
SET @x = 25;
SET @y = 10;
SELECT @x, @y, @x*@y;
END $$
DELIMITER ;
```

Ακολουθεί κλήση (εκτέλεση(και τα αποτελέσματα.

```
CALL my_procedure_Local_Variables();
```

@x	@y	@x*@y
25	10	250

9.7 Διαχείριση ακεραιότητας παραγγελιών με triggers και procedures

Ακολουθούν παραδείγματα προγραμματισμού που αν επεκταθούν διασφαλίζουν (μαζί με τον κατάλληλο ορισμό των πινάκων (CREATE TABLE) και τον ορισμό κατάλληλων PRIMARY KEY και FOREIGN KEY, την ισχύ του πρώτου (integrity rule one or the entity integrity rule) και του δεύτερου κανόνα ακεραιότητας (referential integrity).

Με τον trigger delete_orders διαγράφουμε μία (ή πολλές) παραγγελία και τις γραμμές της (τους). Ο trigger αφυπνίζεται από δήλωση DELETE FROM orders WHERE …; Η δήλωση DELETE διαγράφει τα βασικά στοιχεία της παραγγελίας και ο trigger διαγράφει τις γραμμές της.

Με την procedure delete_orders_orderlines διαβάζουμε τον κωδικό της παραγγελίας και διαγράφουμε τις γραμμές της παραγγελίας και την παραγγελία. Στην procedure μπορούν εύκολα να προστεθούν κατάλληλοι έλεγχοι. Σχετικές τεχνικές θα δούμε στη συνέχεια (στην επόμενη procedure).

Με την procedure add_order_line εισάγουμε γραμμές παραγγελίας στον πίνακα orderlines μόνον αν υπάρχει η παραγγελία ήδη στον πίνακα orders. Δεν επιτρέπουμε την εισαγωγή γραμμών για παραγγελία που δεν υπάρχει στον πίνακα orders.

```
# δημιουργία εκ νέου της βάσης δεδομένων
DROP DATABASE IF EXISTS delete_orders;

CREATE DATABASE delete_orders;

USE delete_orders;

CREATE TABLE orders (orderno INT AUTO_INCREMENT NOT NULL, custno INT NOT NULL,
odate DATETIME NOT NULL, PRIMARY KEY(orderno));
```

```
CREATE TABLE orderlines (orderno INT NOT NULL, stockno INT NOT NULL, qty INT NOT
NULL,
   PRIMARY KEY (orderno,stockno));

DESCRIBE orders;
DESCRIBE orderlines;
```

```
SET AUTOCOMMIT=0; -- start transaction processing

INSERT INTO orders (custno,odate) VALUES (1,current_timestamp);
INSERT INTO orderlines (orderno,stockno,qty) VALUES (1,10,1),(1,50,2);

COMMIT;

SELECT * FROM orders;
SELECT * FROM orderlines;
```

```
DROP TRIGGER IF EXISTS delete_orders;

CREATE TRIGGER delete_orders
AFTER DELETE ON orders
FOR EACH ROW
DELETE FROM orderlines WHERE orderno= OLD.orderno;
/* testing */
```

Αν προσθέσετε ξένο κλειδί στον πίνακα orderlines αλλάζει οτιδήποτε;

Ορισμός Procedure για τη διαγραφή παραγελίας

```
Use delete_orders;
DROP PROCEDURE delete_orders_orderlines;
```

```
DELIMITER $
CREATE PROCEDURE  delete_orders_orderlines(IN del_order INT)
BEGIN
  DELETE FROM orderlines WHERE orderno=del_order;
  DELETE FROM orders WHERE orderno=del_order;
END;
$
Delimiter ;
```

```
/* testing */
Set autocommit=0;
select * from orders;
select * from orderlines;
CALL delete_orders_orderlines(1);
-- διαγραφή της παραγγελίας με κωδικό 1 και των γραμμών της
select * from orders;
select * from orderlines;
rollback;
select * from orders;
select * from orderlines;
```

Προσθήκη γραμμής παραγγελίας μόνο αν η παραγγελία υπάρχει

```
use delete_orders;
drop table if exists orders;
create table orders (orderno int auto_increment not null, custno int not null,
odate datetime not null, primary key(orderno));
drop table if exists orderlines;
create table orderlines (orderno int not null, stockno int not null, qty int not
null,
   primary key (orderno,stockno));
DESCRIBE orders;
DESCRIBE orderlines;
```

```
SET AUTOCOMMIT=0; -- start transaction
```

```
insert into orders (custno,odate) values (1,current_timestamp);
insert into orderlines (orderno,stockno,qty) values (1,10,1),(1,50,2);


# επικύρωση μεταβολών
COMMIT;
```

Η παραγγελία με orderno=1 του πελάτη με custno=1 «απλώνεται» σε δύο πίνακες και αποτελείται από μία γραμμή βασικών στοιχείων και δύο γραμμές παραγγελίας.

```
select * from orders;
```

Orderno	Custno	O_date
1	1	2018-06-02 14:01:28

```
select * from orderlines;
```

Orderno	Stockno	qty
1	10	1
1	50	2

```
DROP PROCEDURE IF EXISTS add_order_line;
```

```
DELIMITER $
CREATE PROCEDURE add_order_line(IN o_no int, IN s_no int, IN qty int)
BEGIN
DECLARE count_var INT;
SELECT COUNT(orderno)
-- if count_var=1 then the order number (o_no) is found in orders table
INTO count_var
FROM orders
WHERE orderno=o_no;
IF (count_var <>1) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT='Order number is not found in orders table';
ELSE
   INSERT INTO orderlines VALUES (o_no, s_no, qty);
END IF;
END;
$
DELIMITER ;
```

```
CALL add_order_line(2,10,1);

ERROR 1644(45000): Order number is not found in orders table
```

```
SELECT * FROM Orders;
SELECT * FROM Orderlines;
```

```
insert into orders (custno,odate) values (1,current_timestamp);
CALL add_order_line(2,10,100);
CALL add_order_line(2,20,150);
```

```
select * from orders;
```

Orderno	Custno	O_date
1	1	2022-05-22 14:01:28

2	1	2022-05-22 14:03:21

```
select * from orderlines;
```

Orderno	Stockno	qty
1	10	1
1	50	2
2	10	100
2	20	150

9.8 Θέμα. Δημιουργία Procedure για βάση δεδομένων συγγραφέων

Δημιουργήστε τη βάση publications που περιλαμβάνει τον παρακάτω πίνακα συγγραφέων και τη χώρα κατοικίας τους

```
DROP TABLE IF EXISTS author;
CREATE TABLE author(a_id INT NOT NULL,
                    name CHAR(10), surname CHAR(15), country CHAR(10));

INSERT INTO author VALUES (1, 'PETROS', 'BELSIS', 'GREECE');
INSERT INTO author VALUES (2, 'NIKITAS', 'KARANIKOLAS', 'GREECE');
INSERT INTO author VALUES (3, 'CHRISTOS', 'SKOURLAS', 'GREECE');
INSERT INTO author VALUES (4, 'TASOS', 'TSOLAKIDIS', 'GREECE');
INSERT INTO author VALUES (5, 'DIMITRIS', 'VASSIS', 'UK');
```

```
SELECT * FROM author;
```

A_id	Name	Surname	Country
1	Petros	Belsis	GREECE
2	Nikitas	Karanikolas	GREECE
3	Christos	Skourlas	GREECE
4	Tasos	Tsolakidis	GREECE
5	Dimitris	Vassis	UK

Γράψτε την procedure GetAuthorByCountry που θα υπολογίζει τους συγγραφείς του πίνακα που κατοικούν σε μια χώρα.

```
DELIMITER //
CREATE PROCEDURE GetAuthorByCountry(IN countryName VARCHAR(255))
BEGIN
   SELECT *
   FROM author
   WHERE country = countryName;
END //
 DELIMITER ;
```

Δοκιμή

```
CALL GetAuthorByCountry('GREECE');
```

A_id	Name	Surname	Country
1	Petros	Belsis	GREECE
2	Nikitas	Karanikolas	GREECE
3	Christos	Skourlas	GREECE
4	Tasos	Tsolakidis	GREECE

Γράψτε την procedure CountAuthorsByCountry που θα υπολογίζει πόσοι είναι οι συγγραφείς μιας χώρας. Στην ενδεικτική λύση που παρατίθεται δείτε τις κλήσεις της procedure.

```
DELIMITER $$
CREATE PROCEDURE CountAuthorsByCountry(
  IN AuthorCountry VARCHAR(25),
  OUT total INT)
BEGIN
   SELECT count(A_ID)
   INTO total
   FROM author
   WHERE country = AuthorCountry;
END$$
 DELIMITER ;
```

Δοκιμή

```
CALL CountAuthorsByCountry('GREECE',@total);

Select @total;
```

@total
4

```
 Call CountAuthorsByCountry ('UK', @total);
Select @total AS TOTAL_BY_UK  FROM DUAL;
```

@total_BY_UK
1

9.9 Διαδικαστικές επεκτάσεις (procedural extensions) της γλώσσας SQL. Εντολές IF, CASE, WHILE, REPEAT, LOOP

Οι διαδικαστικές επεκτάσεις (procedural extensions) της γλώσσας SQL στα προϊόντα ΣΔΒΔ υποστηρίζουν γενικά τις γνωστές δομές ελέγχου (control structure). Ο developer πρέπει να ελέγχει τις διαφορές που υπάρχουν στη σύνταξη των εντολών στα διάφορα προϊόντα.

Ακολουθούν παραδείγματα δομών ελέγχου (control structure) σε mySQL

9.9.1 IF Statement

```
IF expression THEN commands
  [ELSEIF expression THEN commands]
  [ELSE commands]
  END IF;
```

9.9.2 CASE Statement

```
CASE
  WHEN expression THEN commands
  WHEN expression THEN commands
  ELSE commands
  END CASE;
```

9.9.3 WHILE loop

```
 WHILE expression DO
   Statements
END WHILE
```

9.9.4 REPEAT loop

```
REPEAT
  Statements
UNTIL expression
END REPEAT
```

9.9.5 LOOP loop

```
 LOOP
Statements
END LOOP
```

9.10 Συναρτήσεις και διαδικασίες (procedure). Τι συμβαίνει σε περιβάλλον χρήσης συναλλαγών

Οι δηλώσεις COMMIT, ROLLBACK επιτρέπονται σε procedure και απαγορεύονται σε function και trigger. Παραθέτουμε παραδείγματα.

Παράδειγμα 1

Αρχικά θυμίζουμε τη χρήση της συνάρτησης MOD που υπολογίζει το υπόλοιπο της διαίρεσης a/b.

```
SET @p_no=3;
SELECT @p_no, MOD(@p_no,2), @p_no%2;
```

@p_no	MOD(@p_no,2)	@p_no%2
3	1	1

```
SET @p_no=8;
SELECT @p_no, MOD(@p_no,2), @p_no%2;
```

@p_no	MOD(@p_no,2)	@p_no%2
8	0	0

Ζητούμενο

Θα κατασκευάσουμε τον πίνακα myTrace, τον οποίο στη συνέχεια θα διαχειριστούμε με τη procedure myProc. Η procedure περιλαμβάνει παραμετρική δήλωση INSERT INTO myTrace. Θα καλούμε τη procedure myProc με τιμή για τη μεταβλητή p_no και όταν η τιμή της μεταβλητής p_no είναι περιττός αριθμός τότε η γραμμή δεν θα γράφεται στον πίνακα myTrace

Δημιουργία πίνακα.

```
DROP TABLE IF EXISTS myTrace;

CREATE TABLE myTrace ( t_no INT,
t_user CHAR(20),
t_date DATE,
t_time TIME,
t_proc VARCHAR(16), t_what VARCHAR(30));
```

```
DROP PROCEDURE IF EXISTS myProc;

DELIMITER !

CREATE PROCEDURE myProc (IN p_no INT, IN p_in VARCHAR(30),
  OUT p_out VARCHAR(30))
LANGUAGE SQL
BEGIN
SET p_out=p_in;
INSERT INTO myTrace (t_no, t_user, t_date, t_time, t_proc, t_what)
  VALUES (p_no, current_user, current_date, current_time, 'myProc', p_in);

IF (MOD(p_no,2)=0) THEN
```

```
   COMMIT;
ELSE ROLLBACK;
END IF;
END  !
DELIMITER ;
```

```
#Δοκιμή
SET AUTOCOMMIT=0;

CALL myProc(1, 'hello1', @p_out);

SELECT @p_out;
CALL myProc(2, 'hello2', @p_out);
SELECT @p_out;
CALL myProc(3, 'hello3', @p_out);
SELECT @p_out;
CALL myProc(4, 'hello4', @p_out);
SELECT @p_out;
CALL myProc(5, 'hello5', @p_out);
SELECT @p_out;
CALL myProc(6, 'hello6', @p_out);
SELECT @p_out;
CALL myProc(7, 'hello7', @p_out);
```

```
SELECT * FROM myTrace;
```

T_no	T_user	T_date	T_time	T_proc	T_what
2	root@localhost	2018-05-29	16:24:35	Myproc	Hello2
4	root@localhost	2018-05-29	16:24:35	Myproc	Hello4
6	root@localhost	2018-05-29	16:24:35	Myproc	Hello6

9.10.1 Οι δηλώσεις Commit, Roolback δεν επιτρέπονται σε stored function

Ακολουθεί παράδειγμα.

```
DROP FUNCTION IF EXISTS myFun;

DELIMITER !
CREATE FUNCTION myFun (p_no INT, p_in VARCHAR(30))
RETURNS VARCHAR(30)
LANGUAGE SQL
BEGIN
INSERT INTO myTrace (t_no, t_user, t_date, t_time, t_proc, t_what)
  VALUES (p_no, current_user, current_date, current_time, 'myProc', p_in);
IF (MOD(p_no,2)=0) THEN
   COMMIT;
ELSE ROLLBACK;
END IF;
RETURN p_in;
```

```
END  !
DELIMITER ;
```

Η συνάρτηση δεν δημιουργείται και επιστρέφει σφάλμα,

```
ERROR 1422(HY000): Explicit or implicit commit is not allowed in stored function
or trigger
```

Προσοχή! Η συνάρτηση new_Fun από την οποία έχουμε αφαιρέσει τις δηλώσεις COMMIT, ROLLBACK εισάγει στοιχεία στον πίνακα myTrace αλλά βγάζει μήνυμα 'COMMIT' ή 'ROLLBACK'

```
DROP FUNCTION IF EXISTS new_Fun;
DELIMITER !
CREATE FUNCTION new_Fun (p_no INT, p_in VARCHAR(30))
RETURNS VARCHAR(30)
LANGUAGE SQL
BEGIN
INSERT INTO myTrace (t_no, t_user, t_date, t_time, t_proc, t_what)
  VALUES (p_no, current_user, current_date, current_time, 'myProc', p_in);
IF (MOD(p_no,2)=0) THEN
   RETURN 'COMMIT';
ELSE
   RETURN 'ROLLBACK';
END IF;
END  !
DELIMITER ;
SELECT new_fun(10, 'hello10');
SELECT new_fun(11, 'hello11');
```

9.10.2 Χρησιμοποίηση δήλωσης MySQL RESIGNAL σε χειριστή σφάλματος ή προειδοποίησης (error or warning handler)

Εκτός από τη δήλωση SIGNAL το προϊόν MySQL παρέχει και τη δήλωση RESIGNAL για την εμφάνιση σφάλματος ή προειδοποίησης. Οι δηλώσεις είναι παρόμοιες αλλά η δήλωση RESIGNAL χρησιμοποιείται μόνο με χειριστή (error or warning handler).

Ακολουθεί παράδειγμα από το εγχειρίδιο MySQL 8.0 Reference Manual.

MySQL :: MySQL 8.0 Reference Manual :: 13.6.7.4 RESIGNAL Statement

Παράδειγμα

```
DROP TABLE IF EXISTS xx;
delimiter //
CREATE PROCEDURE p ()
BEGIN
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
  BEGIN
SET @error_count = @error_count + 1;
IF @a = 0 THEN RESIGNAL; END IF;
  END;
  DROP TABLE xx;
END//
delimiter ;
```

```
SET @error_count = 0;
SET @a = 0;
CALL p();
```

Δείτε στη συνέχεια τα παραδείγματα της επόμενης υποενότητας.

9.11 Handler στο προϊόν MySQL

Παραθέτουμε τη σύνταξη DECLARE ... HANDLER.

DECLARE handler_action HANDLER

FOR condition_value [, condition_value] ...

statement

handler_action:

CONTINUE -- Execution of the current program continues

| EXIT -- Execution terminates

| UNDO -- Not supported

condition_value:

mysql_error_code

| SQLSTATE [VALUE] sqlstate_value

| condition_name

| SQLWARNING

| NOT FOUND

| SQLEXCEPTION

9.11.1 Παράδειγμα διαχείρισης διαίρεσης δια του 0 με CONTINUE HANDLER.

```
DELIMITER $$
CREATE PROCEDURE Divide_by_zero (IN numerator INT, IN denominator DECIMAL(10,2),
OUT results DECIMAL(10,2))
BEGIN
DECLARE div_by_zero CONDITION FOR SQLSTATE '22012';
DECLARE CONTINUE HANDLER FOR div_by_zero RESIGNAL SET MESSAGE_TEXT ='Division by
zero';
IF denominator = 0 THEN
SIGNAL div_by_zero;
ELSE
SET results := numerator/denominator;
END IF;
END;
$$
DELIMITER ;
```

Δοκιμή

```
CALL Divide_by_zero(100, 0, @results);
```

Εμφανίζει το μήνυμα.

```
ERROR 1644(22012): Division by zero
```

Ακολουθεί η ίδια procedure με παραλλαγή του ορισμού του handler.

```
DROP PROCEDURE IF EXISTS Divide_by_zero;
DELIMITER $$
CREATE PROCEDURE Divide_by_zero (IN numerator INT, IN denominator DECIMAL(10,2),
OUT results DECIMAL(10,2))
BEGIN
DECLARE CONTINUE HANDLER FOR SQLSTATE '22012' RESIGNAL SET MESSAGE_TEXT
='Division by zero';
IF denominator = 0 THEN
SIGNAL SQLSTATE '22012';
ELSE
SET results := numerator/denominator;
END IF;
END;
$$
DELIMITER ;
```

Δοκιμή

```
CALL Divide_by_zero(100, 0, @results);
```

Η διαφορά στον ορισμό του HANDLER φαίνεται στον πίνακα 9.6.

Πίνακας 9.6 Διαφορετικοί ορισμοί HANDLER

DECLARE div_by_zero CONDITION FOR SQLSTATE '22012'; DECLARE CONTINUE HANDLER FOR div_by_zero RESIGNAL SET MESSAGE_TEXT ='Division by zero'; … SIGNAL div_by_zero; …
DECLARE CONTINUE HANDLER FOR SQLSTATE '22012' RESIGNAL SET MESSAGE_TEXT ='Division by zero'; … SIGNAL SQLSTATE '22012'; …

9.12 Διαχείριση Cursor με χρήση function

Η παρουσίαση της έννοιας cursor και η διαχείριση cursor θα βασιστεί σε παράδειγμα. Η βάση δεδομένων training αποτελείται από τους πίνακες καθηγητών, μαθημάτων. Υποτίθεται ότι ένα μάθημα διδάσκεται από πολλούς καθηγητές και ένας καθηγητής κάνει ένα μάθημα αποκλειστικά και μόνο.

Lecturer

Lecturer_id	Lecturer_surname	Lecturer_name	City	salary	Course_id

Course

Course_id	course_name

Θα γράψουμε τη συνάρτηση lecturer_list() που θα δείχνει τα επώνυμα των καθηγητών όπως παρακάτω:

DATE, WIEDERHOLD, CHEN, ULLMAN, CODD,

Η συνάρτηση θα διαχειρίζεται τον CURSOR με όνομα my_cursor και θα περιλαμβάνει χειριστή CONTINUE HANDLER:

```
DECLARE my_cursor CURSOR
FOR SELECT lecturer_name, lecturer_surname
FROM lecturer;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET record_not_found = 1;
```

Προσοχή!

Αν χρησιμοποιήσουμε EXIT HANDLER τότε έχουμε:

```
ERROR 1321 (21005): FUNCTION lecturer_list ended without RETURN.
```

Δημιουργία βάσης δεδομένων και πινάκων

```
Drop database IF EXISTS training;
CREATE DATABASE training;
USE training;
CREATE TABLE lecturer(lecturer_id int(3), lecturer_surname varchar(15),
lecturer_name varchar(15), city varchar(15), salary decimal (8,2),  course_id
int);
CREATE TABLE course(course_id int, course_name varchar(50));
```

Εισαγωγή στοιχείων

```
INSERT INTO course VALUES (1, 'DATABASE');
INSERT INTO course VALUES (2, 'WEB DEVELOPMENT');
INSERT INTO course VALUES (3, 'DATA MINING');
INSERT INTO course VALUES (4, 'SEMANTIC WEB');
```

```
Select * From COURSE;
```

Course_id	course_name
1	DATABASE
2	WEB DEVELOPMENT
3	DATA MINING
4	SEMANTIC WEB

```
INSERT INTO lecturer(lecturer_id, lecturer_name, lecturer_surname, city, salary,
course_id) VALUES (1, 'CHRIS', 'DATE', 'LONDON', 2000, 1), (2, 'GIO',
```

```
'WIEDERHOLD', 'ATHENS', 1500, 1), (3, 'PETER', 'CHEN', 'ATHENS', 3500, 2), (4,
'JEFF', 'ULLMAN', 'ATHENS', 1700, 1), (5,  'TED', 'CODD', 'ATHENS', 2500, 2);
SELECT * FROM LECTURER;
```

Lecturer_id	Lecturer_surname	Lecturer_name	City	salary	Course_id
1	DATE	CHRIS	LONDON	2000	1
2	WIEDERHOLD	GIO	ATHENS	1500	1
3	CHEN	PETER	ATHENS	3500	2
4	ULLMAN	JEFF	ATHENS	1700	1
5	CODD	TED	ATHENS	2500	2

```
SELECT lecturer_id, lecturer_surname, lecturer_name, course_id FROM lecturer;
```

Lecturer_id	Lecturer_surname	Lecturer_name	Course_id
1	DATE	CHRIS	1
2	WIEDERHOLD	GIO	1
3	CHEN	PETER	2
4	ULLMAN	JEFF	1
5	CODD	TED	2

Κατασκευή της συνάρτησης lecturer_list

```
DELIMITER //
CREATE FUNCTION lecturer_list() RETURNS VARCHAR(255)
BEGIN
DECLARE record_not_found INTEGER DEFAULT 0;
DECLARE lecturer_name_var VARCHAR(150) DEFAULT "";
DECLARE lecturer_surname_var VARCHAR(150) DEFAULT "";
DECLARE lect_list VARCHAR(255) DEFAULT "";
DECLARE my_cursor CURSOR FOR SELECT lecturer_name, lecturer_surname  FROM
lecturer;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET record_not_found = 1;
OPEN my_cursor;
 allLecturers: LOOP
  FETCH my_cursor INTO lecturer_name_var, lecturer_surname_var;
  IF record_not_found THEN
   LEAVE allLecturers;
  END IF;
 SET lect_list = CONCAT(lect_list, lecturer_surname_var, ", ");
   END LOOP allLecturers;
   CLOSE my_cursor;
   RETURN SUBSTR(lect_list, 1, 70);
   END //
DELIMITER ;
-- Execute function
SELECT lecturer_list();
```

lecturer_list()
DATE, WIEDERHOLD, CHEN, ULLMAN, CODD,

9.13 Διαχείριση Cursor με χρήση procedure

Δημιουργία εκ νέου βάσης δεδομένων παραγγελιών.

```
drop database if exists orders_db;

create database orders_db;

use orders_db;

drop table if exists orders;

create table orders (order_num int not null, custno int not null, odate datetime
DEFAULT now(), total int(10), primary key(order_num));

SET AUTOCOMMIT=0;

insert into orders values (1,1,current_timestamp,1000);
insert into orders values (2,1,current_timestamp,5000);
insert into orders values (3,2,current_timestamp,6000);
insert into orders values (4,1,current_timestamp,7000);
insert into orders values (5,4,current_timestamp,1000);
insert into orders values (6,3,current_timestamp,2000);
insert into orders values (7,2,current_timestamp,4000);
insert into orders values (8,1,current_timestamp,6000);
insert into orders values (9,1,current_timestamp,3000);
insert into orders values (10,2,current_timestamp,7000);

commit;

select * from orders;

DROP TABLE IF EXISTS temp;

CREATE TABLE temp(t_num int NOT NULL auto_increment,
  orderno INT, total INT, PRIMARY KEY(t_num));
```

Η procedure processorders βρίσκει όλες τις παραγγελίες με συνολικό ποσό μεγαλύτερο των 3500 ευρώ και ενημερώνει σχετικά έναν πίνακα με όνομα temp.

```
DROP PROCEDURE IF EXISTS processorders;
DELIMITER $
CREATE PROCEDURE processorders()
BEGIN
   -- Declare local variables
   DECLARE done BOOLEAN DEFAULT 0; -- FALSE, NOT DONE
   DECLARE o_num INT; -- FETCH is used to retrieve the current order_num
  -- into the declared variable named o_num
   DECLARE o_total INT; -- FETCH is used to retrieve the current total
  -- into the declared variable named o_total
   -- Declare the cursor
   DECLARE ordernumbers CURSOR
   FOR SELECT order_num, total FROM orders;
```

```
   -- Declare continue handler
   DECLARE EXIT HANDLER FOR SQLSTATE '02000' SET done=1;
-- when SQLSTATE '02000' occurs, then SET done=1
-- SQLSTATE '02000' is a not found condition
   -- Open the cursor
   OPEN ordernumbers;
   -- Loop through all rows
   REPEAT
  -- Get order number and total
  FETCH ordernumbers INTO o_num, o_total;
 IF o_total>3500 THEN
  INSERT INTO temp(orderno, total) VALUES(o_num, o_total);
END IF;
   -- End of loop
   UNTIL done END REPEAT;
   -- Close the cursor
   CLOSE ordernumbers;
END;
$
DELIMITER ;
CALL processorders();
```

```
SELECT * FROM temp;
```

T_num	Orderno	Total
1	2	5000
2	3	6000
3	4	7000
4	7	4000
5	8	6000
6	10	7000

Προσοχή! Αν χρησιμοποιήσω

```
   DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done=1;
```

Τότε το αποτέλεσμα είναι λανθασμένο.

T_num	Orderno	Total
1	2	5000
2	3	6000
3	4	7000
4	7	4000
5	8	6000
6	10	7000
7	10	7000

9.14 Πως θα αντιμετωπίσουμε Error 1442 στο προϊόν MySQL

Σε περιπτώσεις που ενεργοποιείται ένας trigger από κατάλληλη δήλωση SQL σε έναν πίνακα που έχει όνομα table_name (π.χ. δήλωση AFTER DELETE ON table_name) και θέλουμε να μεταβάλουμε τα στοιχεία αυτού του ίδιου του πίνακα (table_name) **δεν μπορούμε να χρησιμοποιήσουμε τον trigger μας.** Δηλαδή, απαγορεύεται το εξής:

```
CREATE TRIGGER erroneous_trigger
AFTER UPDATE ON table_name
FOR EACH ROW
UPDATE table_name
SET …
;
```

Αν το χρησιμοποιήσουμε θα δημιουργηθεί ο trigger αλλά όταν κάνουμε αλλαγές στον πίνακά μας θα δούμε το μήνυμα:

```
Error 1442 (HY000): Can't update table in stored function/trigger because it is
already used by statement which invoked this stored function/trigger
```

Παράδειγμα εμφάνισης Error 1442

Έστω η βάση προσωπικού error_1442 που περιλαμβάνει τους πίνακες emp, dept.

emp (πίνακας υπαλλήλων)

Empno	Ename	Sal	Deptno
10	CODD	2000	10
15	ELMASRI	1200	10
20	NAVATHE	2000	20
30	DATE	1800	10

dept (πίνακας τμημάτων)

Deptno	Dname
10	ACCOUNTING
20	SALES

Δημιουργία της βάσης δεδομένων error_1442 και των πινάκων της Dept, Emp

```
DROP DATABASE IF EXISTS error_1442;

CREATE DATABASE error_1442;

USE error_1442;

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

CREATE TABLE EMP(EMPNO INT(4) NOT NULL,
ENAME VARCHAR(10), SAL DECIMAL(7,2),
DEPTNO INT(2));
```

```
SHOW TABLES;

INSERT INTO DEPT(DEPTNO, DNAME) VALUES (10, 'ACCOUNTING');
INSERT INTO DEPT(DEPTNO, DNAME) VALUES (20, 'SALES');
INSERT INTO EMP VALUES (10, 'CODD', 2000, 10);
INSERT INTO EMP VALUES (15, 'ELMASRI', 1200, 10);
INSERT INTO EMP VALUES (20, 'NAVATHE', 2000, 20);
INSERT INTO EMP VALUES (30, 'DATE', 1800, 10);

SELECT * FROM EMP;
SELECT * FROM DEPT;

DELIMITER %

CREATE TRIGGER emp_sal
AFTER UPDATE ON emp
FOR EACH ROW
BEGIN
IF NEW.SAL>2000 THEN
 UPDATE emp
  SET sal=2000
  WHERE empno=NEW.empno;
END IF;
END;

%
DELIMITER ;
-- testing

UPDATE emp
SET sal=sal*1.2;
```

```
Error 1442 (HY000): Can't update table in stored function/trigger because it is
already used by statement which invoked this stored function/trigger
```

Λύση του προβλήματος είναι να γράψουμε την κατάλληλη procedure.

Προσοχή! Αν θέλουμε μπορούμε να χρησιμοποιήσουμε trigger για να αλλάξουμε κάτι σε άλλον πίνακα. Τότε δεν υπάρχει πρόβλημα!

Δηλαδή, δεν απαγορεύεται το εξής:

```
CREATE TRIGGER erroneous_trigger
AFTER UPDATE ON table_name
FOR EACH ROW
UPDATE other_table_name
SET …
;
```

9.15 MySQL stored procedure vs function. Πότε χρησιμοποιούμε procedure και πότε function

Παραθέτουμε διαφορές:

- Δεν μπορούμε να χρησιμοποιήσουμε stored procedures με τις συνηθισμένες δηλώσεις SQL.
- Χρησιμοποιούμε stored functions, όπως ακριβώς χρησιμοποιούμε τις συνήθεις συναρτήσεις (built-in functions), δηλαδή με τις δηλώσεις SQL. Στο επόμενο παράδειγμα, η συνάρτηση συμμετέχει σε μία έκφραση (expression) που χρησιμοποιείται σε SELECT:

```
SELECT DaysBetween(startdate, enddate)/30, contract_id, customer_id
FROM contracts;
```

- Η συνάρτηση «συμμετέχει» σε μία έκφραση (expression) και επιστρέφει μία τιμή (a single value) για να χρησιμοποιηθεί στον υπολογισμό της έκφρασης.
- Δεν μπορούμε να καλέσουμε συνάρτηση με δήλωση CALL και μία procedure δεν μπορεί να κληθεί από μία έκφραση (expression).
- Μία procedure καλείται με δήλωση CALL συνήθως για να τροποποιήσει κάποιον πίνακα ή να επεξεργαστεί ανακτηθέντα δεδομένα από πίνακα.
- Συνήθως οι Functions χρησιμοποιούνται σε υπολογισμούς ενώ οι procedures για την εκτέλεση επιχειρησιακής λογικής (for executing business logic).
- Όταν ορίζετε μια αποθηκευμένη συνάρτηση ή έναυσμα (stored function, trigger) δεν επιτρέπεται να τροποποιήσετε έναν πίνακα που θέλετε να χρησιμοποιείται (για ανάγνωση ή μεταβολή) από δήλωση που χρησιμοποιεί τη συνάρτηση (function) ή δήλωση που «αφυπνίζει» το έναυσμα (trigger).

9.16 Σύνοψη διαφορών στη σύνταξη procedure και function

Παραθέτουμε διαφορές:

1) Οι **παράμετροι (parameters)** σε Procedure μπορεί να είναι **IN (input-only)**, **OUT (output-only)**, ή **INOUT ((input-output)**. Δηλαδή, η procedure μπορεί να επιστρέψει τιμές χρησιμοποιώντας output παραμέτρους. Οι τιμές μπορούν να χρησιμοποιηθούν από τις δηλώσεις που ακολουθούν τη δήλωση CALL. Οι Functions έχουν μόνο input παραμέτρους. Δηλαδή, η δήλωση παραμέτρων γίνεται διαφορετικά σε procedures και functions.
2) Οι Functions επιστρέφουν μία τιμή (return value), άρα στον ορισμό τους πρέπει να υπάρχει RETURNS clause για να καθορίσει τον τύπο δεδομένων (data type) της τιμής που επιστρέφουν. Επίσης, πρέπει να υπάρχει τουλάχιστον μία δήλωση RETURN στο «σώμα» της function για να επιστρέψει την τιμή στον καλούντα. Δεν υπάρχουν RETURNS και RETURN στον ορισμό μιας procedure.
3) Αν οι παράμετροι σε Procedure δεν δηλωθούν ως IN ή OUT ή INOUT τότε θεωρείται ότι είναι IN. Στις παραμέτρους μίας **function δεν γράφουμε IN, OUT, ή INOUT**. Όλες οι παράμετροι μιας function θεωρείται ότι είναι IN (input-only).
4) Μπορούμε να έχουμε **procedures και functions με το ίδιο όνομα σε μία βάση δεδομένων**.
5) Δυναμική SQL έχουμε όταν κατασκευάζουμε δηλώσεις SQL ως strings και στη συνέχεια τις εκτελούμε. Μπορούμε μόνο σε **Stored procedures να χρησιμοποιήσουμε dynamic SQL και όχι σε**

functions ή triggers. Επομένως, δηλώσεις PREPARE, EXECUTE, DEALLOCATE PREPARE μπορούν να χρησιμοποιηθούν μόνο σε stored procedures.

Ενδεικτική βιβλιογραφία

Martti Laiho, Matti Kurki, Malcolm Crowe, Fritz Laux, Dimitris Dervos and Kari Hirvonen (2018), Introduction to Transaction Programming, Draft 2018-12-03, DBTechNet.org, p. 654, διατίθεται με άδεια χρήσης creative commons

MySQL Error Handling in Stored Procedures ("this tutorial shows you how to use MySQL handler to handle exceptions or errors encountered in stored procedures")

http://www.mysqltutorial.org/mysql-error-handling-in-stored-procedures/

Raising Error Conditions with MySQL SIGNAL / RESIGNAL Statements ("in this tutorial, you will learn how to use SIGNAL and RESIGNAL statements to raise error conditions inside stored procedures").

http://www.mysqltutorial.org/mysql-signal-resignal/

MySQL 8.0 Reference Manual

9.17 Περιήγηση στη διαχείριση προβλημάτων από πλεονάζοντα δεδομένα. Περιήγηση στη διαχείριση ακεραιότητας και συνέπειας δεδομένων με χρήση triggers. Καταγραφή ενεργειών χρήστη με trigger.

Συχνά η βάση δεδομένων μιας επιχείρησης έχει πλεονάζοντα δεδομένα, γεγονός που μπορεί να οδηγήσει σε προβλήματα ασυνέπειας δεδομένων, όπως φαίνεται στο παρακάτω παράδειγμα εκπαιδευτικής βάσης δεδομένων (training data base) η οποία περιλαμβανει εισηγητές (lecturer) και μαθήματα (course).

Lecturer

Lecturer_id	Lecturer_name	Lecturer_surname	City	salary	Course_id	Course_name
1	CHRIS	DATE	LONDON	2000	1	DATA BASE
2	GIO	WIEDERHOLD	ATHENS	1500	1	DATABASES
3	PETER	CHEN	ATHENS	3500	2	WEB
4	JEFF	ULLMAN	ATHENS	1700	1	DATABASE
5	TED	CODD	ATHENS	2500	2	WEB

Course

Course_id	course_name
1	DATABASE
2	WEB DEVELOPMENT
3	DATA MINING
4	SEMANTIC WEB

Αυτή η βάση δεδομένων δε βρίσκεται στην τρίτη κανονική μορφή γεγονός που θα οδηγήσει σ προβλήματα.

Άσκηση

Ανασχεδιάστε τη βάση στην τρίτη κανονική μορφή.

Δημιουργία της βάσης δεδομένων training και των πινάκων με πλεονασμούς

```
Drop database training;
CREATE DATABASE training;
USE training;
CREATE TABLE lecturer(lecturer_id int(3), lecturer_surname varchar(15),
lecturer_name varchar(15),
city varchar(15), salary decimal (8,2), course_id int, course_name varchar(15));
CREATE TABLE course(course_id int, course_name varchar(50));
```

Εισαγωγή στοιχείων

```
INSERT INTO course VALUES (1, 'DATABASE');
INSERT INTO course VALUES (2, 'WEB DEVELOPMENT');
INSERT INTO course VALUES (3, 'DATA MINING');
INSERT INTO course VALUES (4, 'SEMANTIC WEB');
INSERT INTO lecturer(lecturer_id, lecturer_name, lecturer_surname, city, salary,
course_id) VALUES
(1, 'CHRIS', 'DATE', 'LONDON', 2000, 1),
(2, 'GIO', 'WIEDERHOLD', 'ATHENS', 1500, 1),
(3, 'PETER', 'CHEN', 'ATHENS', 3500, 2),
(4, 'JEFF', 'ULLMAN', 'ATHENS', 1700, 1),
(5,  'TED', 'CODD', 'ATHENS', 2500, 2);
```

Βλέπουμε τα δεδομένα της βάσης χωρίς την εκτέλεση εναύσματος.

```
Select * From COURSE;
```

```
mysql>
mysql> Select * From COURSE;
+-----------+-----------------+
| course_id | course_name     |
+-----------+-----------------+
|         1 | DATABASE        |
|         2 | WEB DEVELOPMENT |
|         3 | DATA MINING     |
|         4 | SEMANTIC WEB    |
+-----------+-----------------+
4 rows in set (0.03 sec)
```

```
SELECT * FROM LECTURER;
SELECT lecturer_id, lecturer_surname, lecturer_name, course_id, course_name
FROM lecturer;
```

```
mysql>
mysql> SELECT lecturer_id, lecturer_surname, lecturer_name, course_id, course_na
me
    -> FROM lecturer;
+-------------+-----------------+---------------+-----------+-------------+
| lecturer_id | lecturer_surname | lecturer_name | course_id | course_name |
+-------------+-----------------+---------------+-----------+-------------+
|           1 | DATE            | CHRIS         |         1 | NULL        |
|           2 | WIEDERHOLD      | GIO           |         1 | NULL        |
|           3 | CHEN            | PETER         |         2 | NULL        |
|           4 | ULLMAN          | JEFF          |         1 | NULL        |
|           5 | CODD            | TED           |         2 | NULL        |
+-------------+-----------------+---------------+-----------+-------------+
5 rows in set (0.00 sec)

mysql>
```

Επισημαίνουμε ότι με τις δηλώσεις INSERT INTO lecturer θα μπορούσαμε να εισάγουμε τιμές και για το όνομα του μαθήματος (course_name), π.χ.

```
INSERT INTO lecturer(lecturer_id, lecturer_name, lecturer_surname, city, salary,
course_id, course_name) VALUES (1, 'CHRIS', 'DATE', 'LONDON', 2000, 1,
'DATABASE');
```

Στην περίπτωση αυτή όμως από λάθος στην εισαγωγή δεδομένων (data entry) θα είχαμε ασυνέπεια δεδομένων (inconsistent data), π.χ.,

```
INSERT INTO lecturer(lecturer_id, lecturer_name, lecturer_surname, city, salary,
course_id, course_name) VALUES (1, 'CHRIS', 'DATE', 'LONDON', 2000, 1, 'DBASE');
```

Να πως θα επιλύσουμε το πρόβλημα της ασυνέπειας δεδομένων.

Δημιουργία trigger bi_lect_add_dname

```
Delimiter $$
create trigger bi_lect_add_dname
before insert on lecturer
for each row
begin
declare dname_var varchar(40);
   select course_name into dname_var from course where course_id =
new.course_id;
   set new.course_name = dname_var;
end;
$$
Delimiter ;
```

Διαγράφουμε τα δεδομένα του πίνακα lecturer και τα εισάγουμε εκ νέου.

```
DELETE FROM lecturer;
INSERT INTO lecturer(lecturer_id, lecturer_name, lecturer_surname, city, salary,
course_id) VALUES
(1, 'CHRIS', 'DATE', 'LONDON', 2000, 1), (2, 'GIO', 'WIEDERHOLD', 'ATHENS',
1500, 1), (
3, 'PETER', 'CHEN', 'ATHENS', 3500, 2), (4, 'JEFF', 'ULLMAN', 'ATHENS', 1700,
1),
(5,  'TED', 'CODD', 'ATHENS', 2500, 2);
```

Βλέπουμε ότι τα δεδομένα των δύο πινάκων είναι συνεπή.

```
Select * From COURSE;
```

```
mysql>
mysql> Select * From COURSE;
+-----------+-----------------+
| course_id | course_name     |
+-----------+-----------------+
|         1 | DATABASE        |
|         2 | WEB DEVELOPMENT |
|         3 | DATA MINING     |
|         4 | SEMANTIC WEB    |
+-----------+-----------------+
4 rows in set (0.03 sec)
```

```
Select lecturer_id, lecturer_surname, course_id, course_name from lecturer;
```

```
mysql> SELECT lecturer_id, lecturer_surname, lecturer_name, course_id, course_na
me
    -> FROM lecturer;
+-------------+-----------------+---------------+-----------+-----------------+
| lecturer_id | lecturer_surname | lecturer_name | course_id | course_name     |
+-------------+-----------------+---------------+-----------+-----------------+
|           1 | DATE            | CHRIS         |         1 | DATABASE        |
|           2 | WIEDERHOLD      | GIO           |         1 | DATABASE        |
|           3 | CHEN            | PETER         |         2 | WEB DEVELOPMENT |
|           4 | ULLMAN          | JEFF          |         1 | DATABASE        |
|           5 | CODD            | TED           |         2 | WEB DEVELOPMENT |
+-------------+-----------------+---------------+-----------+-----------------+
5 rows in set (0.14 sec)
```

9.18 Παράνομες ενέργειες στη βάση δεδομένων και αντιμετώπισή τους

Θα κατασκευάσουμε το έναυσμα (trigger) bi_lecturer το οποίο προσθέτει στο μισθό του καθηγητή παράνομα ένα ποσό κατά την εισαγωγή των στοιχείων του.

9.18.1 Δημιουργία εναύσματος (trigger) το οποίο προσθέτει παράνομα χρήματα σε μισθό

Ακολουθεί η δημιουργία του εναύσματος (trigger) bi_lecturer. Αρχικά δημιουργούμε εκ νέου τη βάση δεδομένων και εισάγουμε τα στοιχεία κάποιων καθηγητών κανονικά.

```
Drop database training;
```

```
CREATE DATABASE training;
USE training;
CREATE TABLE lecturer(lecturer_id int(3), lecturer_surname varchar(15),
lecturer_name varchar(15), city varchar(15), salary decimal (8,2),  course_id
int, course_name varchar(15));
CREATE TABLE course(course_id int, course_name varchar(50));
INSERT INTO course VALUES (1, 'DATABASE');
INSERT INTO course VALUES (2, 'WEB DEVELOPMENT');
INSERT INTO course VALUES (3, 'DATA MINING');
INSERT INTO course VALUES (4, 'SEMANTIC WEB');
```

Εισάγουμε στοιχεία χωρίς να έχουμε δημιουργήσει το έναυσμα που θα κάνει παράνομες ενέργειες.

```
INSERT INTO lecturer(lecturer_id, lecturer_name, lecturer_surname, city, salary,
course_id) VALUES
(1, 'CHRIS', 'DATE', 'LONDON', 2000, 1), (2, 'GIO', 'WIEDERHOLD', 'ATHENS',
1500, 1),
(3, 'PETER', 'CHEN', 'ATHENS', 3500, 2), (4, 'JEFF', 'ULLMAN', 'ATHENS', 1700,
1);
```

Δεν υπάρχει πρόβλημα στα δεδομένα όπως μπορείτε να διαπιστώσετε με τη δήλωση SELECT.

```
SELECT lecturer_id, lecturer_surname, lecturer_name, salary, course_id FROM
lecturer;
```

```
ysql>
ysql> SELECT lecturer_id, lecturer_surname, lecturer_name, salary, course_id
   -> FROM lecturer;
+-------------+-------------------+---------------+---------+-----------+
| lecturer_id | lecturer_surname  | lecturer_name | salary  | course_id |
+-------------+-------------------+---------------+---------+-----------+
|           1 | DATE              | CHRIS         | 3000.00 |         1 |
|           2 | WIEDERHOLD        | GIO           | 1950.00 |         1 |
|           3 | CHEN              | PETER         | 5250.00 |         2 |
|           4 | ULLMAN            | JEFF          | 2380.00 |         1 |
|           5 | CODD              | TED           | 3250.00 |         2 |
+-------------+-------------------+---------------+---------+-----------+
rows in set (0.00 sec)
```

Ορίζουμε τον trigger bi_lecturer ο οποίος θα οδηγήσει σε παράνομες ενέργειες.

Το έναυσμα κάθε φορά που εισάγουμε στοιχεία νέου καθηγητή «αφυπνίζεται| και προσθέτει στο μισθό του το γινόμενο του μισθού επί το μήκος του ονόματός του. Για παράδειγμα, αν ο μισθός είναι 2500 και ο νέος καθηγητής ονομάζεται TED τότε ο μισθός που εισάγουμε παράνομα είναι (2500+((2500*3)/100)=2575. Όσοι καθηγητές υπάρχουν ήδη στη βάση δεν επηρεάζονται. Αν καταργήσουμε το έναυσμα τα στοιχεία των νέων υπαλλήλων θα εισάγονται κανονικά.

```
DROP TRIGGER IF EXISTS bi_lecturer;
DELIMITER //
CREATE TRIGGER bi_lecturer
BEFORE INSERT ON lecturer
FOR EACH ROW
BEGIN
DECLARE name_l int;
SET name_l=length(NEW.lecturer_name);
SET NEW.salary=NEW.salary+(NEW.salary * name_l)/100;
END;
//
DELIMITER ;
```

```
mysql>
mysql> DELIMITER //
mysql> create trigger bi_lecturer before insert on lecturer for each row
    -> Begin
    ->    declare name_l int;
    ->    set name_l = length(new.lecturer_name);
    ->    set new.salary = new.salary + (new.salary * name_l)/10;
    -> end;
    -> //
Query OK, 0 rows affected (0.11 sec)

mysql> DELIMITER ;
```

Εισαγωγή στοιχείων νέου καθηγητή

```
INSERT INTO lecturer(lecturer_id, lecturer_name, lecturer_surname, city, salary,
course_id)
VALUES (5,  'TED', 'CODD', 'ATHENS', 2500, 2);
```

Δείτε τις τιμές της στήλης salary για να διαπιστώσετε την παρανομία.

```
SELECT lecturer_id, lecturer_surname, lecturer_name, salary, course_id
FROM lecturer;
```

Στη συνέχεια καταργούμε τον trigger.

Ένας τρόπος να διαπιστωθεί η παρανομία είναι η καταγραφή και ο έλεγχος των ενεργειών του χρήστη. Ακολουθεί παράδειγμα trigger που καταγράφει τις ενέργειες χρήστη.

9.18.2 Ορισμός trigger για την καταγραφή ενεργειών του χρήστη με την αυτόματη εισαγωγή στοιχείων σε πίνακα audit

```
Drop table if exists audit;

create table audit (user_name varchar(30), table_name varchar(30),  sal
decimal(8,2), update_date date);

drop trigger bi_lecturer_audit;
DELIMITER //

create trigger bi_lecturer_audit
before insert on lecturer
for each row
begin
   insert into audit (user_name, table_name, sal, update_date) values
(current_user(), 'lecturer', NEW.salary, now());
end;
//
DELIMITER ;

INSERT INTO lecturer(lecturer_id, lecturer_name, lecturer_surname, city, salary,
course_id)
 VALUES (6,  'TED', 'CODD', 'ATHENS', 2500, 2);

SELECT * FROM audit;
```

Βλέπουμε στον πίνακα audit ποιος χρήστης (user_name) έκανε ενέργεια σε κάποιο πίνακα (table_name) και πότε (update_date) , π.χ.,

user_name	table_name	update_date
…….	……	…….
root@localhost	lecturer	2021-10-10
…….	……	…….

9.18.3 Δοκιμή ορισμού εναυσμάτων (triggers) με την ίδια συνθήκη ενεργοποίησης

Προσοχή! Στις μόνο στις νέες version του προϊόντος MySQL επιτρέπεται ο ορισμός εναυσμάτων (triggers) με την ίδια συνθήκη ενεργοποίησης. Επιπλέον σε αυτές μπορούμε να ορίσουμε τη σειρά εκτέλεσης των εναυσμάτων.

```
mysql> DELIMITER //
mysql> create trigger bi_lecturer before insert on lecturer for each row
    -> Begin
    ->    declare name_l int;
    ->    set name_l = length(new.lecturer_name);
    ->    set new.salary = new.salary + (new.salary * name_l)/10;
    -> end;
    -> //
Query OK, 0 rows affected (0.10 sec)

mysql> DELIMITER ;
mysql> DELIMITER //
mysql> create trigger bi_stud_add_dname
    -> before insert on lecturer
    -> for each row
    -> begin
    -> declare dnamev varchar(40);
    ->    select course_name into dnamev from course where course_id = new.cours
e_id;
    ->    set new.course_name = dnamev;
    -> end;
    ->  //
ERROR 1235 (42000): This version of MySQL doesn't yet support 'multiple triggers
 with the same action time and event for one table'
mysql> DELIMITER ;
```

9.18.4 Ενοποίηση εναυσμάτων (trigger) που ενεργοποιούνται με την ίδια συνθήκη.

Δείτε τους πίνακες dept, student που ανήκουν σε βάση δεδομένων υποτρόφων φοιτητών/φοιτητριών του πανεπιστημίου και παρέχουν με κάποιο μισθό υπηρεσίες στο Πανεπιστήμιο.

Student

Student_id	Student _name	Student _surname	City	salary	Dept_id	Dname
1	CHRIS	DATE	LONDON	2000	1	COMPUTER SCIENCE
2	GIO	WIEDERHOLD	ATHENS	1500	1	COMPUTER SCIENCE
3	PETER	CHEN	ATHENS	3500	2	INFORMATICS
4	JEFF	ULLMAN	ATHENS	1700	1	COMPUTER SCIENCE
5	TED	CODD	ATHENS	2500	2	INFORMATICS

Dept

Dept_id	Dname
1	COMPUTER SCIENCE
2	INFORMATICS

Γράψτε trigger έτσι ώστε όταν εισάγουμε τα στοιχεία φοιτητή/φοιτήτριας να εκτελείται ο trigger, να «διαβάζει» τον πίνακα dept και να εισάγει αυτόματα τιμή στη στήλη dname του πίνακα του σπουδαστή. Επιπλέον, να «πειράζει» λίγο το μισθό.

```
Drop database training;
CREATE DATABASE training;
USE training;
CREATE TABLE dept(dept_id int, dname varchar(50));
CREATE TABLE student(student_id int(3), student_surname varchar(15),
student_name varchar(15), city varchar(15), salary decimal (8,2),  dept_id int,
dname varchar(15));
DELIMITER //
```

```
create trigger bi_student
before insert on student for each row
Begin
   declare name_length int;
   declare dnamev varchar(40);
   select dname into dnamev from dept where dept_id = new.dept_id;
   set new.dname = dnamev;
   set name_length = length(new.student_name);
   set new.salary = new.salary + (new.salary * name_length)/100;
end;
 //
DELIMITER ;
```

```
INSERT INTO dept VALUES (1, 'COMPUTER SCIENCE');
INSERT INTO dept VALUES (2, 'INFORMATICS');
Select * From dept;
INSERT INTO student
(student_id, student_surname, student_name, city, salary, dept_id) VALUES
(1, 'CHRIS', 'DATE', 'LONDON', 2000, 1), (2, 'GIO', 'WIEDERHOLD', 'ATHENS',
1500, 1),
(3, 'PETER', 'CHEN', 'ATHENS', 3500, 2), (4, 'JEFF', 'ULLMAN', 'ATHENS', 1700,
1),
(5,  'TED', 'CODD', 'ATHENS', 2500, 2);
SELECT * FROM student;
SELECT student_id, student_surname, student_name, dept_id, dname FROM student;
```

Προσοχή! Δεν επιτρέπετε να ενημερώσουμε NEW.row σε AFTER trigger. Ακολουθεί παράδειγμα.

```
DELIMITER //
CREATE TRIGGER my_trig
AFTER INSERT ON dept
FOR EACH ROW
BEGIN
   SET NEW.dname=UPPER(NEW.dname);
END;
//
DELIMITER ;
```

Δείτε το μήνυμα.

```
ERROR 1362 9HY000): Updating of NEW row is not allowed in after trigger.
```

Κεφάλαιο 10
Τεχνολογία Oracle PL/SQL. Μελέτη του ΣΔΒΔ της Oracle. Μελέτη περίπτωσης με χρήση τεχνολογίας Oracle PL/SQL και γλώσσας προγραμματισμού JAVA.

Σύνοψη

Στο Κεφάλαιο αυτό κεντρικός στόχος μας είναι η περιεκτική σκιαγράφηση κάποιων συνιστωσών του προϊόντος της Oracle και κυρίως η επισκόπηση της φιλοσοφίας κατασκευής εμπορικών, επιστημονικών κ.λπ. εφαρμογών που το συνοδεύει.

Σκιαγραφούνται το περιβάλλον (Συνιστώσα) ***SQL*PLUS,*** *δηλαδή το περιβάλλον χρησιμοποίησης της SQL, και κυρίως η γλώσσα (τεχνολογία) ORACLE PL/SQL.*

Στη συνέχεια εστιάζουμε σε υλοποίηση εφαρμογών με χρήση Java και Oracle.

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.

Προαπαιτούμενη γνώση
Προτείνεται η μελέτη των κεφαλαίων 3, 4, 5, 6 του παρόντος συγγράμματος

10.1 Εισαγωγή

Στο Κεφάλαιο αυτό κεντρικός στόχος μας είναι η περιεκτική σκιαγράφηση κάποιων συνιστωσών του προϊόντος της Oracle και κυρίως η επισκόπηση της φιλοσοφίας κατασκευής εμπορικών εφαρμογών που το συνοδεύει.

Κύρια χαρακτηριστικά του προϊόντος διαχείρισης βάσης δεδομένων της ORACLE είναι τα παρακάτω:

1) Συμβατότητα με όλα τα γνωστά πρότυπα, π.χ., ANSI SPARC, ANSI SQL
2) Συνοδεύεται από πλήθος εργαλείων ανάπτυξης εφαρμογών και υποστήριξης αποφάσεων αλλά και επεκτάσεις (συνιστώσες), όπως:
    - Γεννήτριες Εφαρμογών / Φορμών (Application Generator)
    - Γεννήτριες Αναφορών (Report generator)
    - εργαλεία CASE
    - πακέτα ανάκτησης κειμένου (text retrieval)
    - εργαλεία κατασκευής εφαρμογών βάσεων στο διαδίκτυο
    - υποστήριξη για βάσεις εικόνων (image data bases), βάσεις πολλαπλών μέσων (multimedia data bases) και βάσεις χώρου (spatial data bases)

3) Ενσωματώνει μία δυναμική γλώσσα προγραμματισμού που επεκτείνει τη γλώσσα SQL, τη γλώσσα PL/SQL
4) "Τρέχει" σε μηχανές κάθε μεγέθους, άρα εξασφαλίζει τη μεταφερσιμότητα των εφαρμογών.
5) Υποστηρίζει LAN και WAN.
6) Υποστηρίζει κατανεμημένες εφαρμογές στο υπολογιστικό νέφος (cloud)
7) Συνεργασία σε σχήμα πελάτη και εξυπηρετητή (Client / Server) με όλα τα γνωστά προϊόντα
8) Συμβατότητα με άλλα γνωστά προϊόντα , όπως το DB2 της IBM.
9) Οι απαιτήσεις του σε μνήμη εξαρτώνται από το μέγεθος της μηχανής όπου " τρέχει " η ORACLE.

Παραθέτουμε τις τεχνολογίες και τα εργαλεία της Oracle που μας ενδιαφέρουν στο σύγγραμμα αυτό:

- Το περιβάλλον (Συνιστώσα) **SQL*PLUS**, δηλαδή το περιβάλλον χρησιμοποίησης της γλώσσας SQL.
- Η γλώσσα PL/SQL.
- Ο Οπτικός Προγραμματισμός και η υλοποίηση εφαρμογών Client/Server με χρήση Java και Oracle.

10.1.1 Συνιστώσα SQL*PLUS της ORACLE. Περιβάλλον χρησιμοποίησης της γλώσσας SQL.

Η γλώσσα SQL της ORACLE προσφέρει:

- γλώσσα ορισμού δεδομένων, DDL-Data Definition Language, δηλαδή μια γλώσσα που χρησιμοποιείται για τη δημιουργία και την τροποποίηση της δομής των αντικειμένων της βάσης δεδομένων
- γλώσσα χειρισμού δεδομένων, DML-Data Manipulation Language, δηλαδή μια γλώσσα που χρησιμοποιείται για τη διαχείριση των δεδομένων της βάσης δεδομένωνς. Η διαχείριση περιλαμβάνει την εισαγωγή, τροποποίηση, ανάκτηση και τη διαγραφή δεδομένων από τους πίνακες της βάσης δεδομένων
- γλώσσα ελέγχου δεδομένων, DCL-Data Control Language, δηλαδή μια γλώσσα που χρησιμοποιείται για τον έλεγχο της πρόσβασης των χρηστών στα αντικείμενα της βάσης δεδομένων και τα περιεχόμενά τους.
- αποθήκευση δεδομένων με χρήση διαφόρων δομών δεδομένων
- βελτιστοποίηση αναζήτησης.
- χειρισμό λεξικού δεδομένων.

Για τη συμμετοχή σου σε μια συζήτηση πρέπει να γνωρίζεις:

- Τη γλώσσα των συνομιλητών σου
- Τους κανόνες του διαλόγου, πότε μιλάς, πως, για πόσο κλπ.

Κατά αναλογία, για να "συνομιλήσεις" με τη βάση δεδομένων στο προϊόν της ORACLE πρέπει να γνωρίζεις:

- Τη γλώσσα SQL
- Τους κανόνες του διαλόγου που επιβάλει η συνιστώσα SQL*PLUS.

Το ξεκαθάρισμα των εννοιών και των εντολών που θα αναφέρουμε θα γίνει με τη βοήθεια παραδειγμάτων.

Για το σκοπό αυτό θα χρησιμοποιηθεί η εξής απλοποιημένη βάση προσωπικού:

Πίνακας των υπαλλήλων,

```
EMP(EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO,PROJNO),
```

όπου EMPNO-κωδικός υπαλλήλου, ENAME-όνομα, JOB-θέση, MGR-ο επικεφαλής του υπαλλήλου, HIREDATE-ημερομηνία πρόσληψης, SAL-μισθός, COMM-προμήθεια, DEPTNO-κωδικός τμήματος στο οποίο ανήκει, PROJNO-κωδικός έργου στο οποίο εργάζεται

Πίνακας των τμημάτων,

```
DEPT(DEPTNO,DNAME,LOC), όπου DEPTNO- κωδικός τμήματος, DNAME-όνομα, LOC-έδρα.
```

Ο σχεδιασμός αυτός δεν είναι άρτιος επειδή λείπουν χρήσιμες στήλες από τους πίνακες και επιπλέον δεν καλύπτει περιπτώσεις στις οποίες ένας υπάλληλος μοιράζει το χρόνο του σε πολλά έργα. Είναι, όμως, πολύ απλός και επαρκής για την παρουσίαση των θεμάτων που θα μας απασχολήσουν.

10.1.2 Τι είναι SQL*PLUS.

Η συνιστώσα SQL*Plus είναι ένα διαλογικό περιβάλλον που εξασφαλίζει την άμεση (on-line) επικοινωνία του χρήστη και της βάσης δεδομένων. Η επικοινωνία γίνεται σε γλώσσα SQL εμπλουτισμένη με σειρά εντολών φορμαρίσματος αποτελεσμάτων κ.λπ.

10.1.3 Προσωρινή μνήμη εντολών (SQL Buffer)

Η προσωρινή μνήμη SQL (SQL buffer) είναι μια περιοχή εσωτερικής μνήμης. Χρησιμοποιείται όταν πληκτρολογούμε μια δήλωση SQL ή μπλοκ εντολών PL/SQL στη γραμμή εντολών (prompt) και διατηρεί ένα αντίγραφο της πιο πρόσφατα εισαγόμενης δήλωσης SQL (ή μπλοκ PL/SQL). Η συνιστώσα SQL*Plus παρέχει τη δυνατότητα επεξεργασίας και εκτέλεσης του περιεχομένου της προσωρινής μνήμης SQL (SQL buffer).

Οι δηλώσεις (statements) της SQL γράφονται και αποθηκεύονται σε SQL buffer.

- Ο buffer κρατά κάθε φορά μία και μόνο δήλωση SQL.
- Για τη πληκτρολόγηση της δήλωσης χρησιμοποιείται ένας απλός εκδότης (επεξεργαστής) κειμένου (editor) με εντολές:

  ```
  Append, Change, Delete, Input, List, Run.
  ```

Κάθε δήλωση SQL τελειώνει με το χαρακτήρα ";".

Μόλις ο χρήστης ολοκληρώσει την πληκτρολόγηση μίας δήλωσης (ή μπλοκ PL/SQL) που αποθηκεύεται στην προσωρινή μνήμη και ζητήσει εκτέλεση, τότε το περιεχόμενο του SQL buffer "παραδίδεται" στην ORACLE (στην πραγματικότητα είναι ένα αίτημα προς τον εξυπηρετητή της) για εκτέλεση.

Προσοχή! Ο χαρακτήρας ";" δεν αποθηκεύεται στο SQL buffer της δήλωσης. Απλά ειδοποιεί τη συνιστώσα SQL*PLUS ότι η δήλωση ολοκληρώθηκε.

Οι εντολές (commands) SQL*PLUS αντίθετα με τις δηλώσεις (statements) της γλώσσας SQL τελειώνουν σε μία γραμμή, μόλις πατήσουμε <Enter>. Βέβαια, για την ευκολία του χρήστη επιτρέπεται να τερματίζουμε και εντολές SQL*PLUS με το χαρακτήρα ";" .

Δηλαδή οι επόμενες εντολές είναι και οι δύο σωστές:

```
SQL> COLUMN DEPTNO HEADING DEPARTMENT;
SQL> COLUMN DEPTNO HEADING DEPARTMENT
```

Αν θέλουμε μια εντολή SQL*PLUS να γραφτεί σε περισσότερες γραμμές χρησιμοποιούμε τον χαρακτήρα "-":

```
  SQL> COLUMN DEPTNO -
 > HEADING DEPARTMENT
```

10.1.4 Τι μπορείτε να κάνετε με εντολές SQL*PLUS

Παραθέτουμε τις δυνατότητες με τις οποίες μας εφοδιάζει το προϊόν:

- Να βοηθηθείτε στη χρήση της γλώσσας SQL και της συνιστώσας SQL*PLUS καλώντας το on-line Help του συστήματος.
- Να εισάγετε, να διορθώσετε και να εκτελέσετε εντολές SQL
- Να σχηματίσετε πρόχειρες αναφορές (reports) που να περιλαμβάνουν ανακτηθέντα δεδομένα.
- Να εκτυπώσετε, να αποθηκεύσετε και να ανακτήσετε αναζητήσεις (queries) και τα αποτελέσματα.
- Να διαχειριστείτε μια βάση δεδομένων Oracle όταν ως χρήστης είστε συνδεδεμένος σε προστατευμένο περιβάλλον (π.χ. Linux), και οι πολιτικές ασφαλείας του εξυπηρετητή (Server) δεν επιτρέπουν χρήση (απομακρυσμένων) εναλλακτικών εργαλείων (του SQL*PLUS).

10.1.4.1 Εντολή HELP

```
SQL>  HELP; {δείχνει δηλώσεις SQL και εντολές SQL*PLUS}
SQL>  HELP  όνομα_εντολής;
```

10.1.5 Εισαγωγή-διόρθωση-εκτέλεση δηλώσεων SQL

Στον πίνακα 10.1 δίδονται εντολές της SQL*PLUS για την επεξεργασία του περιεχομένου του buffer εντολής.

*Πίνακας 10.1 Οι εντολές SQL*PLUS*

ΕΝΤΟΛΗ	ΣΥΝΤΟΜΟΓΡΑΦΙΑ	ΑΠΟΤΕΛΕΣΜΑ
APPEND κείμενο	A κείμενο	προσθέτει κείμενο
CHANGE /παλιό / νέο	C / παλιό / νέο	αντικαθιστά με νέο
CHANGE / κείμενο /	C / κείμενο /	διαγράφει κείμενο
CLEAR BUFFER	CL BUFF	καθαρίζει buffer
DEL		διαγράφει γραμμή
INPUT	I	εισάγει γραμμή (ές)
INPUT κείμενο	I κείμενο	εισάγει κείμενο
LIST	L	δείχνει γραμμές
LIST n	L n	δείχνει γραμμή n
LIST m n	L m n	δείχνει γραμμές m-n
RUN	R	εκτελεί εντολή

10.1.6 Παράδειγμα χρήσης εντολών SQL*PLUS

```
SQL> SELECT DPTNO, ENAME, SAL
  2  FROM EMP
  3  WHERE DPTNO = 10;
```

όπου στη τρίτη γραμμή από λάθος γράφτηκε DPTNO

αντί του σωστού DEPTNO.

```
SQL> LIST 3
SQL> CHANGE /DPTNO/DEPTNO/
SQL> RUN
```

Αν θέλουμε να προσθέσουμε μια γραμμή στη δήλωση:

```
SQL> INPUT
  4  ORDER BY SAL DESC
  5
SQL> RUN
```

10.1.7 Σχηματισμός απλών αναφορών Εντολές COLUMN, TTITLE, BTITLE

Θα παραθέσουμε ένα απλό παράδειγμα.

Παράδειγμα

Να σχηματίσετε αναφορά που να περιλαμβάνει τους εργαζόμενους που έχουν ως επικεφαλή τον Ανδρέου Ν. σύμφωνα με το υπόδειγμα της εικόνας 10.1.

DATE PAGE No

όνομα επικεφαλής

ΛΙΣΤΑ ΥΠΑΛΛΗΛΩΝ

EMPNO	ENAME	JOB	MGR	HIREDATE	SALARY	COMM	DEPARTMENT
9999	A.....A	A.A	99	9999-99-99	$9999.99	$9999.99	99

ΤΕΛΟΣ ΑΝΑΦΟΡΑΣ

Εικόνα 10.1 Αναφορά με λίστα υπαλλήλων για συγκεκριμένο επικεφαλή

Μετά την εκτύπωση της αναφοράς πρέπει να ακυρώσετε τις εντολές σχηματισμού της. Διαφορετικά, οι εντολές αυτές θα δεσμεύουν τη παρουσίαση των αποτελεσμάτων των επόμενων αναζητήσεών σας.

```
SQL> SELECT * FROM EMP WHERE MGR = 7698;
```

εμφανίζει τα αποτελέσματα της αναζήτησης

```
SQL> COLUMN DEPTNO HEADING DEPARTMENT;
```

αλλάζει την επικεφαλίδα DEPTNO σε DEPARTMENT κατά την παρουσίαση των αποτελεσμάτων

```
SQL> COLUMN SAL FORMAT $9999.99;
```

Αλλάζει την παρουσίαση του μισθού προσθέτοντας το σύμβολο του $ μπροστά από τον αριθμό. Ο αριθμός γράφεται με δύο δεκαδικά ψηφία.

```
SQL> COLUMN COMM FORMAT $9999.99;
SQL> TTITLE "Ανδρέου Ν.| ΛΙΣΤΑ ΥΠΑΛΛΗΛΩΝ";
```

Η εντολή προσθέτει ημερομηνία, τίτλο στην αρχή της αναφοράς και αριθμούς σελίδων. Ο χαρακτήρας "|" διατάζει την αλλαγή γραμμής κατά την παρουσίαση της αναφοράς.

```
SQL> BTITLE  "Τέλος Αναφοράς";
SQL> RUN;
SQL> COLUMN DEPTNO CLEAR;
SQL> COLUMN COMM CLEAR;
SQL> TTITLE OFF;
SQL> BTITLE OFF;
```

10.1.8 Εκτύπωση - αποθήκευση αποτελεσμάτων αναζήτησης. Εντολή SPOOL

```
   SQL> SPOOL όνομα_αρχείου.sql;
```

Αποθηκεύει τα αποτελέσματα των αναζητήσεων σε αρχείο μέχρι να δοθεί SPOOL OFF

```
   SQL> SPOOL OFF;
```

τερματίζει την αποθήκευση

```
   SQL> SPOOL OUT;
```

εκτυπώνει τα αποθηκευμένα αποτελέσματα.

10.1.9 Αποθήκευση, ανάκτηση και εκτέλεση εντολής. Εντολές SAVE, GET, START

Θα παραθέσουμε ένα απλό παράδειγμα.

Παράδειγμα

- Γράψε μια δήλωση SQL που να βρίσκει κωδικό υπαλλήλου, όνομα, μισθό, προμήθεια για τους πωλητές.
- Αποθήκευσε τη δήλωση στο αρχείο SALES.sql.
- Πάρε το αρχείο και άλλαξε επάγγελμα με τη μεταβλητή &1 και αποθήκευσε τις αλλαγές στο αρχείο JOB.
- Εκτέλεσε τη δήλωση για επικεφαλή.

- Χρησιμοποίησε ένα buffer διαφορετικό από το buffer εντολής για να αποθηκεύσεις τη δήλωση μαζί με τις εντολές σχηματισμού αναφοράς για τα αποτελέσματα.
- Αποθήκευσε το περιεχόμενο του buffer στο αρχείο PERFORMANCE.
- Εκτέλεσε το τελικό πρόγραμμα που αποθήκευσες στο αρχείο PERFORMANCE.

```
  SQL> INPUT
1  SELECT EMPNO, ENAME, SAL, COMM
2  FROM EMP
3  WHERE JOB='ΠΩΛΗΤΗΣ'
```

```
  SQL> SAVE SALES.sql;
  SQL> GET SALES;
```

δείχνει τις γραμμές της εντολής

```
  SQL> LIST 3;
  SQL> CHANGE/ΠΩΛΗΤΗΣ/&1/
  SQL> SAVE JOB;
  SQL> START JOB ΕΠΙΚΕΦΑΛΗΣ;
```

παρουσιάζει τα αποτελέσματα

```
  SQL> SET BUFFER C;
```

φτιάχνουμε ένα buffer διαφορετικό από το buffer εντολής

```
  SQL> INPUT
1  COLUMN ENAME HEADING NAME;
2  TTITLE 'ΑΠΟΔΟΣΗ ΠΡΟΣΩΠΙΚΟΥ';
3  SELECT EMPNO, ENAME, SAL, COMM
4  FROM EMP
5  WHERE JOB='&1'
```

```
  SQL> SAVE PERFORMANCE;
  SQL> START PERFORMANCE ΠΩΛΗΤΗΣ;
  SQL> START PERFORMANCE ΕΠΙΚΕΦΑΛΗΣ;
```

Άσκηση

Να εκτυπώσετε την ίδια κατάσταση (αναφορά) για περισσότερες ειδικότητες π.χ., για πωλητές, επικεφαλείς και αναλυτές.

10.1.10 Αρχεία εντολών

Με την εντολή SAVE, όπως είδαμε, μπορούμε να αποθηκεύσουμε, σε αρχείο εντολών, εντολές που μπορούν να ανακτηθούν και να εκτελεσθούν άμεσα με εντολή START.

Στο αρχείο εντολών μπορούμε να παρεμβάλουμε και γραμμές σχολίων με χρήση του προθέματος REM.

Για παράδειγμα, το αρχείο εντολών

```
REM
REM Δημιουργία βάσης προσωπικού
REM
CREATE  TABLE emp (empno  NUMBER  NOT  NULL,
   name  CHAR(30),
   address   CHAR (20));
```

δημιουργεί τη βάση προσωπικού emp.

Είναι δυνατόν να χρησιμοποιηθούν δηλώσεις INSERT, UPDATE κλπ. για να εμπλουτίσουν ένα αρχείο εντολών.

Είναι επιτρεπτό επίσης, να σχηματίσουμε αρχεία εντολών με εντολές της μορφής:

```
  STARTemp.sql
  STARTemp.dat
```

Τέλος, είναι επιτρεπτό να χρησιμοποιήσουμε τοπικές μεταβλητές σε εντολές ενός αρχείου εντολών. π.χ.. INSERT INTO emp VALUES (&1, '&2', '&3');

10.2 Γλώσσα ή ακριβέστερα τεχνολογία PL/SQL (Procedural Language / SQL)

10.2.1 Αρχή με παράδειγμα και λυμένες ασκήσεις

Το επόμενο πρόγραμμα ελέγχει και βρίσκει τα τριήμερα που περιλαμβάνουν την ημέρα της πρωτομαγιάς απο 1-5-2010 μέχρι και 1-5-2020. Υποτίθεται ότι αν η πρωτομαγιά είναι Σάββατο ή Κυριακή θα δοθεί ακόμη μια ημέρα αργίας.

Σημείωση

Η συνάρτηση TO_DATE μετατρέπει μία τιμή συμβολοσειράς (a string value) σε τιμή τύπου δεδομένων DATE χρησιμοποιώντας τον μορφότυπο (format) που καθορίζουμε στη δήλωση, π.χ., DD/MON/YY HH:MI:SS. Δηλαδή, με τη συνάρτηση μετατρέπουμε ημερομηνία και ώρα στον επιθυμητό μορφότυπο (format). Ακολουθεί παράδειγμα:

```
SELECT TO_DATE('2026-02-11', 'YYYY-MM-DD') FROM dual;
```

Ο αναγνώστης παραπέμπεται και στην ενότητα 3.3.8.2 «Πώς εισάγουμε στοιχεία ημερομηνίας και ώρας με χρήση της συνάρτησης TO_DATE στο προϊόν της Oracle» στο παρόν σύγγραμμα.

Υπόδειξη

Προτείνουμε να χρησιμοποιείτε πάντοτε στις πραγματικές εφαρμογές ένα συγκεκριμένο μορφότυπο, π.χ., DD/MM/YYYY.

```
DECLARE
  v_test_date DATE;
  v_day_of_week VARCHAR2(3);
  v_years_ahead INTEGER;
/* Η εκχώρηση αρχικών τιμών στις μεταβλητές γίνεται μέσα στο πρόγραμμα */
BEGIN
v_test_date := TO_DATE('01-May-2010', 'dd-mon-yyyy');
```

```
FOR v_years_ahead IN 1..11 LOOP
v_day_of_week :=  TO_CHAR(v_test_date,'Dy');
IF v_day_of_week IN ('Mon','Fri','Sat','Sun') THEN
   begin
 dbms_output.put_line(to_char(v_test_date,'dd-Mon-yyyy'));
 dbms_output.put_line('Είναι τριήμερο αργίας');
   end;
ELSE
   begin
 dbms_output.put_line(TO_CHAR(v_test_date,  'dd-Mon-yyyy'));
 dbms_output.put_line( ' Δεν είναι τριήμερο αργίας');
   end;
END IF;
v_test_date := ADD_MONTHS(v_test_date, 12);
END LOOP;
END;
/
```

Το επόμενο πρόγραμμα, επίσης ελέγχει και βρίσκει τα τριήμερα αργίας που περιλαμβάνουν την ημέρα της πρωτομαγιάς από 1-5-2010 μέχρι 1-5-2020.

Η διαφορά από το προηγούμενο πρόγραμμα βρίσκεται στα παρακάτω σημεία:

Οι μεταβλητές δεν ορίζονται μέσα στο πρόγραμμα. Προσέξτε ότι όταν ο τύπος δεδομένων (datatype) είναι DATE η αρχική τιμή ημερομηνίας δίνεται με συγκεκριμένο τρόπο. Θα μπορούσατε όμως να εκχωρήσετε τιμή και ως εξής:

```
   v_test_date DATE := TO_DATE('01-May-2010','DD-MON-YYYY');
```

Η μεταβλητή v_years_ahead παίρνει τιμές από 1 έως v_top_year

```
DECLARE
   v_test_date DATE := '01-May-90';   v_day_of_week VARCHAR2(3);
   v_years_ahead INTEGER ;v_top_year NUMBER :=11;
BEGIN
FOR v_years_ahead IN 1..v_top_year LOOP
v_day_of_week :=  TO_CHAR(v_test_date,'Dy');
IF v_day_of_week IN ('Mon','Fri','Sat','Sun') THEN
   begin
 dbms_output.put_line(to_char(v_test_date,'dd-Mon-yyyy'));
 dbms_output.put_line('Είναι τριήμερο αργίας');
   end;
ELSE
   begin
 dbms_output.put_line(TO_CHAR(v_test_date,  'dd-Mon-yyyy'));
 dbms_output.put_line( 'Δεν είναι τριήμερο αργίας');
   end;
END IF;
v_test_date := ADD_MONTHS(v_test_date, 12);
END LOOP;
END;
/
```

Το επόμενο πρόγραμμα επίσης ελέγχει και βρίσκει τα τριήμερα αργίας που περιλαμβάνουν την ημέρα της πρωτομαγιάς από 1-5-2010 και μέχρι 1-5-2020. Η διαφορά από τα προηγούμενα βρίσκεται στα παρακάτω σημεία:

Οι μεταβλητές δεν ορίζονται μέσα στο πρόγραμμα.

Γίνεται χρήση δομής while loop .

```
DECLARE
v_test_date DATE := TO_DATE('01-May-1990','DD-MON-YYYY');
v_day_of_week VARCHAR2(3);
v_years_ahead  INTEGER :=1;v_top_year INTEGER :=11;
BEGIN
WHILE v_years_ahead <= v_top_year LOOP
v_day_of_week :=  TO_CHAR(v_test_date,'Dy');
IF v_day_of_week IN ('Mon','Fri','Sat','Sun') THEN
   begin
 dbms_output.put_line(to_char(v_test_date,'dd-Mon-yyyy'));
 dbms_output.put_line('Είναι τριήμερο αργίας');
   end;
ELSE
   begin
 dbms_output.put_line(TO_CHAR(v_test_date,  'dd-Mon-yyyy'));
 dbms_output.put_line( 'Δεν είναι τριήμερο αργίας');
   end;
END IF;
v_test_date := ADD_MONTHS(v_test_date, 12);
v_years_ahead := v_years_ahead + 1;
END LOOP;
END;
/
```

Λυμένη άσκηση 1

Το παρακάτω πρόγραμμα βρίσκει τα δίσεκτα έτη από το 2010 μέχρι το 2020.

```
DECLARE
v_test_date DATE ;   v_last_day_of_February NUMBER(2);

v_years_ahead INTEGER :=1;   v_top_year INTEGER :=11;
BEGIN
v_test_date := TO_DATE('01-FEB-2010','dd-mon-yyyy');
   WHILE v_years_ahead <= v_top_year LOOP
   v_last_day_of_February := TO_NUMBER(TO_CHAR(LAST_DAY(v_test_date),'dd'));
   IF v_last_day_of_February = 29 THEN
  begin
dbms_output.put_line(to_char(v_test_date,'yyyy'));
dbms_output.put_line('Είναι δίσεκτο έτος');
  end;
   ELSE
  begin
dbms_output.put_line(TO_CHAR(v_test_date,'yyyy'));
dbms_output.put_line( 'Δεν είναι δίσεκτο έτος');
  end;
```

```
   END IF;
   v_test_date := ADD_MONTHS(v_test_date, 12);
   v_years_ahead := v_years_ahead + 1;
   END LOOP;
END;
/
```

```
DECLARE
v_test_date DATE ;   v_last_day_of_February NUMBER(2);

v_years_ahead INTEGER :=1;   v_top_year INTEGER :=11;
BEGIN
v_test_date := TO_DATE('01-FEB-2010','dd-mon-yyyy');
   WHILE v_years_ahead <= v_top_year LOOP
   v_last_day_of_February := TO_NUMBER(TO_CHAR(LAST_DAY(v_test_date),'dd'));
   IF v_last_day_of_February = 29 THEN
  begin
dbms_output.put_line(to_char(v_test_date,'yyyy'));
dbms_output.put_line('Είναι δίσεκτο έτος');
  end;
   ELSE
  begin
dbms_output.put_line(TO_CHAR(v_test_date,'yyyy'));
dbms_output.put_line( 'Δεν είναι δίσεκτο έτος');
  end;
   END IF;
   v_test_date := ADD_MONTHS(v_test_date, 12);
   v_years_ahead := v_years_ahead + 1;
   END LOOP;
END;
/
```

Λυμένη άσκηση 2

Το παρακάτω πρόγραμμα δείχνει τον τρόπο χρήσης της εντολής GOTO

```
DECLARE
  v_count INTEGER(2) := 1;
BEGIN
<<printhello>>
  dbms_output.put_line('Hello');
  v_count := v_count+1;
  IF v_count <= 5 THEN
 GOTO printhello;
  ELSE
 dbms_output.put_line('GOOD BUY!');
  END IF;
END;
/
```

Λυμένη άσκηση 3

Το παρακάτω πρόγραμμα διαβάζει και γράφει τη σημερινή ημερομηνία.

```
DECLARE
v_date INTEGER(2);
BEGIN
SELECT TO_NUMBER(TO_CHAR(sysdate,'dd'))
INTO v_date
FROM DUAL;
dbms_output.put_line('Σήμερα ο μήνας έχει :');
dbms_output.put_line(v_date);
end;
/
```

Λυμένη άσκηση 4

Το παρακάτω πρόγραμμα διαβάζει τη σημερινή ημερομηνία και γράφει σε ποιό δεκαήμερο βρισκόμαστε

```
DECLARE
v_date INTEGER(2);
BEGIN
SELECT TO_NUMBER(TO_CHAR(sysdate,'dd'))
INTO v_date
FROM DUAL;
dbms_output.put_line('Σήμερα ο μήνας έχει :');
dbms_output.put_line(v_date);
IF v_date <= 9 THEN
   begin
dbms_output.put_line('1ο δεκαήμερο');
   end;
ELSIF (v_date > 10 AND v_date <= 19) THEN
   begin
 dbms_output.put_line('2ο δεκαήμερο');
   end;
ELSE
   begin
 dbms_output.put_line('3ο δεκαήμερο');
   end;
END IF;
END;
/
```

10.2.2 Πως γράφουμε πρόγραμμα PL/SQL. Χειριστικές οδηγίες

Στο περιβάλλον της συνιστώσας SQL*PLUS εισάγουμε με τη βοήθεια εκδότη (editor) το πρόγραμμά μας.

```
SQL> EDIT filename.sql
```

Η εντολή αυτή καλεί τον προκαθορισμένο εκδότη (editor) που υπάρχει στο περιβάλλον του λειτουργικού συστήματος για να γράψουμε το πρόγραμμά μας.

Για παράδειγμα, σε περιβάλλον Windows καλούμε τον εκδότη wordpad, πληκτρολογούμε όλες τις εντολές του προγράμματος, σώζουμε και επιστρέφουμε στο περιβάλλον της SQL.

Σε περιβάλλον UNIX/LINUX χρησιμοποιούμε τον εκδότη vi(m) ή nano.

Γενικά, μπορούμε να εργαστούμε και σε οποιονδήποτε άλλο επεξεργαστή. Αρκεί να "σώσουμε" το πρόγραμμά μας σε αρχείο κειμένου με επέκταση .sql

```
SQL>
```

Επιστροφή στο περιβάλλον της SQL*PLUS.

```
SQL> SPOOL filename
```

Όλος ο διάλογός μας με το σύστημα θα σωθεί σε αρχείο για να μελετήσουμε τα αποτελέσματα.

```
SQL>SET SERVEROUTPUT ON
```

Ενεργοποίηση της δυνατότητας εκτύπωσης των μηνυμάτων που υπάρχουν στο πρόγραμμα. Τα μηνύματα εκτυπώνονται με εντολές της μορφής

```
DBMS_OUTPUT.PUT_LINE (' Μήνυμα ');

SQL> @filename.sql
```

Εκτέλεση του προγράμματος

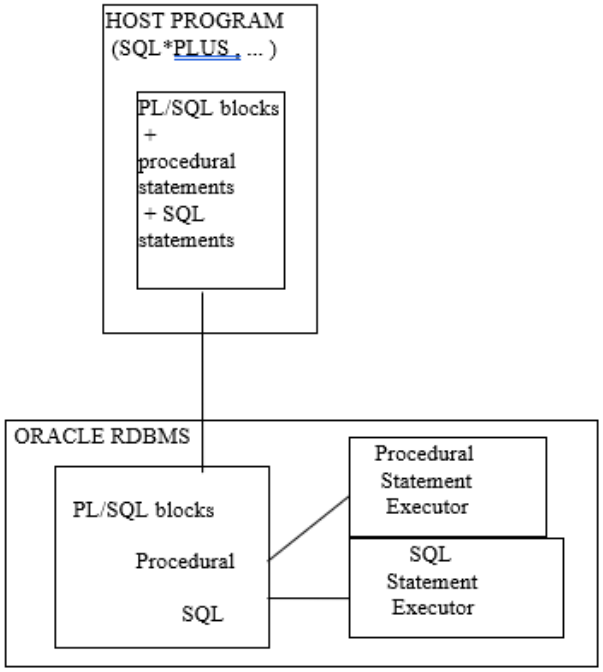
```
SQL> SPOOL OFF
```

Σταματάμε να κρατάμε τον διάλογο με το σύστημα. Επομένως, μπορούμε να τον σώσουμε, να τον δούμε, να τον τυπώσουμε κτλ.

10.2.3 Επισκόπηση της γλώσσας (τεχνολογία) PL/SQL

Η Oracle παρουσιάζει τη γλώσσα PL/SQL περισσότερο σαν μία τεχνολογία παρά σαν ένα προϊόν. Η τεχνολογία αυτή ενσωματώνεται σε πολλά προϊόντα της εταιρείας. Η εκτέλεσή της είναι ξεχωριστή από την εκτέλεση της γλώσσας SQL και ανατίθεται σε μία "μηχανή" PL/SQL (engine). Στην πραγματικότητα δε γράφουμε αυτοτελή προγράμματα στη γλώσσα αυτή αλλά προσθέτουμε κώδικα (υπό τη μορφή αποθηκευμένων στο σύστημα διαδικασιών ή συναρτήσεων που γράφουμε εμείς οι ίδιοι για να χρησιμοποιούνται από τις εφαρμογές μας και διαχείριση του εξυπηρετητή), εναυσμάτων (triggers) κτλ. Τα προϊόντα της Oracle συνοδεύονται από ειδικούς εκδότες (editors) για τη συγγραφή των PL/SQL προγραμμάτων.

Στην εικόνα 10.2 περιγράφεται η θέση της μηχανής στο ΣΔΒΔ και η "συνεργασία" με πρόγραμμα γραμμένο στο περιβάλλον SQL*PLUS .



Εικόνα 10.2 Η μηχανή PL/SQL στο ΣΔΒΔ της Oracle

10.2.4 Δομή της γλώσσας PL/SQL

Η γλώσσα PL/SQL είναι μία γλώσσα που οδηγεί στη συγγραφή προγραμμάτων δομημένων σε μπλοκ. Στην εικόνα 10.3 ξεκαθαρίζεται η έννοια αυτή.

Όπως είδαμε η γλώσσα PL/SQL είναι μία γλώσσα "δομημένη" σε μπλοκς (Block-Structured Language). Το πρώτο μπλοκ είναι το μπλοκ των δηλώσεων (declarations).

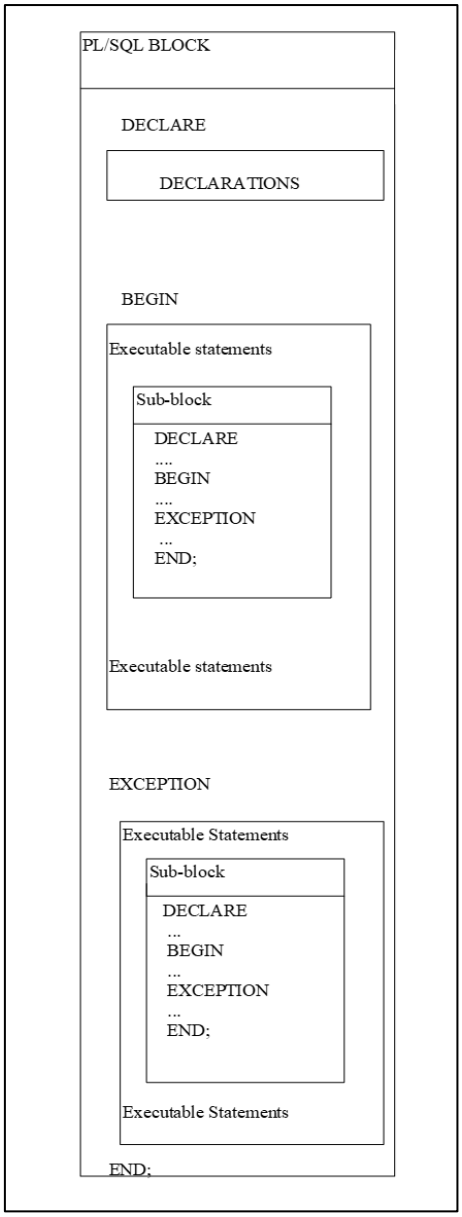
Ακολουθούν παραδείγματα :

```
Fax_Number VARCHAR2(10);
Current_Used_Value  NUMBER(6,2) := 100.00;
Max_Current_Used_Value  REAL := 9999.99;
State VARCHAR2(2) := 'CA';
```

Όπως μπορείτε να διαπιστώσετε εκτός από τη δήλωση του τύπου δεδομένων και του μήκους της μεταβλητής μπορείτε να εκχωρήσετε και αρχική τιμή.

Επίσης, πρέπει να σημειώσουμε ότι υπάρχουν περισσότεροι τύποι δεδομένων από ότι στη γλώσσα SQL και μάλιστα συνήθως τα προκαθορισμένα και τα μέγιστα μήκη είναι διαφορετικά, π.χ., ο τύπος δεδομένων VARCHAR2 στη γλώσσα SQL μας επιτρέπει να διαχειριστούμε (και να αποθηκεύσουμε) μέχρι 4000 bytes ενώ στη γλώσσα PL/SQL μέχρι 32767 bytes.

Προσοχή! Αν θέλουμε να αποθηκεύσουμε στη βάση μπορούμε μόνο μέχρι 4000 bytes



Εικόνα 10.3 Τα διάφορα μπλοκ της γλώσσας

Μπορούμε, επίσης, να δηλώσουμε μεταβλητές αντιγράφοντας τύπο δεδομένων και μήκος από στήλες ή και γραμμές της βάσης. Η ισχυρή αυτή δυνατότητα μας επιτρέπει να γράφουμε πιο φορητά προγράμματα PL/SQL.

variable-name table-name.column-name%TYPE;

όπου column-name είναι στήλη οριζόμενη στον πίνακα table-name.

Tech_Name Depot_Estimate.Technician%TYPE;

variable-name table-name%ROWTYPE;

Depot_Est_Row Depot_Estimate%ROWTYPE;

epot_Est_Row.Technician := ‘RICHARD’;

10.3 Ποιες είναι οι εκτελέσιμες εντολές μέσα σε μπλοκ

Μέσα σε μπλοκ μπορούμε να κάνουμε χρήση εντολών:

- IF-THEN-ELSIF
- LOOP EXIT
- WHILE LOOP
- FOR LOOP
- GOTO
- NULL (μιλάμε για εντολή)

Ακολουθεί η σύνταξη των εντολών και κάποια παραδείγματα.

10.3.1 Εντολή (Statement) IF-THEN-ELSIF

```
IF condition THEN
   statement; ...  statement;
[ELSIF condition THEN
statement; ...  statement;]
...
[ELSIF condition THEN
statement; ...  statement;]
[ELSE
statement; ...  statement;]
END IF;
```

10.3.2 Εντολή απλού βρόχου (LOOP) και εντολή EXIT

```
LOOP
  statement; ... statement;
END LOOP;
```

Χρήση εντολής EXIT για να τερματίσει ο βρόγχος

```
SQL> declare
  2  i  positive := 1;
  3  max_loops constant positive := 100;
  4  begin
  5  loop
  6i := i + 1;
  7exit when i > max_loops;
  8  end loop;
  9  end;
10  /
```

10.3.3 Εντολή WHILE-LOOP

```
WHILE condition LOOP
  statement; ... statement;
END LOOP;
```

Παρατίθεται παράδειγμα.

```
WHILE I < 100 LOOP
  I := I + 1;
```

```
  insert into temp_table (rec_number) values (I);
END LOOP;
```

10.3.4 Εντολή FOR-LOOP

```
FOR loop-variable IN [REVERSE] lower-bound..upper-bound LOOP
  statement;
   ...
  statement;
END LOOP;
```

Ακολουθεί παράδειγμα.

```
for i in 1..max_loops loop
  j := j + j;
  dbms_output.put_line('j: ' || to_char(j));
end loop;
```

10.3.5 Εντολή GOTO

```
<<my_label>>
```

```
SQL> declare
  2  i  positive := 1;
  3  max_loops constant positive := 100;
  4  begin
  5  i := 1;
  6  loop
  7i := i + 1;
  8if i > max_loops then
  9   goto more_processing;
 10end if;
 11  end loop;
 12  <<more_processing>>
 13  i := 1;
 14  end;
 15  /
```

10.3.6 Εντολή NULL

```
if (mod(i,10) = 0) then
  i := i + 1;
else
NULL;
end if;
```

10.4 Μη αναμενόμενα γεγονότα και μπλοκ exception

Υπάρχει ένας αριθμός συνθηκών ορισμένων εκ των προτέρων στο σύστημα, π.χ.,

NO_DATA_FOUND, TOO_MANY_ROWS,

PROGRAM_ERROR, INVALID_NUMBER, LOGON_DENIED

που αναπαριστούν κοινά γεγονότα και εκφράζουν κάποιες μη αναμενόμενες καταστάσεις που πρέπει να αντιμετωπιστούν προγραμματιστικά.

Υπάρχει βέβαια η δυνατότητα να οριστούν εξαιρέσεις και από το σχεδιαστή της εφαρμογής.

Παράδειγμα 1

```
BEGIN
EXCEPTION
   WHEN NO_DATA_FOUND THEN
  v_price_per_rental := 0
  INSERT INTO temp VALUES (v_model_name,
 v_price_per_rental);
  COMMIT;
   WHEN VALUE_ERROR OR ZERO_DIVIDE THEN
  ....
   WHEN OTHERS THEN
END;
```

Παράδειγμα 2

```
DECLARE
....
over_limit EXCEPTION /* user-defined exception */
BEGIN
....
   IF v_status NOT IN ('A','B') THEN
 RAISE INVALID_NUMBER;
   ELSE
 ....
   END IF;
   .....
EXCEPTION
 WHEN over_limit OR INVALID_NUMBER THEN
  .....
END;
```

Τέλος υπάρχει δυνατότητα ορισμού διαδικασιών και συναρτήσεων.

10.4.1 Δήλωση διαδικασίας (Procedure)

```
CREATE OR REPLACE PROCEDURE procedure-name
   [(argument1 ... [, argumentN) ] IS
[local-variable-declarations]
BEGIN
executable-section
[exception-section]
END [procedure-name];
where argument-name [IN | OUT] datatype [ {:= | DEFAULT} value]
```

10.4.2 Δήλωση συνάρτησης (function)

```
CREATE OR REPLACE FUNCTION function-name
[(argument1 ... [, argumentN) ]
RETURN function-datatype IS
[local-variable-declarations]
BEGIN
executable-section
[exception-section]
END [function-name];
```

10.5 Διαφορά τύπων δεδομένων varchar2, char. Χρήση συνάρτησης RTRIM.

Ακολουθεί ξεκαθάρισμα κάποιων βασικών τύπων δεδομένων με χρήση παραδειγμάτων.

Παράδειγμα 1

Έστω ότι δίνω την ίδια τιμή 'P.H. James' στις μεταβλητές ENAME_V (VARCHAR2(30)), ENAME_C (CHAR(30)) μέσα από πρόγραμμα PL/SQL. Πώς αποθηκεύεται η τιμή αυτή σε κάθε μια από τις δύο περιπτώσεις και πως μπορούμε να "εξισώσουμε" τις μεταβλητές ;

Το παρακάτω πρόγραμμα δεν χρησιμοποιεί τη συνάρτησης rtrim.

```
DECLARE
v_ename_varchar varchar2(30) := 'P.H.James';
v_ename_charchar(30) := 'P.H.James';

BEGIN
IF v_ename_varchar = v_ename_char THEN
 DBMS_OUTPUT.PUT_LINE ('Δεν έγινε χρήση της rtrim. Άραγε οι τιμές διαφέρουν;');
ELSE
  DBMS_OUTPUT.PUT_LINE ('Δεν έγινε χρήση της rtrim, άρα οι τιμές διαφέρουν');
END IF;
END;
/
```

Σαν αποτέλεσμα έχω το μήνυμα :

'Δεν έγινε χρήση της rtrim, άρα οι τιμές διαφέρουν'

Το παρακάτω πρόγραμμα χρησιμοποιεί τη συνάρτηση rtrim.

```
DECLARE
v_ename_varchar varchar2(30) := 'P.H.James';
v_ename_charchar(30) := 'P.H.James';

BEGIN
IF v_ename_varchar = rtrim(v_ename_char) THEN
 DBMS_OUTPUT.PUT_LINE ('Έγινε χρήση της rtrim, άρα οι τιμές ΔΕΝ διαφέρουν');
ELSE
  DBMS_OUTPUT.PUT_LINE ('Έγινε χρήση της rtrim. Άραγε οι τιμές διαφέρουν;');
END IF;
```

```
END;
/
```

Σαν αποτέλεσμα έχω το μήνυμα :

΄Εγινε χρήση της rtrim, άρα οι τιμές ΔΕΝ διαφέρουν΄

Παράδειγμα 2

Το παράδειγμα αυτό δείχνει ότι έχουμε κάποιες επιπλέον δυνατότητες όταν θέλουμε να εμφανίσουμε τα αποτελέσματα των προγραμμάτων μας. Παρατηρήστε ακόμη τη χρήση της μεταβλητής v_integer και τη χρήση του τελεστή || με τον οποίο παραθέτουμε μαζί συμβολοσειρές (String Concatenation).

```
declare
 v_xnum number := 1;
begin
for v_integer in 1..10 loop
  dbms_output.put_line ( v_xnum|| 'PRINT'||v_integer);
end loop;
end;
```

10.6 Χρήση δηλώσεων SQL μέσα σε προγράμματα PL/SQL

Θα παραθέσουμε κάποια παραδείγματα.

Παράδειγμα 1

Το παρακάτω πρόγραμμα επεξεργάζεται παραγγελίες για ρακέτες. Πραγματοποιεί την παραγγελία και ελαττώνει την ποσότητα του είδους μόνο όταν υπάρχει τουλάχιστον μια ρακέτα στην αποθήκη μας (στοκ).

```
/* Ορίζουμε τους πίνακες και εκχωρούμε αρχικές τιμές .*/
drop table inventory;
create table inventory ( prod_idnumber(5) not null,
product char(15), quantitynumber(5));
drop table purchase_record;
create table purchase_record(user char(10), mesg  char(45),   purch_date  date);
delete from inventory;
insert into inventory values(1234, 'ΡΑΚΕΤΑ ΤΕΝΝΙΣ', 3);
insert into inventory values(8159, 'ΜΠΑΣΤΟΥΝΙ ΓΚΟΛΦ', 4);
insert into inventory values(2741, 'ΜΠΑΛΑ ΠΟΔΟΣΦΑΙΡΟΥ', 2);
delete from purchase_record;
DECLARE
v_qty_on_hand  NUMBER(5);
v_item  := inventory.product%type := 'ΡΑΚΕΤΑ ΤΕΝΝΙΣ';
BEGIN


/* Η παρακάτω εντολή διαβάζει την ποσότητα του είδους από το αρχείο */
SELECT quantity INTO v_qty_on_hand FROM inventory
WHERE product  =  v_item
FOR UPDATE OF quantity;


/* ελέγχει και ενημερώνει το αρχείο αν υπάρχει στοκ διαφορετικά γράφει μήνυμα
```

```
στο δεύτερο αρχείο */
IF v_qty_on_hand > 0 THEN
UPDATE inventory SET quantity = quantity - 1
WHERE product =  v_item ;
INSERT INTO purchase_record
VALUES (USER,'Item purchased', SYSDATE);
DBMS_OUTPUT.PUT_LINE(sysdate||' Items purchased');
ELSE
INSERT INTO purchase_record
VALUES (USER,'Out of stock', SYSDATE);
DBMS_OUTPUT.PUT_LINE(SYSDATE||' Out of stock');
END IF;
COMMIT;
END;
```

Παράδειγμα 2

Χρέωση λογαριασμού, π.χ. του λογαριασμού 3, με κάποιο ποσό, π.χ. 500, μόνο όταν υπάρχει κάλυψη. Αν δεν υπάρχει κάλυψη γράφουμε το κατάλληλο μήνυμα σε αρχείο.

```
/*ορισμός βάσης δεδομένων */
drop table accounts;
create table accounts(account_id  number(4) not null,
   ename varchar2(20),bal number(11,2));
create unique index accounts_index on accounts (account_id);
drop table msg_file;
create table msg_file( num_col1   number(9,4),
   num_col2   number(9,4), char_col   char(55));

insert into accounts values (1,'date',1000.00);
insert into accounts values (2,'ulman',2000.00);
insert into accounts values (3,'codd',1500.00);
insert into accounts values (4,'date',6500.00);
insert into accounts values (5,'codd',500.00);

commit;

DECLARE
v_acct_balance  NUMBER(11,2);
v_acct   CONSTANT NUMBER(4) := 3;
v_debit_amt CONSTANT NUMBER(5,2) := 500.00;
BEGIN


/* Διαβάζουμε το υπόλοιπο του λογαριασμού */
SELECT bal INTO v_acct_balance FROM accounts
WHERE account_id = v_acct
FOR UPDATE OF bal;


/* και ελέγχουμε αν υπάρχουν τα απαραίτητα χρήματα για να ολοκληρώσουμε τη
συναλλαγή. Αν δεν υπάρχουν  βγάζουμε το κατάλληλο μήνυμα */
IF v_acct_balance >= v_debit_amt THEN
UPDATE accounts SET bal = bal - debit_amt
```

```
WHERE account_id = v_acct;
DBMS_OUTPUT.PUT_LINE('New Balance');
ELSE
INSERT INTO msg_file VALUES (v_acct, v_acct_balance , 'Insufficient funds' );


/* Καταχωρούμε σε αρχείο (v_acct, v_acct_balance) και το μήνυμα

  'Insufficient funds' */
   DBMS_OUTPUT.PUT_LINE('Insufficient funds');
END IF;
COMMIT;
END;
```

Παράδειγμα 3

Εύρεση του πρώτου υπαλλήλου που έχει μισθό πάνω από 4000 και βρίσκεται ψηλότερα στην ιεραρχία από τον υπάλληλο 7902.

```
/*Στο πρόγραμμα: παρατηρήστε τις εντολές δήλωσης μεταβλητών και ενσωματώστε (στο
πρόγραμμα)  όλα τα σχόλια με εντολές DBMS_OUTPUT.PUT_LINE */

DECLARE
  v_salary  emp.sal%TYPE;
  v_mgr_numemp.mgr%TYPE;
  v_last_name   emp.ename%TYPE;
  v_starting_empno  CONSTANT NUMBER(4) := 7902;
BEGIN
SELECT sal, mgr
INTO v_salary, v_mgr_num
FROM emp
WHERE empno = v_starting_empno;
WHILE v_salary < 4000 LOOP
SELECT sal, mgr, ename INTO v_salary, v_mgr_num, v_last_name
FROM emp
WHERE empno = v_mgr_num;
END LOOP;
DBMS_OUTPUT.PUT_LINE('Ename='||v_last_name || ' sal= ' || v_salary);
INSERT INTO temp VALUES (NULL, v_salary, v_last_name);
COMMIT;
END;
```

και το αποτέλεσμα του προγράμματος είναι

Ename=KING sal= 5000

10.7 Χρήση cursor στο προϊόν της Oracle.

Σε μια πρώτη προσέγγιση μπορούμε να περιγράψουμε την έννοια του cursor σαν ένα πίνακα στον οποίο κρατούνται τα αποτελέσματα μιας αναζήτησης. Υπάρχει ένας δείκτης τον οποίο μπορούμε να μετακινούμε με χρήση ειδικής εντολής στις γραμμές του πίνακα και να επεξεργαζόμαστε κάθε φορά μια γραμμή από τα αποτελέσματα.

10.7.1 Ορισμός CURSOR

Η σύνταξη της εντολής ορισμού είναι απλή.

```
DECLARE CURSOR cursor_name IS
  SELECT statement;
```

Ακολουθεί ένα παράδειγμα ορισμού που χρησιμοποιείται στη συνέχεια.

```
DECLARE CURSOR my_cursor IS
SELECT sal + NVL(comm, 0) wages, ename
FROM emp;
```

10.7.2 Εντολές διαχείρισης CURSOR (Open-Close-Fetch)

Παραθέτουμε την εντολή ενεργοποίησης (ανοίγματος) του CURSOR.

```
 OPEN cursor_name;
```

Δηλαδή στο παράδειγμα μας πληκτρολογούμε OPEN my_cursor;

Η εντολή αυτή καθορίζει (βρίσκει) τις γραμμές των αποτελεσμάτων της δήλωσης SELECT που περιλαμβάνεται στον ορισμό του cursor, τα αποτελέσματα "αποθηκεύονται" στον πίνακα των αποτελεσμάτων και ο δείκτης δείχνει στην πρώτη γραμμή των αποτελεσμάτων.

Ο cursor πρέπει υποχρεωτικά στο τέλος των εργασιών να κλείσει με την εντολή

```
CLOSE cursor_name;
```

Δηλαδή στο παράδειγμά μας πληκτρολογούμε CLOSE my_cursor;.

Όσο εργαζόμαστε με τις γραμμές του πίνακα των αποτελεσμάτων αυτές οι γραμμές δεν επηρεάζονται, δηλαδή δεν αλλάζουν από δηλώσεις INSERT, UPDATE , DELETE , SELECT.

Όταν εργαζόμαστε με τις γραμμές των αποτελεσμάτων κάνουμε τα εξής:

1) δηλώνουμε μια μεταβλητή
2) κάθε φορά που εκτελούμε την εντολή FETCH αποθηκεύουμε στη μεταβλητή μια γραμμή αποτελεσμάτων και
3) μετακινούμε τον δείκτη κατά μια γραμμή παρακάτω στα αποτελέσματα.

Οι δηλώσεις

```
DECLARE
CURSOR my_cursor IS SELECT sal + NVL(comm, 0) wages, ename
FROM emp;
my_rec  my_cursor%ROWTYPE;
```

που περιλαμβάνονται στο πρόγραμμα ορίζουν cursor με το όνομα my_cursor και μία εγγραφή με το όνομα my_rec που αποτελείται από τα πεδία wages και ename (δες και ορισμό του cursor).

Η εντολή FETCH,

```
FETCH  my_cursor INTO  my_rec;
```

«περνά» (μεταφέρει) την τρέχουσα γραμμή των αποτελεσμάτων που προέκυψαν από το άνοιγμα του my_cursor (όπου τρέχουσα γραμμή είναι η γραμμή των αποτελεσμάτων στην οποία δείχνει ο δείκτης) στην εγγραφή my_rec και επιπλέον αυξάνει τον δείκτη κατά ένα.

10.7.2.1 Χρήση βρόχων για την προσπέλαση όλων των γραμμών του CURSOR

Για να μπορέσω να χρησιμοποιήσω όλες τις γραμμές των αποτελεσμάτων πρέπει να κατασκευάσω ένα βρόχο (loop). Υπάρχουν χαρακτηριστικά (Attributes) συνδεδεμένα με την έννοια του cursor που μπορώ να χρησιμοποιώ στους βρόχους. Ακολουθούν παραδείγματα.

%NOTFOUND

Ακολουθεί παράδειγμα.

```
  LOOP
   FETCH my_cursor INTO my_rec;
   EXIT WHEN my_cursor%NOTFOUND ;
   other statements eg, IF … THEN… END IF;
  END LOOP;
```

%FOUND

Ακολουθεί παράδειγμα.

```
LOOP

FETCH my_cursor  INTO my_rec;
IF my_cursor%FOUND THEN
other statements
ELSE EXIT;
END IF;
END LOOP;
```

%ROWCOUNT

Ακολουθεί παράδειγμα.

```
LOOP
   FETCH my_cursor INTO my_rec;
   IF my_cursor%ROWCOUNT <= v_num_rec -1 THEN
/* where v_num_rec -1 is the number of the rows */
/* Στην ενότητα DECLARE δηλώνεται η μεταβλητή αυτή
   πχ.  v_num_rec   NUMBER(3)   */
   other statements
ELSE EXIT;
END IF;
END LOOP;
```

WHILE condition LOOP

Ακολουθεί παράδειγμα.

```
FETCH my_cursor INTO my_rec;
  other statements;
```

```
END LOOP;
```

WHILE my_cursor%ROWCOUNT < v_num_rec –1 LOOP

…

END LOOP;

Ακολουθεί παράδειγμα.

```
  v_counter := 1;
FOR  v_counter IN 1..v_num_rec LOOP
 FETCH my_cursor INTO my_rec;
 v_counter := v_counter + 1;
 …
END LOOP;
```

WHILE my_cursor%FOUND LOOP

```
FETCH …
END LOOP;
```

10.7.3 Παράδειγμα αναζήτησης (select) που χρησιμοποιεί Cursor

Μελετήστε το παρακάτω πρόγραμμα - αναζήτηση που βρίσκει όλους τους υπαλλήλους με μηνιαία αμοιβή μεγαλύτερη των 2000. Στη συνέχεια μπορείτε να γράψετε δικά σας προγράμματα παραλλαγές του παραδείγματος. Για παράδειγμα, ένα πρόγραμμα που βρίσκει όλους τους υπάλληλους που είναι πωλητές ή αναλυτές, προσλήφθηκαν μετά τις 7-6-2019 και έχουν αμοιβή πάνω από 2300.

```
Drop table temp;
Create table temp(msg  varchar2(20) , sal number(6), ename char(10));
DECLARE
CURSOR my_cursor IS SELECT sal + NVL(comm, 0) wages, ename
FROM emp;
my_rec  my_cursor%ROWTYPE;
BEGIN
OPEN my_cursor;
LOOP
FETCH my_cursor INTO my_rec;
EXIT WHEN my_cursor%NOTFOUND;
IF my_rec.wages > 2000 THEN
INSERT INTO temp VALUES ('loop exit when', my_rec.wages,
my_rec.ename);
DBMS_OUTPUT.PUT_LINE(my_rec.wages ||' '|| my_rec.ename);
END IF;
END LOOP;
CLOSE my_cursor;
END;
/

CLOSE my_cursor;
END;
/
```

Ακολουθούν παραλλαγές του ιδίου προγράμματος.

```
DECLARE
CURSOR my_cursor IS SELECT sal + NVL(comm, 0) wages, ename
FROM emp;
my_rec  my_cursor%ROWTYPE;
v_num_rec number(2);

BEGIN
select count(*) into v_num_rec from emp;
OPEN my_cursor;
LOOP
FETCH my_cursor INTO my_rec;
IF my_cursor%ROWCOUNT <=  v_num_rec-1 THEN
  IF my_rec.wages > 2000 THEN
 INSERT INTO temp VALUES ('loop ROW COUNT', my_rec.wages,
my_rec.ename);
 DBMS_OUTPUT.PUT_LINE(my_rec.wages ||' '|| my_rec.ename);
   END IF;
ELSE  -- LAST LINE OF THE CURSOR
EXIT;
END IF;
END LOOP;
CLOSE my_cursor;
END;
/
```

```
DECLARE
CURSOR my_cursor IS SELECT sal + NVL(comm, 0) wages, ename
FROM emp;
my_rec  my_cursor%ROWTYPE;
v_num_rec number(2):= 1;
BEGIN
SELECT COUNT(*) INTO v_num_rec FROM emp;
OPEN my_cursor;
WHILE my_cursor%ROWCOUNT < v_num_rec-1 LOOP
   FETCH  my_cursor INTO my_rec;
   IF my_rec.wages > 2000 THEN
INSERT INTO temp VALUES ('while rowcount', my_rec.wages,
my_rec.ename);
DBMS_OUTPUT.PUT_LINE(my_rec.wages ||' '|| my_rec.ename);
   END IF;
END LOOP;
CLOSE my_cursor;
END;
/
```

Ακολουθεί το αποτέλεσμα του προγράμματος και τελικά έχουμε το μήνυμα

PL/SQL procedure successfully completed.

10.8 Πως θα εργαστείτε για τη δημιουργία-ενημέρωση του ορισμού δικών σας συναρτήσεων (functions), διαδικασιών (procedures) και εναυσμάτων (triggers)

Στον εκδότη (editor) ορίστε το αντικείμενο σας αρχίζοντας με:

```
CREATE OR REPLACE, π.χ.,
CREATE OR REPLACE FUNCTION name … ;
```

όπου η εντολή CREATE δημιουργεί για πρώτη φορά το αντικείμενο

Αν το αντικείμενο δε δημιουργηθεί σωστά:

```
SQL> Show Errors;
```

Μόλις το αντικείμενο δημιουργηθεί παραμένει αποθηκευμένο (stored) και μπορούμε να το χρησιμοποιήσουμε μέσα από ανώνυμα μπλοκ, για τον ορισμό άλλων αντικειμένων κ.λπ., π.χ.,

```
 1) SELECT my_function   … ;
 2) declare …
  begin
 …
  new_record( …
  end;
```

Μπορούμε να διαγράψουμε τα αντικείμενα με drop:

```
drop procedure new_record;
drop  function my_function;
drop trigger my_trigger_on_Insert;
```

Ειδικά οι triggers ενεργοποιούνται όταν συμβεί κάποιο συγκεκριμένο γεγονός.

Παράδειγμα 1

```
create or replace procedure new_record(emp_no number,emp_sal number) IS
   temp_sal NUMBER(7,2);
begin
   temp_sal := emp_sal*0.1;
   insert into emp(empno,sal) values(emp_no,temp_sal);
end;
/
```

```
declare
new_employee_no emp.empno%type;
high_sal constant real:= 10000;
begin
   new_employee_no := 9999;
   new_record(new_employee_no, high_sal);
   dbms_output.put_line('create new employee');
end;
/
```

Παράδειγμα 2

```
create or replace function my_function(v_date DATE)
return number as
   v_num number;
begin
   v_num := FLOOR( MONTHS_BETWEEN(sysdate,v_date) / 12);
   RETURN v_num;
end;
/
```

Παράδειγμα 3

```
declare
  v_birthdate DATE :='28-jun-54';
  v_num number;
begin
   select my_function(v_birthdate) into v_num from dual;
   dbms_output.put_line('You are '||v_num||' years old');
end;
/
```

Παράδειγμα 4

```
create or replace trigger my_trigger_on_Insert
   AFTER INSERT ON emp
   FOR EACH ROW
begin
   dbms_output.put_line('new record');
end;
/
```

Παράδειγμα 5

```
create or replace trigger my_trigger_on_Delete
   AFTER DELETE ON emp
   FOR EACH ROW
begin
   dbms_output.put_line('record deleted');
end;
/
```

Δοκιμή

```
begin
  insert into emp(empno,sal) values(9999,1000);
  delete emp where empno=9999;
end;
/
```

10.9 Απλές ασκήσεις επισκόπησης PL/SQL

1) Υπολογίστε τριήμερα 25ης Μαρτίου από το 2010 έως και το 2020. Χρησιμοποιήστε δομές που περιγράφτηκαν κατά την επίλυση παρόμοιου παραδείγματος. Χρησιμοποιήστε, επίσης, διαδικασίες ή/και συναρτήσεις,.

2) Δημιουργήστε τον πίνακα Test_table (record_number, current_date). Γράψτε πρόγραμμα που εισάγει 100 γραμμές χρησιμοποιώντας δομές: for … loop , while…loop , loop…exit , loop…goto

3) Το ίδιο χρησιμοποιώντας διαφορετικές συνθήκες IF…THEN…ELSIF… που θα κατασκευάσετε με χρήση και της συνάρτησης MOD, πχ.,

   IF mod(v_int , 5) = 0 THEN rec_number := 5

   ELSIF mod(v_int , 7) = 0 THEN …

4) Δημιουργήστε πίνακα patient(patient_id,body_temp_deg_f). Βρείτε μέσον όρο θερμοκρασίας ασθενών

5) Γράψτε διαδικασία που εισάγει νέο ασθενή και τη θερμοκρασία του στον πίνακα PATIENT. Υποτίθεται ότι η θερμοκρασία δίδεται σε βαθμούς C και μετατρέπεται σε F. Ισχύει ο τύπος 5*(F-32)=9*C

6) Γράψτε διαδικασία που να διαγράφει ασθενή.

7) Γράψτε διαδικασία που να αλλάζει στοιχεία ασθενή.

8) Γράψτε συνάρτηση που να βρίσκει ασθενή που έχει τη μέγιστη θερμοκρασία .

10.10 Μελέτη περίπτωσης. Πληροφοριακό σύστημα διαχείρισης Dvd-Club

Οι φοιτητικοί σύλλογοι του Πανεπιστημίου αποφάσισαν να ιδρύσουν την ταινιοθήκη "Λα Στράντα" που θα λειτουργεί ως DVD club για τους φοιτητές-μέλη της και θα δανείζει DVD. Η βάση δεδομένων του πληροφοριακού συστήματος club καταχωρεί στοιχεία για τα μέλη του club και για τα DVD.

Για κάθε μέλος του club καταχωρούμε τα εξής χαρακτηριστικά (attributes): (μοναδικό) αριθμό μέλους, ονοματεπώνυμο, ημερομηνία εγγραφής, διεύθυνση, αριθμό τηλεφώνου και πλήθος των ταινιών που έχει δανειστεί.

Κάθε ταινία υπάρχει μόνο σε ένα αντίγραφο (DVD) και για να την περιγράψουμε χρησιμοποιούμε τα εξής χαρακτηριστικά: μοναδικός κωδικός, τίτλος, σκηνοθέτης, πρωταγωνιστές, κόστος, κατηγορία.

Κατηγορίες που χαρακτηρίζουν ένα DVD είναι οι εξής: κωμωδία, δράμα, αστυνομικό, περιπέτεια, τρόμου, σινεφίλ κλπ.

Κάθε dvd ανήκει σε μια ή περισσότερες κατηγορίες και έχει έναν ή περισσότερους πρωταγωνιστές. Για κάθε κατηγορία μας ενδιαφέρει και το πλήθος των ταινιών που της ανήκουν. Για κάθε ηθοποιό μας ενδιαφέρει και το πλήθος των oscar που έχει.

Κάθε μέλος μπορεί να δανεισθεί όσα dvd επιθυμεί (αν φυσικά είναι διαθέσιμα). Για κάθε δανεισμό dvd καταγράφονται η ημερομηνία δανεισμού και η ημερομηνία επιστροφής και μας ενδιαφέρει το ιστορικό δανεισμού ανά μέλος.

Για κάθε dvd μας ενδιαφέρει να ξέρουμε πόσες φορές έχει ενοικιαστεί, και το πλήθος των κατηγοριών στις οποίες ανήκει.

Για κάθε πρωταγωνιστή μας ενδιαφέρει να ξέρουμε αν είχε πρώτο ή δεύτερο ρόλο στις ταινίες που έχει παίξει.

Ζητούμενα

Να σχεδιαστεί βάση δεδομένων και να υλοποιηθεί πιλοτική εφαρμογή που θα διαχειρίζεται τα στοιχεία του dvd club.

Αναλυτικότερα:

Α) Να σχεδιαστεί η βάση δεδομένων του πληροφοριακού συστήματος με χρήση ΜΟΣ-Μοντέλου Οντοτήτων-Συσχετίσεων (Entity Relationship Diagram) λαμβάνοντας υπόψη τις παραπάνω απαιτήσεις ή και εμπλουτίζοντας με νέες αν το κρίνετε σκόπιμο.

Παράδειγμα επέκτασης

Πως θα αλλάξει ο σχεδιασμός αν αλλάξουν οι προδιαγραφές λειτουργίας και οι απαιτήσεις; Για παράδειγμα οι φοιτητικοί σύλλογοι που αποφάσισαν να ιδρύσουν το DVD club θέλουν να εξυπηρετεί όχι μόνο τους φοιτητές αλλά και όλα τα μέλη της κοινότητας του ιδρύματος. Στην περίπτωση αυτή θα υπάρχει διαφορετική χρέωση. Επιπλέον, το club θα δανείζει DVD σε μέλη του για προβολή είτε σε ειδικό χώρο μέσα στο ανώτατο ίδρυμα ή στο σπίτι. Και στη περίπτωση αυτή θα υπάρχει διαφορετική χρέωση. Τέλος, είναι επιθυμητό να χρησιμοποιήσετε και UML για τη σχεδίαση του μοντέλου σας.

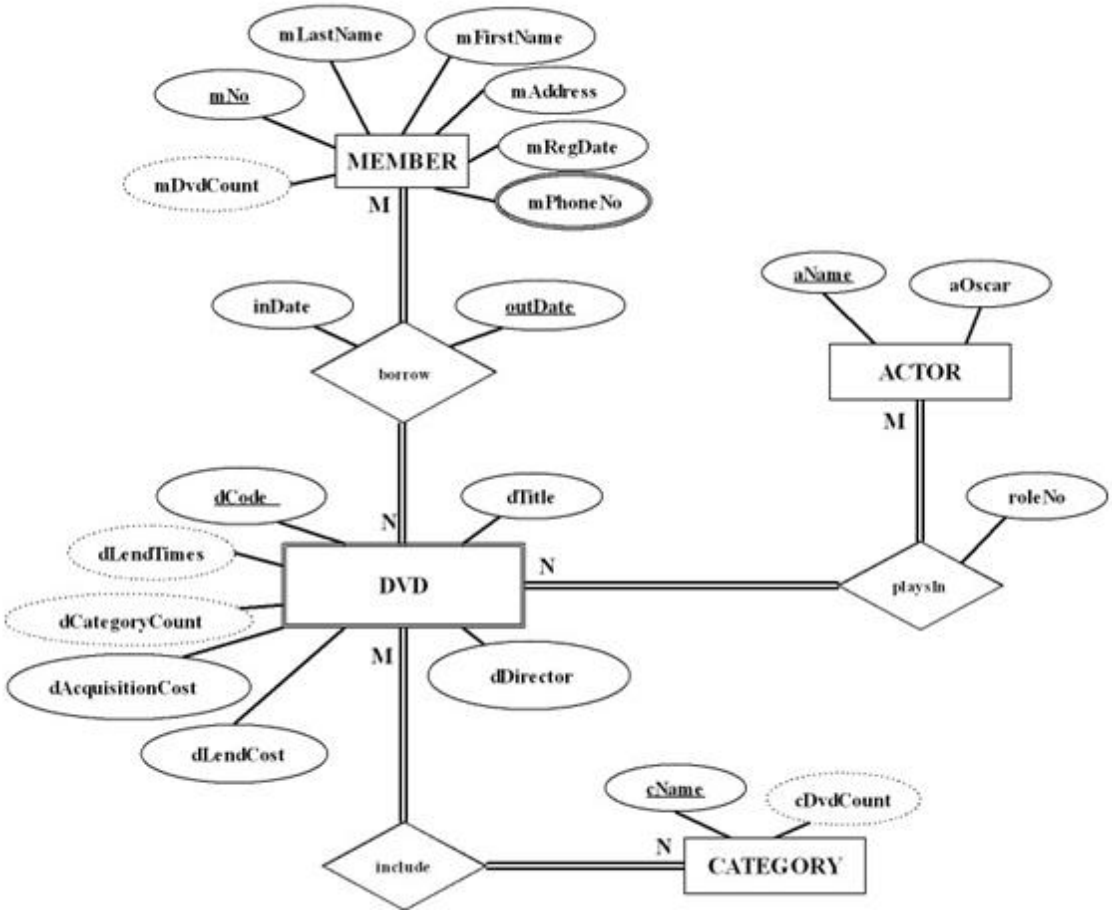
β). Να μεταγράψετε το μοντέλο που κατασκευάσατε σε σχήμα σχέσεων (πίνακες).

Υπόδειξη

Χρησιμοποιήστε αντιπροσωπευτικό δείγμα δεδομένων αφενός μεν για να "αφομοιώσετε" καλύτερα τη σχεδίαση και την υλοποίηση της βάσης δεδομένων στην οποία εμπλέκεστε, αφετέρου δε για να διαπιστώσετε λάθη που έγιναν κατά τη σχεδίαση της βάσης.

γ) Αν δεν διαπιστώσετε προβλήματα στη βάση που έχετε σχεδιάσει προχωρήστε στη δημιουργία των πινάκων της βάσης δεδομένων στο DBMS της Oracle χρησιμοποιώντας τις γνωστές δηλώσεις SQL (CREATE TABLE).

Στην Εικόνα 10.4 δίδεται ενδεικτική κατασκευή μοντέλου οντοτήτων συσχετίσεων (ΜΟΣ)



Εικόνα 10.4 Μοντέλο οντοτήτων συσχετίσεων για το Πληροφοριακό σύστημα διαχείρισης Dvd-Club

10.10.1 Σχήμα σχέσεων. Μετασχηματισμός ΜΟΣ σε σχεσιακή βάση και ενδεικτική υλοποίηση.

Παραθέτουμε κανόνες μετάφρασης Μοντέλου Οντοτήτων Συσχετίσεων σε Σχήμα Σχέσεων:

Κανόνας 1: Για κάθε (συνήθη) τύπο οντοτήτων Ε γράψε μια σχέση με χαρακτηριστικά τα (απλά) χαρακτηριστικά του τύπου οντοτήτων. Ως κλειδί βάλε κάποιο από τα κλειδιά του τύπου οντοτήτων.

Αν υπάρχει σύνθετο χαρακτηριστικό C:

- Αν δεν υπάρχει κλειδί του C που δεν περιλαμβάνει όλα τα χαρακτηριστικά του, στη σχέση που γράφουμε για τον Ε παραθέτουμε τα χαρακτηριστικά-συνιστώσες του C χωρίς το (συνολικό) όνομα.
- Αν υπάρχει κλειδί του C που δεν περιλαμβάνει όλα τα χαρακτηριστικά του, γράφουμε χωριστή σχέση για το σύνθετο χαρακτηριστικό και στη σχέση που προκύπτει από τον Ε παραθέτουμε το κλειδί του C που γίνεται και μέρος του κλειδιού.

Κανόνας 1α: Για κάθε πλειότιμο χαρακτηριστικό γράψε χωριστή σχέση με χαρακτηριστικά α. το κλειδί της οντότητας στην οποία βρίσκεται (το πλειότιμο χαρακτηριστικό) και β. το ίδιο το πλειότιμο χαρακτηριστικό. Κλειδί της νέας σχέσης είναι ο συνδυασμός των δυο αυτών χαρακτηριστικών. Επίσης το πλειότιμο χαρακτηριστικό δεν εμφανίζεται στη βασική σχέση που προκύπτει από τη βασική οντότητα. Αν το πλειότιμο χαρακτηριστικό είναι σύνθετο εφάρμοσε αυτά που λέει ο κανόνας 1 και βάλε ως μέρος του κλειδιού τα κατάλληλα χαρακτηριστικά.

Κανόνας 1β: Για κάθε τύπο ασθενών οντοτήτων W, με ιδιοκτήτη τύπο T γράψε καινούργια σχέση με χαρακτηριστικά τα (απλά) χαρακτηριστικά του W. Ως κλειδί (που είναι πάντα σύνθετο) βάλε το μερικό κλειδί του W και το κύριο κλειδί του T –το τελευταίο (κλειδί του T) είναι επίσης και ξένο κλειδί στη νέα σχέση.

Κανόνας 2: Για κάθε συσχέτιση 1 : N ή N : 1 που δεν έχει δικά της χαρακτηριστικά-κλειδιά μην γράψεις ξεχωριστή σχέση αλλά βάλε, ως ξένο κλειδί στη σχέση που αντιστοιχεί στο N, το κλειδί της σχέσης που αντιστοιχεί στο 1. Το ξένο κλειδί δεν γίνεται μέρος του κλειδιού της σχέσης. Στην ίδια σχέση βάλε και όλα τα χαρακτηριστικά που (μπορεί να) έχει η συσχέτιση.

Κανόνας 3: Για κάθε συσχέτιση 1 : 1 που δεν έχει δικά της χαρακτηριστικά-κλειδιά μην γράψεις ξεχωριστή σχέση αλλά βάλε, στη μία από τις συσχετιζόμενες οντότητες ως ξένο κλειδί το κλειδί της άλλης. Το ξένο κλειδί δεν γίνεται μέρος του κλειδιού της σχέσης. Στην ίδια σχέση βάλε και όλα τα χαρακτηριστικά που (μπορεί να) έχει η συσχέτιση. Αν έχουμε διαφορά στη συμμέτοχη προτιμούμε να κάνουμε την πρόσθεση στη σχέση που αντιστοιχεί στην οντότητα με την ολική συμμετοχή.

Κανόνας 4: Για κάθε συσχέτιση που είναι M : N ή έχει χαρακτηριστικά-κλειδιά γράψε μια σχέση που έχει ως ξένα κλειδιά τα κλειδιά όλων των συσχετιζόμενων τύπων οντοτήτων και όλα τα χαρακτηριστικά της συσχέτισης. Το κλειδί της θα αποτελείται από τα ξένα κλειδιά και τα τυχόν χαρακτηριστικά-κλειδιά της συσχέτισης. Αν κάποιος τύπος οντοτήτων συσχετίζεται :1 (και η συσχέτιση έχει χαρακτηριστικά κλειδιά) το ξένο κλειδί από αυτόν δεν γίνεται μέρος του κλειδιού της νέας σχέσης.

Κανόνας 5: Για κάθε συσχέτιση που συσχετίζει περισσότερους από δύο τύπους οντοτήτων γράψε μια σχέση που έχει ως ξένα κλειδιά τα κλειδιά όλων των συσχετιζόμενων τύπων οντοτήτων και όλα τα χαρακτηριστικά της συσχέτισης. Το κλειδί της θα αποτελείται από τα ξένα κλειδιά και τα τυχόν χαρακτηριστικά -κλειδιά της συσχέτισης. Αν κάποιος τύπος οντοτήτων συσχετίζεται :1 το ξένο κλειδί από αυτόν δεν γίνεται μέρος του κλειδιού της σχέσης. Προσοχή! Αν υπάρχει μόνο ένας τύπος οντοτήτων που συσχετίζεται :N και υπόλοιποι συσχετίζονται με :1 και επιπλέον η συσχέτιση δεν έχει χαρακτηριστικά κλειδιά τότε μη δημιουργήσεις νέα σχέση, αλλά βάλε ως ξένα κλειδιά στη σχέση που αντιστοιχεί στο N τα κλειδιά των σχέσεων που αντιστοιχούν στο 1, καθώς και όλα τα απλά χαρακτηριστικά της συσχέτισης.

Κανόνας 6: Αν τελικώς προκύψουν σχεσιακά σχήματα με το ίδιο κλειδί συγχωνεύονται σε ένα που έχει ως κλειδί το κοινό κλειδί και ως χαρακτηριστικά τα εκτός κλειδιού χαρακτηριστικά όλων των συγχωνευόμενων σχημάτων.

Ο τελευταίος είναι ένας συμπληρωματικός κανόνας που δεν έχει σχέση με τη διαδικασία απεικόνισης αλλά με αυτονόητες δουλειές που δεν πρέπει να ξεχάσουμε.

Επισήμανση: Τα χαρακτηριστικά που βρίσκονται σε διακεκομμένη έλλειψη (παράγωγα-derived ή υπολογιζόμενα-calculated) μπορούμε είτε να τα συμπεριλάβουμε είτε όχι στη σχέση που προκύπτει. Συνήθως πρέπει να γραφτεί κώδικας για να υπολογιστεί η ακριβής τιμή ενός υπολογιζόμενου πεδίου. Σε περίπτωση που κατά την υλοποίηση έχουμε τη δυνατότητα να χρησιμοποιήσουμε σκανδαλισμούς (triggers) βασιζόμενοι στην τεχνολογία PL/SQL τότε προτείνεται η διατήρηση των υπολογιζόμενων πεδίων στη σχέση.

Εφαρμογή των κανόνων μετάφρασης

Στη συνέχεια κάνοντας χρήση των παραπάνω κανόνων θα δούμε βήμα-βήμα πως κατασκευάζεται το σχήμα σχέσεων (πίνακες). Για κάθε σχέση θα σημειώσουμε το κύριο κλειδί (Primary Key: PK) καθώς και τα τυχόν ξένα κλειδιά (Foreign Key: FK).

Χρήση Κανόνα 1

Εφαρμόζοντας τον κανόνα 1 δημιουργούμε από τις οντότητες που βρίσκονται στο μοντέλο τις εξής σχέσεις(πίνακες)

Member (*mNo, mLastName, mFirstName, mAddress, mRegDate*) PK: *mNo*

Dvd (*dCode, dTitle, dDirector, dvdLendCost, dAcquisitionCost*) PK: *dCode*

Actor (*aName,aOscar*) PK: *aName*

Category (*cName*) PK: *cName*

Χρήση Κανόνα 1α

Εφαρμόζοντας τον κανόνα 1α για τα πλειότιμα χαρακτηριστικά δημιουργούμε μια επιπλέον σχέση:

Phone*(mNo,* mPhoneNo) PK: *mNo,mPhoneNo FK: mNo (Member)*

Χρήση Κανόνα 1β

Επειδή δεν υπάρχουν στο μοντέλο μας ασθενείς οντότητες δεν εφαρμόζουμε τον κανόνα 1β.

Χρήση Κανόνα 2

Επειδή δεν υπάρχουν συσχετίσεις οντοτήτων τύπου 1:N δεν εφαρμόζουμε τον κανόνα 2.

Χρήση Κανόνα 3

Επειδή δεν υπάρχουν συσχετίσεις οντοτήτων τύπου 1:1 δεν εφαρμόζουμε τον κανόνα 3.

Χρήση Κανόνα 4

Εφαρμόζουμε τον κανόνα 4 για τις συσχετίσεις τύπου M:N ή αυτές που έχουν χαρακτηριστικά κλειδιά και προκύπτουν οι εξής σχέσεις:

Borrows (mNo,dCode, outDate, inDate) PK: outDate, mNo, dCode

FK: *mNo* (*Member*) FK: dCode (Dvd)

PlaysIn (*aName, dCode, roleNo*) PK: *aName, dCode*

FK: *aName* (*Actor*) FK: *dCode* (*Dvd*)

Includes (*cName, dCode*) PK: cName, dCode

FK: cName (Category) FK: dCode (Dvd)

Χρήση Κανόνα 5

Επειδή δεν υπάρχουν συσχετίσεις που να συνδέουν 3 οντότητες και άνω δεν εφαρμόζουμε τον κανόνα 5.

Για τα υπολογιζόμενα ή παράγωγα πεδία, εφόσον έχουμε αποφασίσει πως θα τα συμπεριλάβουμε στις σχέσεις μας:

Προσθέτουμε το υπολογιζόμενο χαρακτηριστικό mDvdCount στη σχέση Member

Member (*mNo, mLastName, mFirstName, mAddress, mRegDate, mDvdCount*) PK: *mNo*

Προσθέτουμε τα υπολογιζόμενα χαρακτηριστικά *dCategoryCount, dLendTimes* στη σχέση Dvd

Dvd (*dCode, dTitle, dDirector, dvdLendCost, dAcquisitionCost, dCategoryCount, dLendTimes*) PK: *dCode*

Προσθέτουμε το υπολογιζόμενο χαρακτηριστικό mDvdCount στη σχέση Category

Category (*cName, cDvdCount*) PK: *cName*

Το σχήμα σχέσεων που προκύπτει εφαρμόζοντας τους κανόνες μεταγραφής φαίνεται ολοκληρωμένο στη συνέχεια. Παραθέτουμε τους πίνακες που αντιστοιχούν σε οντότητες και συσχετίσεις.

Σε οντότητες

Member (mNo, mLastName, mFirstName, mAddress, mRegDate, mPhoneNo, mDvdCount)

Primary Key: mNo

Phone(mNo, mPhoneNo)

Primary Key: mNo,mPhoneNo Foreign Key: mNo (Member)

Dvd (dCode, dTitle, dDirector, dvdLendCost,dAcquisitionCost, dCategoryCount, dLendTimes)

Primary Key: dCode

Actor (aName,aOscar)

Primary Key: aName

Category (cName,cDvdCount)

Primary Key: cName

Σε συσχετίσεις

Borrows (mNo,dCode, outDate, inDate)

Primary Key: outDate Foreign Key: mNo (Member) Foreign Key: dCode (Dvd)

PlaysIn (aName, dCode, roleNo)

Primary Key: aName, dCode Foreign Key: aName (Actor), dCode (Dvd)

Includes (cName, dCode)

Primary Key: cName, dCode Foreign Key: cName (Category), dCode (Dvd)

10.10.1.1 Ενδεικτική δημιουργία πινάκων

/* Δημιουργία πίνακα μελών */

```
create table Member ( mNo number, mLastName varchar2(20) not null,
mFirstName varchar2(15) not null, mAddress varchar2(30),
mRegDate DATE not null, mDvdCount number(4),
primary key (mNo),
check (mDvdCount >= 0) );
```

/* Δημιουργία πίνακα DVD */

```
create table Dvd ( dCode number, dTitle varchar2(30) not null, dDirector
varchar2(35),
      dvdLendCost number(4,2) not null, dAcquisitionCost number(5,2) not null,
```

```
      dCategoryCount number(2), dLendTimes number,
      primary key (dcode),
      check (dvdLendCost >= 0),
      check (dAcquisitionCost >= 0),
      check (dCategoryCount >= 0),
      check (dLendTimes >= 0) );
```

/* Δημιουργία πίνακα κατηγοριών */

```
create table Category ( cName varchar2(15), cDvdCount number,
      primary key (cName),
      check (cDvdCount > 0) );
```

/* Δημιουργία πίνακα πρωταγωνιστών */

```
create table Actor ( aName varchar2(30), aOscar number(2),
       primary key (aName),
       check (aOscar >= 0) );
```

/* Δημιουργία πίνακα τηλεφώνων μελών */

```
create table Phone( mNo number, mPhoneNo varchar2(10),
      primary key (mNo,mPhoneNo),
      foreign key (mNo) references Member(mNo) );
```

/* Δημιουργία πίνακα δανεισμών dvd απο μέλη */

```
create table Borrow ( mNo number,dCode number, outDate DATE, inDate DATE,
      primary key (outDate),
      foreign key (mNo) references Member(mNo),
      foreign key (dCode) references Dvd(dCode) );
```

/* Δημιουργία πίνακα συσχετισης ηθοποιών που παίζουν σε dvd */

```
create table PlaysIn (  aName varchar2(30) , dCode number, roleNo number(1),
      primary key (aName,dCode),
      foreign key (aName) references Actor(aName),
      foreign key (dCode) references Dvd(dCode) );
```

/* Δημιουργία πίνακα συσχετισης dvd - κατηγοριων */

```
create table Include ( cName varchar2(15), dCode number,
      primary key (cName,dCode),
      foreign key (cName) references Category(cName),
      foreign key (dCode) references Dvd(dCode) );
```

10.10.1.2 Ενδεικτική εισαγωγή στοιχείων

-- Εισαγωγή μελών

```
insert into Member ( mNo, mLastName, mFirstName, mAddress, mRegDate, mDvdCount )
values (100,'NIKOY','NIKOS','TZAVARA 12',TO_DATE('01/03/08','DD/MM/YY'),NULL);

insert into Member ( mNo, mLastName, mFirstName, mAddress, mRegDate, mDvdCount )
values (101,'PETROY','PETROS','RIGA 23',TO_DATE('15/02/09','DD/MM/YY'),NULL);
```

```
insert into Member ( mNo, mLastName, mFirstName, mAddress, mRegDate, mDvdCount )
values (102,'FWTIOY','FWTIS','KANARH 20',TO_DATE('17/03/09','DD/MM/YY'),NULL);
```

-- Εισαγωγή dvds

```
insert into Dvd (dCode, dTitle, dDirector, dvdLendCost, dAcquisitionCost,
dCategoryCount, dLendTimes )
values (10001, 'scarface', 'Brian De Palma', 1.5, 20, 2, 2);

insert into Dvd ( dCode, dTitle, dDirector, dvdLendCost,
dAcquisitionCost,  dCategoryCount, dLendTimes )
values (10002, 'Instinct', 'Jon Turtelaub', 1.5, 22, 2, 2);

insert into Dvd ( dCode, dTitle, dDirector, dvdLendCost,
dAcquisitionCost,  dCategoryCount, dLendTimes )
values (10003, 'Scent of woman', 'Martin Brest', 1.7, 21, 1, 1);
```

-- Εισαγωγή κατηγοριών

```
insert into Category ( cName, cDvdCount)
values ('crime',1);

insert into Category ( cName, cDvdCount)
values ('drama',3);

insert into Category ( cName, cDvdCount)
values ('thriller',1);
```

-- Εισαγωγή πρωταγωνιστών

```
insert into Actor ( aName,aOscar )
values ('Al Pacino',NULL);

insert into Actor ( aName,aOscar )
values ('Michelle Pfeiffer',NULL);

insert into Actor ( aName,aOscar )
values ('Anthony Hopkins',NULL);

insert into Actor ( aName,aOscar )
values ('Cuba Gooding',NULL);

insert into Actor ( aName,aOscar )
values ('Chris O ''Donnell',NULL);
```

-- Εισαγωγή τηλεφώνων

```
insert into Phone( mNo, mPhoneNo )
values(100,'6944100001');

insert into Phone( mNo, mPhoneNo )
values(100,'6944100002');

insert into Phone( mNo, mPhoneNo )
values(101,'6944100003');
```

```
insert into Phone( mNo, mPhoneNo )
values(102,'6944100004');
```

-- Εισαγωγή δανεισμών

```
insert into Borrow (mNo,dCode, outDate, inDate )
values(100,10002,TO_DATE('02/03/08 13:30:25','DD/MM/YY HH24:MI:SS'),
TO_DATE('03/03/08 20:22:19','DD/MM/YY HH24:MI:SS'));

insert into Borrow (mNo,dCode, outDate, inDate )
values(100,10001,TO_DATE('15/08/09 14:22:12','DD/MM/YY HH24:MI:SS'),
NULL);

insert into Borrow (mNo,dCode, outDate, inDate )
values(101,10002,TO_DATE('10/07/09 10:23:01','DD/MM/YY HH24:MI:SS'),
TO_DATE('12/07/09 21:25:04','DD/MM/YY HH24:MI:SS'));

insert into Borrow (mNo,dCode, outDate, inDate )
values(101,10001,TO_DATE('13/08/09 09:13:14','DD/MM/YY HH24:MI:SS'),
TO_DATE('14/08/09 11:59:08','DD/MM/YY HH24:MI:SS'));

insert into Borrow (mNo,dCode, outDate, inDate )
values(102,10003,TO_DATE('17/08/09 12:58:02','DD/MM/YY HH24:MI:SS'),
TO_DATE('25/08/09 20:00:00','DD/MM/YY HH24:MI:SS'));
```

-- Εισαγωγή εγγραφών συσχέτισης πρωταγωνιστών σε dvd

```
insert into PlaysIn ( aName, dCode, roleNo )
values('Al Pacino',10001,1);

insert into PlaysIn ( aName, dCode, roleNo )
values('Michelle Pfeiffer',10001,2);

insert into PlaysIn ( aName, dCode, roleNo )
values('Anthony Hopkins',10002,1);

insert into PlaysIn ( aName, dCode, roleNo )
values('Cuba Gooding',10002,1);

insert into PlaysIn ( aName, dCode, roleNo )
values('Al Pacino',10003,1);

insert into PlaysIn ( aName, dCode, roleNo )
values('Chris O ''Donnell',10003,1);
```

-- Εισαγωγή εγγραφών συσχέτισης dvd σε κατηγορίες

```
insert into Include ( cName, dCode )
values('crime',10001);

insert into Include ( cName, dCode )
values('drama',10001);

insert into Include ( cName, dCode )
```

```
values('drama',10002);

insert into Include ( cName, dCode )
values('thriller',10002);

insert into Include ( cName, dCode )
values('drama',10003);
```

10.11 Εισαγωγή στον οπτικό προγραμματισμό και στην υλοποίηση εφαρμογών client/server με χρήση java και oracle

Στη συνέχεια περιγράφεται η υλοποίηση τμήματος της πιλοτικής εφαρμογής διαχείρισης Dvd Club χρησιμοποιώντας τη γλώσσα προγραμματισμού Java, του πακέτου ανάπτυξης εφαρμογών Apache Netbeans IDE και του συστήματος διαχείρισης βάσεων δεδομένων(ΣΔΒΔ-DBMS) της εταιρείας ORACLE.

Αναλυτικότερα, τα εργαλεία που χρησιμοποιήθηκαν, και η σειρά με την οποία πρέπει να γίνει η εγκατάστασή τους είναι τα παρακάτω:

1) Java Development Kit (JDK) jdk 11.x ή νεότερη έκδοση (πρόσβαση στις 9/5/2022) (https://www.oracle.com/java/technologies/javase-jdk8-downloads.html)
2) Περιβάλλον ανάπτυξης εφαρμογής: Apache Netbeans IDE 12.x ή νεότερη έκδοση
3) Σύστημα Διαχείρισης Βάσεων Δεδομένων: Oracle XE 11 ή Oracle Enterprise Edition Version 11g ή νεότερη έκδοση

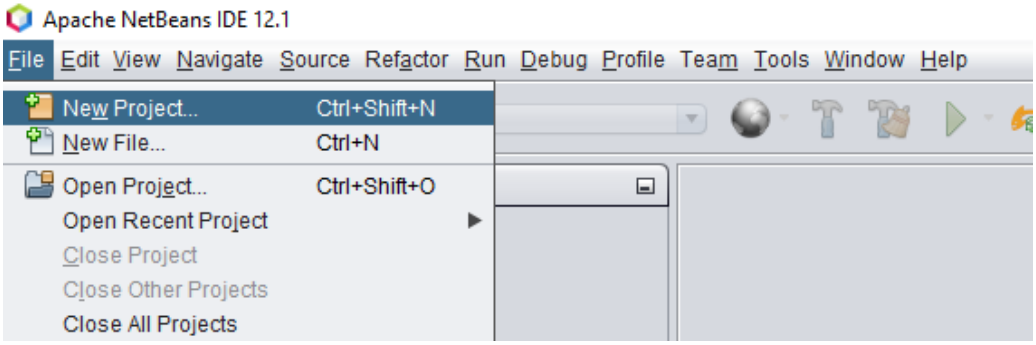
Σημείωση

Η εγκατάσταση της Oracle (3) μπορεί να γίνει και πριν από την εγκατάσταση του JDK. Η δημιουργία της εφαρμογής περιγράφεται αναλυτικά στις επόμενες ενότητες και ακολουθεί μία βήμα προς βήμα διαδικασία.

10.12 Δημιουργία του project της εφαρμογής, ενσωμάτωση της βιβλιοθήκης της ORACLE και γενικές αρχές σχεδίασης της διεπαφής (interface)

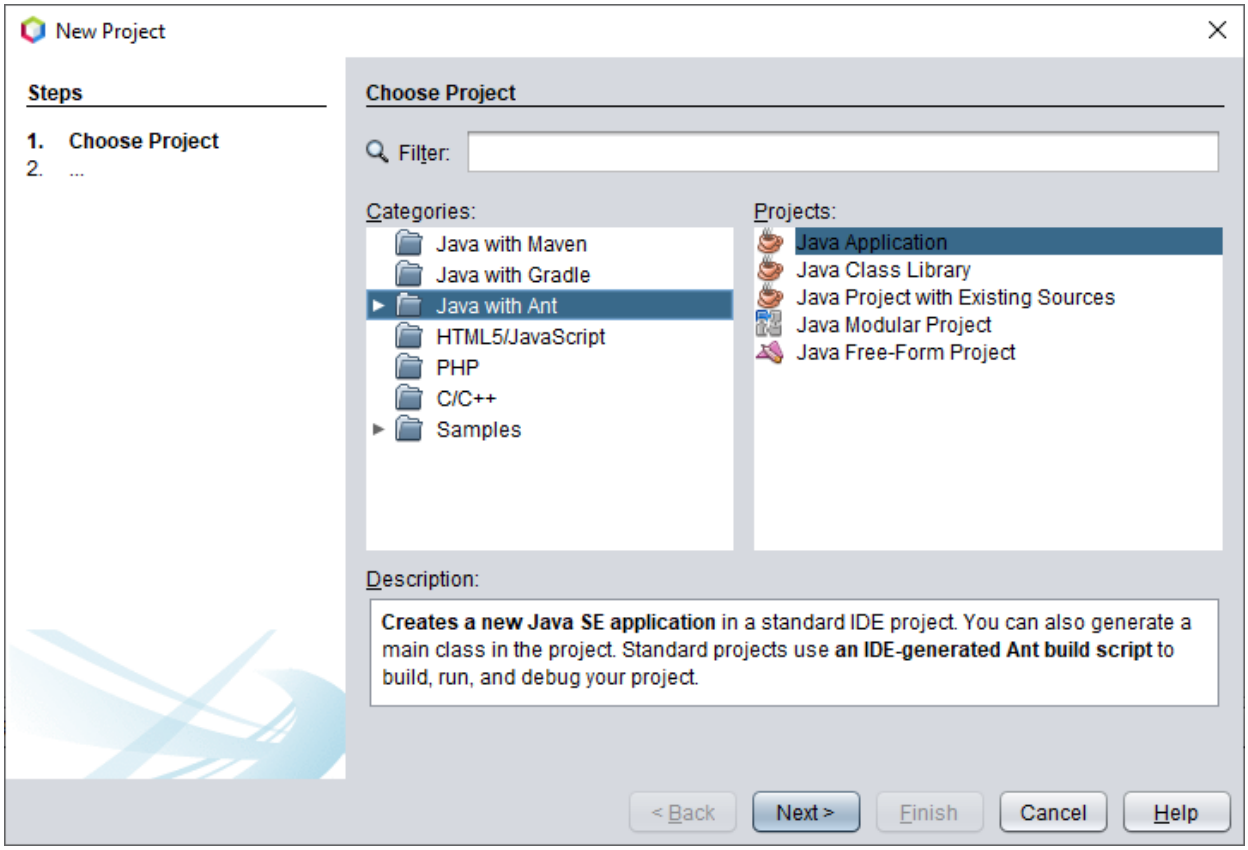
Βήμα 1ο – Κατασκευάζουμε νέο project

Οι εφαρμογές που αναπτύσσονται με το πακέτο ανάπτυξης Netbeans IDE οργανώνονται σε projects. Για να κατασκευάσουμε ένα project μας ανοίγουμε το Netbeans IDE και επιλέγουμε File→New Project, όπως φαίνεται στην εικόνα 10.5.



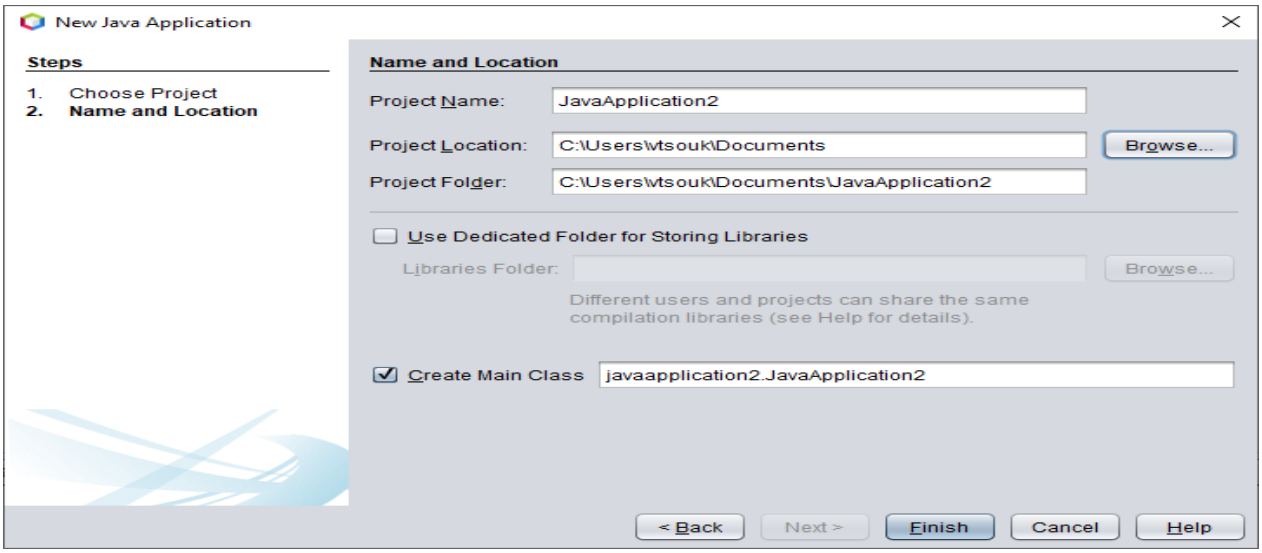
Εικόνα 10.5 Δημιουργία νέου project

Στη συνέχεια εμφανίζεται το παράθυρο της εικόνας 10.6, επιλέγουμε κατηγορία Java with Ant και τύπο project 'Java Application' και 'Next'.



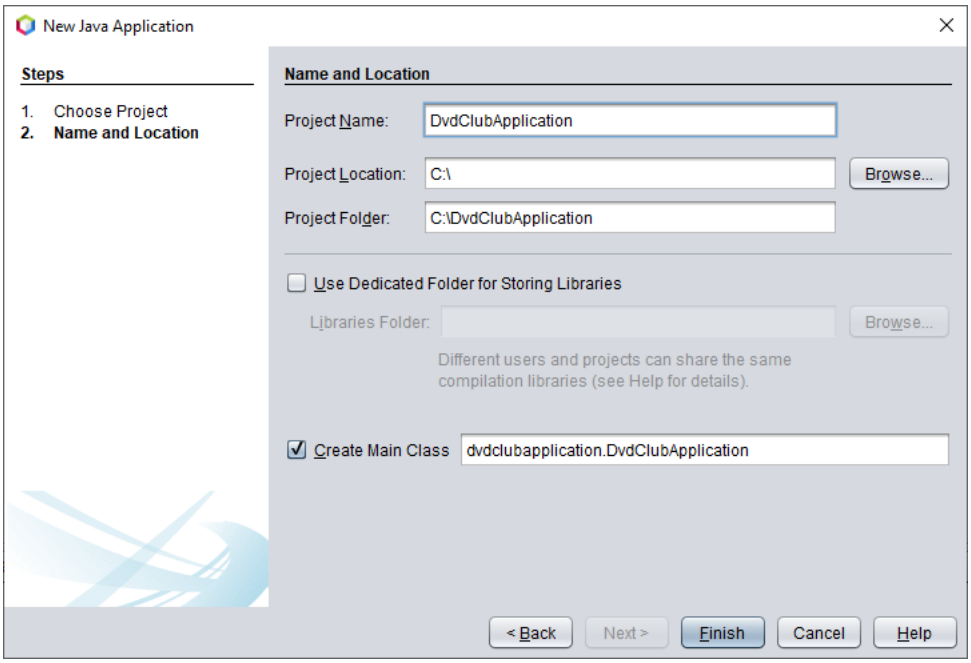
Εικόνα 10.6 Επιλογή Java Application

Εμφανίζεται το παράθυρο της εικόνας 10.7 για να δώσουμε τα στοιχεία του project μας, όπως όνομα και τοποθεσία αποθήκευσης στο σκληρό δίσκο:



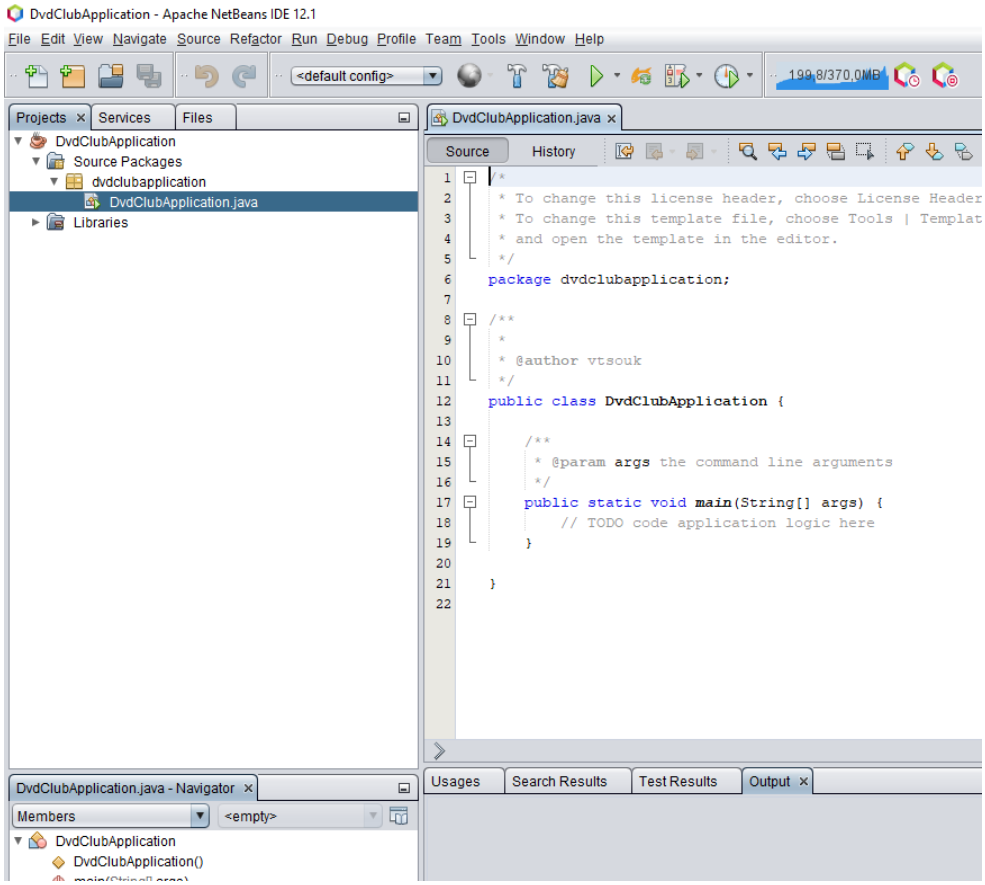
Εικόνα 10.7 Στοιχεία του project

Δίνουμε στο πλαίσιο κειμένου 'Project Name: DvdClubApplication', ενώ χρησιμοποιώντας το κουμπί Browse ρυθμίζουμε την τοποθεσία αποθήκευσης του project, 'Project Location: C:\' (βλέπε εικόνα 10.8).



Εικόνα 10.8 Όνομα και θέση αποθήκευσης του project

Στην ετικέτα 'Project Folder' φαίνεται η διαδρομή αποθήκευσης του project μας. Μόλις επιλέξουμε 'Finish' θα δημιουργηθεί το project μας στο σκληρό δίσκο στη διαδρομή 'C:\DvdClubApplication'. Όλα τα αρχεία που αφορούν την ανάπτυξη της εφαρμογής μας βρίσκονται μέσα σε αυτό το φάκελο. Μετά την επιλογή του Finish εμφανίζεται η εικόνα 10.9.



Εικόνα 10.9 Φάκελος project

Όπως βλέπουμε στην εικόνα 10.9, στο tab Projects, έχει δημιουργηθεί ένα project με όνομα DvdClubApplication που αποτελείται από ένα σύνολο φακέλων και αρχεία. Ο πηγαίος κώδικας αποθηκεύεται στον υποφάκελο 'Source Packages'. Για καλύτερη οργάνωση του πηγαίου κώδικα της εφαρμογής η java οργανώνει τα αρχεία του πηγαίου κώδικα σε βιβλιοθήκες –τα λεγόμενα packages-. Στην προκειμένη περίπτωση έχει κατασκευαστεί το package 'dvdclubapplication' που περιέχει το αρχείο 'DvdClubApplication.java'. Το αρχείο αυτό είναι το σημείο εκκίνησης της εφαρμογής μας. Στα δεξιά φαίνεται το περιεχόμενο του αρχείου DvdClubApplication.java, που αποτελείται από μια κλάση DvdClubApplication που περιέχει μια μέθοδο public static με όνομα main. Μόλις τρέξουμε την εφαρμογή μας η συνάρτηση main είναι η πρώτη που εκτελείται

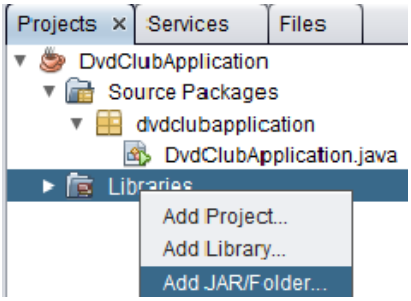
Σημείωση

Στην κορυφή του αρχείου DvdClubApplication.java υπάρχει η εντολή-δήλωση 'package dvdclubapplication'. Η εντολή αυτή δηλώνει πως το αρχείο DvdClubApplication.java ανήκει στη βιβλιοθήκη dvdclubapplication. Κάθε αρχείο της γλώσσας java πρέπει να έχει μια τέτοια εντολή ώστε να δηλώνει σε ποια βιβλιοθήκη ανήκει.

Βήμα 2ο – Ενσωμάτωση βιβλιοθήκης της oracle στο project

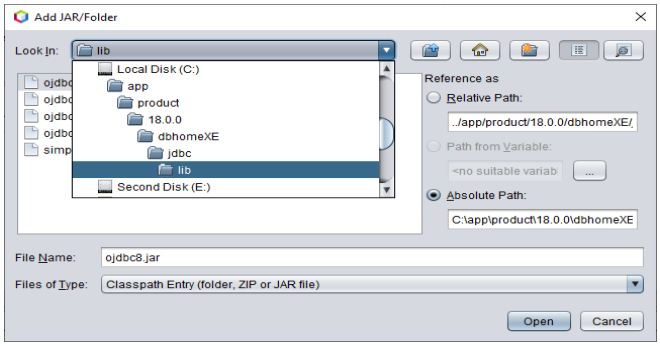
Πριν ξεκινήσουμε την ανάπτυξη της εφαρμογής και επειδή η εφαρμογή μας θα συνδεθεί με το ΣΔΒΔ της oracle θα πρέπει να ενσωματώσουμε στο project μας μια βιβλιοθήκη της. Η βιβλιοθήκη αυτή της oracle (που ανάλογα με την έκδοση έχει διαφορετικό αριθμό X) έχει όνομα ojdbcX.jar. Αν αυτή η βιβλιοθήκη δεν ενσωματωθεί τότε το project θα παράγει συντακτικά σφάλματα κατά τη μεταγλώττιση. Η βιβλιοθήκη **'ojdbcX.jar'** αποτελείται από ένα σύνολο packages που το καθένα περιέχει κλάσεις γραμμένες σε γλώσσα java που εξομοιώνουν την επικοινωνία με το ΣΔΒΔ της oracle.

Για να ενσωματώσουμε τη βιβλιοθήκη στο project μας κάνουμε δεξί κλικ στον υποφάκελο 'Libraries' του DvdClubApplication project και επιλέγουμε 'Add JAR/Folder', όπως φαίνεται στην εικόνα 10.10.



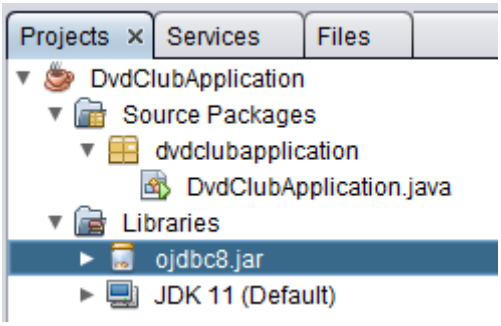
Εικόνα 10.10 Προσθήκη JAR/Folder

Επιλέγουμε το αρχείο **'ojdbc8.jar'** που βρίσκεται στον υποφάκελο 'C:\app\product\18.0.0\dbhomeXE\jdbc\lib' και επιλέγουμε 'Open', όπως φαίνεται στην εικόνα 10.11.



Εικόνα 10.11 Επιλογή του αρχείου 'ojdbc8.jar'

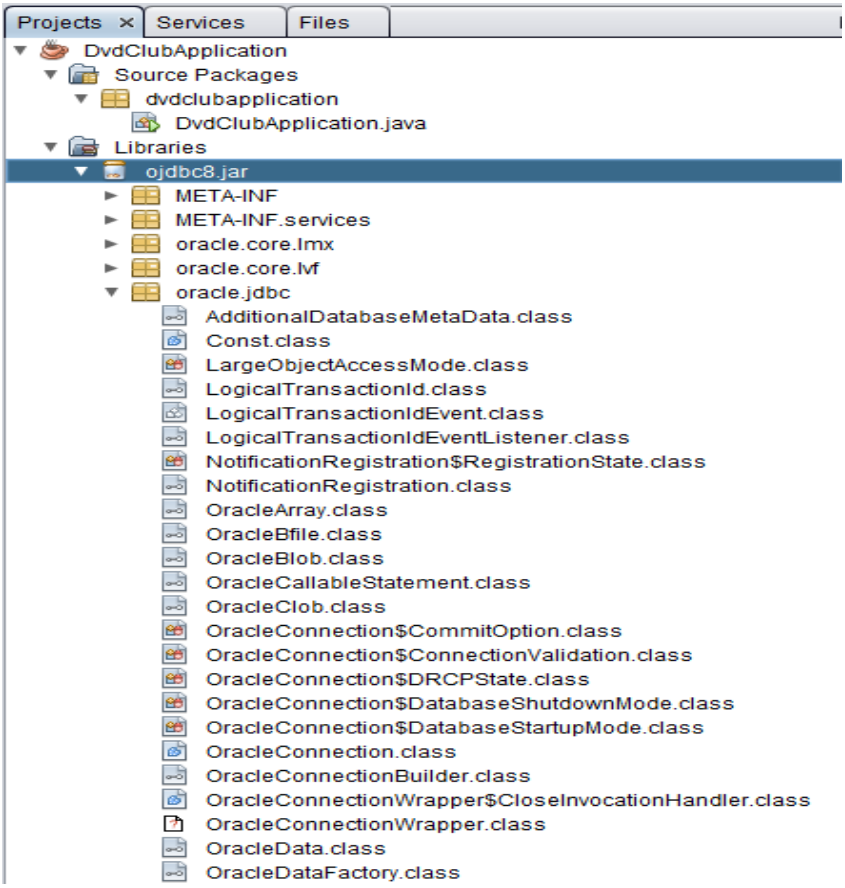
Η βιβλιοθήκη έχει πλέον ενσωματωθεί στο project μας και το αρχείο **'ojdbc8.jar'** θα πρέπει να φαίνεται στον υποφάκελο 'Libraries', όπως φαίνεται στην εικόνα 10.12.



Εικόνα 10.12 Εμφάνιση του αρχείου 'ojdbc8.jar' στον υποφάκελο 'Libraries'

Σημείωση

Όλες οι εξωτερικές βιβλιοθήκες που ενσωματώνονται σε κάποιο project φαίνονται στον υποφάκελο 'Libraries'. Αν επιλέξουμε το ▶ για να δούμε τα περιεχόμενα της βιβλιοθήκης διαπιστώνουμε πως αποτελείται από ένα σύνολο πακέτων καθένα από τα οποία έχει τις δικές του κλάσεις. Δείτε την εικόνα 10.13.



Εικόνα 10.13 Εμφάνιση κλάσεων του πακέτου (package) oracle.jdbc

Σημείωση

Για να χρησιμοποιήσουμε σε ένα πηγαίο αρχείο .java τις κλάσεις ενός package δίνουμε την εντολή import στην αρχή του πηγαίου αρχείου. Την ακριβή σύνταξη της εντολής θα τη δούμε αργότερα σε σχετικό παράδειγμα.

Τώρα που έχουμε ενσωματώσει τη βιβλιοθήκη διασύνδεσης με το ΣΔΒΔ της oracle μπορούμε να προχωρήσουμε στην ανάπτυξη της εφαρμογής μας. Η ανάπτυξη της εφαρμογής περιλαμβάνει το σχεδιασμό του interface (οθονών), τον εμπλουτισμό κλάσεων που συνοδεύουν το interface και την ενδεχόμενη συγγραφή ξεχωριστών βοηθητικών κλάσεων όταν πρέπει να κατασκευάσουμε πιο σύνθετες εφαρμογές.

Βήμα 3ο Σχεδιασμός περιβάλλοντος διεπαφής (interface). Γενικές οδηγίες

Το interface μιας εφαρμογής σε γενικές γραμμές αποτελείται από ένα σύνολο παραθύρων ή φορμών. Η φιλοσοφία είναι η εξής: κάποιο από αυτά τα παράθυρα κατέχει κεντρική θέση στο interface γιατί εμφανίζεται πρώτο, έχει οργανωμένο μέσα του τις πολύ βασικές λειτουργίες της εφαρμογής μας, ενώ συντονίζει πότε θα ανοίξουν τα υπόλοιπα βασικά παράθυρα. Καθένα από τα υπόλοιπα βασικά παράθυρα εξομοιώνει μια πολύ βασική λειτουργία της εφαρμογής. Επειδή κάθε βασική λειτουργία συνήθως περιέχει και επιπλέον υπολειτουργίες ένα βασικό παράθυρο μπορεί με τη σειρά του να συντονίσει την εμφάνιση επιπλέον παραθύρων που αντιστοιχούν στις υπολειτουργίες. κ.ο.κ.

Οποιοδήποτε παράθυρο σχεδιάζεται κάνοντας χρήση κάποιων αντικειμένων (components), όπως είναι το μενού, πλαίσια κειμένου, λίστες, μπάρες ολίσθησης, πλέγματα, κουμπιά, μπάρα εργαλείων κ.ο.κ. Κάθε τέτοιο αντικείμενο περιέχει αρκετά χαρακτηριστικά ή ιδιότητες (properties) (που εξαρτώνται από το είδος του αντικειμένου), ενώ μπορεί να προκαλέσει συμβάντα/γεγονότα (events) τα οποία μπορούμε να συνδέσουμε με κάποιες συναρτήσεις-μέλη ή μεθόδους (event handlers/actions). Π.χ. Όταν ο χρήστης κάνει κλικ σε ένα κουμπί πυροδοτείται το συμβάν (event) onclick και εμείς μπορούμε να το συνδέσουμε με τη συνάρτηση π.χ. doSomething(). Έτσι μόλις ο χρήστης πατήσει το κουμπί θα εκτελεστεί η συνάρτηση doSomething(). Κάθε αντικείμενο μπορεί να πυροδοτήσει αρκετά συμβάντα που μπορούν να συνδεθούν με αντίστοιχες μεθόδους.

Η σχεδίαση των παραθύρων γίνεται συνήθως κάνοντας χρήση έτοιμων εργαλείων που παρέχουν τα περιβάλλοντα ανάπτυξης εφαρμογών όπως είναι το Apache Netbeans IDE. Κάθε τέτοιο περιβάλλον ανάπτυξης έχει κάπου μια παλέτα αντικειμένων και ένα σχεδιαστικό εργαλείο παραθύρων όπου τοποθετούμε πάνω του τα αντικείμενα και παραμετροποιούμε τις ιδιότητές τους (properties).

Τη στιγμή που χρησιμοποιούμε το σχεδιαστικό εργαλείο παραθύρου, το περιβάλλον ανάπτυξης παράγει κώδικα java που αντικατοπτρίζει πλήρως την εικόνα του παραθύρου που έχουμε σχεδιάσει. Έτσι για κάθε παράθυρο δημιουργείται μια κλάση java που έχει δεδομένα μέλη (data members) που αντιστοιχούν στα χαρακτηριστικά-ιδιότητες (properties) του παραθύρου, ενσωματωμένα αντικείμενα (embedded objects) που αντιστοιχούν στα οπτικά αντικείμενα που έχουμε πάρει από την παλέτα αντικειμένων και τα έχουμε «ρίξει» πάνω στο παράθυρο, καθώς και συναρτήσεις μέλη που αντιστοιχούν στους event handlers που έχουμε δημιουργήσει.

Οι προγραμματιστές έχουμε την ελευθερία/δυνατότητα να επέμβουμε στον κώδικα που έχει παράγει το περιβάλλον ανάπτυξης. Αυτό σημαίνει πως μπορούμε να προσθέσουμε επιπλέον δεδομένα μέλη (data members) καθώς και συναρτήσεις μέλη (function members) στην κλάση του παραθύρου. Αυτό είναι κάτι που γίνεται κατά κανόνα μόλις ολοκληρωθεί η σχεδίαση του παραθύρου/ων, αλλιώς δε μπορεί να εμπλουτιστεί η κλάση έτσι ώστε να υλοποιηθεί αποτελεσματικά η λειτουργικότητα του παραθύρου και της εφαρμογής. Για παράδειγμα θα πρέπει να γραφτεί μια μέθοδος που θα κάνει σύνδεση με το ΣΔΒΔ της oracle, μια άλλη μέθοδος που θα κάνει εισαγωγή στοιχείων στη ΒΔ, μια άλλη που θα υπολογίζει τη λύση μιας μαθηματικής εξίσωσης,

μια άλλη που θα βρίσκει το μέσον όρο από ένα πίνακα(array) βαθμολογιών, μια άλλη που θα γράφει σε αρχείο, κλπ.

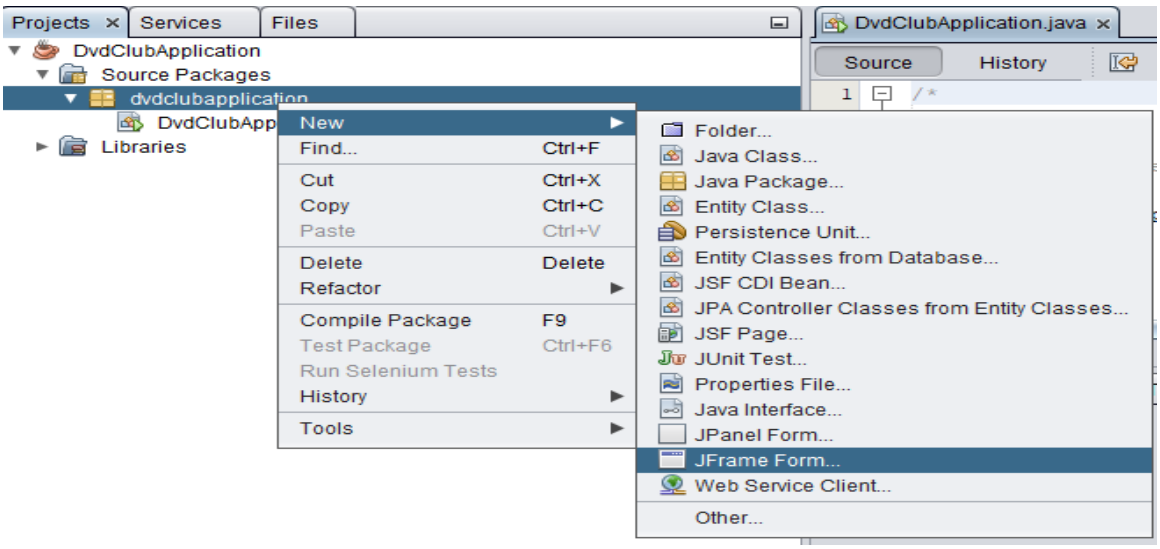
Οι προγραμματιστές πρέπει να μπορούν να γράφουν συναρτήσεις-μέλη ώστε μέσα από ένα παράθυρο να μπορεί να ανοίξει ένα καινούργιο παράθυρο και επιπλέον να στέλνει παραμέτρους στο νέο παράθυρο που θα ανοίξει (επικοινωνία παραθυρικών αντικειμένων), καθώς και να λαμβάνει παραμέτρους από αυτό που μόλις έκλεισε.

10.13 Σχεδίαση της κεντρικής φόρμας της εφαρμογής

Ακολουθεί η δημιουργία της φόρμας.

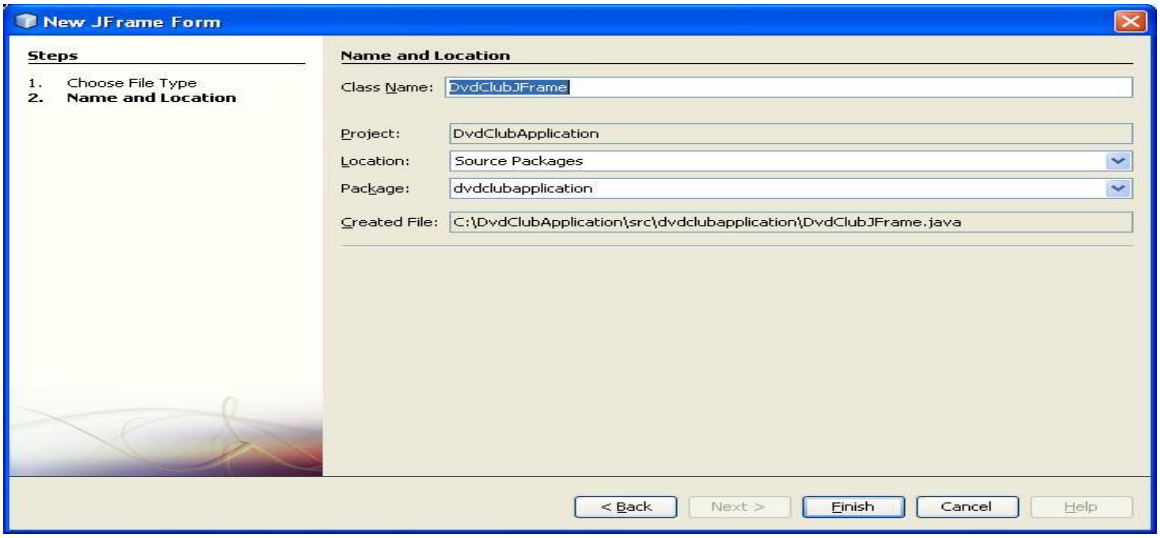
10.13.1 Δημιουργία κλάσης της κεντρικής φόρμας

Για να δημιουργήσουμε την κεντρική φόρμα κάνουμε δεξί κλικ στο πακέτο "dvdclubapplication" και επιλέγουμε «New →'JFrame Form' όπως φαίνεται στην εικόνα 10.14.



Εικόνα 10.14 Επιλογή JFrame Form για τη δημιουργία της κεντρικής φόρμας

Εμφανίζεται το παράθυρο της εικόνας 10.15 για να δώσουμε το όνομα της κλάσης της κεντρική φόρμας.



Εικόνα 10.15 Εκχώρηση ονόματος κλάσης κεντρικής φόρμας

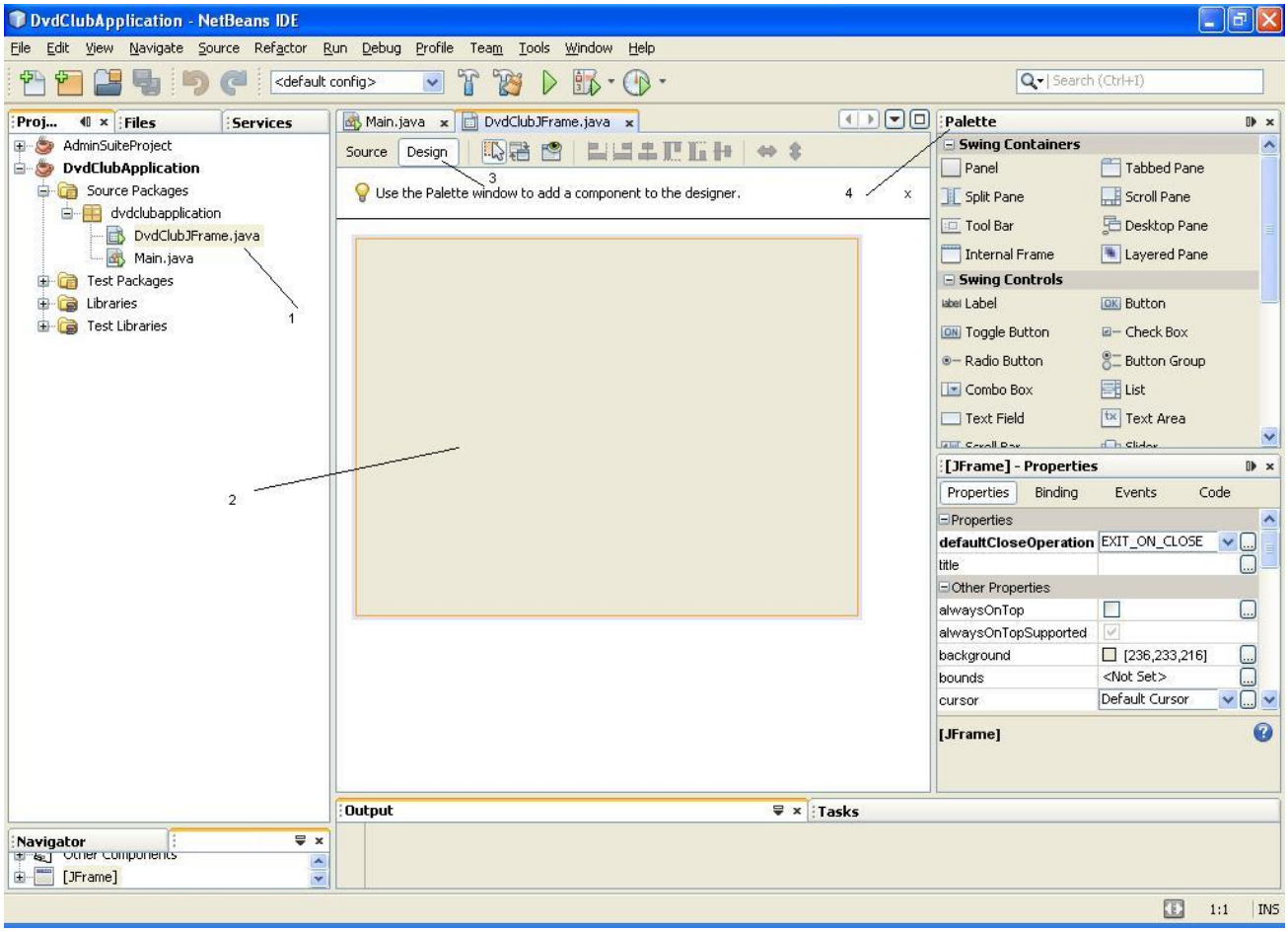
Στο πλαίσιο Class Name δίνουμε DvdClubJFrame. Σε αυτό το παράθυρο βλέπουμε επίσης σε ποιο πακέτο (dvdclubapplication) θα αποθηκευτεί η νέα μας κλάση (φόρμα). Τέλος, επιλέγουμε Finish.

Σημείωση

Η κεντρική φόρμα σε μια παραθυρική εφαρμογή java συνηθίζεται να ανήκει στην κλάση JFrame, ενώ όλες οι θυγατρικές φόρμες συνηθίζεται να ανήκουν στην κλάση JDialog.

Αφού επιλέξουμε Finish, όπως φαίνεται στην εικόνα 10.16, παρατηρούμε τα εξής:

1) Μέσα στο πακέτο dvdclubApplication δημιουργείται το αρχείο DvdClubJFrame.java το οποίο περιέχει τον πηγαίο κώδικα της κεντρικής μας φόρμας.
2) Στο κέντρο εμφανίζεται μια κενή φόρμα έτοιμη για να τη σχεδιάσουμε.
3) Εμφανίζεται το κουμπί Design. Όταν το επιλέγουμε εμφανίζεται το εργαλείο σχεδίασης παραθύρων (Design editor). Εδώ είναι προεπιλεγμένο. Δίπλα από το κουμπί Design υπάρχει το κουμπί Source που όταν το επιλέξουμε μας δείχνει τον πηγαίο κώδικα της φόρμας.
4) Δεξιά εμφανίζεται η παλέτα (Palette) των αντικειμένων σχεδίασης της φόρμας που μπορούμε να ρίξουμε πάνω στη φόρμα και να τα παραμετροποιήσουμε.



Εικόνα 10.16 Περιβάλλον σχεδίασης φόρμας

10.13.2 Προσθήκη αντικειμένων της παλέτας πάνω στην κεντρική φόρμα

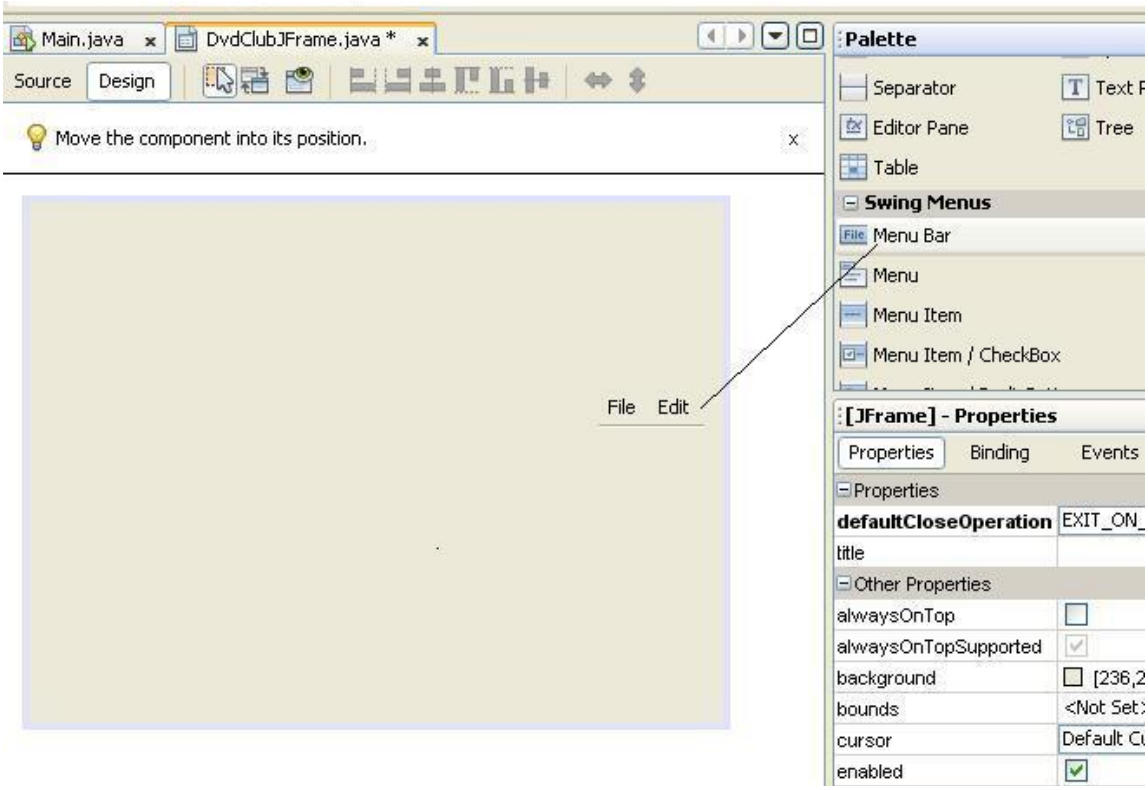
Για να προσθέσουμε ένα αντικείμενο πάνω στη φόρμα επιλέγουμε το αντικείμενο που θέλουμε από την παλέτα αντικειμένων και το κάνουμε drag & drop πάνω στη φόρμα. Στην επόμενη παράγραφο επεξηγείται πώς ενσωματώνουμε και παραμετροποιούμε τα αντικείμενα που αφορούν το μενού, αλλά με παρόμοιο τρόπο ενσωματώνουμε και παραμετροποιούμε όλα τα αντικείμενα της παλέτας. Ανάλογα με το αντικείμενο όμως πρέπει να γνωρίζουμε τα συγκεκριμένα χαρακτηριστικά του που χρειάζονται παραμετροποίηση.

Προσθήκη μενού στην φόρμα μας

Για να δημιουργηθεί ένα μενού σε μια φόρμα πρέπει να της ενσωματώσουμε τρεις τύπους αντικειμένων: μπάρα μενού(Menu Bar), μενού (Menu), και στοιχείο μενού (Menu Item).

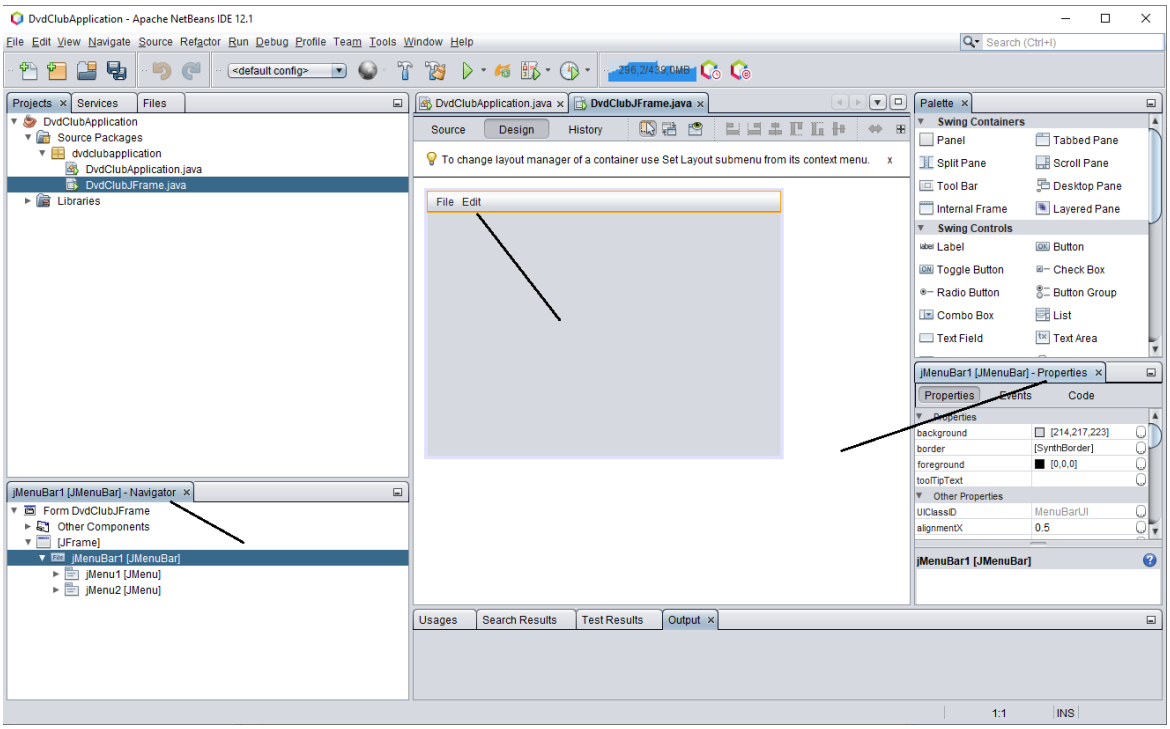
Menu Bar (Μπάρα μενού)

Για να ενσωματώσουμε το Menu Bar πηγαίνουμε στην παλέτα των αντικειμένων, ανοίγουμε τα Swing Menus και κάνουμε drag & drop το αντικείμενο Menu Bar πάνω στη φόρμα, όπως φαίνεται στην εικόνα 10.17.



Εικόνα 10.17Προσθήκη του αντικειμένου Menu Bar στη φόρμα

Η μπάρα μενού προστίθεται στη φόρμα όπως φαίνεται στην εικόνα 10.18.



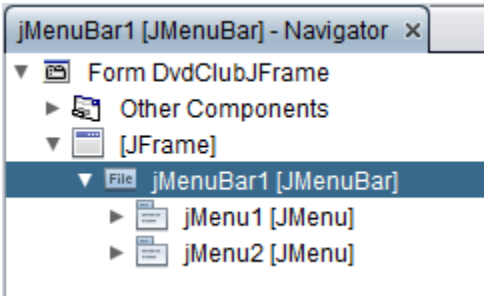
Εικόνα 10.18 Προσθήκη μπάρας μενού στη φόρμα

Στην εικόνα 10.18 (κάτω δεξιά) φαίνονται τα χαρακτηριστικά (Properties-Property Editor) του επιλεγμένου αντικειμένου jMenuBar1. Κάθε φορά που επιλέγουμε οπτικό αντικείμενο από τη φόρμα φαίνονται τα χαρακτηριστικά του και μπορούμε να τα ρυθμίζουμε-παραμετροποιούμε σε αυτό το σημείο.

Στην παραπάνω εικόνα φαίνεται επίσης κάτω αριστερά ο Navigator. Ο Navigator έχει οργανωμένα σε δεντρική μορφή όλα τα αντικείμενα και τα υπο-αντικείμενα της εφαρμογής μας. Έτσι σε αυτή τη φάση δείχνει το αντικείμενο της φόρμας μας JFrame να περιέχει τη μπάρα μενού που μόλις προσθέσαμε.

Menu (Βασικά μενού)

Η μπάρα μενού έχει επίσης δημιουργήσει από μόνη της δυο βασικά μενού όπως θα έχετε παρατηρήσει. Το μενού File και το μενού Edit. Αν ανοίξουμε στον Navigator το αντικείμενο jMenuBar θα δούμε αυτά τα αντικείμενα File & Edit, όπως φαίνεται στην εικόνα 10.19.



Εικόνα 10.19 Δημιουργία των μενού File και μενού Edit

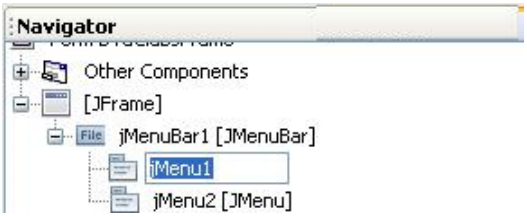
Το jMenu1 είναι το File. Γιατί όμως έχει διαφορετικό όνομα από αυτό που δείχνει η φόρμα; Το **jMenu1** είναι το **όνομα** του αντικειμένου που χρησιμοποιεί στον κώδικα ο προγραμματιστής, ενώ το **File** είναι η τιμή που έχει δοθεί στο χαρακτηριστικό **text** του αντικειμένου jMenu1. Αυτό μπορεί να επιβεβαιωθεί αν επιλέξουμε το αντικείμενο jMenu1 πάνω στον Navigator και δούμε τα χαρακτηριστικά στον Property Editor. Δείτε την εικόνα 10.20.

Εικόνα 10.20 Το αντικείμενο jMenu1 πάνω στον Inspector και τα χαρακτηριστικά του στον Property Editor

Σημείωση

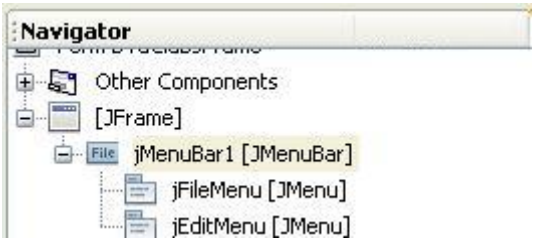
Τα ονόματα των αντικειμένων που δίνουν οι προγραμματιστές πρέπει να έχουν νόημα για να γίνεται ευκολότερη η ανάπτυξη και η συντήρηση της εφαρμογής. Έτσι είναι απαράδεκτο να υπάρχει όνομα αντικειμένου jMenu1 & jMenu2. Θα ήταν πολύ καλυτέρα αν ονομάζονταν jFileMenu & jEditMenu.

Στον Navigator, για να αλλάξουμε το όνομα του jMenu1 κάνουμε απλό κλικ πάνω του και αλλάζουμε το όνομα σε jFileMenu (εικόνα 10.21).



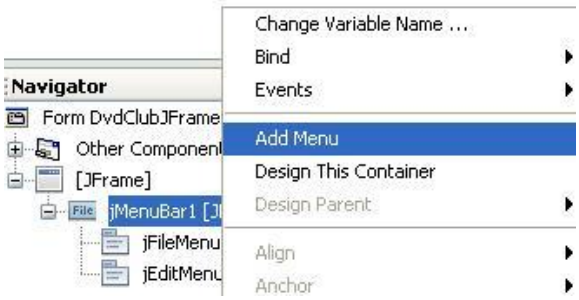
Εικόνα 10.21 Αλλαγή ονόματος μενού

Μετά τις αλλαγές των ονομάτων ο Navigator πρέπει να δείχνει όπως στην εικόνα 10.22,:



Εικόνα 10.22 Τα νέα ονόματα μενού jFileMenu, jEditMenu

Αν θέλουμε να δημιουργήσουμε καινούργιο βασικό μενού κάνουμε δεξί κλικ πάνω στο αντικείμενο jMenuBar1 και επιλέγουμε ‘Add Menu’. Δείτε εικόνα 10.23.



Εικόνα 10.23 Προσθήκη νέου βασικού μενού

Δημιουργούμε ένα βασικό μενού με όνομα αντικειμένου jViewMenu και διορθώνουμε το χαρακτηριστικό text σε 'View'.

Menu Items (Στοιχεία μενού)

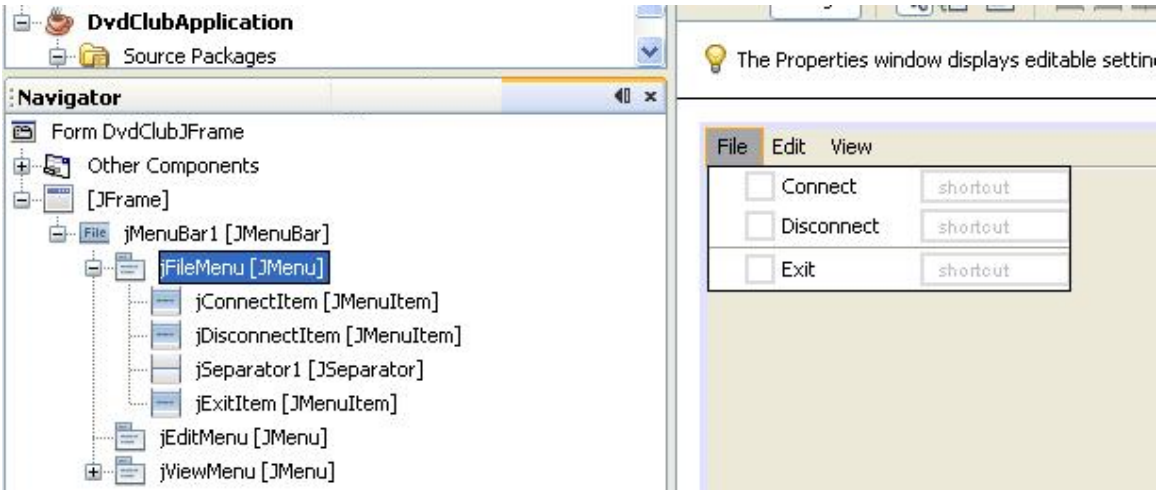
Κάθε βασικό μενού πρέπει να έχει τα στοιχεία μενού (menu items) που είναι και τα σημαντικότερα γιατί αυτά αντιστοιχούν στις λειτουργίες που εκτελεί ο χρήστης όταν χρησιμοποιεί την εφαρμογή.

Για να δημιουργήσουμε ένα menu item κάνουμε δεξί κλικ πάνω σε κάποιο βασικό μενού και επιλέγουμε 'Add From Palette' → 'Menu Item'. Δείτε εικόνα 10.24.

Εικόνα 10.24 Δημιουργία menu item

Για κάθε menu item δίνουμε κατάλληλο όνομα αντικειμένου και κατάλληλη τιμή στο property 'text' με τον τρόπο που είδαμε στην προηγουμένη ενότητα.

Στην παρακάτω εικόνα φαίνεται πως έχουμε δημιουργήσει τρία menu items με ονόματα αντικειμένων jConnectItem,jDisconnectItem,jExitItem με τιμές text αντίστοιχα 'Connect','Disconnect','Exit'.



Εικόνα 10.25 Δημιουργία τριών menu items

10.13.3 Προσθήκη συμβάντων και συναρτήσεων χειρισμού (event handlers) σε αντικείμενα

Κάθε αντικείμενο μπορεί να πυροδοτήσει αρκετά γεγονότα(events), αλλά στις περισσότερες περιπτώσεις χρησιμοποιούμε ένα πολύ μικρό αριθμό από αυτά. Κάθε γεγονός μπορεί να συνδεθεί με μια συγκεκριμένη μέθοδο/συνάρτηση χειρισμού του (event handler), ώστε αυτή να καλείται αυτόματα μόλις προκύψει το συμβάν. Ανάλογα με το είδος του αντικειμένου μπορούν να πυροδοτηθούν διαφορετικά σύμβαντα, που μπορούμε να χειριστούμε. Στην επόμενη παράγραφο θα ασχοληθούμε με τη διαχείριση των γεγονότων των menu items αλλά όλα τα γεγονότα των αντικειμένων τα χειριζόμαστε με παρόμοιο τρόπο. Θα πρέπει όμως να γνωρίζουμε τα συγκεκριμένα σύμβαντα που χρησιμοποιεί το αντικείμενο που χρησιμοποιούμε.

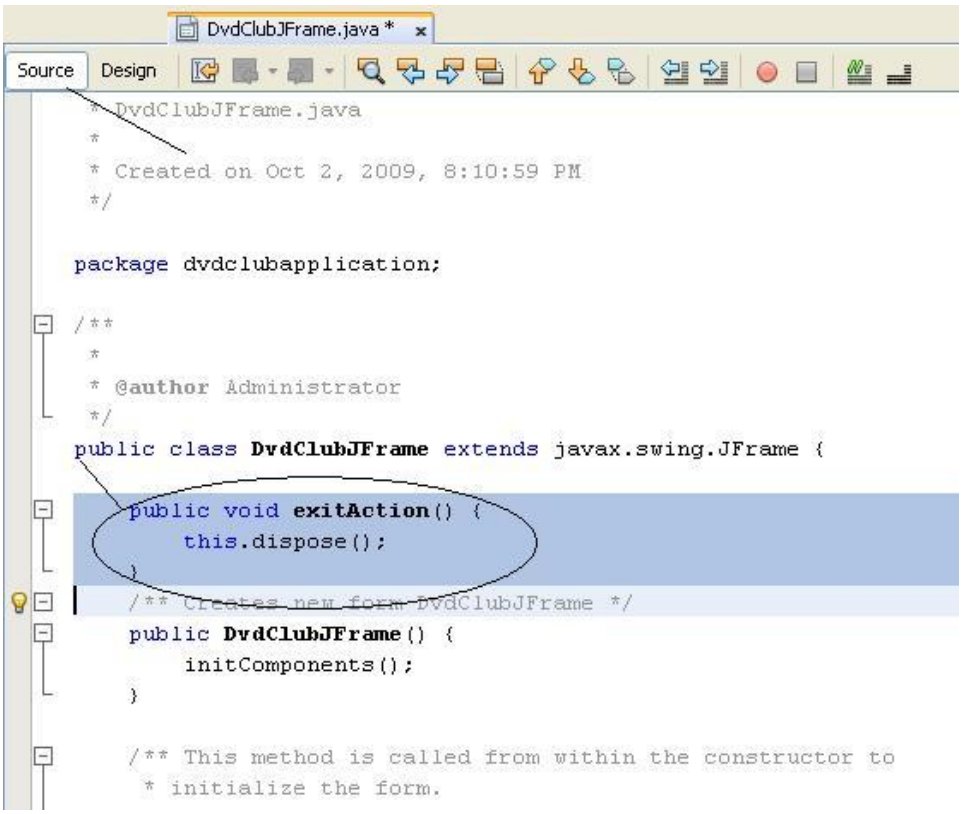
Χειρισμός γεγονότων menu item

Κάθε φορά που ο χρήστης κάνει κλικ σε ένα menu item για να εκτελέσει κάποια λειτουργία πυροδοτείται το συμβάν actionPerformed. Για να εκτελεστεί αυτή η λειτουργία θα πρέπει να έχουμε συνδέσει αυτό το event με μια μέθοδο/συνάρτηση μέλος της φόρμας (event/handlers). Έτσι όταν ο χρήστης επιλέξει Exit από το μενού θα πρέπει να εκτελεστεί π.χ. η συνάρτηση exitAction() για να τερματιστεί/κλίσει η εφαρμογή μας.

Χειρισμός των γεγονότων του menu item 'Exit'

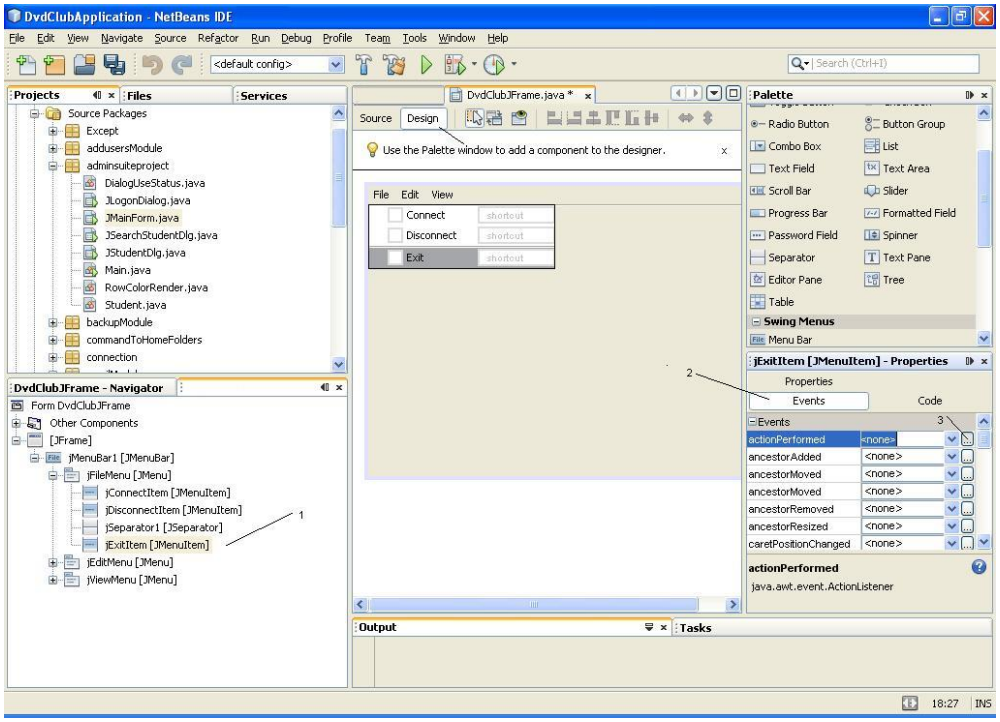
1) Όταν ο χρήστης επιλέξει από το File μενού την επιλογή Exit πρέπει να εκτελεστεί μια λειτουργία (action) η οποία θα τερματίσει την εφαρμογή μας. Για να προγραμματιστεί αυτό θα πρέπει να γίνουν τα εξής:
2) Να κατασκευάσουμε μια μέθοδο/συνάρτηση μέλος της κλάσης DvdClubJFrame που θα έχει όνομα π.χ. exitAction() ο κώδικας της οποίας θα προκαλεί τερματισμό της εφαρμογής.
3) Να συνδέσουμε το event 'actionPerformed' του αντικειμένου jExitItem με μια άλλη μέθοδο (π.χ. με όνομα jExitItemActionPerformed) που θα έχει το ρόλο του event handler.
4) Στον κώδικα του event handler(jExitItemActionPerformed) θα πρέπει να καλείται η μέθοδος exitAction().

Για να γίνει το (**α)** επιλέγουμε 'Source' για να δούμε τον πηγαίο κώδικα της φόρμας και προσθέτουμε τη μέθοδο/συνάρτηση exitAction. Δείτε την εικόνα 10.26.



Εικόνα 10.26 Πηγαίος κώδικας της φόρμας

Για το **(β)** επιλέγουμε 'Design' στη φόρμα, κάνουμε κλικ στο αντικείμενο jExitItem του Navigator, επιλέγουμε 'Events' στον property editor, και επιλέγουμε το κουμπί που είναι δίπλα από το event 'actionPerformed' (εικόνα 10.27).



Εικόνα 10.27 Δημιουργία event handler

Εμφανίζεται το παράθυρο (εικόνα 10.28) για να δώσουμε το όνομα του event handler

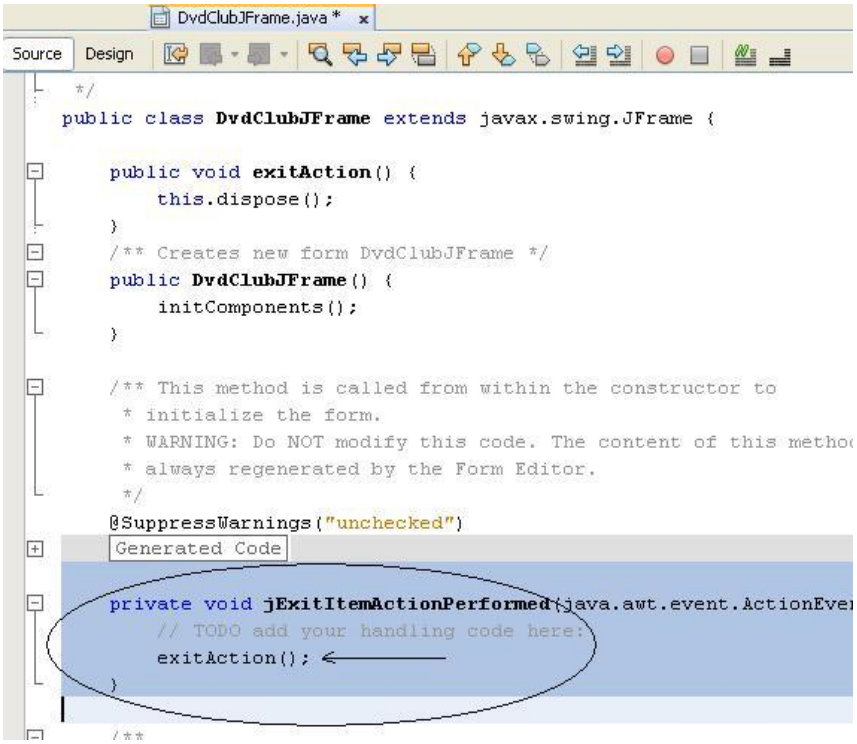
(jExitItemActionPerformed)

Εικόνα 10.28 Όνομα event handler

Σημείωση

Μπορούμε να απλοποιήσουμε την παραπάνω διαδικασία κάνοντας διπλό κλικ στο jExitItem του Inspector.

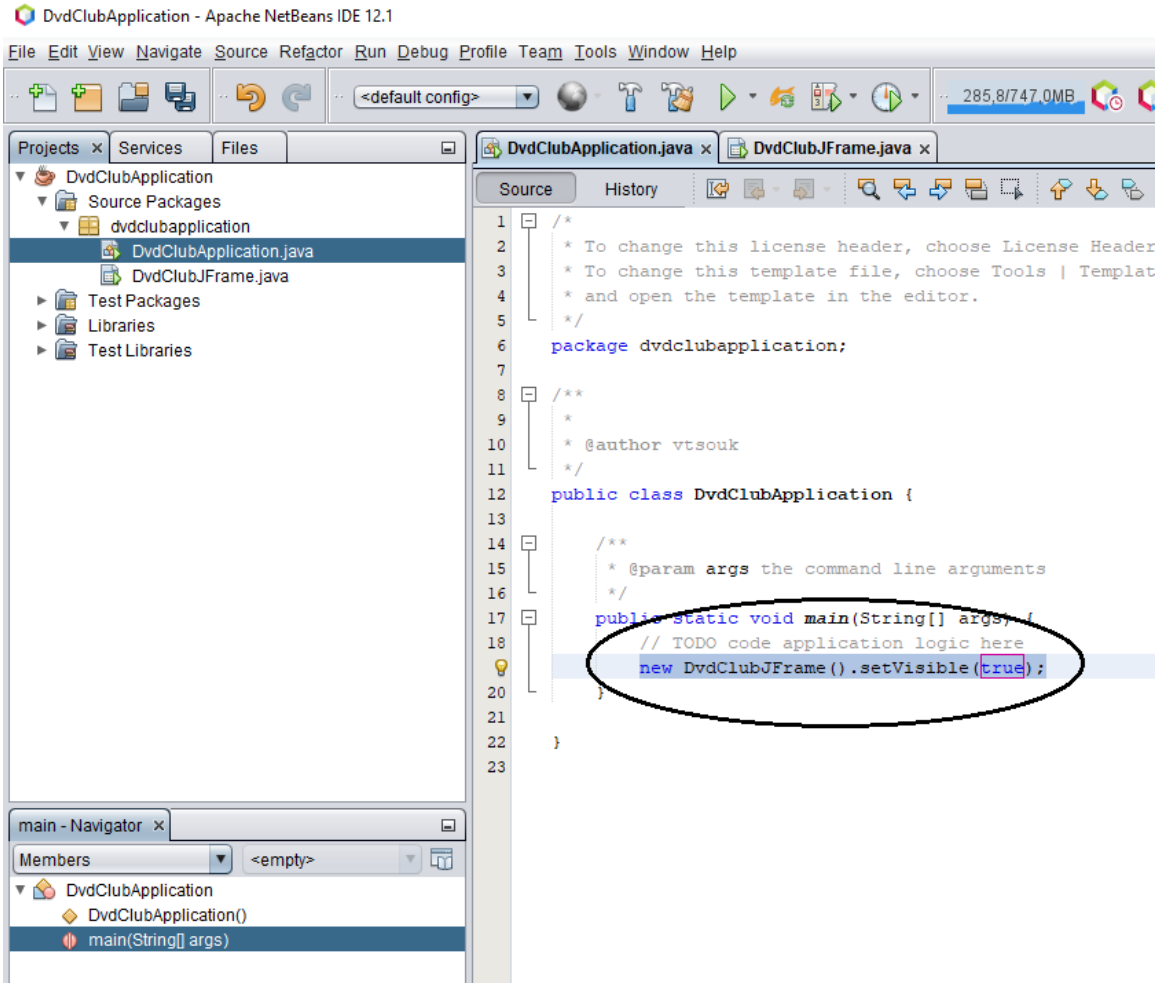
Για το (γ) επιλέγουμε 'Source' για να δούμε τον πηγαίο κώδικα της φόρμας, βρίσκουμε τον event handler - jExitItemActionPerformed που έχει παράγει αυτόματα το Netbeans και προσθέτουμε την κλήση της συνάρτησης exitAction() για να τερματίσει η εφαρμογή.



Εικόνα 10.29 Πηγαίος κώδικας

Βήμα 4ο Εκτέλεση της εφαρμογής

Η εκτέλεση της εφαρμογής μας ξεκινά από το πηγαίο αρχείο DvdClubApplication.java το οποίο διαθέτει τη συνάρτηση main. Ανοίγουμε λοιπόν το αρχείο και προσθέτουμε την εντολή που φαίνεται στην παρακάτω εικόνα για να ανοίξει η κεντρική φόρμα μας. Για να τρέξει η εφαρμογή επιλέγουμε Run->'Run Project' από το NetBeans IDE.



Εικόνα 10.30 Προσθήκη εντολής στον πηγαίο κώδικα για μα εμφανίσουμε τη φόρμα

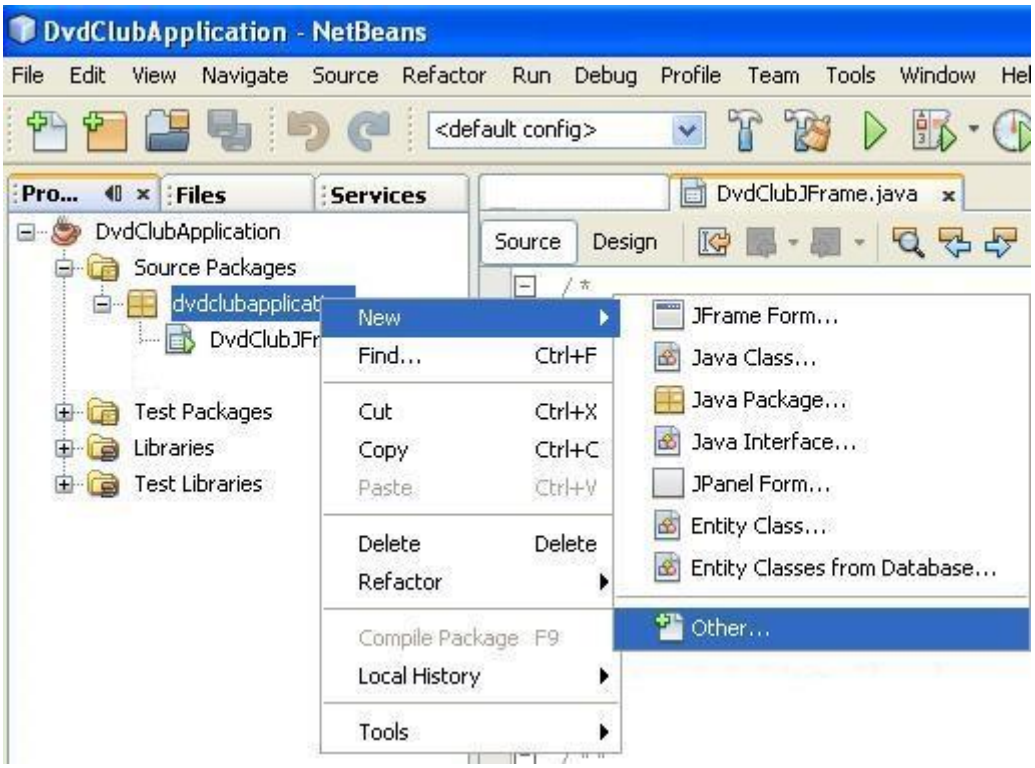
Στη μέθοδο main δημιουργούμε ένα αντικείμενο τύπου DvdClubJFrame (η κεντρική φόρμα) και αμέσως εκτελούμε τη μέθοδο setVisible με παράμετρο true για να εμφανίσουμε τη φόρμα.

Βήμα 5ο – Σχεδίαση βοηθητικών φορμών ή πλαισίων διαλόγου

Οι βοηθητικές φόρμες/πλαίσια διαλόγου είναι αντικείμενα τύπου JDialog και σχεδιάζονται/δημιουργούνται με τον ίδιο τρόπο που σχεδιάσαμε τη κεντρική μας φόρμα.

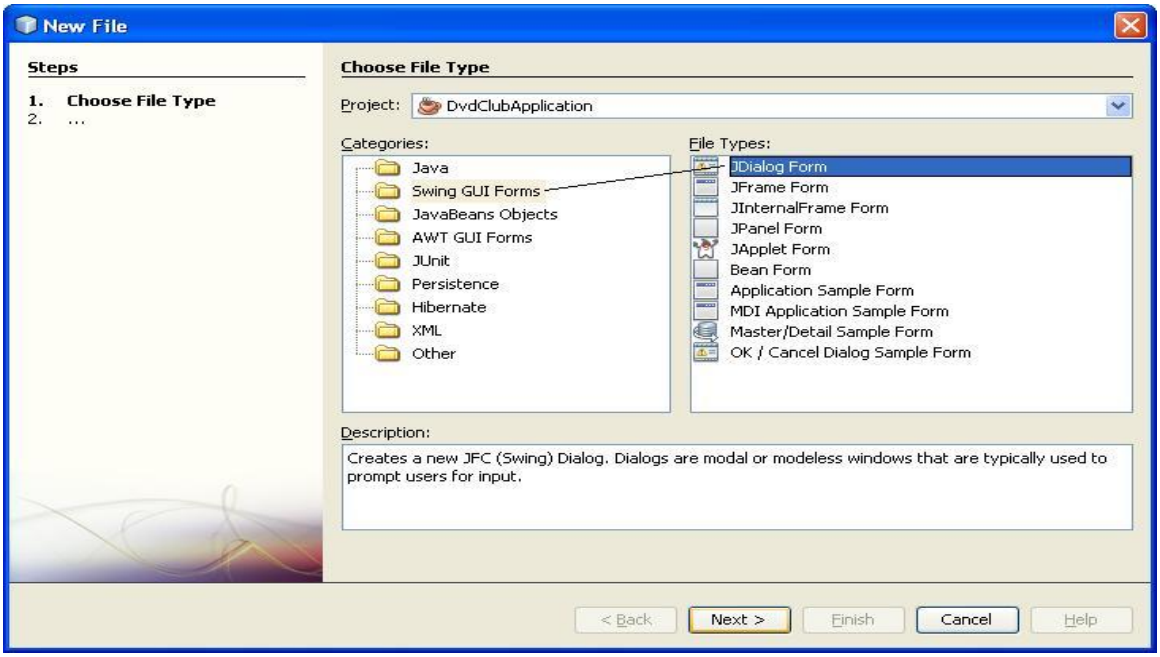
10.14 Εισαγωγή πλαισίου διαλόγου σύνδεσης με ΣΔΒΔ.

Για να δημιουργήσουμε το πλαισίου διαλόγου σύνδεσης πρέπει να τοποθετήσουμε ένα αντικείμενο τύπου JDialog στην εφαρμογή μας. Κάνουμε δεξί κλικ στο πακέτο dvdclubapplication→New→Other, όπως φαίνεται στην εικόνα 10.31.



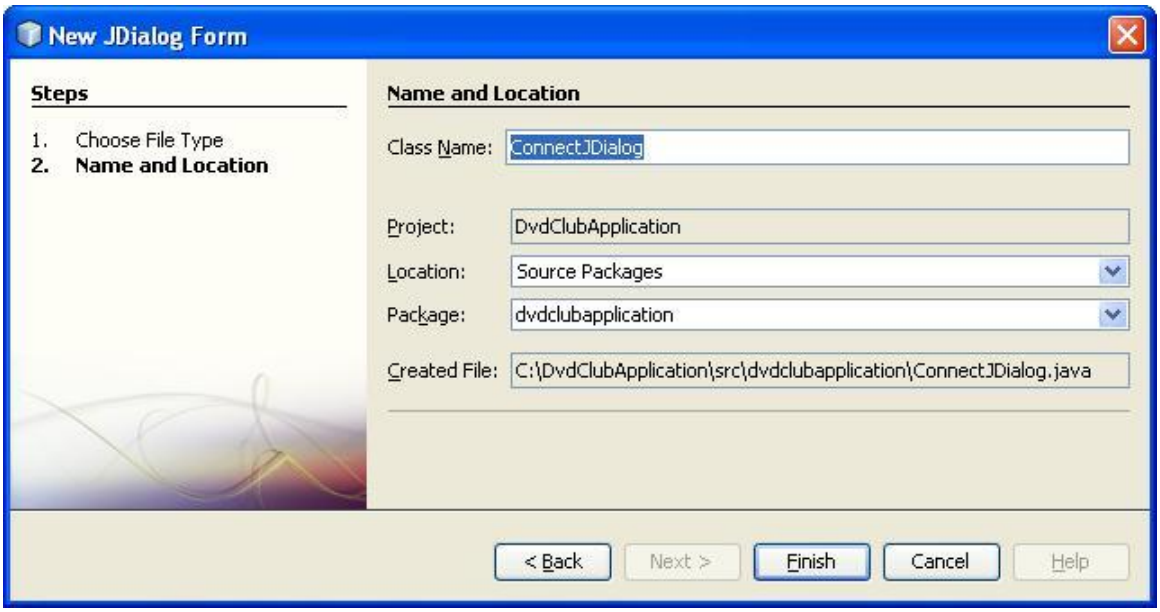
Εικόνα 10.31Δημιουργία πλαισίου σύνδεσης

Εμφανίζεται παράθυρο (εικόνα 10.32) στο οποίο επιλέγουμε 'Swing GUI Forms'→'JDialog Form' και Next.



Εικόνα 10.32 Επιλογή 'Swing GUI Forms' και στη συνέχεια 'JDialog Form'

Εμφανίζεται το πλαίσιο διαλόγου (εικόνα 10.33) στο οποίο δίνουμε το όνομα της κλάσης του πλαισίου διαλόγου 'Class Name': ConnectJDialog. Επιλέγουμε Finish.

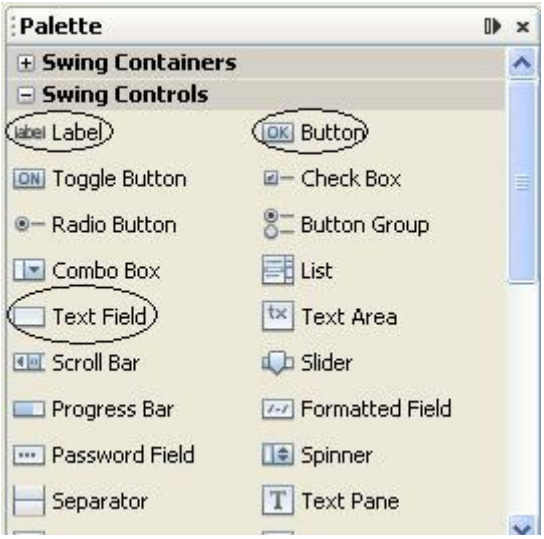


Εικόνα 10.33 Εκχώρηση ονόματος της κλάσης του πλαισίου διαλόγου ('Class Name')

Το νέο πλαίσιο διαλόγου εμφανίζεται στον Design Editor για να το σχεδιάσουμε, ενώ το αρχείο με τον πηγαίο κώδικά του αποθηκεύεται στο πακέτο dvdclubapplication με όνομα 'ConnectJDialog.java'.

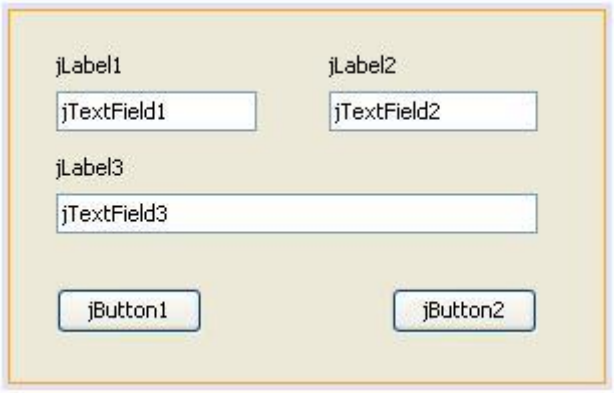
10.15 Σχεδίαση της φόρμας ConnectJDialog

Για να σχεδιάσουμε τη φόρμα σύνδεσης χρησιμοποιούμε τρεις τύπους αντικειμένων από την παλέτα αντικειμένων. Οι κλάσεις αυτές είναι **JLabel**, **JTextField** και **JButton** και βρίσκονται στην ομάδα Swing Controls της παλέτας. Δείτε την εικόνα. 10.34



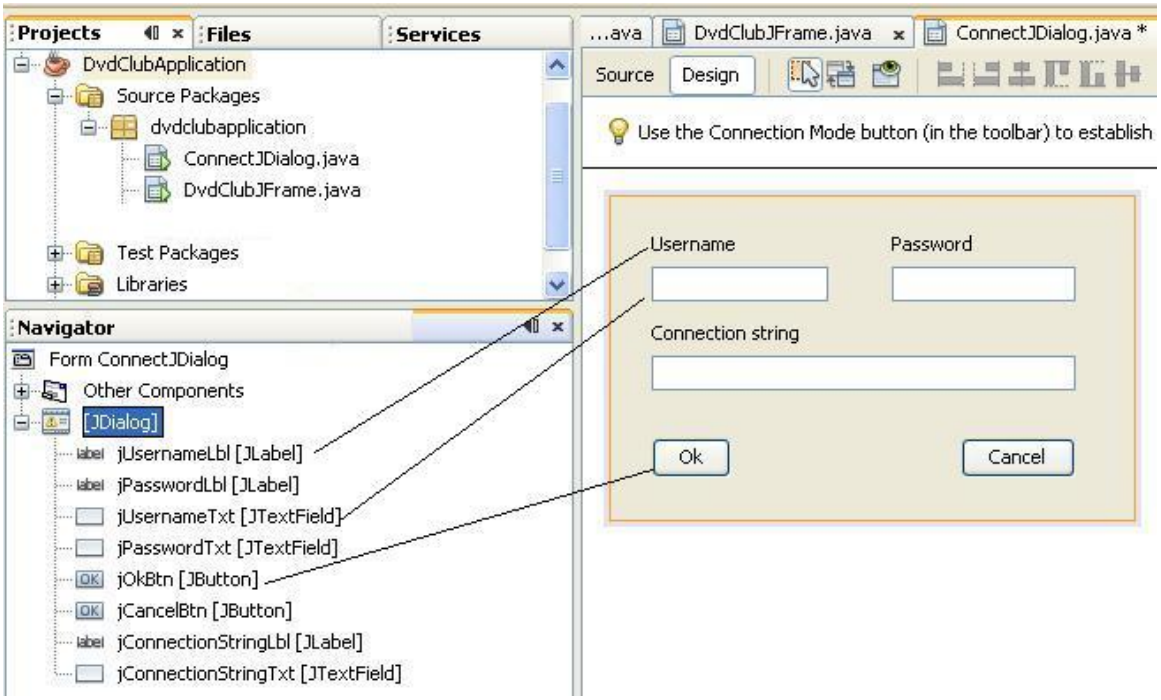
Εικόνα 10.34 Οι κλάσεις JLabel, JTextField και JButton

Η φόρμα σύνδεσής μας θα αποτελείται από τρεις ετικέτες, τρία πλαίσια κειμένου όπου ο χρήστης θα δίνει username, password και ρυθμίσεις σύνδεσης και δυο κουμπιά για αποδοχή ή ακύρωση. Τραβάμε λοιπόν από την παλέτα ένα-ένα όλα αυτά τα αντικείμενα και τα ρίχνουμε πάνω στη φόρμα όπως φαίνεται στην εικόνα 10.35.



Εικόνα 10.35 Η φόρμα σύνδεσης με τα αντικείμενά της

Παραμετροποιούμε όλα τα αντικείμενα της φόρμας ώστε να βλέπουμε την εικόνα 10.36. Οι παραμετροποιήσεις που κάναμε αφορούν την τροποποίηση του χαρακτηριστικού 'text' σε κάθε αντικείμενο και την απόδοση κατάλληλων ονομάτων στα αντικείμενα (δείτε Navigator).



Εικόνα 10.36 Η τελική μορφή της φόρμας σύνδεσης

10.16 Εμπλουτισμός της φόρμας σύνδεσης ConnectJDialog με κώδικα

Αφού έχουμε σχεδιάσει και παραμετροποιήσει τη φόρμα σύνδεσης πρέπει να την εμπλουτίσουμε με τον απαραίτητο κώδικα, ώστε να είναι λειτουργική. Η λειτουργία της φόρμας είναι πολύ απλή και σκοπός της είναι να δίνει ο χρήστης username, password και αλφαριθμητικό σύνδεσης και να επιλέγει ο χρήστης Ok 'η Cancel. Η φόρμα η ίδια δε θα διαθέτει κώδικα που θα επιχειρεί σύνδεση με τη βάση δεδομένων (αυτό θα το κάνει η κεντρική φόρμα), αλλά θα πρέπει να αποθηκεύει τα στοιχεία που έχει δώσει ο χρήστης καθώς και αν έχει επιλέξει Ok ή Cancel. Επιπλέον η φόρμα θα πρέπει να έχει ανενεργό το κουμπί Ok όταν ο χρήστης δεν έχει δώσει τίποτα στα πλαίσια κειμένου jUsernameTxt και jPasswordTxt.

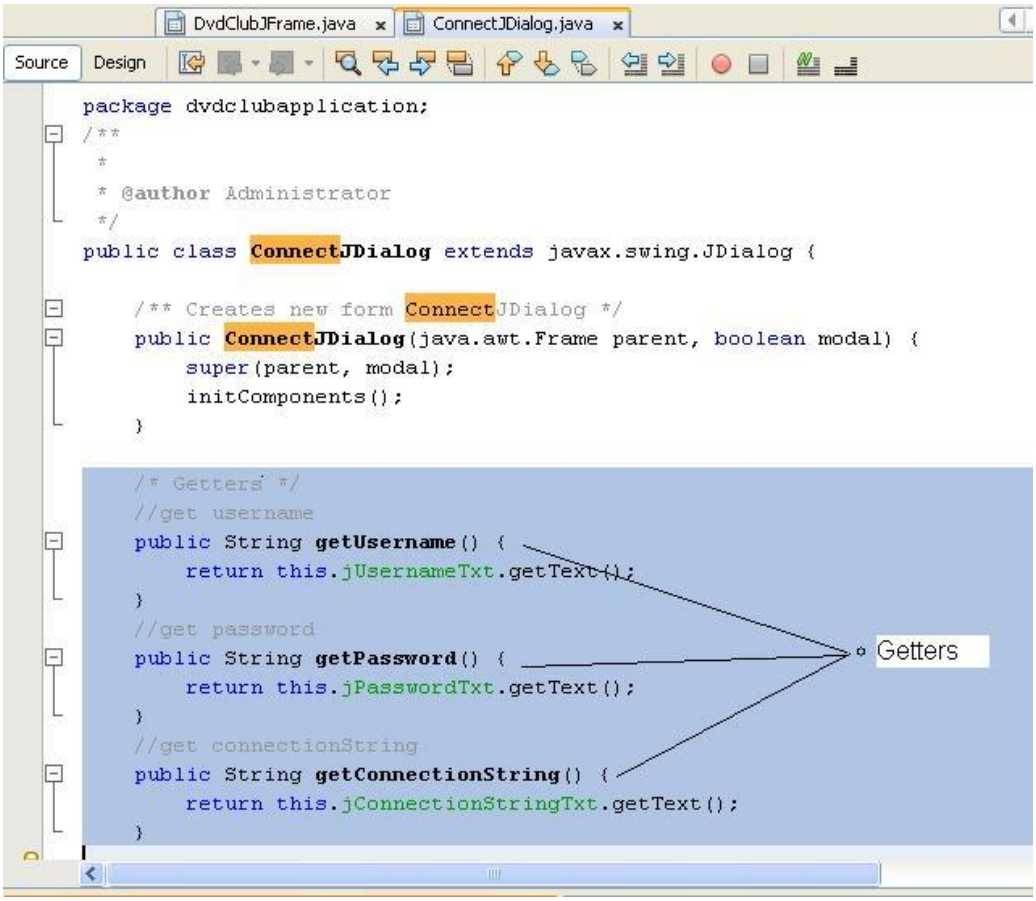
10.16.1 Επικοινωνία παραθύρων/αντικειμένων - Ανταλλαγή μηνυμάτων αντικειμένων

Όταν μια φόρμα ή αντικείμενο δημιουργεί/χρησιμοποιεί κάποια άλλη φόρμα ή αντικείμενο συνήθως στέλνει και παίρνει στοιχεία από αυτή/ο. Για παράδειγμα η κεντρική μας φόρμα κάποια στιγμή θα δημιουργήσει ένα αντικείμενο της φόρμας ConnectJDialog ώστε ο χρήστης να δώσει στοιχεία σύνδεσης και μόλις ο χρήστης πατήσει Ok για να κλίσει η φόρμα σύνδεσης τότε η κεντρική φόρμα παίρνει τα στοιχεία από τη φόρμα σύνδεσης και στη συνέχεια προσπαθεί να συνδεθεί ή όχι με ΣΔΒΔ ανάλογα με το αν ο χρήστης επιλέξει Ok ή Cancel στη φόρμα σύνδεσης. Σε αυτή την περίπτωση η κεντρική φόρμα τυγχάνει να μην στέλνει στοιχεία στη φόρμα σύνδεσης, αλλά μόνο να λαμβάνει στοιχεία από αυτή.

Για να γίνει ανταλλαγή στοιχείων από μια φόρμα ή αντικειμένου σε άλλη χρησιμοποιούνται επιπλέον μέθοδοι που στην ορολογία του αντικειμενοστραφή προγραμματισμού λέγονται **getters** και **setters**. Οι setters είναι μέθοδοι που χρησιμοποιούνται για να στείλουμε στοιχεία στο αντικείμενο μιας κλάσης, ενώ οι getters είναι μέθοδοι που χρησιμοποιούνται για να πάρουμε (επιστρέφουν) στοιχεία από το αντικείμενο μιας κλάσης.

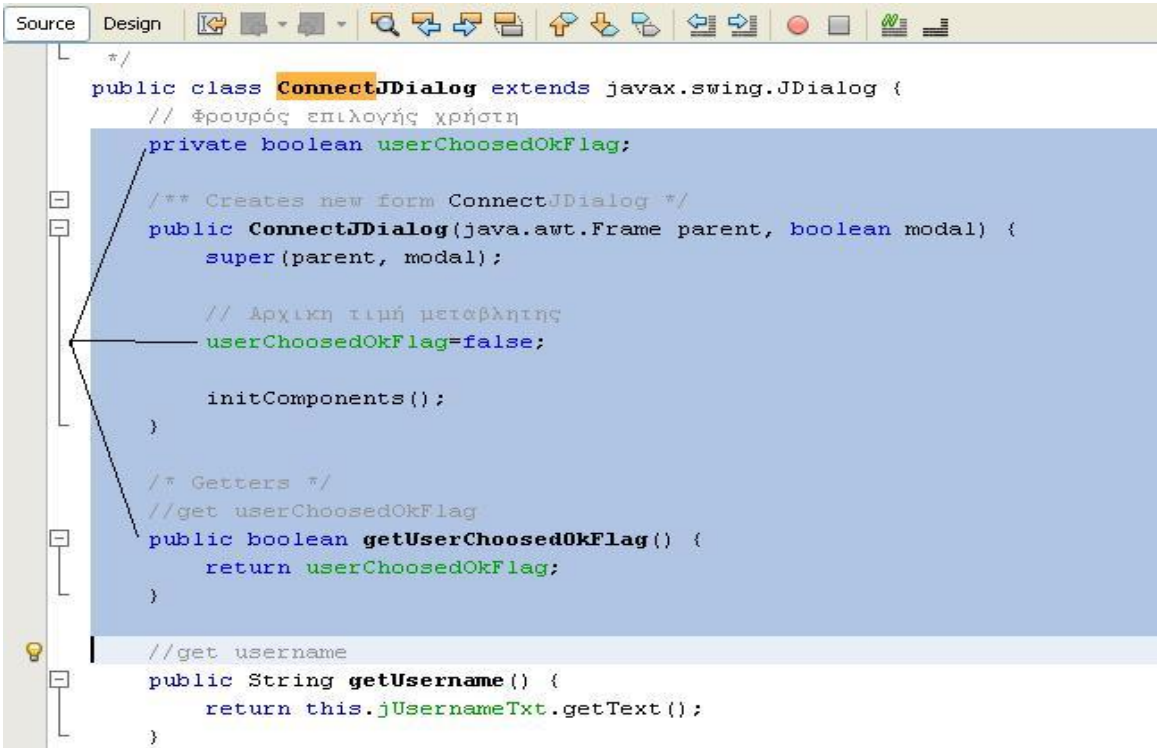
10.16.2 Οι getters τις φόρμας ConnectJDialog.

Όταν η κεντρική φόρμα χρειαστεί να κάνει σύνδεση με το ΣΔΒΔ θα πάρει (θα κάνει get) τα στοιχεία username, password, connectstring από το αντικείμενο της φόρμας ConnectJDialog. Για να είναι δυνατό αυτό πρέπει να κατασκευάσουμε τρεις συναρτήσεις μέλη/μέθοδοι που θα έχουν το ρόλο του getter στην κλάση ConnectJDialog. Οι συναρτήσεις αυτές επιστρέφουν το περιεχόμενο του property **text** των αντικειμένων jUsernameTxt, jPasswordTxt και jConnectionStringTxt και ο κώδικας τους φαίνεται στην εικόνα 10.37.



Εικόνα 10.37 Οι τρεις συναρτήσεις/μέθοδοι (getters) της φόρμας ConnectJDialog

Η κεντρική φόρμα θα χρειαστεί επίσης να ξέρει αν ο χρήστης έχει επιλέξει Ok ή Cancel. Για να γίνει εφικτό αυτό θα πρέπει να δηλώσουμε μια ακόμα μεταβλητή (π.χ. με όνομα userChoosedOkFlag) στη φόρμα ConnectJDialog και έναν ακόμα getter (π.χ. με όνομα getUserChoosedOkFlag) που θα επιστρέφει την τιμή της μεταβλητής. Η μεταβλητή αυτή επίσης θα πρέπει να παίρνει αρχική τιμή στο δημιουργό της φόρμας. Δείτε εικόνα 10.38.



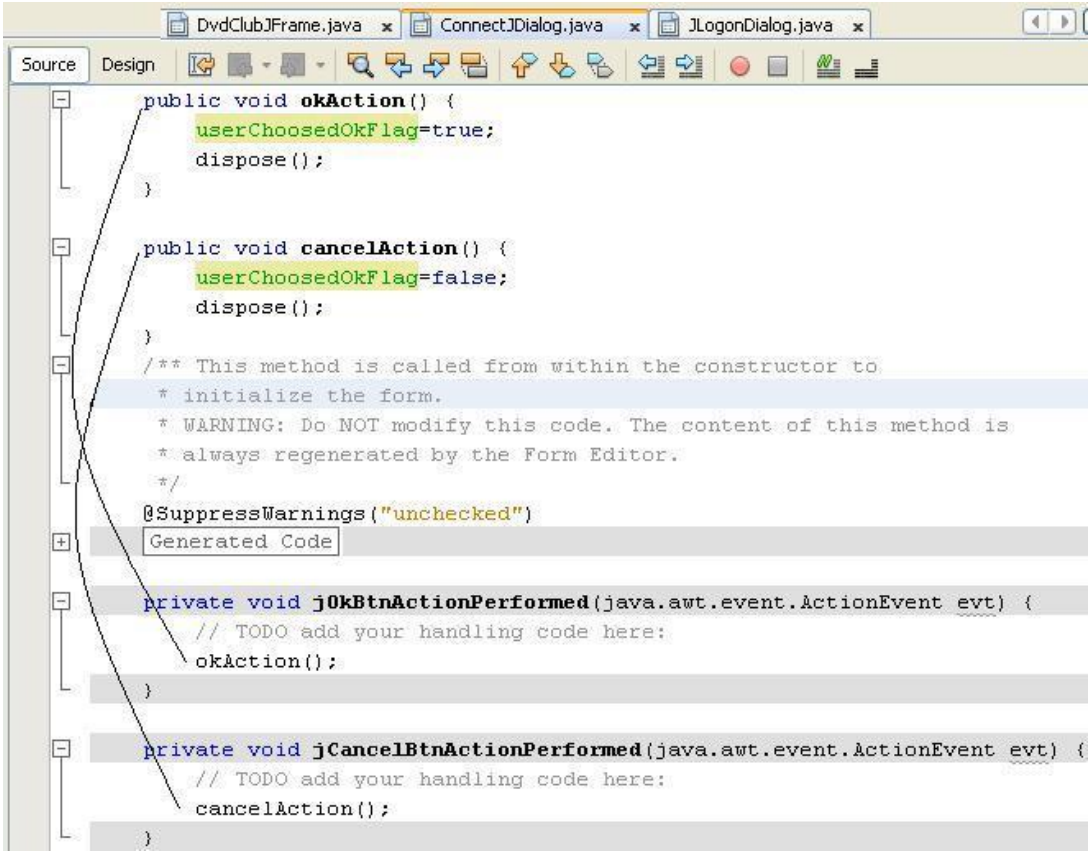
Εικόνα 10.38 Προσθήκη μεταβλητής και getter για να γνωρίζουμε αν ο χρήστης επιλέγει OK ή Cancel

10.16.3 Οι event handlers της φόρμας *ConnectJDialog*

Η φόρμα ***ConnectJDialog*** πρέπει να έχει τους παρακάτω event handlers για την ορθή χρήση της λειτουργικότητάς της:

- Έναν actionPerformed event handler για το κουμπί Ok. Μόλις ο χρήστης κάνει κλικ στο κουμπί jOkBtn τότε θα γίνεται true το δεδομένο μέλος userChoosedOkFlag και θα κλείνει η φόρμα. Δημιουργούμε λοιπόν μια μέθοδο okAction και τον jOkBtnActionPerformed handler και καλούμε από αυτόν τη μέθοδο okAction.
- Έναν actionPerformed event handler για το κουμπί Cancel. Μόλις ο χρήστης κάνει κλικ στο κουμπί jCancelBtn τότε θα γίνεται false το δεδομένο μέλος userChoosedOkFlag και θα κλείνει η φόρμα. Δημιουργούμε λοιπόν μια μέθοδο cancelAction και τον jCancelBtnActionPerformed handler και καλούμε από αυτόν τη μέθοδο cancelAction.

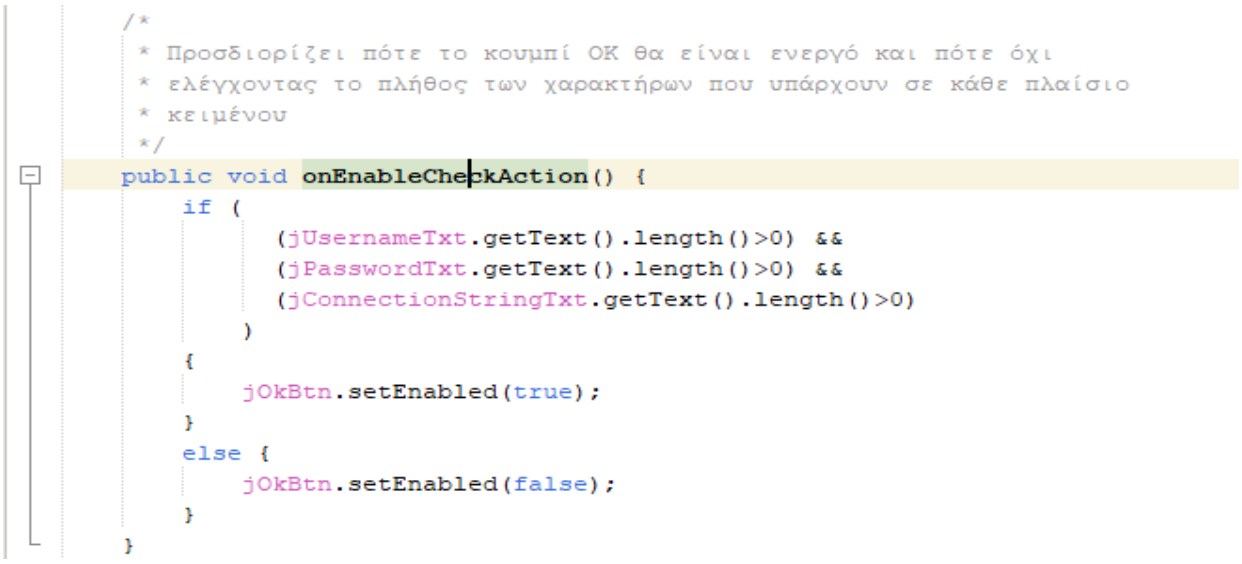
Στην εικόνα 10.39 φαίνεται ο κώδικας που απαιτείται.



Εικόνα 10.39 Πηγαίος κώδικας για handlers

Επιπλέον θα πρέπει η φόρμα να καθορίζει πότε θα είναι ενεργό το κουμπί jOkBtn. Το κουμπί Ok θα απενεργοποιείται όταν κάποιο από τα πλαίσια κειμένου Username, Password ή ConnectionString είναι κενά.

Για να υλοποιηθεί αυτό χρειαζόμαστε μια μέθοδο (π.χ. με όνομα OnEnableCheckAction) η οποία κάθε φορά που εκτελείται θα ελέγχει τα περιεχόμενα των πλαισίων κειμένου και θα κάνει ενεργό ή ανενεργό το jOkBtn. Η μέθοδος αυτή φαίνεται υλοποιημένη στην εικόνα 10.40.



Εικόνα 10.40 Υλοποίηση μεθόδου OkEnableCheck

Η μέθοδος αυτή θα πρέπει να καλείται κάθε φορά που ο χρήστης θα γράφει κείμενο σε κάποιο από τα τρία πλαίσια κειμένου. Κάθε φορά που πληκτρολογούμε κείμενο σε ένα πλαίσιο κειμένου πυροδοτείται το event

CaretUpdate. Έτσι θα πρέπει να δημιουργήσουμε τρεις event handlers (έναν για κάθε πλαίσιο κειμένου) οι οποίοι θα καλούν τη μέθοδο OnEnableCheck. Στην επόμενη εικόνα φαίνονται οι εν λόγω event handlers.

```
    private void jUsernameTxtCaretUpdate(javax.swing.event.CaretEvent evt) {
        // TODO add your handling code here:
        onEnableCheckAction();
    }

    private void jPasswordTxtCaretUpdate(javax.swing.event.CaretEvent evt) {
        // TODO add your handling code here:
        onEnableCheckAction();
    }

    private void jConnectionStringTxtCaretUpdate(javax.swing.event.CaretEvent evt) {
        // TODO add your handling code here:
        onEnableCheckAction();
    }
```

Εικόνα 10.41 Οι τρεις handlers

Βήμα 6ο Δημιουργία/Άνοιγμα/Κλήση βοηθητικών φορμών

Όταν ο χρήστης επιλέξει από το μενού την επιλογή File→Connect τότε θα πρέπει να ανοίξει η φόρμα ***ConnectJDialog*** ώστε να δώσει ο χρήστης τα στοιχεία σύνδεσης. Για να υλοποιηθεί αυτό πρέπει να γίνουν τα εξής:

- Να κατασκευαστεί μια συνάρτηση (π.χ. με όνομα connectAction()) στην κεντρική φόρμα η οποία θα δημιουργεί/ανοίγει τη φόρμα ***ConnectJDialog.*** Η συνάρτηση connectAction θα ελέγξει αν ο χρήστης απάντησε Ok ‘η Cancel (χρησιμοποιώντας τον getter getUserChoosedOkFlag) και αναλόγως θα επιχειρήσει να συνδεθεί στο ΣΔΒΔ ή όχι.
- Να κατασκευαστεί ένας actionPerformed event handler (π.χ. με όνομα jConnectItemActionPerformed) ο οποίος θα καλεί την παραπάνω συνάρτηση connectAction.

Στην εικόνα 10.42 φαίνεται ο jConnectItemActionPerformed event handler και το τμήμα της συνάρτησης connectAction που ανοίγει τη φόρμα και κάνει τον έλεγχο για το αν ο χρήστης διάλεξε Ok ή Cancel.

```
private void jConnectItemActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    connectAction();
}

public void connectAction() {
    // Δημιούργησε το αντικείμενο cDlg που είναι τύπου ConnectJDialog
    // Δηλαδή αντικείμενο της φόρμας σύνδεσης
    ConnectJDialog cDlg= new ConnectJDialog(this, true);
    // Εμφάνισε τη φόρμα σύνδεσης χρησιμοποιώντας τη συνάρτηση
    // setVisible με παράμετρο true. Η παράμετρος false κρύβει τη φόρμα
    cDlg.setVisible(true);
    // Έλεγξε αν ο χρήστης πάτησε Ok ή Cancel ελέγχοντας
    // τον getter getUserChoosedOkFlag.
    // Αν ο getter επιστρέψει τιμή false τότε ο χρήστης είχε πατήσει cancel
    // οπότε τερματίζουμε τη συνάρτηση, διαφορετικά συνεχίζουμε στις εντολές
    // σύνδεσης με το ΣΔΒΔ.
    if (!cDlg.getUserChoosedOkFlag()) {
        System.out.println("User pressed Cancel!");
        return;
    }
    // Ακολουθεί ο κώδικας σύνδεσης με τον Oracle Database Server
```

Εικόνα 10.42 Ο jConnectItemActionPerformed event handler και το τμήμα της συνάρτησης connectAction που ανοίγει τη φόρμα και κάνει τον έλεγχο

Βήμα 7ο Σύνδεση με το ΣΔΒΔ της Oracle (Oracle Database Server)

Για να συνδεθούμε με τον Oracle Database Server πρέπει να γίνουν τα εξής:

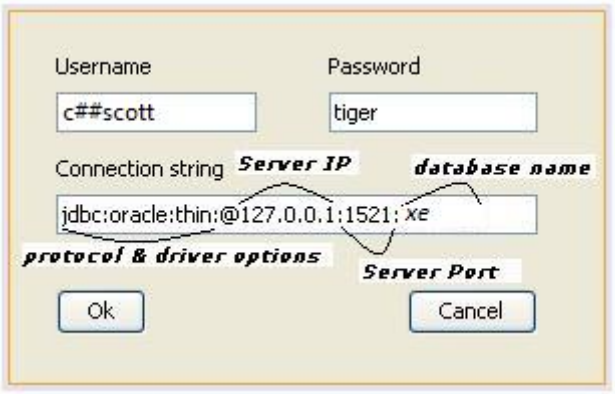
- Εισάγουμε στον πηγαίο κώδικα της κεντρικής φόρμας το πακέτο java.sql.
- Δηλώνουμε ένα αντικείμενο Connection που διεξάγει τη σύνδεση (session) του προγράμματός μας με τον database server. Το αντικείμενο αυτό διατηρεί τη σύνδεσή μας με τον server καθ' όλη τη διάρκεια χρήσης της ΒΔ και χρησιμοποιείται για να εκτελούμε εντολές sql.
- Συμπληρώνουμε την συνάρτηση connectAction με τον κατάλληλο κώδικα.

Δείτε σχετικά και την εικόνα 10.43.

```
import java.sql.*;
import static javax.swing.JOptionPane.showMessageDialog;
/* @author vtsouk */
public class DvdClubJFrame extends javax.swing.JFrame {

    static Connection con;

    public void connectAction() {
        // Δημιούργησε το αντικείμενο cDlg που είναι τύπου ConnectJDialog
        // Δηλαδή αντικείμενο της φόρμας σύνδεσης
        ConnectJDialog cDlg= new ConnectJDialog(this, true);
        // Εμφάνισε τη φόρμα σύνδεσης χρησιμοποιώντας τη συνάρτηση
        // setVisible με παράμετρο true. Η παράμετρος false κρύβει τη φόρμα
        cDlg.setVisible(true);
        // Έλεγξε αν ο χρήστης πάτησε Ok ή Cancel ελέγχοντας
        // τον getter getUserChoosedOkFlag.
        // Αν ο getter επιστρέψει τιμή false τότε ο χρήστης είχε πατήσει cancel
        // οπότε τερματίζουμε τη συνάρτηση, διαφορετικά συνεχίζουμε στις εντολές
        // σύνδεσης με το ΣΔΒΔ.
        if (!cDlg.getUserChoosedOkFlag()) {
            System.out.println("User pressed Cancel!");
            return;
        }
        // Ακολουθεί ο κώδικας σύνδεσης με τον Oracle Database Server
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            // Δημιουργία του αντικειμένου σύνδεσης.
            // Η μέθοδος getConnection δέχεται 3 παραμέτρους: αλφαριθμητικό σύνδεσης,username και password.
            // Τα τρία αυτά στοιχεία τα παίρνουμε από τους getters της φόρμας σύνδεσης ConnectJDialog
            // Αν η σύνδεση αποτύχει τότε προκαλείται κάποια εξαίρεση(exception) γι'αυτό
            // χρησιμοποιούμε try ... catch. Απαιτουνται 2 catch γιατί μπορεί να προκληθούν 2 τύποι εξαιρέσεων
            // SQLException ή Exception.
            con = DriverManager.getConnection(cDlg.getConnectionString(), cDlg.getUsername(), cDlg.getPassword());
            showMessageDialog(this, "Connected Successfully!!!");
            //Απενεργοποίησε την επιλογή σύνδεσης απο το μενού
            jConnectItem.setEnabled(false);
            //Κάλεσε τη συνάρτηση προγραματισμού ενεργειών
            setActionStatus();
        }
        catch (ClassNotFoundException | SQLException ex) {
            // Η εντολή αυτη εμφανίζει παράθυρο με το σφάλμα που προέκυψε
            showMessageDialog(this, ex.getMessage());
        }
    }
```

Εικόνα 10.43 Δημιουργία αντικειμένου Connection για τη διεξαγωγή της σύνδεσης

Επιπλέον κατά την εκτέλεση της φόρμας σύνδεσης ο χρήστης πρέπει να δώσει τα σωστά στοιχεία και να προσέξει πώς θα συμπληρώσει το πλαίσιο «Connection string». Δείτε και την εικόνα 10.44.



Εικόνα 10.44 Στοιχεία που συμπληρώνουμε στη φόρμα

Σημείωση

Στις πραγματικές εφαρμογές πρέπει να αποκρύπτουμε από τον τελικό χρήστη τις δυσνόητες για αυτόν ρυθμίσεις όπως είναι αυτές που φαίνονται στο πλαίσιο «Connection string». Η φόρμα σύνδεσης σας δίνεται στην παραπάνω μορφή καθαρά για εκπαιδευτικούς λόγους.

Βήμα 8ο – Αποσύνδεση από το ΣΔΒΔ της Oracle (Oracle Database Server)

Για να αποσυνδεθούμε από τον Oracle Database Server πρέπει να γίνουν τα εξής:

- Να κατασκευάσουμε μια συνάρτηση (π.χ. με όνομα disconnectAction()) η οποία θα κλείνει τη σύνδεση από το αντικείμενο τύπου Connection.
- Να κατασκευάσουμε ένα event handler (π.χ. με όνομα jDisconnectItemActionPerformed) για την επιλογή του μενού File→Disconnect η οποία θα καλεί τη συνάρτηση disconnectAction.

Δείτε σχετικά και την εικόνα 10.45.

```
    private void jDisconnectItemActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        disconnectAction();
    }

/* Κλείσε τη σύνδεση με τον database server εκτελώντας την εντολή close.
 * Κατά την αποσύνδεση μπορεί να προκληθεί εξαίρεση(exception),έτσι
 * χρησιμοποιούμε try ... catch. Απαιτείται 1 catch για τον τύπο
 * εξαίρεσης SQLException.
 */
    public void disconnectAction() {
        try {
            con.close();
        }
        catch (SQLException ex) {
            // Η εντολή αυτή εμφανίζει παράθυρο με το σφάλμα που προέκυψε
            showMessageDialog(this, ex.getMessage());
        }

    }
```

Εικόνα 10.45 Κώδικας για αποσύνδεση από τη βάση δεδομένων

Βήμα 9ο – Προγραμματισμός ενεργειών περιβάλλοντος διεπαφής. Καθοδήγηση χρήστη

Ο προγραμματισμός ενεργειών περιβάλλοντος διεπαφής ασχολείται με το ποιες λειτουργίες/ενέργειες θα είναι ενεργές στο χρήστη ή όχι κατά την πορεία εκτέλεσης της εφαρμογής. Ο προγραμματισμός αυτός πρέπει να είναι καθοδηγητικός στο χρήστη της εφαρμογής ώστε να συνεισφέρει στην ευκολία χρήσης της. Κακός προγραμματισμός ενεργειών μπορεί να σημαίνει, χωρίς λόγο, αύξηση της πολυπλοκότητας χρήσης και μπορεί να προκαλέσει την ενόχληση των χρηστών ή ακόμα και απόρριψη της εφαρμογής. Μια εφαρμογή που είναι παρά πολύ καλά οργανωμένη προγραμματιστικά με εξαίρεση το περιβάλλον επαφής υπάρχει πιθανότητα να αποτύχει.

Σημείωση

Οι προγραμματιστές της εφαρμογής πρέπει ακολουθούν αυστηρή πειθαρχία σε καθετί που σχεδιάζουν/προγραμματίζουν. Ένας σχετικά έμπειρος προγραμματιστής γνωρίζει πότε πειθαρχεί και πότε όχι, και δε θα πρέπει να θυσιάζει την προγραμματιστική πειθαρχία σε βάρος της ταχύτητας ολοκλήρωσης της εφαρμογής. Το τεμπέλιασμα των προγραμματιστών και η πίεση ολοκλήρωσης οδηγούν σε κακό προγραμματισμό των εφαρμογών με σοβαρό αντίτιμο την κακή σχεδίαση και τη δημιουργία σοβαρών προγραμματιστικών προβλημάτων (bugs) τα οποία θα κληθούν να επιλύσουν στο μέλλον ενώ θα χρειαστεί πολλαπλάσιος χρόνος διόρθωσης από αυτόν που έπρεπε να αφιερώσουν αρχικά για να υλοποιήσουν μια πειθαρχημένη προγραμματιστικά εφαρμογή.

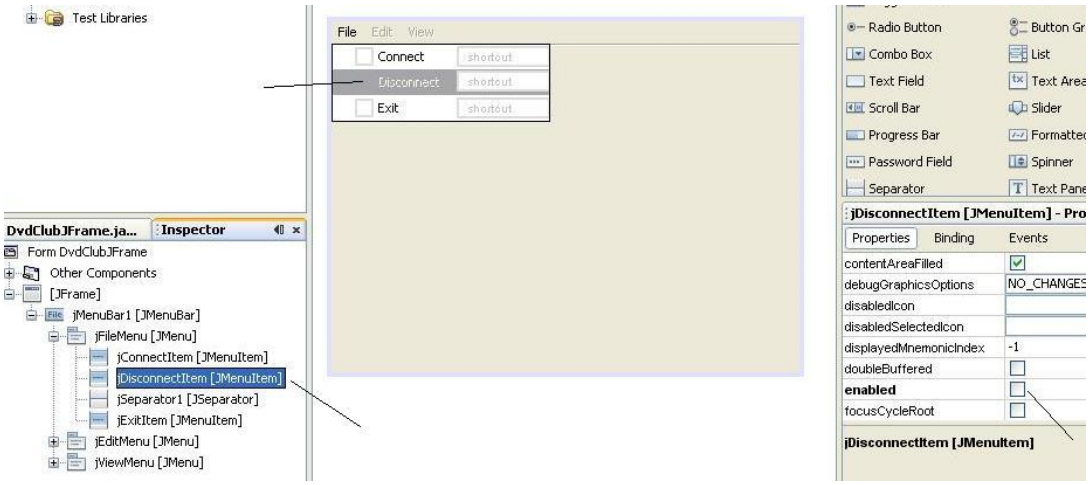
Στην εφαρμογή μας ο προγραμματισμός ενεργειών που πρέπει να γίνει περιγράφεται παρακάτω:

- Όταν εκκινήσει η εφαρμογή όλες οι επιλογές των μενού πρέπει να είναι ανενεργές, με εξαίρεση τις επιλογές File→Connect και File→Exit.
- Όταν διεξαχθεί σωστά η σύνδεση τότε η επιλογή File→Connect γίνεται ανενεργή, ενώ η File→Disconnect και όλες οι υπόλοιπες γίνονται ενεργές.
- Αν στη συνέχεια ο χρήστης επιλέξει File→Disconnect τότε όλες οι επιλογές των μενού γίνονται ανενεργές, με εξαίρεση τις επιλογές File→Connect και File→Exit.

Με αυτόν τον τρόπο, δηλαδή με το ποιες επιλογές βλέπει ενεργές ή ανενεργές, και ανάλογα και με τις επιλογές που κάνει, ο χρήστης καθοδηγείται-εκπαιδεύεται σε έναν απλό και ξεκάθαρο τρόπο χρήσης της εφαρμογής.

Για να υλοποιήσουμε τον παραπάνω προγραμματισμό ακολουθούμε τα εξής βήματα:

α) Επιλέγουμε τον Design editor της κεντρικής φόρμας και επιλέγουμε από τον Navigator μια μια τις επιλογές των μενού που θέλουμε να γίνουν ανενεργές και στον Property editor αποτσεκάρουμε το χαρακτηριστικό **enabled**. Ενεργές θα είναι μόνον οι File→Connect και File→Exit, οπότε όλες τις υπόλοιπες τις αποτσεκάρουμε. Το βήμα αυτό εξασφαλίζει πως με το άνοιγμα της εφαρμογής θα έχουμε ενεργές μόνον τις λειτουργίες του μενού που πρέπει. Η εικόνα 10.46 δείχνει με ποιο τρόπο απενεργοποιήσαμε το jDisconnectItem.



Εικόνα 10.46 Απενεργοποίηση του jDisconnectItem

β) Κατασκευάζουμε μια συνάρτηση (π.χ. με όνομα setActionsStatus) η οποία θα ελέγχει ποιες λειτουργίες των μενού (ή των κουμπιών) είναι ενεργές/ανενεργές και αναλόγως θα αποφασίζει ποιες άλλες λειτουργίες των μενού θα γίνουν ενεργές/ανενεργές. Στην εικόνα 10.47 φαίνεται η συνάρτηση setActionsStatus.

```
public class DvdClubJFrame extends javax.swing.JFrame {

    //Δήλωση αντικειμένου σύνδεσης
    static Connection con;

    /*
     * Προγραμμάτισε ποιες επιλογές των μενού θα είναι ενεργές και ποιές όχι.
     */
    public void setActionStatus() {
        if ( !jConnectItem.isEnabled() ) {
            jDisconnectItem.setEnabled(true);
            jEditMenu.setEnabled(true);
            jViewMenu.setEnabled(true);
        }
        else {
            jDisconnectItem.setEnabled(false);
            jEditMenu.setEnabled(false);
            jViewMenu.setEnabled(false);
        }
    }
```

Εικόνα 10.47 Η συνάρτηση setActionsStatus

γ) Πρέπει να καθορίσουμε πότε θα καλούμε την παραπάνω συνάρτηση setActionsStatus. Θα καλείται κάθε φορά που αλλάζουν οι συνθήκες μιας λειτουργίας. (Συνήθως καλούμε την setActionsStatus στην τελευταία εντολή της συνάρτησης που τροποποίησε τις συνθήκες προγραμματισμού των ενεργειών). Στην παρακάτω εικόνα φαίνεται τροποποιημένη η συνάρτηση σύνδεσης connectAction σε συνδυασμό με την τροποποίηση των συνθηκών προγραμματισμού ενεργειών και της κλήσης της συνάρτησης setActionsStatus.

```
public void connectAction() {
    // Δημιουργησε το αντικειμενο cDlg που ειναι τυπου ConnectJDialog
    // Δηλαδη αντικείμενο της φόρμας συνδεσης
    ConnectJDialog cDlg= new ConnectJDialog(this,true);
    // Εμφάνισε τη φορμα συνδεσης χρησιμοποιωντας τη συναρτηση
    // setVisible με παραμετρο true. Η παραμετρος false κρυβει τη φορμα
    cDlg.setVisible(true);
    // Ελεγξε αν ο χρηστης πατησε Ok ή Cancel ελεγχοντας
    // τον getter getUserChoosedOkFlag.
    // Αν ο getter επιστρεψει τιμη false τοτε ο χρηστης ειχε πατησει cancel
    // οποτε τερματιζουμε τη συναρτηση, διαφορετικα συνεχιζουμε στις εντολες
    // συνδεσης με το ΣΔΒΔ.
    if (cDlg.getUserChoosedOkFlag()== false) {
        return;
    }
    // Ακολουθεί ο κώδικας σύνδεσης με τον Oracle Database Server
    try{
        // Φορτωση του driver της Oracle
        Class.forName("oracle.jdbc.driver.OracleDriver");
        // Δημιουργία του αντικειμενου σύνδεσης.
        // Η μεθοδος getConnection δεχεται 3 παραμετρους: αλαφαριθμητικο συνδεσης,username και password.
        // Τα τρια αυτα στοιχεια τα παιρνουμε απο τους getters της φορμας συνδεσης ConnectJDialog
        // Αν η συνδεση αποτυχει τοτε προκαλειται καποια εξαιρεση(exception) γι'αυτο
        // χρησιμοποιουμε try ... catch. Απαιτουνται 2 catch γιατι μπορει να προκληθουν 2 τυποι εξαιρεσεων
        // SQLException ή Exception.
        con = DriverManager.getConnection(cDlg.getConnectionString(),
                                          cDlg.getUsername(),
                                          cDlg.getPassword());
        //Απενεργοποιησε την επιλογή σύνδεσης απο το μενου
        jConnectItem.setEnabled(false);

    }
    catch (java.sql.SQLException e1) {
            // Η εντολη αυτη εμφανιζει παραθυρο με το σφαλμα που προεκυψε
            javax.swing.JOptionPane.showMessageDialog(this,e1.getMessage());
    }
    catch (Exception e2) {
            // Η εντολη αυτη εμφανιζει παραθυρο με το σφαλμα που προεκυψε
            javax.swing.JOptionPane.showMessageDialog(this,e2.getMessage());
    }
    //Καλεσε τη συναρτηση προγραματισμου ενεργειων
    setActionsStatus();
}
```

Εικόνα 10.48 Η συνάρτηση setActionsStatus καλείται κάθε φορά που αλλάζουν οι συνθήκες μιας λειτουργίας

Στην πρώτη έλλειψη φαίνεται η αλλαγή της συνθήκης προγραμματισμού ενεργειών(η επιλογή σύνδεσης του μενού γίνεται false αμέσως μετά τη σύνδεση), ενώ στη δεύτερη φαίνεται η κλήση της setActionsStatus. Η κλήση της τελευταίας θα μπορούσε να γίνει και αμέσως μετά την αλλαγή της συνθήκης προγραμματισμού ενεργειών

Στην εικόνα 10.49 φαίνεται τροποποιημένη η συνάρτηση αποσύνδεσης disconnectAction σε συνδυασμό με την τροποποίηση των συνθηκών προγραμματισμού ενεργειών (η επιλογή σύνδεσης του μενού γίνεται true αμέσως μετά την αποσύνδεση) και της κλήσης της συνάρτησης setActionsStatus.

```
/* Κλείσε τη σύνδεση με τον database server εκτελώντας την εντολή close.
 * Κατά την αποσύνδεση μπορεί να προκληθεί εξαίρεση(exception),έτσι
 * χρησιμοποιούμε try ... catch. Απαιτείται 1 catch για τον τύπο
 * εξαίρεσης SQLException.
 */
  public void disconnectAction() {
      try {
          con.close();
          con=null;
          showMessageDialog(this, "Disconnected!!!");
          //Ενεργοποίησε την επιλογή σύνδεσης απο το μενού
          jConnectItem.setEnabled(true);
          //Κάλεσε τη συνάρτηση προγραματισμού ενεργειών
          setActionStatus();
      }
      catch (SQLException ex) {
          // Η εντολή αυτή εμφανίζει παράθυρο με το σφάλμα που προέκυψε
          showMessageDialog(this, ex.getMessage());
      }

  }
```

Εικόνα 10.49 Η τροποποιημένη συνάρτηση αποσύνδεσης disconnectAction

Σημείωση

Όσο προστίθενται νέες λειτουργίες στην εφαρμογή πρέπει να αναπροσαρμόζεται ο κώδικας της συνάρτησης προγραμματισμού και οι συνθήκες ενεργειών. Επιπλέον η προσέγγιση αυτή είναι ενδεικτική και η μεθοδολογία υλοποίησης είναι στην κρίση του προγραμματιστή.

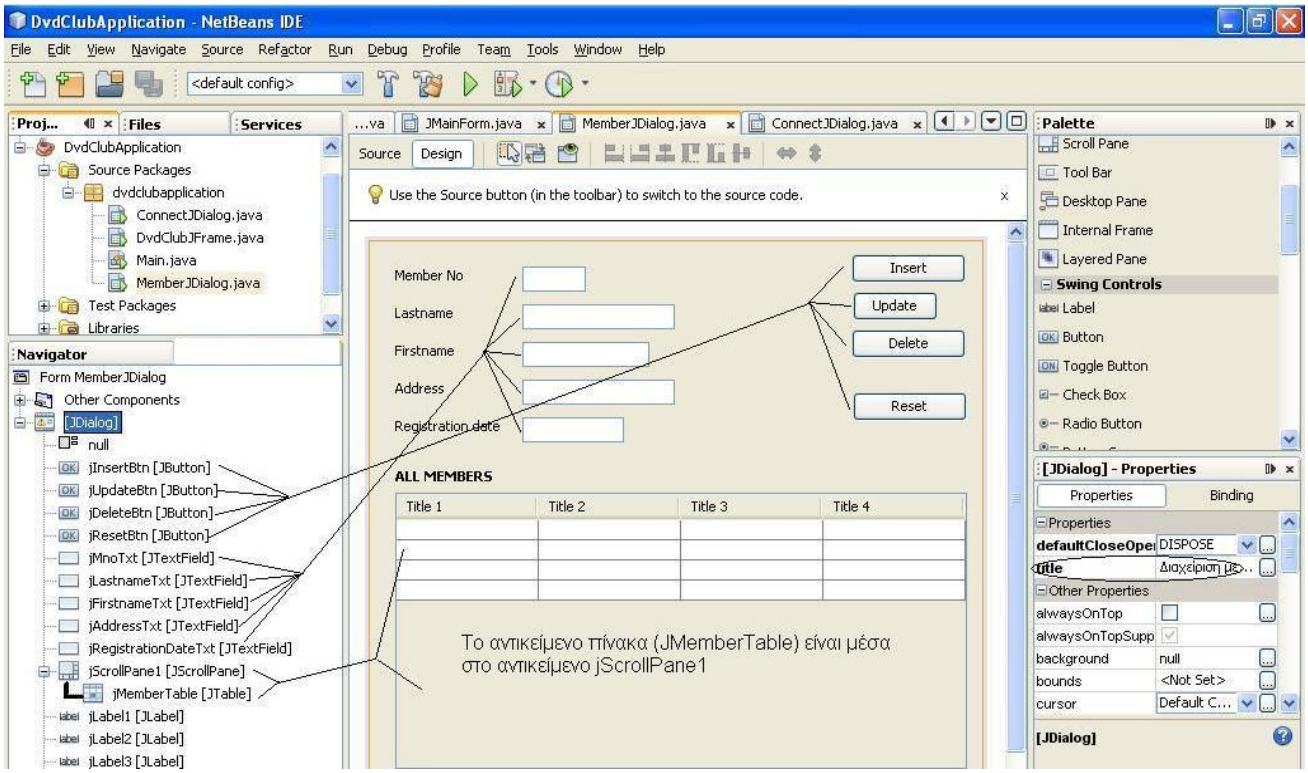
Βήμα 10ο – Επικοινωνία με τη βάση δεδομένων

Στις επόμενες παραγράφους θα δείξουμε μια φόρμα που θα διαχειρίζεται τα στοιχεία των μελών του club. Θα σχεδιάσουμε μια φόρμα όπου θα βλέπουμε σε ένα πινακάκι/πλέγμα όλα τα μέλη, ενώ θα υπάρχουν κουμπιά εισαγωγής, επεξεργασίας και διαγραφής μελών τα οποία αντίστοιχα θα είναι συνδεδεμένα με συναρτήσεις(event handlers) που θα κάνουν εισαγωγή, επεξεργασία και διαγραφή εγγραφών από τη βάση δεδομένων.

10.17 Σχεδίαση της φόρμας διαχείρισης μελών.

Δημιουργούμε μια φόρμα τύπου JDialog και δίνουμε class name MemberJDialog.

Η φόρμα διαχείρισης μελών φαίνεται στην εικόνα 10.50.



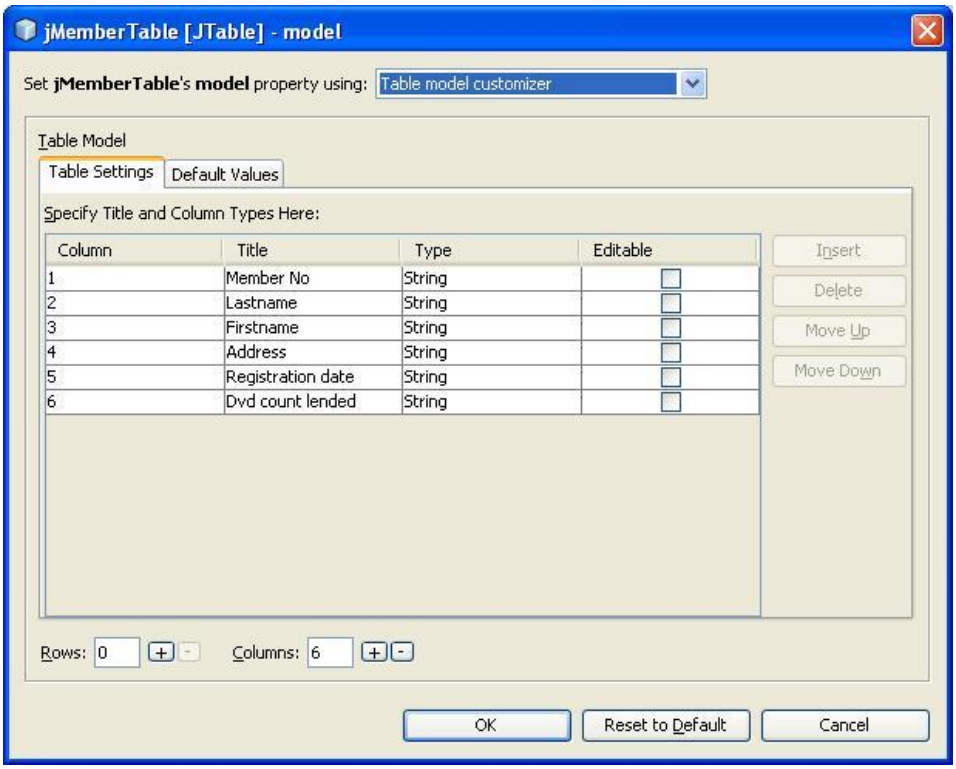
Εικόνα 10.50 Η φόρμα διαχείρισης μελών

Στην εικόνα 10.50 φαίνονται τα αντικείμενα και οι ονομασίες τους. Επίσης βλέπουμε πως για την εμφάνιση των μελών χρησιμοποιούμε δυο αντικείμενα: το ένα είναι τύπου JScrollPane και χρησιμοποιείται για να γίνεται αυτόματη ολίσθηση(scrolling) δεξιά-αριστερά ή πάνω-κάτω όταν τα μέλη που θα εμφανιστούν δε χωρούν στο παράθυρο. Το δεύτερο αντικείμενο είναι τύπου JTable και είναι αυτό που περιέχει τα μέλη. Τα αντικείμενα αυτά τα κάνουμε drag & drop από την παλέτα (δείτε Swing Containers & Swing Controls). Πρέπει να προσέξουμε το δεύτερο αντικείμενο να είναι φωλιασμένο μέσα στο πρώτο (ρίχνουμε το δεύτερο πάνω στο

πρώτο), αλλιώς δε θα λειτουργεί η ολίσθηση. Επιπλέον στην έλλειψη δεξιά φαίνεται ένα ενδιαφέρον property- το **title** το οποίο περιέχει τον τίτλο της φόρμας (ή των φορμών γενικότερα).

10.17.1 Παραμετροποίηση του αντικειμένου jMemberTable

Επιλέγουμε στον Inspector το αντικείμενο jMemberTable και στη συνέχεια επιλέγουμε από τον property editor τροποποίηση του χαρακτηριστικού **model** για να θέσουμε τις στήλες του πλέγματος. Δείτε και εικόνα 10.51.



Εικόνα 10.51 Ορισμός στηλών πλέγματος

10.17.2 Πέρασμα του αντικειμένου σύνδεσης από την κεντρική φόρμα στη φόρμα διαχείρισης μελών

Για να εκτελέσουμε οποιαδήποτε λειτουργία πάνω στη βάση δεδομένων θα πρέπει να έχουμε συνδεθεί. Η σύνδεση που γίνεται στη συνάρτηση connectAction της κεντρικής φόρμας DvdClubJFrame αποθηκεύεται στο αντικείμενο **con** τύπου Connection. Αν θέλουμε, όπως στην προκειμένη περίπτωση, η διαχείριση των μελών της ΒΔ να γίνεται από τη φόρμα MemberJDialog τότε αυτή η φόρμα θα πρέπει να χρησιμοποιήσει το αντικείμενο **con**. Υπάρχουν τρεις επιλογές για να γίνει αυτό:

α) Να τροποποιήσουμε/δηλώσουμε public static το δεδομένο μέλος σύνδεσης στην κεντρική φόρμα DvdClubJFrame ως εξής: public static Connection con;

Με αυτό τον τρόπο όλες οι άλλες φόρμες μπορούν να έχουν πρόσβαση στο αντικείμενο σύνδεσης χρησιμοποιώντας το όνομα της κλάσης της κεντρικής φόρμας ως εξής : DvdClubJFrame.con

β) Να δηλώσουμε ένα (ακόμα) δεδομένο μέλος σύνδεσης στη φόρμα MemberJDialog και να κατασκευάσουμε έναν setter (π.χ. με όνομα setConnectionParam()) με παράμετρο αντικείμενο τύπου Connection στην MemberJDialog τον οποίο θα καλεί το αντικείμενο της βασικής φόρμας.

γ) Να δηλώσουμε ένα (ακόμα) δεδομένο μέλος σύνδεσης στη φόρμα MemberJDialog και θα περνάμε το αντικείμενο con τύπου Connection στο δημιουργό της.

Στην υλοποίησή μας χρησιμοποιούμε την επιλογή α.

10.17.3 Κατασκευή συνάρτησης εισαγωγής μέλους στη βάση δεδομένων.

Η συνάρτηση insertMemberAction παίρνει τα στοιχεία από τα αντικείμενα πλαισίων κειμένου δημιουργεί ένα query αποδεκτό συντακτικά από την γλώσσα SQL της Oracle. Χρησιμοποιεί το αντικείμενο σύνδεσης con για να δημιουργήσει ένα αντικείμενο τύπου Statement το οποίο και εκτελεί το query με τη μέθοδο executeUpdate. Στην εικόνα 10.52 επεξηγείται αναλυτικά η συνάρτηση. Το query δεν περιέχει τη στήλη πλήθος ταινιών. Στο πλαίσιο κειμένου ημερομηνίας χρησιμοποιείται το format 'DD/MM/YY'.

```
/*
 * Η συνάρτηση αυτη εισαγει ενα μελος στον πινακα MEMBER της ΒΔ.
 * Για να γινει η εισαγωγη κανουμε τα εξης:
 * α. Δημιουργουμε ενα Insert query σε μια μεταβλητη τυπου String.
 * β. Δημιουργουμε απο το αντικειμενο con ενα αλλο αντικειμενο(insertStmt) τυπου  Statement
 * γ. Εκτελουμε το query χρησιμοποιωντας τη μεθοδο executeUpdate του αντικειμενου τυπου Statement
 * δ. Αποδεσμευουμε τους πορους του αντικειμενου τυπου Statement χρησιμοποιωντας την εντολη close
 * ε. Οι εντολες αυτες υποχρεωτικα πρεπει να βρισκονται σε block κωδικα try...catch
 *    για να εντοπισουμε ενδεχομενο exception, το οποιο και εμφανιζουμε με την showMessageDialog
 *    για να δουμε τι σφαλμα προεκυψε.
 */
public void insertMemberAction() {

    String query="INSERT INTO MEMBER(MNO,MLASTNAME,MFIRSTNAME,MADDRESS,MREGDATE) " +
                 "VALUES(" + jMnoTxt.getText() + "," +
                 "'"+ jLastnameTxt.getText() + "'," +
                 "'" + jFirstnameTxt.getText() + "'," +
                 "'" + jAddressTxt.getText() + "'," +
                 "TO_DATE('"+jRegistrationDateTxt.getText()+"','DD/MM/YY')" + ")";

    java.sql.Statement insertStmt;
    try {
        insertStmt = DvdClubJFrame.con.createStatement( );
        insertStmt.executeUpdate(query);
        insertStmt.close();
    }
    catch( java.sql.SQLException e) {
          javax.swing.JOptionPane.showMessageDialog(this,e.getMessage());
    }
}
```

α β γ δ

Εικόνα 10.52 Δημιουργία query σε γλώσσα SQL της Oracle

10.17.4 Κατασκευή συνάρτησης τροποποίησης στοιχείων ενός μέλους από τη ΒΔ.

Η συνάρτηση updateMemberAction παίρνει τα στοιχεία από τα αντικείμενα πλαισίων κειμένου, δημιουργεί ένα query αποδεκτό συντακτικά από την γλώσσα SQL της Oracle. Χρησιμοποιεί το αντικείμενο σύνδεσης con για να δημιουργήσει ένα αντικείμενο τύπου Statement το οποίο και εκτελεί το query με τη μέθοδο executeUpdate. Στην εικόνα που ακολουθεί επεξηγείται αναλυτικά η συνάρτηση. Το query δεν μπορεί να τροποποιήσει τον κωδικό μέλους καθώς και τη στήλη πλήθος ταινιών.

```
/*
* Η συνάρτηση αυτή τροποποιεί τα στοιχεία του μέλους που έχει κωδικό αυτόν που ο χρήστης έχει
* θέσει στο πλαίσιο κειμένου jMnoTxt με αυτά που έχει θέσει στα υπόλοιπα πλαίσια κειμένου.
* Η συνάρτηση μπορεί να τροποποιήσει όλα τα στοιχεία του μέλους -εκτός του κωδικού μέλους
* και του πλήθους των ταινιών που έχει δει.
*
* Για να γίνει η τροποποίηση κάνουμε τα εξής:
* α. Δημιουργούμε ένα update query σε μια μεταβλητή τύπου String.
* β. Δημιουργούμε από το αντικείμενο con ένα άλλο αντικείμενο(updateStmt) τύπου  Statement
* γ. Εκτελούμε το query χρησιμοποιώντας τη μέθοδο executeUpdate του αντικειμένου τύπου Statement
* δ. Αποδεσμεύουμε τους πόρους του αντικειμένου τύπου Statement χρησιμοποιώντας την εντολή close
* ε. Οι εντολές αυτές υποχρεωτικά πρέπει να βρίσκονται σε block κώδικα try...catch
*    για να εντοπίσουμε ενδεχόμενο exception, το οποίο και εμφανίζουμε με την showMessageDialog
*    για να δούμε τι σφάλμα προέκυψε.
*/
```

```
public void updateMemberAction() {
    String query="UPDATE MEMBER " +
            " SET MLASTNAME='"+jLastnameTxt.getText()+"'," +
            "     MFIRSTNAME='"+jFirstnameTxt.getText()+"',"+
            "     MADDRESS='"+jAddressTxt.getText()+"',"+
            "     MREGDATE=TO_DATE('"+jRegistrationDateTxt.getText()+"','DD/MM/YYYY')" +
            " WHERE MNO=" + jMnoTxt.getText();

    System.out.println("Query:\t" + query);

    Statement updateStatement;

    try {
        updateStatement= DvdClubJFrame.con.createStatement();
        updateStatement.executeUpdate(query);
        updateStatement.close();
    }
    catch (SQLException ex) {
        showMessageDialog(this, ex.getMessage());
    }
}
```

Εικόνα 10.53 Η συνάρτηση updateMemberAction παίρνει τα στοιχεία από τα αντικείμενα πλαισίων κειμένου και δημιουργεί ένα query σε γλώσσα SQL

10.17.5 Κατασκευή συνάρτησης διαγραφής ενός μέλους από τη ΒΔ.

Η συνάρτηση deleteMemberAction διαγράφει το μέλος με κωδικό που έχει θέσει ο χρήστης στο αντίστοιχο πλαίσιο κειμένου. Δημιουργεί ένα query αποδεκτό συντακτικά από την γλώσσα SQL της Oracle. Χρησιμοποιεί το αντικείμενο σύνδεσης con για να δημιουργήσει ένα αντικείμενο τύπου Statement το οποίο και εκτελεί το query με τη μέθοδο executeUpdate. Στην εικόνα 10.54 επεξηγείται αναλυτικά η συνάρτηση.

```
/*
 * Η συνάρτηση αυτή διαγράφει το μέλος που έχει κωδικό αυτόν που ο χρήστης έχει
 * θέσει στο πλαίσιο κειμένου jMnoTxt.
 *
 * Για να γίνει η διαγραφή κάνουμε τα εξής:
 * α. Δημιουργούμε εάν delete query σε μια μεταβλητή τύπου String.
 * β. Δημιουργούμε από το αντικείμενο con ένα άλλο αντικείμενο(deleteStmt) τύπου  Statement
 * γ. Εκτελούμε το query χρησιμοποιώντας τη μέθοδο executeUpdate του αντικειμένου τύπου Statement
 * δ. Αποδεσμεύουμε τους πόρους του αντικειμένου τύπου Statement χρησιμοποιώντας την εντολή close
 * ε. Οι εντολές αυτές υποχρεωτικά πρέπει να βρίσκονται σε block κώδικα try...catch
 *    για να εντοπίσουμε ενδεχόμενο exception, το οποίο και εμφανίζουμε με την showMessageDialog
 *    για να δούμε τι σφάλμα προέκυψε.
 */
public void deleteMemberAction() {
    String query="DELETE MEMBER WHERE MNO=" + jMnoTxt.getText();

    System.out.println("Query:\t" + query);

    Statement deleteStatement;

    try {
        deleteStatement= DvdClubJFrame.con.createStatement();
        deleteStatement.executeUpdate(query);
        deleteStatement.close();
    }
    catch (SQLException ex) {
        showMessageDialog(this, ex.getMessage());
    }
}
```

Εικόνα 10.54 Κατασκευή συνάρτησης διαγραφής μέλους

10.17.6 Κατασκευή συνάρτησης αναζήτησης όλων των μελών από τη βάση δεδομένων

Η συνάρτηση loadMembers εκτελεί ένα select query και τοποθετεί όλα τα μέλη στο πλέγμα/πίνακα jMemberTable. Στις εικόνες 10.55 και 10.56 ακολουθεί αναλυτική επεξήγηση.

```
/*
 * Η συναρτηση αυτη αναζητα ολα τα μελη απο τον πινακα MEMBER της ΒΔ και τα τοποθετει
 * στο πλέγμα jMemberTable.
 *
 * Για να γινει η αναζητηση κανουμε τα εξης:
 * α. Δημιουργουμε ενα select query σε μια μεταβλητη τυπου String.
 * β. Δημιουργουμε απο το αντικειμενο con ενα αλλο αντικειμενο(searchStmt) τυπου  Statement
 *    --Προσοχη στις παραμετρους της μεθοδου createStatement.
 * γ. Εκτελουμε το query χρησιμοποιωντας τη μεθοδο executeQuery του αντικειμενου τυπου Statement,
 *    ενω η executeQuery μας επιστρεφει ολα τις εγγραφες που βρηκε σε ενα αλλο αντικειμενο(searchRS)
 *    τυπου ResultSet
 * δ. Καλουμε μια συναρτηση (clearTable) που εχουμε κατασκευασει εμεις για να αδειασουμε/καθαρισουμε τις γραμμες
 *    του πλεγματος.
 * ε. Χρησιμοποιουμε σε μια επαναληψη τη μεθοδο next του αντικειμενου τυπου ResultSet για
 *    να επεξεργαστουμε ενα προς ενα ολα τα records, και τα τοποθετουμε στο πλεγμα χρησιμοποιωντας
 *    τη μεθοδο addRow του δεδομενου μελους model του πλεγματος jMemberTable. Η μεθοδος addRow
 *    δεχεται σαν ορισμα εναν δυναμικο μονοδιαστατο πινακα τυπου Object. Καθε στοιχειο του πινακα,
 *    ειναι τοποθετειται σε ενα κελι της νεας γραμμης του πλεγματος. Η ακριβης συνταξη φαινεται στη
 *    συναρτηση.
 *    - Η μεθοδος next επιστρεφει true καθε φορα που προχωρα στο επομενο record, ενω αν αποτυχει
 *      επιστρεφει false.
 * στ. Προαιρετικα (μετα το γεμισμα του πλεγματος) χρησιμοποιουμε τη μεθοδο changeSelection του πλεγματος
 *    για να φωτισουμε-επιλεξουμε την 1η γραμμη του πλεγματος. Η πρωτη παραμετρος της changeSelection
 *    δινει τον αριθμο της γραμμης που θα φωτιστει-επιλεχτει, και ειναι αυτη που μας ενδιαφερει. Η
 *    αριθμηση των γραμμων στο πλεγμα ξεκινα απο τη θεση 0 (= πρωτη γραμμη).
 *    - Η μεθοδος first του αντικειμενου τυπου ResultSet μας πηγαινει στην πρωτη εγγραφη των records
 *      που υπαρχουν στο αντικειμενο searchRS. Αν υπαρχει τουλαχιστο ενα record επιστρεφει true, αλλιως false.
 * ζ. Αποδεσμευουμε τους πορους του αντικειμενου τυπου ResultSet χρησιμοποιωντας την εντολη close
 * η. Αποδεσμευουμε τους πορους του αντικειμενου τυπου Statement χρησιμοποιωντας την εντολη close
 * θ. Οι εντολες αυτες υποχρεωτικα πρεπει να βρισκονται σε block κωδικα try...catch
 *    για να εντοπισουμε ενδεχομενο exception, το οποιο και εμφανιζουμε με την showMessageDialog
 *    για να δουμε τι σφαλμα προεκυψε.
 */
```

Εικόνα 10.55 Συζήτηση-επεξήγηση της συνάρτησης αναζήτησης όλων των μελών

```
public void loadMembers() {
    String query="SELECT MNO,MLASTNAME,MFIRSTNAME,MADDRESS,MREGDATE,MDVDCOUNT "
          α     + " FROM MEMBER "
                + " ORDER BY MLASTNAME,MFIRSTNAME ";

    java.sql.Statement searchStmt;
    java.sql.ResultSet searchRS;

    try {
      β searchStmt = DvdClubJFrame.con.createStatement(java.sql.ResultSet.TYPE_SCROLL_INSENSITIVE,
                                                       java.sql.ResultSet.CONCUR_READ_ONLY);
      γ searchRS= searchStmt.executeQuery(query);

      δ clearTable(( javax.swing.table.DefaultTableModel )jMemberTable.getModel() );
        while (searchRS.next()) {
          (( javax.swing.table.DefaultTableModel )jMemberTable.getModel()).addRow(
                    new Object[] {  searchRS.getString("MNO"),
                                    searchRS.getString("MLASTNAME"),
                                    searchRS.getString("MFIRSTNAME"),
                                    searchRS.getString("MADDRESS"),
                                    searchRS.getString("MREGDATE"),
                                    searchRS.getString("MDVDCOUNT")
                                 }
                ) ;
        }
      ε
        if ( searchRS.first()) {                                  στ   table row index
            jMemberTable.changeSelection(0,0,false,false);
        }
        searchRS.close(); ζ
        searchStmt.close();
      η
    }
    catch (java.sql.SQLException s) {
       javax.swing.JOptionPane.showMessageDialog(this, s.getMessage());
    }
}
```

new record data

Εικόνα 10.56 Κατασκευή συνάρτησης αναζήτησης όλων των μελών

10.18 Προγραμματισμός ενεργειών-λειτουργικότητας της φόρμας διαχείρισης μελών

Εφόσον έχουμε κατασκευάσει τις βασικές συναρτήσεις της φόρμας πρέπει να προγραμματίσουμε την επικοινωνία-αλληλεπίδραση με τον χρήστη που θα τη χρησιμοποιήσει. Ο προγραμματισμός αυτός περιγράφεται στη συνέχεια.

10.18.1 Κατασκευή event handlers για τα κουμπιά Insert, Update, Delete.

Οι event handlers θα καλούν τις αντίστοιχες συναρτήσεις που κάνουν εισαγωγή, τροποποίηση και διαγραφή μέλους από τη βάση δεδομένων. Δείτε και την εικόνα 5.57.

```
    private void jInsertBtnActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        insertMemberAction();
    }

    private void jUpdateBtnActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        updateMemberAction();
    }

    private void jDeleteBtnActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        deleteMemberAction();
    }
```

Εικόνα 10.57 Οι event handlers για την κλήση των συναρτήσεις εισαγωγής, τροποποίησης και διαγραφής μέλους

10.18.2 Δημιουργία event handler εμφάνισης στη φόρμα του πλέγματος/πίνακα με τα μέλη.

Κατά το άνοιγμα της φόρμας θα πρέπει άμεσα να γεμίζει το πλέγμα/πίνακας με τα μέλη. Για να γίνει αυτό δημιουργούμε έναν event handler για το event **componentShown** της φόρμας και καλούμε τη μέθοδο loadMembers. Το event ComponentShown πυροδοτείται αυτόματα κάθε φορά που γίνεται ορατή μια φόρμα JFrame ή JDialog (συνήθως μετά την εκτέλεση κάποιας εντολής όπως η setVisible(true)).

```
    private void formComponentShown(java.awt.event.ComponentEvent evt) {

        // Φορτωσε το πλεγμα απο την αρχη με τα ολα τα μελη
        loadMembers();

    }
```

Εικόνα 10.58 Δημιουργία handler εμφάνισης των μελών στη φόρμα

10.18.3 Κατασκευή συνάρτησης που καθορίζει ποια κουμπιά θα είναι ενεργά ή όχι.

Ακολουθεί η κατασκευή συνάρτησης με όνομα setActionsStatus η οποία θα καθορίζει ποια κουμπιά θα είναι ενεργά ή όχι. Η συνάρτηση setActionStatus θα συμπεριφέρεται ως εξής:

Αν το πλαίσιο κειμένου κωδικού μέλους είναι κενό τότε τα κουμπιά εισαγωγής, τροποποίησης και διαγραφής θα είναι ανενεργά. Αν το πλαίσιο κωδικού μέλους έχει τιμή τότε σίγουρα θα είναι ενεργό το κουμπί διαγραφής, ενώ τα κουμπιά εισαγωγής και τροποποίησης θα είναι ενεργά μόνο αν και τα υπόλοιπα πλαίσια κειμένου έχουν τιμή. Το κουμπί Reset θα είναι ενεργό όταν έστω και ένα από τα πλαίσια κειμένου έχουν τιμή, αλλιώς θα είναι ανενεργό. Στην εικόνα 10.59 βλέπουμε τη συνάρτηση.

```
/*
* Καθορίζει ποια κουμπιά θα είναι ενεργά ή όχι με βάση τις τιμές που
* υπάρχουν στα πλαίσια κειμένου. Αρχικά απενεργοποιεί όλα τα κουμπιά
* και στη συνέχεια αποφασίζει ποια από αυτά θα ενεργοποιηθούν.
*/
public void setActionStatus() {
    jInsertBtn.setEnabled(false);
    jUpdateBtn.setEnabled(false);
    jDeleteBtn.setEnabled(false);
    jResetBtn.setEnabled(false);

    if (jMnoTxt.getText().length() > 0) {
        jDeleteBtn.setEnabled(true);
        if ((jLastnameTxt.getText().length() > 0)
                && (jFirstnameTxt.getText().length() > 0)
                && (jAddressTxt.getText().length() > 0)
                && (jRegistrationDateTxt.getText().length() > 0)) {
            jInsertBtn.setEnabled(true);
            jUpdateBtn.setEnabled(true);
        }
    }
    if ((jMnoTxt.getText().length() > 0)
            || (jLastnameTxt.getText().length() > 0)
            || (jFirstnameTxt.getText().length() > 0)
            || (jAddressTxt.getText().length() > 0)
            || (jRegistrationDateTxt.getText().length() > 0)) {
        jResetBtn.setEnabled(true);
    }

}
```

Εικόνα 10.59 Η συνάρτηση setActionStatus

Για να είναι λειτουργική η συνάρτηση setActionStatus θα πρέπει να καλείται και τις κατάλληλες στιγμές, δηλαδή κάθε φορά που ο χρήστης πληκτρολογεί κάτι σε οποιοδήποτε από τα πλαίσια κειμένου. Για το σκοπό αυτό δημιουργούμε έναν event handler για το event **caretUpdate** για όλα τα πλαίσια κειμένου. Δείτε εικόνα 10.60.

```
    private void jMnoTxtCaretUpdate(javax.swing.event.CaretEvent evt) {
        // TODO add your handling code here:
        setActionStatus();
    }

    private void jLastnameTxtCaretUpdate(javax.swing.event.CaretEvent evt) {
        // TODO add your handling code here:
        setActionStatus();
    }

    private void jFirstnameTxtCaretUpdate(javax.swing.event.CaretEvent evt) {
        // TODO add your handling code here:
        setActionStatus();
    }

    private void jAddressTxtCaretUpdate(javax.swing.event.CaretEvent evt) {
        // TODO add your handling code here:
        setActionStatus();
    }

    private void jRegistrationDateTxtCaretUpdate(javax.swing.event.CaretEvent evt) {
        // TODO add your handling code here:
        setActionStatus();
    }
```

Εικόνα 10.60 Δημιουργία event handler για όλα τα πλαίσια κειμένου

10.18.4 Εμφάνιση στοιχείων μέλους

Όταν ο χρήστης κάνει κλικ πάνω σε κάποιο μέλος (γραμμή του πλέγματος) τότε τα στοιχεία του πρέπει να τοποθετούνται στα αντίστοιχα πλαίσια κειμένου της φόρμας. Για να γίνει αυτό πρέπει να γράψουμε μια συνάρτηση (π.χ. με όνομα memberTableMouseClickedAction) η οποία διαβάζει τα στοιχεία της επιλεγμένης γραμμής του πλέγματος και τα βάζει στα πλαίσια κειμένου (εικόνα 10.61).

```
/*
 * Η συνάρτηση αυτή παίρνει τις τιμές από την επιλεγμένη γραμμή του πλέγματος και τις
 * τοποθετεί στα αντίστοιχα πλαίσια κειμένου. Για να θέσουμε τιμή σε ένα πλαίσιο κειμένου
 * χρησιμοποιούμε τη μέθοδο setText. Για να διαβάσουμε την τιμή ενός κελιού του πλέγματος
 * χρησιμοποιούμε τη μέθοδο getValueAt (που είναι μέθοδος του model του πίνακα, ενώ για να
 * βρούμε το model χρησιμοποιούμε τη μέθοδο getModel του πίνακα) όπου η πρώτη παράμετρος
 * είναι η γραμμή του πλέγματος και η δεύτερη παράμετρος είναι ο αριθμός στήλης της γραμμής
 * της πρώτης παραμέτρου.
 * Για να βρούμε την επιλεγμένη γραμμή που έχει κάνει κλικ ο χρήστης χρησιμοποιούμε την μέθοδο
 * getSelectedRow του JTable.
 * Επειδή η μέθοδος setText παίρνει όρισμα τύπου String, αλλά η μέθοδος getValueAt δεν επιστρέφει
 * String αλλά αντικείμενο τύπου Object χρησιμοποιούμε
 * το "" + object το οποίο μετατρέπει το object σε String.
 * Αν δεν υπάρχει επιλεγμένη γραμμή ( jMemberTable.getSelectedRow() < 0 ) τότε η συνάρτηση
 * memberTableMouseClickedAction αμέσως τερματίζεται.
 */
public void memberTableMouseClickedAction() {
    jMnoTxt.setText(""+jMemberTable.getModel().getValueAt(jMemberTable.getSelectedRow(), 0));
    jLastnameTxt.setText(""+jMemberTable.getModel().getValueAt(jMemberTable.getSelectedRow(), 1));
    jFirstnameTxt.setText(""+jMemberTable.getModel().getValueAt(jMemberTable.getSelectedRow(), 2));
    jAddressTxt.setText(""+jMemberTable.getModel().getValueAt(jMemberTable.getSelectedRow(), 3));
    jRegistrationDateTxt.setText(""+jMemberTable.getModel().getValueAt(jMemberTable.getSelectedRow(), 4));
}
```

Εικόνα 10.61 Συνάρτηση για την εμφάνιση των στοιχείων κάθε μέλους

Η συνάρτηση memberTableMouseClickedAction πρέπει να καλείται κάθε φορά που ο χρήστης κάνει κλικ σε κάποια γραμμή του πίνακα/πλέγματος. Για το λόγο αυτό κατασκευάζουμε έναν event handler για το event **mouseClicked** του αντικειμένου τύπου JTable. Δείτε εικόνα 10.62

```
private void jMemberTableMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    memberTableMouseClickedAction();

}
```

Εικόνα 10.62 event handler για το event mouseClicked του αντικειμένου τύπου JTable.

Επιπλέον, αν θέλουμε με το που γεμίζει το πλέγμα με τα μέλη να γεμίζουν ταυτόχρονα και τα πλαίσια κειμένου με την τρέχουσα επιλεγμένη γραμμή του πλέγματος (που είναι η πρώτη) τότε, τοποθετούμε μια κλήση της memberTableMouseClickedAction στο τέλος της συνάρτησης loadMembers.

10.18.5 Ενημέρωση του πλέγματος κατά την εισαγωγή, τροποποίηση, διαγραφή μέλους.

Κάθε φορά που εκτελείται μια εισαγωγή, τροποποίηση ή διαγραφή μέλους θα πρέπει αυτόματα να ενημερώνεται το περιεχόμενο του πλέγματος με τις αλλαγές. Για να γίνει αυτό προσθέτουμε στους αντιστοίχους event handlers μια κλήση της loadMembers. Δείτε εικόνα 10.63.

```
private void jInsertBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    insertMemberAction();
    loadMembers();
}

private void jUpdateBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    updateMemberAction();
    loadMembers();
}

private void jDeleteBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    deleteMemberAction();
    loadMembers();
}
```

Εικόνα 10.63 Ορισμοί handlers για την ενημέρωση του περιεχομένου του πλέγματος κατά την εισαγωγή, τροποποίηση, διαγραφή μέλους

10.18.6 Καθαρισμός των πλαισίων κειμένου.

Όταν ο χρήστης επιλέγει το κουμπί Reset τότε θα καθαρίζονται τα περιεχόμενα των πλαισίων κειμένου. Για να γίνει αυτό κατασκευάζουμε μια συνάρτηση (π.χ. με όνομα resetBtnAction) και κατασκευάζουμε έναν event handler που θα την καλεί. Δείτε τις εικόνες 10.64, 10.65.10.66.

```
/*
 * Η συνάρτηση αυτή καθαρίζει τα πλαίσια κειμένου.
 */
public void resetAction() {
    jMnoTxt.setText("");
    jLastnameTxt.setText("");
    jFirstnameTxt.setText("");
    jAddressTxt.setText("");
    jRegistrationDateTxt.setText("");
}
```

Εικόνα 10.64 Η συνάρτηση resetBtnAction

```
private void jResetBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    resetAction();
}
```

Εικόνα 10.65 event handler που καλεί τη συνάρτηση

```
/*
 *  Η συνάρτηση αυτή διαγράφει όλες τις εγγραφές ενός πλέγματος/πίνακα
 *  Δέχεται όρισμα το δεδομένο μέλος model το οποίο μας δίνει πρόσβαση
 *  στις εγγραφές του πλέγματος. Η διαγραφή μιας γραμμής γίνεται με
 *  την εντολή removeRow. Η συνάρτηση μας είναι προγραμματισμένη να
 *  διαγράφει όλες τις γραμμές μια προς μια ξεκινώντας από την τελευταία
 *  γραμμή προς την πρώτη.
 */
public void clearTable(DefaultTableModel model) {
    int numrows=model.getRowCount();
    for (int i=numrows -1; i>=0; i--) {
        model.removeRow(i);
    }
}
```

Εικόνα 10.66 Επεξήγηση της συνάρτησης clearTable.

10.19 Κλήση αποθηκευμένων διαδικασιών

Στο παράδειγμά μας θα επιχειρήσουμε να διαγράψουμε ένα μέλος ακόμα και αν υπάρχουν εξαρτώμενες γραμμές σε άλλους πίνακες. Αυτό μπορεί να συμβεί για παράδειγμα όταν θελήσουμε να διαγράψουμε ένα μέλος που έχει δανειστεί ταινίες. Μια αποθηκευμένη διαδικασία (MEMBERREMOVE) σε PL/SQL θα διαγράφει το μέλος ακόμα και αν συσχετίζεται με άλλους πίνακες. Η διαδικασία αυτή θα δέχεται παράμετρο εισόδου τον κωδικό μέλους και παράμετρο εξόδου το συνολικό ποσό που έχει πληρώσει.

Για να πραγματοποιηθεί η εν λόγω λειτουργία θα πρέπει να γράψουμε μια συνάρτηση java που θα καλεί την αποθηκευμένη διαδικασία, ενώ θα πρέπει να τροποποιήσουμε λίγο το περιβάλλον διεπαφής και τον προγραμματισμό ενεργειών.

10.19.1 Κατασκευή συνάρτησης java που καλεί αποθηκευμένη διαδικασία με παραμέτρους εισόδου εξόδου.

Στη συνέχεια παραθέτουμε στην εικόνα 10.67 την αποθηκευμένη διαδικασία MEMBERREMOVE.

/* Αποθηκευμένη διαδικασία διαγραφής μέλους αφού προηγουμένως έχει υπολογίσει πόσα έχει πληρώσει συνολικά το μέλος. Η διαδικασία έχει μια παράμετρο εισόδου (MEMBERCODE) και μια παράμετρο εξόδου (TOTALPAID). Για τον υπολογισμό του συνολικού ποσού του μέλους χρησιμοποιείται η αποθηκευμένη συνάρτηση MEMBERTOTALPAID. Για να γίνει η διαγραφή του member πρώτα διαγράφουμε από τους εξαρτημένους πίνακες όλες τις εγγραφές που σχετίζονται με το συγκεκριμένο μέλος */

```
create or replace procedure MEMBERREMOVE (      MEMBERCODE IN NUMBER,
TOTALPAID OUT NUMBER    )
IS

BEGIN
      TOTALPAID:= MEMBERTOTALPAID(MEMBERCODE);
      DELETE FROM PHONE WHERE MNO=MEMBERCODE;
      DELETE FROM BORROW WHERE MNO=MEMBERCODE;
      DELETE FROM MEMBER WHERE MNO=MEMBERCODE;
END;
/
```

Εικόνα 10.67 Η αποθηκευμένη διαδικασία MEMBERREMOVE σε PL/SQL η οποία διαγράφει ένα μέλος ακόμα και αν συσχετίζεται με άλλους πίνακες

Ακολουθεί η κλήση της αποθηκευμένης διαδικασίας (εικόνα 10.68).

```
/*
* Η συνάρτηση forceDeleteMemberAction δείχνει πως καλούμε μια αποθηκευμένη διαδικασία
* που περιέχει παραμέτρους εισόδου/εξόδου. Για να γίνει αυτό κάνουμε τα εξής:
* α. Δηλώνουμε τις απαραίτητες μεταβλητές java που αντιστοιχούν στις παραμέτρους
* της αποθηκευμένης διαδικασίας/συνάρτησης
* β. Δημιουργούμε ένα αντικείμενο τύπου CallableStatement χρησιμοποιώντας τη συνάρτηση
* prepareCall του αντικειμένου con με παράμετρο την εντολή call όπως φαίνεται παρακάτω
* γ. Θέτουμε τις τιμές των παραμέτρων εισόδου. Ανάλογα με τον τύπο δεδομένων της παραμέτρου
* δίνουμε και την κατάλληλη εντολή setXXXX.
* δ. Πριν την εκτέλεση της διαδικασίας πρέπει να δηλώσουμε πως αναμένουμε τιμές απο
* παραμέτρους εξόδου. Για κάθε παράμετρο εξόδου δίνουμε την εντολή registerOutParameter
* ε. Εκτελούμε την αποθηκευμένη διαδικασία
* στ. Τοποθετούμε σε μεταβλητές java τις επιστρεφόμενες τιμές με κλήση των συναρτήσεων getXXXX
* ανάλογα με τον τύπο.
* ζ. Αποδεσμεύουμε το αντικειμένο τύπου CallableStatement που περιέχει τους πόρους
* επικοινωνίας με την Oracle (DBMS). Η μη αποδέσμευση μπορεί να προκαλέσει σφάλματα κατά την
* εκτέλεση των προγραμμάτων. Δεν τα καθαρίζει ο garbage collector.
*/
```

```
public void forceDeleteMemberAction() {
    try {
        // α. Δήλωση των μεταβλητών κωδικός μέλους(MNO) και συνολικό ποσό(TOTALPAID)
        int MNO = java.lang.Integer.parseInt("" + jMnoTxt.getText());
        float TOTALPAID;
        // β. Δημιουργησε ένα αντικείμενο τύπου CallableStatement
        // Καθε ερωτηματικό αντιστοιχεί σε μία παράμετρο της αποθηκευμένης διαδικασίας/συνάρτησης
        java.sql.CallableStatement cstmt
                = DvdClubJFrame.con.prepareCall("{ call MEMBERREMOVE( ?,? ) }");
        // γ. Θέσε την τιμή της πρώτης παραμέτρου, που είναι εισόδου (κωδικος μέλους).
        cstmt.setInt(1, MNO);
        // δ. Δήλωσε ρητά πως η δεύτερη παράμετρος είναι παράμετρος εξόδου και πως έχει τυπο FLOAT
        cstmt.registerOutParameter(2, java.sql.Types.FLOAT);
        // ε. Εκτέλεσε την αποθηκευμένη διαδικασία
        cstmt.executeUpdate();
        // στ. Πάρε την επιστρεφόμενη τιμή από την παράμετρο εξόδου (συνολικό ποσό)
        TOTALPAID = cstmt.getFloat(2);
        // ζ. Αποδεσμεύουμε τους πόρους του αντικειμένου cstmt
        cstmt.close();

        //Εμφάνισε διαγνωστικό μήνυμα μέσα σε παράθυρο
        javax.swing.JOptionPane.showMessageDialog(this, "The removed member had paid: " + TOTALPAID);
    } catch (java.sql.SQLException e) {
        javax.swing.JOptionPane.showMessageDialog(this, e.getMessage());
    }
}
```

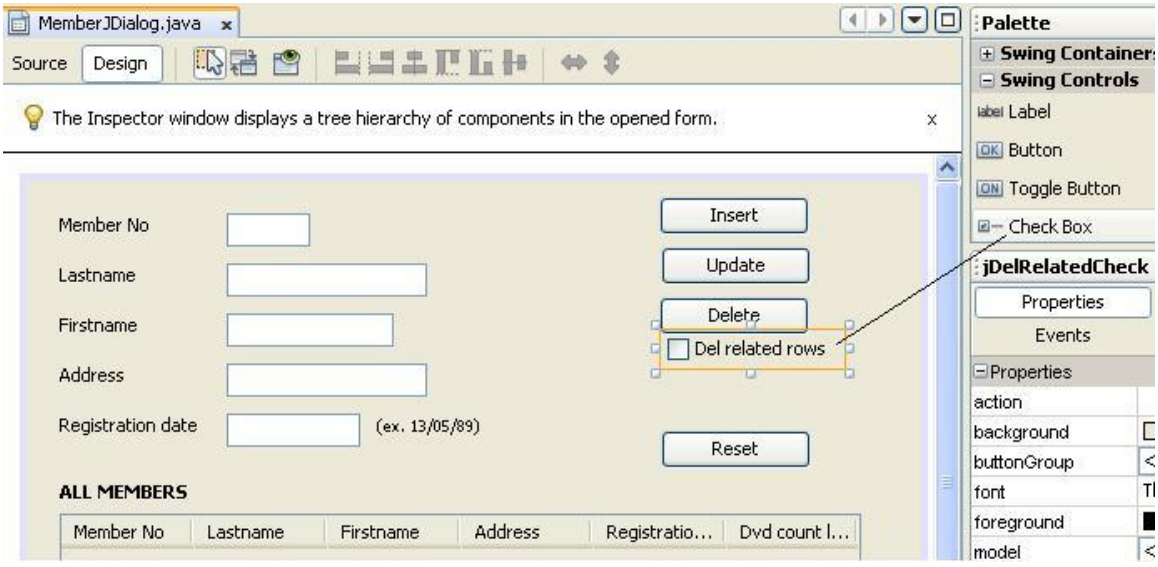
Εικόνα 10.68 Κλήση αποθηκευμένης διαδικασίας

Σημείωση

Η εντολή parseInt δείχνει πως μπορούμε να μετατρέψουμε ένα String σε ακεραιο.

10.19.2 Τροποποίηση του περιβάλλοντος διεπαφής και του προγραμματισμού ενεργειών

Επιλέγουμε ένα οπτικό αντικείμενο τύπου CheckBox από την κατηγορία Swing controls της παλέτας και το «ρίχνουμε» (drag & drop) πάνω στο πλαίσιο διαλόγου όπως φαίνεται στην εικόνα 10.68.



Εικόνα 10.69 Τροποποίηση του περιβάλλοντος διεπαφής

Δίνουμε στην ιδιότητα text του αντικειμένου την τιμή 'Del related rows' και αλλάζουμε το όνομα του αντικειμένου σε jDelRelatedCheck. Σκοπός μας είναι όταν ο χρήστης έχει «τσεκάρει» (επιλέξει) το check box να εκτελεστεί η συνάρτηση forceDeleteMemberAction. Για το να γίνει αυτό τροποποιούμε και τη συνάρτηση (εικόνα 10.70).

```
private void jDeleteBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // Αν δεν έχει τσεκαριστεί το check box καλεσε την deleteMemberAction,
    // αλλιώς κάλεσε την forceDeleteMemberAction
    if (jDelRelatedCheck.isSelected()==false)
        deleteMemberAction();
    else
        forceDeleteMemberAction();
    loadMembers();
}
```

Εικόνα 10.70 Όταν ο χρήστης έχει «τσεκάρει» το check box τότε εκτελείται η συνάρτηση forceDeleteMemberAction.

Κατά την εκτέλεση του προγράμματος αν ο χρήστης επιλέξει να διαγράψει ένα μέλος με «τσεκαρισμένη» την επιλογή (ενεργοποιημένη την επιλογή) 'Del related rows' τότε θα εμφανιστεί ένα μήνυμα όπως αυτό της εικόνας 10.71.

Εικόνα 10.71 Εμφάνιση μηνύματος

10.20 Κλήση αποθηκευμένων συναρτήσεων

Στο παράδειγμά μας θα εμφανίσουμε το συνολικό ποσό που έχει ξοδέψει ένα επιλεγμένο μέλος. Θα χρησιμοποιήσουμε την αποθηκευμένη συνάρτηση (MEMBERTOTALPAID) σε PL/SQL η οποία δέχεται παράμετρο εισόδου τον κωδικό μέλους και επιστρέφει το συνολικό ποσό που έχει πληρώσει. Για να πραγματοποιηθεί η εν λόγω λειτουργία θα πρέπει να γράψουμε μια συνάρτηση java που θα καλεί την αποθηκευμένη συνάρτηση, ενώ θα πρέπει να τροποποιήσουμε λίγο το περιβάλλον διεπαφής και τον προγραμματισμό ενεργειών.

10.20.1 Κατασκευή συνάρτησης java που καλεί αποθηκευμένη συνάρτηση με παραμέτρους εισόδου.

Στη συνέχεια παραθέτουμε στην εικόνα 10.72 την αποθηκευμένη συνάρτηση MEMBERTOTALPAID.

```
/*
       Συνάρτηση υπολογισμού του συνολικού κόστους που έχει πληρώσει το μέλος με

      κωδικό MEMBERCODE
       Η συνάρτηση χρησιμοποιεί δύο κέρσορες:
       - Ο 1ος βρίσκει το κόστος δανεισμού ενος dvd
       - Ο 2ος βρίσκει όλες τις εγγραφές δανεισμού dvd από τον πίνακα BORROW
*/
create or replace function MEMBERTOTALPAID(MEMBERCODE IN NUMBER)
return number
```

```
AS
        --Δήλωση μεταβλητής κωδικου DVD
        DVDCODE BORROW.DCODE%TYPE;
        --Δήλωση μεταβλητής ημερησιου κοστους DVD
        DVDDAYCOST DVD.DVDLENDCOST%TYPE := -1;

        --Δήλωση μεταβλητής κέρσορα για την ευρεση του ημερησιου κοστους ενοικιασης ενος DVD.
        -- Ο κερσορας εχει παραμετρο τη μεταβλητη DVDCODE
        --Το κοστος του dvd υπαρχει στη στηλη DVDLENDCOST του πινακα DVD
        cursor SEARCHDVDCOST_CURSOR is
                select DVDLENDCOST from DVD WHERE DCODE=DVDCODE;
        --Δήλωση μεταβλητής εγγραφής του παραπανω κερσορα
        SEARCHDVDCOST_REC SEARCHDVDCOST_CURSOR%ROWTYPE;


        --Δήλωση μεταβλητής κέρσορα ευρεσης ολων των δανεισμων που εχει κανει το μελος
        --Αναζητουνται οι εγγραφες οπου η INDATE δεν εχει τιμη NULL
        --(Δηλαδη ενδιαφερουν μονο οι ενοικιασεις που εχουν πληρωθει)
        cursor BORROWDATE_CURSOR is
                select DCODE, OUTDATE, INDATE from BORROW WHERE MNO=MEMBERCODE AND INDATE IS NOT
NULL;
        --Δήλωση μεταβλητής εγγραφής
        BORROWDATE_REC BORROWDATE_CURSOR%ROWTYPE;

        --Δήλωση μεταβλητης συνολικου κοστους που εχει πληρωσει το μελος
        TOTALCOST NUMBER(7,2) :=0;
BEGIN

        -- Βρες ολους τους δανεισμους που εχει κανει το μελος με κωδικο MEMBERCODE
        -- Για καθε δανεισμο βαλε τον κωδικο του dvd στη μεταβλητη DVDCODE
        OPEN BORROWDATE_CURSOR;
        FETCH BORROWDATE_CURSOR INTO BORROWDATE_REC;
        WHILE ( BORROWDATE_CURSOR%FOUND = TRUE ) LOOP
                DVDDAYCOST:=0;
                DVDCODE:= BORROWDATE_REC.DCODE;
                -- Για καθε dvd που εχει δανειστει το μελος βρες ποσο ειναι το ημερησιο κοστος του
                -- και βαλτο στη μεταβλητη DVDDAYCOST
                -- ! Προσεξε πως η μεταβλητη DVDCODE ειναι ταυτοχρονα παραμετρος στο select του
                -- κερσορα SEARCHDVDCOST_CURSOR
                OPEN SEARCHDVDCOST_CURSOR;
                FETCH SEARCHDVDCOST_CURSOR INTO SEARCHDVDCOST_REC;
                IF ( SEARCHDVDCOST_CURSOR%FOUND = TRUE ) THEN
                                DVDDAYCOST:= SEARCHDVDCOST_REC.DVDLENDCOST;
                END IF;
                CLOSE SEARCHDVDCOST_CURSOR;
                --Ενημερωσε το αθροιστη συνολικου κοστους που εχει πληρωσει το μελος
                --αφαιρωντας την ημερομηνια δανεισμου απο την ημερομηνια επιστροφης και
                -- πολλαπλασιαζοντας με το κοστος του τρεχοντος dvd
                TOTALCOST:= TOTALCOST + (round(BORROWDATE_REC.INDATE) -
round(BORROWDATE_REC.OUTDATE)) * DVDDAYCOST;

                FETCH BORROWDATE_CURSOR INTO BORROWDATE_REC;
        END LOOP;
        CLOSE BORROWDATE_CURSOR;

        -- Επεστρεψε το ποσο πληρωμης
        RETURN TOTALCOST;
END;
/
```

```
/*
        Συνάρτηση υπολογισμού του συνολικού κόστους που έχει πληρώσει το μέλος με

      κωδικό MEMBERCODE
        Η συνάρτηση χρησιμοποιεί δύο κέρσορες:
```

```
        - Ο 1ος βρίσκει το κόστος δανεισμού ενος dvd
        - Ο 2ος βρίσκει όλες τις εγγραφές δανεισμού dvd από τον πίνακα BORROW
*/
create or replace function MEMBERTOTALPAID(MEMBERCODE IN NUMBER)
return number
AS
        --Δήλωση μεταβλητής κωδικου DVD
        DVDCODE BORROW.DCODE%TYPE;
        --Δήλωση μεταβλητής ημερησιου κοστους DVD
        DVDDAYCOST DVD.DVDLENDCOST%TYPE := -1;

        --Δήλωση μεταβλητής κέρσορα για την ευρεση του ημερησιου κοστους ενοικιασης ενος DVD.
        -- Ο κερσορας εχει παραμετρο τη μεταβλητη DVDCODE
        --Το κοστος του dvd υπαρχει στη στηλη DVDLENDCOST του πινακα DVD
        cursor SEARCHDVDCOST_CURSOR is
                select DVDLENDCOST from DVD WHERE DCODE=DVDCODE;
        --Δήλωση μεταβλητής εγγραφής του παραπανω κερσορα
        SEARCHDVDCOST_REC SEARCHDVDCOST_CURSOR%ROWTYPE;


        --Δήλωση μεταβλητής κέρσορα ευρεσης ολων των δανεισμων που εχει κανει το μελος
        --Αναζητουνται οι εγγραφες οπου η INDATE δεν εχει τιμη NULL
        --(Δηλαδη ενδιαφερουν μονο οι ενοικιασεις που εχουν πληρωθει)
        cursor BORROWDATE_CURSOR is
                select DCODE, OUTDATE, INDATE from BORROW WHERE MNO=MEMBERCODE AND INDATE IS NOT
NULL;
        --Δήλωση μεταβλητής εγγραφής
        BORROWDATE_REC BORROWDATE_CURSOR%ROWTYPE;

        --Δήλωση μεταβλητης συνολικου κοστους που εχει πληρωσει το μελος
        TOTALCOST NUMBER(7,2) :=0;
BEGIN

        -- Βρες ολους τους δανεισμους που εχει κανει το μελος με κωδικο MEMBERCODE
        -- Για καθε δανεισμο βαλε τον κωδικο του dvd στη μεταβλητη DVDCODE
        OPEN BORROWDATE_CURSOR;
        FETCH BORROWDATE_CURSOR INTO BORROWDATE_REC;
        WHILE ( BORROWDATE_CURSOR%FOUND = TRUE ) LOOP
                DVDDAYCOST:=0;
                DVDCODE:= BORROWDATE_REC.DCODE;
                -- Για καθε dvd που εχει δανειστει το μελος βρες ποσο ειναι το ημερησιο κοστος του
                -- και βαλτο στη μεταβλητη DVDDAYCOST
                -- ! Προσεξε πως η μεταβλητη DVDCODE ειναι ταυτοχρονα παραμετρος στο select του
                -- κερσορα SEARCHDVDCOST_CURSOR
                OPEN SEARCHDVDCOST_CURSOR;
                FETCH SEARCHDVDCOST_CURSOR INTO SEARCHDVDCOST_REC;
                IF ( SEARCHDVDCOST_CURSOR%FOUND = TRUE ) THEN
                                DVDDAYCOST:= SEARCHDVDCOST_REC.DVDLENDCOST;
                END IF;
                CLOSE SEARCHDVDCOST_CURSOR;
                --Ενημερωσε το αθροιστη συνολικου κοστους που εχει πληρωσει το μελος
                --αφαιρωντας την ημερομηνια δανεισμου απο την ημερομηνια επιστροφης

            και
                -- πολλαπλασιαζοντας με το κοστος του τρεχοντος dvd
                TOTALCOST:= TOTALCOST + (round(BORROWDATE_REC.INDATE) -
round(BORROWDATE_REC.OUTDATE)) * DVDDAYCOST;

                FETCH BORROWDATE_CURSOR INTO BORROWDATE_REC;
        END LOOP;
        CLOSE BORROWDATE_CURSOR;

        -- Επεστρεψε το ποσο πληρωμης
        RETURN TOTALCOST;
END;
/
```

Εικόνα 10.72 Η αποθηκευμένη συνάρτηση MEMBERTOTALPAID σε PL/SQL η οποία υπολογίζει το συνολικό κόστος που έχει πληρώσει το μέλος με κωδικό MEMBERCODE.

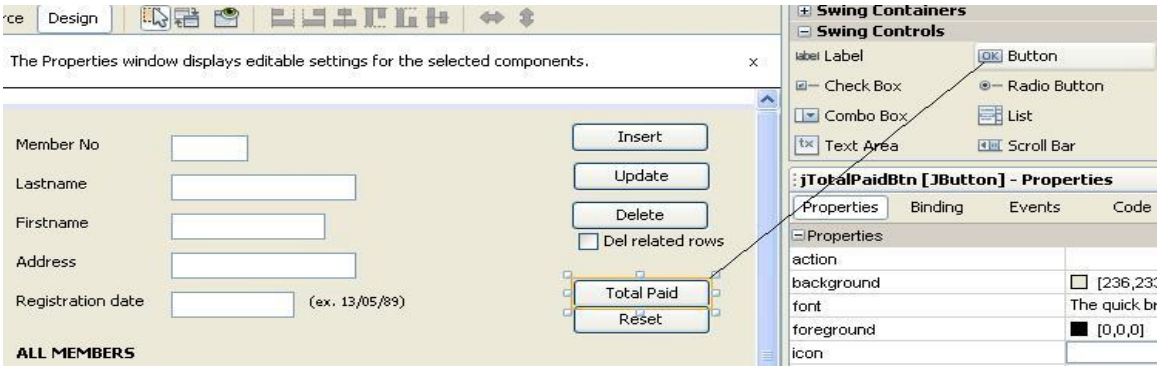
Ακολουθεί η κατασκευή της συνάρτησης η οποία καλεί την αποθηκευμένη συνάρτηση (εικόνα 10.73).

```
/*
* Η συναρτηση showTotalPaidAction δείχνει πώς καλούμε μια αποθηκευμένη συνάρτηση
* που περιέχει παραμέτρους εισόδου. Για να γίνει αυτό κάνουμε τα εξής:
* α. Δηλώνουμε μια μεταβλητη που αντιστοιχεί την τιμή επιστροφής. Επίσης δηλώνουμε
* τις απαραίτητες μεταβλητές java που αντιστοιχούν στις παραμέτρους (εισόδου/εξόδου)
* της αποθηκευμένης συνάρτησης
* β. Δημιουργούμε ένα αντικείμενο τύπου CallableStatement χρησιμοποιώντας τη συνάρτηση
* prepareCall του αντικειμενου con με παράμετρο την εντολή call οπως φαίνεται παρακάτω
* γ. Θέτουμε τις τιμές των παραμέτρων εισόδου. Ανάλογα με τον τύπο δεδομένων της παραμέτρου
* δίνουμε και την κατάλληλη εντολή setXXXX.
* δ. Πριν την εκτέλεση της συνάρτησης πρέπει να δηλώσουμε πως αναμένουμε τιμές από
* παραμέτρους εξόδου (αν υπάρχουν) αλλά οπωσδήποτε την τιμή επιστροφής. Για κάθε
* παράμετρο εξόδου ή τιμή επιστροφής δίνουμε την εντολή registerOutParameter
* ε. Εκτελούμε την αποθηκευμένη συνάρτηση
* στ. Τοποθετούμε σε μεταβλητές java την επιστρεφόμενη τιμή (και τις ενδεχόμενες τιμές
* των παραμέτρων εξόδου) με κληση των συναρτήσεων getXXXX ανάλογα με τον τύπο.
* ζ. Αποδεσμεύουμε το αντικείμνο τύπου CallableStatement που περιέχει τους πόρους
* επικοινωνίας με την Oracle (DBMS). Η μη αποδέσμευση μπορεί να προκαλέσει σφάλματα κατά την
* εκτέλεση των προγραμμάτων. Δεν τα καθαρίζει ο garbage collector.
*/

public void showTotalPaidAction() {
        try {
            // α. Δήλωση της μεταβλητής εισόδου (κωδικός μέλους-MNO) και της μεταβλητής τιμ'ης
            // επιστροφής (συνολικό ποσό-TOTALPAID)
            // Η parseInt εδώ δέχεται όρισμα string ακεραίων και το μετατρέπει σε ακέραιο.
            int MNO= java.lang.Integer.parseInt(""+jMnoTxt.getText());
            float TOTALPAID;
            // β. Δημιούργησε ένα αντικείμενο τύπου CallableStatement
            // Καθε ερωτηματικό αντιστοιχεί σε μία παράμετρο/ή τιμή επιστροφής της
            // αποθηκευμένης συνάρτησης
            java.sql.CallableStatement cstmt =
                    DvdClubJFrame.con.prepareCall( "{ ? = call MEMBERTOTALPAID( ? ) }");
            // γ. Θέσε την τιμή της παραμέτρου εισόδου που είναι το 2ο ερωτηματικό (?).
             cstmt.setInt(2,MNO );
            // δ. Δήλωσε μια (εσωτερική) παράμετρο που θα μπει η τιμή επιστροφής
            //     (Το 1ο ερωτηματικό αντιστοιχεί στην τιμη επιστροφής)
            //      Δήλωσε και τον τύπο της ( FLOAT )
             cstmt.registerOutParameter(1, java.sql.Types.FLOAT);
            // ε. Εκτέλεσε την αποθηκευμένη συνάρτηση
             cstmt.executeUpdate( );
             // στ. Πάρε την επιστρεφόμενη τιμή (συνολικό ποσό)
            TOTALPAID= cstmt.getFloat(1);
            // ζ. Αποδεσμεύουμε τους πόρους του αντικειμένου cstmt
            cstmt.close();

            //Εμφάνισε διαγνωστικό μήνυμα μέσα σε παράθυρο
            javax.swing.JOptionPane.showMessageDialog(this,"The member has paid: "+ TOTALPAID);
        }
        catch (java.sql.SQLException e) {
            javax.swing.JOptionPane.showMessageDialog(this,e.getMessage());
        }
    }
```

Εικόνα 10.73Η κατασκευή της συνάρτησης η οποία καλεί την αποθηκευμένη συνάρτηση

10.20.2 Τροποποίηση του περιβάλλοντος διεπαφής και του προγραμματισμού ενεργειών

Επιλέγουμε ένα οπτικό αντικείμενο τύπου Button από την κατηγορία Swing controls της παλέτας και το ρίχνουμε πάνω στο πλαίσιο διαλόγου όπως φαίνεται στην εικόνα 10.74.



Εικόνα 10.74 Προσθήκη οπτικού αντικειμένου τύπου Button

Δίνουμε στην ιδιότητα text του αντικειμένου την τιμή 'Total Paid' και αλλάζουμε το όνομα του αντικειμένου σε jTotalPaidBtn. Όταν ο χρήστης επιλέξει μέλος και πατήσει το κουμπί θα πρέπει να εμφανιστεί το συνολικό ποσό που έχει πληρώσει το μέλος. Στη εικόνα 10.75 φαίνεται ο event handler που είναι συνδεδεμένος με το jTotalPaidBtn.

```
    private void jTotalPaidBtnActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        showTotalPaidAction();
    }
```

Εικόνα 10.75 Ο event handler που είναι συνδεδεμένος με το jTotalPaidBtn

Κατά την εκτέλεση του προγράμματος αν ο χρήστης επιλέξει μέλος και πατήσει το κουμπί θα εμφανιστεί ένα μήνυμα όπως αυτό στην εικόνα 10.76.

Εικόνα 10.76 Εμφάνιση μηνύματος

10.21 Εγκατάσταση Java JDK,Oracle EXpress 18c και SQL Developer

Επισημαίνουμε ότι για να κατεβάσετε λογισμικό από την Oracle θα πρέπει να δημιουργήσετε δωρεάν λογαριασμό στην ιστοσελίδα της.

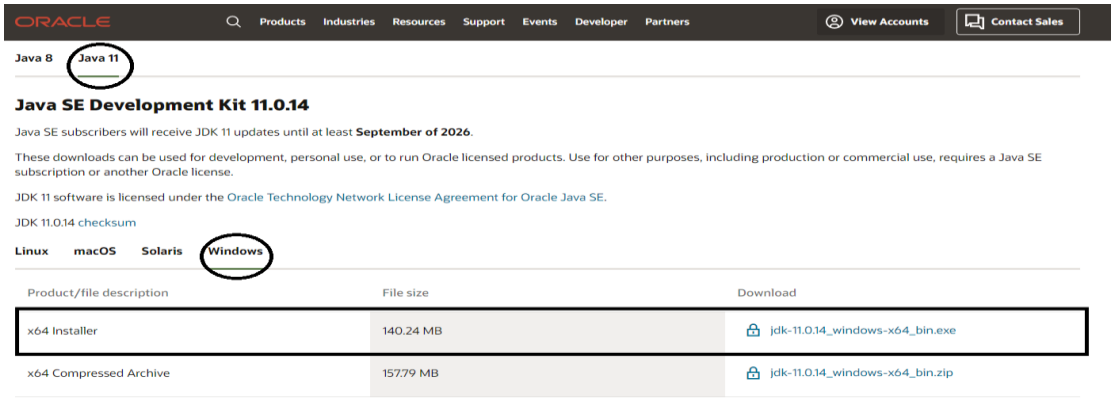
Αρχικά στον υπολογιστή μας δημιουργούμε το φάκελο TEMP μέσα στο partition C:\ .

10.21.1 Εγκατάσταση Java JDK

Σημείωση: Το βήμα αυτό πραγματοποιείται μόνον αν δεν έχουμε εγκατεστημένο Java JDK.

Κατεβάζουμε το java JDK (**Java SE 11**) από τη σελίδα (Εικόνα 10.77).

https://www.oracle.com/java/technologies/downloads/#java11-windows



*Εικόνα 10.77 java JDK (**Java SE 11**)*

Σημείωση: Για να κατεβάσουμε το αρχείο εγκατάστασης θα μας ζητηθεί να πληκτρολογήσουμε στοιχεία σύνδεσης στο site της Oracle. Αν δεν έχουμε λογαριασμό τότε μπορούμε να δημιουργήσουμε έναν λογαριασμό δωρεάν.

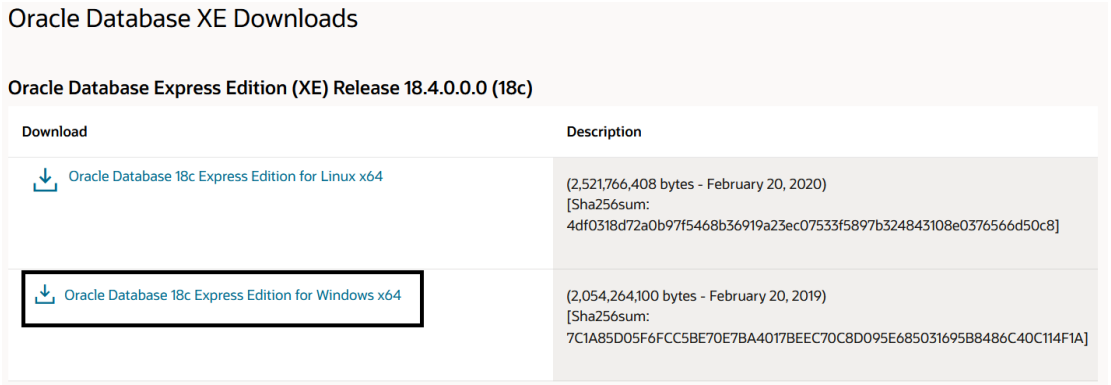
Το αρχείο το αποθηκεύουμε στο φάκελο C:\TEMP

Τέλος εκτελούμε το αρχείο που κατεβάσαμε και επιλέγουμε τις προεπιλεγμένες ρυθμίσεις κατά την εγκατάσταση.

10.21.2 Εγκατάσταση Oracle Database Express Edition

Κατεβάζουμε την Oracle Express Edition 18c από την παρακάτω διεύθυνση (Εικόνα 10.78).

https://www.oracle.com/database/technologies/xe18c-downloads.html



*Εικόνα 10.78 Oracle Express Edition **18c***

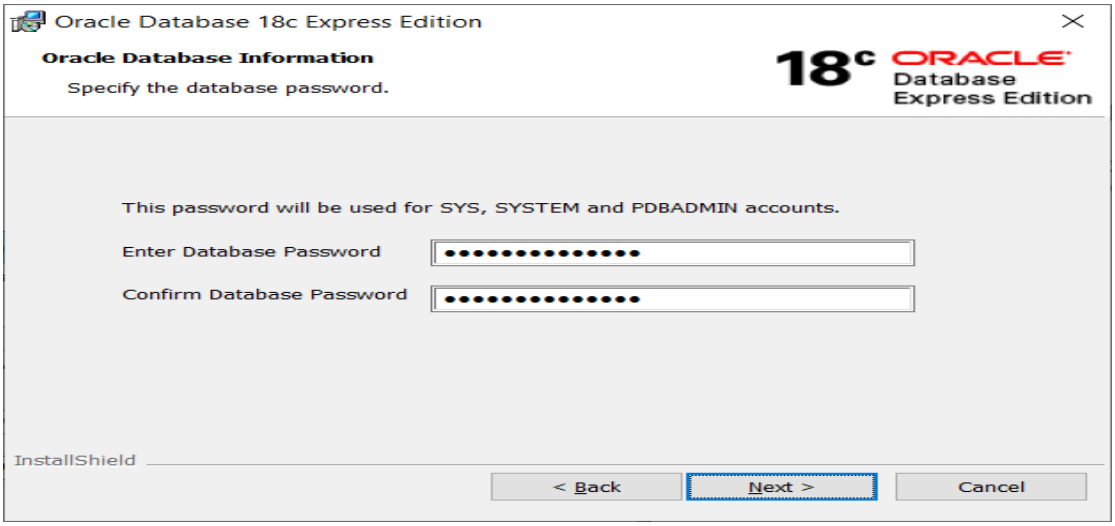
Σημείωση: Για να κατεβάσουμε το αρχείο εγκατάστασης θα μας ζητηθεί να πληκτρολογήσουμε στοιχεία σύνδεσης στο site της Oracle. Αν δεν έχουμε λογαριασμό τότε μπορούμε να δημιουργήσουμε έναν λογαριασμό δωρεάν.

- Αποθηκεύουμε το αρχείο που κατεβάσαμε στο φάκελο C:\TEMP και το αποσυμπιέζουμε.

Εκτελούμε το αρχείο που βρίσκεται στη διαδρομή **.\setup.exe**.

Κάποια στιγμή κατά την εγκατάσταση θα εμφανιστεί το παρακάτω παράθυρο για να δώσουμε τον ίδιο κωδικό στους χρήστες –**SYS, SYSTEM PDBADMIN**- που είναι διαχειριστές του Oracle DataBase Management System (DBMS) (Εικόνα 10.79).

Φυλάξτε τον κωδικό για να μην τον ξεχάσετε.



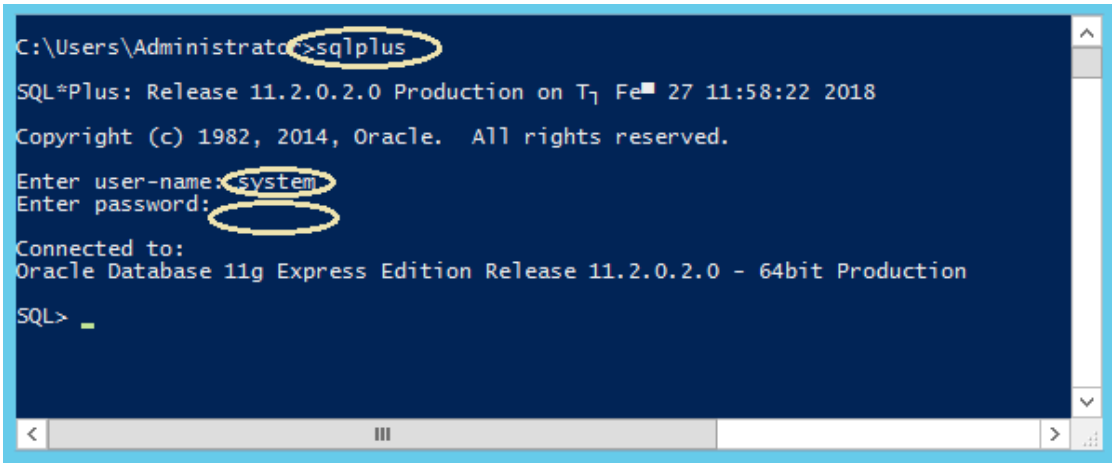
Εικόνα 10.79 Κωδικοί και συνθηματικά

Η εγκατάσταση θα ολοκληρωθεί σε λίγα λεπτά ανάλογα με τις προδιαγραφές του Η/Υ σας.

Προσοχή!

Αν στον υπολογιστή σας έχετε εγκατεστημένο firewall (τείχος προστασίας) και ερωτηθείτε για πρόσβαση των υπηρεσιών σχετικών με το setup της oracle θα επιλέξετε «ΑΠΟΔΟΧΗ».

Στη συνέχεια (μετά την ολοκλήρωση της εγκατάστασης) από τα Windows ανοίγουμε ένα Command Prompt (Γραμμή Εντολών) και εκτελούμε το πρόγραμμα **sqlplus** (Εικόνα 10.80)



*Εικόνα 10.80 Το πρόγραμμα **sqlplus***

Σημείωση: Δίνουμε ως user-name: **system** και password αυτό που βάλαμε προηγουμένως. Όταν πληκτρολογήσουμε τον κωδικό -δεν φαίνεται τι γράφουμε- πατάμε το πλήκτρο Enter.

Για να μπορούμε να δημιουργήσουμε πίνακες της βάσης και να εκτελέσουμε δηλώσεις SQL πρέπει να δημιουργήσουμε ένα χρήστη με τα κατάλληλα δικαιώματα. Θα δημιουργήσουμε (ενδεικτικά) τον χρήστη C##SCOTT με κωδικό TIGER (Εικόνα 10.81).

```
SQL> create user c##scott identified by tiger;

User created.

SQL> grant connect,resource,create view to c##scott;

Grant succeeded.
```

```
SQL> alter user c##scott quota unlimited on users;

User altered.
```

Εικόνα 10.81 Δημιουργία χρήστη C##SCOTT με κωδικό TIGER

Δίνουμε την εντολή SQL>**exit** και βγαίνουμε από τη συνιστώσα sqlplus.

10.21.3 Εγκατάσταση SQL Developer

Κατεβάζουμε τον SQL Developer από τη σελίδα (Εικόνα 10.82)

https://www.oracle.com/tools/downloads/sqldev-downloads.html

SQL Developer 21.4.2 Downloads

Version 21.4.2.018.1706 - January 19, 2022

- Release Notes
- Documentation

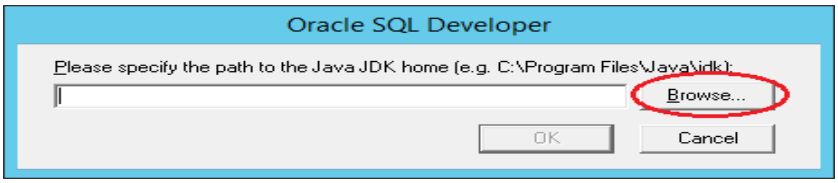
Platform	Download	Notes
Windows 64-bit with JDK 8 included	Download (436 MB)	• MD5: 11a72a967b1c9bebe0a7769d85aff7e1 • SHA1: 42d8defbbd0b0d6a4a2d78a684783f9144236a2e • Installation Notes
Windows 32-bit/64-bit	Download (448 MB)	• MD5: 02baeabd99d8fb3529162cb1bee1db5f • SHA1: 62e35d3541877c0f762835b98fa6a98fef022200 • Installation Notes • JDK 8 or 11 required

Εικόνα 10.82 SQL Developer

Αποθηκεύουμε το αρχείο που κατεβάσαμε στο φάκελο C:\TEMP και το αποσυμπιέζουμε.

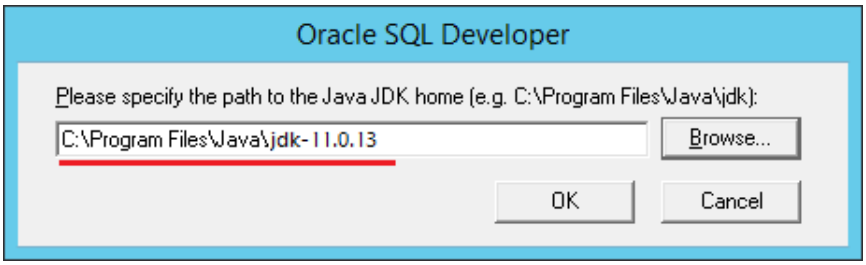
Μεταφέρουμε το φάκελο C:\TEMP\sqldeveloper στη διαδρομή c:\sqldeveloper\ και εκτελούμε το αρχείο c:\sqldeveloper**sqldeveloper**.exe

Στο παράθυρο που θα εμφανιστεί επιλέγουμε Browse και βρίσκουμε τη διαδρομή εγκατάστασης του Java JDK (Εικόνα 10.83).



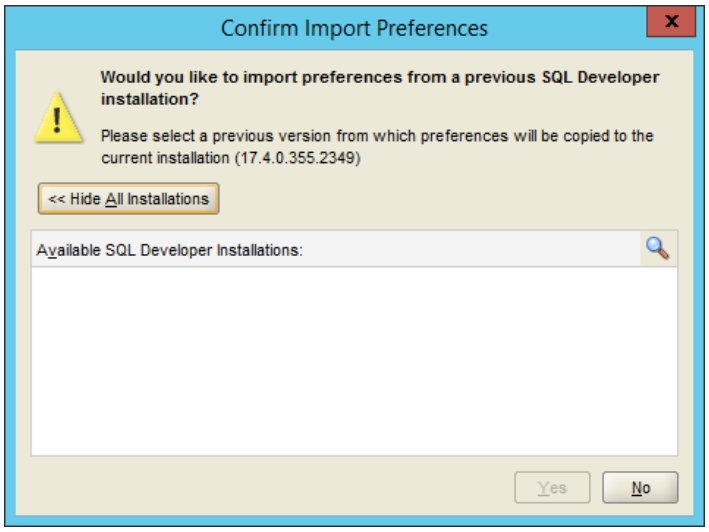
Εικόνα 10.83 Πρέπει να ορίσουμε τη διαδρομή εγκατάστασης του Java JDK

Η παρακάτω διαδρομή (Εικόνα 10.84) είναι ενδεικτική. Θα επιλέξετε τη διαδρομή της έκδοσης JDK που έχετε εγκαταστήσει.



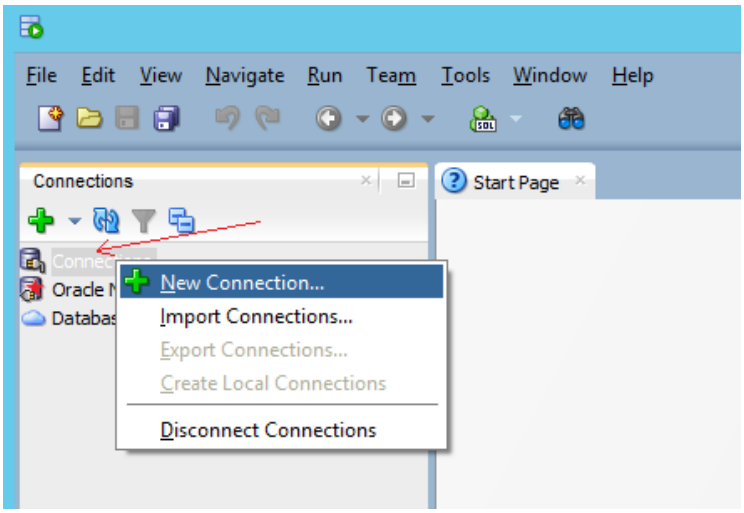
Εικόνα 10.84 Η διαδρομή εγκατάστασης του Java JDK

Αν εμφανιστεί το παρακάτω παράθυρο (Εικόνα 10.85) επιλέγουμε **No**.



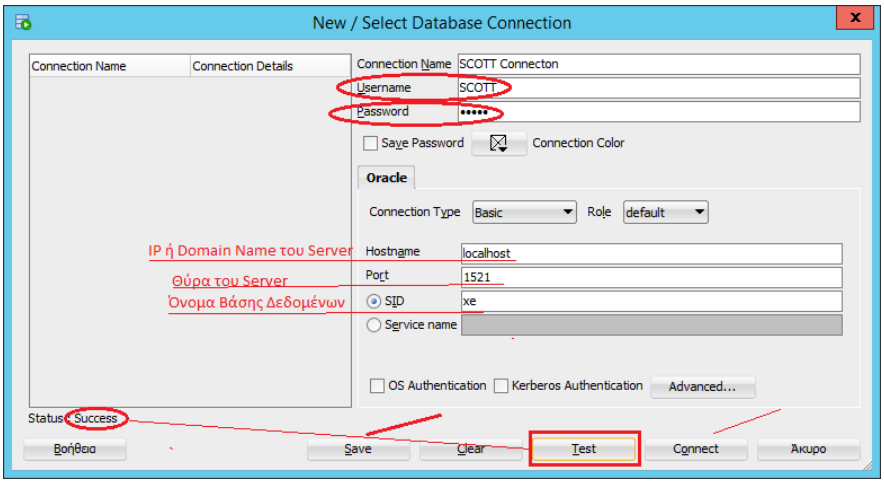
Εικόνα 10.85 Αν κατά την εγκατάσταση εμφανιστεί αυτό το παράθυρο επιλέγουμε No.

Όταν ανοίξει ο SQL Developer κάνουμε δεξί κλικ στο εικονίδιο Connections και επιλέγουμε New Connection όπως φαίνεται στην Εικόνα 10.86.



Εικόνα 10.86 Επιλέγουμε New Connection

Όπως δείχνει η Εικόνα 10.87 δίνουμε στο Username/Password τα στοιχεία π.χ C##SCOTT/TIGER και επιλέγουμε το κουμπί **Test**. Αν εμφανιστεί στο Status Success τότε είμαστε εντάξει. Στη συνέχεια επιλέγουμε **Save** και **Connect**. Η εικόνα είναι ενδεικτική: όπου scott βάζετε username του χρήστη που δημιουργήσατε με την εντολή create user (π.χ. c##scott)



Εικόνα 10.87 Πληκτρολόγηση συνθηματικών (credentials)

Προσοχή!

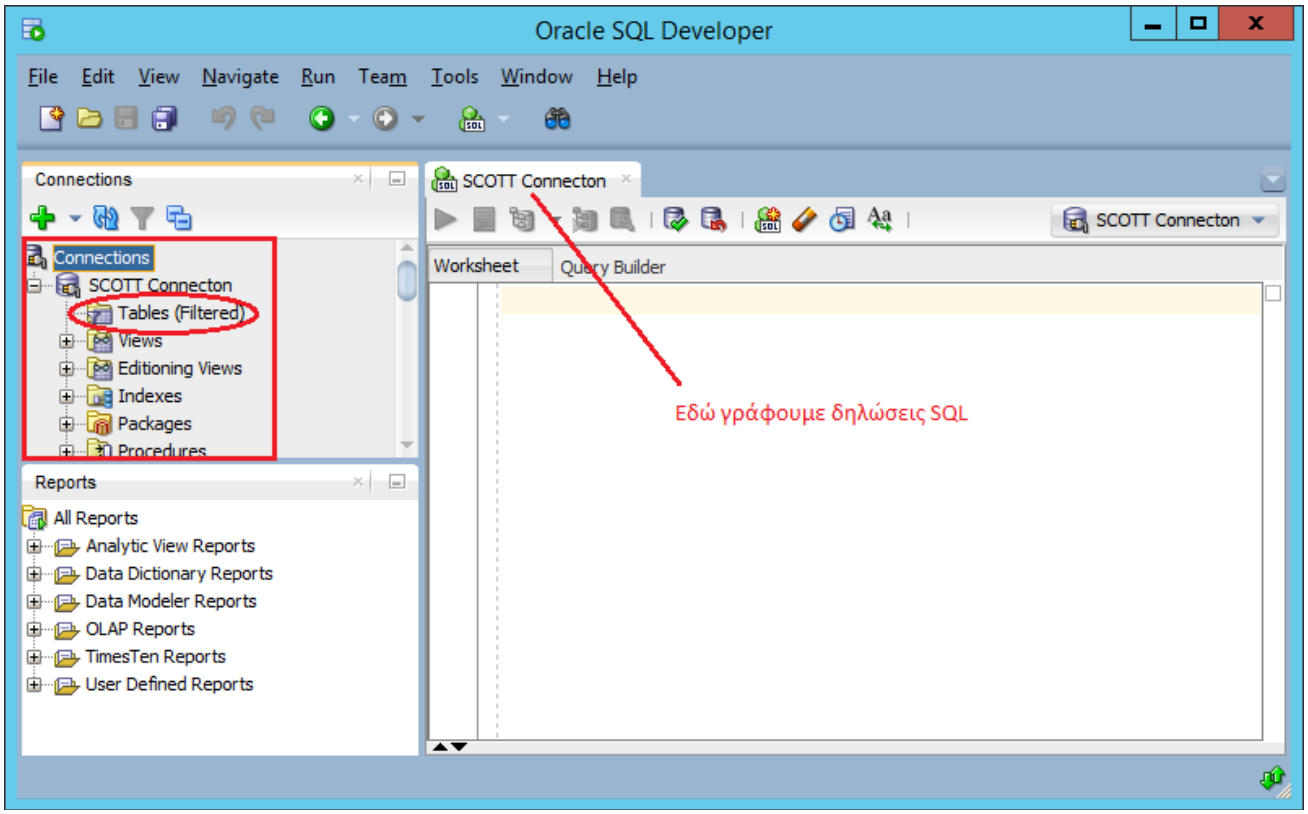
Υπάρχει ενδεχόμενο να μην μπορέσετε να συνδεθείτε λόγω εσφαλμένων ρυθμίσεων σύνδεσης που έχει επιλέξει το πρόγραμμα εγκατάστασης (setup) της Oracle. Σε αυτή την περίπτωση πρέπει να εντοπίσετε το φάκελο που έχει εγκατασταθεί η Oracle στα Windows,π.χ., C:\app\vtsouk\product\18.0.0**dbhomeXE**\. Μόλις βρεθεί ο φάκελος θα πρέπει να ξεναγηθείτε στον υποφάκελο .**network\admin**\ να ανοίξετε τα αρχεία **listener.ora** και **tnsnames.ora** και να τροποποιηθεί η IP διεύθυνση που υπάρχει σε αυτά στη διεύθυνση 127.0.0.1, δηλαδή HOST=127.0.0.1. Μόλις τροποποιηθούν τα αρχεία πρέπει να κάνετε επανεκκίνηση τον Η/Υ σας. Ενδεικτικά δείτε τις ρυθμίσεις στην Εικόνα 10.88 στα δυο αρχεία:

```
# listener.ora Network Configuration File: C:\app\dell\product\18.0.0\dbhomeXE\NETWORK\ADMIN\listener.ora
# Generated by Oracle configuration tools.

DEFAULT_SERVICE_LISTENER = XE

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = CLRExtProc)
      (ORACLE_HOME = C:\app\dell\product\18.0.0\dbhomeXE)
      (PROGRAM = extproc)
      (ENVS = "EXTPROC_DLLS=ONLY:C:\app\dell\product\18.0.0\dbhomeXE\bin\oraclr18.dll")
    )
  )

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP) (HOST = 127.0.0.1) (PORT = 1521))
      (ADDRESS = (PROTOCOL = IPC) (KEY = EXTPROC1521))
    )
  )
```

```
# tnsnames.ora Network Configuration File: C:\app\dell\product\18.0.0\dbhomeXE\NETWORK\ADMIN\tnsnames.ora
# Generated by Oracle configuration tools.

XE =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (HOST = 127.0.0.1) (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = XE)
    )
  )

LISTENER_XE =
  (ADDRESS = (PROTOCOL = TCP) (HOST = 127.0.0.1) (PORT = 1521))

ORACLR_CONNECTION_DATA =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = IPC) (KEY = EXTPROC1521))
    )
    (CONNECT_DATA =
      (SID = CLRExtProc)
      (PRESENTATION = RO)
    )
  )
```

Εικόνα 10.88 Ρυθμίσεις

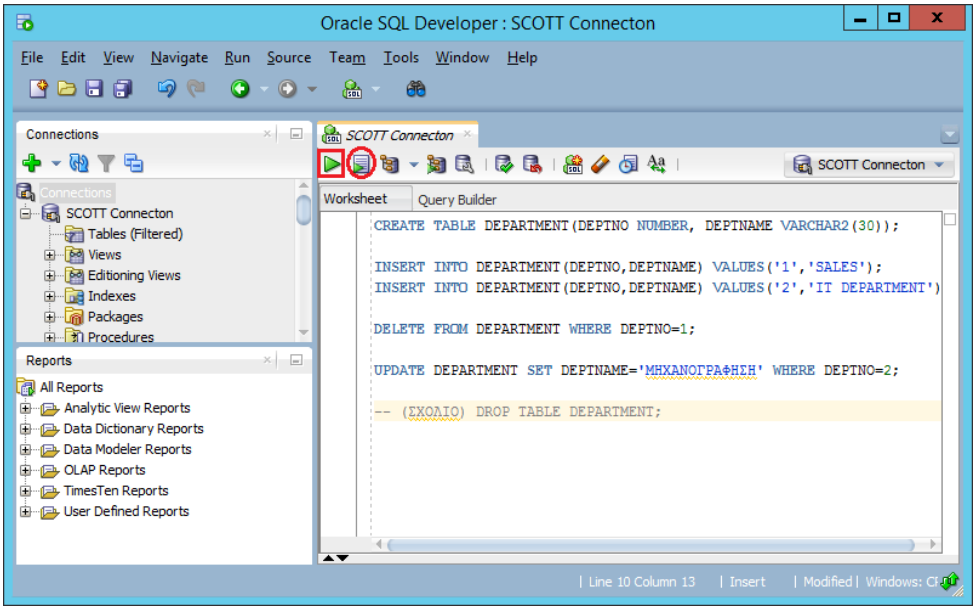
Όταν συνδεθούμε με επιτυχία εμφανίζονται τα εξής (Εικόνα 10.89):



Εικόνα 10.89 Επιτυχής σύνδεση

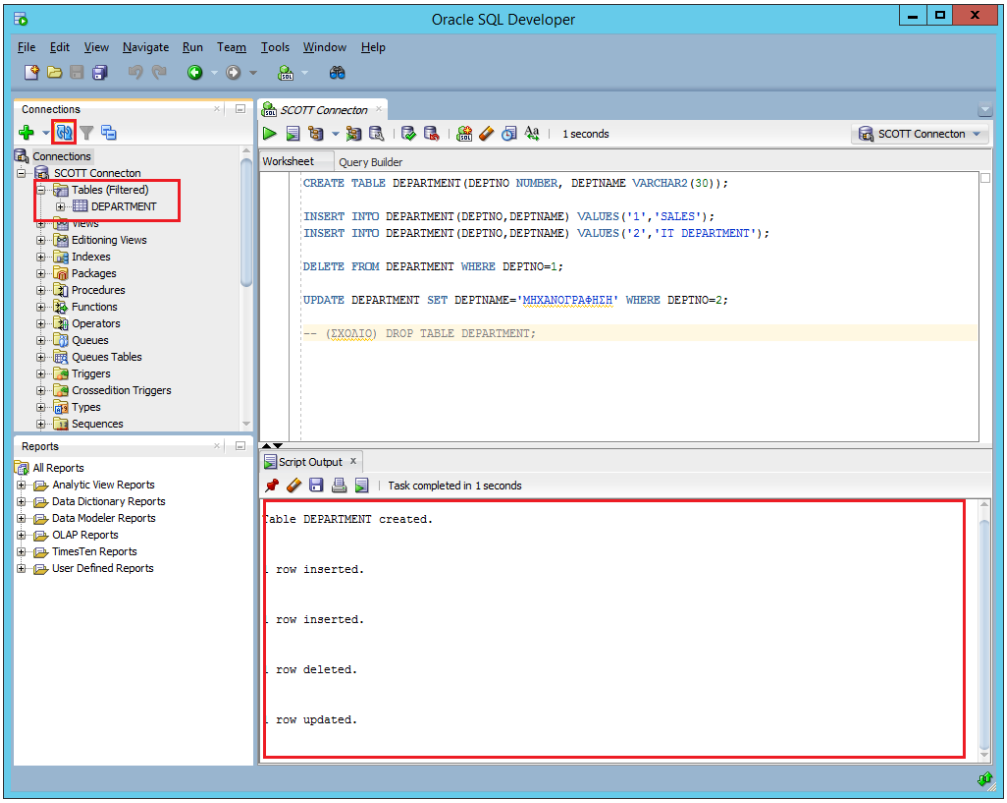
Αριστερά εμφανίζονται όλες οι πληροφορίες της βάσης δεδομένων του c##scott. Στον μικρό κύκλο θα εμφανιστούν οι πίνακες όταν τους δημιουργήσουμε. Δεξιά στο λευκό φόντο γράφουμε τις δηλώσεις SQL.

Στην Εικόνα 10.90 γράψαμε κάποιες δηλώσεις SQL: για να τις εκτελέσουμε όλες μαζί επιλέγουμε το εικονίδιο στον μικρό κόκκινο κύκλο. Αν θέλουμε να εκτελεστεί μόνο μια δήλωση επιλέγουμε το κόκκινο τετράγωνο.



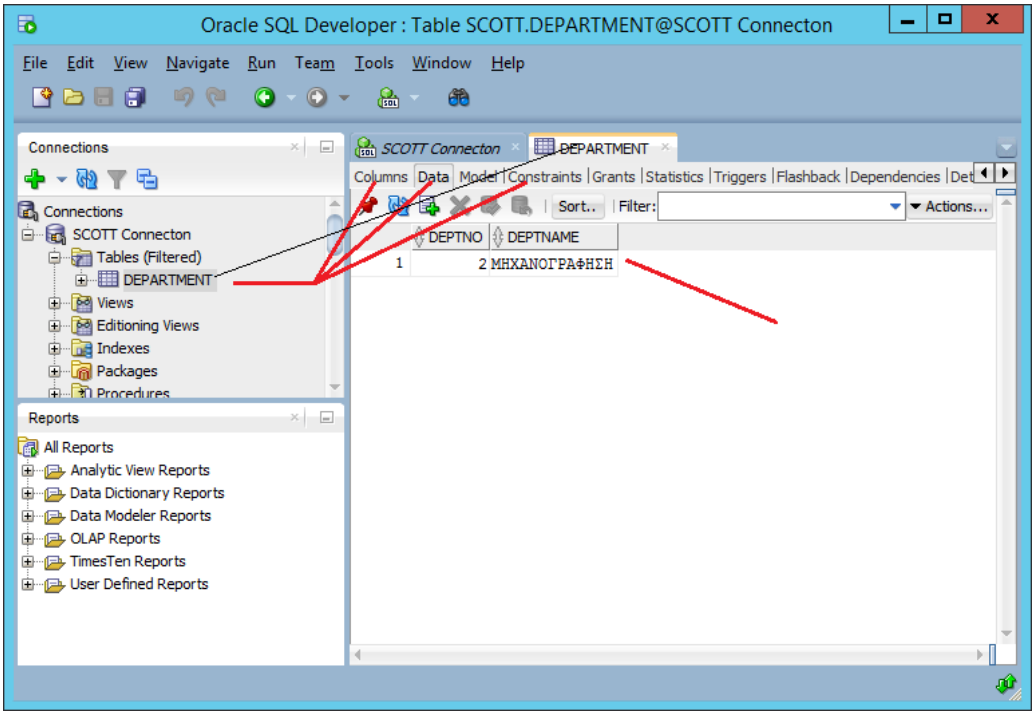
Εικόνα 10.90 Παράδειγμα δηλώσεων σε περιβάλλον SQL Developer

Η εκτέλεση των παραπάνω δηλώσεων SQL έχει τα παρακάτω αποτελέσματα (Εικόνα 10.91):



Εικόνα 10.91 Εκτέλεση δηλώσεων

Επιλέγοντας τον πίνακα DEPARTMENT μπορούμε να δούμε τις στήλες (Columns), τα δεδομένα (Data) και τους περιορισμούς (Constraints) που έχουμε δημιουργήσει (ορίσει) (Εικόνα 10.92).



Εικόνα 10.92 Πως βλέπουμε στοιχεία πίνακα στο περιβάλλον SQL Developer

10.21.4 Δ. Αδυναμία εγκατάστασης των εργαλείων

Αν παρά τις προσπάθειες που έχετε κάνει –και εφόσον έχετε εξαντλήσει κάθε ενδεχόμενο- δεν καταφέρετε να ολοκληρώσετε την εγκατάσταση όλων των εργαλείων με επιτυχία τότε μπορείτε να εφαρμόσετε τον εξής εναλλακτικό τρόπο για να τρέχετε τις εντολές των εργασιών του μαθήματος

Δημιουργήστε έναν λογαριασμό στην oracle (αν δεν έχετε ήδη) και χρησιμοποιήστε το online εργαλείο της oracle.

```
https://livesql.oracle.com
```

Επιλέγετε την καρτέλα **Introduction to SQL** και από εκεί θα ανοίξετε την ειδική σελίδα για να μπορείτε να δοκιμάζετε τις δηλώσεις

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

Σύνοψη
Στο κεφάλαιο αυτό περιγράφονται σημαντικές έννοιες και εργαλεία του προγραμματισμού Web εφαρμογών βάσεων δεδομένων για χρήση σε δυναμικούς ιστότοπους στο διαδίκτυο. Επεξηγούνται τεχνολογίες HTML και PHP. Παρατίθενται εφαρμογές βάσεων δεδομένων με χρήση PHP και διεπαφής προγραμματισμού εφαρμογών API. Το κεφάλαιο περιλαμβάνει τις παρακάτω ενότητες και είναι καλό να μελετηθούν και να δοκιμαστούν και σε υπολογιστή:

- *Εισαγωγή στις τεχνολογίες HTML και PHP*

  Παρουσιάζονται οι έννοιες στατικές και δυναμικές ιστοσελίδες, διακομιστής εξυπηρέτησης ιστοσελίδων (Web Server), στοίβα διακομιστή ιστού (web server stack) και γίνεται αναφορά σε γνωστές στοίβες, και στο τι περιλαμβάνει ένας εταιρικός Server. Παρουσιάζεται η δομή των σελίδων HTML και γίνεται αναφορά στις HTML ετικέτες (tags) ανά κατηγορία: Βασικές, μορφοποίησης (Formatting), ορισμού φόρμας, εικόνων (Images), Audio/Video, συνδέσμων (Links), ετικέτες για λίστες (Lists), ετικέτες για την κατασκευή πινάκων (Tables). Δίνονται παραδείγματα χρήσης ετικετών. Γίνεται εισαγωγή στην PHP και σε θέματα όπως σταθερές και μεταβλητές, arrays, δυναμική δημιουργία πίνακα (table) κ.λπ. Παρουσιάζεται η σχέση-συνεργασία HTML, PHP και παρατίθενται απλά προγράμματα. Στα παραδείγματα της ενότητας χρησιμοποιείται εγκατάσταση του XAMPP.

- *Δημιουργία ιστοτόπου Ted Codd Fun Club με χρήση HTML και του εργαλείο Netbeans.IDE*

  Περιγράφεται η δημιουργία ενός ιστότοπου για το διάσημο ερευνητή Edgar Frank "Ted" Codd με χρήση του εργαλείου Netbeans. Χρησιμοποιείται ένα μενού πλοήγησης (navigation menu) που συνδέεται με την κλήση σελίδων-επιλογών: Home, Ted Codd's quotes, Contact Us, Photos, Videos. Στόχος, είναι η εμπέδωση των εννοιών και του προγραμματισμού σελίδων με χρήση HTML.

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

  Παρουσιάζεται η έννοια των διεπαφών προγραμματισμού εφαρμογών API (Application Programming Interface) που μπορεί να είναι διαδικαστικές (procedural) ή αντικειμενοστρεφείς (object-oriented) και παρέχουν αναλόγως τις κλάσεις (classes), μεθόδους (methods), συναρτήσεις (functions) και μεταβλητές (variables) που είναι απαραίτητες για τον προγραμματισμό PHP εφαρμογών βάσεων δεδομένων.

Παρουσιάζονται οι PHP εφαρμογές που χρειάζονται επικοινωνία με βάσεις δεδομένων και οι απαραίτητες διεπαφές API που παρέχονται μέσω PHP επεκτάσεων (PHP extensions): MySQL API, mysqli (MySQL improved) API, PHP Data Objects (PDO) API. Γίνεται συζήτηση των επεκτάσεων και αναφορά στο ρόλο των προγραμμάτων οδήγησης (driver) και τη σημασία τους για το λογισμικό εφαρμογής. Τέλος, παρατίθεται η κατασκευή Web εφαρμογής που περιλαμβάνει: Δημιουργία βάσης δεδομένων και πινάκων, διαχείριση χρηστών, σύνδεση στη βάση (Making a connection) με PHP, έλεγχο αν έχουμε σφάλμα σύνδεσης και διαχείριση επιτυχούς σύνδεσης, εισαγωγή στοιχείων (Inserting data with PHP), διαχείριση δηλώσεων SQL, INSERT/UPDATE/DELETE αλλά και CREATE TABLE κ.λπ. με χρήση HTML, PHP και MySQL. Επισημαίνονται τα δύο διαφορετικά είδη διαχείρισης ανάλογα με τη δήλωση SQL: δηλώσεις INSERT/UPDATE/DELETE/CREATE κ.λπ. και δηλώσεις SELECT.

Προαπαιτούμενη γνώση
Προτείνεται η μελέτη των κεφαλαίων 1 και 3 του παρόντος συγγράμματος.

11.1 Εισαγωγή στην τεχνολογία-γλώσσα σήμανσης υπερκειμένου HTML (HyperText Markup Language)

Η γλώσσα HTML (HyperText Markup Language), γλώσσα σήμανσης υπερκειμένου, παρέχει τη δυνατότητα συγγραφής του κώδικα που χρησιμοποιείται για τη δομή μιας ιστοσελίδας και τη μορφοποίηση του περιεχομένου της. Για παράδειγμα, το περιεχόμενο θα μπορούσε να δομηθεί μέσα σε ένα σύνολο παραγράφων, μια λίστα σημείων με κουκκίδες (bullets) και να περιλαμβάνει εικόνες, πίνακες δεδομένων και συνδέσμους (links) σε άλλες ιστοσελίδες. Αρχικά θα γίνει μια εισαγωγή στην τεχνολογία HTML με απλά παραδείγματα για αναγνώστες που πρώτη φορά μελετούν το αντικείμενο.

Για να κατασκευάσουμε στατικές ή δυναμικές ιστοσελίδες χρειαζόμαστε μία «στοίβα διακομιστή ιστού» (web server stack), δηλαδή μια σουίτα λογισμικού που μετατρέπει τον υπολογιστή σε ένα διακομιστή εξυπηρέτησης ιστοσελίδων.

Συνήθως για να κατασκευάσουμε Web Server εγκαθιστούμε σε μηχανήματά μας τουλάχιστον τις συνιστώσες που περιλαμβάνονται στον πίνακα 11.1.

Πίνακας 11.1 Συνιστώσες μίας «στοίβας διακομιστή ιστού» (web server stack)

Λογισμικό	Λειτουργία
Web Server	Συνήθως HTTP server για τη διαχείριση των ιστοσελίδων
Database Server	Διαχείριση συναλλαγών (transactions) και διαχείριση δεδομένων σε συνεργασία με το ΣΔΒΔ
Scripting Language	Κατασκευή “scripts” για τη διαχείριση δεδομένων, τη σύνδεση στη βάση δεδομένων κ.λπ.

Γνωστές «στοίβες διακομιστή ιστού» (web server stack) βλέπουμε στον πίνακα 11.2.

Πίνακας 11.2 Γνωστές «στοίβες διακομιστή ιστού» (web server stack)

XAMPP	Cross-platform (X), Apache, MariaDB, PHP, Perl
WAMP	Windows, Apache, MySQL, PHP
LAMP	Linux, Apache, MySQL, PHP
MAMP	Mac, Apache, MySQL, PHP

Σημείωση

Για να κατασκευάσουμε έναν εταιρικό Server εγκαθιστούμε και άλλα λογισμικά, όπως λογισμικό Mail Server για την αποστολή και λήψη e-mail.

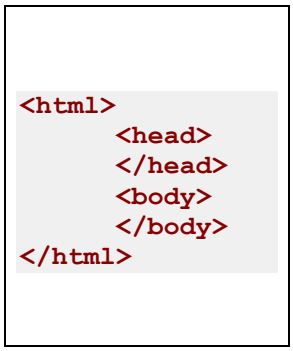
Για τα παραδείγματα σελίδων HTML και PHP προτείνεται η εγκατάσταση του XAMPP, η ενεργοποίηση του XAMPP control panel, η εκκίνηση (START) Apache, MySQL κ.λπ.

Επίσης για να γράψουμε και να δοκιμάσουμε σελίδες HTML και PHP μπορούμε να χρησιμοποιήσουμε κάποιους ιστοτόπους που «εκτελούν» τα scripts που γράφουμε. Για παράδειγμα, στη σελίδα http://phpfiddle.org/ μπορούμε να δοκιμάσουμε κώδικα HTML/PHP και να τον «τρέξουμε» online. Γράφουμε τον κώδικα στο “Code-Space” tab και επιλέγουμε “Run-F9”.

Αν θέλουμε να εργαστούμε με σελίδες HTML και JSP (Java Server Pages) μπορούμε να χρησιμοποιήσουμε περιβάλλοντα όπως το Netbeans που εγκαθιστά και «τρέχει» Glassfish server/ Apache server.

11.1.1 Δημιουργία του «σκελετού» μιας σελίδας σε HTML

Κάθε σελίδα HTML δομείται με ετικέτες (tags) που ξεκινούν με **<tag>** και τερματίζουν με **</tag>** (εικόνα 11.1)



```
<html>
      <head>
      </head>
      <body>
      </body>
</html>
```

Εικόνα 11.1 «Σκελετός» μιας σελίδας HTML

Πολλά προϊόντα ετοιμάζουν τον «σκελετό» της σελίδας στον οποίο και μπορούμε να προσθέσουμε τα στοιχεία που επιθυμούμε. Για παράδειγμα αν χρησιμοποιήσουμε το προϊόν Netbeans έχουμε τον παρακάτω «σκελετό».

σελίδα index.html

```
<!DOCTYPE html>
<html>
<head>
<title>TODO supply a title</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
  <div>TODO write content</div>
</body>
</html>
```

Εικόνα 11.2 «Σκελετός» μιας σελίδας HTML με χρήση του προϊόντος Netbeans

- Η ετικέτα <!DOCTYPE> ορίζει τον τύπο του εγγράφου και είναι προαιρετική αλλά αν τη χρησιμοποιήσουμε πρέπει να τη γράψουμε πρώτη.
- Η ετικέτα <html> ορίζει ένα HTML έγγραφο.
- Η ετικέτα <head> ορίζει πληροφορίες για το έγγραφο. Στο παράδειγμά μας χρησιμοποιείται η ετικέτα <title> για να ορίσει τον τίτλο του εγγράφου και η ετικέτα <meta> για να ορίσουμε πληροφορίες όπως character set.
- Η ετικέτα <body> ορίζει το «σώμα» (body) του εγγράφου. Στο παράδειγμά μας χρησιμοποιείται και η ετικέτα για να δούμε και ένα παράδειγμα ενότητας στη σελίδα.

Μπορούμε να δημιουργήσουμε τη σελίδα index.html (εικόνα 11.2) χρησιμοποιώντας πολλά προϊόντα. Για παράδειγμα, μπορούμε να χρησιμοποιήσουμε notepad++ και να γράψουμε το HTML έγγραφο.

Αν χρησιμοποιούμε το XAMPP η σελίδα index.html πρέπει να τοποθετείται στην περιοχή (folder) htdocs που βρίσκεται στο folder XAMPP. Εκτελέστε το script στον browser:

```
http://localhost/ index.html
```

Θα δείτε τη σελίδα σας με το μήνυμα:

TODO write content

Μπορείτε να αλλάξετε τη σελίδας σας (εικόνα 11.3).

σελίδα index.html

```
<!DOCTYPE html>
<html>
<head>
<title>index</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
  <div>What a Wonderful World!
</div>
</body>
</html>
```

Εικόνα 11.3 Εμφάνιση μηνύματος What a Wonderful World! σε ιστοσελίδα χωρίς μορφοποίηση

Θα δείτε τη σελίδα σας με το μήνυμα:

What a Wonderful World!

Στη σελίδα index.html δεν υπάρχει οποιαδήποτε μορφοποίηση κειμένου.

Γράψτε τη σελίδα index.html (εικόνα 11.4), χωρίς κάποιο tag μορφοποίησης, επίσης.

σελίδα index.html

```
<!DOCTYPE html>
<html>
<head>
<title>index</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
   What a Wonderful World!
   (by Louis Armstrong)
   I hear babies crying, I watch them grow
   They'll learn much more than I'll never know
   And I think to myself what a wonderful world
   Yes I think to myself what a wonderful world
</body>
</html>
```

Εικόνα 11.4 Σελίδα χωρίς μορφοποίηση

Θα δείτε τη σελίδα σας με το μήνυμα (εικόνα 11.5):

What a Wonderful World! (by Louis Armstrong) I hear babies crying, I watch them grow They'll learn much more than I'll never know And I think to myself what a wonderful world Yes I think to myself what a wonderful world

Εικόνα 11.5 Εμφάνιση σελίδας χωρίς μορφοποίηση

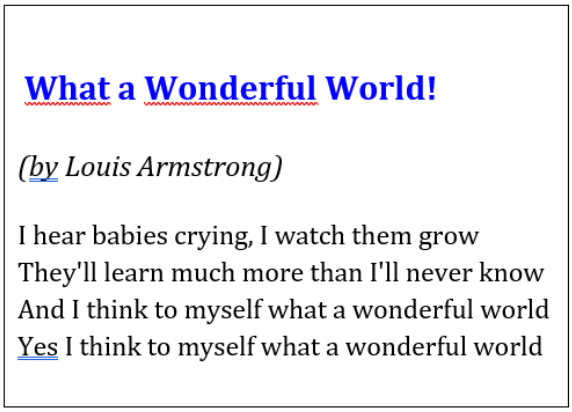
Να πως θα προσθέσετε κάποια μορφοποίηση (χρώμα σε γράμματα, πλάγια γράμματα) στον ορισμό της ιστοσελίδα σας (εικόνα 11.6).

σελίδα index.html

```
<!DOCTYPE html>
<html>
<head>
<title>index</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
   <h1><font color="blue"> What a Wonderful World!</font></h1>

   <p><em>(by Louis Armstrong)</em></p>
   <p>I hear babies crying, I watch them grow<br>
      They'll learn much more than I'll never know<br>
      And I think to myself what a wonderful world<br>
      Yes I think to myself what a wonderful world<br>
   </p>
</body>
</html>
```

Εικόνα 11.6 Προσθήκη μορφοποίησης στον ορισμό της ιστοσελίδας

Να πως θα δείτε τη σελίδα σας (εικόνα 11.7)



Εικόνα 11.7 Εμφάνιση της ιστοσελίδας με μορφοποίηση

11.1.2 Προσθήκη τίτλου, επικεφαλίδας και παραγράφου σε ιστοσελίδα

Στην εικόνα 11.8 βλέπουμε την ενημέρωση του ορισμού της σελίδας index.html με προσθήκη τίτλου, επικεφαλίδας και παραγράφου. Επιπλέον βλέπουμε και το αποτέλεσμα.

σελίδα index.html

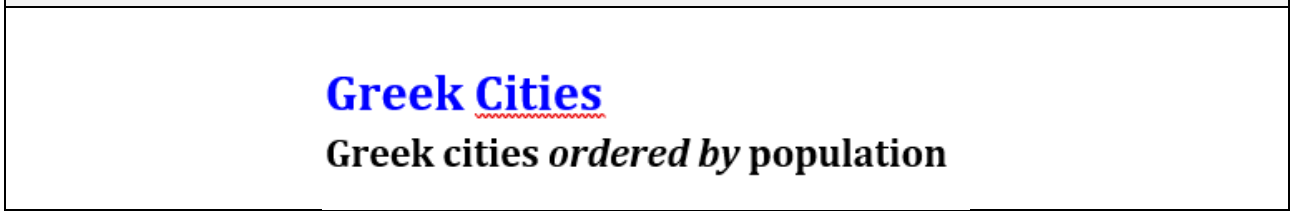
```
<html>
      <head>
            <title>Greek Cities</title>
      </head>
      <body>
            <h1>Greek Cities</h1>
            <p>Greek cities ordered by population</p>
      </body>
</html>
```

Εικόνα 11.8 Προσθήκη τίτλου, επικεφαλίδας και παραγράφου στην ιστοσελίδα. Αποτέλεσμα μορφοποίησης.

11.1.3 Προσθήκη μορφοποίησης

Στην εικόνα 11.9 βλέπουμε την ενημέρωση του ορισμού της σελίδας index.html με την προσθήκη τίτλου, επικεφαλίδας, παραγράφου, πλαγίων γραμμάτων και χρώματος για τα γράμματα. Επιπλέον βλέπουμε και το αποτέλεσμα.

```
<html>
	<head>
		<title>Greek Cities</title>
	</head>
	<body>
		<h1><font color="blue">Greek Cities</font></h1>
		<p><b>Greek cities <i>ordered by</i> population</b></p>
	</body>
</html>
```



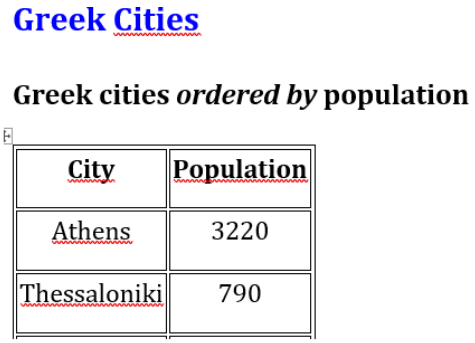
Εικόνα 11.9 Προσθήκη τίτλου, επικεφαλίδας, παραγράφου, πλαγίων γραμμάτων και χρώματος για τα γράμματα στην ιστοσελίδα. Αποτέλεσμα μορφοποίησης.

11.1.4 Προσθήκη πίνακα με δεδομένα

Στην εικόνα 11.10 βλέπουμε την ενημέρωση του ορισμού της σελίδας index.html με την προσθήκη και πίνακα με δεδομένα. Επιπλέον βλέπουμε και το αποτέλεσμα.

```
<html>
	<head>
		<title>Greek Cities</title>
	</head>
	<body>
		<h1><font color="blue">Greek Cities</font></h1>
		<p><b>Greek cities <i>ordered by</i> population</b></p>
		<table border="1px">
			<tr><th>City</th><th>Population</th></tr>
			<tr align="center"><td>Athens</td><td>3220</td></tr>
			<tr align="center"><td>Thessaloniki</td><td>790</td></tr>
```

```
            </table>
      </body>
</html>
```



Εικόνα 11.10 Προσθήκη πίνακα στην ιστοσελίδα και αποτέλεσμα μορφοποίησης.

11.1.5 Οι ετικέτες (tags) HTML ανά κατηγορία.

Στη συνέχεια παρουσιάζονται ομαδοποιημένες κάποιες χρήσιμες ετικέτες με σύντομη επεξήγηση.

Στην εικόνα 11.11 βλέπουμε τις βασικές ετικέτες της HTML

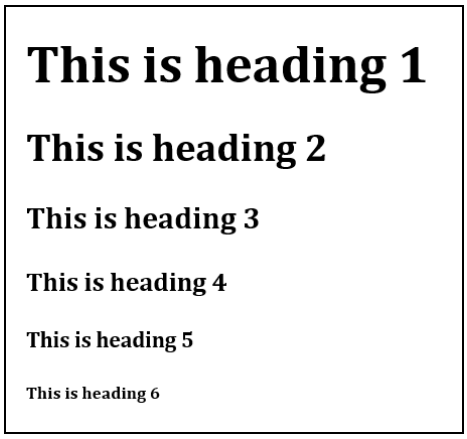
<!DOCTYPE>	Ορίζει τον τύπο του εγγράφου
<html>	Ορίζει HTML έγγραφο
<head>	Ορίζει πληροφορίες για το έγγραφο
<title>	Ορίζει τίτλο για το έγγραφο
<body>	Ορίζει το «σώμα» του εγγράφου
<h1> μέχρι <h6>	Ορίζει HTML επικεφαλίδες (headings) με 6 τρόπους
<p>	Ορίζει παράγραφο
 	Ορίζει αλλαγή γραμμής
<!--...-->	Ορίζει σχόλιο

Εικόνα 11.11 Βασικές ετικέτες της HTML με σύντομη επεξήγηση.

Μπορείτε να δοκιμάσετε όλες τις επικεφαλίδες (εικόνα 11.12). Στην εικόνα 11.13 βλέπετε τη διαφορά μεταξύ τους.

```
<html>
<head>
<title>Example of Headings</title>
</head>
<body>
   <h1>This is heading 1</h1> <br>
   <h2>This is heading 2</h2><br>
   <h3>This is heading 3</h3><br>
   <h4>This is heading 4</h4><br>
   <h5>This is heading 5</h5><br>
   <h6>This is heading 6</h6><br>
</body>
</html>
```

Εικόνα 11.12 Δοκιμή όλων των επικεφαλίδων σε ιστοσελίδα



Εικόνα 11.13 Διαφορά των επικεφαλίδων στην εμφάνιση της ιστοσελίδας

Στην εικόνα 11.14 βλέπουμε ετικέτες μορφοποίησης (Formatting).

<b>	Ορίζει κείμενο με χοντρά γράμματα (bold text)
	Ορίζει κεντραρισμένο κείμενο
	Ορίζει κείμενο στο οποίο προσδίδουμε κάποια έμφαση (emphasized text)
<font>	Ορίζει γραμματοσειρά (font), χρώμα (color) και μέγεθος (size) κειμένου
<i>	Ορίζει τμήμα ενός κειμένου με διαφορετικό τρόπο, π.χ., σε ένα κανονικό κείμενο ορίζει τμήμα του με πλάγια γράμματα. Προσοχή! Ετικέτες , κ.λπ. υπερισχύουν της ετικέτας αυτής.
	Ορίζει σημαντικό κείμενο
<sub>	Ορίζει κείμενο ως δείκτη (subscripted text)
<sup>	Ορίζει κείμενο ως εκθέτη (superscripted text)

Εικόνα 11.14 Ετικέτες μορφοποίησης

Ετικέτες για ορισμό φόρμας

Στην εικόνα 11.15 βλέπουμε ετικέτες για ορισμό φόρμας.

<form>	Ορίζει μια φόρμα
<input>	Ορίζει ένα στοιχείο της φόρμας (input control), π.χ., ένα «κουτί» για να εισάγει ο χρήστης κάποιο στοιχείο
<textarea>	Ορίζει μια περιοχή πολλών γραμμών για την εισαγωγή κειμένου (multiline input control)
<button>	Ορίζει ένα κουμπί στο οποίο μπορούμε να κάνουμε κλικ (clickable button)
<label>	Ορίζει ένα κείμενο (label) για ένα στοιχείο <input>

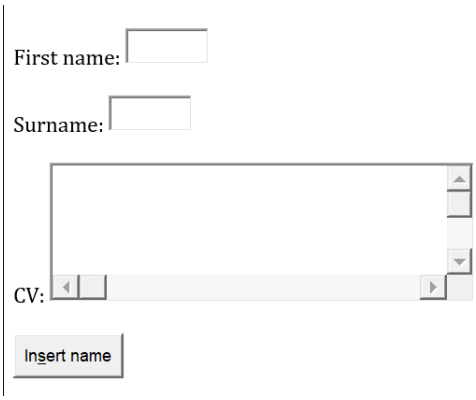
Εικόνα 11.15 Ετικέτες για ορισμό φόρμας

Στην εικόνα 11.16 παραθέτουμε παράδειγμα ορισμού φόρμας, τον κώδικα της και το αποτέλεσμα της εκτέλεσης.

Στην εικόνα 11.17 παραθέτουμε παράδειγμα ορισμού radio button με το αποτέλεσμα εκτέλεσης. Στην εικόνα 11.18 παραθέτουμε παράδειγμα ορισμού options με το αποτέλεσμα εκτέλεσης. Στην εικόνα 11.17 παραθέτουμε παράδειγμα ορισμού λίστας με το αποτέλεσμα εκτέλεσης.

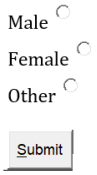
```
<html>
<head> <title>Example</title> </head>
<body>
```

```
  <form name="my_form" action="/my_page.php" method="get"><br>
   First name: <input type="text" name="fname"><br><br>
   Surname: <input type="text" name="surname"><br><br>
   CV: <textarea rows="4" cols="50" name="cv"></textarea> <br><br>
   <input type="submit" value="Insert name">
</form>
</body>
</html>
```



Εικόνα 11.16 Ορισμός και εμφάνιση φόρμας

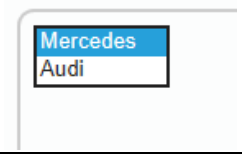
```
<html>
<head> <title>Example</title> </head>
<body>
<form action="/my_page.php">
  <label for="male">Male</label>
  <input type="radio" name="gender" id="male" value="male"><br>
  <label for="female">Female</label>
  <input type="radio" name="gender" id="female" value="female"><br>
  <label for="other">Other</label>
  <input type="radio" name="gender" id="other" value="other"><br><br>
  <input type="submit" value="Submit">
</form>
</body>
</html>
```



Εικόνα 11.17 Ορισμός και εμφάνιση radio button

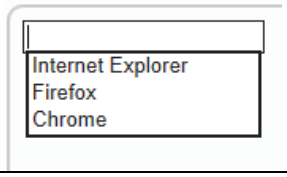
```
<html>
<head>
<title> Select Brandname</title>
</head>
<body>
<select>
   <option value="mercedes">Mercedes</option>
   <option value="audi">Audi</option>
</select>
```

```
</body>
</html>
```



Εικόνα 11.18 Ορισμός και εμφάνιση options

```
<html>
<head>
<title> Select browser</title>
</head>
<body>
  <input list="browsers">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
  </datalist>
</body>
</html>
```



Εικόνα 11.19 Ορισμός και εμφάνιση λίστας

11.1.6 Ετικέτες για εικόνες (Images)

Στην εικόνα 11.20 βλέπουμε ένα πρώτο παράδειγμα ορισμού εικόνας σε ιστοσελίδα και το αποτέλεσμα εκτέλεσης.

```
<html>
<head>
  <title> Mercury </title>
</head>
<body>
  <img src="spectra_mercury.jpg" alt="This colorful version of Mercury was
assembled from spectral data taken with the MESSENGER spacecraft, highlighting
various minerals on the planet's surface. NASA/Johns Hopkins University
Applied Physics Laboratory/Carnegie Institution of Washington" height="547"
width="547">
</body>
</html>
```

Παρατηρήστε το κείμενο που συνοδεύει την εικόνα.

Εικόνα 11.20 Ορισμός και εμφάνιση εικόνας

Στην εικόνα 11.21βλέπουμε βασικές ετικέτες διαχείρισης εικόνας.

<img>	Ορίζει μια εικόνα
<map>	Ορίζει την εικόνα χάρτη στον πελάτη (a client-side image-map)
<area>	Ορίζει περιοχή (area) σε εικόνα χάρτη (image-map)

Εικόνα 11.21 Βασικές ετικέτες διαχείρισης εικόνας

Στην εικόνα 11.22 βλέπουμε τον ορισμό και την εμφάνιση ιστοσελίδας με εικόνα του ηλιακού συστήματος και σύνδεσμο στη Wikipedia.

```
<html>
<head>
  <title>Sun</title>
</head>
<body>
   <img src="640px-Planets2013.png" width="640" height="280" alt="Planets"
   usemap="#planetmap">
   <map name="planetmap">
       <area shape="rect" coords="0,0,85,180"
       href="https://en.wikipedia.org/wiki/Solar_System" alt="Solar system">
   </map>
</body>
</html>
```

Εικόνα 11.22 Ορισμός και εμφάνιση ιστοσελίδας με εικόνα του ηλιακού συστήματος και σύνδεσμο στη Wikipedia

Στην εικόνα 11.23 βλέπουμε και άλλες κατηγορίες ετικετών: Ετικέτες Audio / Video, ετικέτες σύνδεσμοι (Links), ετικέτες για λίστες (Lists), ετικέτες για την κατασκευή πινάκων (Tables)

Ετικέτες Audio / Video

<audio>	Ορίζει περιεχόμενο ήχου
<video>	Ορίζει video

Ετικέτες σύνδεσμοι (Links)

<a>	Ορίζει σύνδεσμο (hyperlink)
<nav>	Ορίζει συνδέσμους πλοήγησης (navigation links)

Ετικέτες για λίστες (Lists)

	Ορίζει μη διατεταγμένη λίστα (unordered list)
	Ορίζει διατεταγμένη λίστα (ordered list)
	Ορίζει στοιχείο λίστας (list item)

Ετικέτες για την κατασκευή πινάκων (Tables)

<table>	Ορίζει πίνακα
<th>	Ορίζει όνομα κελιού στήλης (header cell) σε ένα πίνακα
<tr>	Ορίζει μια γραμμή (row) του πίνακα
<td>	Ορίζει ένα κελί (cell) στον πίνακα

Εικόνα 11.23 Ετικέτες Audio/Video, ετικέτες σύνδεσμοι (Links), ετικέτες για λίστες (Lists), ετικέτες για την κατασκευή πινάκων (Tables)

11.2 Εισαγωγή στην PHP

Η λέξη PHP είναι ένα αρκτικόλεξο του όρου "PHP: Hypertext Preprocessor". Η ανοικτού κώδικα γλώσσα PHP είναι μια ευρέως χρησιμοποιούμενη γλώσσα για τη συγγραφή «δέσμης ενεργειών» (script). Η «δέσμη ενεργειών» (PHP script), δηλαδή ο κώδικας που γράφουμε σε PHP, εκτελείται στον διακομιστή (server) και το αποτέλεσμα επιστρέφεται στο πρόγραμμα περιήγησης ως απλή ιστοσελίδα HTML. Η λήψη και χρήση της γλώσσας PHP είναι δωρεάν.

Τα αρχεία (scripts) PHP έχουν επέκταση ".php" και μπορούν να περιέχουν κείμενο, κώδικα PHP και HTML, CSS (CSS είναι γλώσσα που χρησιμοποιούμε για να προσδώσουμε κάποιο στυλ εμφάνισης σε έγγραφα HTML), JavaScript (JavaScript είναι μια εύκολη στην εκμάθηση, δημοφιλής γλώσσα προγραμματισμού του Παγκόσμιου Ιστού).

Η γλώσσα PHP είναι σχετικά απλή στην εκμάθηση και δίνει πολλές δυνατότητες στους προγραμματιστές δημιουργίας ιστοσελίδας δυναμικού περιεχομένου. Δηλαδή η γλώσσα επιτρέπει να εμφανίσουμε δεδομένα της βάσης στην ιστοσελίδα, και να προσθέσουμε, να διαγράψουμε, να τροποποιήσουμε δεδομένα στη βάση δεδομένων. Έτσι δικαιολογείται και ο όρος δυναμικό περιεχόμενο ιστοσελίδας.

Μας επιτρέπει να δημιουργήσουμε και να διαχειριστούμε αρχεία (άνοιγμα και κλείσιμο, ανάγνωση, ενημέρωση, διαγραφή) στον διακομιστή (server). Ο κώδικάς μας μπορεί να συλλέξει δεδομένα από ηλεκρονική φόρμα (ιστοσελίδας), να διαχειριστεί εικόνες, αρχεία PDF, XML κ.λπ. Μπορούμε να κάνουμε έλεγχο της πρόσβασης χρηστών, να κρυπτογραφήσουμε δεδομένα, να διαχειριστούμε cookies.

Στη συνέχεια θα προσεγγίσουμε το θέμα μας με μία σειρά από παραδείγματα για αναγνώστες που πρώτη φορά μελετούν το αντικείμενο.

Οι ενδιαφερόμενοι αναγνώστες για μια παραπέρα χρηστική συζήτηση του θέματος του προγραμματισμού εφαρμογών Web παραπέμπονται στα:

- PHP Tutorial PHP Tutorial (w3schools.com)

- HTML Tutorial HTML Tutorial (w3schools.com)
- CSS Tutorial CSS Tutorial (w3schools.com)
- JavaScript Tutorial JavaScript Tutorial (w3schools.com)

11.2.1 Κάθε σελίδα HTML είναι και σελίδα PHP

Κάθε html σελίδα είναι και php σελίδα. Για παράδειγμα, η σελίδα index.php (εικόνα 11.24) είναι ίδια με την index.html που είδαμε προηγουμένως και εμφανίζει την επικεφαλίδα **Greek Cities**, το κείμενο **Greek cities *ordered by* population** και τον πίνακα που είδαμε.

Σελίδα index.php

```
<html>
	<head>
		<title>Greek Cities</title>
	</head>
	<body>
		<h1><font color="blue"><?php echo "Greek Cities" ?></font></h1>
		<p><b>Greek cities <i>ordered by</i> population</b></p>
		<table border="1px">
			<tr><th>City</th><th>Population</th></tr>
			<tr align="center"><td>Athens</td><td>3220</td></tr>
			<tr align="center"><td>Thessaloniki</td><td>790</td></tr>
		</table>
	</body>
</html>
```

Εικόνα 11.24 Μία σελίδα HTML είναι και σελίδα PHP

11.2.2 Προσθήκη και χρήση μεταβλητών

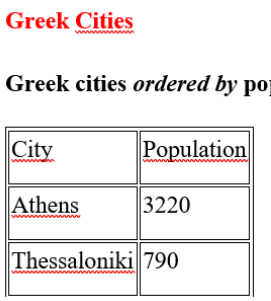
Στην εικόνα 11.25 βλέπουμε την προσθήκη και χρήση μεταβλητών και το αποτέλεσμα εκτέλεσης του κώδικα.

Παρατηρήστε ότι ο κώδικάς μας έχει τη δομή που είδαμε στη γλώσσα HTML. Εμφυτευμένες στον κώδικα HTML βλέπουμε τις δηλώσεις και τις εντολές της γλώσσας php. Οι εντολές αυτές περικλείονται σε ετικέτες (tag): <?php ……. ?>, π.χ., η εντολή <?php echo "Greek Cities" ? > δείχνει το μήνυμα "Greek Cities". Η μορφοποίηση (δηλαδή το πως θα φαίνεται στην ιστοσελίδα των αποτελεσμάτων) αποτελεί έργο του κώδικα HTML που «πλαισιώνει» τον κώδικα PHP.

Παρατηρήστε και τον ορισμό της μεταβλητής $cheader. Η «τιμή» της είναι μία συμβολοσειρά (string) που περιλαμβάνει το κείμενο Greek cities ordered by population και τις ετικέτες μορφοποίησης του κειμένου σε HTML.

```
<html>
	<head>
		<title>Greek Cities</title>
	</head>
	<?php $cheader = "<b>Greek cities <i>ordered by</i> population</b>"; ?>
	<body>
		<h1><font color="red"><?php echo "Greek Cities" ?></font></h1>
		<p><?php echo $cheader ?></p>
		<table border="1px">
			<tr><th>City</th><th>Population</th></tr>
			<tr align="center"><td>Athens</td><td>3220</td></tr>
```

```
                <tr align="center"><td>Thessaloniki</td><td>790</td></tr>
            </table>
      </body>
</html>
```



Εικόνα 11.25 Προσθήκη και χρήση μεταβλητών και το αποτέλεσμα εκτέλεσης του κώδικα. Παρατηρήστε ότι οι εντολές της γλώσσας περικλείονται σε ετικέτες (tag), <?php ....?>.

Στην εικόνα 11.26 παραθέτουμε ανάλογο script σε Java Server Pages. Μπορούμε να το γράψουμε και να το εκτελέσουμε, π.χ., στο περιβάλλον Netbeans χρησιμοποιώντας Glassfish/Apache server. Η τεχνολογία Java Server Pages παρουσιάζεται σε ξεχωριστό κεφάλαιο του συγγράμματος.

```
<html>
 <head>
      <title>Greek Cities</title>
 </head>
 <% String cheader= "<b>Greek cities <i>ordered by</i> population</b>"; %>
      <body>
            <h1><font color="red"><%= "Greek Cities" %></font></h1>
            <p><%= cheader %></p>
            <table border="1px">
                  <tr><th>City</th><th>Population</th></tr>
                  <tr align="center"><td>Athens</td><td>3220</td></tr>
                  <tr align="center"><td>Thessaloniki</td><td>790</td></tr>
            </table>
      </body>
</html>
```

Εικόνα 11.26 Χρήση μεταβλητών σε script σε Java Server Pages

11.2.3 Δημιουργία array

Στην εικόνα 11.27 βλέπουμε τον ορισμό array σε γλώσσα php και την εμφάνιση στην ιστοσελίδα.

```
<html>
	<head>
		<title>Greek Cities</title>
	</head>
	<?php $cheader = "<b>Greek cities</b> ordered by population";
	$cities = array("Athens", "Thessaloniki", "Patra", "Heraklion");
	$population = array(3220, 790, 170, 160); ?>
	<body>
		<h1><font color="green"><?php echo "Greek Cities" ?></font></h1>
		<p><?php echo $cheader ?></p>
		<table border="1px">
			<tr><th>City</th><th>Population</th></tr>
			<tr align="center"><td>Athens</td><td>3220</td></tr>
			<tr align="center"><td>Thessaloniki</td><td>790</td></tr>
<tr align="center"><td>Patra</td><td>170</td></tr>
<tr align="center"><td>Heraklion</td><td> 160</td></tr>
		</table>
	</body>
</html>
```

Greek Cities

Greek cities ordered by population

City	Population
Athens	3220
Thessaloniki	790
Patra	170
Heraklion	160

Εικόνα 11.27 Ορισμός array σε γλώσσα php και εμφάνιση στην ιστοσελίδα

Ακολουθεί επεξήγηση για τη δημιουργία array.

Στο παράδειγμά μας ορίζονται δύο arrays:

```
$cities = array("Athens", "Thessaloniki", "Patra", "Heraklion");
$population = array(3220, 790, 170, 160); ?>
Θα μπορούσαμε να ορίσουμε array και με διαφορετικό τρόπο. Για παράδειγμα:
$cities[0]="Athens";
$cities[1]="Thessaloniki";
$cities[2]="Patra";
$cities[3]="Heraklion";
```

Εναλλακτικά θα μπορούσαμε να ορίσουμε και ως εξής:

```
$cities[]="Athens";
$cities[]="Thessaloniki";
$cities[]="Patra";
$cities[]="Heraklion";
```

Ορισμός array σε Java Server Pages

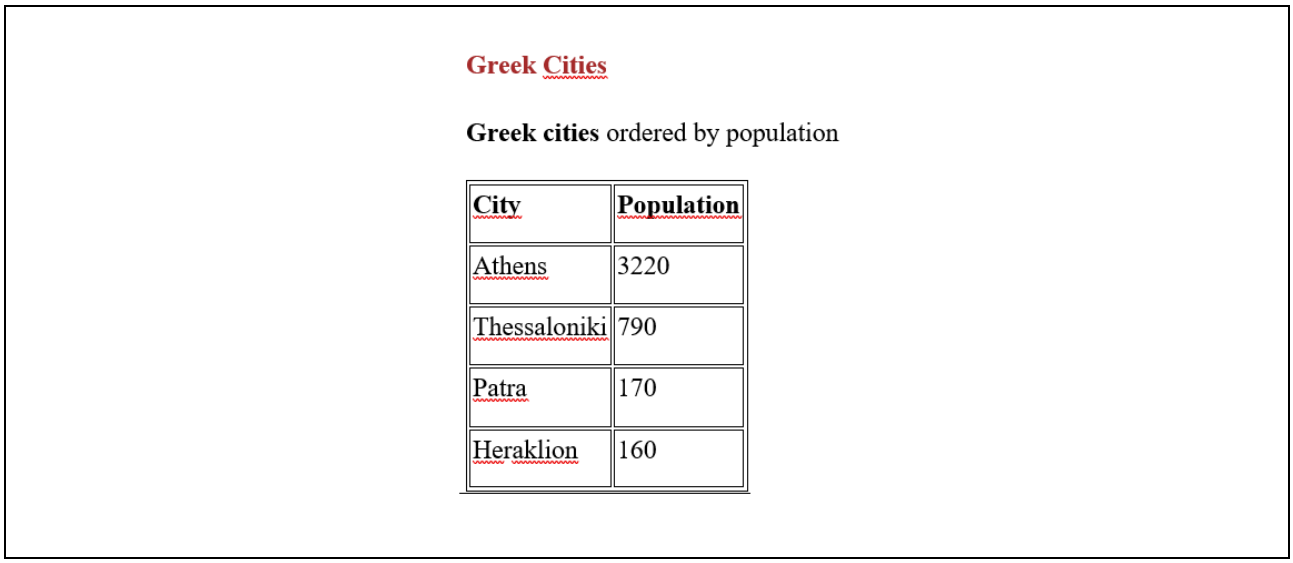
Ακολουθεί ορισμός array σε Java Server Pages για να δούμε τη διαφορά.

```
<% String cheader = "<b>Greek cities</b> ordered by population";
   String []cities=new String[100];

   cities[0]="Athens";
   cities[1]="Thessaloniki";
   cities[2]="Patra";
   cities[3]="Heraklion";

   int [] population=new int[100];
   population[0]=3220;
   population[1]=790;
   population[2]=170;
   population[3]=160;
 %>
```

11.2.4 Δυναμική δημιουργία του πίνακα (table)

Στην εικόνα 11.28 παραθέτουμε τον δυναμικό ορισμό array σε γλώσσα php και την εμφάνιση στην ιστοσελίδα

```
<html>
      <head>
            <title>Greek Cities</title>
      </head>
      <?php $cheader = "<b>Greek cities</b> ordered by population";
      $cities = array("Athens", "Thessaloniki", "Patra", "Heraklion");
      $population = array(3220, 790, 170, 160); ?>
      <body>
            <h1><font color="brown">
            <?php echo "Greek Cities" ?></font></h1>
            <p><?php echo $cheader ?></p>
            <table border="1px">
                  <tr><th>City</th><th>Population</th></tr>
                  <?php for ($i = 0; $i < sizeof($cities); $i++) { ?>
                  <tr align="center"><td><?php echo $cities[$i] ?></td>
                        <td><?php echo $population[$i] ?></td>
                  </tr>
                  <?php } ?>
            </table>
      </body>
</html>
```



Εικόνα 11.28 Δυναμικός ορισμός array σε γλώσσα php και εμφάνιση στην ιστοσελίδα

11.3 Πρόσθετα παραδείγματα χρήσης HTML ετικετών

Η εμφάνιση της ιστοσελίδας είναι σημαντικό ζήτημα στις εφαρμογές WEB. Μελετήσαμε ήδη την αλληλεπίδραση-συνεργασία του κώδικα HTML και του κώδικα PHP μέσα στο ίδιο πρόγραμμά μας. Θα αναφερθούμε στη χρήση περισσότερων ετικετών HTML που είναι χρήσιμες στις εφαρμογές.

11.3.1 Αλλαγή γραμμής και σύνδεσμος (link) μετάβασης στη σελίδα Πανεπιστημίου

Στην εικόνα 11.29 βλέπουμε τη χρήση συνδέσμου (link) μετάβασης στη σελίδα Πανεπιστημίου και το αποτέλεσμα της εκτέλεσης.

```
<html>
	<head>
		<title>Greek Cities</title>
	</head>
	<body>
		<h1><font color="blue">Greek Cities</font></h1>
		<br>
		<br>
		<p><a href="https://www.uniwa.gr">UNIWA Web site</a></p>
	</body>
</html>
```

Εικόνα 11.29 Χρήση συνδέσμου (link) μετάβασης στη σελίδα Πανεπιστημίου

11.3.2 Χρήση λογοτύπου

Η ιστοσελίδα ενός οργανισμού πρέπει να περιλαμβάνει λογότυπο. Στην εικόνα 11.30 βλέπουμε τη χρήση λογοτύπου και συνδέσμου (link) μετάβασης στη σελίδα Πανεπιστημίου. Δείτε και το αποτέλεσμα την εμφάνιση της ιστοσελίδας.

```
<html>
	<head>
		<title>Greek Cities</title>
	</head>
	<body>
		<h1><font color="blue">Greek Cities</font></h1>
		<br>
		<br>
		<p><img src="https://www.uniwa.gr/wp-content/uploads/2018/11/logo-
pada.png" alt="UNIWA" width="150" height="150"></p>
		<p><a href="https://www.uniwa.gr">UNIWA Web site</a></p>
		<p>
	</body>
</html>
```

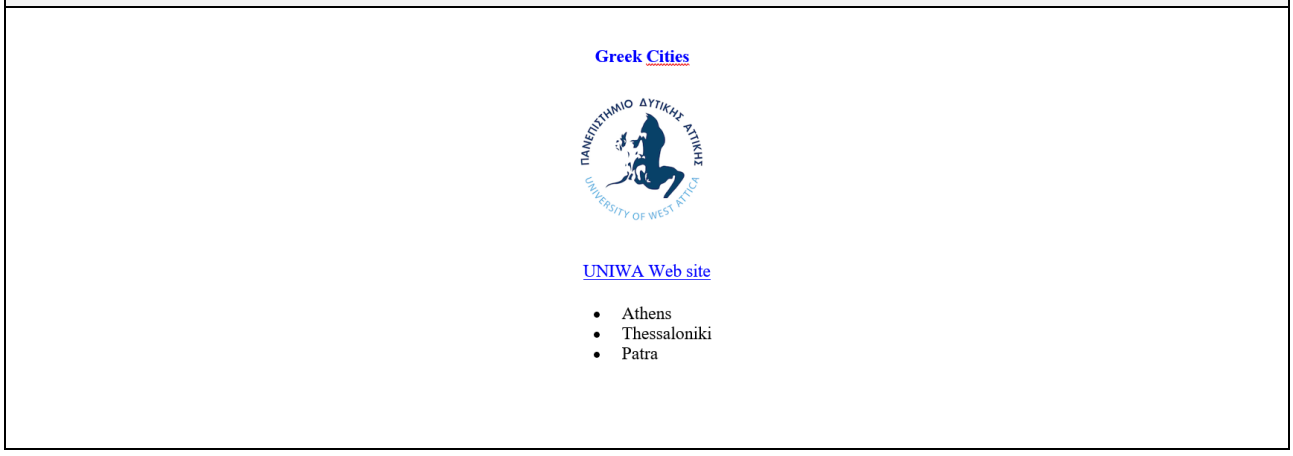
Εικόνα 11.30 Χρήση λογοτύπου και συνδέσμου (link) μετάβασης στη σελίδα Πανεπιστημίου

11.3.3 Λίστες

Στην εικόνα 11.31 βλέπουμε τον κώδικα που επιτρέπει χρήση λογοτύπου, συνδέσμου (link) μετάβασης στη σελίδα Πανεπιστημίου και την εμφάνιση λίστας. Παραθέτουμε και το αποτέλεσμα την εμφάνιση της ιστοσελίδας.

```
<html>
	<head>
		<title>Greek Cities</title>
	</head>
	<body>
		<h1><font color="blue">Greek Cities</font></h1>
		<br>
		<br>
		<p><img src="https://www.uniwa.gr/wp-content/uploads/2018/11/logo-
pada.png" alt="UNIWA" width="150" height="150"></p>
		<p><a href="https://www.uniwa.gr">UNIWA Web site</a></p>
		<p>
		<ul>
			<li>Athens</li>
			<li>Thessaloniki</li>
			<li>Patra</li>
		</ul>
```

```
    </body>
</html>
```



Εικόνα 11.31 Χρήση λογοτύπου, συνδέσμου (link) μετάβασης στη σελίδα Πανεπιστημίου και εμφάνιση λίστας

11.4 Δημιουργία του ιστοτόπου Ted Codd Fun Club με χρήση HTML

Δημιουργήστε ιστότοπο για το διάσημο ερευνητή Edgar Frank "Ted" Codd χρησιμοποιώντας το εργαλείο Netbeans. Οι σελίδες του ιστοτόπου είναι οι εξής:

1) Codd_index_cr.html
2) quotes.html
3) contact.html
4) photos.html
5) video.html

Προσοχή! Κεφαλαία και πεζά έχουν τη σημασία τους!

Τα βασικά στοιχεία για τον Codd υπάρχουν στη σελίδα Codd_index_cr.html

Στην αρχική σελίδα αυτή υπάρχει ένα navigation menu ως εξής:

```
Home | Ted Codd's quotes | Contact Us | Photos | Videos
```

Κάθε επιλογή του μενού συνδέεται με την κλήση μιας σελίδας και οι αντίστοιχες σελίδες των επιλογών κατά σειρά είναι:

Η επιλογή home μας ξαναγυρίζει στη σελίδα Codd_index_cr.html.

Η σελίδα quotes.html δείχνει σημαντικές φράσεις του Codd.

Η σελίδα contact.html δείχνει στοιχεία επικοινωνίας. Στη φάση αυτή επιτρέπει εισαγωγή στοιχείων αλλά ουσιαστικά είναι ανενεργή.

Η σελίδα photos.html δείχνει δύο φωτογραφίες του Codd.

Η σελίδα video.html δείχνει ένα βίντεο σχετικό με τη βράβευση του Codd με το βραβείο Turing και έχει και σύνδεσμο για τη βράβευση αυτή.

Αν υποτεθεί ότι κατά την υλοποίηση με το εργαλείο Netbeans κατασκευάζετε τον ιστότοπο στην εφαρμογή WebApplication11 τότε οι σελίδες σας είναι στο folder:

Documents/NetBeansProjects/WebApplication11/Web

Ακολουθούν οι σελίδες.

11.4.1 Δημιουργία αρχικής σελίδας (Ted Codd webpage - Start screen) index.html

Στην εικόνα 11.32 βλέπουμε την ιστοσελίδα Ted Codd webpage-Start screen την οποία θα κατασκευάσουμε. Δείτε πως θα φαίνεται η αρχική σελίδα. Περιλαμβάνονται λίγα βιογραφικά στοιχεία:

Bio: Edgar Frank Ted Codd was an English computer scientist who, while working for IBM, invented the relational model for database management, the theoretical basis for relational databases. He made other valuable contributions to computer science, but the relational model, a very influential general theory of data management, remains his most mentioned achievement.

Επιπλέον περιλαμβάνει ένα μικρό απόσπασμα από σύγγραμμά του:

```
"The most important motivation for the research work that resulted in the
relational model was the objective of providing a sharp and clear boundary
between the logical and physical aspects of database management."
E. F. Codd, Relational Database: A Practical Foundation for Productivity (1982).
```

Home | Ted Codd's quotes | Contact Us | Photos | Videos

Welcome to Ted Codd Fun Club's Webpage

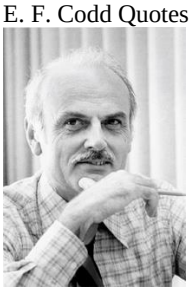
Edgar Frank "Ted" Codd.
Born: 19 August 1923

Died: 18 April 2003

Known for:

- OLAP
- Relational model
- Codd's cellular automaton
- Codd's 12 rules
- Boyce Codd normal form

E. F. Codd Quotes



"The most important motivation for the research work that resulted in the - relational model was the objective of providing a sharp and clear boundary - between the logical and physical aspects of database management."

E. F. Codd, *Relational Database: A Practical Foundation for Productivity (1982).* - Source

Bio: Edgar Frank Ted Codd

was an English computer scientist who, while working for IBM,

invented the relational model for database management,

the theoretical basis for relational databases.

He made other valuable contributions to computer science,

but the relational model,

a very influential general theory of data management,

remains his most mentioned achievement.

Read more at

- wikipedia

Εικόνα 11.32 Ιστοσελίδα Ted Codd webpage-Start screen που θα κατασκευάσουμε

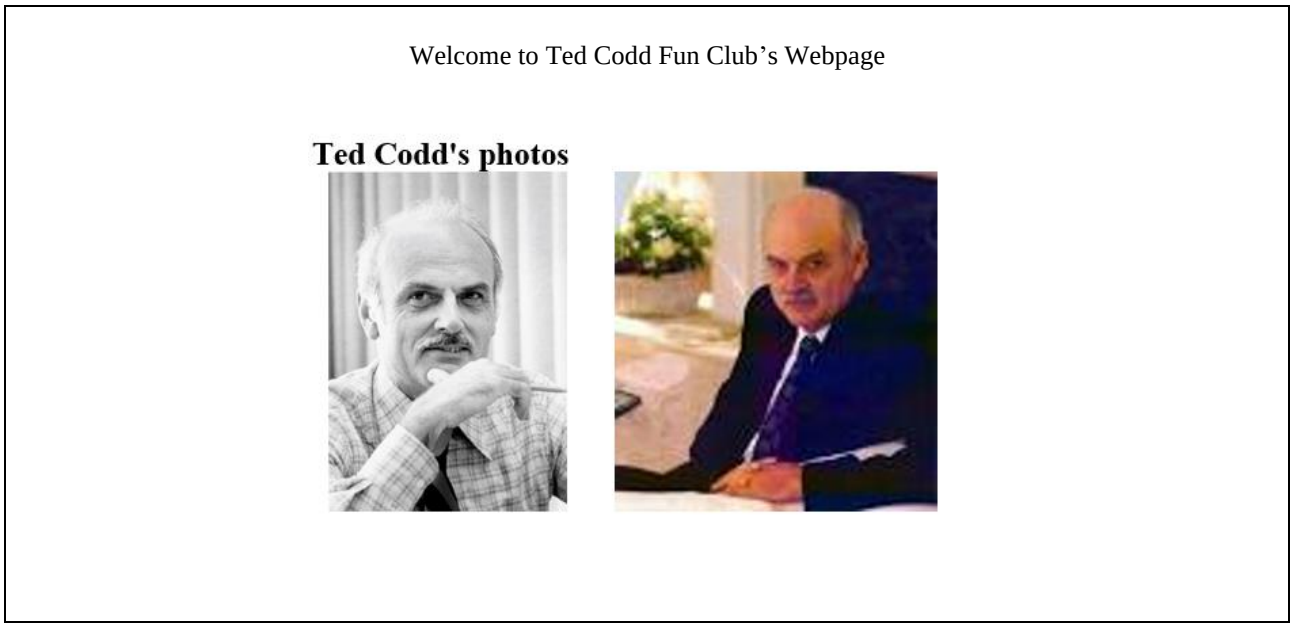
Ακολουθεί ο κώδικας της αρχικής σελίδας (Start screen): Codd_index_cr.html

```
<!DOCTYPE html>
<html>
<head>
<title>Ted Codd Fun Club's Webpage</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
      <div id="header">
<div id="nav"><a href="Codd_index_cr.html">Home</a> |
<a href="quotes.html">Ted Codd's quotes</a> |
<a href="contact.html">Contact Us</a> |
<a href="photos.html">Photos</a> |
<a href="video.html">Videos</a></div>
</div>
 <div id="main-content">
                  <div id="logo">
                     Welcome to Ted Codd Fun Club's Webpage
                  </div>
                  <h1>Edgar Frank "Ted" Codd.
  </h1>
<p>Born: 19 August 1923</p>
<p>Died: 18 April 2003</p>
            <p>Known for: </p>
                        <ul style="margin-top:10px;">
                              <li>OLAP</li>
```

```
                              <li>Relational model</li>
                              <li>Codd's cellular automaton</li>
                              <li>Codd's 12 rules</li>
                              <li>Boyce Codd normal form</li>
                        </ul>
                  <h2>E. F. Codd Quotes </h2>
                  <p>
                        <img src="Edgar_F_Codd.jpg" width="139" height="150"
style="margin: 0 10px 10px 0;float:left;" />
                        <em>"The most important motivation for the research work
that resulted in the</em> - <br/>
                        <em>relational model was the objective of providing a
sharp and clear boundary</em> - <br/>
                        <em>between the logical and physical aspects of database
management."</em> - E. F. Codd<br/>
                        <em>   Relational Database: A Practical Foundation for
Productivity (1982).</em> - Source
  <br/>
                  </p> <br/> <br/> <br/>
                        <h3>Bio: Edgar Frank Ted Codd</h3>
              <p>was an English computer scientist who, while working for
IBM,</p>
<p>invented the relational model for database management,</p>
<p>the theoretical basis for relational databases.</p>
<p>He made other valuable contributions to computer science,</p>
<p>but the relational model,</p>
<p>a very influential general theory of data management,</p>
<p>remains his most mentioned achievement.</p>
      <h3>Read more at</h3>
                        <div>
                              <ul>
            <li><a
href="https://en.wikipedia.org/wiki/Edgar_F._Codd">wikipedia</a></li>
                              </ul>
                        </div>
            </div>
            <div id="footer" style="color:blue">
Copyright; 2021 Ted Codd Fun Club. All rights reserved.<br/>
      </div>
</body>
</html>
```

11.4.2 Ιστοσελίδα σελίδα που εμφανίζει φωτογραφίες του Codd

Στην εικόνα 11.33 βλέπουμε ιστοσελίδα με φωτογραφίες του Ted Codd (photos.html).



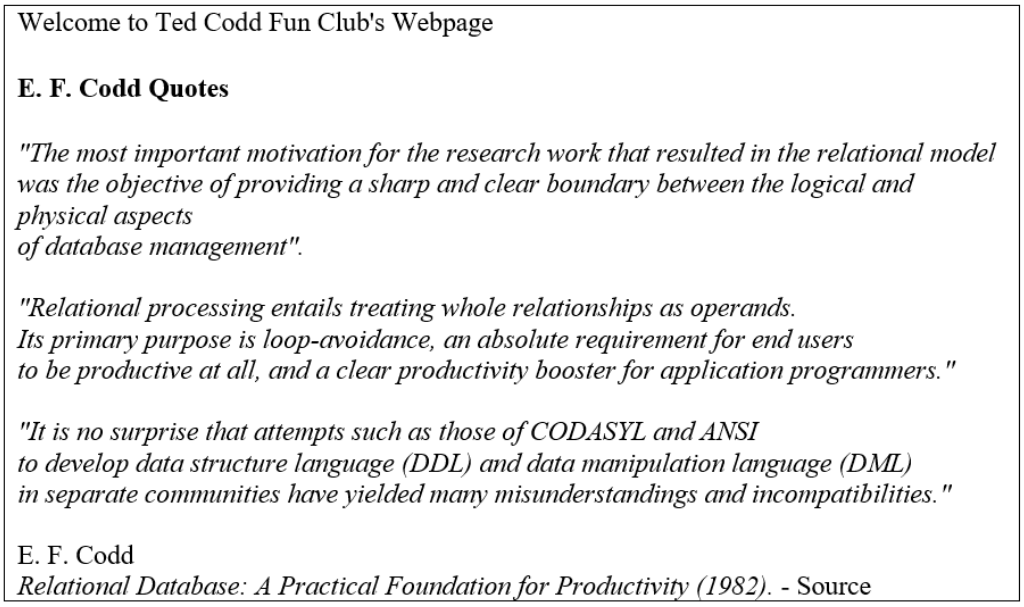
Εικόνα 11.33 Ιστοσελίδα με φωτογραφίες του Ted Codd

Παραθέτουμε τον κώδικα.

```
<!DOCTYPE html>
<html>
<head>
<title>E.F.Codd photos</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
      <div id="logo">
Welcome to Ted Codd Fun Club's Webpage <p></p>
      </div>
<div>
<h2>Ted Codd's photos</h2>
<p>
<img src="Edgar_F_Codd.jpg" width="139" height="150" style="margin: 0 10px 10px
0;float:left;" />
<img src="codd_1.jpg" width="139" height="150" style="margin: 0 10px 10px
0;float:left;" /><br/>
</p>
</div>
</body>
</html>
```

11.4.3 Ιστοσελίδα με σημαντικές φράσεις (quotes) του Codd:

Στην εικόνα 11.34 βλέπουμε ιστοσελίδα με σημαντικές φράσεις (quotes) του Ted Codd (quotes.html).


Welcome to Ted Codd Fun Club's Webpage

E. F. Codd Quotes

"The most important motivation for the research work that resulted in the relational model was the objective of providing a sharp and clear boundary between the logical and physical aspects of database management".

"Relational processing entails treating whole relationships as operands. Its primary purpose is loop-avoidance, an absolute requirement for end users to be productive at all, and a clear productivity booster for application programmers."

"It is no surprise that attempts such as those of CODASYL and ANSI to develop data structure language (DDL) and data manipulation language (DML) in separate communities have yielded many misunderstandings and incompatibilities."

E. F. Codd
Relational Database: A Practical Foundation for Productivity (1982). - Source

Εικόνα 11.34 Ιστοσελίδα με σημαντικές φράσεις (quotes) του Ted Codd

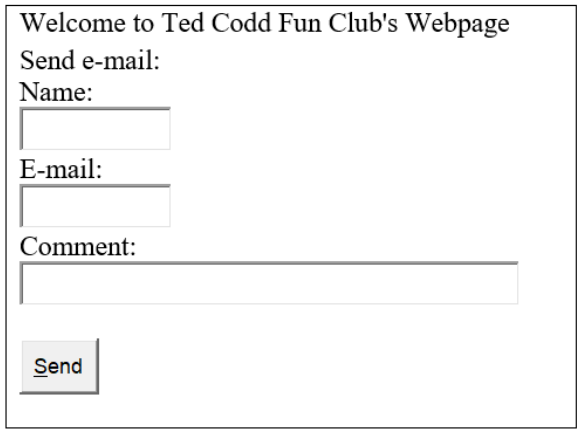
Παραθέτουμε τον κώδικα.

```
<!DOCTYPE html>
<html>
<head>
<title>Ted Codd’ s quotes</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
                  <div id="logo">
                     Welcome to Ted Codd Fun Club’s Webpage
                  </div>
<div>
      <h2>E. F. Codd Quotes </h2>
        <p>
<em>"The most important motivation for the research work that resulted in the
relational model </em> <br/>
<em> was the objective of providing a sharp and clear boundary between the
logical and physical aspects </em> <br/>
<em> of database management".</em> <br/><br/>
<em>"Relational processing entails treating whole relationships as operands.
</em> <br/>
<em> Its primary purpose is loop-avoidance, an absolute requirement for end
users </em> <br/>
<em> to be productive at all, and a clear productivity booster for application
programmers."</em> <br/><br/>
<em>"It is no surprise that attempts such as those of CODASYL and ANSI </em>
<br/>
<em> to develop data structure language (DDL) and data manipulation language
(DML) </em> <br/>
<em> in separate communities have yielded many misunderstandings and
incompatibilities."</em><br/><br/> E. F. Codd<br/>
```

```
<em>   Relational Database: A Practical Foundation for Productivity (1982).</em>
- Source<br/>
      </p>
 </div>
</body>
</html>
```

11.4.4 Ιστοσελίδα επικοινωνίας με τον ιστότοπο του Codd

Στην εικόνα 11.35 βλέπουμε την ιστοσελίδα επικοινωνίας με τον ιστότοπο του Codd (contact.html).



Εικόνα 11.35 Ιστοσελίδα επικοινωνίας με τον ιστότοπο του Codd

Προσοχή! Ο κώδικας που ακολουθεί απλά εμφανίζει τη σελίδα συγγραφής και αποστολής του μηνύματος. Για να στείλουμε πραγματικά το email απαιτείται περαιτέρω προγραμματισμός σε γλώσσα script. Για παράδειγμα, μπορείτε να χρησιμοποιήσετε γλώσσα PHP. Βέβαια η PHP απαιτεί ένα εγκατεστημένο και λειτουργικό σύστημα email.

Να και ο κώδικας εμφάνισης της σελίδας!

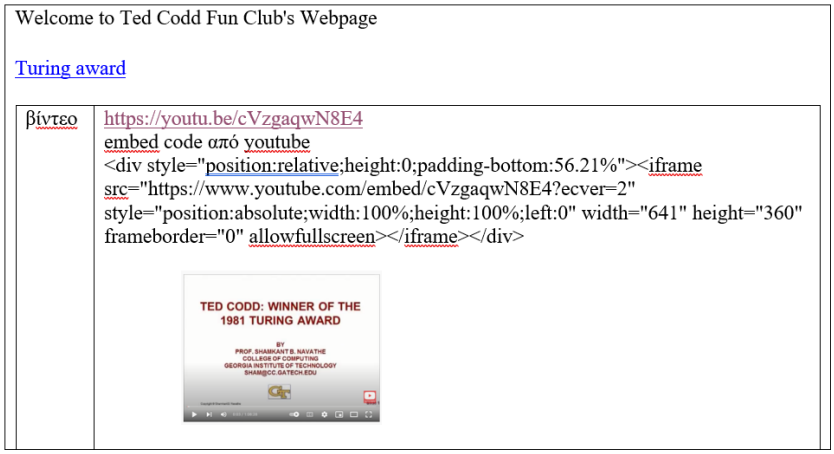
```
<!DOCTYPE html>
<html>
<head>
<title>contact</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<div id="logo">
Welcome to Ted Codd Fun Club’s Webpage
   </div>
<div>
<h2>Send e-mail:</h2>
<form action="mailto:someone@example.com" method="post" enctype="text/plain">
Name:<br>
<input type="text" name="name"><br>
E-mail:<br>
<input type="text" name="mail"><br>
Comment:<br>
<input type="text" name="comment" size="50"><br><br>
```

```
<input type="submit" value="Send">
</form>
</div>
</body>
</html>
```

11.4.4.1 Ιστοσελίδα που δείχνει ένα βίντεο σχετικό με τη βράβευση του Codd με το βραβείο Turing και έχει και σύνδεσμο για τη βράβευση αυτή

Στην εικόνα 11.36 βλέπουμε ιστοσελίδα με βίντεο από τη βράβευση του Codd με το βραβείο Turing και το σύνδεσμο για τη βράβευση (video.html). Στη εικόνα προσθέσαμε και τον τρόπο με τον οποίο ο κωδικός του βίντεο από τη σελίδα https://youtu.be/cVzgaqwN8E4 (youtube) είναι εμφυτευμένο (embed code) στην ιστοσελίδα μας:

```
<div style="position:relative;height:0;padding-bottom:56.21%"><iframe
src="https://www.youtube.com/embed/cVzgaqwN8E4?ecver=2"
style="position:absolute;width:100%;height:100%;left:0" width="641" height="360"
frameborder="0" allowfullscreen></iframe></div>
```



Εικόνα 11.36 Ιστοσελίδα με βίντεο από τη βράβευση του Codd με το βραβείο Turing. Στην εικόνα προστέθηκε και ο τρόπος «εμφύτευσης» του κωδικού του βίντεο σε youtube

Να και ο κώδικας!

```
<!DOCTYPE html>
<html>
<head>
<title>E.F.Codd video</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
  <div id="logo">
         Welcome to Ted Codd Fun Club's Webpage
  </div>
  <div>
     <p>
       <a href="http://amturing.acm.org/award_winners/codd_1000892.cfm">Turing
```

```
        award</a>
      </p>
      <iframe width="854" height="480" src=
      "https://www.youtube.com/embed/cVzgaqwN8E4" frameborder="0"
allowfullscreen>
      </iframe>
   </div>
</body>
</html>
```

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

Μια διεπαφή προγραμματισμού εφαρμογών, Application Programming Interface-API, παρέχει κλάσεις (classes), μεθόδους (methods), συναρτήσεις (functions) και μεταβλητές (variables) που είναι απαραίτητες για τον προγραμματισμό PHP εφαρμογών βάσεων δεδομένων. Οι διεπαφές προγραμματισμού–API μπορούν να είναι διαδικαστικές (procedural) ή αντικειμενοστρεφείς (object-oriented):

1) Procedural API , ο προγραμματιστής καλεί συναρτήσεις για να πραγματοποιήσει διάφορες διεργασίες (tasks).

2) Object-oriented API, ο προγραμματιστής δημιουργεί στιγμιότυπα (instantiate) κλάσεων και στη συνέχεια καλεί-εφαρμόζει μεθόδους στα προκύπτοντα αντικείμενα.

Για τις PHP εφαρμογές που χρειάζονται επικοινωνία με βάσεις δεδομένων, οι απαραίτητες διεπαφές API συνήθως παρέχονται μέσω PHP επεκτάσεων (PHP extensions). Γνωστές επεκτάσεις και αντίστοιχες διεπαφές API, είναι:

1) MySQL API, η επέκταση παρέχει μόνο διαδικαστική διεπαφή (procedural interface) και δε χρησιμοποιείται πλέον για νέες εφαρμογές

2) mysqli (MySQL improved) API, η διεπαφή παρέχει διαδικαστική διεπαφή (procedural interface) και αντικειμενοστρεφή διεπαφή (Object-oriented interface), υποστηρίζει Prepared Statements, πολλαπλές δηλώσεις (Multiple Statements), συναλλαγές (Transactions) κ.λπ. Ακολουθούν παραδείγματα κλάσεων και κάποιων μεθόδων των κλάσεων:

mysqli-The mysqli class

mysqli::$affected_rows — Gets the number of affected rows in a previous MySQL operation
mysqli::autocommit — Turns on or off auto-committing database modifications
mysqli::begin_transaction — Starts a transaction
mysqli::close — Closes a previously opened database connection
mysqli::commit — Commits the current transaction

mysqli::$connect_errno — Returns the error code from last connect call
mysqli::$connect_error — Returns a string description of the last connect error
mysqli::prepare — Prepare an SQL statement for execution
mysqli::query — Performs a query on the database

mysqli_stmt-The mysqli_stmt class

mysqli_stmt::close — Closes a prepared statement
mysqli_stmt::__construct — Constructs a new mysqli_stmt object
mysqli_stmt::data_seek — Seeks to an arbitrary row in statement result set
mysqli_stmt::$errno — Returns the error code for the most recent statement call
mysqli_stmt::$error — Returns a string description for last statement error
mysqli_stmt::execute — Executes a prepared Query

mysqli_result — The mysqli_result class

mysqli_result::data_seek — Adjusts the result pointer to an arbitrary row in the result
mysqli_result::fetch_all — Fetches all result rows as an associative array, a numeric array, or both

3) PHP Data Objects (PDO) API, η επέκταση προσφέρει μόνο αντικειμενοστρεφή διεπαφή (Object-oriented interface) και παρέχει στις PHP εφαρμογές την ανεξαρτησία από τον τύπο του διακομιστή βάσης δεδομένων στον οποίο θα συνδεθεί η εφαρμογή,.

Στην περίπτωση που χρησιμοποιούμε PHP Data Objects (PDO) API έχουμε ένα **επίπεδο αφαίρεσης βάσεων δεδομένων (database abstraction layer)** ειδικά για εφαρμογές PHP. Τι σημαίνει επίπεδο αφαίρεσης βάσεων δεδομένων:

Αν χρησιμοποιείτε το API του PDO, μπορείτε να αλλάξετε τον διακομιστή βάσης δεδομένων (database server), π.χ., αντί για τον MySQL διακομιστή να χρησιμοποιήσετε PostgreSQL database server, χωρίς να κάνετε πολλές αλλαγές στον κώδικα PHP.

Προσοχή! Κάθε φορά ανάλογα με τον διακομιστή βάσης δεδομένων (database server) που χρησιμοποιούμε πρέπει να χρησιμοποιούμε και το κατάλληλο πρόγραμμα οδήγησης (driver). Για παράδειγμα με MySQL database server χρησιμοποιούμε MySQL Driver.

11.5.1 Πρόγραμμα οδήγησης (driver)

Ένα πρόγραμμα οδήγησης (driver) είναι ένα λογισμικό σχεδιασμένο να επικοινωνεί με έναν συγκεκριμένο τύπο διακομιστή βάσης δεδομένων. Το πρόγραμμα οδήγησης μπορεί επίσης να καλέσει μια βιβλιοθήκη, όπως η βιβλιοθήκη MySQL Native Driver. Η βιβλιοθήκη υλοποιεί το πρωτόκολλο χαμηλού επιπέδου (low-level protocol) που χρησιμοποιείται για την επικοινωνία με το διακομιστή βάσης δεδομένων MySQL. Βλέπε και https://www.php.net/manual/en/set.mysqlinfo.php.

11.5.2 Παραδείγματα επιπέδων αφαίρεσης βάσεων δεδομένων (database abstraction layer)

Το PHP Data Objects, PDO, είναι ένα παράδειγμα επιπέδου αφαίρεσης βάσεων δεδομένων (database abstraction layer) ειδικά για τις εφαρμογές PHP. Άλλα παραδείγματα Επιπέδου αφαίρεσης βάσεων δεδομένων είναι το JDBC για εφαρμογές Java και το DBI για εφαρμογές Perl.

11.5.3 Σύγκριση MySQLi API και PDO API

Ο προγραμματιστής μπορεί να χρησιμοποιήσει από τα API τόσο το MySQLi API όσο και το PDO API.

Το MySQLi API είναι κάπως πιο ισχυρό αλλά και κάπως πιο πολύπλοκο στην εκμάθηση. Είναι προσαρμοσμένο στο προϊόν MySQL.

Ένα πλεονέκτημα του PDO API είναι ότι δεν περιορίζεται στο προϊόν mysql αλλά υποστηρίζει και άλλες μηχανές βάσεων δεδομένων όπως PostgreSQL κ.λπ.

11.6 Υλοποίηση εφαρμογών με τα προϊόντα PHP και MySQL. Χρήση προϊόντος XAMPP

Κατεβάζουμε (Download) και εγκαθιστούμε (Install) το προϊόν XAMPP. Προχωρούμε σε ενεργοποίηση του XAMPP control panel και εκκίνηση (START) των Apache, MySQL. Στα παραδείγματά μας χρησιμοποιήθηκε το XAMPP control panel v3.2.3

11.6.1 Δημιουργία βάσης δεδομένων και πινάκων

Πως εργαζόμαστε για τη δημιουργία βάσης δεδομένων. Αρχικά εκτελούμε τη mysql από το command.

```
# Σύνδεση με MySQL (cmd)
cd c:\xampp\mysql\bin
mysql -u root
```

Ακολουθεί η δημιουργία της βάσης δεδομένων grocery και του πίνακα grocery_inventory

```
DROP DATABASE IF EXISTS grocery;
CREATE DATABASE grocery;
USE grocery;
CREATE TABLE grocery_inventory(
  id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  item_name VARCHAR(50) NOT NULL,
  item_desc TEXT,
  item_price FLOAT NOT NULL,
  curr_qty INT NOT NULL );
```

11.6.2 Διαχείριση χρηστών

Ακολουθούν παραδείγματα (εικόνα 11.37) διαχείρισης χρηστών και των δικαιωμάτων τους.

`GRANT ALL`
`ON grocery`
`TO user IDENTIFIED BY '12<34';`
	Δημιουργία χρήστη user με συνθηματικό 12<34
	Ο user έχει όλα τα δικαιώματα, σύνδεσης, χρήσης πινάκων κ.λπ., στη βάση δεδομένων grocery

`REVOKE ALL`
`ON grocery`
`FROM user;`
	Αφαίρεση όλων των δικαιωμάτων από τον user

`GRANT ALL`
`ON grocery`
`TO christos IDENTIFIED BY '1234';`
	Δημιουργία χρήστη user με συνθηματικό 1234 και όλα τα δικαιώματα στη βάση δεδομένων grocery

`REVOKE ALL`
`ON grocery`
`FROM christos;`
	Αφαίρεση όλων των δικαιωμάτων από τον Christos

`GRANT ALL`
`ON grocery`
`TO christos@localhost`
`IDENTIFIED BY '1234';`
	Δημιουργία χρήστη christos@localhost με συνθηματικό 1234
	Ο χρήστης έχει όλα τα δικαιώματα, σύνδεσης στον τοπικό υπολογιστή μέσω διεύθυνσης IP 127.0.0.1 (localhost), χρήσης πινάκων κ.λπ., στη βάση δεδομένων grocery

`REVOKE ALL`
`ON grocery`
`FROM christos@localhost;`
	Αφαίρεση όλων των δικαιωμάτων από τον χρήστη

`GRANT SELECT, INSERT`
`ON grocery_inventory`
`TO christos IDENTIFIED BY '1234';`
	Δημιουργία χρήστη και εκχώρηση δικαιωμάτων χρήσης πίνακα

Εικόνα 11.37 Παραδείγματα διαχείρισης χρηστών και δικαιώματα

Για τη συνέχεια θα χρειαστεί ο χρήστης christos (εικόνα 11.38)

`GRANT ALL`
`ON 'grocery'.*`
`TO 'christos'@'localhost'`
`IDENTIFIED BY '1234';`
	Δημιουργία χρήστη με όλα τα δικαιώματα, σύνδεσης κ.λπ., στη βάση δεδομένων

Εικόνα 11.38 Δημιουργία χρήστη με όλα τα δικαιώματα

11.6.3 Εφαρμογές PHP διαχείρισης βάσεων δεδομένων. Αρχή με παράδειγμα

Θέλετε να δείτε πληροφορίες για την εγκατάσταση του προϊόντος PHP κ.λπ.

Χρησιμοποιήστε το notepad++ και γράψτε το έγγραφο phpinfo.php και τοποθετήστε το στο folder htdocs (βρίσκεται στο folder XAMPP).

Να ο κώδικας του script

```
<?php echo phpinfo(); ?>
```

Εκτελέστε το script στον browser

```
http://localhost/phpinfo.php ή http://localhost/phpinfo
```

Στην εικόνα 11.39 βλέπετε τις πληροφορίες για php μετά την εκτέλεση <?php echo phpinfo(); ?>

PHP Version 7.1.28

System	Πληροφορίες για το λειτουργικό σύστημα του υπολογιστή σας
Build Date	Apr 2 2021 17:19:31
Compiler	MSVC14 (Visual C++ 2015)
Architecture	x64
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--without-analyzer" "--with-pgo"
Server API	

Εικόνα 11.39Πληροφορίες για php <?php echo phpinfo(); ?>

11.6.3.1 Σχέση PHP και HTML - Αρχή με παράδειγμα.

Θα δείξουμε το μήνυμα: What a wonderful world!

Στον πίνακα 11.3 βλέπουμε τη σχέση PHP, HTML.

Πίνακας 11.3 Υλοποίηση ιστοσελίδας σε PHP, HTML

HTML: helloworld.html	PHP: helloworld.php
<html> <head> <title>A PHP script including HTML</title> </head> <body> What a wonderful world! ?> </body> </html>	<html> <head> <title>A PHP script including HTML</title> </head> <body> <?php echo "What a wonderful world!"; ?> </body> </html>

Να τι βλέπουμε.

http://localhost/helloworld.php
What a wonderful world!

Σημείωση

Θα μπορούσαμε να έχουμε την εφαρμογή μας σε ένα folder π.χ.. Examples, του folder htdocs. Τότε εκτελούμε το πρόγραμμά μας ως εξής:

http://localhost/Examples/helloworld.php

11.7 Κατασκευή εφαρμογής. Σύνδεση στη βάση και εισαγωγή δεδομένων

Στόχος μας στη συνέχεια είναι η κατασκευή μίας εφαρμογής λογισμικού βήμα-βήμα. Θα χρειαστεί όμως πρώτα να περιγράψουμε τον τρόπο σύνδεσης με τη βάση δεδομένων και την εισαγωγή στοιχείων. Θα ξεκινήσουμε με την περιγραφή του τρόπου σύνδεσης με τη βάση δεδομένων.

11.7.1 Σύνδεση στη βάση (Making a connection) με PHP

Στη συνέχεια παραθέτουμε τον κώδικα του προγράμματος. Η εκτέλεση του προγράμματος γίνεται ως εξής:

http://localhost/mysqlconnect.php

```
<?php
$mysqli = mysqli_connect("localhost", "christos", "1234",
"grocery");

if (mysqli_connect_errno()) {
      printf("Connect failed: %s\n",
mysqli_connect_error());
      exit();
} else {
      printf("Host information: %s\n",
mysqli_get_host_info($mysqli));
      mysqli_close($mysqli);
}
?>
```

Η απάντηση που θα δούμε είναι:

```
Host information: localhost via TCP/IP
```

Ακολουθεί συζήτηση του προγράμματος:

Δημιουργία σύνδεσης

Χρησιμοποιούμε τη συνάρτηση mysqli_connect για να ορίσουμε μία σύνδεση με τη βάση δεδομένων. Η βασική σύνταξη της σύνδεσης είναι:

```
$mysqli = mysqli_connect("hostname", "username", "password", "database");
```

Στην περίπτωσή μας έχουμε:

```
Hostname=localhost
Username=Christos
Password=1234
Database=grocery
```

Η συνάρτηση mysqli_connect_errno() επιστρέφει τον κωδικό του σφάλματος από την τελευταία κλήση σύνδεσης.

Έλεγχος σύνδεσης

Ελέγχουμε αν έχουμε σφάλμα σύνδεσης. Η δομή ελέγχου είναι:

```
if (mysqli_connect_errno()) { διαχείριση σφάλματος σύνδεσης  }
else
{  διαχείριση επιτυχούς σύνδεσης  }
```

Η συνάρτηση mysqli_connect_error() επιστρέφει το μήνυμα για το σφάλμα σύνδεσης.

Για παράδειγμα, αν υπάρχει λάθος τιμή στις παραμέτρους της σύνδεσης μπορεί να έχουμε μήνυμα σφάλματος όπως το παρακάτω:

```
   Access denied for user (using password: YES)
```

Η συνάρτηση printf("Connect failed: %s\n", mysqli_connect_error()); εμφανίζει τη φράση Connect failed: και τη συμβολοσειρά (string) του μηνύματος σφάλματος που επιστρέφει η συνάρτηση mysqli_connect_error()

Η συνάρτηση exit() προκαλεί έξοδο από το πρόγραμμα.

Διαχείριση επιτυχούς σύνδεσης

Όταν η σύνδεση είναι επιτυχής χρησιμοποιούμε τη συνάρτηση mysqli_get_host_info() που δίνει στοιχεία για τη σύνδεση. Η συνάρτηση καλείται με παράμετρο τα στοιχεία της σύνδεσης: mysqli_get_host_info($mysqli)

Η συνάρτηση mysqli_close() κλείνει τη σύνδεση.

Παραδείγματα σύνδεσης σε βάση δεδομένων με αντικειμενοστρεφή τρόπο: mysqli, pdo

Παραθέτουμε το πρόγραμμα pdo_index.php (Χρήση PDO API)

```
<?php
// PDO
$pdo = new PDO("mysql:host=localhost;dbname=grocery", "christos",
"1234");
$statement = $pdo->query("SELECT 'Hello, MySQL PDO user!' AS
_message FROM DUAL");
$row = $statement->fetch(PDO::FETCH_ASSOC);
echo htmlentities($row['_message']);
?>
```

Εκτέλεση κώδικα: localhost/pdo_index.php Αποτέλεσμα: Hello, MySQL PDO user!

Παραθέτουμε το πρόγραμμα mysqli_index.php (Χρήση Mysqli API)

```
<?php
// mysqli
$mysqli = new mysqli("localhost", "christos", "1234", "grocery");
$result = $mysqli->query("SELECT 'Hello, MySQLi user!' AS _message
FROM DUAL");
$row = $result->fetch_assoc();
echo htmlentities($row['_message']);
?>
```

Εκτέλεση κώδικα: localhost/mysqli_index.php Αποτέλεσμα: Hello, MySQLi user!

11.7.2 Εισαγωγή στοιχείων (Inserting data with PHP)

Θέλουμε να εισάγουμε μία γραμμή στον πίνακα:

Id	Item_name	Item_desc	Item_price	Curr_qty
	APPLES	First class apples	2	30

Παραθέτουμε το πρόγραμμα. Για την εκτέλεση έχουμε: http://localhost/mysqlinsert.php

```
<?php
$mysqli = mysqli_connect("localhost", "christos", "1234",
"grocery");

if (mysqli_connect_errno()) {
      printf("Connect failed: %s\n", mysqli_connect_error());
      exit();
} else {
      printf("Host information: %s\n",
              mysqli_get_host_info($mysqli));

      $sql = "INSERT INTO grocery_inventory (item_name, item_desc,
                      item_price, curr_qty)
              VALUES ('APPLES','First class apples', 2, 30 )";
      $res = mysqli_query($mysqli, $sql);

      if ($res === TRUE) {
            echo "A record has been inserted.";
      } else {
            printf("Could not insert record: %s\n",
            mysqli_error($mysqli));
      }
      mysqli_close($mysqli);
}
?>
```

Η απάντηση που θα δούμε είναι:

```
Host information: localhost via TCP/IP A record has been inserted
```

Ακολουθεί συζήτηση του προγράμματος.

Στην αρχή συνδεόμαστε και κάνουμε έλεγχο σύνδεσης όπως προηγουμένως.

Αν έχουμε επιτυχή σύνδεση κατασκευάζουμε τη μεταβλητή $sql που έχει τιμή τη συμβολοσειρά (string) της δήλωσης INSERT INTO …

```
"INSERT INTO grocery_inventory (item_name, item_desc, item_price, curr_qty)
VALUES ('APPLES','First class apples', 2, 30) )"
```

Δεν είναι απαραίτητο η δήλωση INSERT INTO να εκχωρείται σε μεταβλητή (εδώ στην $sql) αλλά προσωπικά θεωρούμε ότι ο κώδικας γίνεται πιο ευανάγνωστος.

Παρατηρήστε ότι η δήλωση INSERT INTO … δεν τελειώνει με τον χαρακτήρα ;

Η συνάρτηση mysqli_query() εκτελεί τη δήλωση INSERT INTO. Παράμετροι της συνάρτησης είναι η σύνδεση και η δήλωση:

```
 mysqli_query($mysqli, $sql)
```

Η mysqli_query() επιστρέφει TRUE/FALSE.

Ελέγχουμε την εκτέλεση της δήλωσης ως εξής:

```
$res = mysqli_query(σύνδεση, δήλωση SQL);
      if ($res === TRUE) { μήνυμα επιτυχούς εισαγωγής γραμμής στον πίνακα }
  else { μήνυμα προβλήματος στην εισαγωγή της γραμμής στον πίνακα }
```

Η συνάρτηση mysqli_close() κλείνει τη σύνδεση.

Σημείωση για τη διαχείριση SQL δηλώσεων με PHP

Για τις δηλώσεις SQL, INSERT/UPDATE/DELETE αλλά και CREATE TABLE κ.λπ. ακολουθούμε το σχήμα που παρατέθηκε παραπάνω:

```
Σύνδεση στη βάση δεδομένων
$sql = " δήλωση SQL ";
$res = mysqli_query(σύνδεση, δήλωση SQL);
      if ($res === TRUE)
   { μήνυμα επιτυχούς εισαγωγής γραμμής στον πίνακα }
  else { μήνυμα προβλήματος στην εισαγωγή της γραμμής στον πίνακα }
Κλείσιμο σύνδεσης
```

11.8 Παρουσίαση μιας απλής εφαρμογής με χρήση HTML, PHP και MySQL

Με το προηγούμενο πρόγραμμα mysqlinsert.php έγινε εισαγωγή μίας γραμμής στον πίνακα και ουσιαστικά στο πρόγραμμα χρησιμοποιήθηκε η δήλωση

```
INSERT INTO grocery_inventory (item_name, item_desc, item_price, curr_qty)
VALUES ('APPLES','First class apples', 2, 30));
```

Σε πραγματικές εφαρμογές ο χρήστης θα μπορούσε να βλέπει αρχικά ένα μενού (εικόνα 11.40) Το συγκεκριμένο μενού θα αναλάβει να το εμφανίζει ο κώδικας μας. Θα ονομάσουμε την ιστοσελίδα (το πρόγραμμά) μας menu.php ή menu.html.

Αν ο χρήστης πατήσει το κουμπί INSERT RECORD θα καλείται μία σελίδα (με όνομα insert_form.php ή insert__form.html) που θα του επιτρέπει να πληκτρολογήσει τα στοιχεία είδους.

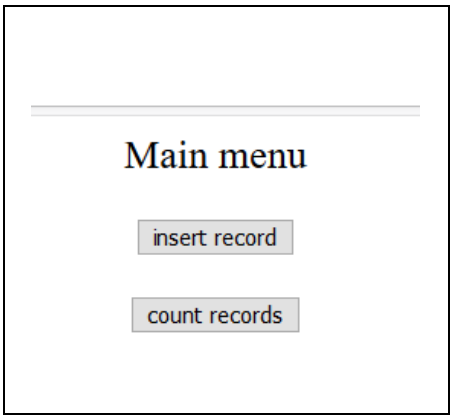
Η σελίδα αυτή θα καλεί στη συνέχεια τη σελίδα insert.php η οποία θα διαβάζει τις τιμές που πληκτρολόγησε ο χρήστης και θα εισάγει τα στοιχεία του πίνακα με παραμετρική δήλωση. Παραθέτουμε τη δήλωση:

```
"INSERT INTO grocery_inventory(item_name, item_desc,  item_price, curr_qty)
      VALUES ('".$_POST["item_name"]."', '".$_POST["item_description"]."',
'".$_POST["item_price"]."', '".$_POST["item_quantity"]."' )"
```

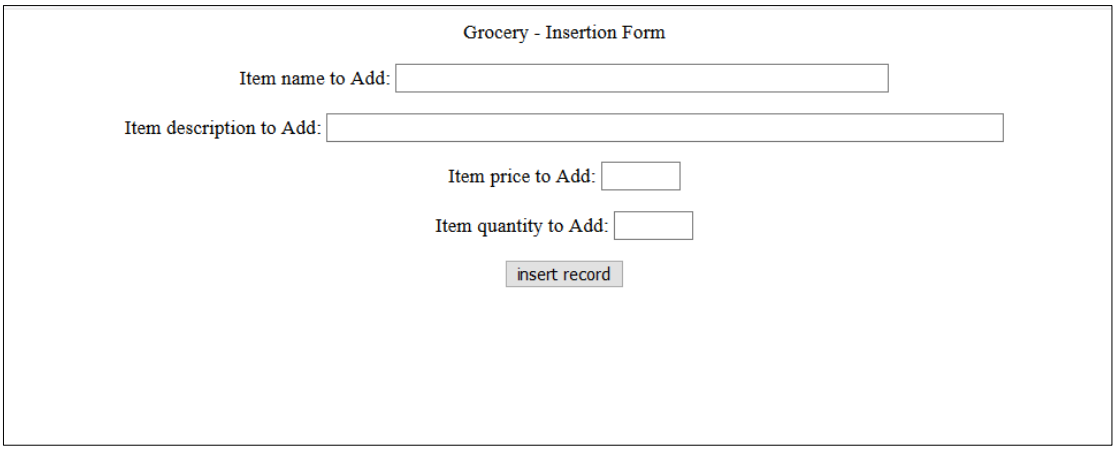
Μετά την εισαγωγή των στοιχείων η σελίδα insert.php θα επιστρέφει στο μενού.

Αν ο χρήστης πατήσει το κουμπί COUNT RECORDS θα καλείται το πρόγραμμα count.php και θα εμφανίζει πόσες γραμμές έχει ο πίνακας.

Στην εικόνα 11.40 βλέπουμε το κυρίως μενού και στην εικόνα 11.41 την αρχική διεπαφή χρήστη (User Interface), για την εφαρμογή μας.



Εικόνα 11.40 Μενού εφαρμογής



Εικόνα 11.41 Διεπαφή χρήστη για την εισαγωγή δεδομένων

11.9 Κατασκευή της εφαρμογής με χρήση HTML, PHP MySQL

Θα ξεκινήσουμε με την κατασκευή του μενού.

11.9.1 Κατασκευή μενού

Το μενού μπορεί να υλοποιηθεί με χρήση html με πολλούς τρόπους. Ακολουθεί μια απλουστευμένη λύση.

Να πως εκτελείται.

http://localhost/menu.php

Ακολουθεί ο κώδικας εμφάνισης του Menu στην ιστοσελίδα (menu.php)

```
<html>
<head>
<title>MENU</title>
</head>
<body>
<center>
   <font size="+2" Menu </font>
   Main menu
<form name=insert" action="insert_form.php" method="POST">
<p><input type="submit" name="submit" value="insert record"></p>
```

```
</form>
<form name=retrieve" action="count.php" method="POST">
<p><input type="submit" name="submit" value="count records"></p>
</form>
</body>
</html>
```

Ο κώδικας θα εμφανίσει το μενού της εικόνας 11.40.

Συζήτηση του κώδικα

Η σελίδα του μενού περιλαμβάνει τρία δομικά στοιχεία:

Ένα κείμενο κεντραρισμένο στη σελίδα: Main menu
Φόρμα με όνομα insert και ένα κουμπί με όνομα insert name. Όταν πατηθεί το κουμπί καλείται η σελίδα insert_form.php
Φόρμα με όνομα retrieve και ένα κουμπί με όνομα count records. Όταν πατηθεί το κουμπί καλείται η σελίδα count.php

11.9.2 Κατασκευή σελίδας εισαγωγή στοιχείων

Η σελίδα **insert_form.php** χρησιμοποιείται από τον χρήστη για την εισαγωγή στοιχείων είδους.

Ακολουθεί ο κώδικας της σελίδας insert_form.php.

```
<html>
 <head>
   <title>Insertion Form</title>
 </head>
 <body>
    <center>
       Grocery - Insertion Form

    </center>
       <form action="insert.php" method="POST">
        <p>Item name to Add:
        <input type="text" name="item_name" size="50"></p>
        <p>Item description to Add:
        <input type="text" name="item_description" size="70"></p>
        <p>Item price to Add:
        <input type="text" name="item_price" size="5"></p>
        <p>Item quantity to Add:
        <input type="text" name="item_quantity" size="5">
        <p><input type="submit" name="submit" value="insert record"></p>
       </form>
 </body>
</html>
```

Ο κώδικας θα εμφανίζει την ιστοσελίδα της εικόνας 1.41.

Συζήτηση του κώδικα

Η σελίδα περιλαμβάνει δύο δομικά στοιχεία:

Ένα κείμενο κεντραρισμένο στη σελίδα: Grocery - Insertion Form
Φόρμα που εμφανίζει τέσσερα κουτιά. Δίπλα στα κουτιά υπάρχουν κατά σειρά τα μηνύματα: Item name to Add: Item description to Add: Item price to Add: Item quantity to Add: Τα στοιχεία που πληκτρολογεί ο χρήστης αποθηκεύονται κατά σειρά στις παραμέτρους: item_name, item_description, item_price, item_quantity Η φόρμα εμφανίζει και ένα κουμπί με όνομα insert name. Όταν πατηθεί το κουμπί καλείται η σελίδα insert.php στην οποία στέλνονται με χρήση της μεθόδου POST οι τιμές (string) που πληκτρολόγησε ο χρήστης στα κουτιά.

11.9.3 Κατασκευή σελίδας που διαβάζει τις τιμές που πληκτρολόγησε και τις γράφει στη βάση δεδομένων

Η σελίδα **insert.php** χρησιμοποιείται για να διαβάσει τις τιμές (string) που πληκτρολόγησε ο χρήστης στα κουτιά της προηγούμενης σελίδας και στη συνέχεια να τις γράψει στη βάση δεδομένων.

Παραθέτουμε τον κώδικα της σελίδας insert.php.

```
<?php
    $mysqli = mysqli_connect("localhost", "christos", "1234", "grocery");
    if (mysqli_connect_errno()) {
      printf("Connect failed: %s\n", mysqli_connect_error());
      exit();
    } else {
      $sql = "INSERT INTO grocery_inventory(item_name, item_desc,  item_price,
curr_qty)
       VALUES ('".$_POST["item_name"]."', '".$_POST["item_description"]."',
'".$_POST["item_price"]."', '".$_POST["item_quantity"]."' )";
      $res = mysqli_query($mysqli, $sql);
      if ($res === TRUE) {
            echo "A record has been inserted.";
      } else {
            printf("Could not insert record: %s\n", mysqli_error($mysqli));
      }
      mysqli_close($mysqli);
      }
?>
<html>
<body>
<center>
form name="goBack" method="post" action="menu.php">
<script>
        document.forms["goBack"].submit();
 </script>
      </form>
  </body>
</html>
```

Συζήτηση του κώδικα

Στην αρχή συνδεόμαστε και κάνουμε έλεγχο σύνδεσης όπως προηγουμένως.

Αν έχουμε επιτυχή σύνδεση κατασκευάζουμε τη μεταβλητή $sql που έχει τιμή τη συμβολοσειρά (string) της δήλωσης INSERT INTO …

Η επικοινωνία των δύο σελίδων για να διαβάσουμε τις τιμές που πληκτρολόγησε ο χρήστης φαίνεται στην εικόνα 11.42.

Αποστολή	Ανάγνωση
<input type="text" name="item_name" κ.λπ	$_POST["item_name"]
<input type="text" name="item_description" κ.λπ	$_POST["item_description"]
<input type="text" name="item_price" κ.λπ	$_POST["item_price"]
<input type="text" name="item_quantity" κ.λπ	$_POST["item_quantity"]

Εικόνα 11.42 Η επικοινωνία των δύο σελίδων, της σελίδας εισαγωγής δεδομένων από τον χρήστη και της σελίδας που καταχωρεί στη βάση δεδομένων

Υποτίθεται ότι οι τιμές αυτές (string) είναι οι παρακάτω:

$_POST["item_name"]	'LEMONS'
$_POST["item_description"]	'First class lemons'
$_POST["item_price"]	'3'
$_POST["item_quantity"]	'50'

Σε μία σελίδα όπως η σελίδα mysqlinsert.php που μελετήσαμε προηγουμένως θα γράφαμε:

```
$sql ="INSERT INTO grocery_inventory (item_name, item_desc, item_price,
curr_qty) VALUES ('APPLES','First class apples', '3', '50'))";
```

Η καταγραφή της δήλωσης INSERT INTO στη μεταβλητή $sql στην περίπτωσή μας θα γίνει με την παράθεση (concatenation) επιμέρους συμβολοσειρών (string). Για την παράθεση χρησιμοποιείται ο χαρακτήρας '.'

```
"INSERT INTO grocery_inventory(item_name, item_desc,  item_price, curr_qty)
      VALUES ('".$_POST["item_name"]."', '".$_POST["item_description"]."',
'".$_POST["item_price"]."', '".$_POST["item_quantity"]."' )"
```

Δείτε και την εικόνα 11.43.

```
"INSERT INTO grocery_inventory(item_name, item_desc, item_price, curr_qty) VALUES ('"
.
$_POST["item_name"]
.
"', '"
.
$_POST["item_description"]
.
"', '"
.
$_POST["item_price"]
.
"', '"
.
$_POST["item_quantity"]
.
"')"
```

Εικόνα 11.43 Πως γράφουμε τη δήλωση *INSERT INTO* στη *μεταβλητή* $sql

Τέλος, δείτε στον κώδικα πως η σελίδα επιστρέφει στο μενού.

```
<html>
<body>
<center>
<form name="goBack" method="post" action="menu.php">
<script>
        document.forms["goBack"].submit();
 </script>
      </form>
  </body>
</html>
```

11.9.4 Κατασκευή σελίδας που εμφανίζει πόσα είδη υπάρχουν στη βάση δεδομένων

Η σελίδα **count.php** χρησιμοποιείται για να δείξει πόσα είδη υπάρχουν στη βάση δεδομένων.

Παραθέτουμε τον κώδικα της σελίδας count.php

```
<?php
$mysqli = mysqli_connect("localhost", "christos", "1234", "grocery");
if (mysqli_connect_errno()) {
      printf("Connect failed: %s\n", mysqli_connect_error());
      exit();
} else {
      $sql = "SELECT * FROM grocery_inventory";
      $res = mysqli_query($mysqli, $sql);
      if ($res) {
            $number_of_rows = mysqli_num_rows($res);
            printf("Result set has %d rows.\n", $number_of_rows);
      } else {
```

```
            printf("Could not retrieve records: %s\n", mysqli_error($mysqli));
      }
      mysqli_free_result($res);
      mysqli_close($mysqli);
}
?>
```

Συζήτηση του κώδικα.

Στην αρχή συνδεόμαστε και κάνουμε έλεγχο σύνδεσης.

Αν έχουμε επιτυχή σύνδεση κατασκευάζουμε τη μεταβλητή $sql που έχει τιμή τη συμβολοσειρά (string) της δήλωσης SELECT * FROM grocery_inventory

Θυμίζουμε ότι δεν είναι απαραίτητο η δήλωση SELECT να εκχωρείται σε μεταβλητή αλλά γενικά θεωρούμε αυτήν την πρακτική καλή για να έχουμε ευανάγνωστο κώδικα.

Παρατηρήστε ότι η δήλωση SELECT * FROM grocery_inventory δεν τελειώνει με τον χαρακτήρα ‘;’

Η συνάρτηση mysqli_query() εκτελεί τη δήλωση SELECT. Παράμετροι της συνάρτησης είναι η σύνδεση και η δήλωση:

```
mysqli_query($mysqli, $sql)
```

Η mysqli_query() επιστρέφει το σύνολο αποτελεσμάτων (result set) $res.

Ελέγχουμε την εκτέλεση της δήλωσης ως εξής:

```
$res = mysqli_query(σύνδεση, δήλωση SQL);
   if ($res) {
```

Κλήση συνάρτησης mysqli_num_rows που επιστρέφει τον αριθμό των ανακτηθεισών γραμμών που υπάρχουν στο σύνολο των αποτελεσμάτων $res

Εμφάνιση μηνύματος με τον αριθμό των γραμμών του πίνακα }

```
   else { μήνυμα προβλήματος στην ανάκτηση των γραμμών του πίνακα }
```

Η συνάρτηση `mysqli_free_result()` ελευθερώνει τη μνήμη που συνδέεται με την αναζήτηση (query) και τα αποτελέσματά της για να μπορεί να χρησιμοποιηθεί από άλλα scripts.

Η συνάρτηση `mysqli_close()` κλείνει τη σύνδεση.

11.10 Δύο είδη διαχείρισης ανάλογα με τη δήλωση SQL

Για τις δηλώσεις SQL ακολουθούνται δύο είδη διαχείρισης όπως φαίνεται στην εικόνα 11.44.

δηλώσεις INSERT/UPDATE/DELETE/CREATE κ.λπ
Σύνδεση στη βάση δεδομένων $sql = " δήλωση SQL "; $res = mysqli_query(σύνδεση, δήλωση SQL); if ($res === TRUE) { μήνυμα επιτυχούς ενέργειας, π.χ., εισαγωγή γραμμής στον πίνακα } else { μήνυμα προβλήματος, π.χ., στην εισαγωγή της γραμμής στον πίνακα } Κλείσιμο σύνδεσης

δηλώσεις SELECT
Σύνδεση στη βάση δεδομένων $sql = " δήλωση SELECT"; $res = mysqli_query(σύνδεση, δήλωση SQL); if ($res) { διαχείριση συνόλου αποτελεσμάτων, π.χ. με κλήση συνάρτησης mysqli_num_rows() ή διαχείριση των ανακτηθεισών γραμμών του συνόλου με while } else { μήνυμα προβλήματος στην ανάκτηση πίνακα } Ελευθέρωση μνήμης Κλείσιμο σύνδεσης

Εικόνα 11.44 Είδη διαχείρισης ανάλογα με τη δήλωση SQLl

11.11 Παρουσίαση διαχείρισης συνόλου αποτελεσμάτων με βρόχο

Με το προηγούμενο πρόγραμμα η δήλωση SELECT επιστρέφει το σύνολο αποτελεσμάτων $res και με χρήση της συνάρτησης mysqli_num_rows() εμφανίζουμε πόσες γραμμές έχει το σύνολο. Στη συνέχεια παραθέτουμε το πρόγραμμα select.php. Στο πρόγραμμα δείχνουμε τα αποτελέσματα (τις γραμμές) με χρήση βρόχου. Παραθέτουμε τον κώδικα.

```
<?php
$mysqli = mysqli_connect("localhost", "christos", "1234",
"grocery");

if (mysqli_connect_errno()) {
      printf("Connect failed: %s\n", mysqli_connect_error());
      exit();
} else {
      $sql = "SELECT * FROM grocery_inventory";
      $res = mysqli_query($mysqli, $sql);

      if ($res) {
      while ($newArray = mysqli_fetch_array($res, MYSQLI_BOTH)) {
      $id  = $newArray['id'];
  $item_name = $newArray['item_name'];
  $item_desc = $newArray['item_desc'];
  $item_price = $newArray['item_price'];
  $curr_qty = $newArray['curr_qty'];
      echo "The ID is ". $id.
" the name is ".$item_name.
" the description is ".$item_desc.
" the price is ".$item_price.
" and the current_qty is ".$curr_qty. "<br/>";
            }
      } else {
      printf("Could not retrieve records: %s\n",
mysqli_error($mysqli));
      }

      mysqli_free_result($res);
      mysqli_close($mysqli);
}
?>
```

Για την εκτέλεση πληκτρολογείτε:

```
http://localhost/select.php
```

Συζήτηση του κώδικα

Στην αρχή συνδεόμαστε και κάνουμε έλεγχο σύνδεσης.

Αν έχουμε επιτυχή σύνδεση κατασκευάζουμε τη μεταβλητή $sql που έχει τιμή τη συμβολοσειρά (string) της δήλωσης SELECT * FROM grocery_inventory

Η συνάρτηση mysqli_query() εκτελεί τη δήλωση SELECT. Παράμετροι της συνάρτησης είναι η σύνδεση και η δήλωση:

```
mysqli_query($mysqli, $sql)
```

Η mysqli_query() επιστρέφει το σύνολο αποτελεσμάτων (result set) $res.

Ελέγχουμε την εκτέλεση της δήλωσης ως εξής:

```
$res = mysqli_query(σύνδεση, δήλωση SQL);
if ($res) {
```

Χρησιμοποιούμε βρόχο while

Μέσα στον βρόχο παίρνουμε διαδοχικά τις γραμμές του συνόλου αποτελεσμάτων (result set)

Για κάθε γραμμή η τιμή κάθε στήλης της γράφεται σε ξεχωριστή μεταβλητή.

Τέλος, δείχνουμε τα αποτελέσματα

```
   }
 else { μήνυμα προβλήματος στην ανάκτηση των γραμμών του πίνακα }
```

Η συνάρτηση mysqli_free_result() ελευθερώνει τη μνήμη που συνδέεται με την αναζήτηση (query) και τα αποτελέσματά της.

Η συνάρτηση mysqli_close() κλείνει τη σύνδεση.

11.11.1 Επεξήγηση της δομής ελέγχου while και της συνάρτησης mysqli_fetch_array

Η δομή ελέγχου που χρησιμοποιούμε είναι βρόχος while:

```
while ($newArray = mysqli_fetch_array($res, MYSQLI_ASSOC))
{
}
```

Η συνάρτηση mysqli_fetch_array($res, MYSQLI_ASSOC) χρησιμοποιείται στον βρόχο για να ανακαλεί κάθε φορά μία γραμμή του συνόλου αποτελεσμάτων (result set) σαν array.

Η σύνταξη της συνάρτησης mysqli_fetch_array() είναι:

```
mysqli_fetch_array(result, result_type);
```

Η παράμετρος result είναι υποχρεωτική και ορίζει ένα αναγνωριστικό συνόλου αποτελεσμάτων (a result set identifier) που επιστρέφεται από συναρτήσεις όπως mysqli_query(), mysqli_store_result() ή mysqli_use_result().

Η παράμετρος result_type είναι προαιρετική και καθορίζει τον τύπο του array. Πιο συγκεκριμμένα μπορεί να πάρει τιμές:

```
MYSQLI_ASSOC - έχει τιμές συμβολοσειρές ("associative array")
MYSQLI_NUM - έχει τιμές αριθμητικές ("numeric array")
MYSQLI_BOTH - έχει τιμές συμβολοσειρές και αριθμητικές.
```

11.12 Κατασκευή πίνακα με χρήση PHP και MySQL

Στη συνέχεια παραθέτουμε πρόγραμμα δημιουργίας πίνακα με χρήση PHP και MySQL

```
<?php
$mysqli = mysqli_connect("localhost", "christos", "1234",
"grocery");

if (mysqli_connect_errno()) {
      printf("Connect failed: %s\n", mysqli_connect_error());
      exit();
} else {
      $sql = "CREATE TABLE my_table(id int not null AUTO_INCREMENT,
       tab_col VARCHAR(50), PRIMARY KEY(id))";
      $res = mysqli_query($mysqli, $sql);

      if ($res === TRUE) {
            echo "Table my_table successfully created.";
      } else {
            printf("Could not create table: %s\n",
mysqli_error($mysqli));
      }

      mysqli_close($mysqli);
}
?>
```

Μπορούμε να δούμε τα δικαιώματα των χρηστών της εφαρμογής ή/και να εκχωρήσουμε δικαιώματα. Στο παράδειγμα εκχωρούμε όλα τα δικαιώματα στον χρήστη christos:

```
SHOW GRANTS FOR christos;
GRANT ALL PRIVILEGES ON `grocery`.*
TO 'christos'@'localhost'
IDENTIFIED BY '1234' ;
```

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

Σύνοψη
Στο κεφάλαιο αυτό περιγράφονται και επεξηγούνται εφαρμογές JSP Pages με χρήση JDBC API. Περιλαμβάνονται οι παρακάτω ενότητες και είναι καλό να μελετηθούν και να δοκιμαστούν και σε υπολογιστή:

- *Εισαγωγή με παράδειγμα στις δυναμικές Java Server Pages (JSP) σελίδες. Κατασκευή απλοποιημένου Calculator.*

  Η παρουσίαση της τεχνολογίας των δυναμικών JSP σελίδες γίνεται με τη βοήθεια ενός παραδείγματος στο οποίο κατασκευάζεται βήμα-βήμα μία αριθμομηχανή (Calculator) των τεσσάρων βασικών πράξεων της αριθμητικής. Η κατασκευή γίνεται στο προϊόν Netbeans. Παρουσιάζεται ο έλεγχος ορθότητας των δεδομένων που πληκτρολογεί ο χρήστης με χρήση try/catch Block.

- *Εισαγωγή με παράδειγμα στη σύνδεση δυναμικών σελίδων JSP και βάσης δεδομένων. Κατασκευή Login.*

  Κατασκευάζουμε βήμα-βήμα μια απλή εφαρμογή σύνδεσης (login) σε ιστοσελίδα. Στην εφαρμογή μας οι κωδικοί των χρηστών και τα συνθηματικά τους για την εφαρμογή αποθηκεύονται σε βάση δεδομένων. Περιγράφεται αναλυτικά με τη βοήθεια του παραδείγματος η σύνδεση των δυναμικών σελίδων JSP και της βάσης δεδομένων και παρατίθεται "To-do list" και τρόπος διαμόρφωσης σελίδων με χρήση CSS.

- *Εισαγωγή με παράδειγμα στις εφαρμογές Δυναμικών Ιστοσελίδων (JSP pages, mySQL).*

  Παρουσιάζεται η υλοποίηση συστήματος πωλήσεων της εταιρείας Mythical Car, μιας (υποθετικής) εταιρείας πώλησης αυτοκινήτων. Η εταιρεία περιγράφεται σύντομα, δημιουργούνται οι κατάλληλοι πίνακες και κατασκευάζεται η διεπαφή μέσω της οποίας ο πωλητής καταγράφει τις πωλήσεις αυτοκινήτων (jsp σελίδες). Κατασκευάζονται, επιπλέον, οι απαραίτητοι triggers.

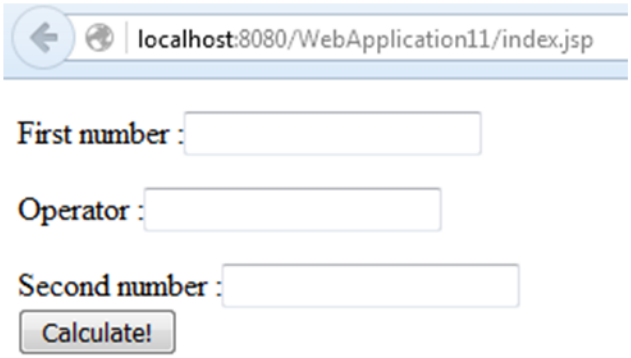
- *Επισκόπηση της χρήσης jDBC API. Μελέτη Περίπτωσης. Διαχείριση προσωπικού με εφαρμογή Δυναμικών Ιστοσελίδων και χρήση τεχνολογίας JSP pages και mySQL*

  Παρατίθεται μία επισκόπηση της χρήσης του JDBC API για τη δημιουργία εφαρμογών διαχείριση βάσεων δεδομένων με τεχνολογία JSP PAGES. Στη συνέχεια παρουσιάζεται Μελέτη Περίπτωσης με αναφορά στην επισκόπηση. Μετά τη μελέτη της ενότητας και την επεξεργασία της Μελέτης Περίπτωσης ο αναγνώστης θα έχει κατανοήσει τα θέματα δημιουργίας μιας εφαρμογής τεχνολογίας JSP PAGES. Η εφαρμογή μπορεί εύκολα να επεκταθεί και να βελτιωθεί σε διάφορες κατευθύνσεις.

Προαπαιτούμενα
Προτείνεται η μελέτη του κεφαλαίου 11 του παρόντος συγγράμματος.

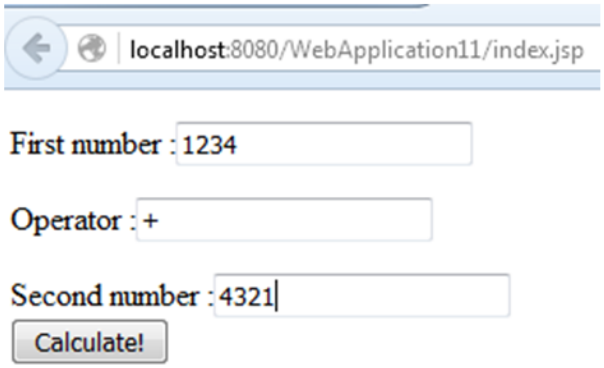
12.1 Εισαγωγή στις δυναμικές JSP σελίδες με παράδειγμα

Στην ενότητα αυτή θα κατασκευάσουμε μία απλοποιημένη αριθμομηχανή (Calculator) η οποία μας επιτρέπει να κάνουμε πρόσθεση (addition) δύο ακεραίων. Στην αρχή θα κατασκευάσουμε τη σελίδα index.html/index.jsp η οποία εμφανίζει μία ηλεκτρονική φόρμα (εικόνα 12.1).



Εικόνα 12.1 Ηλεκτρονική φόρμα για πράξη με δύο αριθμούς

Στη φόρμα αυτή μπορούμε να πληκτρολογήσουμε τους δύο αριθμούς και την πράξη. (εικόνα 12.2)



Εικόνα 12.2 Ηλεκτρονική φόρμα για την πρόσθεση δύο αριθμών

Θα αναλύσουμε βήμα-βήμα την κατασκευή της σελίδας index.html/index.jsp η οποία περιλαμβάνει στον κώδικά της τη φόρμα της εικόνας 12.1. Να η πρώτη γραμμή της φόρμας:

```
<form name="Formname" method="post" action="calc.jsp" >
```

Το όνομα της φόρμας είναι Formname. Η μέθοδος είναι "post" και η ενέργειά της (action) είναι η κλήση της σελίδας "calc.jsp"

Στη φόρμα υπάρχουν τρία κουτιά εισαγωγής κειμένου (text boxes) και ένα κουμπί.

Να τι κάνει η ακόλουθη γραμμή κώδικα:

```
First number :<input type="text" name="nu1" >
```

Εμφανίζει την οδηγία/λεκτικό (caption) “First number :” και δίπλα το κουτί κειμένου (input type="text"). Ο αριθμός που πληκτρολογεί ο χρήστης στο κουτί αποθηκεύεται στη μεταβλητή nu1.

Να τι κάνει η ακόλουθη γραμμή κώδικα:

```
Operator :<input type="text" name="op"  maxlength="1">
```

Εμφανίζει την οδηγία/λεκτικό (caption) " Operator :" και δίπλα το κουτί κειμένου (input type="text") που έχει μήκος 1. Το σύμβολο που πληκτρολογεί ο χρήστης στο κουτί αποθηκεύεται στη μεταβλητή op.

Να τι κάνει η ακόλουθη γραμμή κώδικα:

```
Second number :<input type="text" name="nu2">
```

Εμφανίζει την οδηγία/λεκτικό (caption) " Second number :" και δίπλα το κουτί κειμένου (input type="text"). Ο αριθμός που πληκτρολογεί ο χρήστης στο κουτί αποθηκεύεται στη μεταβλητή nu2.

Να τι κάνει η ακόλουθη γραμμή κώδικα:

```
<input type="submit" name="name" value="Calculate!">
```

Εμφανίζει το κουμπί που γράφει "Calculate!". Όταν πατηθεί το κουμπί που γράφει "Calculate!" η σελίδα μας καλεί τη σελίδα calc.jsp.

Η μέθοδος της φόρμας είναι post, και η φόρμα όταν πατηθεί το κουμπί που γράφει "Calculate!" καλεί τη σελίδα calc.jsp. Η μέθοδος post «περνάει» στη φόρμα calc.jsp τις τιμές που πληκτολογήσαμε στα κουτιά κειμένου. Οι μεταβλητές nu1, nu2, op, που στέλνει η σελίδα μας, έχουν τις τιμές που πληκτρολογήσαμε, και οι τιμές αυτές είναι είναι συμβολοσειρές (string).

Ακολουθεί ο κώδικας της φόρμας Formname:

```
<form name="Formname" method="post" action="calc.jsp" >
  <p>First number :<input type="text" name="nu1" >
  <p>Operator :<input type="text" name="op"  maxlength="1">
  <p>Second number :<input type="text" name="nu2">
  <br><input type="submit" name="name" value="Calculate!">
</form>
```

Στη συνέχεια, παρατίθεται ο κώδικας όλης της σελίδας index.html / index.jsp.

index.html

```
<!DOCTYPE html>
<!--
Calculator
-->
<html>
<head>
<title>Simple calculator</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
  <form name="Formname" method="post" action="calc.jsp" >
    <p>First number :<input type="text" name="nu1" >
    <p>Operator :<input type="text" name="op"  maxlength="1">
    <p>Second number :<input type="text" name="nu2">
    <br><input type="submit" name="name" value="Calculate!">
  </form>
</body>
</html>
```

Η jsp σελίδα calc.jsp καλείται από την index.html και είναι μία html σελίδα που περιλαμβάνει scriptlets (κώδικα java) ανάμεσα σε tag:

```
<%    %>
```

Στο επόμενο scriptlet ορίζουμε μεταβλητές:

```
<%
   int result;
   int n1=Integer.parseInt(request.getParameter("nu1"));
   String op=request.getParameter("op");
   int n2=Integer.parseInt(request.getParameter("nu2"));
%>
```

Χρησιμοποιούμε τις ακέραιες μεταβλητές result για το αποτέλεσμα της πράξης, n1 για τον πρώτο αριθμό, n2 για τον δεύτερο αριθμό. Χρησιμοποιούμε τη μεταβλητή συμβολοσειρά (String) op για την πράξη.

Για να αποθηκεύσουμε τις τιμές που έστειλε η σελίδα index.html/index.jsp χρησιμοποιούμε τη μέθοδο getParameter της κλάσης request. Επειδή οι τιμές είναι συμβολοσειρές (String) αν θέλουμε να τις μετατρέψουμε σε ακέραιους αριθμούς τότε χρησιμοποιούμε τη μέθοδο parseInt της κλάσης Integer.

Αν θέλουμε να τις μετατρέψουμε σε πραγματικούς αριθμούς τότε χρησιμοποιούμε τη μέθοδο parseFloat της κλάσης Float.

Στο επόμενο scriptlet κάνουμε την πράξη και εμφανίζουμε το αποτέλεσμα.

```
<%
if(op.equals("+")) {
  result=n1+n2;
  out.println(result);
}
%>
```

Προσοχή! Δε χρησιμοποιούμε τη System.out.println γιατί τότε το αποτέλεσμα είναι στον server. Παρατηρήστε, επιπλέον, ότι δε γινεται κανένας έλεγχος για την πράξη που πληκτρολογεί ο χρήστης κ.λπ. Στη συνέχεια, παρατίθεται ο κώδικας όλης της σελίδας calc.jsp.

calc.jsp (-- simple addition)

```
<%--
Document   : calc
Created on : 20 Νοε 2021, 8:23:54 πμ
Author : Christos
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.util.*" %>
<%@ page import="java.*" %>
<%
   int result;
   int n1=Integer.parseInt(request.getParameter("nu1"));
   String op=request.getParameter("op");
   int n2=Integer.parseInt(request.getParameter("nu2"));
%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Simplified calculator - additin</title>
</head>
<body>
  <h1>Results</h1>
```

```
  <%
    if(op.equals("+")) {

      result=n1+n2;
      out.println(result);

    }
  %>
</body>

</html>
```

Αλλάζουμε τη σελίδα calc.jsp έτσι ώστε όταν πληκτρολογούμε λάθος πράξη να απαντά με μήνυμα "wrong operation".

```
<%
String opname="none";
if(op.equals("+")) {

  result=n1+n2; opname="addition";
  out.println(result);

}
else {

  out.println("wrong operation");

}
%>
```

Το scriptlet αυτό εύκολα ξαναγράφεται ώστε η σελίδα μας να υποστηρίζει τις τέσσερις πράξεις.

Στη συνέχεια, παρατίθεται ο κώδικας όλης της νέας σελίδας calc.jsp.

calc.jsp (-- addition and wrong operator)

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.util.*" %>
<%@ page import="java.*" %>
<%
   int result;
   int n1=Integer.parseInt(request.getParameter("nu1"));
   String op=(String)request.getParameter("op");
   int n2=Integer.parseInt(request.getParameter("nu2"));
%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Simple calculator</title>
</head>
<body>
<h1>Results</h1>
<%
  String opname="none";
  if(op.equals("+")){

    result=n1+n2; opname="addition";
    out.println(result);
```

```
  }
  else {

    out.println("wrong operation");

  }
%>
</body>
</html>
```

Στη συνέχεια, παρατίθεται ο τελικός κώδικας της αρχικής σελίδας calc.jsp με μικρές τροποποιήσεις που επιτρέπουν την πρόσθεση πραγματικών αριθμών.

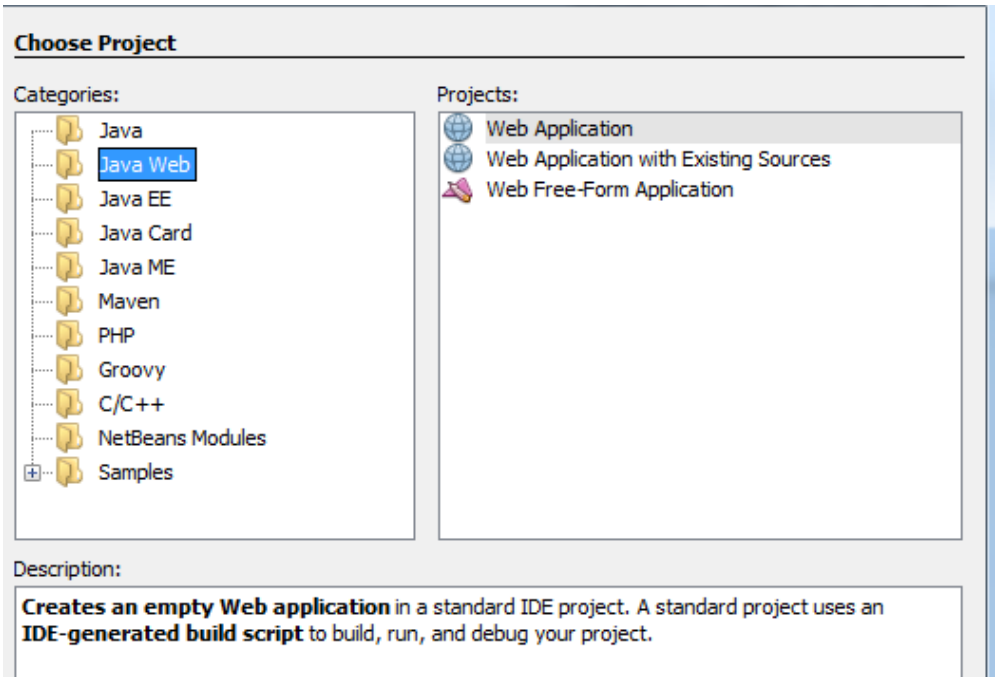
calc.jsp (-- addition – Float – parseFloat)

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.util.*" %>
<%@ page import="java.*" %>
<%
   float result;
   float n1=Float.parseFloat(request.getParameter("nu1"));
   String op=(String)request.getParameter("op");
   float n2=Float.parseFloat(request.getParameter("nu2"));
%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Simple calculator</title>
</head>
<body>
<h1>Results</h1>
<%
  String opname="none";
  if(op.equals("+")) {

    result=n1+n2; opname="addition";
    out.println(result);

  }
  else {

    out.println("wrong operation");

  }
%>
</body>
</html>
```

12.1.1 Πως θα κατασκευάσουμε τις σελίδες στο προϊόν Netbeans.

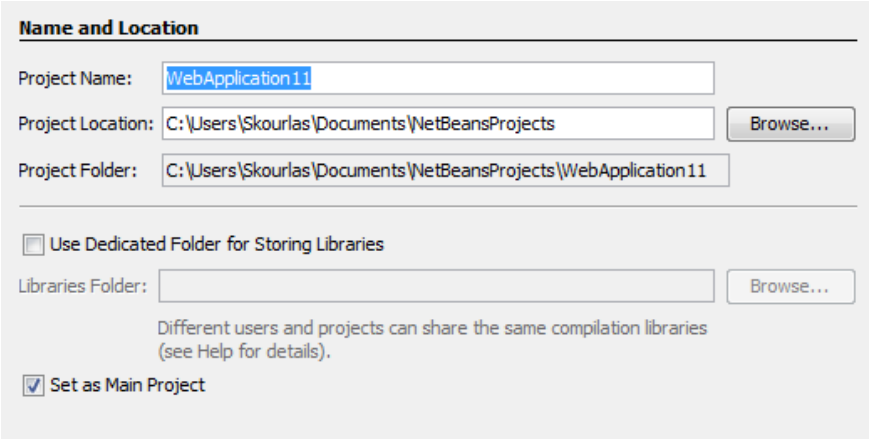
Παραθέτουμε τις ενέργειες που πρέπει να κάνουμε στη διεπαφή του προϊόντος Netbeans για να κατασκευάσουμε την πρώτη εφαρμογή μας, δηλαδή να δημιουργήσουμε τις σελίδες index, calc.

Στην εικόνα 12.3 επιλέγουμε ως κατηγορία εφαρμογών (categories) Java Web και ως τύπο project Web Application.



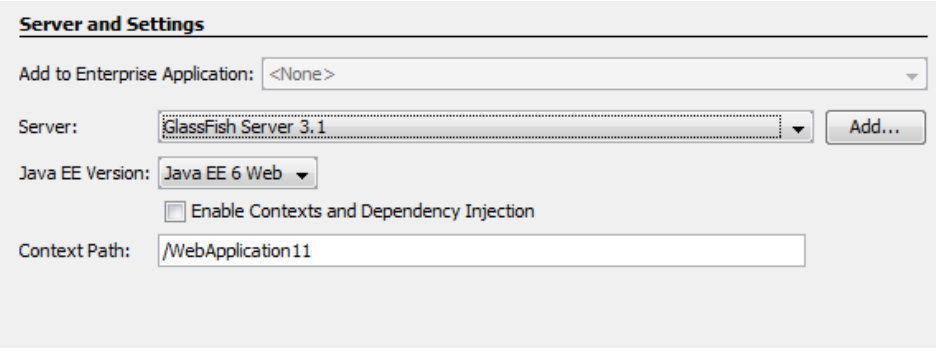
Εικόνα 12.3 Επιλογή κατηγορίας εφαρμογών (categories) Java Web και τύπου έργου (project) Web Application.

Στην εικόνα 12.4 επιλέγουμε όνομα (Web Application11), θέση και περιοχή του project Web Application11.



Εικόνα 12.4 Επιλογή ονόματος (Web Application11), θέσης και περιοχής του project.

Στην εικόνα 12.5 επιλέγουμε τον διακομιστή μας για τις ιστοσελίδες του project Web Application11. Μπορούμε να επιλέξουμε GlassFish server ή Apache server.



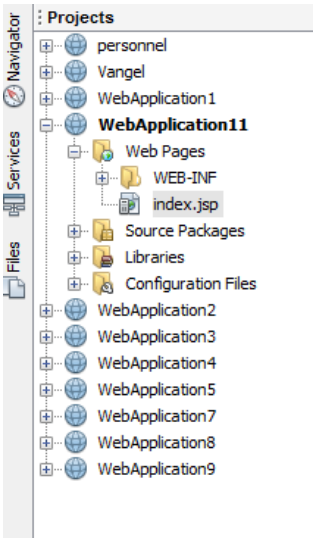
Εικόνα 12.5 Επιλογή διακομιστή για τις ιστοσελίδες του project. GlassFish server ή Apache server.

Στην εικόνα 12.6 βλέπουμε πως κατασκευάζουμε τη σελίδα index.html. Ουσιαστικά κάνουμε copy & paste στο Nertbeans IDE τον κώδικα της σελίδας index.html που είδαμε παραπάνω.

```
<!DOCTYPE html>
<!--
Calculator
-->
<html>
    <head>
        <title>Simple calculator</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
          <form name="Formname" method="post" action="calc.jsp" >
              <p>   First number :<input type="text" name="nu1" >
              <p>   Operator :<input type="text" name="op"  maxlength="1">
              <p>   Second number :<input type="text" name="nu2">
              <br><input type="submit" name="name" value="Calculate!">
          </form>
    </body>
</html>
```

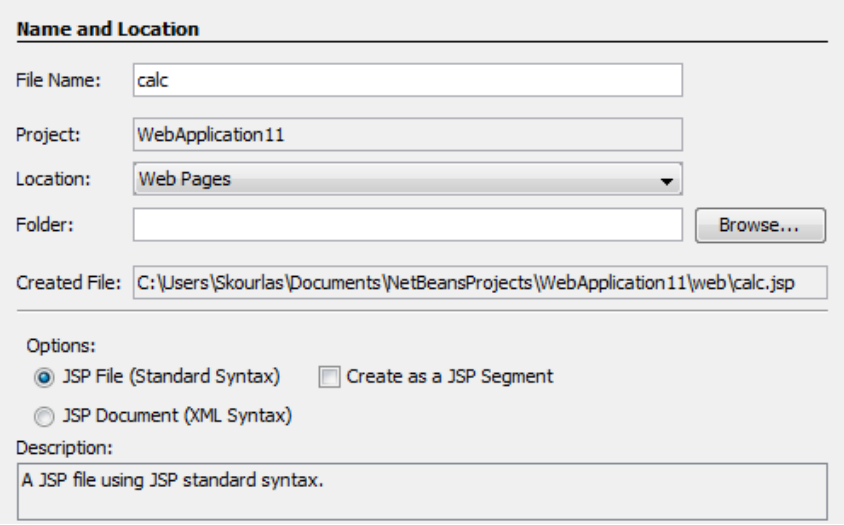
Εικόνα 12.6 Κατασκευή σελίδας index.html στο Nertbeans IDE

Στην εικόνα 12.7 βλέπουμε τα project που κατασκευάσαμε και ειδικότερα τις ιστοσελίδες του τρέχοντος project Web Application11 που έχουν υλοποιηθεί μέχρι στιγμής.



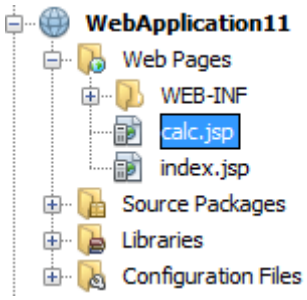
Εικόνα 12.7 Τα project που κατασκευάσαμε και ειδικότερα οι ιστοσελίδες του τρέχοντος project Web Application11 που έχουν υλοποιηθεί

Στην εικόνα 12.8 βλέπουμε πως κατασκευάζουμε τη σελίδα calc.jsp. Ουσιαστικά κάνουμε copy & paste στο Nertbeans IDE τον κώδικα της σελίδας calc.jsp που είδαμε παραπάνω.



Εικόνα 12.8 Κατασκευή σελίδας calc.jsp στο Nertbeans IDE

Στην εικόνα 12.9 βλέπουμε τις ιστοσελίδες του project Web Application11



Εικόνα 12.9 Ιστοσελίδες του project Web Application11.

12.1.2 Έλεγχος ορθότητας των δεδομένων που πληκτρολογεί ο χρήστης . Χρήση try/catch Block

Ακολουθεί παράδειγμα χρήσης try/catch Block στην περίπτωση απόπειρας διαίρεσης ακεραίου αριθμού με μηδέν. Βλέπε και http://www.java2s.com/Code/Java/JSP/UsingatrycatchBlock.htm

```
<HTML>
<HEAD>
<TITLE>Using a try/catch Block</TITLE>
</HEAD>

<BODY>
<H1>Using a try/catch Block</H1>
<%
try{
  int i = 1;
  i = i / 0;
  out.println("The answer is " + i);
}
  catch (Exception e){
  out.println("An exception occurred: " + e.getMessage());
}
%>
```

```
</BODY>
</HTML>
```

Παραθέτουμε παράδειγμα χρήσης try/catch Block στην περίπτωση υπολογισμού τόκου.

```
<%--
Document   : try_catch
Created on : 2 Δεκ 2021, 7:45:44 πμ
Author : Christos
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<HTML>
<HEAD>
<TITLE>Using a try/catch Block</TITLE>
</HEAD>
<BODY>
<H1>Using a try/catch Block</H1>
<%
try{
  int interest = 0;
  int balance = 1000;
  int interest_rate = 0;
  interest = (balance * interest_rate)/100;
  interest = interest / interest_rate;
  out.println("The answer is " + interest);
}
catch (Exception ex){
  out.println("An exception occurred: " + ex.getMessage());
}
%>
</BODY>
</HTML>
```

Το μπλοκ try προστατεύει τον κώδικα που περιλαμβάνει τη μέθοδο που μπορεί να προκαλέσει εξαίρεση (exception). Το μπλοκ catch διαχειρίζεται την εξαίρεση. Να πως θα «μεταφράζαμε» τον κώδικα:

Δοκίμασε (try) τον κώδικα

```
int interest = 0;
int balance = 1000;
int interest_rate = 0;
interest = (balance * interest_rate)/100;
interest = interest / interest_rate;
out.println("The answer is " + interest);
```

Αν εκτελείται σωστά συνέχισε το πρόγραμμά σου.

Αλλιώς, «παγίδευσε»/«άδραξε» (catch) την εξαίρεση και διαχειρίσου την.

```
catch (Exception ex)
{
out.println("An exception occurred: " + ex.getMessage());
}
```

Exception είναι το όνομα της κλάσης που θα «παγιδευτεί» και ex το όνομα της μεταβλητής.

Στη συνέχεια δίδεται μια πληρέστερη λύση για τη σελίδα calc.jsp.

12.1.3 Αριθμομηχανή (Calculator) τεσσάρων πράξεων

Παραθέτουμε εκ νέου τις δύο ιστοσελίδες της αριθμομηχανής.

index.html (-- αρχική σελίδα)

```
<!DOCTYPE html>
<!--
Calculator
-->
<html>
<head>
<title>Simple calculator</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
  <form name="Formname" method="post" action="calc.jsp" >
    <p>First number :<input type="text" name="nu1" >
    <p>Operator :<input type="text" name="op"  maxlength="1">
    <p>Second number :<input type="text" name="nu2">
    <br><input type="submit" name="name" value="Calculate!">
  </form>
</body>
</html>
```

calc.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.util.*" %>
<%@ page import="java.*" %>
<%

try{

   String n1=(String)request.getParameter("nu1");
   String op=(String)request.getParameter("op");
   String n2=(String) request.getParameter("nu2");
%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Simple calculator</title>
</head>
<body>
<h1>Results</h1>
<%
   float nu1=Float.parseFloat(n1);
   float nu2=Float.parseFloat(n2);
   float result=0;
   String opname="none";
```

```
   if(op.equals("+")){ result=nu1+nu2; opname=" addition";}
   else if (op.equals("*")) {result=nu1*nu2; opname=" multiplication";}
   else if (op.equals("-"))  {result=nu1-nu2; opname="      subtraction";}
   else if (op.endsWith("/")) {result=nu1/nu2; opname="     division";}
   else { opname="wrong";}


   if(opname.equals("wrong")){

     out.println("Wrong operation");
   }
   else {

     out.print("Your operation is "+opname+"<br>");
     out.println("Result="+result);
   }
}
catch(Exception ex)
{
   out.println("Something is wrong or missing");
}
%>
</body>
</html>
```

Στην εικόνα 12.10 βλέπουμε πως κατασκευάζουμε τη σελίδα calc.jsp. Ουσιαστικά κάνουμε copy & paste στο Nertbeans IDE τον κώδικα της σελίδας calc.jsp που είδαμε παραπάνω.

Στις εικόνες 12.1 και 12.2 βλέπουμε τη δοκιμή των προγραμμάτων μας για την πρόσθεση δύο ακεραίων αριθμών.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.util.*" %>
<%@ page import="java.*" %>
    <% try{String n1=(String)request.getParameter("nu1");
   String op=(String)request.getParameter("op");
   String n2=(String) request.getParameter("nu2");
%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Simple calculator</title>
    </head>
    <body>
        <h1>Results</h1>
```

```
<%
        float nu1=Float.parseFloat(n1);
        float nu2=Float.parseFloat(n2);
        float result=0;
        String opname="none";
        if(op.equals("+")){ result=nu1+nu2; opname="    addition";}
           else if (op.equals("*")) {result=nu1*nu2; opname=" multiplication";}
           else if (op.equals("-"))  {result=nu1-nu2; opname="  subtraction";}
           else if (op.endsWith("/")) {result=nu1/nu2; opname=" division";}
           else { opname="wrong";}
           if(opname.equals("wrong"))
           {    out.println("Wrong operation");
           }
           else
           {    out.print("Yor operation is "+opname+"<br>");
                out.println("Result="+result);
           }
    }
               catch(Exception ex)
{
    out.println("Something is wrong or missing");
}
        %>
 </body>
</html>
```

Εικόνα 12.10 Κατασκευή νέας σελίδας calc.jsp στο Nertbeans IDE η οποία περιλαμβάνει ελέγχους

12.2 Εισαγωγή με παράδειγμα στη σύνδεση δυναμικών σελίδων Java Server Pages (JSP) και βάσης δεδομένων. Κατασκευή βήμα-βήμα μια απλουστευμένης εφαρμογής σύνδεσης (login)

Θα κατασκευάσουμε βήμα-βήμα μια απλουστευμένη εφαρμογή σύνδεσης (login) σε ιστοσελίδα. Οι κωδικοί των χρηστών (userid, password) αποθηκευόνται σε βάση δεδομένων. Ο χρήστης δίνει τα στοιχεία του στην αρχική σελίδα, γίνεται έλεγχος αν τα στοιχεία αυτά είναι αποθηκευμένα στη βάση δεδομένων. Αν υπάρχουν τα στοιχεία του πηγαίνει στην επόμενη σελίδα αλλιώς επιστρέφει στην αρχική.

12.2.1 Σύνδεση δυναμικών σελίδων JSP και βάσης δεδομένων. To-do list

Αρχικά περιγράφουμε τι απαιτείται για τη σύνδεση δυναμικής σελίδας JSP με βάση δεδομένων MySQL. Το περιβάλλον ανάπτυξης που θα χρησιμοποιήσουμε είναι Netbeans. Ακολουθεί η To-do list:

Δήλωση του JDBC driver στο περιβάλλον Netbeans

Εισάγετε το package java.sql στο πρόγραμμά σας:

```
<%@page import="java.sql.*" %>
```

Σύνδεση στη βάση

Για παράδειγμα, οι παράμετροι της βάσης δεδομένων είναι:

```
IP= localhost, Port= 3306, Database= users_login, Username= root, Password= 1234
String DB = "jdbc:mysql://localhost:3306/users_login?user=root&password=1234";
Connection myConnection = DriverManager.getConnection(DB);
```

Δημιουργία αντικειμένου για να γράψουμε SQL statement:

```
Statement SMT = myConnection.createStatement();
```

Αναγραφή SQL statement στο αντικείμενο SMT και εκτέλεση:

5.1) Για δηλώσεις SELECT

```
String sql="SELECT * FROM user WHERE Uname='"+a_user+"'AND Upass='"+a_pass+"' ";
ResultSet rs=SMT.executeQuery(sql); -- select
found= rs.first();
if (found){ …}  else { … }
```

5.2) Για δηλώσεις insert, update

```
String sqlString = "INSERT INTO mytable VALUES(4, 'Jim')";
myStatement.executeUpdate(sqlString);
```

Κλείσιμο αντικειμένων «αναγραφής» SQL statement:

```
SMT.close();
```

Κλείσιμο σύνδεσης:

```
myConnection.close();
```

12.2.2 Τρόποι μετάβασης από σελίδα σε σελίδα

Για τη μετάβαση από σελίδα σε σελίδα μπορούμε να χρησιμοποιήσουμε διάφορες τεχνικές. Στη συνέχεια παραθέτουμε τρεις από αυτές:

Πρώτη τεχνική

```
<a href="index.jsp">Try Again</a><%
```

Δεύτερη τεχνική

```
<form name="formName" method="post" action="check.jsp" >
  Name: <input type="text" name="uname"><p></p>
  Password:<input type="text" name="upass"><P></P>
  <input type="submit" value="LOGIN"><p></p>
</form>
```

Τρίτη τεχνική

```
<INPUT TYPE="BUTTON" VALUE="GO BACK" style="font-size:10px"
ONCLICK="window.location.href='index.jsp'">
```

12.2.3 Κατασκευή της εφαρμογής

Στη συνέχεια παραθέτουμε τα βήματα που πρέπει να ακολουθήσετε για να κατασκευάσετε την εφαρμογή.

Δημιουργήστε τη βάση δεδομένων users_login που περιλαμβάνει τον πίνακα χρηστών user. Οι στήλες του πίνακα είναι: Uname-όνομα χρήστη, upass-συνθηματικό, Uid-αριθμός χρήστη, Uphone-τηλέφωνο, Ucity-πόλη

```
CREATE DATABASE users_login;
USE users_login;
CREATE TABLE user(uname text, upass text, Uid int(11),
 Uphone varchar(45), Ucity varchar(45));
```

Εισάγετε γραμμές (χρήστες) στον πίνακα.

```
INSERT INTO user VALUES ('admin','1111',1,NULL,NULL);
Είναι ο χρήστης admin με συνθηματικό 1111
INSERT INTO user VALUES ('scott','tiger',2,NULL,NULL);
Είναι ο χρήστης scott με συνθηματικό tiger
Select * from user;
```

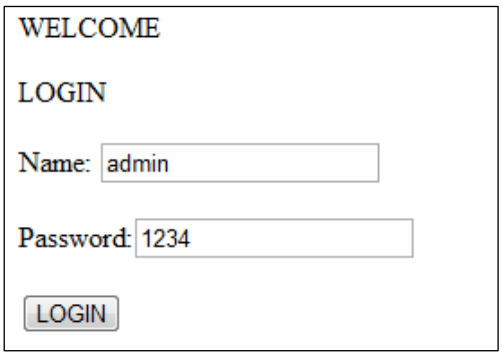
Οι παράμετροι της βάσης είναι:

```
IP= localhost
Port= 3306
Database= users_login
Username= root
Password= 1234
```

Τι θα κάνετε:

Στην αρχική σελίδα Index.jsp ο χρήστης δίνει όνομα και συνθηματικό και πατά το κουμπί που γράφει LOGIN.

Στην εικόνα 12.11 βλέπουμε παράδειγμα σύνδεσης χρήστη στη φόρμα που εμφανίζει η αρχική σελίδα Index.jsp.



Εικόνα 12.11Παράδειγμα σύνδεσης χρήστη στη φόρμα που εμφανίζει η αρχική σελίδα Index.jsp.

Τότε η σελίδα καλεί τη σελίδα check.jsp και τις «περνάει» τις τιμές που έδωσε ο χρήστης. Στο παράδειγμα, οι τιμές είναι τα string "admin" και "1234".

Η ιστοσελίδα check.jsp διαβάζει τις τιμές (που έστειλε η ιστοσελίδα Index.jsp)

Στη συνέχεια συνδέεται με τη βάση και εκτελεί τη δήλωση:

```
SELECT * FROM user WHERE Uname='admin'AND Upass='1234';
```

Στην περίπτωσή μας η σελίδα δε βρίσκει τον χρήστη admin με συνθηματικό 1234 στον πίνακα user οπότε βγάζει το μήνυμα **INCORRECT TRY AGAIN** και καλεί την αρχική σελίδα index.jsp

Αν ο χρήστης δώσει στοιχεία που υπάρχουν στον πίνακα user (στην περίπτωσή μας υπάρχει καταχώρηση για τον χρήστη admin με συνθηματικό 1111, και τον χρήστη scott με συνθηματικό tiger) τότε η check.jsp καλεί την σελίδα home.jsp

Σημείωση

Η δήλωση SELECT με παραμέτρους θα είχε τη μορφή:

```
SELECT * FROM user WHERE Uname='a_user' AND Upass='a_pass';
```

Τότε στη σελίδα check :jsp ορίζουμε το string:

```
String sql=
"SELECT * FROM user WHERE Uname='"+a_user+"'AND Upass='"+a_pass+"' ";
```

Ακολουθεί η σελίδα index.jsp

index.jsp

```
<%--
Document   : index
Created on : Apr 4, 2021, 6:04:18 PM
Author : Christos
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
WELCOME<p></p>
LOGIN<p></p>
<p></p>
<form name="formName" method="post" action="check.jsp" >
Name: <input type="text" name="uname"><p></p>
Password:<input type="text" name="upass"><P></P>
 <input type="submit" value="LOGIN"><p></p>
</form>
</body>
</html>
```

Ακολουθεί η σελίδα **check.jsp**

check.jsp

```
<%--
Document   : check
Created on : Apr 4, 2021, 6:15:58 PM
Author : Christos
--%>
<%@page import="java.sql.*" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<%
Boolean found;
String a_user=request.getParameter("uname");
```

```
String a_pass=request.getParameter("upass");
String  URL;
Class.forName("com.mysql.jdbc.Driver");
String DB = "jdbc:mysql://localhost:3306/users_login?user=root&password=1234";
Connection myConnection = DriverManager.getConnection(DB);
Statement SMT = myConnection.createStatement();
String sql="SELECT * FROM user WHERE Uname='"+a_user+"'AND Upass='"+a_pass+"'
";
ResultSet rs=SMT.executeQuery(sql);
found= rs.first();
if (found){
  URL = "home.jsp";
  response.sendRedirect(URL);
}
else {
  out.println("INCORRECT TRY AGAIN");%><P></P>
    <a href="index.jsp">Try Again</a><%
}
SMT.close();
myConnection.close();
%>
</body>
</html>
```

Ακολουθεί η σελίδα home.jsp

home.jsp

```
<%--
Document   : home
Created on : Apr 4, 2021, 6:32:14 PM
Author : Christos
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Home Page</title>
</head>
<body>
Welcome! <br>
You are authorized user!
   <INPUT TYPE="BUTTON" VALUE="GO BACK" style="font-size:10px"
   ONCLICK="window.location.href='index.jsp'">
</body>
</html>
```

12.2.4 Κατασκευή login – registration form με χρήση JSP-MySql

Αρχικά θα αναφερθούμε στη διεπαφή του χρήστη (user interface) και στη λειτουργικότητα της εφαρμογής. Στη συνέχεια θα μελετήσουμε τον κώδικα.

Η εφαρμογή χρησιμοποιεί βάση δεδομένων και η λειτουργικότητα περιγράφεται στα παρακάτω βήματα:

1) Βάση δεδομένων

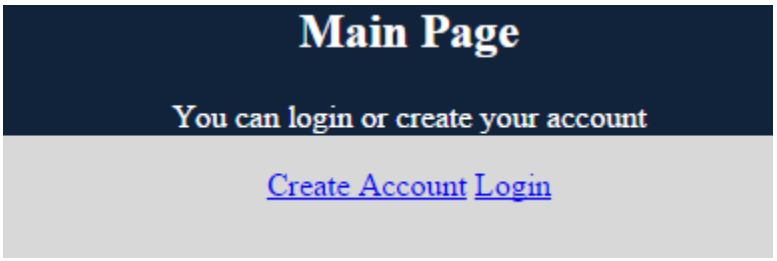
Η εφαρμογή συνδέεται με τη βάση webusers με connection που έχει host:localhost:3306 username: root και δεν έχουμε password. Η βάση webusers αποτελείται από ένα πίνακα users με την εξής περιγραφή

Field	Type	Null	Key	Default	Extra
U_id	int(11)	NO	PRI	NULL	auto_increment
username	varchar(32)	YES	UNI	NULL	
password	varchar(32)	YES		NULL	
email	varchar(32)	YES	UNI	NULL	

Όπως παρατηρείτε username και email είναι μοναδικά.

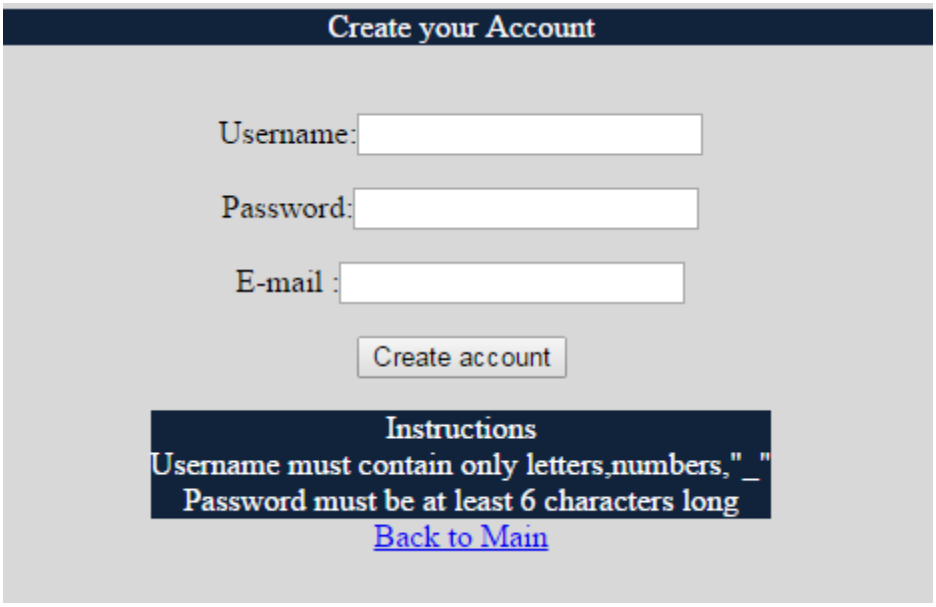
2) Λειτουργία

Η εκκίνηση γίνεται από τη σελίδα index.html που παρουσιάζει τις επιλογές είτε για login είτε για create account



2.1) Create account

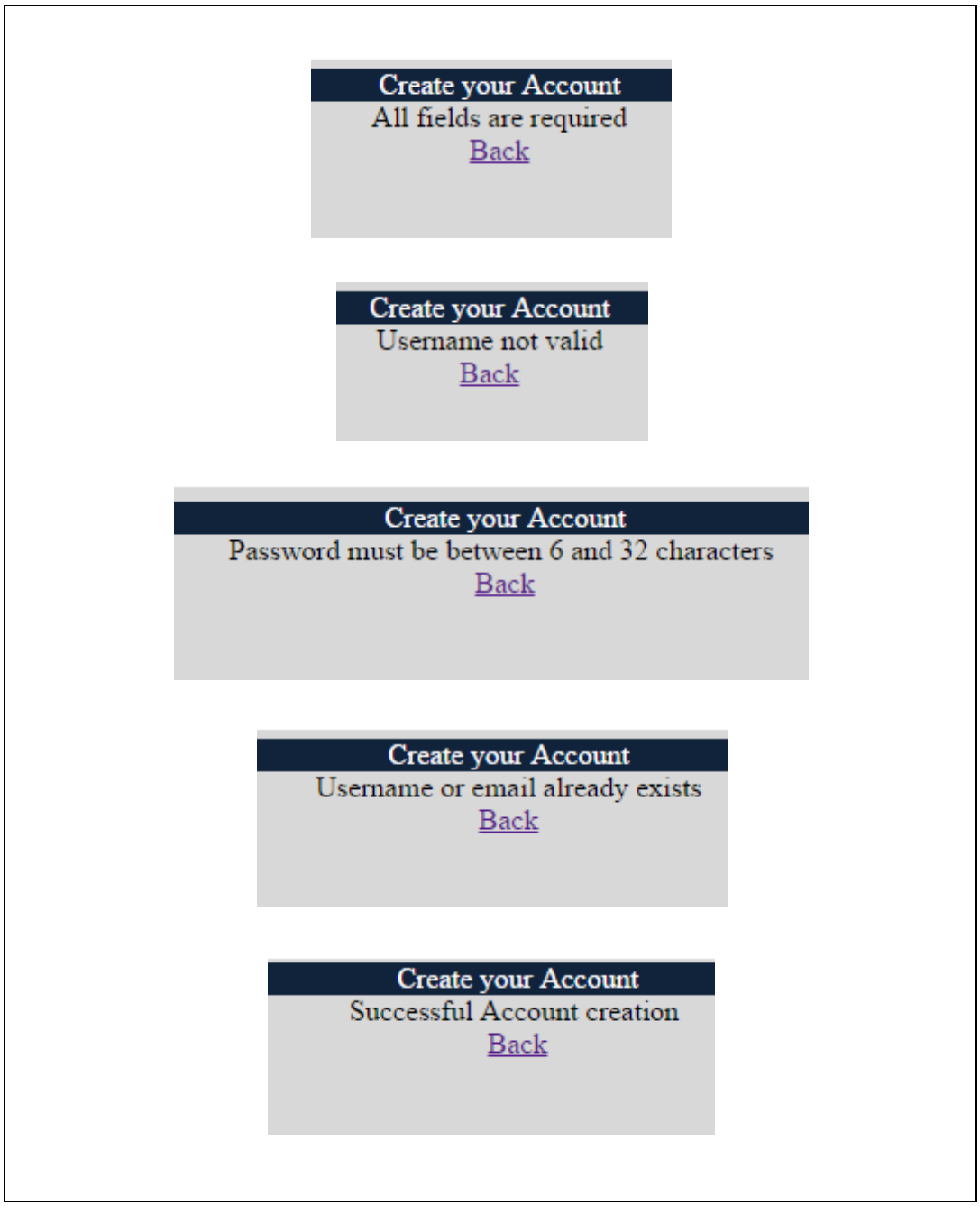
Με την επιλογή create account καλείται η createform.html που είναι η ακόλουθη



Όπως φαίνεται και στις οδηγίες το όνομα χρήστη (username) μπορεί να περιέχει μόνο γράμματα, αριθμούς και την κάτω παύλα και το συνθηματικό πρέπει να είναι τουλάχιστον 6 χαρακτήρες. Σε όλα τα πεδία έχει τεθεί ως μέγιστο οι 32 χαρακτήρες.

Τι θα συμβεί κατά την υποβολή της φόρμας

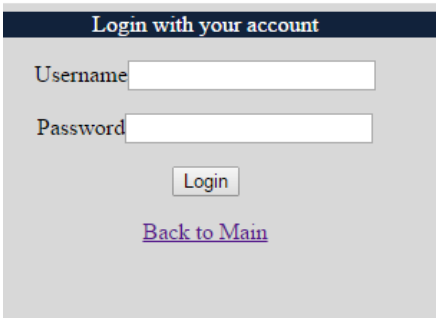
Κατά την υποβολή της φόρμας καλείται η createacc.jsp για τη σύνδεση με τη βάση, τον έλεγχο των πεδίων και την εισαγωγή τους. Η φόρμα ελέγχει και αν όλα τα πεδία της φόρμας έχουν συμπληρωθεί. Μέσω regular expressions της java ελέγχεται και ο περιορισμός του username αλλά και αν το email έχει την μορφή κάποιου πραγματικού. Επίσης, πριν γίνει η εισαγωγή, η φόρμα ελέγχει εάν το username ή το email υπάρχουν. Σε όλες τις παραπάνω περιπτώσεις αλλά και για την επιτυχή εισαγωγή εμφανίζει σχετικό μήνυμα (εικόνα 12.12).



Εικόνα 12.12 Μηνύματα εφαρμογής κατά τη λειτουργία create account σύνδεσης χρήστη

2.2) Login

Με την επιλογή login καλείται η loginform.html που ζητά από τον χρήστη να δώσει username και password για να κάνει login.



Τι θα συμβεί κατά την υποβολή της φόρμας

Όταν ο χρήστης πατήσει login η login.jsp αναλαμβάνει τον έλεγχο των πεδίων του. Αρχικά, ελέγχει αν ο χρήστης έχει συμπληρώσει όλα τα πεδία και αν δεν το έχει κάνει εμφανίζει κατάλληλο μήνυμα. Αν έχει συμπληρώσει όλα τα πεδία ελέγχει αν υπάρχει στην βάση ο συνδυασμός username και password. Αν υπάρχει του εμφανίζει μήνυμα αλλιώς ελέγχει αν υπάρχει το όνομα του χρήστη με νέα αναζήτηση ώστε να διαπιστώσει εάν δεν υπάρχει ο χρήστης ή αν ο κωδικός είναι λάθος. Στις περιπτώσεις ειδικότερα, που είτε δεν υπάρχει ο χρήστης είτε ο κωδικός είναι λάθος, υπάρχει η επιλογή να μεταβεί στην δημιουργία λογαριασμού

Εικόνα 12.13 Μηνύματα εφαρμογής κατά τη σύνδεση χρήστη (login)

3) Διαμόρφωση σελίδων

Για τη διαμόρφωση των σελίδων που περιλαμβάνει χρώμα, backround σελίδων και tags αλλά και θέση φορμών και links μπορεί να χρησιμοποιηθεί CSS αρχείο. Βλέπε style.css στην ενότητα 12.2.5 στην οποία παραθέτουμε τα προγράμματα.

12.2.5 Κατασκευή login – registration forms

Δημιουργία βάσης δεδομένων

```
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='TRADITIONAL,ALLOW_INVALID_DATES';
CREATE SCHEMA IF NOT EXISTS `webusers` DEFAULT CHARACTER SET latin1 ;
USE `webusers` ;
-- -----------------------------------------------------
-- Table `webusers`.`users`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `webusers`.`users` (
  `U_id` INT(11) NOT NULL AUTO_INCREMENT ,
  `username` VARCHAR(32) NULL DEFAULT NULL ,
  `password` VARCHAR(32) NULL DEFAULT NULL ,
  `email` VARCHAR(32) NULL DEFAULT NULL ,
  PRIMARY KEY (`U_id`) ,
  UNIQUE INDEX `username` (`username` ASC) ,
  UNIQUE INDEX `email` (`email` ASC) )
ENGINE = InnoDB
AUTO_INCREMENT = 21
DEFAULT CHARACTER SET = latin1;
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Ακολουθεί η περιγραφή του πίνακα webusers.

```
mysql> describe users;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| U_id     | int(11)     | NO   | PRI | NULL    | auto_increment |
| username | varchar(32) | YES  | UNI | NULL    |                |
| password | varchar(32) | YES  |     | NULL    |                |
| email    | varchar(32) | YES  | UNI | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.03 sec)
```

Διαμόρφωση σελίδων. style.css

style.css

```
@charset "UTF-8";
body{
  background-color:#D8D8D8;
  width:960px;
  margin-left: auto;
  margin-right: auto;
  text-align: center;
```

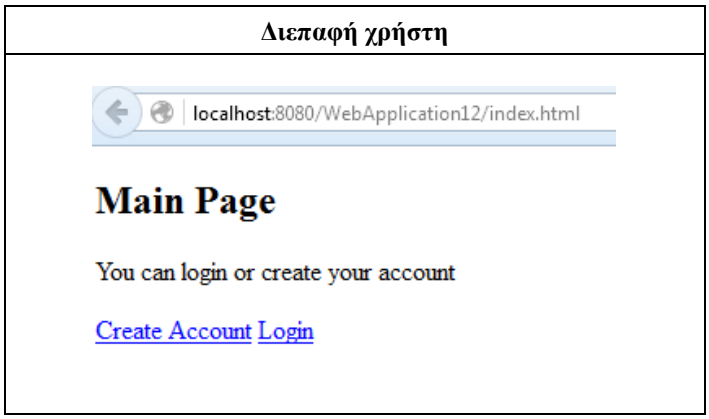
```
}
header{
  display: block;
  text-align: center;
  background-color: #11223B;
  color: white;
}
aside{
  color:white;
  text-align:center;
}
aside div{
  display:inline-block;
  background-color: #11223B;
  padding-left: auto;
  padding-right: auto;
}
form{
  text-align: center;
}
.ln {
  text-align: center;
}
.ln a{
  display:inline-block;
}
```

Παραθέτουμε την αρχική σελίδα index.html.

index.html

```
<html>
<head>
<title>Main Page</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <header>

    <h2>Main Page</h2>
    <p> You can login or create your account</p>

  </header>
  <div class="ln">
    <a href="createform.html">Create Account</a>
    <a href="loginform.html">Login</a>
  </div>
</body>
</html>
```

Στην εικόνα 12.14 βλέπουμε την αρχική εικόνα της διεπαφής της εφαρμογής σύνδεσης χρήστη (login. Παράγεται από τη σελίδα index.html



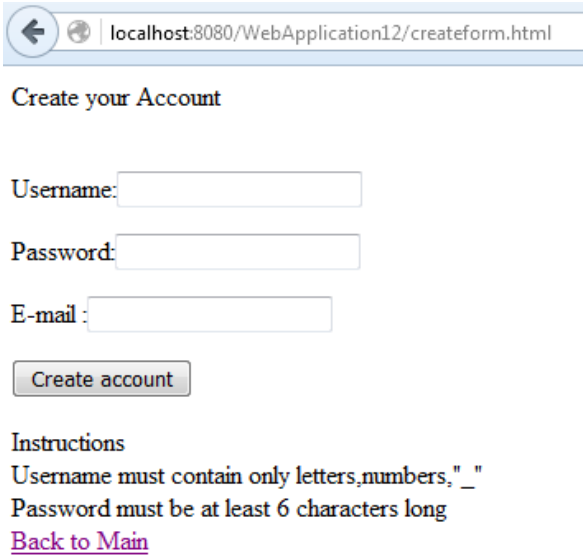
Εικόνα 12.14 Αρχική εικόνα διεπαφής της σύνδεσης χρήστη (login. Παράγεται από τη σελίδα index.html

Παραθέτουμε την κατασκευή της σελίδας createform.html η οποία εμφανίζει τη φόρμα στην οποία εισάγει στοιχεία ο χρήστης.

createform.html

```
<html>
<head>
<title>Create Account</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<header>Create your Account</header><br>
<form name="Createform" method="post" action="createacc.jsp">
  <p> Username:<input type="text" maxlength="32" name="usname">
  <p> Password:<input type="password" maxlength="32" name="passw">
  <p> E-mail  :<input type="text" maxlength="32" name="mail">
  <p> <input type="submit"  name="subcreate" value="Create account">
</form>
<aside>

  <div>Instructions<br> Username must contain only letters,numbers,"_"<br>
    Password must be at least 6 characters long<br>

  </ div >

</aside>
<div class="ln">
  <a href="index.html">Back to Main</a>
</div>
</body>
</html>
```

Στην εικόνα 12.15 βλέπουμε τη φόρμα στην οποία εισάγει στοιχεία ο χρήστης. Η φόρμα αυτή είναι στοιχείο της διεπαφής της εφαρμογής σύνδεσης χρήστη (login). Παράγεται από τη σελίδα createform.html.



Εικόνα 12.15 Ηλεκτρονική φόρμα στην οποία εισάγει στοιχεία ο χρήστης. Παράγεται από τη σελίδα createform.html.

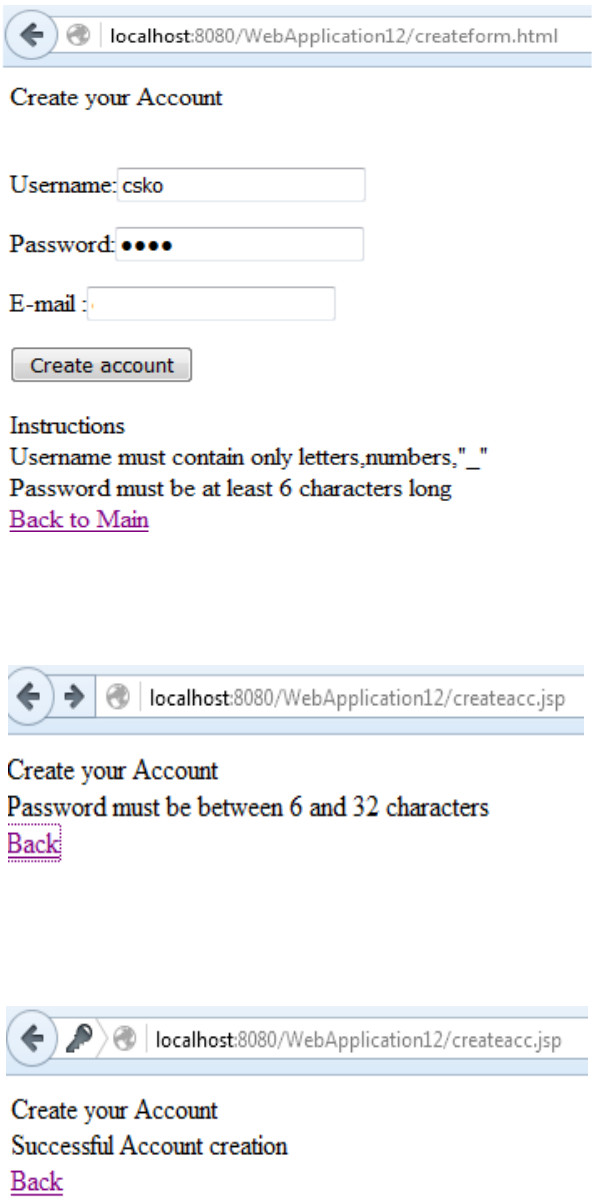
Παραθέτουμε τη βασική σελίδα createacc.jsp η οποία θα «διαβάσει» τα στοιχεία που έδωσε ο χρήστης, θα κάνει έλεγχο και αν όλα πάνε καλά θα εισάγει τα στοιχεία του λογαριασμού χρήστη στον πίνακα users. Αυτή είναι η πρώτη μας επαφή με πρόγραμμα που χρησιμοποιεί JDBC API. Στο υπόλοιπο του κεφαλαίου θα παρουσιάσουμε διεξοδικά τη διεπαφή JDBC API και τον τρόπο χρήσης.

createacc.jsp

```
<%@page import="java.util.regex.*"%>
<%@page import="java.sql.*" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%String nam=(String)request.getParameter("usname");
   String pass=(String)request.getParameter("passw");
   String mail=(String) request.getParameter("mail");%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<link rel="stylesheet" type="text/css" href="style.css">
<title>Create Account</title>
</head>
<body>
<header>Create your Account</header>
<% Pattern check=Pattern.compile("[A-Za-z0-9._-]+@[A-Za-z0-9._-]+\\.[A-Za-
z]+");
Pattern check2=Pattern.compile("\\w+");
Matcher match=check.matcher(mail);//used for email validation
Matcher match2=check2.matcher(nam);  //used for username validation
if(nam.equals("") || pass.equals("") || mail.equals("") )
{
  out.println("All fields are required<br>");
}
else if(!match.matches() && !match2.matches() ) //not valid email and
usernmame
{
  out.print("Username and email are not valid<br>");
}
else if((!match.matches())  ) //not valid email
{
  out.print("Email  not valid<br>");
```

```
}
else if((!match2.matches())) //not valid username
{
  out.print("Username not valid<br>");
}
else if(pass.length()<6)   //password < 6 characters
{
  out.println("Password must be between 6 and 32 characters <br>");
}
else //create form is valid
{
  Class.forName("com.mysql.jdbc.Driver");
  String mydb="jdbc:mysql://localhost:3306/webusers";
  Connection myconnection=DriverManager.getConnection(mydb,"root","");
  Statement mystatement=myconnection.createStatement();
  String sqlstring2="Select username,email from users where username=`"+nam+"`
OR email=`"+mail+"`";
  ResultSet res=mystatement.executeQuery(sqlstring2);
  if(res.next()) //select has records,username-email exists
  {

    out.print("Username or email already exists<br>");
  }
  else  //username and email doesnt exists
  {
    String sqlstring1="INSERT INTO users(username,password,email)

       VALUE  S(`"+nam+"`,'"+pass+"`,'"+mail+"`)";
    mystatement.executeUpdate(sqlstring1);
    out.print("Successful Account creation <br>");
  }
  myconnection.close();
  mystatement.close();
}
%>
<a href="createform.html">Back</a>
</body>
</html>
```

Στην εικόνα 12.16 βλέπουμε την ηλεκτρονική φόρμα στην οποία εισάγει στοιχεία ο χρήστης και τα μηνύματα που βγάζει η σελίδα createacc.jsp όταν τα στοιχεία είναι σωστά και όταν είναι λανθασμένα.



Εικόνα 12.16 Ηλεκτρονική φόρμα στην οποία εισάγει στοιχεία ο χρήστης και τα μηνύματα που βγάζει η σελίδα createacc.jsp.

12.2.6 Λυμένο θέμα

Θέλουμε να κατασκευάσουμε μία σελίδα στην οποία ο χρήστης θα πληκτρολογεί τα στοιχεία του. Η σελίδα θα καλεί μία jsp σελίδα η οποία θα διαβάζει τα στοιχεία και θα κάνει απλό έλεγχο σε όνομα χρήστη και συνθηματικό. Αν είναι σωστά θα βγάζει μήνυμα επιτυχούς σύνδεσης διαφορετικά θα ζητά εκ νέου τα τα στοιχεία. Οι σελίδες σας θα είναι όσο γένεται πιο απλές ώστε να επικεντρωθείτε στη σύνδεση με τη βάση δεδομένων και στη χρήση δήλωσης SELECT.

Στη συνέχεια θα παραθέσουμε ενδεικτική λύση για την κατασκευή της σελίδας loginform.html και της σελίδας login.jsp. Η σελίδα loginform.html θα εμφανίσει τη φόρμα (εικόνα 9.17) στην οποία ο χρήστης θα πληκτρολογήσει όνομα (username) και συνθηματικό (password) (εικόνα 9.18). Όταν ο χρήστης πατήσει το πλήκτρο Login θα κληθεί η σελίδα login.jsp για να «διαβάσει» τα στοιχεία που έδωσε ο χρήστης και να κάνει έλεγχο ορθότητας των στοιχείων. Αν όλα πάνε καλά, δηλαδή, αν υπάρχουν τα στοιχεία του λογαριασμού του

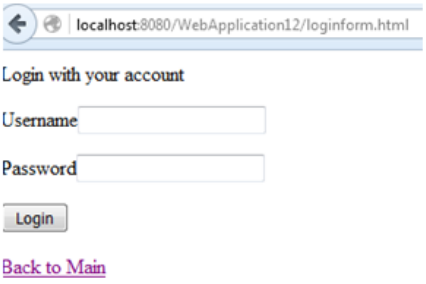
χρήστη στον πίνακα users τότε θα επιστρέψει με σχετικό μήνυμα. Διαφορετικά θα βγάλει σχετικά μηνύματα και θα επιστρέψει. Ο κώδικας της σελίδας login.jsp γράφτηκε όσο γίνεται πιο απλά. Μέσα στον κώδικα (της σελίδας login.jsp) υπάρχει εκτενέστερος σχολιασμός ώστε να κατανοήσετε τον τρόπο σύνδεσης με τη βάση και τον τρόπο ορισμού και εκτέλεσης δηλώσεων SELECT.

Παραθέτουμε τον κώδικα της σελίδας. loginform.html

loginform.html

```
<html>
<head>
<title>Login</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<header>Login with your account</header>
  <form name="Loginform" method="post" action="login.jsp">
    <p>Username<input type="text" maxlength="32" name="usnamelog">
    <p> Password<input type="password" maxlength="32" name="passwlog">
    <p> <input type="submit"  name="sublogin" value="Login">
  </form>
  <div class="ln">
    <a href="index.html">Back to Main</a>
  </div>
</body>
</html>
```

Στην εικόνα 12.16 βλέπουμε την ηλεκτρονική φόρμα στην οποία εισάγει στοιχεία ο χρήστης. Παράγεται από τη σελίδα loginform.html.



Εικόνα 12.17 Ηλεκτρονική φόρμα στην οποία εισάγει στοιχεία ο χρήστης. Παράγεται από τη σελίδα loginform.html..

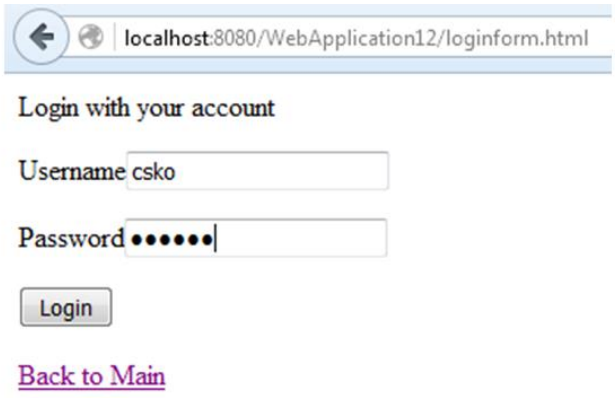
Παραθέτουμε τον κώδικα της σελίδας. login.jsp

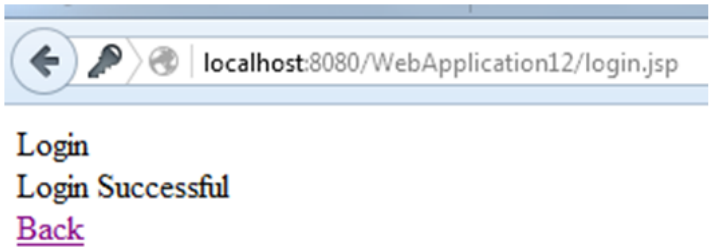
login.jsp

```
<%@page import="java.sql.*" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%String nam=(String)request.getParameter("usnamelog");
  String pass=(String)request.getParameter("passwlog");
%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Login</title>
```

```
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<header>Login</header>
<% if(nam.equals("") || pass.equals("") )
{
  out.println("All fields are required<br>");
}else {
  Class.forName("com.mysql.jdbc.Driver");
  String mydb="jdbc:mysql://localhost:3306/webusers";
  Connection myconnection=DriverManager.getConnection(mydb,"root","");
  Statement mystatement=myconnection.createStatement();
  String sqlstring2="Select username,password from users where
username='"+nam+"'AND password='"+pass+"'";//search  username+password
  String sqlstring3="Select username from users  where
username='"+nam+"'";//search username
  ResultSet res=mystatement.executeQuery(sqlstring2);
  if(res.next())  //user+password corect
  {
    out.print("Login Successful<br>");
  }
  else
  {
    res=mystatement.executeQuery(sqlstring3);//execute select for username
    if(!(res.next()))
    {
      out.print("Username does not exist<br>");
    }
    else   //username exists so the  password is wrong
      out.print( "Wrong Password<br>");
%> <a href="createform.html">Create Account<a/><%
  }

}%>
<a href="loginform.html">Back<a/>
</body>
</html>
```

Στην εικόνα 12.18 βλέπουμε την ηλεκτρονική φόρμα στην οποία εισάγει στοιχεία ο χρήστης συμπληρωμένη.. Παράγεται από τη σελίδα loginform.html. Με το πάτημα του πλήκτρου πλήκτρο Login θα κληθεί η σελίδα login.jsp για να «διαβάσει» τα στοιχεία που έδωσε ο χρήστης, να κάνει έλεγχο ορθότητας των στοιχείων και αν όλα πάνε καλά να εμφανίσει μήνυμα επιτυχούς σύνδεσης (εικόνα 2.17),





Εικόνα 12.18 Ηλεκτρονική φόρμα στην οποία εισάγει στοιχεία ο χρήστη. Καλείται η σελίδα login.jsp για να «διαβάσει» τα στοιχεία και, να κάνει έλεγχο ορθότητας στοιχείων. Αν όλα πάνε καλά θα εμφανίσει μήνυμα επιτυχούς σύνδεσης

Στη συνέχεια απομονώνουμε κάποια κρίσιμα στοιχεία του κώδικα:

Ανάγνωση τιμών που έδωσε ο χρήστης

```
<% String nam=(String)request.getParameter("usnamelog");

   String pass=(String)request.getParameter("passwlog");
%>
```

Κλήση μορφοποίησης για τις σελίδες

```
<link rel="stylesheet" type="text/css" href="style.css">
```

Καθορισμός Driver για τη βάση δεδομένων στο προϊόν ΣΔΒΔ που χρησιμοποιούμε

```
Class.forName("com.mysql.jdbc.Driver");
```

Σύνδεση στη βάση δεδομένων. Βλέπουμε ότι στην εγκατάσταση του προϊόντος δεν ορίστηκε συνθηματικό. Δηλαδή, συνδεόμαστε ως χρήστης root χωρίς συνθηματικό! Δεν είναι σωστό να εγκαταστήσουμε προϊόν χωρίς συνθηματικό για τον παντοδύναμο χρήστη root αλλά θέλαμε να δείξουμε με έμφαση ότι άλλο είναι τα συνθηματικά της βάσης δεδομένων και άλλα το συνθηματικά των χρηστών της εφαρμογής.

```
String mydb="jdbc:mysql://localhost:3306/webusers";

Connection myconnection=DriverManager.getConnection(mydb,"root","");
Statement mystatement=myconnection.createStatement();
```

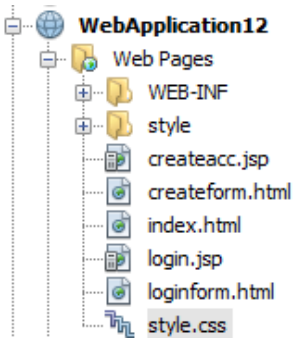
Ορισμός SELECT statement ως String. Η εκτέλεση δήλωσης SELECT επιστρέφει τα αποτελέσματα (ResultSet)

```
String sqlstring2="Select username,password from users where
username='"+nam+"'AND password='"+pass+"'";//search  username+password
String sqlstring3="Select username from users  where
username='"+nam+"'";//search username
ResultSet res=mystatement.executeQuery(sqlstring2);
```

Ακολουθεί **επεξεργασία αποτελεσμάτων ResultSet**.

Στις επόμενες ενότητες θα συζητηθεί εκτενέστερα το θέμα της σύνδεσης της εφαρμογής με βάση δεδομένων.

Στην εικόνα 12.19 βλέπουμε τα αντικείμενα της εφαρμογής WebApplication12. Περιλαμβάνονται ιστοσελίδες και αρχείο css. Για να εκτελέσουμε την εφαρμογή επιλέγουμε αρχική σελίδα και RUN στο Netbeans IDE. Στην εικόνα 12.19 μπορούμε να επιλέξουμε την εκτέλεση της σελίδας index.html (αντιστοιχεί στην εφαρμογή της προηγούμενης ενότητας 12.2.5) ή της σελίδας loginform.html (αντιστοιχεί στην εφαρμογή της παρούσης ενότητας 12.2.6). Τέλος, στην εικόνα 12.20 βλέπουμε ενδεικτικό css της εφαρμογής.



Εικόνα 12.19 Τα αντικείμενα της εφαρμογής WebApplication12. Περιλαμβάνονται ιστοσελίδες και αρχείο css. Ουσιαστικά περιλαμβάνονται δύο εφαρμογές, η πρώτη με αρχική σελίδα index.html και η δεύτερη με αρχική σελίδα loginform.html

```
@charset "UTF-8";
body{
   background-color:#D8D8D8;
    width:960px;
    margin-left: auto;
    margin-right: auto;
      text-align: center;
}
header{
     display: block;
    text-align: center;
background-color: #11223B;
color: white;
}
aside{
color:white;
   text-align:center;
}
aside div{
     display:inline-block;
     background-color: #11223B;
     padding-left: auto;
     padding-right: auto;
}
form{
    text-align: center;
}
.ln {
    text-align: center;
}
.ln a{
    display:inline-block;
}
```

Εικόνα 12.20 Ηλεκτρονικές φόρμες login.html και. login.jsp και css της εφαρμογής

12.3 Εισαγωγή με παράδειγμα στις εφαρμογές Δυναμικών Ιστοσελίδων (JSP pages, mySQL)

Έστω ότι εργάζεστε για την εταιρεία Mythical Car, μια εταιρεία πώλησης πολυτελών αυτοκινήτων, και σας ανατίθεται ο σχεδιασμός της Βάσης Δεδομένων των πωλήσεων της εταιρείας

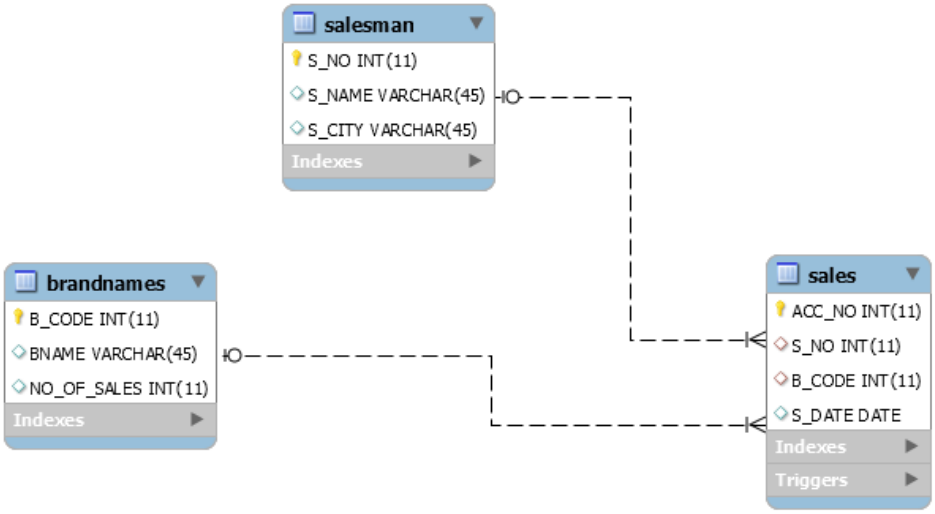
Περιορισμοί

1) Η εταιρία Mythical Car εμπορεύεται πολλές μάρκες αυτοκινήτων (Brandnames): Audi, Opel,....
2) Κάθε πωλητής (salesman) της εταιρείας πουλά αυτοκίνητα για περισσότερες από μία μάρκες.
3) Για κάθε μάρκα υπάρχει καταχωρημένος ο αριθμός πωλήσεων (no_of_sales).

Ζητούμενα

- Για την παραπάνω περιγραφή δημιουργείστε τους κατάλληλους πίνακες .
- Κατασκευάστε τη διεπαφή μέσω της οποίας ο πωλητής θα καταγράφει την πώληση ενός αυτοκινήτου (jsp σελίδες).
- Δημιουργείστε τον κατάλληλο trigger ο οποίος θα ενεργοποιείται με κάθε νέα πώληση προκειμένου να αυξήσει τον αριθμό των πωλήσεων (no_of_sales).

Παραθέτουμε αρχικά το Μοντέλο οντοτήτων συσχετίσεων και επεξηγούμε τις στήλες των πινάκων.



Παραθέτουμε την περιγραφή των πινάκων της βάσης δεδομένων.

```
DESCRIBE SALES;
DESCRIBE BRANDNAMES;
DESCRIBE SALESMAN;
```

```
mysql> DESCRIBE SALES;
+--------+---------+------+-----+---------+----------------+
| Field  | Type    | Null | Key | Default | Extra          |
+--------+---------+------+-----+---------+----------------+
| ACC_NO | int(11) | NO   | PRI | NULL    | auto_increment |
| S_NO   | int(11) | YES  | MUL | NULL    |                |
| B_CODE | int(11) | YES  | MUL | NULL    |                |
| S_DATE | date    | YES  |     | NULL    |                |
+--------+---------+------+-----+---------+----------------+
4 rows in set (0.05 sec)

mysql> DESCRIBE BRANDNAMES;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| B_CODE      | int(11)     | NO   | PRI | NULL    | auto_increment |
| BNAME       | varchar(45) | YES  |     | NULL    |                |
| NO_OF_SALES | int(11)     | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

```
mysql> DESCRIBE SALESMAN;
+--------+-------------+------+-----+---------+----------------+
| Field  | Type        | Null | Key | Default | Extra          |
+--------+-------------+------+-----+---------+----------------+
| S_NO   | int(11)     | NO   | PRI | NULL    | auto_increment |
| S_NAME | varchar(45) | YES  |     | NULL    |                |
| S_CITY | varchar(45) | YES  |     | NULL    |                |
+--------+-------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)

mysql>
```

Η βάση δεδομένων ονομάζεται mythical_car.

Αναφέρουμε τις στήλες του πίνακα brandnames (μάρκες αυτοκινήτων):

- B_CODE κωδικός μάρκας αυτοκινήτου, κύριο κλειδί, αυτόματη διαχείριση της τιμής του ως αύξοντος αριθμού (AUTO_INCREMENT) με αρχική τιμή 201.
- BNAME όνομα μάρκας αυτοκινήτου
- NO_OF_SALES πόσα αυτοκίνητα πωλήθηκαν από κάθε μάρκα. Η διαχείριση θα γίνεται αυτόματα με κατάλληλα εναύσματα.

Αναφέρουμε τις στήλες του πίνακα salesman (στοιχεία πωλητών αυτοκινήτων):

- S_NO κωδικός πωλητή, κύριο κλειδί, αυτόματη διαχείριση της τιμής του ως αύξοντος αριθμού (AUTO_INCREMENT) με αρχική τιμή 21.
- S_NAME όνομα
- S_CITY έδρα

Αναφέρουμε τις στήλες του πίνακα sales (στοιχεία πωλήσεων):

ACC_NO κωδικός πώλησης, κύριο κλειδί, αυτόματη διαχείριση της τιμής του ως αύξοντος αριθμού (AUTO_INCREMENT) με αρχική τιμή 1.

- S_NO κωδικός πωλητή, ξένο κλειδί,
- B_CODE κωδικός μάρκας αυτοκινήτου, ξένο κλειδί
- S_DATE ημερομηνία πώλησης

Παραθέτουμε δείγμα δεδομένων των πινάκων της βάσης δεδομένων. Παρατηρήστε ότι προς το παρόν δεν έχουμε στοιχεία πωλήσεων. Αυτό δηλώνεται στους πίνακες brandnames, sales.

```
SELECT * FROM sales;
SELECT * FROM brandnames;
SELECT * FROM salesman;
```

```
mysql>
mysql> SELECT * FROM brandnames;
+--------+---------+--------------+
| B_CODE | BNAME   | NO_OF_SALES  |
+--------+---------+--------------+
|    201 | PORSCHE |         NULL |
|    202 | LANCIA  |         NULL |
|    203 | FIAT    |         NULL |
+--------+---------+--------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM salesman;
+------+--------+----------+
| S_NO | S_NAME | S_CITY   |
+------+--------+----------+
|   21 | CODD   | ATHENS   |
|   22 | DATE   | NEW YORK |
|   23 | ULMAN  | ATHENS   |
+------+--------+----------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM sales;
Empty set (0.00 sec)

mysql>
```

Παραθέτουμε δηλώσεις δημιουργίας της βάσης δεδομένων και των πινάκων της. Οι δηλώσεις μπορούν να είναι αποτέλεσμα λήψης αντιγράφου (back up) της βάσης με χρήση του προϊόντος MySQL Workbwench.

Δημιουργία βάσης δεδομένων και πινάκων

```
DROP DATABASE mythical_car;
CREATE DATABASE mythical_car;
USE mythical_car;

-- -----------------------------------------------------
-- Table `mythical_car`.`brandnames`
-- -----------------------------------------------------
CREATE TABLE brandnames (
  B_CODE INT(11) NOT NULL AUTO_INCREMENT,
  BNAME VARCHAR(45) NULL DEFAULT NULL,
  NO_OF_SALES INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (B_CODE))DROP DATABASE mythical_car;
CREATE DATABASE mythical_car;
USE mythical_car;

-- -----------------------------------------------------
-- Table `mythical_car`.`brandnames`
-- -----------------------------------------------------
CREATE TABLE brandnames (
  B_CODE INT(11) NOT NULL AUTO_INCREMENT,
  BNAME VARCHAR(45) NULL DEFAULT NULL,
  NO_OF_SALES INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (B_CODE))
ENGINE = InnoDB
AUTO_INCREMENT = 201
DEFAULT CHARACTER SET = utf8;
```

```
-- -----------------------------------------------------
-- Table `mythical_car`.`salesman`
-- -----------------------------------------------------
CREATE TABLE salesman (
  S_NO INT(11) NOT NULL AUTO_INCREMENT,
  S_NAME VARCHAR(45) NULL DEFAULT NULL,
  S_CITY VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (S_NO))
ENGINE = InnoDB
AUTO_INCREMENT = 21
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `mythical_car`.`sales`
-- -----------------------------------------------------
CREATE TABLE sales (
  ACC_NO INT(11) NOT NULL AUTO_INCREMENT,
  S_NO INT(11) NULL DEFAULT NULL,
  B_CODE INT(11) NULL DEFAULT NULL,
  S_DATE DATE NULL DEFAULT NULL,
  PRIMARY KEY (ACC_NO),
  INDEX S_NO (S_NO ASC),
  INDEX B_CODE (B_CODE ASC),
  CONSTRAINT `sales_ibfk_1`
FOREIGN KEY (S_NO)
REFERENCES salesman (S_NO),
  CONSTRAINT sales_ibfk_2
FOREIGN KEY (B_CODE)
REFERENCES brandnames (B_CODE))
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8;

ENGINE = InnoDB
AUTO_INCREMENT = 201
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `mythical_car`.`salesman`
-- -----------------------------------------------------
CREATE TABLE salesman (
  S_NO INT(11) NOT NULL AUTO_INCREMENT,
  S_NAME VARCHAR(45) NULL DEFAULT NULL,
  S_CITY VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (S_NO))
ENGINE = InnoDB
AUTO_INCREMENT = 21
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `mythical_car`.`sales`
-- -----------------------------------------------------
```

```
CREATE TABLE sales (
  ACC_NO INT(11) NOT NULL AUTO_INCREMENT,
  S_NO INT(11) NULL DEFAULT NULL,
  B_CODE INT(11) NULL DEFAULT NULL,
  S_DATE DATE NULL DEFAULT NULL,
  PRIMARY KEY (ACC_NO),
  INDEX S_NO (S_NO ASC),
  INDEX B_CODE (B_CODE ASC),
  CONSTRAINT `sales_ibfk_1`
FOREIGN KEY (S_NO)
REFERENCES salesman (S_NO),
  CONSTRAINT sales_ibfk_2
FOREIGN KEY (B_CODE)
REFERENCES brandnames (B_CODE))
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8;
```

Παραθέτουμε ορισμό εναύσματος. Ενεργοποιείται με κάθε νέα πώληση και ενημερώνει τον πίνακα των αυτοκινήτων με τις μέχρι στιγμής πωλήσεις των αυτοκινήτων.

```
DELIMITER $$

USE `mythical_car`$$
DROP TRIGGER IF EXISTS `mythical_car`.`TOTAL_SALES` $$
USE `mythical_car`$$

CREATE
DEFINER=`root`@`localhost`
TRIGGER `mythical_car`.`TOTAL_SALES`
BEFORE INSERT ON `mythical_car`.`sales`
FOR EACH ROW
UPDATE BRANDNAMES SET NO_OF_SALES=IFNULL(NO_OF_SALES,0)+1
WHERE BRANDNAMES.B_CODE=NEW.B_CODE$$

DELIMITER ;
```

Παραθέτουμε την εισαγωγή στοιχείων στους πίνακες brandnames και salesman.

```
INSERT INTO brandnames(BNAME) VALUES ('PORSCHE');
INSERT INTO brandnames(BNAME) VALUES ('LANCIA');
INSERT INTO brandnames(BNAME) VALUES ('FIAT');

INSERT INTO salesman(S_NAME, S_CITY)VALUES('CODD','ATHENS');
INSERT INTO salesman(S_NAME,S_CITY) VALUES('DATE','NEW YORK');
INSERT INTO salesman(S_NAME,S_CITY) VALUES('ULMAN','ATHENS');
```

Δείτε τα στοιχεία των πινάκων.

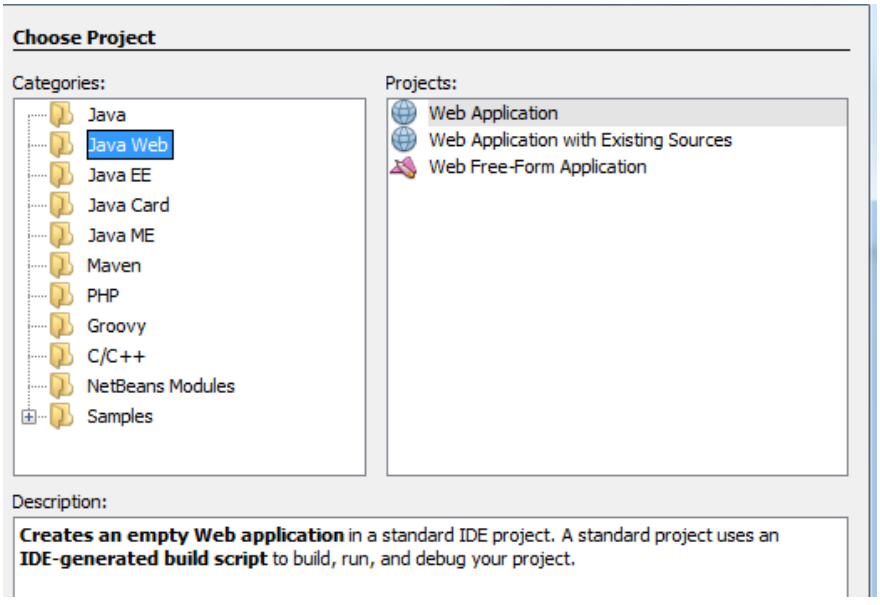
```
SELECT * FROM brandnames;
SELECT * FROM salesman;
```

```
mysql> SELECT * FROM brandnames;
+--------+---------+--------------+
| B_CODE | BNAME   | NO_OF_SALES  |
+--------+---------+--------------+
|    201 | PORSCHE |         NULL |
|    202 | LANCIA  |         NULL |
|    203 | FIAT    |         NULL |
+--------+---------+--------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM salesman;
+------+--------+----------+
| S_NO | S_NAME | S_CITY   |
+------+--------+----------+
|   21 | CODD   | ATHENS   |
|   22 | DATE   | NEW YORK |
|   23 | ULMAN  | ATHENS   |
+------+--------+----------+
3 rows in set (0.00 sec)
```

Ο πίνακας των πωλήσεων sales δεν έχει στοιχεία. Θα κατασκευάσουμε εφαρμογή η οποία θα περιλαμβάνει σελίδα στην οποία θα πληκτρολογούνται στοιχεία πώλησης και στη συνέχεια άλλη σελίδα θα εισάγει τα δεδομένα αυτά στον πίνακα sales. Το έναυσμα θα εκτελείται με κάθε νέα πώληση και θα ενημερώνει τη στήλη no_of_sales του πίνακα brandnames.

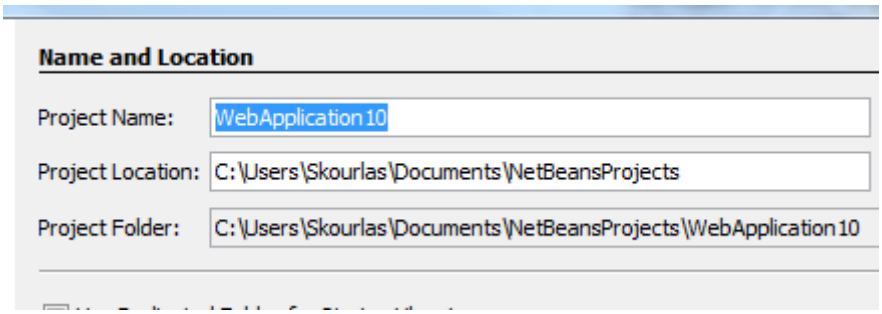
12.3.1 Κατασκευή JSP pages για τη βάση δεδομένων της εταιρείας Mythical Car

Στην εικόνα 12.21 επιλέγουμε για την εφαρμογή Mythical car την κατηγορία εφαρμογών (categories) Java Web και ως τύπο project Web Application.



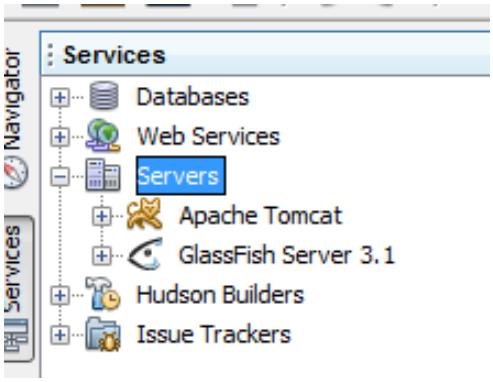
Εικόνα 12.21 Επιλογή για την εφαρμογή Mythical car κατηγορίας εφαρμογών) Java Web και τύπου έργου Web Application.

Στην εικόνα 12.22 επιλέγουμε για την εφαρμογή Mythical car το όνομα (Web Application10), θέση και περιοχή του project Web Application10.



Εικόνα 12.22 Επιλογή ονόματος (Web Application10), θέσης και περιοχής του project για την εφαρμογή Mythical car

Στην εικόνα 12.23 επιλέγουμε τον διακομιστή μας για τις ιστοσελίδες του project Web Application10. Μπορούμε να επιλέξουμε GlassFish server ή Apache server.



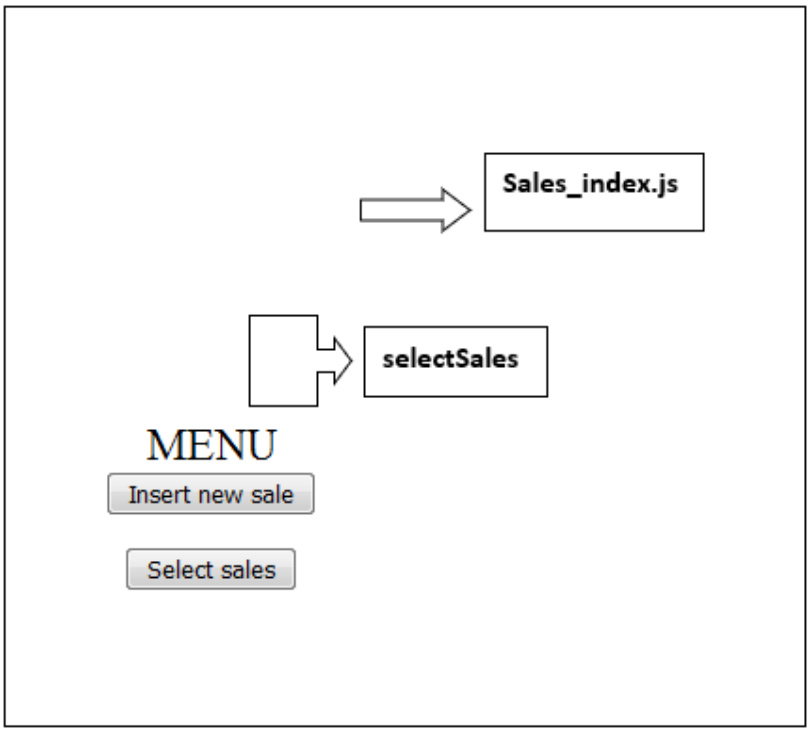
Εικόνα 12.23 Επιλογή διακομιστή για τις ιστοσελίδες του project Web Application10, GlassFish server ή Apache server.

Παραθέτουμε τον ορισμό της σελίδας menu.jsp η οποία θα εμφανίζει το μενού της εφαρμογής μας.

menu.jsp

```
<%--
Document   : menu
Created on : Jun 3, 2020, 3:53:08 PM
Author : Christos
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>MENU</title>
</head>
<body>
<center>
  <font size="+2"> MENU </font>
  <form name="insertsales" method="post" action="sales_index.jsp">
    <input type="submit" value="Insert new sale"><p>
  </form>
  <form name="selectsales" method="post" action="selectSales.jsp">
    <input type="submit" value="Select sales">
  </form>
</center>
</body>
</html>
```

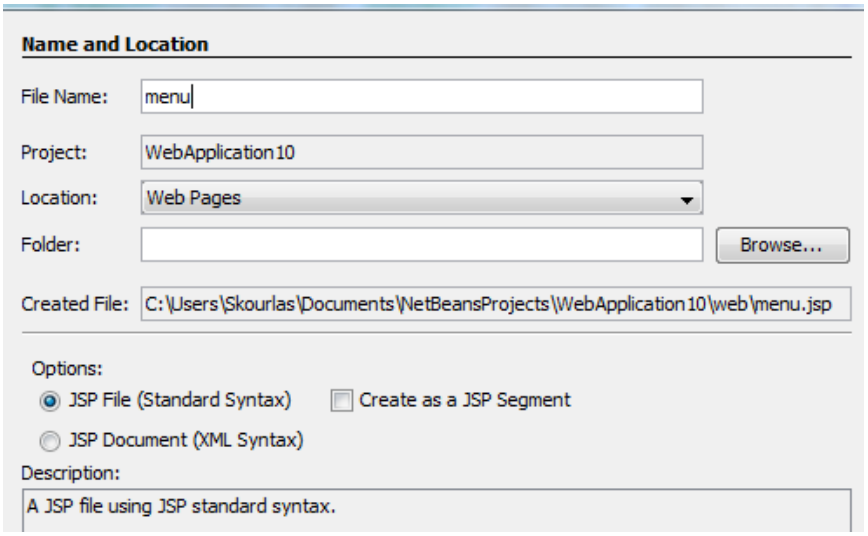
Στην εικόνα 12.24 βλέπουμε το μενού το οποίο εμφανίζει η σελίδα menu.jsp. Ανάλογα με την επιλογή του χρήστη καλείται η σελίδα Sales_index.jsp η οποία επιτρέπει την εισαγωγή νέας πώλησης ή η σελίδα selectSales η οποία δείχνει τις πωλήσεις.



Εικόνα 12.24 Μενού το οποίο εμφανίζει η σελίδα menu.jsp. Ανάλογα με την επιλογή του χρήστη καλείται η σελίδα Sales_index.jsp η οποία επιτρέπει την εισαγωγή νέας πώλησης ή η σελίδα selectSales η οποία δείχνει τις πωλήσεις

Στην εικόνα 12.25 βλέπουμε πως ορίζουμε το όνομα και τη θέση της σελίδας menu.jsp..

Στην εικόνα 12.26 βλέπουμε πως κατασκευάζουμε τη σελίδα menu.jsp. Ουσιαστικά κάνουμε copy & paste στο Nertbeans IDE τον κώδικα της σελίδας menu.jsp που είδαμε παραπάνω.

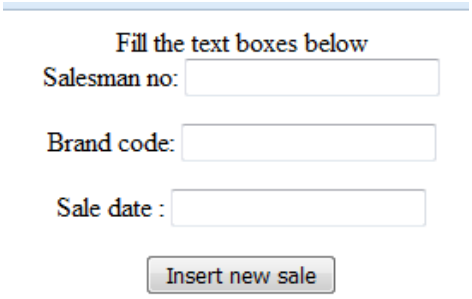


Εικόνα 12.25 Ορισμός ονόματος και θέσης της σελίδας menu.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>MENU</title>
    </head>
    <body>
    <center>
        <font size="+2"> MENU </font>
        <form name="insertsales" method="post" action="sales_index.jsp">
            <input type="submit" value="Insert new sale"><p>
         </form>
         <form name="selectsales" method="post" action="selectSales.jsp">
            <input type="submit" value="Select sales">
        </form>
     </center>
    </body>
</html>
```

Εικόνα 12.26 Κατασκευή της σελίδα menu.jsp. Ουσιαστικά κάνουμε copy & paste στο Nertbeans IDE τον κώδικα της σελίδας menu.jsp

Στην εικόνα 12.27 βλέπουμε τη φόρμα εισαγωγής στοιχείων πώλησης.



Εικόνα 12.27 Φόρμα εισαγωγής στοιχείων πώλησης την οποία εμφανίζει η σελίδα sales_index.jsp

Παραθέτουμε τον ορισμό της σελίδας sales_index.jsp η οποία θα εμφανίζει τη φόρμα εισαγωγής των στοιχείων πώλησης.

sales_index.jsp

```
<%--
Document   : sales_index
Created on : Jun 3, 2020, 3:16:03 PM
Author : Christos
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Insert new sale</title>
</head>
<body>
  <center>
    Fill the text boxes below
    <form name="salesIndex" method="post" action="salesInsert.jsp">
```

```
      Salesman no: <input type="text" name="s_no" > <p>
      Brand  code: <input type="text" name="b_code" > <p>
      Sale date : <input type="text" name="s_date" ><p>
      <input type="submit" value="Insert new sale" >
    </form>
  </center>
</body>
</html>
```

Βλέπετε ότι η σελίδα sales_index.jsp καλεί τη σελίδα salesInsert.jsp η οποία θα «διαβάσει» τα στοιχεία της πώλησης και θα τα γράψει στον πίνακα των πωλήσεων.

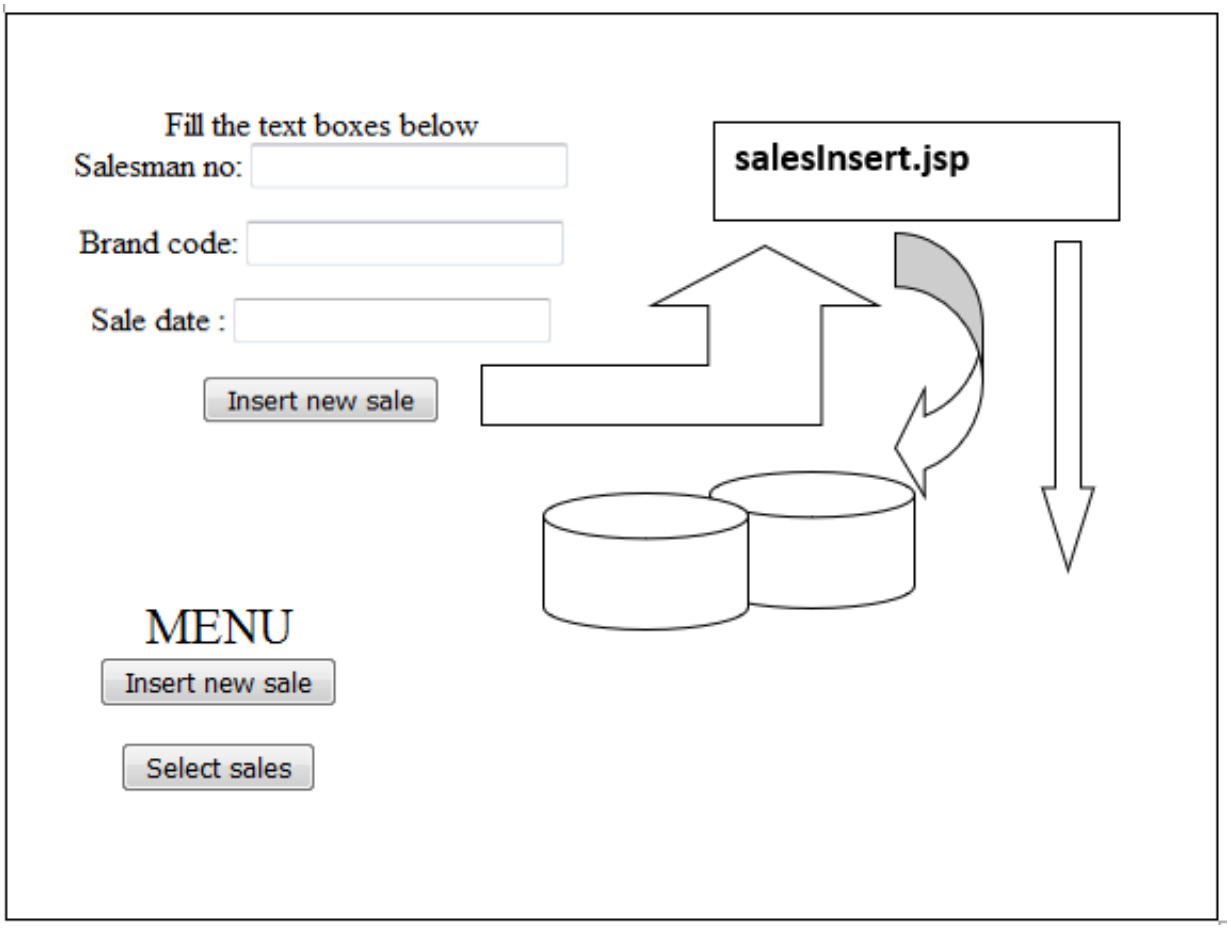
Παραθέτουμε τον κώδικα της σελίδας salesInsert.jsp.

salesInsert.jsp

```
<%--
Document   : salesInsert
Created on : Jun 3, 2018, 3:20:28 PM
Author : Christos
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<%
int s_no=Integer.parseInt(request.getParameter("s_no"));
int b_code=Integer.parseInt(request.getParameter("b_code"));
String s_date=request.getParameter("s_date");
%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<%
Class.forName("com.mysql.jdbc.Driver");
// Define the connection parameters
String myDatabase =
"jdbc:mysql://localhost:3306/MYTHICAL_CAR?user=root&password=1234";
// Connect to the database
Connection myConnection = DriverManager.getConnection(myDatabase);
// Create object to execute statements
Statement myStatement = myConnection.createStatement();
String SQLstring="INSERT INTO SALES (S_NO,B_CODE,S_DATE) VALUES
(\'"+s_no+"\',\'"+b_code+"\',\'"+s_date+"\') ";

myStatement.executeUpdate(SQLstring);
myStatement.close();
myConnection.close();


%>
<form name="goBack" method="post" action="menu.jsp">
<script>
document.forms["goBack"].submit();
</script>
</form>
```

```
</body>
</html>
```

Στην εικόνα 12.28 βλέπουμε το Μενού το οποίο εμφανίζει η σελίδα menu.jsp. Ο χρήστης επιλέγει τη σελίδα Sales_index.jsp η οποία επιτρέπει την εισαγωγή νέας πώλησης, εμφανίζεται η φόρμα εισαγωγής στοιχείων πώλησης, τα στοιχεία πληκτρολογούνται και τελικά καλείται η σελίδα salesInsert.jsp η οποία διαβάζει τα στοιχεία και τα γράφει στη βάση δεδομένων.



Εικόνα 12.28 Κύκλος από την εμφάνιση του μενού μέχρι την εισαγωγή στοιχείων πώλησης στη βάση δεδομένων.

Παραθέτουμε τον κώδικα της σελίδας selectSales.jsp η οποία εμφανίζει τα στοιχεία πωλήσεων.

selectSales.jsp

```
<%--
Document   : selectSales
Created on : Jun 3, 2020, 3:52:48 PM
Author : Christos
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Sales</title>
</head>
<body>
```

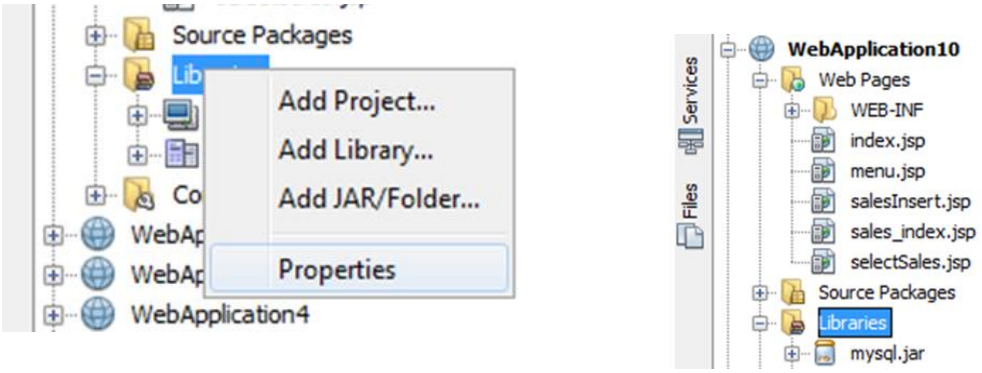
```
<%
Class.forName("com.mysql.jdbc.Driver");
// Define the connection parameters
String myDatabase =
"jdbc:mysql://localhost:3306/MYTHICAL_CAR?user=root&password=1234";
// Connect to the database
Connection myConnection = DriverManager.getConnection(myDatabase);
// Create object to execute statements
Statement myStatement = myConnection.createStatement();
String SQLstring="SELECT * FROM SALES; ";
ResultSet rs=myStatement.executeQuery(SQLstring);
int [] acc_no=new int[200];
int [] s_no=new int[200];
int [] b_no=new int[200];
String [] s_date= new String[200];
int j=0;
while(rs.next())
{
  acc_no[j]=rs.getInt("ACC_NO");
  s_no[j]=rs.getInt("S_NO");
  b_no[j]=rs.getInt("B_CODE");
  s_date[j]=rs.getString("S_DATE");
  j++;
}
for(int i=0; i<j; i++)
{

%>
  Sales_no:<%= acc_no[i]%> <p>
  Salesman_no: <%= s_no[i]%> <p>
  Brand code: <%= b_no[i]%> <p>
  Sales date: <%= s_date[i]%> <p>
  <p><p><p>
<%
}
myStatement.close();
myConnection.close();
%>
<form name="goBack" method="post" action="menu.jsp">
<input type="submit" value="GO BACK TO MENU" >
</form>
</body>
</html>
```

Στην εικόνα 12.29 βλέπουμε τα στοιχεία πωλήσεων. Η αναφορά (report) κατασκευάζετε από τη σελίδα selectSales.jsp

Εικόνα 12.29 Στοιχεία πωλήσεων.

Σημαντική παρατήρηση

Όταν κατασκευάζουμε την εφαρμογή μας πρέπει να «δηλώσουμε» driver. Δείτε σχετικά και την εικόνα 12.30.



Εικόνα 12.30 Δήλωση driver για εφαρμογές σε MySQL.

12.4 Επισκόπηση της χρήσης jDBC API. Μελέτη Περίπτωσης. Διαχείριση προσωπικού με εφαρμογή Δυναμικών Ιστοσελίδων. Τεχνολογία JSP pages, mySQL

Στόχος της ενότητας είναι η επισκόπηση της χρήσης του JDBC API για τη δημιουργία εφαρμογών διαχείριση βάσεων δεδομένων με τεχνολογία JSP PAGES. Μετά την επεξεργασία της ενότητας ο ενδιαφερόμενος θα έχει κατανοήσει τα θέματα δημιουργίας μιας εφαρμογής τεχνολογίας JSP PAGES.

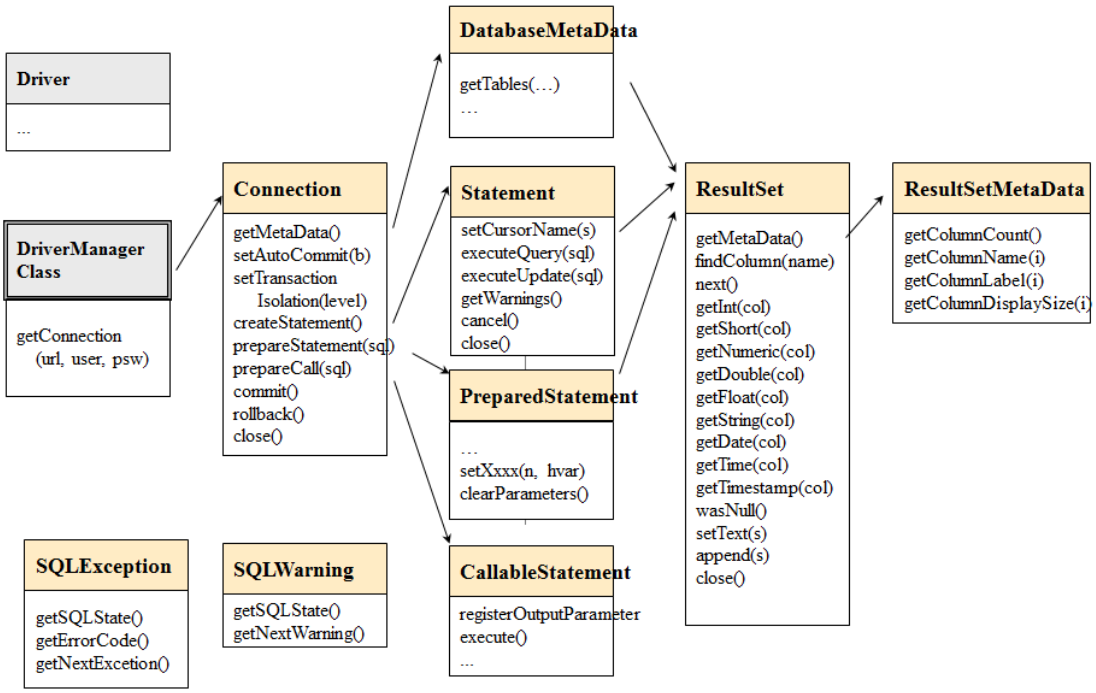
12.4.1 Επισκόπηση της χρήσης JDBC API

Στην εικόνα 12.31 (Martii Leiho) προσφέρεται μία οπτική υπενθύμιση για μία σειρά από σημαντικές κλάσεις, αντικείμενα, μεθόδους (class, objects, methods) του jdbc API και με τα βέλη περιγράφεται η επικοινωνία / σύνδεσή/διάταξή τους. Στο e-book,

Martti Laiho, Matti Kurki, Malcolm Crowe, Fritz Laux, Dimitris Dervos and Kari Hirvonen (2018), Introduction to Transaction Programming, Draft 2018-12-03, DBTechNet.org, p. 654

το οποίο διατίθεται με άδεια χρήσης creative commons παρατίθενται πάρα πολλά στοιχεία για εφαρμογές με χρήση JDBC API.

[(20) (PDF) Introduction toTransaction Programming (researchgate.net)]



Εικόνα 12.31 JDBC API (Martii Leiho)

Η βασική αρχή χρήσης του jdbc API είναι απλή:

Ας εστιάσουμε στα βέλη. Για να ορίσω ένα αντικείμενο μίας κλάσης στην αιχμή ενός βέλους χρησιμοποιώ την κατάλληλη μέθοδο και αντικείμενο της «προηγούμενης» κλάσης που βρίσκεται στην αρχή του ιδίου βέλους.

```
Statement myStatement = myconnection.createStatement();
```

Δηλαδή για να ορίσουμε το αντικείμενο myStatement «εφαρμόζουμε» τη μέθοδο createStatement στο αντικείμενο myconnection.

Τα αντικείμενα της βάσης δεδομένων ορίζονται σαν συμβολοσειρές (string). Ακολουθούν παραδείγματα:

```
  String testDatabase =
"jdbc:mysql://localhost:3306/mydb?user=admin&password=1234";
```

Προσοχή! Η παρακάτω εντολή είναι λανθασμένη

```
  String sqlString = "INSERT INTO USERS VALUES ('user1', 'password1')";
```

Η σωστή διαμορφώνεται ως εξής: … VALUES('"+user1+"','"+password1+"')

Άρα, String sql="insert into users values ('"+user1+"','"+password1+"') ";

12.4.2 Απλά λειτουργικά παραδείγματα χρήσης του API:

Η DriverManagerClass παρέχει τη μέθοδο getConnection. Μπορούμε να ορίσουμε ένα αντικείμενο, για παράδειγμα το αντικείμενο myConnection τύπου Connection, ως εξής:

```
Connection myConnection = DriverManager.getConnection(testDatabase);
```

Αυτό προϋποθέτει ότι έχουμε ορίσει ένα αντικείμενο testDatabase τύπου string που «ορίζει» τον driver που χρησιμοποιούμε, το URL, τη βάση μας, username και password. Άρα θα έχουμε:

```
String testDatabase =
"jdbc:mysql://localhost:3306/mydb?user=admin&password=1234";
Connection myConnection = DriverManager.getConnection(testDatabase);
```

Σημείωση: Το αντικείμενο myConnection είναι τύπου Connection και μπορούμε να χρησιμοποιήσουμε τη μέθοδο CreateStatement της κλάσης Connection για να ορίσουμε ένα αντικείμενο, για παράδειγμα το αντικείμενο myStatement τύπου Statement. Μέσω του αντικειμένου myStatement θα γράφουμε και θα εκτελούμε δηλώσεις SQL (INSERT, UPDATE, DELETE, SELECT).

```
Statement myStatement = myconnection.createStatement();
```

Το αντικείμενο myStatement είναι τύπου Statement και μπορούμε να το «εκτελέσουμε» χρησιμοποιώντας μία από τις δύο σχετικές μεθόδους της κλάσης δηλαδή τη μέθοδο executeUpdate ή τη μέθοδο executeQuery. Ποια είναι κατάλληλη μέθοδος εξαρτάται από τη δήλωση SQL. Στην περίπτωση δήλωσης INSERT/UPDATE/DELETE έχουμε:

```
myStatement.executeUpdate(sqlString);
```

Η συγγραφή της SQL δήλωσης γίνεται σε ένα αντικείμενο τύπου string, για παράδειγμα στο αντικείμενο με όνομα sqlString. Δείτε και την ορθή γραφή της δήλωσης και το λανθασμένο τμήμα που έχουμε διαγράψει.

```
String sqlString = "INSERT INTO USERS VALUES ('"+user1+"','"+password1+"')
('user1', 'password1')" ;
```

Έτσι για να εκτελέσουμε μία SQL δήλωση:

```
  String sql="insert into users  values ('"+user1+"','"+password1+"')";

  myStatement.executeUpdate(sqlString);
```

Ειδικά στην περίπτωση δήλωσης SELECT πρέπει να ορίσουμε ένα αντικείμενο, για παράδειγμα το αντικείμενο με όνομα rs, τύπου ResultSet. Στο αντικείμενο «αποθηκεύονται» τα αποτελέσματα της δήλωσης SELECT. Προηγουμένως, θα χρειασθεί να ορίσουμε το αντικείμενο sqlString που είναι μια συμβολοσειρά που «περιέχει» τη δήλωση SELECT:

```
String sqlString = "SELECT * FROM MYTABLE WHERE PARAM = '"+paramValue+"'";
ResultSet rs = myStatement.executeQuery(sqlString);
```

Η κλάση ResultSet παρέχει μεθόδους για να διαχειριστούμε τις γραμμές που «αποθηκεύονται» στο rs:

next(), previous(), first(), last()

Επίσης, παρέχει μία σειρά από μεθόδους κάθε μία από τις οποίες μπορεί να χρησιμοποιηθεί για να πάρουμε πληροφορία από μια στήλη:

getString, getInt, getFloat, getDate.

12.4.3 Σύνδεση σε βάση δεδομένων στην περίπτωση του προϊόντος MySQL

Για να συνδεθούμε στη βάση δεδομένων στην περίπτωση του προϊόντος MySQL και να γράψουμε δηλώσεις SQL (SQL statements) απαιτούνται οι εξής ενέργειες:

1. **Δήλωση του JDBC driver**

   libraries>δεξί κλικ> Add Jar/Folder και επιλέγουμε το αρχείο mysql-connector-java- 5.1.13 -bin.jar ή mysql.jar

2. **Εισαγωγή του package java.sql στο πρόγραμμά μας**

   <%@page import="java.sql.*" %> (χρησιμοποιούμε .* έτσι ώστε να μπορούμε να χρησιμοποιήσουμε όλες τις κλάσεις του ***package java.sql***)

3. **«Φόρτωση» του JDBC driver στο πρόγραμμά μας**

   Class.forName("com.mysql.jdbc.Driver");

   // Φορτώνουμε την κλάση Driver από το package com.mysql.jdbc.

4. **Σύνδεση στη βάση δεδομένων**

   String testDatabase = "jdbc:mysql://localhost:3306/mydb?user=admin&password=1234";

   // Ορίζουμε για τη βάση δεδομένων μας (έστω ότι την ονομάζουμε mydb) ένα αντικείμενο testDatabase τύπου συμβολοσειράς (string) της μορφής:

   "jdbc:mysql://ServerName:ServerPort/DatabaseName?user=myUsername&password=myPassword";

   Connection myConnection = DriverManager.getConnection(testDatabase);

   // Δημιουργούμε ένα αντικείμενο τύπου *Connection.* Το αντικείμενό μας, myConnection, είναι μια ανοιχτή σύνδεση με τη βάση δεδομένων

5. **Δημιουργία αντικειμένου για να γράψουμε SQL statements**

   Statement myStatement = myconnection.createStatement();

   //Δημιουργούμε το αντικείμενο myStatement τύπου Statement. Μέσω του αντικειμένου myStatement θα γράφουμε δηλώσεις SQL.

6. **Εγγραφή SQL statements**

Λανθασμένη εντολή την οποία και διαγράφουμε.

~~String sqlString = "INSERT INTO USERS VALUES (‘user1’, ‘password1’)";~~

Σωστή εντολή

String sql="insert into users values (‘"+user1+"‘,’"+password1+"‘)";

//Ορίζουμε ένα αντικείμενο sqlString τύπου string για να γράψουμε τη δήλωση SQL (που θέλουμε να εκτελεστεί στη συνέχεια).

7. **Εκτέλεση SQL statements**

   myStatement.executeUpdate(sqlString);

   //Εκτελούμε τη δήλωση SQL που γράψαμε και πιο συγκεκριμένα χρησιμοποιούμε το αντικείμενο myStatement που δημιουργήσαμε. Επειδή έχουμε στην περίπτωση αυτή εντολή INSERT χρησιμοποιήσαμε τη μέθοδο executeUpdate. Το ίδιο θα κάναμε και σε δήλωση UPDATE ή DELETE.

   Αν η sqlString αφορούσε μια SQL statement τύπου select, η μέθοδός μας θα ήταν η executeQuery:

   myStatement.executeQuery(sqlString);

8. **Κλείσιμο του αντικειμένου «αναγραφής» SQL statements**

   myStatement.close();

   //Αφού εκτελέσουμε την SQL statement πρέπει να κλείσουμε το αντικείμενο που χρησιμοποιείται για την εγγραφή της SQL statements.

9. **Κλείσιμο της σύνδεσης**

   myConnection.close();

   // Κλείνουμε τη σύνδεση με τη βάση δεδομένων.

12.4.4 Παραδείγματα διαχείρισης δηλώσεων SQL - Insert-delete-update-select statements

Για να γράψουμε μια δήλωση SQL, χρησιμοποιούμε τη μέθοδο

```
myStatement.methodName(sqlString);
```

όπου myStatement είναι ένα αντικείμενο τύπου Statement και methodName είναι μια από τις μεθόδους που παρέχει η κλάση Statement.

Θα μελετήσουμε διάφορες μορφές της μεθόδου methodName.

Στη συνέχεια, διορθώστε το διαγραμμένο μέρος των δηλώσεων.

Εισαγωγή δεδομένων (insert statement)

Χρησιμοποιούμε τη μέθοδο executeUpdate μέσω της εντολής:

```
myStatement.executeUpdate(sqlString);
```

όπου sqlString είναι μια συμβολοσειρά της μορφής:

```
String sqlString = "INSERT INTO MYTABLE VALUES ('value1', 'value2')";
```

Διαγραφή δεδομένων (delete statement)

Χρησιμοποιούμε τη μέθοδο executeUpdate μέσω της εντολής:

```
myStatement.executeUpdate(sqlString);
```

όπου το sqlString είναι της μορφής:

```
String sqlString = "DELETE FROM MYTABLE WHERE PARAM = 'paramValue'";
```

Ενημέρωση δεδομένων (update statement)

Χρησιμοποιούμε τη μέθοδο executeUpdate μέσω της εντολής:

```
myStatement.executeUpdate(sqlString);
```

όπου το sqlString είναι της μορφής:

```
String sqlString = "UPDATE MYTABLE SET PARAM1='value1' WHERE PARAM2='value2'";
```

Επιλογή δεδομένων (select statement)

Για επιλογή δεδομένων χρησιμοποιούμε τη μέθοδο executeQuery μέσω της

εντολής:

```
ResultSet rs = myStatement.executeQuery(sqlString);
```

όπου sqlString είναι μια συμβολοσειρά της μορφής:

```
String sqlString = "SELECT * FROM MYTABLE WHERE PARAM = 'paramValue'";
```

Η μέθοδος *executeQuery* μας δίνει τη δυνατότητα να ανακτήσουμε γραμμές δεδομένων από τη βάση δεδομένα. Τα δεδομένα αυτά πρέπει να τα αποθηκεύσουμε προσωρινά σε κάποια δομή ώστε να τα χρησιμοποιήσουμε.

Η γλώσσα Java παρέχει μια εύχρηστη δομή μέσω της κλάσης *ResultSet*. Πιο συγκεκριμένα, η μέθοδος *ExecuteQuery* μας επιστρέφει ένα αντικείμενο τύπου *ResultSet*. Ας ονομάσουμε το συγκεκριμένο αντικείμενο *rs*. Μέσα στο αντικείμενο αυτό αποθηκεύονται προσωρινά όλες οι γραμμές δεδομένων που επιστρέφει η δήλωση SELECT που εκτελέσαμε. Μέσω του αντικειμένου αυτού μπορούμε να επεξεργαστούμε τις γραμμές στο πρόγραμμά μας

Το αντικείμενο *rs* μας παρέχει και έναν δείκτη ο οποίος αρχικά δείχνει πρίν από την πρώτη γραμμή αποτελεσμάτων που είναι αποθηκευμένα στο αντικείμενο *rs*.

Το rs (ως αντικείμενο της κλάσης ResultSet) μας παρέχει μεθόδους για να επεξεργαζόμαστε τις γραμμές που ανακτήσαμε από τη βάση δεδομένων. Οι πιο σημαντικές μέθοδοι αναφέρονται παρακάτω:

Για να χρησιμοποιήσουμε την πληροφορία που είναι καταχωρημένη στο αντικείμενο rs, θα πρέπει:

- Να μετακινήσουμε το δείκτη στη γραμμή που θέλουμε να επεξεργασθούμε ή να τον μετακινούμε διαδοχικά αν θέλουμε να επεξεργασθούμε όλες τις αποθηκευμένες γραμμές. Αυτό γίνεται με χρήση των μεθόδων: **next(), previous(), first(), last()**
- Να χρησιμοποιήσουμε την κατάλληλη μέθοδο για να πάρουμε πληροφορία από μια στήλη. Αυτό γίνεται με τις συναρτήσεις **getString, getInt, getFloat getDate** κ.τ.λ.

Πρακτικός κανόνας για τη συγγραφή δήλωσης SQL στα προγράμματά μας:

Δοκιμάζουμε τη δήλωση με συγκεκριμένες τιμές στις μεταβλητές.

Τοποθετούμε την εντολή σε " " και αντικαθιστούμε κάθε συγκεκριμένη τιμή, για παράδειγμα την τιμή 'ANALYST' με το όνομα της μεταβλητής γραμμένο με "+ +", ως εξής: '"+job+"'

Δείτε και το παράδειγμα:

```
String sql="insert into emp(ename,job,sal,deptno) values
(\'"+name+"\',\'"+job+"\',\'"+sal+"\',"+deptno+")";
```

12.4.5 Παραδείγματα χρήσης embedded statement

Ακολουθεί παράδειγμα χρήσης της executeUpdate για Insert statement. Επιπλέον, δίδεται και παράδειγμα χρήσης της executeQuery που βοηθά στην κατανόηση του τρόπου που χρησιμοποιούμε τις διάφορες μεθόδους της κλάσης ResultSet.

Αρχικά παραθέτουμε μία To-do list για τη σύνδεση σελίδων με βάση δεδομένων.

12.4.5.1 To-do list – Σύνδεση σελίδων με βάση.

1. **Δήλωση του JDBC driver στο περιβάλλον Netbeans**
2. **Εισάγετε το package java.sql στο πρόγραμμά σας:**

   <%@page import="java.sql.*" %>
3. **Σύνδεση στη βάση**

   Για παράδειγμα, οι παράμετροι της βάσης είναι:

   IP= localhost, Port= 3306, Database= users_login, Username= root, Password= 1234

   String DB = "jdbc:mysql://localhost:3306/users_login?user=root&password=1234";

   Connection myConnection = DriverManager.getConnection(DB);
4. **Δημιουργία αντικειμένου για να γράψουμε SQL statement:**

   Statement SMT = myConnection.createStatement();
5. **Αναγραφή SQL statement στο αντικείμενο SMT και εκτέλεση:**

   String sql="SELECT * FROM user WHERE Uname='"+a_user+"'AND Upass='"+a_pass+"' ";

   SMT.executeUpdate(sql); -- insert, update

   ResultSet rs=SMT.executeQuery(sql); -- select

   found= rs.first();

   if (found){ …} else { … }
6. **Κλείσιμο αντικειμένου «αναγραφής» SQL statement:**

   SMT.close();
7. **Κλείσιμο σύνδεσης:**

   myConnection.close();

12.4.5.2 Προβολή του περιεχομένου του πίνακα

Έστω ότι έχουμε τον πίνακα myTable της βάσης δεδομένων mydb.

myTable

ID	Name
Κωδικός πελάτη	Όνομα
Int	Text

Βλέπουμε τα στοιχεία του πίνακα.

```
select * from myTable
```

ID	Name
1	John
2	Ann
3	Tom

1) Ορίζουμε δυο παραμέτρους.
2) Στις παραμέτρους αυτές αποθηκεύουμε τις τιμές των δυο στηλών του πίνακα mytable.
3) Θυμίζουμε ότι η εντολή String name=new String(); δημιουργεί το αντικείμενο name που είναι instance της κλάσης String.
4) Η εντολή int m[] = new int[5000]; δημιουργεί τον πίνακα αριθμών 5000 θέσεων που περιέχουν τιμές ακεραίων αριθμών. Όπως δημιουργείται το αντικείμενο πίνακας m[] (με τον τελεστή new) σε όλες τις θέσεις εκχωρείται τιμή 0.
5) Η εντολή String s[] = new String[5000]; δημιουργεί πίνακα συμβολοσειρών 5000 θέσεων που περιέχουν τιμές συμβολοσειρών (String). Όπως δημιουργείται το αντικείμενο πίνακας s[] (με τον τελεστή new) σε όλες τις θέσεις εκχωρείται τιμή NULL.
6) Φορτώνουμε το driver και προετοιμάζουμε τη συμβολοσειρά για σύνδεση.
7) Συνδεόμαστε, στην περίπτωσή μας, με username root και password 1234, στη βάση δεδομένων mydb.
8) Συνδεόμαστε στη βάση δεδομένων, mydb, δημιουργούμε το αντικείμενο τύπου statement και προετοιμάζουμε την SQL string.
9) Εκτελούμε το SQL Statement.
10) Μέχρι στιγμής έχουμε αποθηκεύσει όλον τον πίνακα αποτελεσμάτων στο αντικείμενο rs. Μένει να «διαβάσουμε» τα στοιχεία του rs. Αυτό γίνεται με έναν βρόχο while.
11) Η εντολή while(rs.next()) έχει διπλό ρόλο: Εκτελεί την εντολή rs.next(). Επιπλέον, ελέγχει αν rs.next()=true. Αυτό σημαίνει ότι, αν rs.next()=true, τότε ο βρόχος θα εκτελεστεί. Αν rs.next()=false τότε ο βρόχος δε θα εκτελεστεί και το πρόγραμμα θα προχωρήσει παρακάτω. Η μέθοδος next() της resultSet επιστρέφει true αν υπάρχει επόμενη γραμμή για να διαβάσει. Αν δεν υπάρχει επόμενη γραμμή, δηλαδή έχουμε φτάσει στο τέλος των αποτελεσμάτων, τότε επιστρέφει false. Με τον τρόπο αυτό, αν υπάρχει επόμενη γραμμή, ο βρόχος εκτελείται, και σταματά να εκτελείται όταν οι γραμμές «τελειώσουν». Ο βρόχος κάνει το εξής:

- Διαβάζει, τις στήλες της τρέχουσας γραμμής και τις καταχωρεί στις αντίστοιχες μεταβλητές του προγράμματος (s[j], (m[j],), μέσω της μεθόδου getString() ή getInt().
- Τυπώνει στην οθόνη τις παραπάνω μεταβλητές.

Ζητούμενα

Δημιουργείστε τη σελίδα test.jsp η οποία θα διαβάζει το περιεχόμενο του πίνακα myTable και θα το εμφανίζει.

1. john
2. Ann
3. tom

Δημιουργείστε τη σελίδα test.jsp. Παραθέτουμε τον κώδικα.

test.jsp

```
<%--
Document   : test
Created on : April 31, 2020, 8:24:04 AM
Author :
--%>
// Import the package java.sql - all the classes
<%@page import="java.sql.*" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%
int j=0;
// Define parameters for retrieving database results
int m[] = new int[5000];
String s[] = new String[5000];
// Load the driver
Class.forName("com.mysql.jdbc.Driver");
// Define the connection parameters
String myDatabase = "jdbc:mysql://localhost:3306/personnel?user=root";
// Connect to the database
Connection myConnection = DriverManager.getConnection(myDatabase);
// Create object to execute statements
Statement myStatement = myConnection.createStatement();
// Select everything from the database table
String sqlString = "SELECT * FROM myTable ";
ResultSet rs=myStatement.executeQuery(sqlString);
// Store the results in vectors in order to pass them to the JSP
while(rs.next()){
m[j]=rs.getInt("id");
s[j]=rs.getString("name");
j++;
}
// Close the connection to the database
myStatement.close();
myConnection.close();
// Print
out.println("j="+j+"<br>");
for(int i=0; i<j; i++)
{
out.println(m[i]+"");
out.println(s[i]+"<br>");
}
%>
```

Εισαγωγή γραμμών

Δημιουργείστε τη σελίδα ins.jsp με την οποία εισάγουμε γραμμές στον πίνακα. Παραθέτουμε τον κώδικα. Δείτε ότι εισάγουμε συγκεκριμένες τιμές μέσα στο πρόγραμμά μας, π.χ., 4 Jim

ins.jsp

```
<%@page import="java.sql.*" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
INSERT INTO mytable VALUES(4, 'Jim')
<%
// Load the driver
Class.forName("com.mysql.jdbc.Driver");
// Define the connection parameters
String myDatabase =
"jdbc:mysql://localhost:3306/personnel?user=root&password=1234";
// Connect to the database
Connection myConnection = DriverManager.getConnection(myDatabase);
// Create object to execute statements
Statement myStatement = myConnection.createStatement();
// Select everything from the database table
String sqlString = "INSERT INTO mytable VALUES(4, 'Jim')";
myStatement.executeUpdate(sqlString);
// Close the connection to the database
myStatement.close();
myConnection.close();
%>
```

Στη σελίδα ins.jsp γράψαμε τη δήλωση SQL με πραγματικές τιμές και στην επόμενη σελίδα ins_param.jsp την ξαναγράφουμε με παραμετρικές τιμές. Αντίστοιχα γράφουμε:

```
String sqlString = "INSERT INTO mytable(id) VALUES('"+param+"')";
String sqlString = "INSERT INTO mytable VALUES(4, 'Jim')";
```

Εισαγωγή γραμμών με παραμετρικές τιμές

Δημιουργείστε τη σελίδα ins_param.jsp με την οποία εισάγουμε γραμμές στον πίνακα. Παραθέτουμε τον κώδικα. Δείτε ότι τώρα δεν εισάγουμε συγκεκριμένες τιμές μέσα στο πρόγραμμά μας.

ins_param.jsp

```
<%@page import="java.sql.*" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
INSERT INTO mytable VALUES(4, 'Jim') -- html
<% int param=7; %>
<%
// Load the driver
Class.forName("com.mysql.jdbc.Driver");
// Define the connection parameters
String myDatabase =
"jdbc:mysql://localhost:3306/personnel?user=root&password=1234";
// Connect to the database
Connection myConnection = DriverManager.getConnection(myDatabase);
// Create object to execute statements
Statement myStatement = myConnection.createStatement();
// Select everything from the database table
String sqlString = "INSERT INTO mytable(id) VALUES('"+param+"')";
myStatement.executeUpdate(sqlString);
// Close the connection to the database
```

```
myStatement.close();
myConnection.close();
%>
```

12.4.6 Case Study «Διαχείριση βάσης προσωπικού»

Σας αναθέτουν να σχεδιάσετε και μα υλοποιήσετε εφαρμογή η οποία αφορά τη διαχείριση του προσωπικού εταιρείας. Στην εφαρμογή θα έχουν πρόσβαση μόνο όσοι χρήστες είναι εγγεγραμμένοι και διαθέτουν username –password. Αρχικά θα εμφανίζεται στο χρήστη μια login φόρμα όπου θα δίνεται η δυνατότητα στον χρήστη να συνδεθεί. Αφού συνδεθεί ο χρήστης θα έχει τις παρακάτω δυνατότητες:

1) Προβολή όλων των τμημάτων της εταιρείας μαζί με τα στοιχεία εργαζομένων που ανήκουν σε αυτά.
2) Εισαγωγή ενός νέου εργαζόμενου.
3) Διαγραφή ενός εργαζόμενου
4) Ενημέρωση των στοιχείων των εργαζομένων.

Η βάση δεδομένων ονομάζεται personnel και περιλαμβάνει δύο πίνακες.

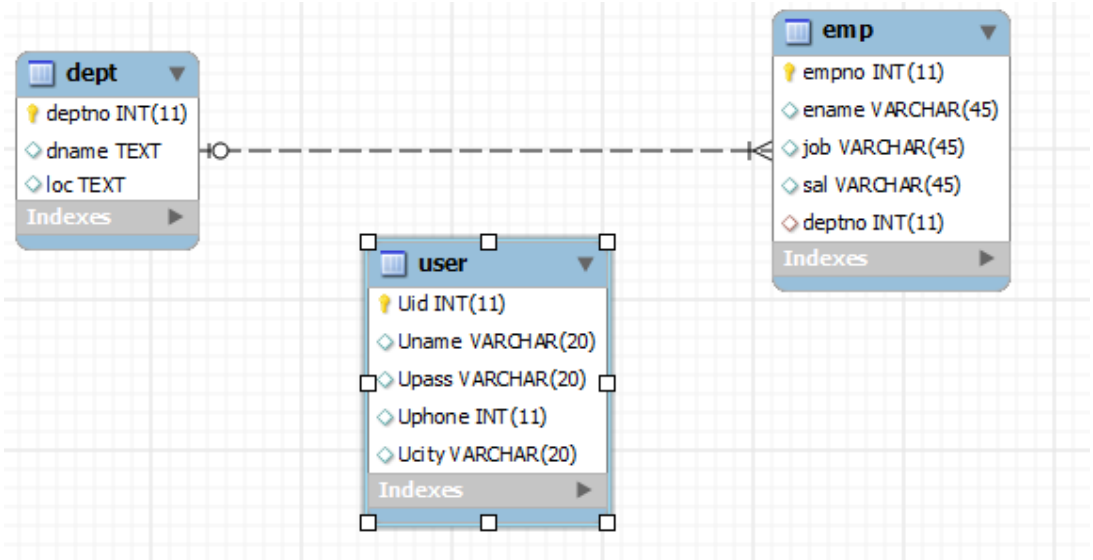
Ο πίνακας Dept(στοιχεία τμημάτων) αποτελείται από τις στήλες:

- DEPTNO κωδικός τμήματος
- DNAME όνομα
- LOC έδρα
- NO_OF_EMPLOYEES αριθμός υπαλλήλων τμήματος

Ο πίνακας Emp (στοιχεία υπαλλήλων) αποτελείται από τις στήλες:

- EMPNO κωδικός υπαλλήλου
- ENAME όνομα
- SAL μισθός
- DEPTNO κωδικός τμήματος υπαλλήλου

Βλέπουμε το ΜΟΣ και τους πίνακες της εφαρμογής.



Κατασκευή βάσης δεδομένων και πινάκων. Εισαγωγή δεδομένων.

```
DROP DATABASE personnel;
CREATE DATABASE personnel;
USE personnel;
CREATE TABLE Dept(DEPTNO INT(2) NOT NULL,
 DNAME VARCHAR(14), LOC VARCHAR(14),
 NO_OF_EMPLOYEES INT(3),
 PRIMARY KEY(DEPTNO));
CREATE TABLE Emp(EMPNO INT(4) NOT NULL AUTO_INCREMENT,
ENAME VARCHAR(10), JOB VARCHAR(9),
SAL 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');
```

Εισάγουμε στοιχεία και στον πίνακα των υπαλλήλων.

```
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO)  VALUES ( 'SMITH', 'CLERK', 800, 20);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO) VALUES ( 'ALLEN', 'SALESMAN', 1600,
30);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO) VALUES ( 'WARD', 'SALESMAN', 1250,
30);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO) VALUES ( 'JONES', 'MANAGER', 2975,
20);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO)  VALUES ( 'MARTIN', 'SALESMAN', 1250,
30);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO) VALUES ( 'BLAKE', 'MANAGER', 2850,
30);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO) VALUES ( 'CLARK', 'MANAGER', 2450,
10);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO) VALUES ( 'SCOTT', 'ANALYST', 3000,
20);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO) VALUES ( 'KING', 'PRESIDENT', 5000,
10);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO) VALUES ( 'TURNER', 'SALESMAN', 1500,
30);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO) VALUES ( 'ADAMS', 'CLERK', 1100, 20);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO) VALUES ( 'JAMES', 'CLERK', 950, 30);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO) VALUES ( 'FORD', 'ANALYST', 3000,
20);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO) VALUES ( 'MILLER', 'CLERK', 1300,
10);
INSERT INTO Emp( ENAME, JOB, SAL,  DEPTNO) VALUES ( 'BATES', 'ANALYST', 1300,
NULL);
```

Για να συμπληρώσουμε τη στήλη no_of_employees του πίνακα dept πρέπει να εκτελέσουμε τη δήλωση UPDATE. Θυμίζουμε ότι είναι προτιμότερο να γράψουμε εναύσματα για να διαχειριστούμε αυτόματα την τιμή της στήλης no_of_employees.

```
UPDATE dept
SET no_of_employees =
  (SELECT COUNT(*)
   FROM emp
   WHERE emp.deptno = dept.deptno);
```

Βλέπουμε τα στοιχεία των δύο πινάκων.

```
Select * from dept;
Select * from  emp;
```

```
mysql> Select * from dept;
+--------+------------+----------+------------------+
| DEPTNO | DNAME      | LOC      | NO_OF_EMPLOYEES  |
+--------+------------+----------+------------------+
|     10 | ACCOUNTING | NEW YORK |                3 |
|     20 | RESEARCH   | DALLAS   |                5 |
|     30 | SALES      | CHICAGO  |                6 |
|     40 | OPERATIONS | BOSTON   |                0 |
+--------+------------+----------+------------------+
4 rows in set (0.00 sec)

mysql> Select * from  emp;
+-------+--------+-----------+---------+---------+
| EMPNO | ENAME  | JOB       | SAL     | DEPTNO  |
+-------+--------+-----------+---------+---------+
|     1 | SMITH  | CLERK     |  800.00 |      20 |
|     2 | ALLEN  | SALESMAN  | 1600.00 |      30 |
|     3 | WARD   | SALESMAN  | 1250.00 |      30 |
|     4 | JONES  | MANAGER   | 2975.00 |      20 |
|     5 | MARTIN | SALESMAN  | 1250.00 |      30 |
|     6 | BLAKE  | MANAGER   | 2850.00 |      30 |
|     7 | CLARK  | MANAGER   | 2450.00 |      10 |
|     8 | SCOTT  | ANALYST   | 3000.00 |      20 |
|     9 | KING   | PRESIDENT | 5000.00 |      10 |
|    10 | TURNER | SALESMAN  | 1500.00 |      30 |
|    11 | ADAMS  | CLERK     | 1100.00 |      20 |
|    12 | JAMES  | CLERK     |  950.00 |      30 |
|    13 | FORD   | ANALYST   | 3000.00 |      20 |
|    14 | MILLER | CLERK     | 1300.00 |      10 |
|    15 | BATES  | ANALYST   | 1300.00 |    NULL |
+-------+--------+-----------+---------+---------+
15 rows in set (0.02 sec)
```

Δημιουργούμε πίνακα χρηστών.

```
CREATE TABLE user(
  uname text,
  upass text,
  Uid int(11),
  Uphone varchar(45),

  Ucity varchar(45));

INSERT INTO `user`

VALUES ('admin','1234',1,NULL,NULL);

Select * from user;
```

12.4.6.1 Χρήση Cursor

Θα δημιουργήσουμε procedure (CursorProc) διαχείρισης cursor, η οποία αρχικά δημιουργεί τον πίνακα infologs που αποτελείται από δύο στήλες, στήλη αύξοντος αριθμού και στήλη μηνύματος (Id, Msg). Στη συνέχεια, ορίζουμε τον cursor:

```
DECLARE  cur_dept CURSOR FOR SELECT  deptno FROM dept;
```

Κάθε γραμμή του cursor έχει τα στοιχεία ενός τμήματος. Η procedure διαβάζει τον cursor γραμμή-γραμμή και αν το τμήμα έχει λιγότερους από 5 υπαλλήλους τότε γράφει στον πίνακα infologs τον κωδικό του τμήματος ως μήνυμα.

Παραθέτουμε τον κώδικα.

procedure CursorProc

```
DELIMITER $$
DROP PROCEDURE IF EXISTS CursorProc$$
CREATE PROCEDURE CursorProc()
BEGIN
DECLARE  no_more_depts, available_employees INT DEFAULT 0;
DECLARE  dept_code VARCHAR(255);
DECLARE  cur_dept CURSOR FOR
 SELECT  deptno FROM dept;
 DECLARE  CONTINUE HANDLER FOR NOT FOUND
 SET  no_more_depts = 1;
 /* for  loggging information */
CREATE  TABLE infologs (
Id int(11) NOT NULL AUTO_INCREMENT,
Msg varchar(255) NOT NULL, PRIMARY KEY (Id));
OPEN  cur_dept;
FETCH  cur_dept  INTO dept_code;
 REPEAT
SELECT  no_of_employees INTO available_employees
FROM  dept
WHERE  deptno = dept_code;
  IF  available_employees < 5 THEN
INSERT  INTO infologs(msg)VALUES  (dept_code);
  END  IF;
  FETCH  cur_dept INTO dept_code;
  UNTIL  no_more_depts = 1
  END REPEAT;
CLOSE  cur_dept;
 SELECT *  FROM infologs;
 DROP TABLE  infologs;
END$$
DELIMITER ;
CALL  CURSORPROC();
```

Παραθέτουμε και το αποτέλεσμα της εκτέλεσης.

```
mysql> CALL  CURSORPROC();
+----+-----+
| Id | Msg |
+----+-----+
|  1 | 10  |
|  2 | 40  |
+----+-----+
2 rows in set (0.20 sec)
```

Τα τμήματα 10 και 40 έχουν λιγότερους από 5 υπλλήλους.

12.4.7 Καθορισμός παραμέτρων και κατασκευή της εφαρμογής

Έστω ότι τα στοιχεία της σύνδεσης με τη βάση δεδομένων είναι:

```
IP: localhost
Port: 3306
Database: personnel
User root
Password χωρίς συνθηματικό.
```

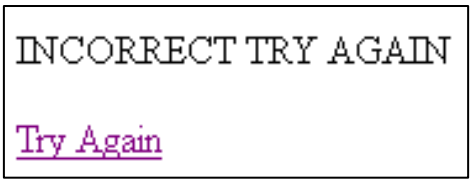
Υπενθυμίζουμε ότι ο χρήστης (βλέπε και πίνακα user) έχει όνομα admin και συνθηματικό (Password) 1234.

12.4.7.1 Διαχείριση Σύνδεσης

Υλοποιείται με χρήση δύο (2) σελίδων: index.jsp, check.jsp

Η σελίδα Index.jsp εμφανίζει φόρμα στην οποία ο χρήστης δίνει τους κωδικούς του.

Αν δοθούν λανθασμένα στοιχεία τότε εμφανίζεται το μήνυμα,



Παραθέτουμε τον κώδικα της σελίδας index.html.

index.html

```
<%--
Document   : index
Created on : April 30, 2018, 2:04:01 PM
Author  :
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>JSP Page</title>
```

```
</head>
<body>
  WELCOME<p></p>
  LOGIN<p></p>
  <p></p>
  <form name="formName" method="post" action="check.jsp" >
    Name: <input type="text" name="y"><p></p>
    Password:<input type="text" name="k"><P></P>
    <input type="submit" value="LOGIN"><p></p>
  </form>
</body>
</html>
```

Παραθέτουμε τον κώδικα της σελίδας check.jsp

check.jsp

```
<%--
Document   : check
Created on : April 30, 2018, 2:12:02 PM
Author :
--%>

<%@page import="java.sql.*" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<%
Boolean found;
String x =request.getParameter("y");
String p = request.getParameter("k");
String  URL;
Class.forName("com.mysql.jdbc.Driver");
String DB = "jdbc:mysql://localhost:3306/personnel?user=root";
Connection myConnection = DriverManager.getConnection(DB);
Statement SMT = myConnection.createStatement();
String sql="SELECT * FROM user WHERE Uname=\'"+x+"\'AND Upass=\'"+p+"\' ";
ResultSet rs=SMT.executeQuery(sql);
found= rs.first();
if (found){
   URL = "home.jsp?p1="+x+"";
 response.sendRedirect(URL);
}
else
 {
out.println("INCORRECT TRY AGAIN");%><P></P>
<a href="index.jsp">Try Again</a><%
  }
   SMT.close();
myConnection.close();
%>
</body>
</html>
```

Σημαντική σημείωση

Έστω ότι τα στοιχεία της σύνδεσης είναι:

```
IP: localhost
Port: 3306
Database: personnel
User= root
Password= 1234
```

Τότε αντί της εντολής

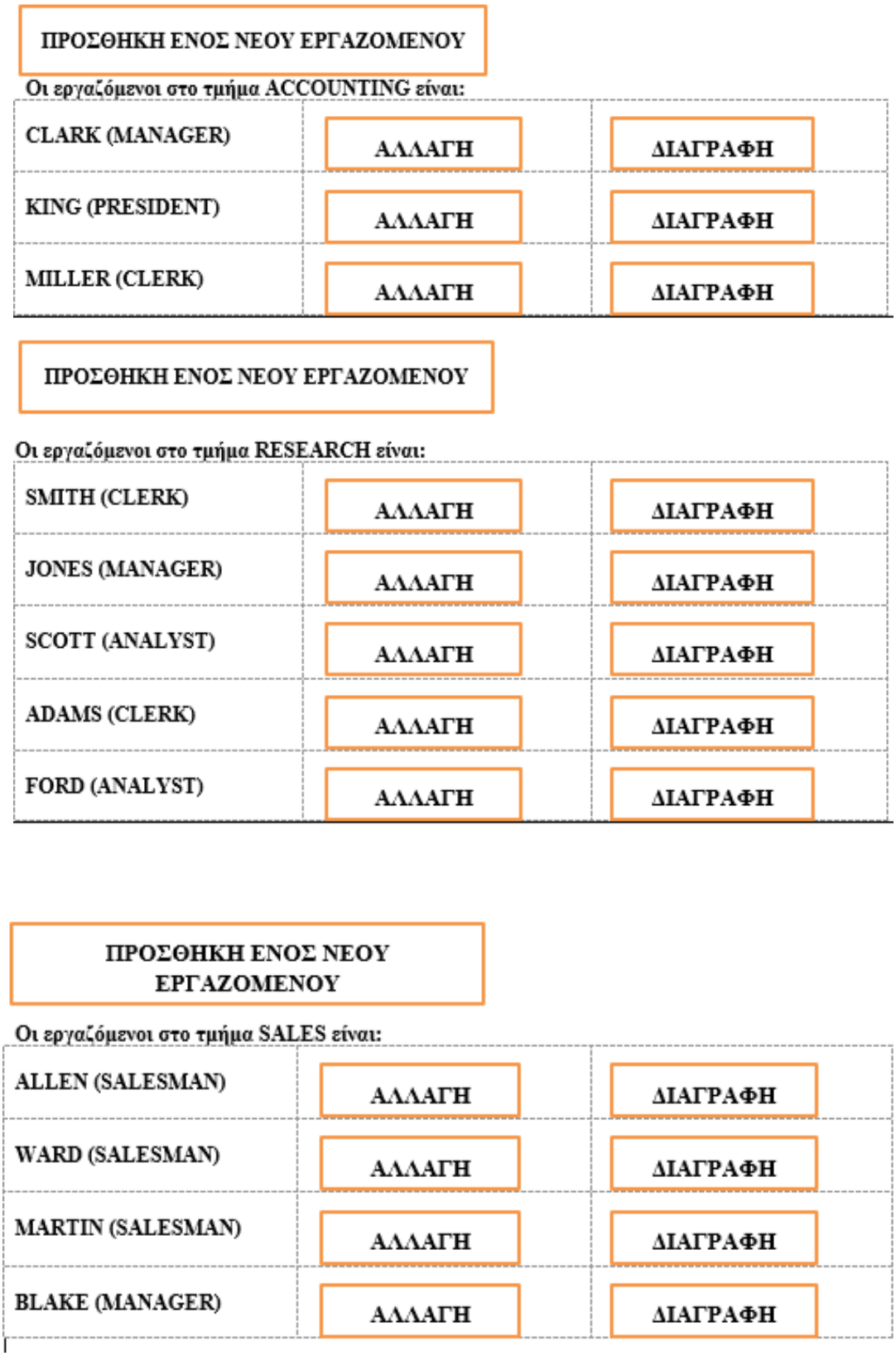
```
String DB = "jdbc:mysql://localhost:3306/personnel?user=root";
```

χρησιμοποιούμε την εντολή:

```
String DB = "jdbc:mysql://localhost:3306/personnel_db?user=root&password=1234";
```

12.4.7.2 Προβολή όλων των εργαζομένων και των τμημάτων τους.

Η βασική σελίδα για όλες τις λειτουργίες της βάσης δεδομένων είναι η σελίδας home.jsp. Η σελίδα παρουσιάζει την προβολή όλων των εργαζομένων ανά τμήμα και υπάρχουν κουμπιά (buttons) για όλες τις λειτουργίες που ενδιαφέρουν: ΠΡΟΣΘΗΚΗ ΕΝΟΣ ΝΕΟΥ ΥΠΑΛΛΗΛΟΥ, ΑΛΛΑΓΗ (ενημέρωση στοιχείων), ΔΙΑΓΡΑΦΗ. Στην εικόνα 12.32 βλέπουμε τη διεπαφή χρήστη που δημιουργείται από τη σελίδα home.jsp.



Εικόνα 12.32 Διεπαφή χρήστη. Προβολή στοιχείων των υπαλλήλων ανά τμήμα και δυνατότητα επιλογής λειτουργιών, εισαγωγής νέου υπαλλήλου, ενημέρωσης στοιχείων κι διαγραφής υπαλλήλου.

Παραθέτουμε τον κώδικα της σελίδας home.jsp.

home.jsp

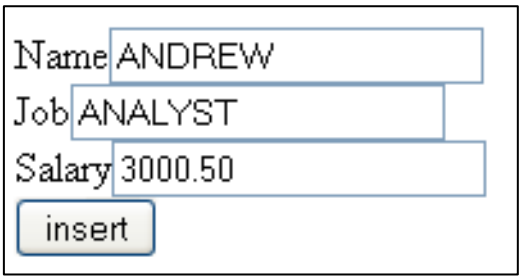
```
<%--
Document   : home
Created on : April 30, 2018, 2:26:58 PM
Author :
--%>
<%@page import="java.sql.*" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<%
Integer empno[]=new Integer[100];
String name[]=new String[100];
String job[]=new String[100];
int i;
Class.forName("com.mysql.jdbc.Driver");
String DB = "jdbc:mysql://localhost:3306/personnel?user=root";
Connection myConnection = DriverManager.getConnection(DB);
Statement SMT = myConnection.createStatement();
Statement SMT1 = myConnection.createStatement();
%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<%
String sql="SELECT * FROM dept ";
ResultSet rs=SMT.executeQuery(sql);
String dname;
int count=0;
while (rs.next())
{
  i=rs.getInt("deptno");
  String sql1="SELECT * FROM emp WHERE deptno='"+i+"' ";
  ResultSet rs1=SMT1.executeQuery(sql1);
  dname=rs.getString("dname");
%>
  <br> <INPUT TYPE="BUTTON" VALUE="ΠΡΟΣΘΗΚΗ ΕΝΟΣ ΝΕΟΥ ΕΡΓΑΖΟΜΕΝΟΥ"
style="font-size:7px"
    ONCLICK="window.location.href='insert.jsp?deptno=<%= i%>'">
<%
  out.println(" <br>Οι εργαζόμενοι στο Τμήμα:"+dname+" είναι οι:");

  while(rs1.next())
  {
    name[count]=rs1.getString("ename");
    job[count]=rs1.getString("job");
    empno[count]=rs1.getInt("empno");
%><br>
<%=  name[count]%>  (<%=  job[count]%>)
  <INPUT TYPE="BUTTON" VALUE="ΑΛΛΑΓΗ" style="font-size:7px"
    ONCLICK="window.location.href='update.jsp?empno=<%= empno[count]%>'">
  <INPUT TYPE="BUTTON" VALUE="ΔΙΑΓΡΑΦΗ " style="font-size:7px"
    ONCLICK="window.location.href='delete.jsp?empno=<%= empno[count]%>'">
 <br>
<%  String x="o";
  }

}
```

```
SMT.close();
SMT1.close();
myConnection.close();
%>
</body>
</html>
```

Εισαγωγή νέου υπαλλήλου

Αν ο χρήστης επιλέξει στη σελίδα home.jsp το κουμπί ΠΡΟΣΘΗΚΗ ΕΝΟΣ ΝΕΟΥ ΥΠΑΛΛΗΛΟΥ τότε καλείται η σελίδα insert.jsp η οποία εμφανίζει μια φόρμα για να πληκτρολογηθούν τα στοιχεία του νέου υπαλλήλου. Βλέπουμε τη φόρμα.

Για παράδειγμα



Ο χρήστης συμπληρώνει τα στοιχεία του νέου υπαλλήλου και πατά το κουμπί insert. Τότε καλείται η σελίδα ins_submit.jsp η οποία γράφει τα στοιχεία στη βάση δεδομένων.

Ενημέρωση στοιχείων υπαλλήλου

(update.jsp, up_submit.jsp)

Αν ο χρήστης επιλέξει στη σελίδα home.jsp το κουμπί ΑΛΛΑΓΗ που υπάρχει δίπλα σε κάποιο υπάλληλο (δηλαδή ζητά ενημέρωση των στοιχείων του υπαλλήλου) τότε καλείται η σελίδα update.jsp η οποία εμφανίζει μια φόρμα με τα στοιχεία του υπαλλήλου για να πληκτρολογηθούν τα νέα στοιχεία του. Για παράδειγμα, έστω ότι έχουμε συνδεθεί και βλέπουμε τη σελίδα home.jsp.

Ενέργεια πρώτη

Βλέπουμε την προβολή όλων των υπαλλήλων ανά τμήμα.

ΠΡΟΣΘΗΚΗ ΕΝΟΣ ΝΕΟΥ ΕΡΓΑΖΟΜΕΝΟΥ

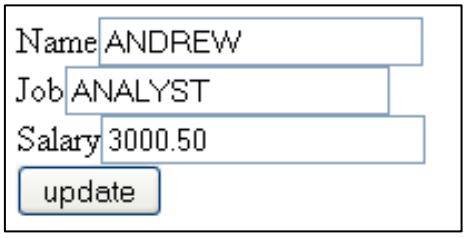
Οι εργαζόμενοι στο τμήμα ACCOUNTING είναι:

CLARK (MANAGER)	**ΑΛΛΑΓΗ**	**ΔΙΑΓΡΑΦΗ**
KING (PRESIDENT)	**ΑΛΛΑΓΗ**	**ΔΙΑΓΡΑΦΗ**
MILLER (CLERK)	**ΑΛΛΑΓΗ**	**ΔΙΑΓΡΑΦΗ**
ANDREW (ANALYST)	**ΑΛΛΑΓΗ**	**ΔΙΑΓΡΑΦΗ**

κ.λπ.

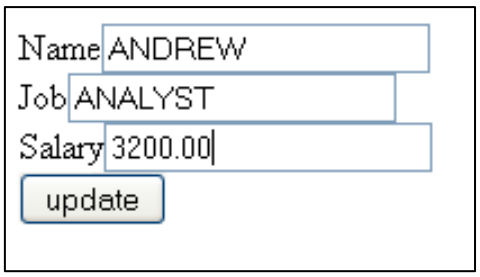
Ενέργεια δεύτερη

Επιλέγουμε το κουμπί ΑΛΛΑΓΗ δίπλα στον υπάλληλο ANDREW. Τότε καλείται η σελίδα update.jsp η οποία εμφανίζει τη φόρμα με τα στοιχεία του υπαλλήλου



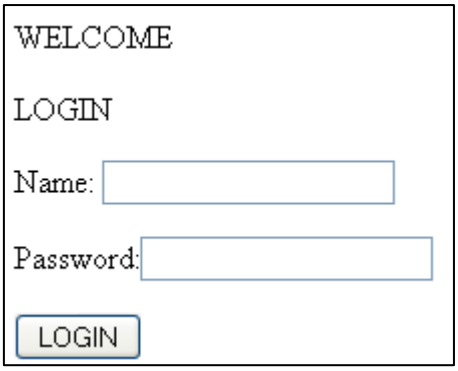
Ενέργεια τρίτη

Αλλάζουμε τα στοιχεία



Ενέργεια τέταρτη

Επιλέγουμε το κουμπί update οπότε καλείται η σελίδα up_submit.jsp η οποία γράφει τα νέα στοιχεία στη βάση δεδομένων και επιστρέφουμε στην αρχική φόρμα.



Διαγραφή ενός υπαλλήλου

Αν ο χρήστης επιλέξει στη σελίδα home.jsp το κουμπί ΔΙΑΓΡΑΦΗ που υπάρχει δίπλα σε κάποιον υπάλληλο τότε καλείται η σελίδα delete.jsp η οποία και τον διαγράφει.

Παρατήρηση

Θα έπρεπε να βλέπουμε όλα τα στοιχεία του υπαλλήλου πριν τον διαγράψουμε. Επιπλέον, θα έπρεπε να υλοποιηθεί και επιβεβαίωση πριν από τη διαγραφή. Γενικά θα μπορούσαμε να βελτιώσουμε σε πολλά σημεία την εφαρμογή με σχετικά απλό τρόπο!

Παραθέτουμε τον κώδικα της σελίδας insert.jsp

insert.jsp

```
<%--
Document   : insert
Created on : April 30, 2020, 2:32:55 PM
Author :
--%>


<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
  <body>
    <form name="insert" method="get" action="ins_submit.jsp">
      Name<input type="text" name="ename"><br>
      Job<input type="text" name="job"><br>
      Salary<input type="text" name="sal"><br>
      <input type="hidden" name="deptno" value="<%=
request.getParameter("deptno") %>">


      <input type="submit" value="insert">
    </form>
  </body>
</html>
```

Παραθέτουμε τον κώδικα της σελίδας ins_submit.jsp

ins_submit.jsp

```
<%--
Document   : ins_submit
Created on : April 30, 2020, 2:34:18 PM
Author :
--%>
<%@page import="java.sql.*"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<%
  Class.forName("com.mysql.jdbc.Driver");
  String DB = "jdbc:mysql://localhost:3306/personnel?user=root";
  Connection myConnection = DriverManager.getConnection(DB);
  Statement SMT = myConnection.createStatement();
  String name=request.getParameter("ename");
  String job=request.getParameter("job");
  String sal=request.getParameter("sal");
  int deptno=Integer.parseInt(request.getParameter("deptno"));
  String sql="insert into emp(ename,job,sal,deptno) values
('"+name+"',"+job+"',"+sal+"',"+deptno+")";
  SMT.executeUpdate(sql);
  String URL = "index.jsp";
  response.sendRedirect(URL);
%>
<%
  SMT.close();
  myConnection.close();
%>
</body>
</html>
```

Παραθέτουμε τον κώδικα της σελίδας update.jsp

update.jsp

```
<%--
Document   : update
Created on : April 30, 2020, 2:35:38 PM
Author :
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<%@page import="java.sql.*"%>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<%
  Class.forName("com.mysql.jdbc.Driver");
```

```
  String DB = "jdbc:mysql://localhost:3306/personnel?user=root";
  Connection myConnection = DriverManager.getConnection(DB);
  Statement SMT = myConnection.createStatement();

  String job="",sal="",ename="";
  int empno=Integer.parseInt(request.getParameter("empno"));
  String sql="select * from emp where empno="+empno;
  ResultSet rs= SMT.executeQuery(sql);
  while(rs.next())

  {
    job=rs.getString("job");
    sal=rs.getString("sal");
    ename=rs.getString("ename");
  }
%>
<form name="insert" method="get" action="up_submit.jsp">
  Name<input type="text" name="ename" value="<%= ename%>"><br>
  Job<input type="text" name="job"value="<%= job%>"><br>
  Salary<input type="text" name="sal" value="<%= sal%>"><br>
  <input type="hidden" name="empno" value="<%= empno%>">
  <input type="submit" value="update">
</form>

</body>
</html>
```

Παραθέτουμε τον κώδικα της σελίδας up_submit.jsp

up_submit.jsp

```
<%--
Document   : up_submit
Created on : April 30, 2020, 2:37:10 PM
Author :
--%>

<%@page import="java.sql.*"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<%
  Class.forName("com.mysql.jdbc.Driver");
  String DB = "jdbc:mysql://localhost:3306/personnel?user=root";
  Connection myConnection = DriverManager.getConnection(DB);
  Statement SMT = myConnection.createStatement();

  String name=request.getParameter("ename");
  String job=request.getParameter("job");
  String sal=request.getParameter("sal");
  int empno=Integer.parseInt(request.getParameter("empno"));

  String sql="update emp set ename =\'"+name+"\' where empno="+empno;
  SMT.executeUpdate(sql);
  sql="update emp set job =\'"+job+"\' where empno="+empno;
```

```
  SMT.executeUpdate(sql);
  sql="update emp set sal =`"+sal+"` where empno="+empno;
  SMT.executeUpdate(sql);
  String URL = "index.jsp";
  response.sendRedirect(URL);
  SMT.close();
  myConnection.close();
%>
</body>
</html>
```

Παραθέτουμε τον κώδικα της σελίδας delete.jsp

delete.jsp

```
<%--
Document   : delete
Created on : April 30, 2020, 2:38:19 PM
Author :
--%>
<%@page import="java.sql.*"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<%
  Class.forName("com.mysql.jdbc.Driver");
  String DB = "jdbc:mysql://localhost:3306/personnel?user=root";
  Connection myConnection = DriverManager.getConnection(DB);
  Statement SMT = myConnection.createStatement();
  int empno=Integer.parseInt(request.getParameter("empno"));
  String sql="delete from emp  where empno="+empno;
  SMT.executeUpdate(sql);
  String URL = "index.jsp";
  response.sendRedirect(URL);
  SMT.close();
  myConnection.close();
%>
</body>
</html>
```