



ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΥΠΟΥΡΓΕΙΟ ΕΣΩΤΕΡΙΚΩΝ, ΑΠΟΚΕΝΤΡΩΣΗΣ & ΗΛΕΚΤΡΟΝΙΚΗΣ ΔΙΑΚΥΒΕΡΝΗΣΗΣ



Εθνικό
Κέντρο
Δημόσιας
Διοίκησης &
Αυτοδιοίκησης

**ΥΠΟΕΡΓΟ: «ΔΗΜΙΟΥΡΓΙΑ ΕΚΠΑΙΔΕΥΤΙΚΟΥ ΥΛΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΩΝ ΓΙΑ ΤΗΝ
ΑΝΑΠΤΥΞΗ ΤΟΥ ΑΝΘΡΩΠΙΝΟΥ ΔΥΝΑΜΙΚΟΥ ΤΗΣ ΔΗΜΟΣΙΑΣ ΔΙΟΙΚΗΣΗΣ ΒΑΣΕΙ
ΣΧΕΔΙΩΝ ΕΚΠΑΙΔΕΥΣΗΣ»**

ΤΙΤΛΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ:

ΕΙΣΑΓΩΓΗ ΣΤΗ ΓΛΩΣΣΑ SQL

ΕΚΠΑΙΔΕΥΤΙΚΟ ΥΛΙΚΟ

Κωδικός εκπαιδευτικού υλικού:

Κωδικός Πιστοποίησης προγράμματος: 392



**ΥΠΟΕΡΓΟ: «ΔΗΜΙΟΥΡΓΙΑ ΕΚΠΑΙΔΕΥΤΙΚΟΥ ΥΛΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΩΝ ΓΙΑ ΤΗΝ
ΑΝΑΠΤΥΞΗ ΤΟΥ ΑΝΘΡΩΠΙΝΟΥ ΔΥΝΑΜΙΚΟΥ ΤΗΣ ΔΗΜΟΣΙΑΣ ΔΙΟΙΚΗΣΗΣ ΒΑΣΕΙ
ΣΧΕΔΙΩΝ ΕΚΠΑΙΔΕΥΣΗΣ»**

ΤΙΤΛΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ: Εισαγωγή στη γλώσσα SQL

ΟΜΑΔΑ ΕΡΓΑΣΙΑΣ

Μέλη Ομάδας

Συντονιστής: Δρ. Μερκούριος Μαργαριτόπουλος, Προϊστάμενος Περιφερειακού Ινστιτούτου Επιμόρφωσης
Θεσσαλονίκης

Συντάκτες/κτριες: Δρ. Εμμανουήλ Ζούλιας, Στέλεχος Τμήματος Εφαρμογών Πληροφορικής και Τεχνικής
Υποστήριξης ΕΚΔΔΑ
Δρ. Ευάγγελος Σακκόπουλος, Στέλεχος ΥΠ.ΕΣ.
Μαρία Τράκα, Στέλεχος Τμήματος Εφαρμογών Πληροφορικής και Τεχνικής
Υποστήριξης ΕΚΔΔΑ

Αξιολογητές: Δρ. Αναστάσιος Σαλής Προϊστάμενος Τμήματος Εφαρμογών Πληροφορικής και Τεχνικής
Υποστήριξης ΕΚΔΔΑ
Τσιμάρας Δημήτριος, Υπεύθυνος Σπουδών και Έρευνας ΕΚΔΔΑ



Περιεχόμενα

1.	Εισαγωγή στις Βάσεις Δεδομένων.....	10
1.1.	Εισαγωγή στις Βάσεις Δεδομένων (ΒΔ) – Ορισμοί	10
1.2.	Πλεονεκτήματα και μειονεκτήματα από τη χρήση Β.Δ.	10
1.3.	Συστήματα Διαχείρισης Βάσης Δεδομένων (ΣΔΒΔ)	11
1.4.	Παραδείγματα Χρήσης ΒΔ	11
2.	Σχεδιασμός Βάσεων Δεδομένων.....	12
2.1.	Εισαγωγή στη Σχεδίαση	12
2.2.	Μοντέλο Οντοτήτων Συσχετίσεων (ΜΟΣ)	14
2.2.1.	Παράδειγμα Βάσης Δεδομένων στη Δημόσια Διοίκηση.....	15
2.2.2.	Διάγραμμα Οντοτήτων-Συσχετίσεων (ΟΣ)	16
2.2.3.	Σύνολα Οντοτήτων.....	17
2.2.4.	Σύνολα Συσχετίσεων.....	19
2.2.5.	Επεκτάσεις του βασικού μοντέλου ΟΣ	20
2.3.	Μοντέλα Δεδομένων	24
2.3.1.	Μοντέλα βασισμένα σε εγγραφές	24
2.3.2.	Μοντέλα βασισμένα σε αντικείμενα	25
2.3.3.	Φυσικά Μοντέλα Δεδομένων.....	25
2.4.	Σχεσιακό Μοντέλο Δεδομένων	25
2.4.1.	Βασικές Έννοιες - Πίνακες, Γραμμές, Στήλες	25
2.4.2.	Κλειδιά Σχέσεων – Πρωτεύον Κλειδί	29
2.4.3.	Περιορισμοί Ακεραιότητας Σχέσεων και Ξένο Κλειδί.....	30
2.4.4.	Μετατροπή του μοντέλου ΟΣ σε Σχεσιακό Μοντέλο	33
2.5.	Η γλώσσα πρότυπο SQL	41
2.5.1.	Γλώσσα Ορισμού Δεδομένων (ΓΟΔ-DDL)	41
2.5.2.	Γλώσσα Χειρισμού Δεδομένων (ΓΧΔ-DML)	41
2.5.3.	Γλώσσα Ελέγχου Δεδομένων (ΓΕΔ-DCL).....	42
2.5.4.	Γλώσσα Ελέγχου Δοσοληψιών	42
3.	Εισαγωγή στο περιβάλλον λειτουργίας της MySQL	43
3.1.	Το Περιβάλλον της MySQL.....	43
3.2.	Επισκόπηση του μοντέλου Πελάτη / Εξυπηρετητή (Client/Server).....	48
3.2.1.	MySQL Server	48
3.2.2.	MySQL Πελάτες & Χρήση αρχείου παραμέτρων.....	49
3.2.3.	MySQL non-Client Utilities	52
3.3.	MySQL Enterprise Monitor	53
3.4.	MySQL Workbench	55
3.5.	MySQL Connectors	56
4.	Ορισμός Δεδομένων.....	59
4.1.	Ιδιότητες ΒΔ (Database Properties)	59
4.2.	Καλές πρακτικές Σχεδιασμού (Good Design Practices).....	59

4.3.	Αναγνωριστικά ΒΔ (Identifiers).....	63
4.4.	Δημιουργία ΒΔ (Creating Databases)	64
4.5.	Τροποποίηση ΒΔ (Altering Databases).....	65
4.6.	Διαγραφή ΒΔ (Dropping Databases).....	65
4.7.	Επισκόπηση τύπων δεδομένων	66
4.8.	Περιγραφή Πινάκων (Describe Table).....	67
4.9.	Δημιουργία Πινάκων (Creating Tables).....	68
4.10.	Επιλογές Στηλών (Column Options).....	69
4.11.	Δημιουργία πινάκων βάσει υπαρχόντων πινάκων (Creating Tables Based on Existing Tables)	70
4.12.	Τροποποίηση Πινάκων (Altering Tables)	70
4.13.	Διαγραφή Πινάκων (Dropping Tables)	72
4.14.	Περιορισμοί Ακεραιότητας (Integrity Constraints) – Πρωτεύοντα κλειδιά	72
4.15.	Δημιουργία Πινάκων με Ξένο Κλειδί	73
4.16.	Ευρετήρια πινάκων (Indexes)	75
5.	Ανάκτηση Δεδομένων – Απλά ερωτήματα.....	78
5.1.	Η εντολή SELECT	78
5.2.	Η συνθήκη φιλτραρίσματος where.....	80
5.3.	Ταξινόμηση – Order by	81
5.4.	Τελεστές (Αριθμητικοί, Λογικοί κλπ)	82
5.5.	Χειρισμός Σφαλμάτων και Προειδοποίησεων / Αντιμετώπιση προβλημάτων.....	83
6.	Γλώσσα Χειρισμού Δεδομένων – Data Manipulation Language (DML)	84
6.1.	Εισαγωγή Δεδομένων.....	85
6.1.1.	Εντολή INSERT	85
6.1.1.1.	INSERT με χρήση του όρου < values>	85
6.1.1.2.	INSERT με χρήση φωλιασμένης ερώτησης	90
6.1.1.3.	INSERT με χρήση του όρου <set>.....	92
6.1.2.	Εντολή REPLACE	93
6.1.2.1.	REPLACE με χρήση του όρου <values>	93
6.1.2.2.	REPLACE με χρήση του όρου <set>	95
6.2.	Ενημέρωση Δεδομένων.....	97
6.2.1.	Εντολή Update	97
6.3.	Διαγραφή Δεδομένων.....	99
6.3.1.	Εντολή DELETE	99
6.3.2.	Εντολή TRUNCATE	102
7.	Συναλλαγές (Transactions)	104
7.1.	Τι είναι η συναλλαγή (Transaction)	104
7.2.	Εντολές συναλλαγών.....	105
7.2.1.	Πρόσθεση SAVEPOINTS στις συναλλαγές	108
7.2.2.	Αυτόματο COMMIT	112
7.3.	Οι Ιδιότητες των συναλλαγών	113
7.4.	Επίπεδα Απομόνωσης – Isolation Levels	114
7.5.	Κλείδωμα πινάκων	118

8.	Συνενώσεις.....	119
8.1.	Καρτεσιανό γινόμενο πινάκων.....	119
8.2.	Συνένωση (JOIN) πινάκων.....	121
8.2.1.	Φυσική Συνένωση πινάκων (Natural Join).....	122
8.2.2.	Εσωτερική Συνένωση πινάκων (Inner Join).....	125
8.2.2.1.	Εσωτερική συνένωση με την χρήση του τελεστή USING.....	128
8.2.2.2.	Εσωτερική συνένωση και επιπλέον συνθήκη.....	130
8.2.3.	Εσωτερική συνένωση σε περισσότερους από δύο πίνακες	132
8.2.4.	Μετονομασία Πινάκων (tables aliases).....	134
8.2.5.	Μετονομασία γνωρισμάτων-στηλών (columns aliases).....	135
8.2.6.	Συνένωση (Self Join).....	136
8.2.7.	Nonequi Join Συνένωση	140
8.2.8.	Εξωτερική Συνένωση (Outer Join).....	140
8.2.9.	Αριστερή εξωτερική συνένωση (LEFT OUTER JOIN).....	142
8.2.10.	Δεξιά εξωτερική συνένωση (RIGHT OUTER JOIN)	144
8.2.11.	Πλήρης εξωτερική συνένωση (FULL OUTER JOIN)	146
9.	Συναρτήσεις	150
9.1.	Συναθροιστικές Συναρτήσεις (Aggregate functions)	150
9.1.1.	Ομαδοποίηση Δεδομένων τελεστής group by	153
9.1.2.	Συναρτήσεις Αλφαριθμητικών (String functions).....	157
9.1.3.	Αριθμητικές συναρτήσεις (Mathematical functions)	162
9.1.4.	Συναρτήσεις ημερομηνίας / ώρας (Date and time functions).....	165
9.1.5.	Άλλες Συναρτήσεις	171
10.	Ανάκτηση Δεδομένων - Σύνθετα ερωτήματα	172
10.1.	Μέλος Συνόλου	172
10.2.	Κενές Τιμές	173
10.3.	Πράξεις Συνόλων	173
10.3.1.	Ο τελεστής Ένωσης.....	174
10.3.2.	Ο τελεστής Τομής.....	176
10.3.3.	Ο τελεστής της Διαφοράς.....	177
11.	Υποερωτήματα – Subqueries	179
11.1.	Τι είναι τα υποερωτήματα (subqueries)	179
11.2.	Κατηγορίες υποερωτημάτων.....	181
11.3.	Single-row Subqueries	182
11.3.1.	Scalar Subqueries	185
11.3.2.	Row Subqueries.....	186
11.4.	Multiple-row Subqueries.....	187
11.4.1.	Με χρήση του τελεστή ALL.....	188
11.4.2.	Με χρήση του τελεστή ANY	189
11.4.3.	Με χρήση των τελεστών IN και NOT IN	190
11.4.4.	Με χρήση των τελεστών EXISTS και NOT EXISTS	191
11.4.5.	Single – Column Subqueries	192

11.4.6.	Correlated Subqueries	193
11.4.7.	Table Subqueries	194
11.5.	Μετατροπή υποερωτημάτων σε ερωτήματα με τη χρήση συνενώσεων (JOIN).....	195
12.	Όψεις (Views).....	198
12.1.	Τι είναι οι Όψεις (Views)	198
12.2.	Κατηγορίες Όψεων.....	199
12.3.	Δημιουργία Όψης.....	199
12.4.	Ανάκτηση δεδομένων από Όψη	202
12.5.	Περιορισμοί όψεων	202
12.6.	Αλγόριθμοι όψεων	203
12.7.	Δημιουργία ανανεούμενης όψης (updatable view)	203
12.8.	Τροποποίηση όψης.....	205
12.9.	Διαγραφή Όψης.....	206
12.10.	Έλεγχος όψης.....	206
12.11.	Μεταδεδομένα όψεων	207
12.12.	Δικαιώματα για δημιουργία όψεων	207
13.	Διαχείριση Βάσης Δεδομένων Mysql και Ασφάλεια (Data Control Language - DCL).....	209
13.1.	Διαχείριση Λογαριασμών Χρηστών.....	209
13.2.	Κατά τη διάρκεια της εγκατάστασης	210
13.3.	Διαχείριση χρηστών, τι ισχύει;.....	212
13.4.	Δημιουργία Χρήστη	213
13.5.	Μετονομασία χρήστη	218
13.6.	Διαγραφή χρήστη	219
13.7.	Αλλαγή κωδικού πρόσβασης (password)	221
13.8.	Δικαιώματα (privileges) χρηστών	223
13.9.	Οι τύποι δικαιωμάτων που υποστηρίζει η MySQL	224
13.10.	Τροποποίηση δικαιωμάτων χρήστη	226
13.10.1.	Grant Privileges	226
13.10.2.	Grant - Database Privileges	227
13.10.3.	Grant - Table Privileges.....	227
13.10.4.	Grant - Column privileges	228
13.10.5.	Global privileges	229
13.11.	Ανάκληση Δικαιωμάτων (Revoking Privileges)	230
13.12.	Περιορισμός δικαιωμάτων πρόσβασης	233
13.13.	Δικαιώματα για διαχείριση χρηστών.....	234
14.	Συντήρηση – Υποστήριξη πινάκων	236
14.1.	Τύποι παρεχόμενων εντολών	236
14.2.	Εντολές SQL για την συντήρηση πινάκων.....	237
14.2.1.	Εντολή ελέγχου πινάκων (CHECK TABLE)	238
14.2.2.	Επιδιόρθωση πινάκα (REPAIR TABLE)	238
14.2.3.	Ανάλυση πινάκα (ANALYZE TABLE).....	239
14.2.4.	Βελτιστοποίηση πινάκα (Optimize table).....	241

14.3.	Εργαλεία πελάτη για την συντήρηση πινάκων	242
14.3.1.	Το εργαλείο mysqlcheck	242
14.3.2.	Το εργαλείο myisamchk	244
14.3.3.	Επιλογές των mysqlcheck και myisamchk	246
14.4.	Διαφορές σε πίνακες τύπου MyISAM και InnoDB	247
14.4.1.	Επιδιόρθωση πινάκων τύπου InnoDB	247
14.4.2.	Επιδιόρθωση πινάκων τύπου MyISAM	249
14.4.3.	Καταστροφή στην Βάση Δεδομένων.....	249
14.5.	Αντίγραφα ασφαλείας	250
14.6.	Δημιουργία Δυαδικών αντιγράφων ασφαλείας για MyISAM	252
14.7.	Δημιουργία Δυαδικών αντιγράφων ασφαλείας για InnoDB	253
14.8.	Παράγοντες δυαδικής φορητότητας	253
14.9.	Αντίγραφα ασφαλείας InnoDB σε λειτουργία.....	254
14.10.	Ανάκτηση βάσης δεδομένων (Restore Database)	255
14.11.	Εξαγωγή Δεδομένων	255
14.12.	Εξαγωγή δεδομένων από γραμμή εντολών	257
14.13.	Εισαγωγή Δεδομένων.....	259
14.14.	Εισαγωγή δεδομένων (import data) από αρχείο	261
14.15.	Εισαγωγή δεδομένων από γραμμή εντολών	263
14.16.	Ανάκτηση μεταδεδομένων του εξυπηρετητή MySQL	264
14.16.1.	Μέθοδος Information_SCHEMA	265
14.16.2.	Ανάκτηση Metadata με χρήση εντολών Show	267
14.16.3.	Διαφορά μεταξύ INFORMATION_SCHEMA Και SHOW	268
14.16.4.	Τα πλεονεκτήματα της SHOW έναντι της INFORMATION_SCHEMA είναι:.....	269
15.	Ευρετήριο Όρων	270
16.	Γλωσσάρι	271
17.	Βιβλιογραφία	273
18.	ΠΑΡΑΡΤΗΜΑ – ΛΙΣΤΑ ΕΙΚΟΝΩΝ	274
19.	ΠΑΡΑΡΤΗΜΑ - ΕΙΚΟΝΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ MySQL 5.6	277
20.	ΜΕΛΕΤΗ ΠΕΡΙΠΤΩΣΗΣ	303
	Σκοπός και στόχοι ενότητας	303
	Εισαγωγή	303
	Σχεσιακό Μοντέλο	304
	Δεδομένα Πινάκων	306
	Ασκήσεις	306
	Ασκηση 1η (Πίνακες – Αλλαγή σχεδίασης)	306
	Ασκηση 2η (Ερωτήματα)	307
	ΑΠΑΝΤΗΣΕΙΣ	310
	Ασκηση 1η (Απαντήσεις)	310
	Ασκηση 2η (Απαντήσεις)	310

1. Εισαγωγή στις Βάσεις Δεδομένων

Μελετώντας το παρόν κεφάλαιο οι επιμορφωνόμενοι θα είναι σε θέση να:

- Περιγράφουν τι είναι βάση δεδομένων και ποια τα πλεονεκτήματα- μειονεκτήματα των βάσεων δεδομένων
- Διακρίνουν τις βάσεις δεδομένων από τα συστήματα διαχείρισης βάσεων δεδομένων

1.1. Εισαγωγή στις Βάσεις Δεδομένων (ΒΔ) – Ορισμοί

Η αποδοτική διαχείριση δεδομένων είναι βασικό χαρακτηριστικό σε πολλές εφαρμογές λογισμικού. Ιστορικά, χρησιμοποιήθηκαν αρχεία για την αποθήκευση δεδομένων με τις εφαρμογές να υλοποιούν τις μεθόδους επεξεργασίας και ανάκτησης. Ωστόσο, καθώς αυξάνεται ο όγκος των δεδομένων και η ευρύτητα των εφαρμογών και ο αριθμός χρηστών που τις χειρίζονται έγινε ξεκάθαρο ότι απαιτείται εξειδικευμένο λογισμικό για τη διαχείριση δεδομένων. Για το λόγο αυτό δημιουργήθηκαν οι Βάσεις Δεδομένων (ΒΔ) ως τρόπος οργάνωσης δεδομένων και τα Συστήματα Διαχείρισης Βάσεων Δεδομένων δηλαδή συστήματα λογισμικού που υλοποιούν όλες τις απαραίτητες εξειδικευμένες λειτουργίες διαχείρισης των ΒΔ.

Μία **Βάση Δεδομένων (ΒΔ)** είναι μία συλλογή από στοιχεία, τα οποία σχετίζονται μεταξύ τους και έχουν καταχωρηθεί και δομηθεί με κατάλληλο τρόπο ώστε να είναι εύκολη η διαχείρισή τους. Οι βάσεις δεδομένων περιέχουν πίνακες, οι οποίοι χρησιμοποιούνται για την αποθήκευση δεδομένων. Οι βάσεις περιέχουν επίσης αντικείμενα διαχείρισης σχεσιακών δεδομένων, όπως όψεις (views), αποθηκευμένες ρουτίνες (stored procedures) και σκανδάλες (triggers). Οι επιμέρους έννοιες παρουσιάζονται σε αντίστοιχες ενότητες.

1.2. Πλεονεκτήματα και μειονεκτήματα από τη χρήση Β.Δ.

- (+) μειώνεται ο όγκος των δεδομένων και η πολυπλοκότητα της πληροφορίας
- (+) ενδεχόμενες αλλαγές γίνονται ευκολότερα και σε λιγότερο χρόνο
- (+) συνδυάζει ευκολότερα αρχεία από διαφορετικές εφαρμογές
- (+) διασφαλίζεται καλύτερα η ακεραιότητα των δεδομένων κατά την ταυτόχρονη εξυπηρέτηση πολλών χρηστών
- (+) δίνει διαφορετικά δικαιώματα σε διαφορετικές ομάδες χρηστών
- (-) δεν δίνει στον προγραμματιστή την ελευθερία να υλοποιήσει τις μεθόδους επεξεργασίας που

αυτός επιθυμεί

1.3. Συστήματα Διαχείρισης Βάσης Δεδομένων (ΣΔΒΔ)

Ένα **Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ)** (Database Management System - DBMS) είναι ένα σύστημα λογισμικού (software), το οποίο επιτρέπει στους χρήστες του και τους προγραμματιστές να πραγματοποιούν μία πληθώρα ενεργειών σε ΒΔ όπως αναζήτηση, εισαγωγή, ενημέρωση, διαγραφή, δημιουργία νέων ΒΔ και όχι μόνο αυτές. Στο παρόν θα μελετήσουμε τις λειτουργίες που μας επιτρέπει να χρησιμοποιήσουμε το ΣΔΒΔ MySQL, κυρίως μέσα από την έκδοση MySQL Community Edition.

1.4. Παραδείγματα Χρήσης ΒΔ

Σήμερα, κάθε προσπάθεια για δημιουργία μεγάλης εφαρμογής συνοδεύεται από την υποστήριξη των Βάσεων Δεδομένων που προσφέρουν γρήγορη προσπέλαση και προστασία στα δεδομένα. Επιπλέον, τα Συστήματα Διαχείρισης Βάσεων Δεδομένων έχουν υιοθετηθεί από οργανισμούς και μεγάλες εταιρίες για την αποθήκευση και διαχείριση των δεδομένων τους. Κάποιοι από τους βασικούς τομείς στους οποίους οι ΒΔ έχουν γίνει αναπόσπαστο κομμάτι είναι οι παρακάτω:

- Δημόσιες Υπηρεσίες με στοιχεία Πολιτών και Νομικών Προσώπων
- Στοιχεία και Δεδομένα Ασφαλιστικών και άλλων Οργανισμών
- Τραπεζικές Συναλλαγές
- Κρατήσεις Θέσεων
- Εταιρικά Δεδομένα

2. Σχεδιασμός Βάσεων Δεδομένων

Μελετώντας το παρόν κεφάλαιο οι επιμορφωνόμενοι θα είναι σε θέση να:

- Σχεδιάζουν βάσεις δεδομένων χρησιμοποιώντας το Μοντέλο Οντοτήτων Συσχετίσεων και το Σχεσιακό Μοντέλο
- Βελτιστοποιούν το σχήμα μίας ΒΔ.
- Περιγράφουν τι είναι η γλώσσα SQL και οι υποκατηγορίες αυτής.
- Διακρίνουν την MySQL από την SQL.

2.1. Εισαγωγή στη Σχεδίαση

Το επόμενο βήμα που ακολουθεί τη ανάλυση των απαιτήσεων για μια εφαρμογή ΒΔ είναι ο εννοιολογικός σχεδιασμός της βάσης δεδομένων. Πρόκειται για τη διαδικασία μοντελοποίησης που στόχο έχει την οργανωμένη και δομημένη μετάβαση σε ένα σχήμα υλοποίησης της ΒΔ. Επιπλέον σκοπός του εννοιολογικού σχεδιασμού είναι η ορθή σημασιολογική αποτύπωση της μορφής και των τύπων των δεδομένων της εφαρμογής.

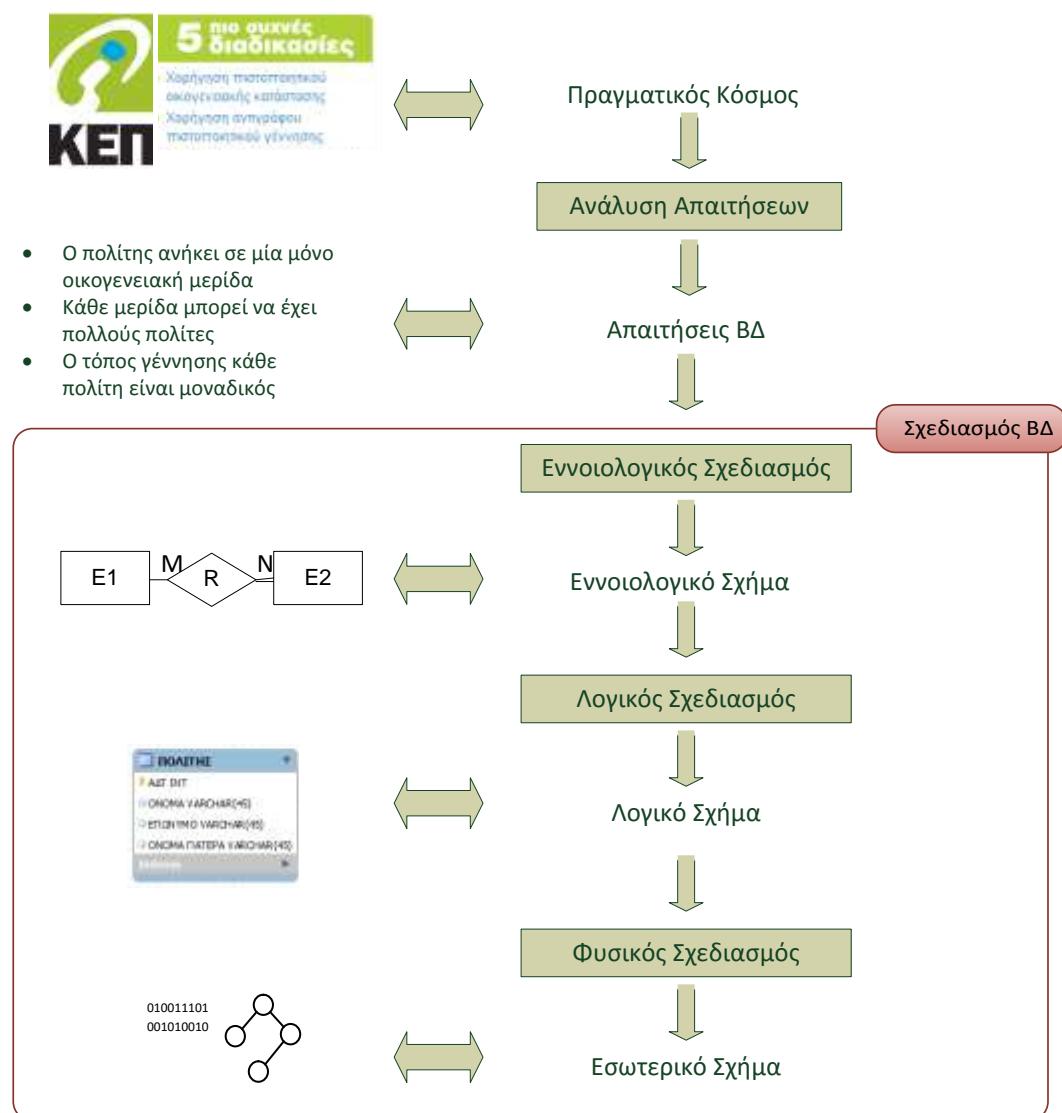
Ο εννοιολογικός σχεδιασμός είναι η φάση της ανάπτυξης μιας ΒΔ που αφορά στην περιληπτική και αφαιρετική αναπαράσταση των επιχειρηματικών εννοιών (κυρίως οντοτήτων, συσχετίσεων και περιορισμών) ενός υποσυνόλου του πραγματικού κόσμου. Η αναπαράσταση αυτή περιλαμβάνει ένα «μοντέλο δεδομένων», δηλαδή τεχνικών που επιτρέπουν να ορίζει κανείς τη μορφή των δεδομένων που θα χειρίζεται και τις λειτουργίες χειρισμού.

Υπάρχουν αρκετά μοντέλα δεδομένων τα οποία μπορούμε να τα κατηγοριοποιήσουμε σε εννοιολογικά (ή υψηλού επιπέδου), λογικά (ή παραστατικά) και σε φυσικά (ή χαμηλού επιπέδου). Τα εννοιολογικά μοντέλα επιτρέπουν να περιγράφουμε με οργανωμένο και καθορισμένο τρόπο τα δεδομένα της εφαρμογής χωρίς τεχνικές λεπτομέρειες. Τα λογικά μοντέλα παρέχουν περισσότερη εξειδίκευση και περιγράφουν έννοιες αντιληπτές από τους τελικούς χρήστες, ενώ παράλληλα παρέχουν μια σχετική καλή αναπαράσταση της αποθήκευσης των δεδομένων στον υπολογιστή. Τα χαμηλού επιπέδου μοντέλα απευθύνονται σε προγραμματιστές καθώς δίνουν όλες τις τεχνικές λεπτομέρειες δόμησης τους στον υπολογιστή, δηλαδή τύπους δεδομένων αλλά και τρόπους αποθήκευσής τους.

Στα φυσικά μοντέλα κανείς συναντά έννοιες όπως εγγραφές, δομές δεδομένων και μηχανισμούς δεικτοδότησης. Τα λογικά μοντέλα περιλαμβάνουν μεταξύ άλλων το σχεσιακό μοντέλο το οποίο έχει ευρέως υιοθετηθεί τα τελευταία χρόνια και βασίζεται στην οργάνωση των δεδομένων σε σύνολα πλειάδων που ονομάζονται σχέσεις και το οποίο θα

χρησιμοποιήσουμε στη συνέχεια. Τέλος, στα εννοιολογικά μοντέλα περιλαμβάνονται υψηλού επιπέδου έννοιες όπως οντότητες, αντικείμενα, συσχετίσεις, περιορισμούς μεταξύ αντικειμένων, κ.λ.π. Ένα τέτοιο μοντέλο είναι και το μοντέλο Οντοτήτων-Συσχετίσεων (Entity-Relationship Model) το οποίο επίσης θα δούμε στη συνέχεια.

Το αποτέλεσμα για κάθενα από τις παραπάνω φάσεις σχεδίασης για τα τρία παραπάνω επίπεδα μοντέλων ονομάζεται αντίστοιχα εννοιολογικό, λογικό και φυσικό σχήμα. Η παρακάτω εικόνα απεικονίζει τα βήματα της σχεδίασης:



Εικόνα 1 Οι φάσεις σχεδιασμού βάσεων δεδομένων

Στην παραπάνω εικόνα παρουσιάζεται η πορεία του σχεδιασμού από υψηλότερα επίπεδα αφαίρεσης σε ολοένα και μεγαλύτερα επίπεδα τεχνικής λεπτομέρειες καθώς ξεκινάμε από τον εννοιολογικό και προχωράμε στον λογικό και φυσικό σχεδιασμό. Τα πλεονεκτήματα από έναν οργανωμένο σχεδιασμό προοδευτικής προσέγγισης όπως ο παραπάνω περιλαμβάνουν:

- η από πάνω προς τα κάτω προσέγγιση βελτιστοποιεί τη σχεδιαστική ακρίβεια ώστε να επιτευχθεί τελικώς το επιδιωκόμενο αποτέλεσμα, διευκολύνοντας την τεχνική εξειδίκευση χωρίς να χάνεται πληροφόρηση περί των απαιτήσεων,
- το εννοιολογικό σχήμα συνδυάζει ικανοποιητική αφαίρεση με αρκετό φορμαλισμό (απαραίτητο στοιχείο για το λογικό σχεδιασμό), που διευκολύνει την επικοινωνία και διαμόρφωση συναντίληψης με το χρήστη για τη μορφή και τη δομή των δεδομένων, όπως προκύπτουν από τις απαιτήσεις του,
- ο δομημένος τρόπος σχεδιασμού εξασφαλίζει μικρότερο κόστος συντήρησης της παραγόμενης βάσης δεδομένων,
- ο πολύ-επίπεδος σχεδιασμός ευνοεί την λογική ανεξαρτησία δεδομένων (data independence), δηλαδή τη δυνατότητα να πραγματοποιούνται αλλαγές στα διαφορετικά επίπεδα σχήματος χωρίς να επηρεάζονται οι επιχειρηματικοί κανόνες που απεικονίζει η εννοιολογική μοντελοποίηση και άρα οι εφαρμογές που χρησιμοποιούν τη ΒΔ. Κατά συνέπεια επιτρέπει επεκτάσεις ή βελτιώσεις στην απόδοση.

2.2.Μοντέλο Οντοτήτων Συσχετίσεων (ΜΟΣ)

Κατά τη διαδικασία σχεδίασης και υλοποίησης μιας Βάσης Δεδομένων, πολλές φορές υπάρχει απόκλιση στον τρόπο σκέψης των σχεδιαστών και προγραμματιστών σε σχέση με τους «πελάτες» που χρειάζονται μία Βάση Δεδομένων δηλαδή τους δυνητικούς χρήστες της. Για να αντιμετωπιστούν οι διαφορετικές απόψεις και προσεγγίσεις έχουν αναπτυχθεί μοντέλα, που παρέχουν έννοιες για να περιγράψουν τη δομή μιας Βάσης Δεδομένων. Τα μοντέλα διευκολύνουν την επικοινωνία και αυξάνουν τις πιθανότητες για ένα επιτυχημένο αποτέλεσμα χωρίς εννοιολογικά και τελικά λειτουργικά λάθη και παραλείψεις.

Στην παρούσα ενότητα θα παρουσιαστεί το μοντέλο οντοτήτων συσχετίσεων (ΜΟΣ). Στο ΜΟΣ θεωρούμε πως ο πραγματικός κόσμος αποτελείται από οντότητες με χαρακτηριστικά και μεταξύ των οντοτήτων αυτών διακρίνονται συσχετίσεις. Αναπτύχθηκε με σκοπό να διευκολύνει τον ορισμό των σχήματος που αναπαριστά τη συνολική λογική δομή, κατά το σχεδιασμό μιας Βάσης Δεδομένων. Το ΜΟΣ έχει απλά δομικά συστατικά και για το λόγο αυτό έχει γίνει εξαιρετικά διαδεδομένο, με πολλά εργαλεία σχεδίασης Βάσεων Δεδομένων να στηρίζονται σε αυτό.

2.2.1. Παράδειγμα Βάσης Δεδομένων στη Δημόσια Διοίκηση

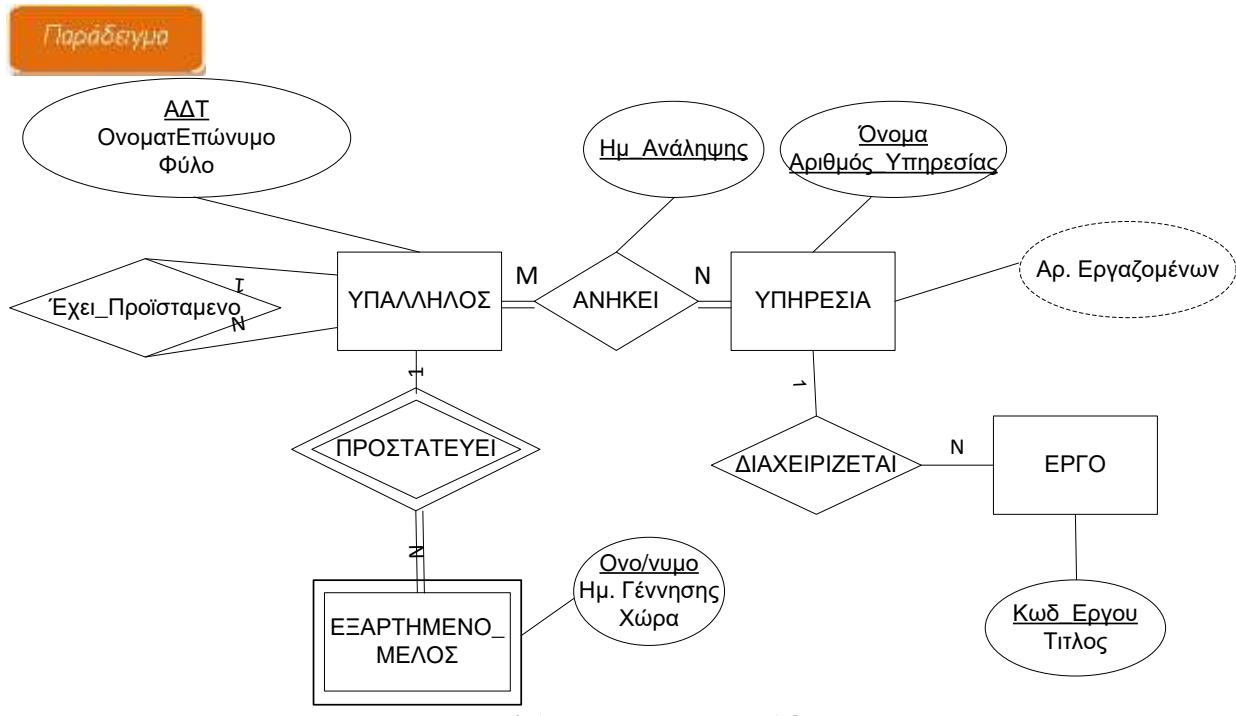
Στο παρόν περιγράφουμε ένα παράδειγμα με όνομα ΥΠΟΥΡΓΕΙΟ που θα χρησιμοποιηθεί για να παρουσιαστούν οι έννοιες των ΜΟΣ αλλά και η σχεδίαση μιας Βάσης Δεδομένων. Παρουσιάζουμε μια σύντομη περιγραφή των απαιτήσεων για τη Βάση Δεδομένων. Στη συνέχεια, χτίζουμε το εννοιολογικό σχήμα της ΒΔ παρουσιάζοντας τις αντίστοιχες έννοιες του μοντέλου.

Η ΒΔ ΥΠΟΥΡΓΕΙΟ περιλαμβάνει τους υπαλλήλους μιας κεντρικής υπηρεσίας Υπουργείου, τα στοιχεία των υπηρεσιών της, και τα έργα της κάθε υπηρεσίας. Θα μπορούσε να πει κανείς ότι μια απλή προσέγγιση των απαιτήσεων στην περίπτωση ενός υπουργείου για τα πολύ βασικά αυτά στοιχεία θα περιγραφόταν ως εξής:

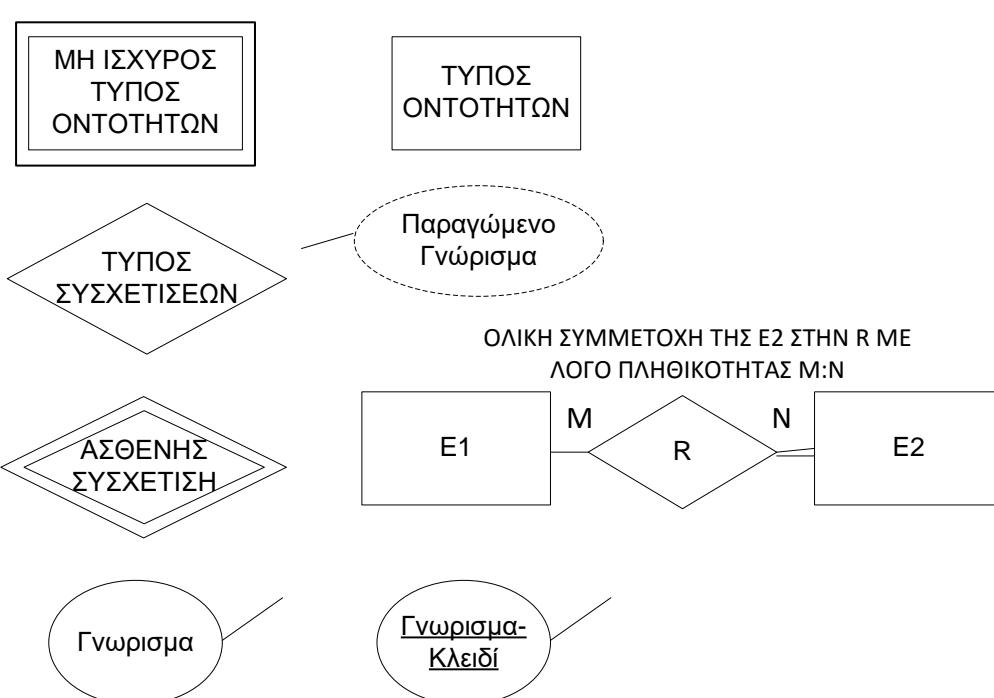
- Το Υπουργείο είναι οργανωμένο σε υπηρεσίες (Γεν. Διευθύνσεις, Διευθύνσεις, Τμήματα, Γραφεία). Για κάθε υπηρεσία υπάρχει μοναδικό όνομα, ένας μοναδικός αριθμός και τα έργα που εκτελεί και διαχειρίζεται (υποθέσεις). Κρατάμε επίσης, πότε κάθε υπάλληλος άρχισε να απασχολείται στην υπηρεσία.
- Κάθε έργο έχει μοναδικό κωδικό (εν είδη αριθμού φακέλου έργου) και τίτλο, ενώ τον διαχειρίζεται μόνο μία υπηρεσία κάθε φορά.
- Για τον υπάλληλο αποθηκεύουμε το ονοματεπώνυμό του, το ΑΔΤ, και το φύλο του. Καταχωρούμε επίσης σε ποια υπηρεσία ανήκει και που έχει παράλληλα καθήκοντα.
- Επιπρόσθετα για κάθε εργαζόμενο κρατάμε τα εξαρτημένα μέλη του όπου για κάθε μέλος έχουμε το ονοματεπώνυμο, την ημ. Γέννησης και τον τόπο/χώρα γέννησης.

Η εικόνα 2 δίνει το ΔΟΣ της παραπάνω Βάσης Δεδομένων.

2.2.2. Διάγραμμα Οντοτήτων-Συσχετίσεων (ΟΣ)



Στην ακόλουθη εικόνα περιλαμβάνονται συνοπτικά οι συμβολισμοί των διαγραμμάτων ΟΣ:



Εικόνα 3 Συμβολισμοί ΔΟΣ (συνοπτικά)

Οι συμβολισμοί χρησιμοποιούνται στις ακόλουθες ενότητες.

2.2.3. Σύνολα Οντοτήτων

Σε ένα ΔΟΣ το κεντρικό στοιχείο που αναπαριστούμε είναι η οντότητα. Η οντότητα αποτελεί ένα μαθηματικά διακριτό στοιχείο του πραγματικού κόσμου με αυτοτέλεια. Μια οντότητα μπορεί να είναι ένα αντικείμενο με φυσική υπόσταση όπως π.χ. ένας πολίτης, ένας υπάλληλος, ένα έγγραφο κ.α. ή χωρίς φυσική υπόσταση όπως π.χ. ένα υπουργείο, ένα τμήμα, ένας λογαριασμός τράπεζας, μια εταιρεία κ.α. Κάθε οντότητα χαρακτηρίζεται από το σύνολο των ιδιαίτερων χαρακτηριστικών της που τα ονομάζουμε γνωρίσματα ή κατηγορήματα ή κάποιες φορές και ιδιότητες. Στην περίπτωση μιας οντότητας Υπάλληλος θα έχει αριθμό ταυτότητας, όνομα, επώνυμο, φύλο, ημερομηνία γέννησης, ημερομηνία πρόσληψης, κ.α. γνωρίσματα. Κάθε γνώρισμα λαμβάνει τιμές από ένα πεδίο ορισμού το οποίο ορίζει και τον τύπο δεδομένων του κατηγορήματος.

Η οντότητα αναφέρεται σημασιολογικά σε ένα σύνολο ομοειδών αντικειμένων/στοιχείων τα οποία ονομάζουμε στιγμιότυπα (instances). Για παράδειγμα, ένα στιγμιότυπο της οντότητας υπάλληλος είναι το στοιχείο <ΑΡ123456, ΓΕΩΡΓΙΟΣ, ΙΩΑΝΝΟΥ, Α, 03-02-1972, ...>. Στη συνέχεια, θα χρησιμοποιούμε τον όρο οντότητα αναφερόμενοι είτε σε ένα στιγμιότυπο, είτε στο σύνολο των στοιχείων (κλάση), όπου αυτό είναι απολύτως σαφές από τα συμφραζόμενα.

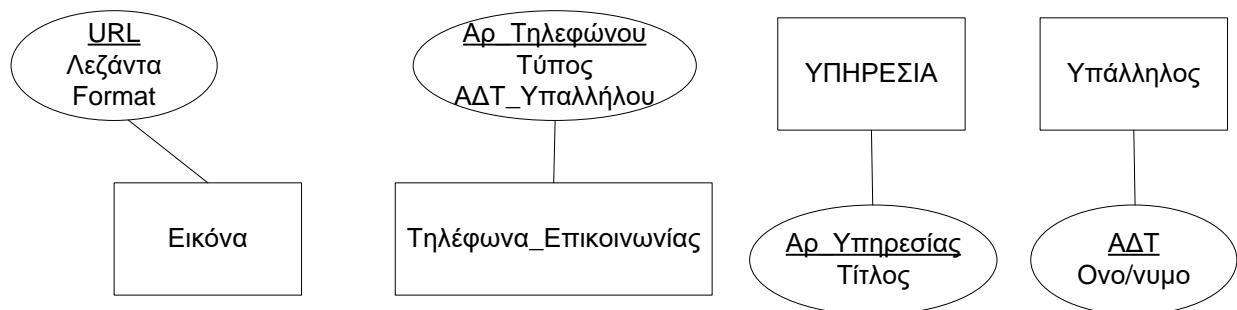
Τα γνωρίσματα λαμβάνουν τιμές από το πεδίο ορισμού τους όπως αναφέραμε. Στην περίπτωση που ένα γνώρισμα δεν έχει τιμή, στις Βάσεις Δεδομένων χρησιμοποιείται η ειδική τιμή NULL για να δηλωθεί αυτή η κατάσταση του γνωρίσματος. Για να εξηγήσουμε την απουσία τιμής υπάρχουν οι ακόλουθες περιπτώσεις στις οποίες σημασιολογικά αντιστοιχείται η τιμή NULL:

- Δεν υπάρχει δυνατή τιμή (not applicable) για τη συγκεκριμένη οντότητα στο εν λόγω γνώρισμα— π.χ. στην περίπτωση του γνωρίσματος ΚΛΙΜΑΚΙΟ της οντότητας ΥΠΑΛΛΗΛΟΣ, για κάποιο στιγμιότυπο επισκέπτη υπαλλήλου τεχνικής ομάδας από την ΕΕ,
- Δε γνωρίζουμε την τιμή που θα πρέπει να περιληφθεί αλλά υπάρχει η τιμή,
- Γνωρίζουμε την τιμή αλλά δεν έχει συμπληρωθεί ακόμη στη ΒΔ λόγω καθυστέρησης ενημέρωσής της.

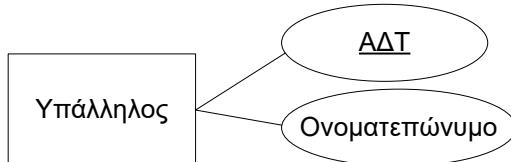
Θα αναλύσουμε την περίπτωση της έννοιας του κλειδιού παρακάτω, ωστόσο ένας αρχικός ορισμός ενός κλειδιού είναι το υποσύνολο των γνωρισμάτων μιας οντότητας που χαρακτηρίζουν μονοσήμαντα κάθε ένα στιγμιότυπο. Ένα σύνολο γνωρισμάτων για να είναι κλειδί θα πρέπει:

- Να μην μπορεί να λάβει την τιμή NULL ώστε να προσδιορίζει μοναδικά κάθε στιγμιότυπο όπως αναφέρει ο επόμενος περιορισμός του ορισμού ενός κλειδιού που περιλαμβάνει ότι,
- δυο διαφορετικά στιγμιότυπα της ίδιας οντότητας δε μπορεί να έχουν ίδια τιμή στο κλειδί τους.

Κύριο ή πρωτεύον κλειδί ορίζουμε το κλειδί εκείνο που ο σχεδιαστής της βάσης επιλέγει να χρησιμοποιήσει σε ένα συγκεκριμένο εννοιολογικό σχήμα. Η επιλογή ανάμεσα σε πολλά διαθέσιμα κλειδιά γίνεται συνήθως με βάση το μικρότερο δυνατό αριθμό γνωρισμάτων του κλειδιού.



Εναλλακτικά Τοποθετούνται τα γνωρίσματα σε χωριστή έλλειψη



Εικόνα 4 Αναπαράσταση οντοτήτων σε ένα ΔΟΣ

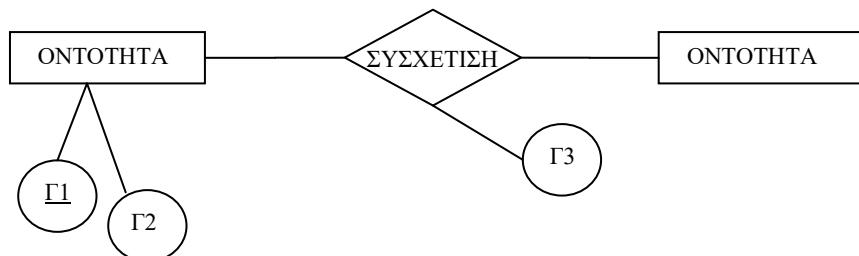
Παραγόμενα γνωρίσματα είναι τα γνωρίσματα που μπορούν να υπολογισθούν από άλλα γνωρίσματα π.χ. ο αριθμός υπαλλήλων μια υπηρεσίας υπολογίζεται με την καταμέτρηση των ΑΔΤ των υπαλλήλων στην υπηρεσία (με διακεκομμένες γραμμές).

Υπάρχουν επίσης σύνθετα γνωρίσματα τα οποία μπορούν να αποσυντεθούν σε επί μέρους γνωρίσματα π.χ. Ονοματεπώνυμο -> Όνομα, Επώνυμο κλπ.

Τέλος μια οντότητα επιτρέπει ως μία τιμή σε κάθε γνώρισμά της. Για το λόγο αυτό υπάρχουν επίσης τα πλειότιμα γνωρίσματα που επιτρέπουν περισσότερες τιμές, π.χ. τα τηλέφωνα ενός υπαλλήλου (σταθερό, γραμματείας, κινητό, fax κλπ). Τα πλειότιμα γνωρίσματα σημειώνονται με διπλή γραμμή έλλειψης σε ένα ΔΟΣ.

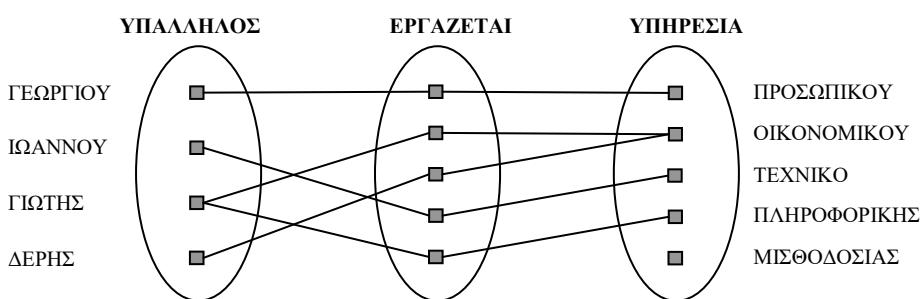
2.2.4. Σύνολα Συσχετίσεων

Μια συσχέτιση είναι μια λογική σύνδεση μεταξύ δύο ή περισσότερων οντοτήτων την οποία επιθυμούμε να αποθηκεύσουμε στη βάση δεδομένων. Σημειώστε ότι τη σύνδεση των οντοτήτων δεν την αποκαλούμε «σχέση» αλλά συσχέτιση. Πρόκειται για τη μοντελοποίηση μιας αλληλεξάρτησης ανάμεσα σε οντότητες η οποία επιθυμούμε να καταγράφεται στη ΒΔ μας. Ο βαθμός μιας συσχέτισης εκφράζει τον αριθμό των οντοτήτων που συμμετέχουν σε αυτή. Στην ακόλουθη εικόνα παρουσιάζεται μία δυαδική συσχέτιση. Μια συσχέτιση μπορεί να έχει γνωρίσματα όπως στην εικόνα παρακάτω είναι το Γ3. Για παράδειγμα η συσχέτιση ΕΡΓΑΖΕΤΑΙ μεταξύ των οντοτήτων ΥΠΑΛΛΗΛΟΣ και ΥΠΗΡΕΣΙΑ μπορεί να έχει κατηγόρημα την ημερομηνία ανάληψης σε συγκεκριμένη υπηρεσία.



Εικόνα 5 Αναπαράσταση οντοτήτων και συσχετίσεων ΔΟΣ

Κατ' αντιστοιχία με τις οντότητες, μια συσχέτιση αποτελεί ένα σύνολο από στιγμιότυπα (ζευγάρια ή n-άδες) στη γενική περίπτωση μεγαλύτερου βαθμού συσχέτισης, μεταξύ των στιγμιοτύπων των οντοτήτων που συσχετίζονται. Για παράδειγμα τα ζευγάρια <ΙΩΑΝΝΟΥ, ΜΙΣΘΟΔΟΣΙΑΣ>, <ΓΕΩΡΓΙΟΥ, ΠΡΟΣΩΠΙΚΟΥ> αποτελούν στιγμιότυπα της συσχέτισης ΕΡΓΑΖΕΤΑΙ μεταξύ των οντοτήτων ΥΠΑΛΛΗΛΟΣ και ΥΠΗΡΕΣΙΑ. Αυτό μπορεί να φανεί καλύτερα διαγραμματικά ως εξής:



Εικόνα 6 Σχηματική απεικόνιση συσχέτισης

Μέσω του παραπάνω σχηματικού θα παρουσιάσουμε και την έννοια του λόγου πληθικότητας (cardinality ratio), που αφορά στον αριθμό των δυνητικών συσχετίσεων ενός στιγμιοτύπου με

στιγμιότυπα της άλλης οντότητας. Υπάρχουν πολλά μοντέλα απεικόνισης της πληθικότητας, ωστόσο το πιο γνωστό περιορίζεται στην αφαιρετική απεικόνιση για τιμές «ένα» (1) και «πολλά» (N). Έτσι οι δυνατοί συνδυασμοί του λόγου πληθικότητας μιας δυαδικής συσχέτισης είναι (διαβάζονται αντίστοιχα):

- 1-1 (ένα προς ένα)
- 1-N (ένα προς πολλά) (διαβάζεται και αντίστροφα N-1 (πολλά προς ένα))
- M-N (πολλά προς πολλά)

Επισημαίνουμε πως η πληθικότητα αφορά το δυνητικό αριθμό πιθανών συσχετίσεων, δηλαδή τι είναι δυνατό να περιληφθεί σε μία συσχέτιση. Δεν προσδιορίζεται απαραιτήτως κατά συνέπεια από τα δεδομένα που περιλαμβάνονται από την τρέχουσα κατάσταση μιας βάσης δεδομένων. Η πληθικότητα προκύπτει από την ανάλυση των απαιτήσεων και τις επιχειρηματικές απαιτήσεις και λογική της εφαρμογής και όχι απλώς από τα έως τώρα δεδομένα της ΒΔ – ειδικά όταν η ΒΔ έχει αρχικά ή αραιά μόνο δεδομένα. Για να μπορεί κανείς να προσδιορίσει το λόγο πληθικότητας μιας συσχέτισης R μεταξύ των οντοτήτων A και B είναι:

- 1) να εξετάσει ένα στιγμιότυπο της οντότητας A με πόσα στιγμιότυπα του B είναι δυνατό να συνδέεται και εφόσον αυτά είναι πολλά, τότε ...-N, αλλιώς ...-1) και
- 2) να εξετάσει και το αντίστροφο, δηλαδή ένα στιγμιότυπο της οντότητας B με πόσα στιγμιότυπα του A είναι δυνατό να συνδέεται και εφόσον αυτά είναι πολλά, τότε M-..., αλλιώς 1...).

Η σύνθεση των δύο παραπάνω ελέγχων δίνει το τελικό αποτέλεσμα π.χ M-N ή 1-N ή 1-1

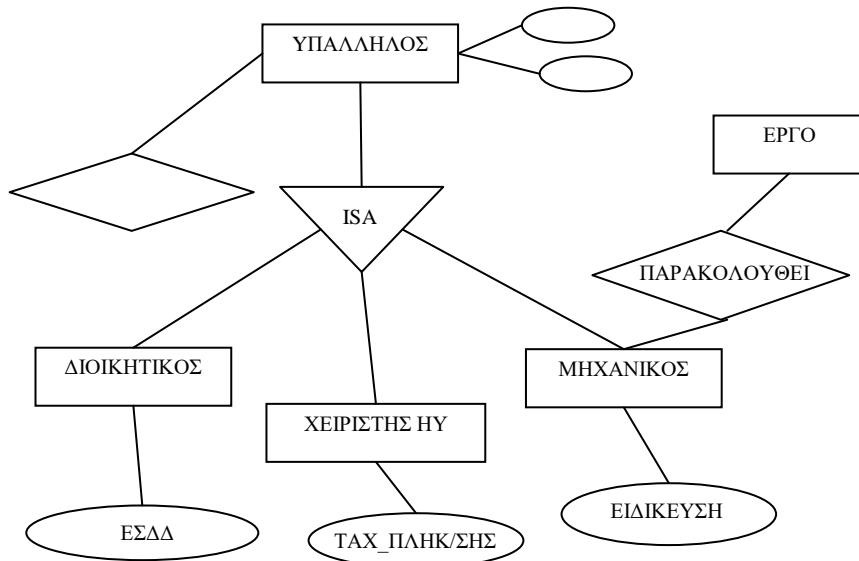
Το κλειδί μιας συσχέτισης καθορίζεται με βάση το λόγο πληθικότητας επιλέγοντας κατάλληλα μία από τις παρακάτω επιλογές:

- 1-1: το κλειδί μιας εκ των 2 οντοτήτων είναι το κλειδί της συσχέτισης
- 1-N: το κλειδί της «N» οντότητας είναι το κλειδί της συσχέτισης
- M-N: η ένωση των κλειδιών των 2 οντοτήτων, δηλαδή το σύνολο των γνωρισμάτων και των δύο κλειδιών είναι το κλειδί της συσχέτισης

2.2.5. Επεκτάσεις του βασικού μοντέλου ΟΣ

Το μοντέλο Οντοτήτων Συσχετίσεων έχει επεκταθεί με το πέρασμα το χρόνων ώστε να καλύπτει τις ανάγκες νέων εφαρμογών. Μία από τις κύριες επεκτάσεις του βασικού μοντέλου αποτελούν οι έννοιες της γενίκευσης και εξειδίκευσης οι οποίες καταρχήν ορίστηκαν ως επεκτάσεις μέσω υποκλάσεων. Στη συνέχεια θα παρουσιάσουμε τις έννοιες αυτές αναλυτικότερα.

Υπάρχουν περιπτώσεις όπου χρειάζεται κανείς να διακρίνει υπο-κλάσεις μιας οντότητας ομαδοποιημένες με βάση τα ξεχωριστά τους γνωρίσματα. Για παράδειγμα, ας χρησιμοποιήσουμε την οντότητα ΥΠΑΛΛΗΛΟΣ στη βάση δεδομένων του προσωπικού ενός Υπουργείου.



Εικόνα 7 Παράδειγμα IS-A

Μπορούμε να διακρίνουμε διαφορετικές ειδικές περιπτώσεις γνωρισμάτων που απαιτούνται ανάλογα με το ρόλο ενός υπαλλήλου, όπως π.χ. ΔΙΟΙΚΗΤΙΚΟΣ, ΜΗΧΑΝΙΚΟΣ, ΧΕΙΡΙΣΤΗΣ HY, κ.λ.π. Στη γενική περίπτωση, η διάκριση αυτή εκφράζει τη διαφοροποίηση στα γνωρίσματα μιας οντότητας (υπερκλάσης) που επιβάλλεται από τις ανάγκες της εφαρμογής. Με τον τρόπο αυτό δημιουργούνται υποσύνολα των στιγμιοτύπων της οντότητας με βάση τα γνωρίσματα που τους χαρακτηρίζουν. Στο συγκεκριμένο παράδειγμα, κάθε διοικητικός, χειριστής HY ή μηχανικός είναι προφανώς και υπάλληλος του υπουργείου. Δεν πρόκειται για χωριστές οντότητες αλλά για ένα σύνολο επιπλέον πρόσθετων γνωρισμάτων που αντιστοιχούν σε κάποιο στιγμότυπο της υπερκλάσης και του προσδίδουν με τον τρόπο αυτό επιπλέον χαρακτηριστικά. Στην περίπτωση της παραπάνω εικόνας:

- Ο διοικητικός (υπάλληλος) μπορεί να έχει πτυχίο από την ΕΣΔΔ, κάτι που δε συμβαίνει με άλλους υπαλλήλους.
- Ένας χειριστής HY έχει ως χαρακτηριστικό γνώρισμα την ταχύτητα πληκτρολόγησης.
- Ένας μηχανικός έχει μια συγκεκριμένη εξειδίκευση. Ταυτόχρονα, μόνο για τους μηχανικούς, ένας μηχανικός παρακολουθεί κάποιο έργο του υπουργείου.

Όπως βλέπουμε και στο σχήμα παραπάνω δημιουργείται ενός είδους ιεραρχία η οποία περιλαμβάνει μια υπερκλάση και κάποιες υποκλασεις – παραγόμενες κλάσεις. Οι υποκλασεις

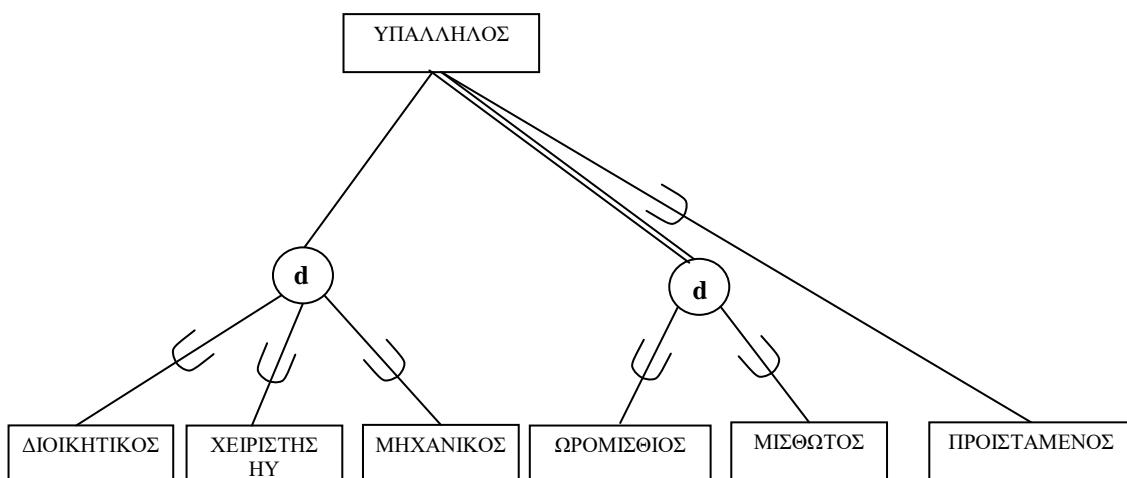
αυτές λέγονται ιεραρχίες IS-A. Οι ιεραρχίες αυτές συμβολίζονται με ένα ανεστραμμένο τρίγωνο με την αντίστοιχη ετικέτα.

Καθώς οι υποκλάσεις είναι παραγόμενες διακρίνετε την ιδιότητα της κληρονομικότητας. Η υπερκλάση κληροδοτεί, υποχρεωτικά, τα γνωρίσματα σε κάθε υποκλάση η οποία την επεκτείνει με επιπλέον γνωρίσματα. Δεν είναι δυνατό επομένως μια υποκλάση να μην περιλαμβάνει τα γνωρίσματα της υπερκλάσης της.

Επιπλέον, μια παρατήρηση στο σημείο αυτό είναι ότι κάθε υποκλάση επίσης κληρονομεί και όλες τις συσχετίσεις στις οποίες συμμετέχει η υπερκλάση. Για το λόγο αυτό παρατηρούμε στο παραπάνω σχήμα ότι δεν χρειάζεται να συνδέονται οι υποκλάσεις με τις συσχετίσεις που κληρονομούν από την υπερκλάση.

Οι υποκλάσεις μπορούν να επεκταθούν ακόμη περισσότερο, ώστε να επιτρέπουν αναλυτικότερη μοντελοποίηση επιπλέον δομικών στοιχείων πάντοτε φορμαλιστικών. Πρόκειται για την εξειδίκευση και την αντίστροφη διαδικασίας της, τη γενίκευση.

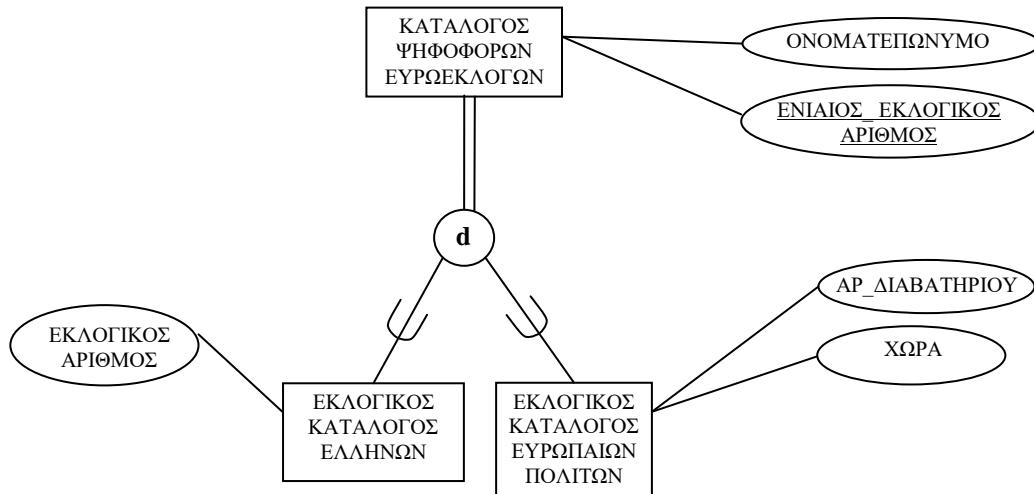
Στο παρακάτω σχήμα παρουσιάζεται ένα παράδειγμα τριών εξειδικεύσεων της οντότητας ΥΠΑΛΛΗΛΟΣ. Στην περίπτωση αυτή, οι υποκλάσεις κάθε εξειδίκευσης, συνδέονται με απλή γραμμή με ένα κύκλο, όταν είναι δύο ή περισσότερες (ο κύκλος περιλαμβάνει είτε το γράμμα d είτε το γράμμα o για τις περιπτώσεις που θα παρουσιαστούν στη συνέχεια). Ο κύκλος στη συνέχεια συνδέεται με απλή ή διπλή γραμμή με την υπερκλάση. Οι γραμμές κάθε υποκλάσης αποτυπώνουν το μαθηματικό σύμβολο του γνησίου υποσυνόλου (\subseteq) στην κατεύθυνση που δηλώνει τη σχέση υποκλάσης. Τέλος στην περίπτωση μιας μόνο υποκλάσης παραλείπουμε τον κύκλο, συνδέοντας την υποκλάση απευθείας με την υπερκλάση (περίπτωση προϊστάμενος).



Εικόνα 8 Παράδειγμα ειδίκευσης

Στην αντίστροφη διαδικασία της ειδίκευσης μπορεί να θεωρηθεί ότι δημιουργούμε μια υπερκλάση όπου προσδιορίζονται τα κοινά χαρακτηριστικά και κατά συνέπεια γενικεύουμε

τους αρχικούς τύπους οντοτήτων. Στο παρακάτω σχήμα, παρουσιάζεται ένα υποθετικό παράδειγμα για την περίπτωση γενίκευσης του καταλόγου ψηφοφόρων Ευρωεκλογών στον οποίο εντάσσονται τα κοινά γνωρίσματα του εκλογικού καταλόγου Ελλήνων και του εκλογικού καταλόγου Πολιτών της ΕΕ. Επισημαίνουμε ότι όπως στην παρακάτω εικόνα, το ΔΟΣ της προηγούμενης εικόνας περιλαμβάνει γνωρίσματα τα οποία για λόγους οικονομίας χώρου έχουν αφαιρεθεί. Σε κάθε περίπτωση πάντως η εξειδίκευση έχει να κάνει με την προσθήκη επιπλέον γνωρισμάτων (υποχρεωτικά!) στα χαρακτηριστικά γνωρίσματα της αρχικής οντότητας/κλάσης.



Εικόνα 9 Παράδειγμα γενίκευσης

Για να ολοκληρώσουμε την περιγραφή της εξειδίκευσης και γενίκευσης, προσδιορίζουμε ότι είναι δυνατό να παρουσιάζονται περιορισμοί ως προς τη συμμετοχή ή την κάλυψη των στιγμιοτύπων της υπερκλάσης στις υποκλάσεις της:

- περιορισμός κάλυψης ‘ο’ ή ‘d’: Στην περίπτωση αυτή ένα στιγμιότυπο μπορεί να συμμετέχει σε πάνω από μία υποκλάσεις. Αν αυτό συμβαίνει (π.χ. ένα ATOMO μπορεί να είναι ΦΟΙΤΗΤΗΣ και ΕΡΓΑΖΟΜΕΝΟΣ στο ίδιο πανεπιστήμιο κ.λ.π.) τότε ο κύκλος της ειδίκευσης/γενίκευσης περιέχει το χαρακτήρα ‘ο’ (από το αγγλικό overlap – επικάλυψη). Στην αντίθετη περίπτωση ο κύκλος περιέχει το χαρακτήρα ‘d’ (από το αγγλικό disjoint – ξεχωρίζω, αποχωρίζω).
- περιορισμός συμμετοχής (μονή ή διπλή γραμμή σύνδεσης με υπερκλάση): η έννοια αυτή είναι γνωστή από την παρουσίαση των συσχετίσεων για την ολική συμμετοχή. Στην περίπτωση που όλα τα στιγμιότητα μιας υπερκλάσης πρέπει να εντάσσονται σε μια υποκλάση τότε η υπερκλάση συνδέεται με τον κύκλο με διπλή γραμμή (**Error! Reference source not found.**), σε αντίθετη περίπτωση (**Error! Reference source not found.**) με μονή οπότε και έχουμε μόνο μερική συμμετοχή στις υποκλάσεις.

Προσοχή χρειάζεται στη διαχείριση των δεδομένων – στιγμιοτύπων όταν εφαρμόζονται εξειδικεύσεις και γενικεύσεις κατά τη διαδικασία εισαγωγής/διαγραφής στιγμιοτύπων για να τηρηθεί η ακεραιότητα της ΒΔ και των δεδομένων της. Στις περιπτώσεις όπου υλοποιούνται υποκλάσεις σημειώνομε ότι:

- η εισαγωγή στιγμιοτύπου σε μια υπερκλάση μιας ολικής ειδίκευσης απαιτεί την εισαγωγή ενός αντίστοιχου στιγμιοτύπου σε μία τουλάχιστο υποκλάση (ή ακριβώς μία υποκλάση αν η ειδίκευση είναι disjoint).
- η διαγραφή ενός στιγμιοτύπου μιας υπερκλάσης απαιτεί τη διαγραφή όλων των σχετικών στιγμιοτύπων των υποκλάσεων που η συγκεκριμένη οντότητα-υπερκλάση συμμετέχει

2.3.Μοντέλα Δεδομένων

Για την περιγραφή του σχήματος μιας βάσης δεδομένων, σε ένα επίπεδο αρκετά υψηλό, όπου δεν περιέχονται οι λεπτομέρειες υλοποίησης, χρησιμοποιούνται τα Μοντέλα Δεδομένων (ΜΔ). Ένα μοντέλο δεδομένων εστιάζει στις οντότητες που ενδιαφέρουν κάθε εφαρμογή και στις μεταξύ τους σχέσεις, δημιουργώντας έτσι μία αναπαράσταση του πραγματικού κόσμου. Ένα μοντέλο δεδομένων πρέπει να σχεδιαστεί με προσοχή ώστε να μην έχει παραλήψεις και λάθη, τα οποία θα οδηγήσουν σε προβλήματα στα ίδια τα δεδομένα αλλά και τις λειτουργίες επεξεργασίας τους και θα προκαλέσει την ανάγκη για αλλαγές του ίδιου του σχήματος και κατά συνέπεια και τμημάτων της βάσης δεδομένων. Τα μοντέλα δεδομένων ανήκουν σε μία από τις κατηγορίες: α) Μοντέλα βασισμένα σε εγγραφές β) Μοντέλα βασισμένα σε αντικείμενα και γ) Φυσικά Μοντέλα Δεδομένων. Λεπτομέρειες για το καθένα παρουσιάζονται στις επόμενες παραγράφους.

2.3.1. Μοντέλα βασισμένα σε εγγραφές

Στην κατηγορία αυτή ανήκει το *Σχεσιακό Μοντέλο* (Relational Model), το *Ιεραρχικό Μοντέλο* (Hierarchical Model) και το *Δικτυακό Μοντέλο* (Network Model).

2.3.2. Μοντέλα βασισμένα σε αντικείμενα

Τα πιο διαδεδομένα μοντέλα που ανήκουν σε αυτή την κατηγορία είναι το *Μοντέλο Οντοτήτων-Συσχετίσεων* (Entity-Relationship Model), το *Αντικειμενοστρεφές Μοντέλο* (Object-Oriented Model), το *Συναρτησιακό Μοντέλο* (Functional Model) και το *Εννοιολογικό Μοντέλο* (Semantic Model).

2.3.3. Φυσικά Μοντέλα Δεδομένων

Σε αυτή την κατηγορία τα πιο γνωστά μοντέλα είναι το *Μοντέλο Ενοποίησης* (Unifying Model) και το *Μοντέλο Πλαισίου-Μνήμης* (Frame-Memory Model).

2.4. Σχεσιακό Μοντέλο Δεδομένων

2.4.1. Βασικές Έννοιες - Πίνακες, Γραμμές, Στήλες

Το σχεσιακό μοντέλο δεδομένων θεωρεί ότι ένα σύνολο από σχέσεις ορίζουν μία Βάση Δεδομένων. Μια σχέση συνήθως αναπαρίσταται ως ένα πίνακας (table) τιμών. Κάθε γραμμή του πίνακα αντιπροσωπεύει ένα σύνολο τιμών δεδομένων που σχετίζονται. Έχοντας διαβάσει για το Μοντέλο Οντοτήτων Συσχετίσεων στις προηγούμενες ενότητες μπορούμε να πούμε ότι μια γραμμή ενός πίνακα αντιπροσωπεύει μια οντότητα ή μια συσχέτιση (ένα στιγμιότυπό τους). Το όνομα του πίνακα αλλά και των στηλών του θα πρέπει να είναι περιγραφικά και αντιπροσωπευτικά ώστε να μπορεί με ευκολία να ερμηνεύεται η σημασία των τιμών σε κάθε γραμμή του πίνακα.

Τυπικά στο σχεσιακό μοντέλο μια γραμμή λέγεται πλειάδα, ο τίτλος της στήλης λέγεται γνώρισμα (attribute) και ολόκληρος ο πίνακας λέγεται σχέση (relation).

Ένα σχήμα σχέσης (relation schema) αποτελείται από το όνομα της σχέσης και μια λίστα από γνωρίσματα. Ο βαθμός μιας σχέσης είναι το πλήθος η των γνωρισμάτων του σχήματός της.

Παράδειγμα σχήματος είναι:

ΥΠΑΛΛΗΛΟΣ (ΑΔΤ, Ονοματεπώνυμο, Φύλο, Ημερομηνία Γέννησης)

Για αυτό το σχήμα σχέσης το όνομα της σχέσης είναι ΥΠΑΛΛΗΛΟΣ και έχει 4 γνωρίσματα.

Μία σχέση σ του σχήματος σχέσης $\Sigma(\Gamma_1, \Gamma_2, \dots, \Gamma_n)$ είναι το σύνολο από n -πλειάδες, δηλαδή μια διατεταγμένη λίστα από η τιμές $\pi\lambda = \langle \tau_1, \tau_2, \dots, \tau_n \rangle$. Ένα στιγμιότυπο $\sigma(\Sigma)$ μιας σχέσης ονομάζεται και relation state.

Κάθε πλειάδα επομένως παριστάνει μια συγκεκριμένη οντότητα. Δείχνουμε

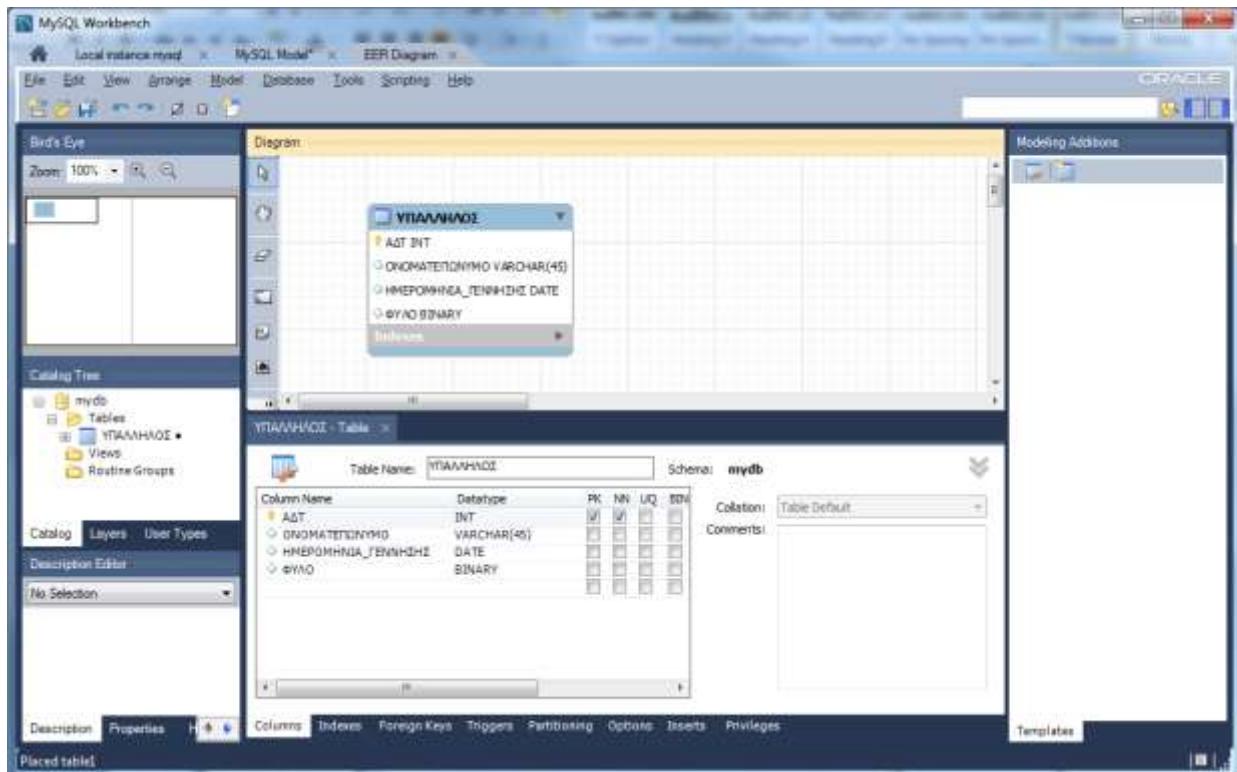
Όνομα Σχέσης

Γνωρίσματα

ΥΠΑΛΛΗΛΟΣ	ΑΔΤ	ΟΝΟΜΑΤΕΠΩΝΥΜΟ	ΦΥΛΟ	ΗΜ. ΓΕΝΝΗΣΗΣ
Πλειάδες→	AB132455	ΠΙΕΤΡΟΣ ΙΩΑΝΝΟΥ	A	02-02-1968
	BA134231	ΜΑΡΙΑ ΡΕΛΛΗ	Θ	03-11-1973
	X123452	ΜΑΡΙΟΣ ΖΗΚΑΣ	A	03-09-1969

Σχήμα 1 Τα γνωρίσματα και οι πλειάδες της σχέσης ΥΠΑΛΛΗΛΟΣ

Ακολούθως δείχνουμε τον πίνακα/σχέση ΥΠΑΛΛΗΛΟΣ στο διάγραμμα του MySQL WorkBench (EER Diagram).



Εικόνα 10 Περιβάλλον σχεδίασης στο MySQL WorkBench

Ας δούμε στη συνέχεια με ποιον τρόπο μπορούμε να δημιουργήσουμε έναν πίνακα με το σχεδιαστικό εργαλείο σχεσιακών μοντέλων του EER Diagram.

Αφού δημιουργήσουμε ένα νέο διάγραμμα επιλέγουμε το εικονίδιο πίνακα στην κάθετη εργαλειοθήκη. Πρόκειται για το εικονίδιο που απεικονίζει ένα παραλληλόγραμμο με ένα πλέγμα. Με το κλικ στο εικονίδιο, ο δείκτης του ποντικού μετατρέπεται σε δείκτη πίνακα. Επιπλέον με την επιλογή του εργαλείου πινάκων, αλλάζουν και τα περιεχόμενα της εργαλειοθήκης που εμφανίζεται κάτω από την κύρια μπάρα μενού οριζόντια στο πάνω μέρος του τρέχοντος παραθύρου. Η νέα οριζόντια εργαλειοθήκη περιλαμβάνει λίστα με τα σχήματα, της μηχανές αποθήκευσης των βάσεων δεδομένων, λίστα των collation καθώς και επιλογή χρώματος. (για τα collation θα αναφερθούμε σε επόμενη ενότητα αναλυτικά). Για να

δημιουργήσετε έναν πίνακα κάνετε κλικ οπουδήποτε μέσα στον καμβά του διαγράμματος. Αυτό δημιουργεί έναν πίνακα με τίτλο table1 ως προεπιλεγμένο όνομα. Σημειώνουμε ότι σε ένα πίνακα στο διάγραμμα EER diagram το πρωτεύον κλειδί παρουσιάζεται με ένα εικονίδιο κλειδιού και τα πεδία ευρετηρίου σημαίνονται με χρωματιστό εικονίδιο για να ξεχωρίζουν από τα υπόλοιπα πεδία-γνωρίσματα. Με δεξί κλικ πάνω στον πίνακα εμφανίζεται βοηθητικό μενού επιλογών ως εξής:

Cut 'table_name' - Αποκοπή του Πίνακα

Copy 'table_name' – Αντιγραφή του Πίνακα

Edit Table... – Επεξεργασία του Πίνακα

Edit in New Window... – Επεξεργασία του πίνακα σε νέο παράθυρο

Copy SQL to Clipboard - Αντιγραφή της SQL για το CREATE TABLE του πίνακα

Copy Insert to Clipboard – Αντιγραφή της SQL για INSERT δεδομένων στον πίνακα

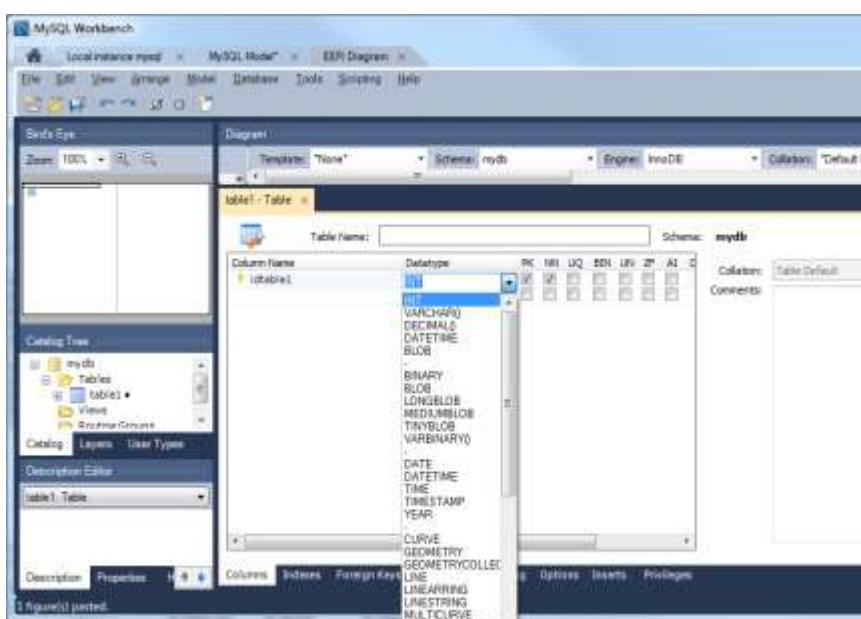
Delete 'table_name' – Διαγραφή του πίνακα

Με την επιλογή της επεξεργασίας του πίνακα είναι δυνατό να γίνει μορφοποίηση όλων των επιλογών ενός πίνακα. Δίνονται δυνατότητες σε διαφορετικές καρτέλες – tabs για ενέργειες επεξεργασίας των στηλών, των ευρετηρίων, των ξένων κλειδιών, των σκανδάλων, των αποθηκευτικών μέσων-partitioning, επιλογών, της εισαγωγής δεδομένων και δικαιωμάτων (βλ. καρτέλες στο κάτω μέρος του παραπάνω σχήματος).

Αναλυτικά στοιχεία για όλες τις σχετικές επιλογές δίνονται στο

<https://dev.mysql.com/doc/workbench/en/wb-table-editor.html>

Στη συνέχεια παρουσιάζονται στοιχεία της καρτέλας στήλες (**columns tab**).



Εικόνα 11 Περιβάλλον σχεδίασης πινάκων Columns Tab στο MySQL WorkBench

Δίνεται δυνατότητα δήλωσης του ονόματος του γνωρίσματος, ο τύπος του γνωρίσματος – επιλέγοντας από τη διαθέσιμη λίστα, όπως στο παραπάνω σχήμα, προεπιλεγμένη τιμή (default) καθώς και μια λίστα από επιλογές:

- PK: PRIMARY KEY
- NN: NOT NULL
- UQ: UNIQUE INDEX
- BIN: BINARY
- UN: UNSIGNED
- ZF: ZEROFILL
- AI: AUTO_INCREMENT

Αν επιθυμείτε να ορίσετε ένα σύνθετο πρωτεύον κλειδί – δηλαδή ένα κλειδί με πολλαπλά γνωρίσματα – επιλέγετε τις πολλαπλές στήλες που επιθυμείτε να απαρτίζουν το κλειδί και δηλώνετε την επιλογή PK στη συνέχεια για να ολοκληρώσετε τον καθορισμό του πρωτεύοντος κλειδιού. Στην περίπτωση χρήσης της αυτόματης αύξησης της τιμής κλειδιού (AI: AUTO_INCREMENT) υλοποιείται ένα κλειδί με αριθμητική τιμή η οποία αυξάνεται αυτόματα κατά ένα (1) κάθε φορά που εισάγεται μία εγγραφή στον πίνακα που ορίζεται το κλειδί αυτό. Με τον τρόπο αυτό το εν λόγω κλειδί είναι ένα πρωτεύον κλειδί με εγγυημένη μοναδική τιμή εκ σχεδιασμού. Η τιμή AI στο κλειδί επιλύει με απλότητα την ανάθεση ενός πρωτεύοντος κλειδιού και επιπλέον έχει το πλεονέκτημα ο προγραμματιστής ή διαχειριστής των δεδομένων της ΒΔ να μην χρειάζεται να αναζητά ποια είναι η επόμενη διαθέσιμη μοναδική τιμή για να την αναθέσει στο πρωτεύον κλειδί. Στη MySQL υπάρχει συνάρτηση που δίνει πρόσβαση στην τελευταία τιμή του AUTO_INCREMENT πεδίου (LAST_INSERT_ID()). Αυτό επιτρέπει να γνωρίζει κανείς την τιμή του πρωτεύοντος κλειδιού που έλαβε μία νέα εγγραφή που εισάχθηκε μόλις ώστε να χρησιμοποιηθεί για επιπλέον στοιχεία που τυχόν προστίθενται σε άλλους πίνακες με αναφορά στη συγκεκριμένη εγγραφή.

Θα συζητήσουμε αναλυτικά για τη χρησιμότητα και τη λογική πίσω από τα κλειδιά σχέσεων στην ακόλουθη ενότητα.

2.4.2. Κλειδιά Σχέσεων – Πρωτεύον Κλειδί

Στο παράδειγμα των παραπάνω σχημάτων παρουσιάζονται σχέσεις οι οποίες είναι οι μοναδικές στη Βάση Δεδομένων. Τις περισσότερες φορές, μία σχεσιακή ΒΔ περιλαμβάνει πολλές σχέσεις και τα δεδομένα που περιλαμβάνουν συσχετίζονται με κάποιους τρόπους. Συνήθως, λόγω

επιχειρηματικών απαιτήσεων αλλά και λογικών και φυσικών απαιτήσεων, υπάρχουν περιορισμοί στις τιμές που λαμβάνουν τα δεδομένα στις σχέσεις μια συγκεκριμένης Βάσης Δεδομένων.

Στην παρούσα ενότητα εξετάζουμε τους πιο βασικούς τύπους περιορισμών που μπορούν να ορισθούν στο σχεσιακό μοντέλο, δηλαδή περιορισμούς που βασίζονται στο σχήμα. Ένας από τους πιο σημαντικούς περιορισμούς αφορά στην έννοια του κλειδιού. Όλες οι πλειάδες σε μία σχέση πρέπει να είναι διαφορετικές, δηλαδή δεν είναι δυνατό δύο πλειάδες να έχουν τον ίδιο συνδυασμό δεδομένων για όλα τα γνωρίσματα της σχέσης. Συνήθως υπάρχει ένα υποσύνολο γνωρισμάτων το οποίο έχει διαφορετική τιμή για όλες τις πλειάδες μιας σχέσης το οποίο ονομάζεται υπερ-κλειδί (superkey). Το υπερ-κλειδί της σχέσης το οποίο είναι το ελάχιστο, δηλαδή πρέπει να διατηρούμε κάθε γνώρισμά του για να έχει διαφορετικές τιμές για κάθε πλειάδα σε οποιαδήποτε κατάσταση της σχέσης ονομάζεται **κλειδί**.

Στην σχέση ΥΠΑΛΛΗΛΟΣ παραπάνω κλειδί είναι το γνώρισμα Αριθμός Δελτίου Ταυτότητας καθώς εκ κατασκευής ο ΑΔΤ εκδίδεται και είναι μοναδικός για κάθε πρόσωπο.

Στην περίπτωση που μία σχέση έχει περισσότερα από ένα κλειδιά τότε κάθε κλειδί ονομάζεται υποψήφιο κλειδί. Για παράδειγμα αν στη σχέση ΥΠΑΛΛΗΛΟΣ περιλαμβανόταν το γνώρισμα ΑΦΜ τότε θα είχε δύο υποψήφια κλειδιά. Συνήθως, καθορίζουμε ένα από τα υποψηφια κλειδιά να είναι το **πρωτεύον κλειδί (primary key)** και μάλιστα συνήθως καλύτερο είναι να επιλέγουμε ως πρωτεύον κλειδί εκείνο με ένα ή τον ελάχιστο αριθμό γνωρισμάτων.

Υιοθετούμε, επίσης, τη σύμβαση να υπογραμμίζουμε τα γνωρίσματα που αποτελούν το πρωτεύον κλειδί ενός σχήματος σχέσης όπως δείχνει το σχήμα της σχέσης ΥΠΑΛΛΗΛΟΣ για το ΑΔΤ.

Στα πλαίσια ορθής χρήσης του πρωτεύοντος κλειδιού, ορίζεται ως περιορισμός ακεραιότητας οντοτήτων η υποχρέωση να τηρείται ένα πρωτεύον κλειδί έτσι ώστε να μην είναι ποτέ NULL. Αυτό ισχύει καθώς με NULL τιμή του πρωτεύοντος κλειδιού θα μπορούσε να σημαίνει ότι μερικές πλειάδες δε μπορούν να αναγνωριστούν μοναδικά.

2.4.3. Περιορισμοί Ακεραιότητας Σχέσεων και Ξένο Κλειδί

Οι περιορισμοί κλειδιού και ακεραιότητας οντοτήτων αναφέρονται σε μία σχέση συγκεκριμένη. Στην περίπτωση περιορισμού αναφορικής ακεραιότητας σχέσεων (referential integrity constraint), ορίζουμε ότι κάθε πλειάδα μιας σχέσης που αναφέρεται σε μία άλλη σχέση δε μπορεί παρά να αναφέρεται σε μία υπαρκτή πλειάδα της άλλης σχέσης. Για παράδειγμα, έστω η σχέση τηλέφωνα επικοινωνίας του υπαλλήλου:

Όνομα Σχέσης

Γνωρίσματα

ΤΗΛΕΦΩΝΑ	<u>ΑΔΤ_ΥΠΑΛΛΗΛΟΥ</u>	ΤΗΛΕΦΩΝΟ	ΤΥΠΟΣ ΤΗΛΕΦΩΝΟΥ
Πλειάδες→	AB132455	2100758222	ΟΙΚΙΑΣ
	BA134231	6908457333	KINHTO
	BA134231	2100785444	ΕΡΓΑΣΙΑΣ
	X123452	6904528777	KINHTO
	X123452	2100447555	ΟΙΚΙΑΣ

Σχήμα 2 Τα γνωρίσματα και οι πλειάδες της σχέσης ΤΗΛΕΦΩΝΑ

Στην περίπτωση της σχέσης ΤΗΛΕΦΩΝΑ ο ΑΔΤ_ΥΠΑΛΛΗΛΟΥ θα πρέπει να υπάρχει στη σχέση ΥΠΑΛΛΗΛΟ, ώστε τα τηλέφωνα που αντιστοιχούνται στο συγκεκριμένο ΑΔΤ να αναφέρονται σε κάποιον υπάλληλο της υπηρεσίας.

Κατά συνέπεια, ορίζουμε ως ένα ξένο κλειδί (foreign key) μιας σχέσης R1 που αναφέρεται σε μια σχέση R2 το σύνολο γνωρισμάτων FK της R1 που έχουν το ίδιο πεδίο ορισμού με τα γνωρίσματα του πρωτεύοντος κλειδιού της R2 (τα γνωρίσματα του FK αναφέρονται στη σχέση R2) και κάθε τιμή του FK είτε εμφανίζεται ως τιμή του PK της R2 είτε είναι null (η πλειάδα με FK αναφέρεται στην πλειάδα με PK της άλλης σχέσης).

Σημειώστε ότι ένα ξένο κλειδί μπορεί να αναφέρεται στην ίδια του τη σχέση, να δημιουργεί επόμενως ενός είδους αναδρομική συσχέτιση. Για παράδειγμα η περιγραφή «έχει προϊστάμενο» του ΔΟΣ σχήματος 2 στο σχεσιακό μοντέλο θα μπορούσε να είναι:

Όνομα Σχέσης

Γνωρίσματα

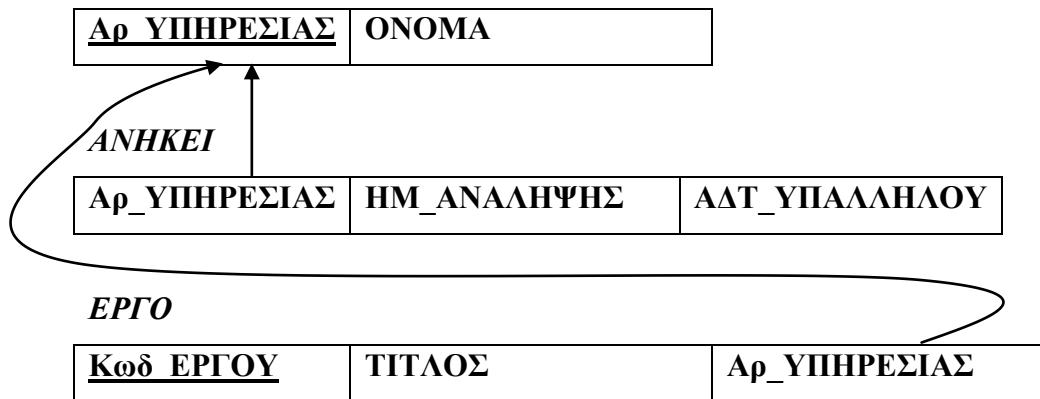
ΥΠΑΛΛΗΛΟΣ	<u>ΑΔΤ</u>	ΟΝΟΜΑΤΕΠΩΝΥΜΟ	ΦΥΛΟ	ΠΡΟΪΣΤΑΜΕΝΟΣ
Πλειάδες→	AB132455	ΠΙΤΡΟΣ ΙΩΑΝΝΟΥ	A	BA134231
	BA134231	ΜΑΡΙΑ ΡΕΛΛΗ	Θ	NULL
	X123452	ΜΑΡΙΟΣ ΖΗΚΑΣ	A	BA134231

Σχήμα 3 Τα γνωρίσματα και οι πλειάδες της σχέσης ΤΗΛΕΦΩΝΑ

Οι περιορισμοί αναφορικής ακεραιότητας για το σχεσιακό σχήμα της ΒΔ ΥΠΟΥΡΓΕΙΟ είναι οι ακόλουθοι συνολικά στην παρακάτω εικόνα.



ΥΠΗΡΕΣΙΑ

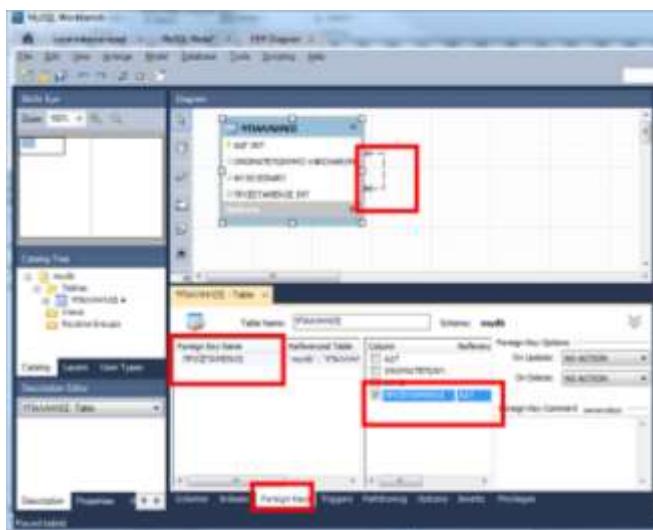


ΕΞΑΡΤΩΜΕΝΟ_ΜΕΛΟΣ

<u>ΑΔΤ_ΥΠΑΛΛΗΛΟΥ</u>	<u>ΟΝΟΜΑΤΕΠΩΝΥΜΟ</u>	<u>ΗΜ_ΓΕΝΝΗΣΗΣ</u>	<u>ΧΩΡΑ</u>
----------------------	----------------------	--------------------	-------------

Εικόνα 12 Περιορισμοί αναφορικής ακεραιότητας για το σχεσιακό σχήμα της ΒΔ ΥΠΟΥΡΓΕΙΟ

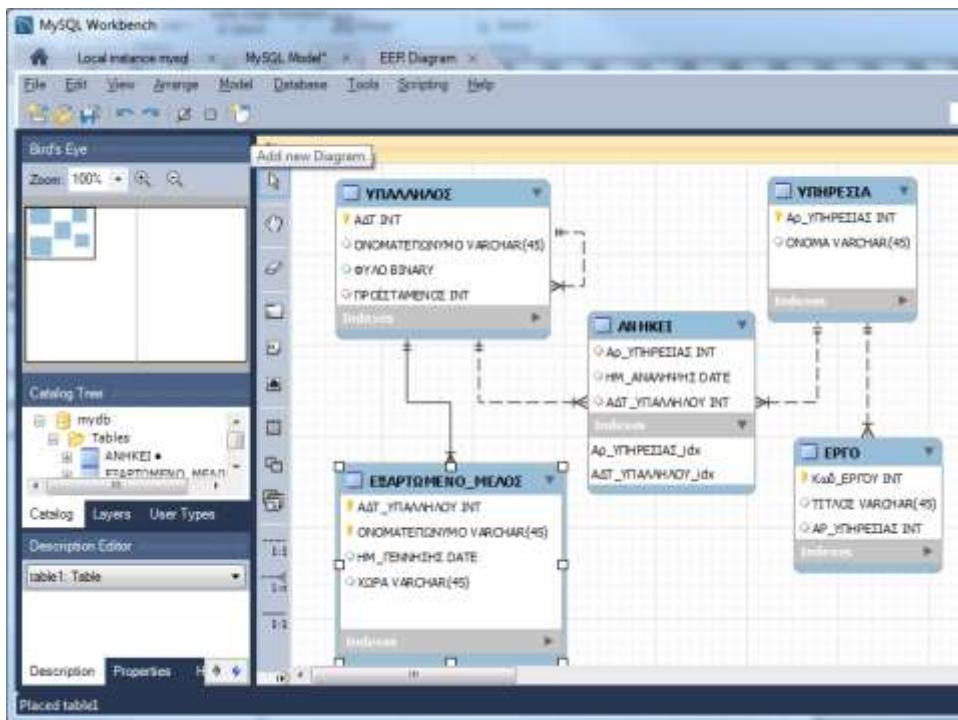
Με το εργαλείο διαγραμμάτων EER Diagram του MySQL WorkBench η αναφορική ακεραιότητα για τη σχέση ΥΠΑΛΛΗΛΟΣ θα ήταν ως ακολούθως:



Εικόνα 13 Περιορισμοί αναφορικής ακεραιότητας για τον Προϊστάμενο στη σχέση ΥΠΑΛΛΗΛΟΣ

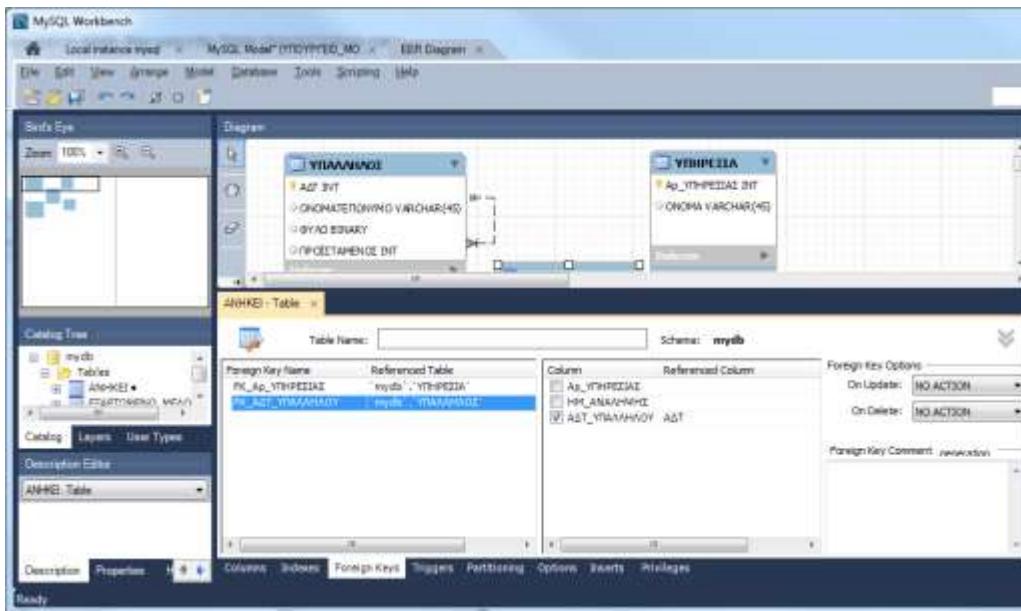
Στο ακόλουθο σχήμα παρατηρούμε την υλοποίηση των αναφορικών ακεραιοτήτων μέσω του σχεδιαστικού εργαλείου MySQL WorkBench για τη ΒΔ ΥΠΟΥΡΓΕΙΟ. Παρατηρούμε ότι για κάθε αναφορική ακεραιότητα εμφανίζονται κατάλληλες συνδέσεις από τη μία σχέση στην άλλη (ή και στην ίδια για την περίπτωση του ΥΠΑΛΛΗΛΟΣ).

Παρατηρούμε επίσης ότι έχουν δηλωθεί με κίτρινο σύμβολο κλειδιού τα πρωτεύοντα κλειδιά κάθε σχέσης. Στη σχέση ΕΞΑΡΤΩΜΕΝΟ_ΜΕΛΟΣ παρατηρούμε ότι το πρωτεύοντον κλειδί περιλαμβάνει δύο (2) γνωρίσματα.



Εικόνα 14 Περιορισμοί αναφορικής ακεραιότητας για τη ΒΔ ΥΠΟΥΡΓΕΙΟ

Ας δούμε στη συνέχεια με ποιον τρόπο μπορούμε να κατασκευάσουμε έναν περιορισμό αναφορικής ακεραιότητας με το σχεδιαστικό εργαλείο που προσφέρει το MySQL WorkBench. Στη συνέχεια παρουσιάζονται στοιχεία της καρτέλας ξένα κλειδιά (foreign keys tab).



Εικόνα 15 Περιβάλλον σχεδίασης πινάκων Foreign Keys Tab στο MySQL WorkBench

Για να προσθέσετε ένα ξένο κλειδί, κάνετε κλικ στη στήλη Foreign Key Name και συμπληρώνετε το όνομα του ξένου κλειδιού. Επιλέξτε τη στήλη του πίνακα και αντιστοιχίστε το στο γνώρισμα του πίνακα που αναφέρεται.

Στις επιλογές του ξένου κλειδιού είναι δυνατό να γίνουν ρυθμίσεις για να αντιμετωπιστούν

πιθανά update ή delete με αντίστοιχη ενέργεια όπως:

RESTRICT

CASCADE

SET NULL

NO ACTION

Αυτό σημαίνει ότι όταν γίνει κάποια ενημέρωση ή διαγραφή μιας γραμμής του πίνακα στον οποίο αναφέρεται το ξένο κλειδί κατά πόσο θα μετακυληθεί η ενέργεια και στον παρόντα πίνακα. Για παράδειγμα αν πραγματοποιηθεί διαγραφή ενός υπαλλήλου από τη σχέση ΥΠΑΛΛΗΛΟΣ κατά πόσο θα διαγραφούν τα τηλέφωνα επικοινωνίας στον πίνακα ΤΗΛΕΦΩΝΑ ή θα τεθούν σε null ή απλώς θα παραμείνουν ως έχουν.

2.4.4. Μετατροπή του μοντέλου ΟΣ σε Σχεσιακό Μοντέλο

Όπως περιγράψαμε, ο εννοιολογικός σχεδιασμός μιας βάσης δεδομένων οδηγεί στη δημιουργία ενός Διαγράμματος Οντοτήτων-Συσχετίσεων. Κάθε ΔΟΣ περιλαμβάνει τύπους οντοτήτων, τύπους συσχετίσεων και τα γνωρίσματα των οντοτήτων και των συσχετίσεων με φορμαλιστικό τρόπο και χωρίς τεχνολογικές πτυχές.

Όπως κάθε σχεδίαση, έτσι και η εννοιολογική σχεδίαση μιας βάσης δεδομένων δεν υπακούει σε αυστηρούς κανόνες για τον προσδιορισμό των οντοτήτων, των συσχετίσεων και των γνωρισμάτων και πώς αυτά πρέπει να απεικονίζονται σε ένα ΔΟΣ. Υπάρχει επομένως δυνατότητα εναλλακτικών ΔΟΣ που να απεικονίζουν τις ίδιες απαιτήσεις για μία βάση δεδομένων. Ωστόσο, υπάρχουν κατευθύνσεις – κανόνες που εφόσον τους ακολουθεί κανείς είναι δυνατό να καθορίσει με μεγάλη επιτυχία τις οντότητες και συσχετίσεις που περιγράφονται στις απαιτήσεις και περιορισμούς μιας βάσης δεδομένων κατά τη φάση της σχεδίασης. Στη συνέχεια παρουσιάζουμε τους κανόνες αυτούς εν είδει γενικών κατευθύνσεων.



Γενική Κατεύθυνση 1:

Ένα σύνολο οντοτήτων (πχ «ΥΠΑΛΛΗΛΟΙ») περιγράφεται βάσει των χαρακτηριστικών, (ή ιδιοτήτων ή κατηγορημάτων) που έχει κάθε οντότητα που ανήκει στο σύνολο (πχ «Ονοματεπώνυμο»). Τα χαρακτηριστικά αυτά πρέπει να απορρέουν από τις απαιτήσεις του συστήματος υπό σχεδίαση και όχι απλώς να συνηθίζονται στην καθημερινή πραγματικότητα αλλά να μην έχουν επιχειρηματική αξία. Κάθε σύνολο οντοτήτων έχει τα δικά του γνωρίσματα και θα πρέπει να προσέχουμε να μην επαναλαμβάνουμε τυχόν γνωρίσματα με το ίδιο περιεχόμενο στη ΒΔ.



Γενική Κατεύθυνση 2:

Οι έννοιες που υπάρχουν στον πραγματικό κόσμο (π.χ. Υπάλληλος) ορίζεται συνήθως ως σύνολο οντοτήτων κατά το σχεδιασμό ενός διαγράμματος οντοτήτων συσχετίσεων όταν η εφαρμογή που σχεδιάζεται χρειάζεται να χρησιμοποιεί δεδομένα που αποθηκεύονται σε κάποια από τα χαρακτηριστικά της οντότητας. Για να υπάρχει η οντότητα αυτή θα πρέπει με βάση τα χαρακτηριστικά να είναι δυνατό να διαχωρίζεται κάθε στιγμιότυπο από κάποιο άλλο.



Γενική Κατεύθυνση 3

Ενα σύνολο οντοτήτων συνήθως είναι δυνατό να εκφραστεί στον πληθυντικό αριθμό. Το τελικό όνομα κάθε οντότητας μπορεί να είναι είτε στον ενικό είτε στον πληθυντικό αριθμό, ωστόσο συνήθως επιλέγεται ο ενικός (π.χ. Υπάλληλος).



Γενική Κατεύθυνση 4

Τα ρήματα σε προτάσεις τα οποία έχουν υποκείμενο και αντικείμενο έννοιες που έχουν απεικονιστεί ως οντότητες είναι συνήθως συσχετίσεις στο ΔΟΣ που δημιουργείται.



Γενική Κατεύθυνση 5

Στην περίπτωση που μια οντότητα διαφαίνεται ότι απαιτεί γνωρίσματα άλλης οντότητας για να ολοκληρωθεί, τότε θα πρέπει να οριστεί πιθανώς ως συσχέτιση μεταξύ οντοτήτων.



Έστω ο εξής εννοιολογικός σχεδιασμός:

Σύνολο οντοτήτων: **ΥΠΑΛΛΗΛΟΣ**,

με κατηγορήματα ΑΔΤ, Ονοματεπώνυμο, Φύλο.

Ο ΑΔΤ – Αριθμός Δελτίου Ταυτότητας προσδιορίζει μοναδικά τον υπάλληλο.

Σύνολο οντοτήτων: **ΥΠΗΡΕΣΙΑ**,

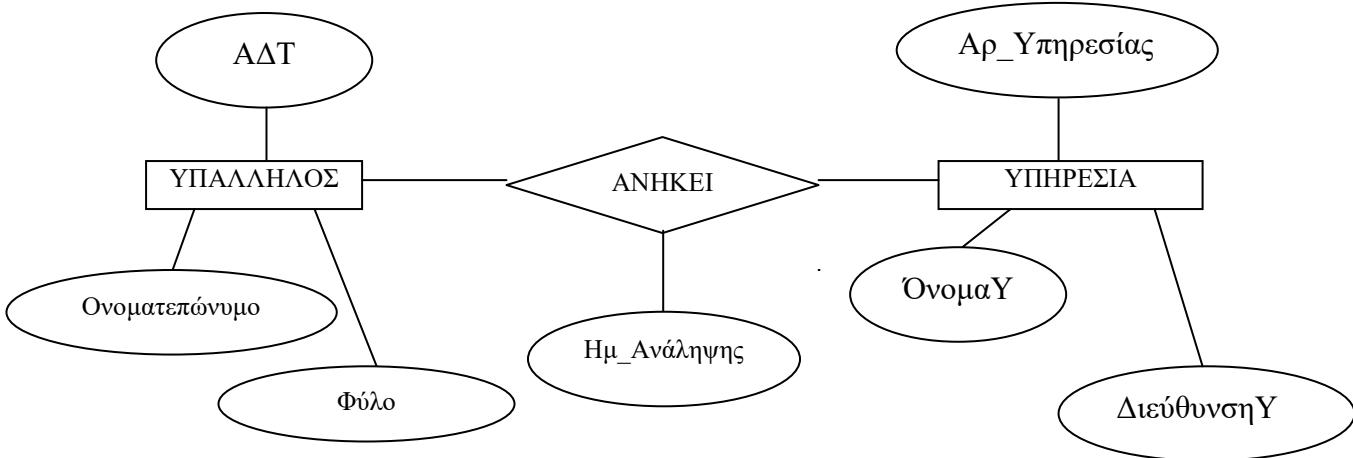
με κατηγορήματα **Αρ_Υπηρεσίας, ΌνομαΥ, ΔιεύθυνσηΥ**.

Ο **Αρ_Υπηρεσίας** προσδιορίζει μοναδικά την κάθε υπηρεσία με έναν κωδικό π.χ. <Γ4, ΥΠΕΞ Δ/ΝΣΗ ΔΙΚΑΙΟΣΥΝΗΣ-ΕΣΩΤ.ΥΠΟΘΕΣΕΩΝ-SCHENGEN, ΒΑΣΙΛΙΣΣΗΣ ΣΟΦΙΑΣ 1>.

Συσχέτιση μεταξύ ΥΠΑΛΛΗΛΟΣ και ΥΠΗΡΕΣΙΑ: **ΑΝΗΚΕΙ**,

με κατηγόρημα Ημ_Ανάληψης.

Ακολουθεί το ΔΟΣ της παραπάνω καταγραφής:



Παράδειγμα

Το παραπάνω ΔΟΣ είναι δυνατό να μετατραπεί στο σχεσιακό μοντέλο ως ακολούθως. Θα καθοριστούν τρεις σχέσεις – πίνακες με μια σειρά από γνωρίσματα:

Σχέση ΥΠΑΛΛΗΛΟΣ(ΑΔΤ, Όνοματεπώνυμο, Φύλο) με κλειδί το γνώρισμα ΑΔΤ,

Σχέση ΥΠΗΡΕΣΙΑ (Αρ_Υπηρεσίας, ΌνομαY, ΔιεύθυνσηY) με κλειδί το γνώρισμα Αρ_Υπηρεσίας ,

Σχέση ΑΝΗΚΕΙ (ΑΔΤ, Αρ_Υπηρεσίας, Ημ_Ανάληψης) με κλειδί το ζεύγος γνωρισμάτων (ΑΔΤ, Αρ_Υπηρεσίας),

Αν θεωρήσουμε ότι οι παραπάνω σχέσεις περιλαμβάνουν δεδομένα τότε θα είχαμε υπό μορφή πινάκων τις ακόλουθες απεικονίσεις για ένα στιγμιότυπο της κάθε σχέσεις με ορισμένες οντότητες:

ΥΠΑΛΛΗΛΟΣ

ΑΔΤ	Όνοματεπώνυμο	Φύλο
1324	Κώστας Ιωάννου	Α
4343	Μάριος Λέκκας	Α
4345	Μαρίνα Γιώτη	Θ

ΥΠΗΡΕΣΙΑ

<u>Αρ_Υπηρεσίας</u>	ΌνομαΥ	ΔιεύθυνσηΥ
AB123	Τμήμα Σχεδιασμού	Ρόδου 4
AB115	Τμήμα Πληροφορικής	Λαμίας 16

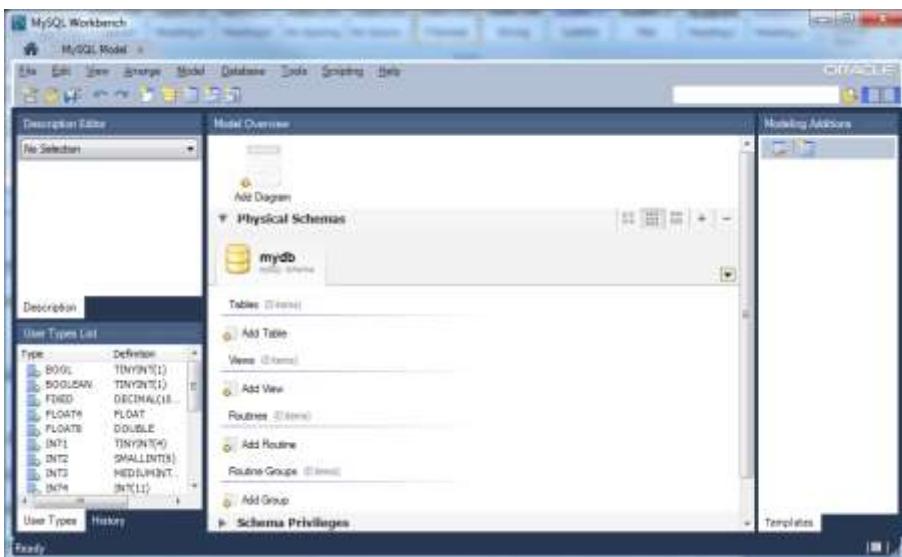
ANHKEI

<u>ΑΔΤ</u>	<u>Αρ_Υπηρεσίας</u>	Ημ_Ανάληψης
1324	AB123	1/1/1997
4343	AB123	15/6/2002
4343	AB115	1/3/2010
1324	AB115	1/10/2006
...

Πίνακες ΒΔ ΥΠΟΥΡΓΕΙΟ

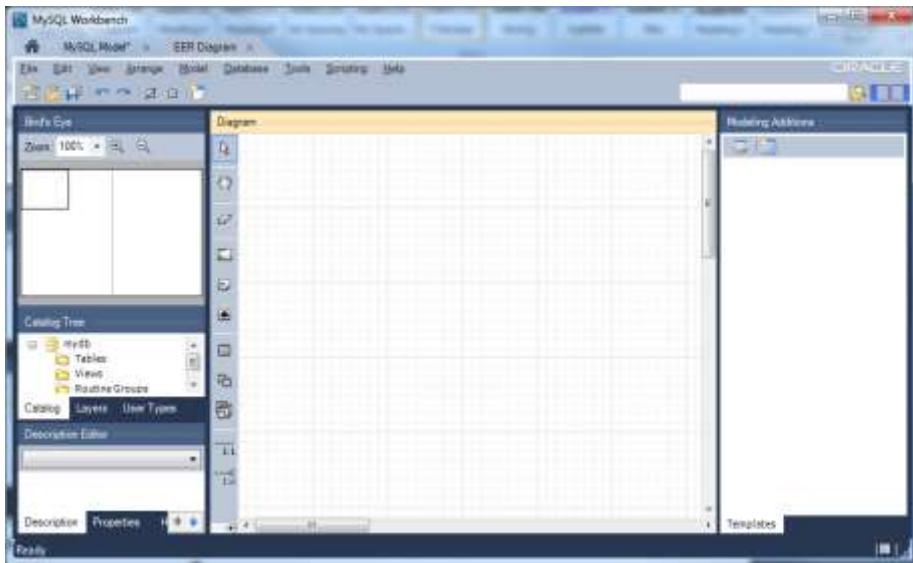
Παράδειγμα

Στη συνέχεια θα δημιουργήσουμε το σχεσιακό μοντέλο που καταγράφαμε παραπάνω μέσα από το εργαλείο σχεδίασης μοντέλων που δίνει το γραφικό περιβάλλον workbench της MySQL.



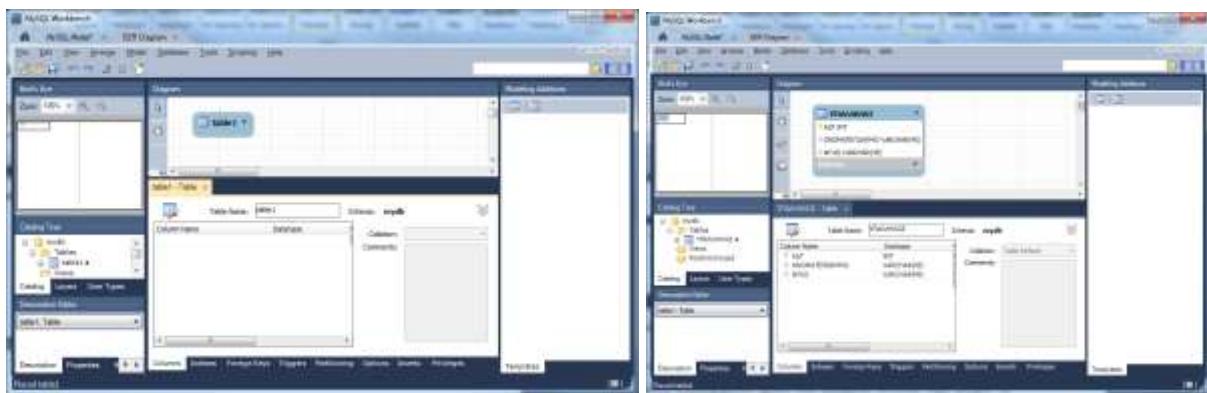
Εικόνα 16 Περιβάλλον Σχεδίασης Σχεσιακού Μοντέλου στη MySQL

Επιλέγουμε Add Diagram και παρουσιάζεται ο χώρος σχεδίασης διαγραμμάτων EER Diagram με ενεργοποιημένη τη γραμμή εργαλείων κάθετα στο κεντρικό παράθυρο.



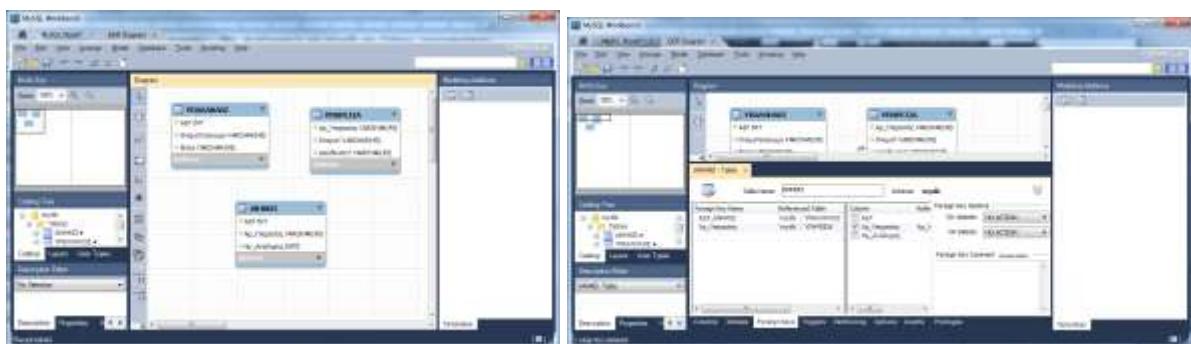
Εικόνα 17 Περιβάλλον Σχεδίασης Διαγράμματος Σχεσιακού Μοντέλου στη MySQL

Στη συνέχεια προσθέτουμε ένα άλλο πίνακα στο σχεσιακό διάγραμμα. Επιλέγουμε την ενότητα **Tables** στην πλευρά των πινάκων και επιλέγουμε την ενότητα **Table**. Στην λεπτομέρεια **Table Name** γράψουμε **EMPLOYEE**, στην λεπτομέρεια **Schema** επιλέγουμε **mydb** και στην λεπτομέρεια **Table Type** επιλέγουμε **InnoDB**. Στην λεπτομέρεια **Temporary** επιλέγουμε **No**. Στην λεπτομέρεια **Comment** γράψουμε **Employee Information**. Στην λεπτομέρεια **Character Set** επιλέγουμε **utf8mb4** και στην λεπτομέρεια **Collation** επιλέγουμε **utf8mb4_unicode_ci**.



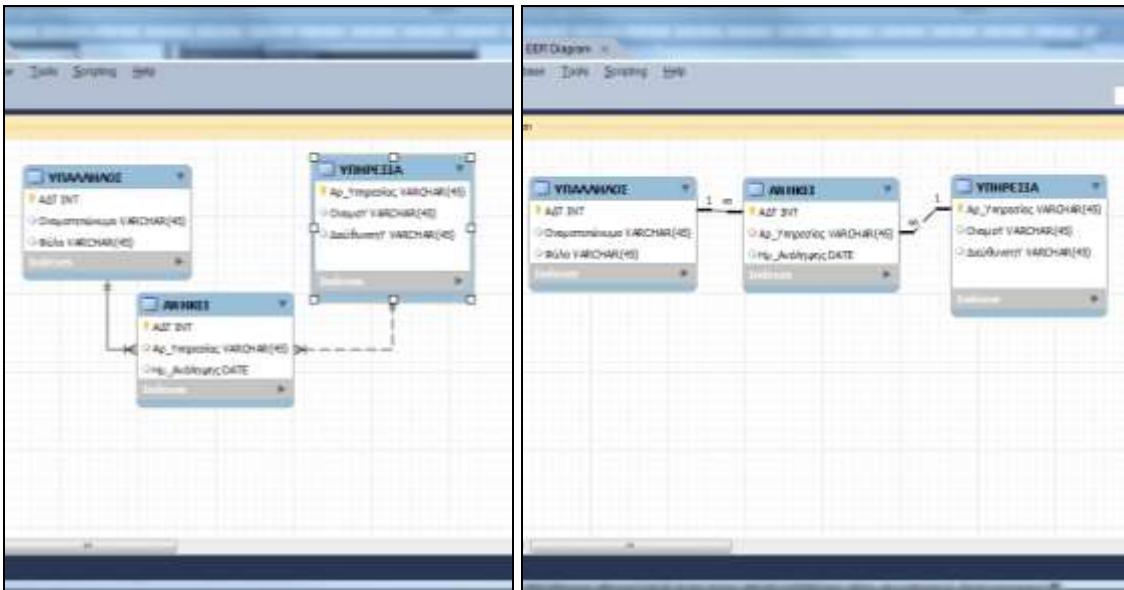
Εικόνα 18 Σχεδίαση ενός πίνακα στο σχεσιακό διάγραμμα

Ολοκληρώνοντας την προσθήκη των σχέσεων βλέπουμε στην πρώτη εικόνα παρακάτω όλους τους πίνακες ενώ στη δεύτερη ρυθμίζουμε τα ξένα κλειδιά.



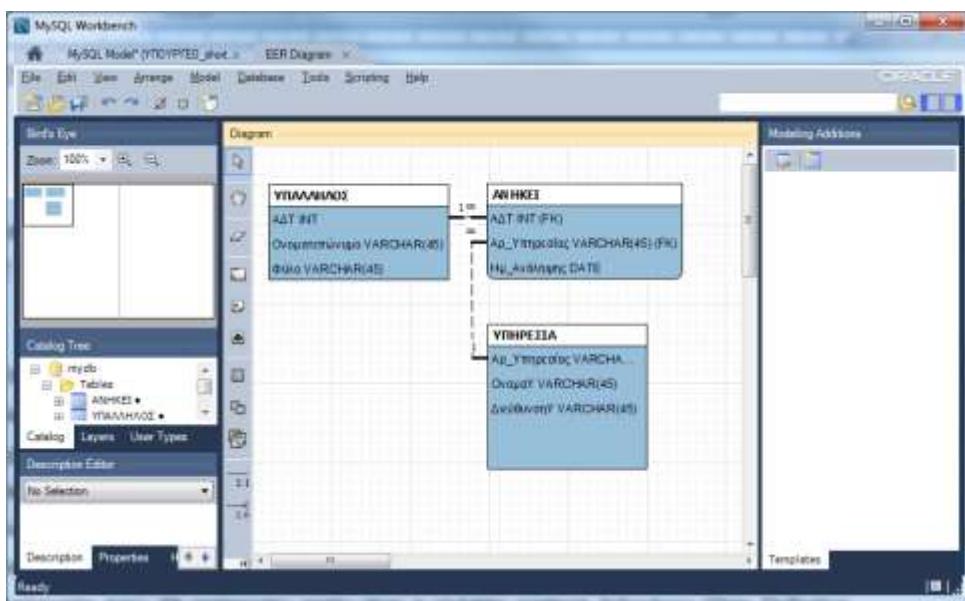
Εικόνα 19 Σχεδίαση των πινάκων και των ξένων κλειδιών στο σχεσιακό διάγραμμα

Η τελική απεικόνιση του σχεδίασμου για το σχεσιακό μοντέλο θα είναι κατά συνέπεια:



Εικόνα 20 Σχεδίαση των πινάκων συνολικά στο σχεσιακό διάγραμμα με δύο διαφορετικές επιλογές απεικόνισης των συσχετίσεων

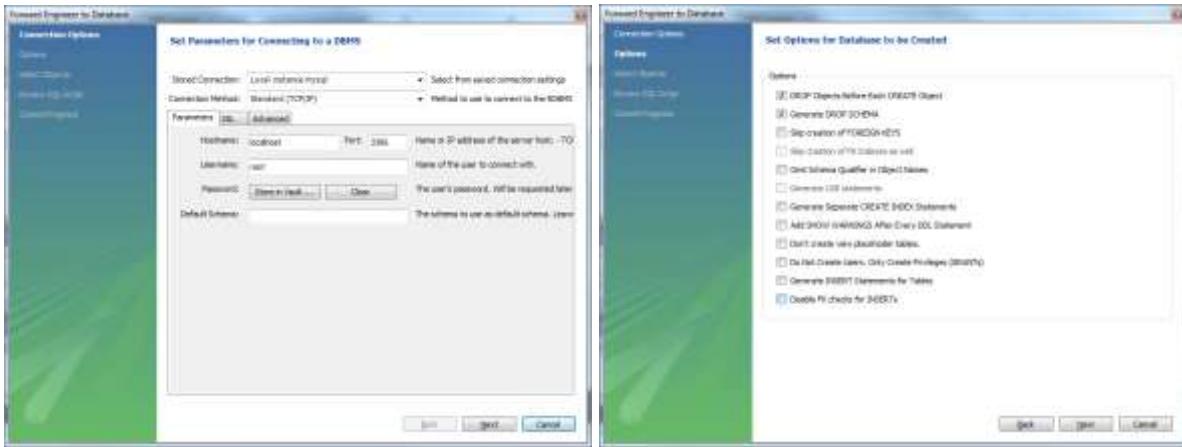
Είναι δυνατό από το μενού model να επιλεγεί ως μορφή απεικόνισης των συσχετίσεων (Relationship Notation) η αντιστοίχηση των πεδίων-χαρακτηριστικών που συσχετίζονται ένα προς ένα (connect to columns) όπως βλέπουμε στην παραπάνω εικόνα δεξιά. Είναι δυνατό επίσης να τροποποιηθεί η απεικόνιση του διαγράμματος σε πιο απλή μορφή όπως παρακάτω:



Εικόνα 21 Σχεδίαση των πινάκων συνολικά στο σχεσιακό διάγραμμα με απλή παρουσίαση

Η απεικόνιση του σχεσιακού μοντέλου επιτρέπει στη MySQL να δημιουργήσει με **Forward Engineering** την τελική ΒΔ μέσω χρήσης κατάλληλου οδηγού. Θα δημιουργηθούν οι απαραίτητοι πίνακες, πεδία στήλες με τους κατάλληλους τύπους δεδομένων καθώς και θα

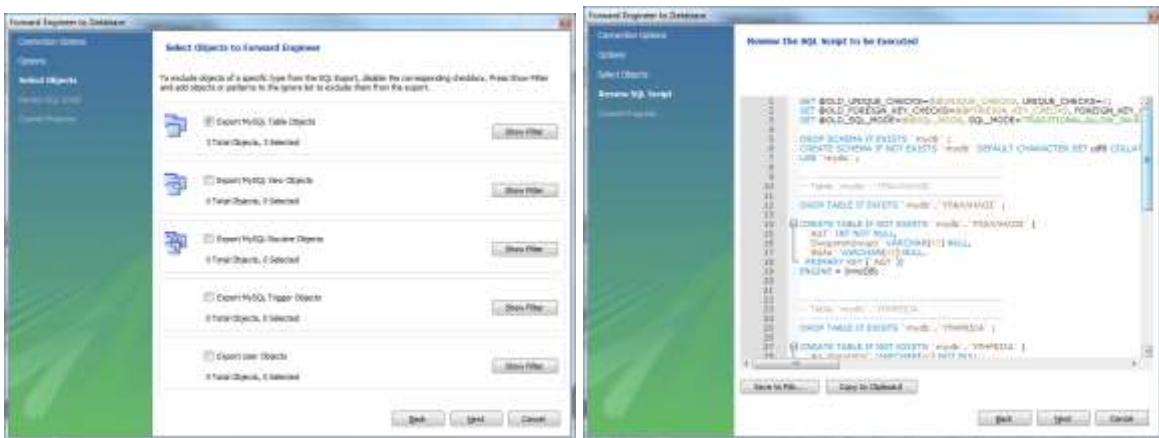
ενεργοποιηθούν οι συσχετίσεις και οι περιορισμοί των ξένων κλειδιών. Από το menu Database επιλέγουμε Forward Engineering και ακολουθούμε τον οδηγό όπως στο παρακάτω παράδειγμα:



Εικόνα 22 Οδηγός Forward Engineering Αρχική επιλογή εξυπηρετητή MySQL και 2o βήμα επιλογών

Στην παραπάνω εικόνα επιλέγουμε αριστερά τον εξυπηρετητή στο host που θέλουμε να συνδεθούμε και στη συνέχεια τις επιλογές για τη δημιουργία της ΒΔ. Μπορούμε μεταξύ άλλων να επιλέξουμε να δημιουργηθούν κατάλληλες SQL εντολές ώστε να σβήνεται η παλαιότερη ΒΔ με το ίδιο όνομα αν υπάρχει αλλά και αντικείμενα (πίνακες κλπ) που θα δημιουργήσουμε αν τυχόν υπάρχουν.

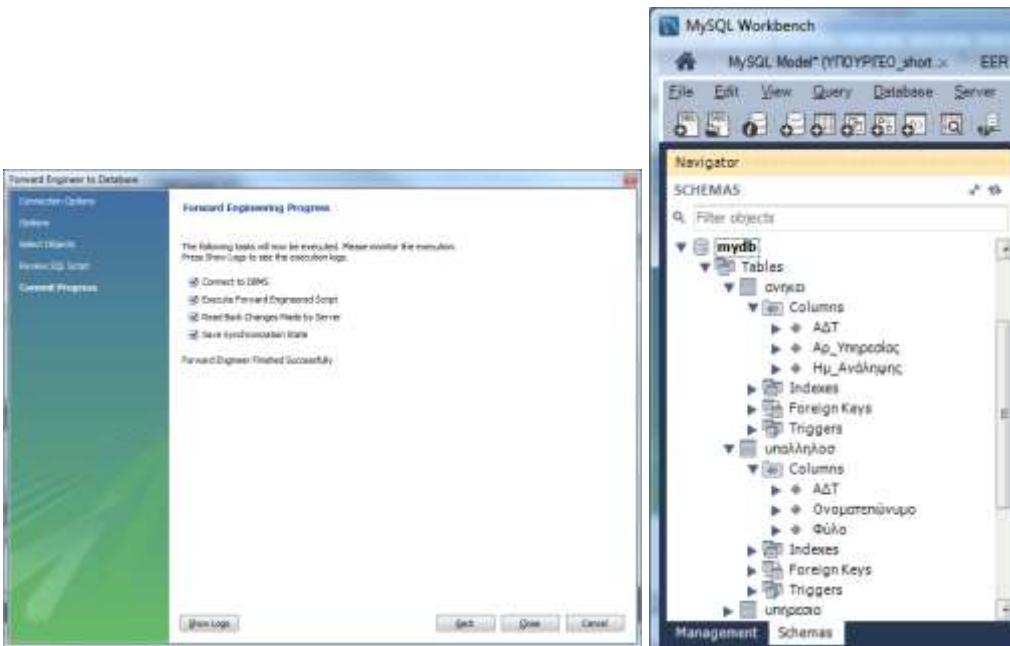
Στη συνέχεια μπορούμε να επιλέξουμε κατά πόσο θα δημιουργηθούν όλα τα αντικείμενα που σχεδιάστηκαν στο διάγραμμα ή μόνο κάποια από αυτά επιλέγοντας κατάλληλα. Αν αφήσουμε τις προεπιλογές θα δημιουργηθεί η πλήρης ΒΔ. Προχωρώντας στο επόμενο βήμα ο οδηγός Forward Engineering μας παρουσιάζει τον κώδικα SQL δημιουργίας της ΒΔ και όλων των αντικειμένων που σχεδιάσαμε.



Εικόνα 23 Οδηγός Forward Engineering 3o βήμα επιλογών αντικειμένων για δημιουργία και 4o βήμα κώδικας SQL δημιουργίας αντικειμένων (δεξιά)

Στη συνέχεια εκτελούνται οι εντολές δημιουργίας και υλοποιείται η ΒΔ. Στην παρακάτω εικόνα δεξιά μπορείτε να δείτε στο WorkBench τη ΒΔ που δημιουργήθηκε με όνομα mydb καθώς και

τους τρεις πίνακες που σχεδιάστηκαν με τις στήλες τους.



Εικόνα 24 Οδηγός Forward Engineering 50 ολοκλήρωση εκτέλεσης και παρουσίαση της ΒΔ στο WorkBench (δεξιά)

Είναι δυνατή επίσης η αντίστροφη διαδικασία. Χρησιμοποιώντας τον οδηγό reverse engineering το περιβάλλον MySQL WorkBench είναι δυνατό να σχεδιάσει το σχεσιακό μοντέλο μιας ΒΔ που ήδη έχει υλοποιηθεί και να παράγει το EER diagram. Κάτι τέτοιο είναι χρήσιμο όταν απαιτούνται τροποποιήσεις στη ΒΔ λόγω νέων απαιτήσεων και δεν υπάρχει ήδη το σχεσιακό διάγραμμα του μοντέλου.

2.5.Η γλώσσα πρότυπο SQL

Οι SQL εντολές είναι εντολές που χρησιμοποιούνται για την επικοινωνία με τη βάση δεδομένων για την εκτέλεση συγκεκριμένου έργου στα δεδομένα. Εντολές SQL μπορεί να χρησιμοποιηθούν όχι μόνο για την αναζήτηση στη βάση δεδομένων, αλλά και για να εκτελούν διάφορες άλλες λειτουργίες, όπως, για παράδειγμα, η δημιουργία πινάκων, η προσθήκη δεδομένων σε πίνακες, ή η τροποποίηση δεδομένων, η αφαίρεση ενός πίνακα, ο ορισμός δικαιωμάτων για τους χρήστες. Οι εντολές SQL ομαδοποιούνται σε κατηγορίες, ανάλογα με τη λειτουργία τους. Οι κατηγορίες αυτές είναι η γλώσσα ορισμού δεδομένων (Data Definition Language - DDL), η γλώσσα χειρισμού δεδομένων (Data Manipulation Language - DML) και η γλώσσα ελέγχου δεδομένων (Data Control Language - DCL).

2.5.1. Γλώσσα Ορισμού Δεδομένων (ΓΟΔ-DDL)

Η ΓΟΔ-DDL χρησιμοποιείται για τον ορισμό της δομής ή του σχήματος της βάσης δεδομένων.

Μερικά παραδείγματα:

- CREATE - για τη δημιουργία αντικειμένων στη βάση δεδομένων,
- ALTER - για την αλλαγή στη δομή της βάσης δεδομένων,
- DROP - για τη διαγραφή αντικειμένων από τη βάση δεδομένων,
- TRUNCATE - για την αφαίρεση των εγγραφών από έναν πίνακα, συμπεριλαμβανομένων όλων των χώρων που διατίθενται για τα αρχεία,
- COMMENT - για να προσθέτουμε σχόλια στο λεξικό δεδομένων,
- RENAME - για τη μετονομασία ενός αντικειμένου.

2.5.2. Γλώσσα Χειρισμού Δεδομένων (ΓΧΔ-DML)

Η ΓΧΔ-DML χρησιμοποιείται για την εισαγωγή, ενημέρωση και διαγραφή δεδομένων και για τη διατύπωση ερωτημάτων στο πλαίσιο του σχήματος των δεδομένων. Μερικά παραδείγματα είναι τα παρακάτω:

- SELECT - για την ανάκτηση δεδομένων από τη βάση δεδομένων,
- INSERT - για την εισαγωγή δεδομένων σε έναν πίνακα,
- UPDATE - για ενημέρωση σε υπάρχοντα δεδομένα σε έναν πίνακα,
- DELETE - για τη διαγραφή όλων των αρχείων από έναν πίνακα, με το χώρο για τα αρχεία να παραμένουν,
- MERGE - UPSERT λειτουργίες ενημέρωσης – εισαγωγής,
- CALL - κλήση ενός PL/SQL ή Java υποπρογράμματος,
- EXPLAIN PLAN - εξήγηση διαδρομής πρόσβασης σε δεδομένα,
- LOCK TABLE - συγχρονισμός ελέγχου/

2.5.3. Γλώσσα Ελέγχου Δεδομένων (ΓΕΔ-DCL)

Η ΓΕΔ-DCL χειρίζεται τα δικαιώματα πρόσβασης και περιλαμβάνει:

- GRANT – δίνει δικαιώματα πρόσβασης στους χρήστες για ΒΔ και αντικείμενά τους,
- REVOKE – αφαιρεί δικαιώματα πρόσβασης των χρηστών της ΒΔ που έχουν αποδοθεί με την GRANT,

2.5.4. Γλώσσα Ελέγχου Δοσοληψιών

Οι εντολές ελέγχου διαχειρίζονται τις αλλαγές που έγιναν από τις εντολές ΓΧΔ. Αυτές οι εντολές SQL χρησιμοποιούνται για τη διαχείριση των αλλαγών που επηρεάζουν τα δεδομένα. Τέτοιες εντολές είναι οι

- COMMIT,
- ROLLBACK και
- SAVEPOINT.

3. Εισαγωγή στο περιβάλλον λειτουργίας της MySQL

Μελετώντας το παρόν κεφάλαιο οι επιμορφωνόμενοι θα είναι σε θέση να:

- Περιγράφουν το μοντέλο client/server
- Κατανοούν τη λειτουργία του MySQL Server
- Χρησιμοποιούν τα διαθέσιμα εργαλεία για την πρόσβαση και διαχείριση των δεδομένων στην MySQL

3.1. Το Περιβάλλον της MySQL

Η MySQL είναι μία πολύ δημοφιλής και ευρέως χρησιμοποιούμενη βάση δεδομένων ανοικτού κώδικα, λόγω της απόδοσής της, της αξιοπιστίας και της ευκολίας στη χρήση. Η MySQL παρέχει στους διαχειριστές και τους προγραμματιστές απόδοση, επεκτασιμότητα και όλες τις βελτιώσεις που απαιτούνται κατά τη δημιουργία προηγμένων Διαδικτυακών εφαρμογών και υπηρεσιών στο υπολογιστικό νέφος. Ανεξάρτητα από το μέγεθος της υπηρεσίας ή του οργανισμού που θα τη χρησιμοποιήσει, είτε πρόκειται για μια ταχέως αναπτυσσόμενη μικρή υπηρεσία, είτε για μία τυπική υπηρεσία ή κάποιο μεγάλο οργανισμό, η MySQL μπορεί να βοηθήσει παρέχοντας υψηλών επιδόσεων κλιμακούμενες λύσεις βάσεων δεδομένων, διατηρώντας μια ισορροπία κόστους και αποτελεσματικότητας. Η MySQL αποτελεί ένα ΣΔΒΔ που χρησιμοποιεί την SQL για να υλοποιήσει λειτουργίες διαχείρισης βάσεων δεδομένων, όπως η διαχείριση πινάκων (δημιουργία, τροποποίηση, διαγραφή), τη διαχείριση δεδομένων εγγραφών (εισαγωγή, τροποποίηση, διαγραφή) και την ανάκτηση δεδομένων από τις βάσεις, οι οποίες ικανοποιούν συγκεκριμένα κριτήρια.

Η MySQL Community Edition είναι η έκδοση η οποία μπορεί να γίνει download από τον χρήστη ελεύθερα. Είναι διαθέσιμη υπό την άδεια GPL και υποστηρίζεται από μία πολύ μεγάλη και ενεργή κοινότητα προγραμματιστών ανοικτού κώδικα. Η MySQL Community Edition συμπεριλαμβάνει τα παρακάτω εργαλεία και βοηθήματα για την πληρέστερη διαχείριση και οργάνωση των δεδομένων σας:

- Pluggable Storage Engine Architecture η οποία σας επιτρέπει να φορτώσετε και να απελευθερώσετε μία μηχανή αποθήκευσης ενόσω εκτελείται ένας εξυπηρετητής MySQL. Η αρχιτεκτονική αυτή επιτρέπει τη χρήση μιας μηχανής αποθήκευσης δηλαδή των λειτουργικών μονάδων εκείνων του φυσικού επιπέδου που εκτελούν τις τελικές επιθυμητές ενέργειες στα δεδομένα. Η συγκεκριμένη αρχιτεκτονική επιτρέπει τη χρήση

πολλαπλών μηχανών αποθήκευσης χωρίς να απαιτεί εξειδικευμένες γνώσεις από το διαχειριστή της ΒΔ ή τον προγραμματιστή.

Οι μηχανές αποθήκευσης (storage engines) αναλαμβάνουν την υλοποίηση των SQL εντολών για πίνακες διαφορετικών τύπων και περιλαμβάνουν:

- InnoDB : πρόκειται για την μηχανή αποθήκευση που χρησιμοποιείται προεπιλεγμένα από τη MySQL και αφορά για γενικού σκοπού αποθήκευση δεδομένων. Στο παρόν εγχειρίδιο οι ενέργειες που θα πραγματοποιήσουμε αφορούν σε αποθήκευση δεδομένων σε InnoDB εκτός αν διαφορετικά αναφέρεται. Μέγιστος αποθηκευτικός όγκος 64 TB.
- MyISAM : πρόκειται για μηχανή αποθήκευσης που έχει βελτιστοποιηθεί για περιπτώσεις όπου πραγματοποιείται ανάγνωση δεδομένων πάντοτε ή σχεδόν πάντοτε όπως στην περίπτωση εφαρμογών web portals και αποθηκών δεδομένων. Ουσιαστικά δεν περιλαμβάνει δυνατότητες για δοσοληψίες (transactions) και εφαρμογή περιορισμών αναφορικής ακεραιότητας (referential integrity). Αν υπάρχουν γενικώς πολλά select και λίγα update και delete για κάθε πίνακα της ΒΔ η MyISAM μπορεί να είναι μια επιλογή. Μέγιστος αποθηκευτικός όγκος 256 TB.
- Memory : πρόκειται για την περίπτωση μηχανής αποθήκευσης που κρατά όλα τα δεδομένα στη μνήμη RAM ώστε να βελτιστοποιεί την ταχύτητα αναζήτησης σε περιπτώσεις δεδομένων που απαιτούν πάρα πολύ γρήγορη απόκριση. Χρησιμοποιείται ολοένα και πιο σπάνια καθώς τόσο η εξέλιξη της InnoDB με το προηγμένο buffering που διαθέτει όσο και η κατανεμημένη έκδοση cluster NDBCluster παρέχουν πάρα πολύ γρήγορη απόκριση σε αναζητήσεις κλειδιού-τιμής για μεγάλου όγκου δεδομένα.
- CSV : πρόκειται για μηχανή αποθήκευσης όπου οι πίνακες κρατούνται σε μορφή αρχείων κειμένου με τιμές χωρισμένες με κόμματα. Δεν είναι δυνατό να υλοποιηθούν indexes σε τέτοιου είδους αρχεία. Συνήθως χρησιμοποιούνται σε συνδυασμό με την InnoDB για εισαγωγή και εξαγωγή δεδομένων.
- Merge: πρόκειται για μηχανή αποθήκευσης που επιτρέπει τη λογική συνένωση πολλαπλών όμοιων πινάκων MyISAM τύπου κάτω από ένα κοινό λογικό όνομα αναφοράς.
- Archive, NDB (MySQL Cluster), BlackHole, Federated, Example είναι ακόμη παραδείγματα υποστηριζόμενων μηχανών αποθήκευσης τα οποία περιγράφονται αναλυτικά στο <http://dev.mysql.com/doc/refman/5.7/en/storage-engines.html>

Το περιβάλλον MySQL επιπλέον περιλαμβάνει εργαλεία όπως:

MySQL Replication επιτρέπει την αντιγραφή δεδομένων από μία κύρια (master) Βάση

Δεδομένων σε μία ή περισσότερες Βάσεις Δεδομένων (slaves) ως αντίγραφα. Η αντιγραφή μπορεί να γίνεται ασύγχρονα χωρίς δηλαδή να είναι οι slave ΒΔ συνεχώς διασυνδεδεμένες με την κύρια. Μπορούν να αντιγράφονται ΒΔ ή μόνο κάποιοι πίνακες με σκοπό τη μεγιστοποίηση της ασφάλειας, της ακεραιότητας, της διαθεσιμότητας και προσβασιμότητας στα δεδομένα.

MySQL Partitioning για τη βελτίωση της απόδοσης και της διαχείρισης των εφαρμογών μεγάλων βάσεων. Παρόλο που η SQL έχει σχεδιαστεί ώστε να μην λαμβάνει υπόψη το φυσικό επίπεδο αποθήκευσης για τις λειτουργίες που πραγματοποιεί, τα σύγχρονα ΣΔΒΔ επιτρέπουν τη διαχείριση και του φυσικού επιπέδου. Η MySQL με την InnoDB επιτρέπει την αποθήκευση συγκεκριμένων ΒΔ ή πινάκων σε φακέλους του συστήματος αρχείων που επιλέγει ο χρήστης (π.χ. σε ένα δεύτερο HDD ή σε ένα άλλο FS Disk Partition ή σε έναν SSD κλπ) για λόγους διευκόλυνσης ή επέκτασης της χωρητικότητας ή αύξησης της ταχύτητας πρόσβασης στα δεδομένα. Με το Partitioning είναι δυνατή η διαχείριση ακόμη περισσότερων παραμέτρων σε φυσικό επίπεδο όπως η διαμοίραση μέρους ενός πίνακα σε διαφορετικά σημεία ενός συστήματος αρχείων. Η διαμοίραση είναι δυνατό να γίνεται με βάση μια υπολογιζόμενη έκφραση σε μία ή περισσότερες στήλες ενός πίνακα ή για κάποιες τιμές μιας ή περισσότερων στηλών ενός πίνακα. Το partitioning συνήθως είναι απαραίτητο μόνο για εφαρμογές ειδικού σκοπού και όταν τα δεδομένα το απαιτούν λόγω του μεγάλου όγκου τους ή της ιδιαίτερης κατανομής τους.

Στη MySQL είναι δυνατό να αποθηκευτούν προγράμματα και όψεις και συγκεκριμένα:

Stored Procedures αποτελούν αποθηκευμένες λειτουργίες – προγράμματα SQL για τη βελτίωση της παραγωγικότητας του προγραμματιστή τις οποίες είναι δυνατό να καλεί κανείς για να πραγματοποιήσει επαναλαμβανόμενες ενέργειες. Οι store procedures δεν επιστρέφουν τιμή ως return value.

Functions αποτελούν αποθηκευμένες λειτουργίες – προγράμματα SQL αντίστοιχα με τις παραπάνω stored procedures που επιστρέφουν return value.

Triggers είναι ένα αντικείμενο της ΒΔ που επιτρέπει την εφαρμογή πολύπλοκων επιχειρησιακών κανόνων στο επίπεδο της βάσης. Συσχετίζεται με ένα πίνακα και ενεργοποιείται – λειτουργεί όταν πραγματοποιείται μια ενέργεια στα δεδομένα του πίνακα όπως insert, update ή delete.

Views αποτελούν μία stored procedure που παράγει έναν πίνακα ως αποτέλεσμα-επιστρεφόμενη τιμή και προσομοιάζει έναν εικονικό πίνακα μόνο για ανάγνωση για να εξασφαλιστεί πως οι ευαίσθητες πληροφορίες δεν τίθενται σε κίνδυνο.

Events αποτελούν προγραμματιζόμενες ενέργειες που εκτελούν περιοδικά SQL ενέργειες σε ένα ή περισσότερα βήματα που είναι ιδιαίτερα χρήσιμο για εργασίες επαναλαμβανόμενες.

Performance Schema για την παρακολούθηση της κατανάλωσης πόρων σε επίπεδο χρήστη/εφαρμογής.

Information Schema για την παροχή εύκολης πρόσβασης σε μεταδεδομένα.

MySQL Connectors (ODBC, JDBC, .NET, κλπ.) για τη δημιουργία εφαρμογών σε πολλαπλές γλώσσες.

MySQL Workbench για την οπτική μοντελοποίηση, ανάπτυξη και διαχείριση της SQL.

Πρακτικά παραδείγματα θα παρουσιαστούν με χρήση του εργαλείου MySQL WorkBench το οποίο αποτελεί μια πλήρη εφαρμογή διαχείρισης ΒΔ MySQL διαθέσιμη στη διανομή ανοικτού κώδικα.

Πέρα από την έκδοση ελεύθερου λογισμικού της MySQL, το ΣΔΒΔ διατίθεται και σε εμπορικές εκδόσεις οι οποίες έχουν επιπλέον ευελιξία για να καλύψουν πιο συγκεκριμένες επιχειρησιακές και τεχνικές απαιτήσεις. Οι εκδόσεις αυτές είναι:

- MySQL Standard Edition
- MySQL Enterprise Edition
- MySQL Cluster Carrier Grade Edition

	MySQL Standard Edition	MySQL Enterprise Edition	MySQL Cluster Carrier Grade Edition
Annual Subscription Server /Year	USD 2,000 ¹	USD 5,000	USD 10,000
Oracle Premier Support ³			
24x7 Support	√	√	√
Unlimited Support Incidents	√	√	√
Knowledge Base	√	√	√
Maintenance Releases, Bug Fixes, Patches, Updates	√	√	√
MySQL Consultative Support	√	√	√
MySQL Features			
MySQL Database Server	√	√	√

¹ Price Listing 2014 <http://www.mysql.com/products/>

MySQL Connectors	√	√	√
MySQL Replication	√	√	√
MySQL Partitioning		√	√
MySQL Workbench	√	√	√
Storage Engine: MyISAM	√	√	√
Storage Engine: InnoDB	√	√	√
Storage Engine: NDB			√
MySQL Enterprise Monitor		√	√
MySQL Enterprise Dashboard		√	√
MySQL Enterprise Advisors		√	√
MySQL Query Analyzer		√	√
MySQL Replication Monitor		√	√
MySQL Enterprise Backup		√	√
Hot backup for InnoDB		√	√
Full, Incremental, Partial backup		√	√
Full, Partial restore		√	√
Point-In-Time-Recovery		√	√
MySQL Enterprise Security		√	√
External Authentication		√	√
MySQL Enterprise Audit		√	√
Policy-based auditing compliance		√	√
MySQL Enterprise Scalability		√	√
Thread Pool		√	√
MySQL Enterprise High-Availability		√	√
HA using Oracle Linux and DRBD		√	√
HA using Solaris Clustering		√	√
HA using Oracle VM Template		√	√
HA using Windows Clustering		√	√
MySQL Cluster Manager			√
Configuration & Provisioning			√

Automatic Scaling			√
Management & Monitoring			√
MySQL Cluster Geo-Replication			√
Oracle Product Certifications			
Certified with Oracle Linux	√	√	√
Certified with Oracle VM	√	√	√
Certified with Oracle Solaris	√	√	√
Certified with Oracle GoldenGate		√	√
Certified with Oracle Data Integrator		√	√
Certified with Oracle Fusion Middleware		√	√
Certified with Oracle Secure Backup		√	√
Certified with Oracle Audit Vault and Database Firewall		√	√

3.2. Επισκόπηση του μοντέλου Πελάτη / Εξυπηρετητή (Client/Server)

Το σύστημα διαχείρισης βάσεων δεδομένων MySQL λειτουργεί σε δικτυακό περιβάλλον χρησιμοποιώντας το μοντέλο πελάτη - εξυπηρετητή (client - server). Συγκεκριμένα, ο εξυπηρετητής (server) είναι ένα κεντρικό πρόγραμμα (mysqld – MySQL Daemon ή MySQL Server) που διαχειρίζεται το περιεχόμενο των βάσεων δεδομένων και τις αιτήσεις των πελατών προς οποιαδήποτε βάση δεδομένων καθώς και τα προγράμματα πελάτη (client programs) που συνδέονται με τον εξυπηρετητή για να πραγματοποιήσουν κάποια αιτήματα, να ανακτήσουν ή να τροποποιήσουν δεδομένα. Μία εγκατάσταση MySQL έχει τα ακόλουθα κύρια στοιχεία: α) έναν εξυπηρετητή MySQL (MySQL Server), β) προγράμματα πελάτη (client programs) και γ) βοηθήματα (non-client utilities).

3.2.1. MySQL Server

Ο MySQL Server είναι το πρόγραμμα εξυπηρετητή του συστήματος διαχείρισης ΒΔ MySQL. Ο εξυπηρετητής διαχειρίζεται ενέργειες και επιτελεί λειτουργίες σε βάσεις δεδομένων που αποθηκεύονται είστε στο σκληρό δίσκο τοπικά ή στο δίκτυο αλλά και στη μνήμη RAM.

Πρόκειται για πολύ-νηματικό λογισμικό και υποστηρίζει πολλές ταυτόχρονες συνδέσεις πελατών είτε φυσικών χρηστών είτε προγραμμάτων λογισμικού. Στη γενική περίπτωση θα θεωρούμε ως πελάτες προγράμματα λογισμικού. Για τη διαχείριση του περιεχομένου της βάσης δεδομένων, ο MySQL Server διαθέτει αρθρωτή αρχιτεκτονική (modular architecture) που υποστηρίζει πολλαπλές μηχανές αποθήκευσης που παρουσιάστηκαν ήδη στην εισαγωγή του κεφαλαίου 3. Σημειώνεται ότι, η ακριβής σύνθεση των χαρακτηριστικών του MySQL Server μπορεί να αλλάξει μεταξύ των εκδόσεων, έτσι κάθε φορά που ο χρήστης κάνει λήψη μιας νέας έκδοσης, θα πρέπει να ελέγχει την τεκμηρίωση.



Θα ήταν καλό στο σημείο αυτό να διευκρινιστεί η έννοια του εξυπηρετητή (server) από το σύστημα φιλοξενείας του (host). Ο εξυπηρετητής είναι ένα λογισμικό (MySQL Server). Τα χαρακτηριστικά του εξυπηρετητή περιλαμβάνουν τον αριθμό της έκδοσης, τη λίστα των χαρακτηριστικών του, τη διανομή του αν είναι εμπορική ή ανοιχτού κώδικα, και ούτω καθεξής. Το σύστημα φιλοξενίας και λειτουργίας (Host Server) είναι η μηχανή (φυσική ή εικονική (Virtual Machine-VM)) στην οποία εκτελείται το πρόγραμμα εξυπηρετητή. Τα χαρακτηριστικά του ξενιστή περιλαμβάνουν τη σύνθεση του υλικού του (CPU, RAM, Disk κλπ), το λειτουργικό σύστημα που τρέχει στη μηχανή, τις δικτυακές του απολήξεις κλπ.

3.2.2. MySQL Πελάτες & Χρήση αρχείου παραμέτρων

Τα προγράμματα-πελάτες χρησιμοποιούνται για την επικοινωνία με τον εξυπηρετητή ΣΔΒΔ για τη διαχείριση της πληροφορίας στις βάσεις δεδομένων που διαχειρίζεται. Η MySQL διαθέτει πολλαπλά προγράμματα πελάτες όπως:

- Το MySQL WorkBench που παρουσιάζεται σε επόμενη ενότητα και αποτελεί το κύριο γραφικό περιβάλλον – πελάτη.
- Το mysql (ή mysql.exe) είναι ένα πρόγραμμα γραμμής εντολών (command-line executable). Παρέχει μία γραμμή εντολών για την αλληλεπίδραση ενός χρήστη με έναν εξυπηρετητή ΣΔΒΔ MySQL. Μέσω των εντολών SQL μπορεί ένας χρήστης στο περιβάλλον του command prompt – γραμμής εντολών να πραγματοποιήσει πρακτικά όλες τις ενέργειες που επιτρέπονται και από το αντίστοιχο γραφικό πρόγραμμα μέσα από ένα τερματικό παράθυρο.

```

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 142
Server version: 5.1.32-community-log MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| 2b |
| company |
| db_loyalty |
| dcap |
| employees |
| giannis |
| health |
| joomla-test |
+-----+

```

Εικόνα 25 To MySql command prompt – Γραμμή εντολών πελάτη Mysql

- Άλλα προγράμματα γραμμής εντολών περιλαμβάνουν το mysqladmin για τη διαχείριση του ΣΔΒΔ MySQL, το mysqlcheck για τον έλεγχο της ακεραιότητας των αρχείων της βάσης δεδομένων, το mysqldump για τη δημιουργία αντιγράφων ασφαλείας, το mysqlimport για την εισαγωγή αρχείων δεδομένων από αρχεία κειμένου, το mysqlshow για την παρουσίαση πληροφοριών για τη ΒΔ, και τους πίνακες και το mysqlslap που αποτελεί έναν εξομοιωτή φορτίου προς MySQL server για εξομοίωση πρόσβασης πολλαπλών πελατών σε έναν εξυπηρετητή.

Τα προγράμματα πελάτη της MySQL μπορεί να κληθούν από τη γραμμή εντολών, όπως από Windows console prompt ή από Unix shell prompt (βλ στον τίτλο της παραπάνω εικόνας). Όταν καλείται ένα πρόγραμμα-πελάτη, μπορούν να καθοριστούν επιλογές μετά από το όνομα του προγράμματος, που ελέγχουν τη συμπεριφορά του. Επιλογές μπορούν επίσης να δοθούν σε αρχεία επιλογών. Ορισμένες επιλογές δηλώνουν στον πελάτη πώς να συνδεθεί με τον διακομιστή MySQL. Άλλες επιλογές υποδεικνύουν στο πρόγραμμα ποιες ενέργειες να εκτελέσει. Στο γραφικό περιβάλλον workbench υπάρχει δυνατότητα να ρυθμιστούν αντίστοιχα αυτές οι παράμετροι λειτουργίας. Τα γραφικά προγράμματα-πελάτες αποθηκεύουν τις παραμέτρους σύνδεσης σε δικά τους αρχεία, σε μορφή XML. Για να καθορίσουν τις επιλογές που υποστηρίζονται από ένα πρόγραμμα MySQL, πρέπει να επικαλούνται με την επιλογή --help.

Για να προσδιοριστεί η έκδοση του προγράμματος, χρησιμοποιείται η επιλογή --version. Για παράδειγμα, η ακόλουθη έξοδος από τον mysql πελάτη δείχνει ότι ο πελάτης είναι από MySQL 5.5.16:

```
shell> mysql --version  
mysql Ver 14.14 Distrib 5.5.16, for Win64 (x86)
```



Δεν είναι απαραίτητο τα προγράμματα-πελάτες που τρέχουν να έχουν την ίδια έκδοση με το εξυπηρετητή. Στις περισσότερες περιπτώσεις, τα προγράμματα-πελάτες που είναι είτε παλαιότερα είτε νεότερα από τον εξυπηρετητή μπορούν να συνδεθούν σε αυτόν με επιτυχία (Για τις εκδόσεις 5.x).

Οι επιλογές για τα προγράμματα MySQL έχουν δύο βασικές μορφές: 1) η αναλυτική εκδοχή, που αποτελείται από μια λέξη που προηγείται από διπλή παύλα και 2) η σύντομη εκδοχή που αποτελείται από ένα μεμονωμένο γράμμα το οποίο προηγείται από μία μόνο παύλα. Σε πολλές περιπτώσεις, μια συγκεκριμένη επιλογή έχει τόσο μια μακρά όσο και μία βραχεία μορφή. Για παράδειγμα, για να εμφανιστεί ο αριθμός της έκδοσης ενός προγράμματος, μπορεί να χρησιμοποιηθεί η μακρά εκδοχή *--version* ή η σύντομη εκδοχή *-V*. Στη συνέχεια παρουσιάζονται ορισμένες από τις παραμέτρους της mysql στο command prompt:

Usage: mysql [OPTIONS] [database]

-?, --help Εμφανίζει τη βοήθεια.

-I, --help Ισοδύναμο του **-?**

--auto-vertical-output

Μετατρέπει το αποτέλεσμα σε κάθετη εμφάνιση αν δε χωρά στο πλάτος του παραθύρου

--column-type-info Παρουσίαση των πληροφοριών για τις στήλες πίνακα.

-D, --database=name Το όνομα της ΒΔ που θα χρησιμοποιηθεί.

--default-character-set=name

Ορισμός του προεπιλεγμένου σετ χαρακτήρων.

-E, --vertical Εκτύπωση των γραμμών αποτελέσματα ενός ερωτήματος (rows) κάθετα.

-h, --host=name Σύνδεση στο συγκεκριμένο σύστημα - host.

-H, --html Παρουσίαση όλων των αποτελεσμάτων των εντολών SQL σε μορφή HTML.

-X, --xml Παρουσίαση όλων των αποτελεσμάτων των εντολών SQL σε μορφή XML.

-p, --password[=name]

Κωδικός χρήστη για τη σύνδεση στον εξυπηρετητή (χρήστη της MySQL)

Αν δε δοθεί ζητείται από το χρήστη στην οθόνη

-P, --port=# Αριθμός της πόρτας σύνδεσης (Port number). Χρησιμοποιείται η επιλογή που δηλώνεται στα αρχεία με την ακόλουθη σειρά προτεραιότητας, my.cnf,

\$MYSQL_TCP_PORT, /etc/services, προεπιλεγμένη πόρτα σύνδεση 3306.

--prompt=name Set the mysql prompt to this value.

--tee=name Πρόσθεσε ότι η εντολή – ενέργεια θα πραγματοποιηθεί σε αρχείο

-u, --user=name Όνομα χρήστη για σύνδεση

Τυπική εντολή σύνδεσης στον τοπικό εξυπηρετητή για το διαχειριστή χωρίς κωδικό:

cmd> mysql -u root

Τυπική εντολή σύνδεσης στον τοπικό εξυπηρετητή για το διαχειριστή με κωδικό:

cmd> mysql -u root -p



Θα πρέπει να σημειωθεί ότι η εντολή mysql στο command prompt θα πρέπει να εκτελεστεί είτε από το φάκελο που βρίσκεται το εκτελέσιμο αρχείο (mysql.exe στα windows) είτε να περιλαμβάνεται η διαδρομή που βρίσκεται το αρχείο στις παραμέτρους περιβάλλοντος ώστε να αναγνωρίζεται η εντολή. Η πλήρης εγκατάσταση της MySQL μαζί με το MySQL workbench ρυθμίζει με αυτόματο τρόπο κατάλληλα τις μεταβλητές περιβάλλοντος που απαιτούνται.

(Εφόσον απαιτηθεί χειριστείτε με προσοχή τις οδηγίες ρύθμισης των διαδρομών βλ. <http://dev.mysql.com/doc/refman/5.7/en/mysql-installation-windows-path.html>).

Η MySQL τρέχει σε πολλές διαφορετικές εκδόσεις των Windows, Unix, Linux και, αλλά η επικοινωνία client / server δεν περιορίζεται σε περιβάλλοντα όπου απαιτείται όλοι οι υπολογιστές να έχουν το ίδιο λειτουργικό σύστημα. **Τα προγράμματα-πελάτες μπορεί να συνδεθούν με έναν εξυπηρετητή που εκτελείται στον ίδιο ή σε διαφορετικό host, και ο πελάτης και ο εξυπηρετητής υποδοχής δεν χρειάζεται καν να έχουν το ίδιο λειτουργικό σύστημα.**



3.2.3. MySQL non-Client Utilities

Αυτά είναι προγράμματα που λειτουργούν ανεξάρτητα από τον εξυπηρετητή, δηλαδή δεν απαιτούν τον mysqld να εκτελείται καν. Το mysql_config_editor επιτρέπει την αποθήκευση στοιχείων αυθεντικοποίησης στη ΒΔ σε ένα κρυπτογραφημένο αρχείο .mylogin.cnf. Ένα άλλο εργαλείο, το myisamchk, εκτελεί λειτουργίες ελέγχου πινάκων και επισκευής. Άλλα εργαλεία παρουσιάζονται αναλυτικά στην τεκμηρίωση της MySQL και δεν αναφέρονται εδώ καθώς ξεπερνούν τους σκοπούς του παρόντος προγράμματος:

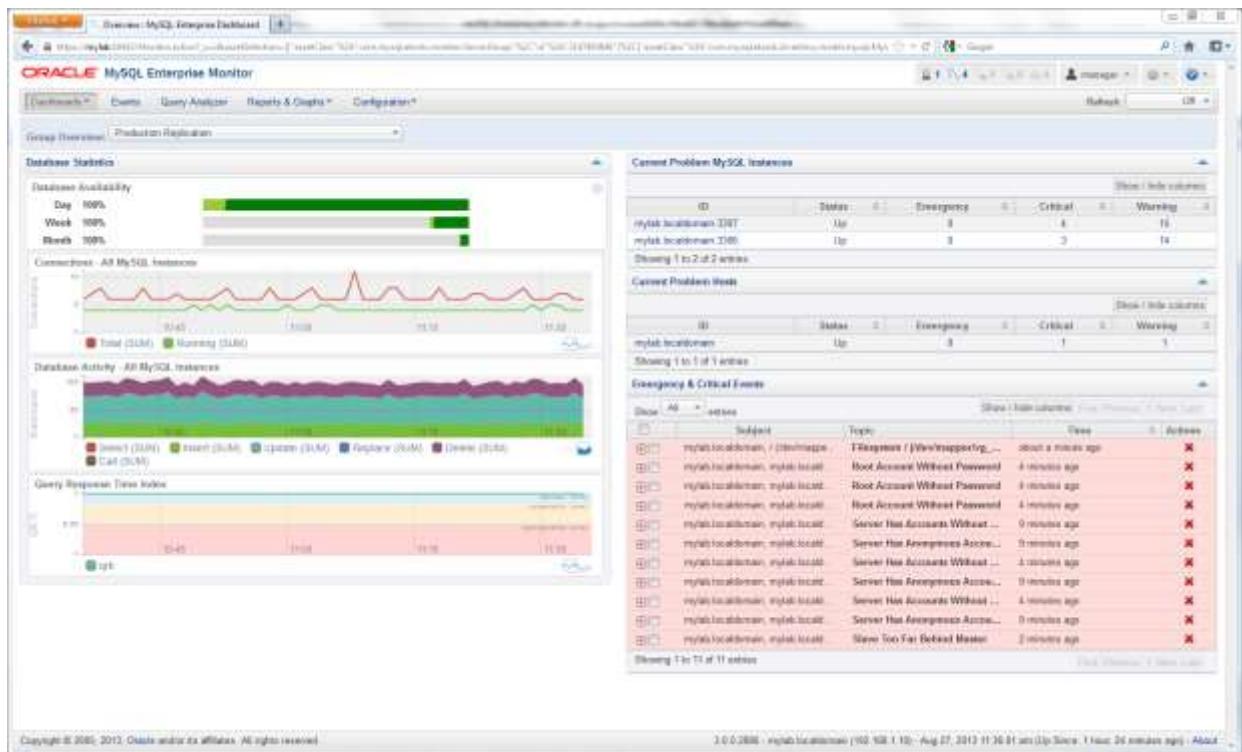
<http://dev.mysql.com/doc/refman/5.7/en/programs-admin-utils.html>

3.3. MySQL Enterprise Monitor

Ο MySQL Enterprise Monitor είναι ένα συνοδευτικό προϊόν του ΣΔΒΔ MySQL της έκδοσης Enterprize και δεν διατίθεται στην έκδοση ανοικτού λογισμικού. Πρόκειται για ένα εργαλείο το οποίο έχει τη δυνατότητα να παρακολουθεί συνεχώς τα στιγμιότυπα της MySQL και μπορεί να χρησιμοποιηθεί για να εντοπιστούν πιθανές προειδοποιήσεις για πιθανά προβλήματα, να βελτιωθούν πιθανώς ερωτήματα SQL ώστε να αποδίδουν καλύτερα, πριν επηρεαστούν βασικά συστήματα ή εφαρμογές. Παρέχει ένα σύνολο ειδικών συμβούλων MySQL που παρέχουν πληροφορίες και λεπτομερείς οδηγίες σχετικά με διορθώσεις και ρυθμίσεις για διαμορφώσεις της MySQL και μεταβλητές, με στόχο τη βελτίωση της ασφάλειας, απόδοσης και διαθεσιμότητας του συστήματος.

Ο MySQL Enterprise Monitor μπορεί να παρακολουθεί, από ένα μόνο στιγμιότυπο MySQL που είναι σημαντικό για μία επιχείρηση, μέχρι και ένα τεράστιο σύστημα μηχανημάτων με εξυπηρετητές βάσεων δεδομένων, που υποστηρίζει ένα πολυσύχναστο portal.

Στη συνέχεια περιγράφονται τα βασικά στοιχεία που συνθέτουν το προϊόν MySQL Enterprise Monitor. Αυτά τα στοιχεία μπορούν να εγκατασταθούν σε διάφορες διαμορφώσεις, ανάλογα με τη βάση δεδομένων και την τοπολογία του δικτύου, για να δώσουν τον καλύτερο συνδυασμό αξιόπιστων και ευέλικτων δεδομένων παρακολούθησης, με την ελάχιστη επιβάρυνση για τα μηχανήματα εξυπηρετητή της βάσης δεδομένων.

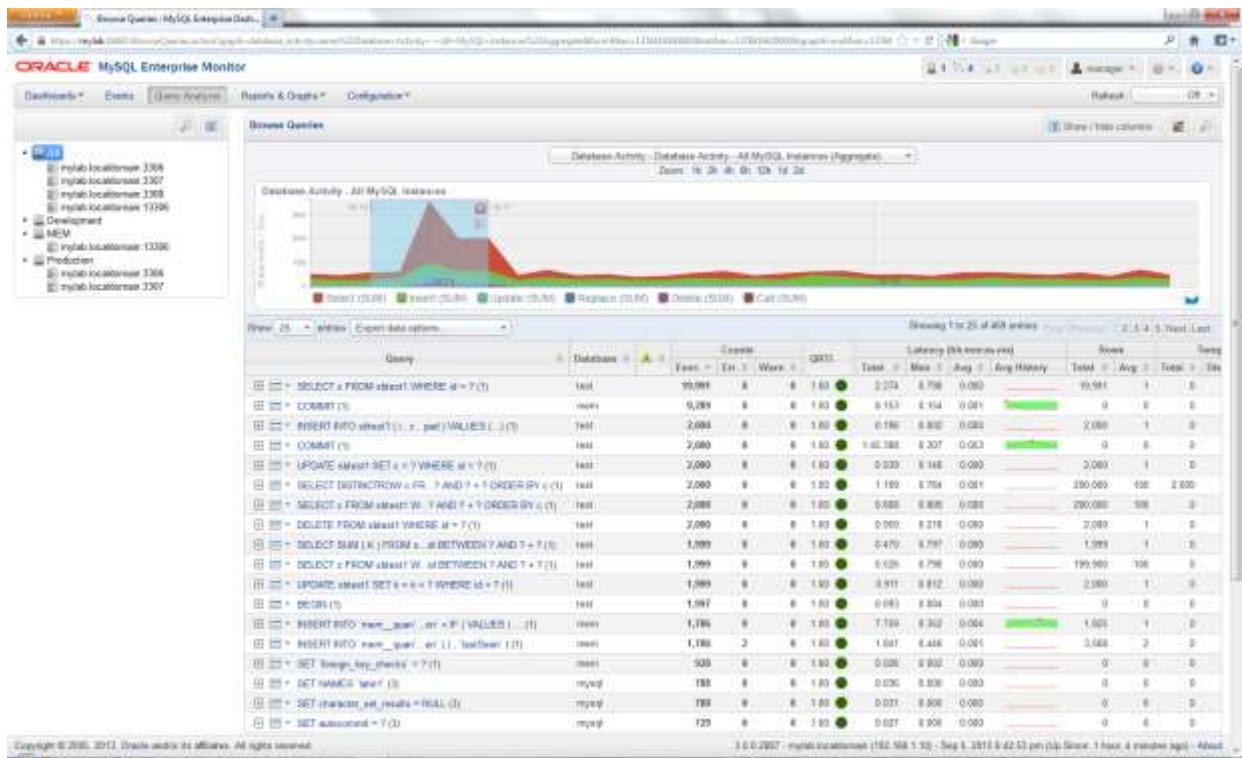


Εικόνα 26 To MySql Dashboard .

Μια τυπική εγκατάσταση MySQL Enterprise Monitor αποτελείται από:

- Ένα ή περισσότερα MySQL στιγμιότυπα προς παρακολούθηση. Ο MySQL Enterprise Monitor μπορεί να παρακολουθήσει τόσο Community όσο και Enterprise εκδόσεις του MySQL εξυπηρετητή.
- Έναν MySQL Enterprise Monitor Agent για κάθε στιγμιότυπο MySQL που παρακολουθείται.
- Έναν ενιαίο MySQL Enterprise Service Manager, ο οποίος συγκεντρώνει πληροφορίες από τους Agents και παρέχει τη διεπαφή χρήστη στα δεδομένα που συλλέγονται μέσω του Dashboard.

Ο Query Analyzer παρέχει επίσης πολλαπλές μετρικές σχετικά με την απόδοση ενός ερωτήματος SQL βοηθώντας τον προγραμματιστή και το διαχειριστή να εντοπίσουν ερωτήματα που είτε ήδη προκαλούν καθυστερήσεις είτε αναμένεται να προκαλέσουν δυσλειτουργίες υπό συνθήκες. Καταγράφει στατιστικά στοιχεία εκτέλεσης των ερωτημάτων χωρίς να εμπλέκει την καταγραφή του συστήματος της MySQL (SQL Query log) κρατώντας όλα τα ιστορικά δεδομένα. Είναι δυνατό να παρουσιάζει τα πιο «βαριά» ερωτήματα που προκαλούν καθυστερήσεις είτε λόγω της απαιτητικής εκτέλεσής τους είτε γιατί επαναλαμβάνονται πολύ συχνά. Συνδυάζεται με γραφικές παραστάσεις στο MySQL Enterprise Monitor Graphs.



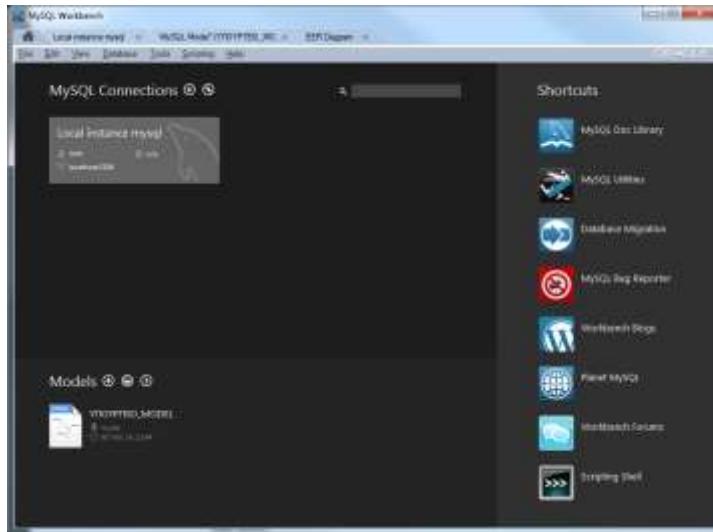
Εικόνα 27 To Query Analyzer.

3.4.MySQL Workbench

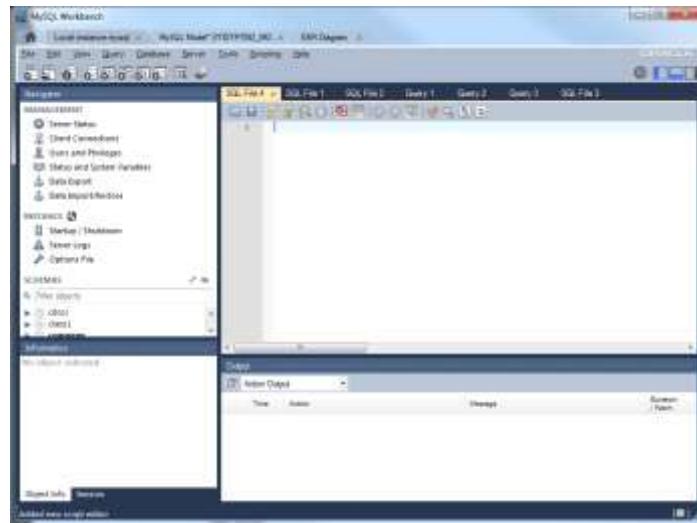
To MySQL Workbench παρέχει τη δυνατότητα μοντελοποίησης δεδομένων βασισμένη σε GUI (Graphical User Interface), SQL ανάπτυξη, εξέλιξη, μεταφορά δεδομένων και ολοκληρωμένα εργαλεία διαχείρισης (διαμόρφωση του εξυπηρετητή, διαχείριση χρηστών, χειρισμός αντικειμένου) για τους αρχιτέκτονες της βάσης δεδομένων, τους προγραμματιστές και οι διαχειριστές. Κατά τη διάρκεια του παρόντος εγχειριδίου παρουσιάζονται οι διάφορες δυνατότητες που δίνει το εργαλείο σε αντιστοιχία με τις έννοιες που περιγράφονται στην αντίστοιχη ενότητα.

To *MySQL WorkBench* αποτελεί την κύρια γραφική διεπαφή χειρισμού του ΣΔΒΔ και των δεδομένων των ΒΔ. Είναι συμβατό πλήρως από την έκδοση MySQL 5.1 και μετά. Ωστόσο είναι συμβατό και με την έκδοση 5.0. Δεν υποστηρίζει προηγούμενες εκδόσεις. Περιλαμβάνει λειτουργικότητα για ανάπτυξη SQL προγραμμάτων. Επιτρέπει τη δημιουργία και διαχείριση συνδέσεων σε εξυπηρετητές MySQL με διαφορετικές παραμέτρους. Ενσωματώνει εργαλείο συγγραφής προγραμμάτων SQL (SQL Editor), σε αντίθεση με τις παλαιότερες εκδόσεις όπου το αντίστοιχο πρόγραμμα ήταν ο Query Browser και παρεχόταν ως χωριστή εφαρμογή. Επιπλέον το περιβάλλον Workbench επιτρέπει τη γραφική απεικόνιση του σχήματος μια ΒΔ. Έχει μάλιστα δυνατότητες να δημιουργεί με αυτοματοποιημένο τρόπο μία ΒΔ από το σχεσιακό

μοντέλο της αλλά και το αντίστροφο δηλαδή να δημιουργεί το σχεσιακό μοντέλο μιας υπάρχουσας ΒΔ. Επιτρέπει μέσω του εργαλείου επεξεργασίας Πινάκων να ρυθμίζονται μέσω γραφικού περιβάλλοντος όλες οι παράμετροι πινάκων, στηλών, ευρετηρίων, σκανδάλων, partitions, δικαιωμάτων, όψεων, ρουτινών κ.α. Η λειτουργικότητα του εργαλείου workbench παρουσιάζεται αναλυτικότερα στη συνέχεια του παρόντος.



Εικόνα 28 To MySql WorkBench – Κεντρική Οθόνη.



Εικόνα 29 To MySql WorkBench – SQL Editor – Επεξεργαστής Εντολών SQL.

3.5. MySQL Connectors

Οι MySQL Connectors παρέχουν συνδεσιμότητα με τον MySQL εξυπηρετητή για τα προγράμματα-πελάτες. Πρόκειται για λογισμικό – οδηγούς, τα οποία ενεργούν ως γέφυρες προς τον MySQL εξυπηρετητή για τα προγράμματα-πελάτες. Είναι διαθέσιμοι ως ξεχωριστά πακέτα:

- ADO.NET Driver for MySQL (Connector/.NET)
- ODBC Driver for MySQL (Connector/ODBC)

- JDBC Driver for MySQL (Connector/J)
- Python Driver for MySQL (Connector/Python)
- C++ Driver for MySQL (Connector/C++)
- C Driver for MySQL (Connector/C)
- C API for MySQL (mysqlclient)
- MySQL Native Driver for PHP

Επιπλέον υπάρχουν οδηγοί που έχουν αναπτυχθεί από τρίτους. Λίγα λόγια για κάποιους από τους πιο δημοφιλείς connects και οδηγούς:

- Ο MySQL Connector / ODBC παρέχει ένα πρόγραμμα οδήγησης MySQL προδιαγραφών για ODBC. Επιτρέπει σε προγράμματα συμβατά με ODBC να αποκτήσουν πρόσβαση στη MySQL .
- Ο MySQL Connector / J είναι ένα πρόγραμμα οδήγησης JDBC, για χρήση σε προγράμματα Java. Επιτρέπει σε JDBC συμβατά προγράμματα να αποκτήσουν πρόσβαση στη MySQL .
- Ο MySQL Connector / NET είναι ένα πρόγραμμα οδήγησης γραμμένο σε C# που υποστηρίζει τις διεπαφές ADO.NET που απαιτούνται για να τρέξουν .NET εφαρμογές που έχουν πρόσβαση στη MySQL .
- PHP Drivers για MySQL είναι προγράμματα οδήγησης που έχουν αναπτυχθεί από τρίτους προγραμματιστές της κοινότητας MySQL για να προσφέρουν διασύνδεση σε βάσεις δεδομένων μέσα από προγράμματα γραμμέτα στη γλώσσα PHP. Το πιο δημοφιλές είναι το mysqli (όπως και το παλαιότερο mysql).

Οι MySQL Connectors είναι διαθέσιμοι για Windows και Unix. Για να χρησιμοποιηθεί ένας Connector, θα πρέπει να εγκατασταθεί σε πελάτη- εξυπηρετητή. Δεν είναι απαραίτητο ο εξυπηρετητής να εκτελείται στον ίδιο μηχάνημα ή να τρέχει το ίδιο λειτουργικό σύστημα με τον πελάτη. Αυτό σημαίνει ότι οι MySQL Connectors είναι πολύ χρήσιμοι για την παροχή συνδεσιμότητα MySQL σε ετερογενή περιβάλλοντα. Για παράδειγμα, όσοι χρησιμοποιούν Windows μπορεί να τρέξουν εφαρμογές που απαιτούν πρόσβαση σε βάσεις δεδομένων MySQL που βρίσκονται σε έναν εξυπηρετητή που φιλοξενείτε σε λειτουργό σύστημα Linux.

Ο MySQL Connector / ODBC λειτουργεί ως γέφυρα μεταξύ του εξυπηρετητή MySQL και των προγραμμάτων-πελατών που χρησιμοποιούν το πρότυπο Open DataBase Connection. Παρέχει ένα πρόγραμμα οδήγησης MySQL προδιαγραφών σύμφωνων με το ODBC, έτσι ώστε οι πελάτες που βασίζονται σε ODBC να μπορούν να έχουν πρόσβαση σε βάσεις

δεδομένων MySQL. Ο MySQL Connector / ODBC χρησιμοποιεί τη βιβλιοθήκη C της MySQL για να υλοποιήσει το πρωτόκολλο επικοινωνίας πελάτη - εξυπηρετητή. Ο MySQL Connector / ODBC είναι διαθέσιμος για Windows και Unix. Ο συγκεκριμένος οδηγός χρησιμοποιείται κυρίως από παλαιότερο λογισμικό και αποτελεί την πλέον ανοικτή ως προς συμβατότητα λύση. Στο σημείο αυτό πρέπει να σημειώσουμε ότι κάθε οδηγός δεν προσφέρει με ασφάλεια το 100% των λειτουργιών της ΒΔ και θα πρέπει να μελετά κανείς την τεκμηρίωσή του για να αντιμετωπίσει τυχόν διαφοροποιήσεις.

Ο MySQL Connector / J χρησιμοποιείται σε προγράμματα Java που απαιτούν JDBC. Είναι γραμμένος σε Java και υλοποιεί το πρωτόκολλο επικοινωνίας client / server άμεσα. Ο MySQL Connector / J είναι οδηγός Τύπου 4 (καθαρή Java) που υλοποιεί την έκδοση 3.0 της προδιαγραφή JDBC. Ο MySQL Connector / J περιλαμβάνει υποστήριξη για δυνατότητες της MySQL, όπως δηλώσεις που κατασκευάζονται στην πλευρά του εξυπηρετητή, αποθηκευμένες ρουτίνες (stored routines) και Unicode. Ο MySQL Connector / J είναι διαθέσιμος για Windows και Unix.

Ο MySQL Connector / NET επιτρέπει σε .NET εφαρμογές να χρησιμοποιούν MySQL. Είναι γραμμένος σε C# και υλοποιεί το πρωτόκολλο επικοινωνίας client / server άμεσα. Ο MySQL Connector / NET περιλαμβάνει υποστήριξη για δυνατότητες MySQL δηλώσεις που κατασκευάζονται στην πλευρά του εξυπηρετητή, αποθηκευμένες ρουτίνες (stored routines) και Unicode.

Ο οδηγός MySQL Native Driver αντικαθιστά τη βιβλιοθήκη MySQL Client Library (libmysqlclient). Περιλαμβάνεται στις εκδόσεις PHP 5.3.0+. Για τη σύνδεση από PHP σε ΒΔ MySQL χρησιμοποιούνται οι επεκτάσεις mysqli και PDO MYSQL. Παλαιότερα οι επεκτάσεις αυτές βασίζονταν στη βιβλιοθήκη libmysqlclient. Πλέον ο MySQL Native Driver αποτελεί μια εναλλακτική ως PHP επέκταση και είναι γραμμένος σε γλώσσα προγραμματισμού C.

4. Ορισμός Δεδομένων

Οι επιμορφωνόμενοι θα είναι σε θέση να:

- δημιουργούν μία ΒΔ με τα επιθυμητά χαρακτηριστικά καθώς και να την προσπελαύνουν, τροποποιούν και να τη διαγράφουν.

4.1. Ιδιότητες ΒΔ (Database Properties)

Ο MySQL Server διαχειρίζεται Βάσεις Δεδομένων και δεδομένα εκτελώντας αποθήκευση, ανάκτηση και χειρισμό εγγραφών δεδομένων. Οι εγγραφές οργανώνονται σε πίνακες και οι πίνακες οργανώνονται σε βάσεις δεδομένων κατ' αντιστοιχία με την μοντελοποίηση όπου πλειάδες συνθέτουν σχέσεις και με τη σειρά τους οι σχέσεις ένα σύνολο Βάσης Δεδομένων. Στην MySQL, οι βάσεις δεδομένων αποθηκεύονται σε μια κοινή θέση που ονομάζεται κατάλογος δεδομένων (data directory). Κάθε εξυπηρετητής MySQL έχει έναν κατάλογο δεδομένων και κατά συνέπεια αν κανείς θελήσει να διαθέτει πολλαπλούς εξυπηρετητές στην ίδια μηχανή φιλοξενίας μπορεί να το κάνει σχεδιάζοντας πολλαπλά στιγμιότυπα με αντίστοιχο κατάλογο δεδομένων. Στη μηχανή αποθήκευσης InnoDB κάθε ΒΔ είναι ένα διαφορετικό αρχείο στον κατάλογο δεδομένων, ενώ στο MyISAM κάθε βάση δεδομένων αποτελείται από έναν υποκατάλογο του καταλόγου δεδομένων. Αυτός ο υποκατάλογος ονομάζεται "κατάλογος της βάσης δεδομένων". Το αρχείο της ΒΔ έχει το ίδιο όνομα με τον τίτλο της ΒΔ για να είναι εύκολο να εντοπιστεί τόσο εντός ΣΔΒΔ όσο και στο σύστημα αρχείων.

4.2. Καλές πρακτικές Σχεδιασμού (Good Design Practices)

Για τον καλό σχεδιασμό μιας βάσης δεδομένων ακολουθούν μερικές συμβουλές-κατευθύνσεις που μπορούν να φανούν χρήσιμες:

- διαβάθμιση του υλικού (hardware) ανάλογα με τις ανάγκες
- κανονικοποίηση των συσχετίσεων (normalization) και σωστών τύπων δεδομένων (δηλαδή να χρησιμοποιούνται μικροί ακέραιοι-int αντί μεγάλων όπου είναι δυνατό κ.α.)
- προσπάθεια αποφυγής του NULL αν είναι δυνατό
- χρήση varchar αντί για κείμενο text/blob, όπου είναι δυνατό
- δεικτοδότηση των πινάκων (ευρετήρια)

- σχεδιασμός ερωτημάτων με σωστό τρόπο (use indexes)
- χρησιμοποίησης συναλλαγών

Μετά το σχεδιασμό και την ανάπτυξη της βάσης δεδομένων, αν η απόδοση δεν είναι επαρκής σκεφτόμαστε τη βελτιστοποίηση:

- έλεγχος επεξηγηματικών σχεδίων και συντονισμός των SQLs
- έλεγχος της χρησιμοποίησης του υλικού και συντονισμός είτε του συστήματος είτε των παραμέτρων mysql (δηλαδή query cache).

Η κανονικοποίηση είναι μια διαδικασία αποδοτικής οργάνωσης των δεδομένων μιας ΒΔ. Κυρίως στοχεύει στην αποθήκευση της πληροφορίας κατά το δυνατό μόνο μία φορά και της αποθήκευση της πληροφορίας σε πίνακες που να έχουν σημασιολογική συσχέτιση, δηλαδή να περιλαμβάνονται μόνο σχετιζόμενα γνωρίσματα σε έναν πίνακα. Οι στόχοι αυτοί σκοπό έχουν τελικώς να ελαχιστοποιείται ο όγκος της ΒΔ και η οργάνωση της παρουσίασης των δεδομένων.

Για την οργάνωση των κατευθύνσεων οργάνωσης έχουν δημιουργηθεί κατηγορίες που ονομάζονται κανονικές μορφές και είναι πέντε σε αριθμό. Στην πράξη χρησιμοποιούνται κυρίως οι τρεις από αυτές, οπότε θα αναφέρουμε τις πλέον συχνά εμφανιζόμενες.

Πρέπει να σημειωθεί ότι οι κατευθύνσεις αυτές υπάρχουν φορές που κάνουν πολύπλοκή την οργάνωση των δεδομένων υπό το πρίσμα συγκεκριμένων επιχειρηματικών κανόνων – απαιτήσεων της εφαρμογής που πρέπει να καλυφθούν. Για το λόγο αυτό υπάρχουν περιπτώσεις όπου θα πρέπει να γίνεται με προσοχή η χρήση τους.

Η πρώτη κανονική μορφή απαιτεί να μην υπάρχουν διπλές φορές τα ίδια δεδομένα σε μία στήλη και κάθε εγγραφή να μπορεί να προσδιοριστεί από ένα πρωτεύον κλειδί με μοναδικό τρόπο. Για το λόγο αυτό τα συσχετιζόμενα δεδομένα οργανώνονται σε πίνακες.

Παράδειγμα

Για παράδειγμα:

ΥΠΑΛΛΗΛΟΣ_MH_KANONIKOPOIHMENOΣ1

ΥΠΑΛΛΗΛΟΣ

<u>ΑΔΤ</u>	Όνοματεπώνυμο	Όνομα_Υπηρεσίας
1324	Χρήστος Ιωάννου	Τμήμα Πληροφορικής

4343	Μάριος Λέκκας	Τμήμα Πληροφορικής
4345	Μαρίνα Γιώτη	Τμήμα Σχεδιασμού
4346	Ρέα Δόρη	Τμήμα Σχεδιασμού

Καθώς η στήλη όνομα_υπηρεσίας περιλαμβάνει επαναλαμβανόμενες φορές το όνομα της υπηρεσίας παραβιάζει τον κανόνα της πρώτης κανονικής μορφής. Επομένως στην ακόλουθη μορφή έχουμε κανονικοποίηση καθώς δεν επαναλαμβάνεται πληροφορία σε κάποια στήλη και κάθε γραμμή έχει πρωτεύον κλειδί. Στη σχέση το κλειδί είναι ο συνδυασμός των γνωρισμάτων ΑΔΤ_Αρ_Υπηρεσίας.

ΥΠΑΛΛΗΛΟΣ

<u>ΑΔΤ</u>	Όνοματεπώνυμο
1324	Χρήστος Ιωάννου
4343	Μάριος Λέκκας
4345	Μαρίνα Γιώτη

ΑΝΗΚΕΙ

<u>ΑΔΤ</u>	<u>Αρ_Υπηρεσίας</u>	Όνομα_Υπηρεσίας
1324	AB123	Τμήμα Σχεδιασμού
4343	AB123	Τμήμα Σχεδιασμού
4343	AB115	Τμήμα Πληροφορικής
1324	AB115	Τμήμα Πληροφορικής
...

Πίνακες ΒΔ 1^{ης} κανονικής μορφής

Η δεύτερη κανονική μορφή θα πρέπει να καλύπτει τις απαιτήσεις της πρώτης κανονικής μορφής και επιπλέον δεν πρέπει να υπάρχει μερική εξάρτηση οποιασδήποτε στήλης σε πρωτεύον κλειδί. Αυτό απαιτεί την απαλοιφή όλων των υποσυνόλων που εμφανίζονται σε πολλαπλές εγγραφές στον πίνακα και τη μεταφορά τους σε χωριστούς πίνακες με αναφορά ξένου κλειδιού.

Παράδειγμα

Για παράδειγμα
ΥΠΑΛΛΗΛΟΣ

<u>ΑΔΤ</u>	Ονοματεπώνυμο	Φύλο
1324	Χρήστος Ιωάννου	Άρρεν
4343	Μάριος Λέκκας	Άρρεν
4345	Μαρίνα Γιώτη	Θήλυ

ANHKEI

<u>ΑΔΤ</u>	<u>Αρ_Υπηρεσίας</u>	Όνομα_Υπηρεσίας
1324	AB123	Τμήμα Σχεδιασμού
4343	AB123	Τμήμα Σχεδιασμού
4343	AB115	Τμήμα Πληροφορικής
1324	AB115	Τμήμα Πληροφορικής
...

Πίνακες ΒΔ 1^{ης} κανονικής μορφής

Θα μετατραπούν για τη δεύτερη μορφή κανονικοποίησης ως εξής με χρήση ξένων κλειδιών.

ΥΠΑΛΛΗΛΟΣ

<u>ΑΔΤ</u>	Ονοματεπώνυμο	Κωδ_Φύλο
1324	Χρήστος Ιωάννου	1
4343	Μάριος Λέκκας	1
4345	Μαρίνα Γιώτη	0

ΥΠΗΡΕΣΙΑ

<u>Αρ_Υπηρεσίας</u>	ΌνομαΥ
AB123	Τμήμα Σχεδιασμού
AB115	Τμήμα Πληροφορικής

ANHKEI

<u>ΑΔΤ</u>	<u>Αρ_Υπηρεσίας</u>
1324	AB123
4343	AB123
4343	AB115
1324	AB115
...	...

Φύλο

<u>Κωδ</u>	<u>Περιγραφή</u>
1	Άρρενα

0	Θήλυ
---	------

Πίνακες ΒΔ 2^{ης} κανονικοποίησης

Ο τρίτος βαθμός κανονικοποίησης προχωρά έτσι ώστε να περικλείει όλες τις απαιτήσεις της 2^{ης} μορφής κανονικοποίησης και να αφαιρεί τις στήλες που δεν εξαρτώνται από το πρωτεύον κλειδί. Στόχος είναι η μείωση των δεδομένων που παρουσιάζονται διπλές ή πολλαπλές φορές και η ακεραιότητα των δεδομένων.

Για παράδειγμα θεωρείστε τον ακόλουθο πίνακα με γνωρίσματα ως εξής:

ΠΑΡΑΓΓΕΛΙΕΣ

Κωδ_Παραγγελίας	Κωδ_Πελάτη	Τιμή	Ποσότητα	Σύνολο
-----------------	------------	------	----------	--------

Παρατηρούμε ότι το σύνολο μπορεί να υπολογιστεί από την τιμή και την ποσότητα και κατά συνέπεια δεν εξαρτάται από το πρωτεύον κλειδί. Για το λόγο αυτό μπορεί να αφαιρεθεί ώστε να έχουμε ολοκληρωμένη κανονικοποίηση κατά την 3^η μορφή.

ΠΑΡΑΓΓΕΛΙΕΣ 3NF

Κωδ_Παραγγελίας	Κωδ_Πελάτη	Τιμή	Ποσότητα
-----------------	------------	------	----------

4.3.Αναγνωριστικά ΒΔ (Identifiers)

Αναγνωριστικά ονομάζουμε τα ονόματα που διαθέτουν τα αντικείμενα ενός ΣΔΒΔ για να τα διαχωρίζουμε και να τα προσδιορίζουμε. Επομένως μπορούν να χρησιμοποιηθούν ονόματα που να αναφέρονται σε βάσεις δεδομένων, πίνακες, αποθηκευμένες ρουτίνες, όψεις, σκανδάλες, στήλες, ευρετήρια κ.α.. Είναι δυνατό να δημιουργηθούν ψευδώνυμα, τα οποία δρουν ως συνώνυμα για τα ονόματα των πινάκων και των στηλών τα οποία συνήθως είναι συντομογραφίες του πλήρους ονόματος. Τα αναγνωριστικά μπορεί να περικλείονται σε εισαγωγικά (μονό ανάποδο εισαγωγικό «`» ή αν ενεργοποιηθεί η κατάσταση συμβατή με ANSI_QUOTES SQL mode τότε μπορούν να χρησιμοποιηθούν τα διπλά αγγλικά εισαγωγικά «"»). Αν δεν περικλείονται από εισαγωγικά τότε θα πρέπει να τηρούν τους ακόλουθους περιορισμούς όπως να περιλαμβάνουν αλφαριθμητικούς χαρακτήρες, την κάτω παύλα (‘_’) και το σύμβολο του δολλαρίου. Κάθε αναγνωριστικό δε μπορεί να περιλαμβάνει μόνο αριθμούς.

Ένα ψευδώνυμο μπορεί να περιλαμβάνει οποιοδήποτε χαρακτήρα εφόσον περικλείεται με εισαγωγικά «"», «"» ή «`».

Τα ονόματα των ΒΔ και των πινάκων είναι εξαρτώμενα από πεζά-κεφαλαία και επηρεάζονται από την παράμετρο του συστήματος lower_case_table_names. Αν η παράμετρος τεθεί στην τιμή 1 ή 2 πριν από τη δημιουργία της ΒΔ και των πινάκων τότε τα ονόματα δεν εξαρτώνται από πεζά-κεφαλαία. Τα ονόματα των στηλών, ευρετηρίων, αποθηκευμένων ρουτινών, σκανδάλων και ψευδωνύμων δεν εξαρτώνται από τα πεζά-κεφαλαία.

Ένα επιπλέον χαρακτηριστικό παράδειγμα αναγνωριστικών είναι η πλήρης μορφή τους όπου μαζί με το όνομά τους προηγείται το όνομα ενός υψηλότερου επιπέδου αντικειμένου χωριζόμενο από μία τελεία. Οι πλήρεις μορφές χρησιμοποιούνται για να συγκεκριμενοποιήσουμε το αντικείμενο που θα χρησιμοποιηθεί είτε λόγω συνωνυμίας είτε απλώς για ακρίβεια. Για παράδειγμα μπορούμε να συγκεκριμενοποιήσουμε τον πίνακα που θα χρησιμοποιηθεί προσδιορίζοντας και το όνομα της ΒΔ στην οποία ανήκει. Αντίστοιχα μπορεί να προσδιοριστεί με ακρίβεια η στήλη που θα χρησιμοποιηθεί περιλαμβάνοντας την πλήρη ονομασία μαζί με το όνομα του πίνακα στον οποίο ανήκει. Μάλιστα πρέπει να σημειωθεί ότι στην περίπτωση αυτή είναι δυνατό να χρησιμοποιηθεί περαιτέρω εξειδίκευση χρησιμοποιώντας και το όνομα της ΒΔ. Κατά συνέπεια στο παρακάτω παράδειγμα οι εντολές είναι ισοδύναμες:

```
SELECT `Όνοματεπώνυμο` FROM `ΥΠΑΛΛΗΛΟΣ`;
SELECT `ΥΠΑΛΛΗΛΟΣ`.`Όνοματεπώνυμο` FROM `ΥΠΑΛΛΗΛΟΣ`;
SELECT           `MyDB`.`ΥΠΑΛΛΗΛΟΣ`.`Όνοματεπώνυμο`           FROM
`MyDB`.`ΥΠΑΛΛΗΛΟΣ`;
```

4.4. Δημιουργία ΒΔ (Creating Databases)

Για να δημιουργηθεί μια νέα βάση δεδομένων, χρησιμοποιείται η εντολή CREATE DATABASE. Η ακόλουθη δήλωση δημιουργεί μια βάση δεδομένων που ονομάζεται 'υπουργείο':

```
CREATE DATABASE `υπουργείο`;
```

Η εντολή SQL CREATE DATABASE είναι δυνατό να ακολουθείται από τις ακόλουθες παραμέτρους:

- CHARACTER SET που επιτρέπει τη δήλωση του προεπιλεγόμενου σετ χαρακτήρων για τη ΒΔ,
- COLLATE που επιτρέπει τη δήλωση της προεπιλεγμένης ταξινόμησης χαρακτήρων για τη ΒΔ

Όταν εκτελούμε εντολές SQL είτε στο command prompt είτε στο workbench αυτές εκτελούνται στην προεπιλεγμένη ΒΔ κατά τη στιγμή της εκτέλεσης εκτός αν αναφέρεται η πλήρης ονομασία-αναγνωριστικό του αντικειμένου που συμμετάσχει στην εντολή. Για να χρησιμοποιήσουμε μία νέα ΒΔ που μόλις δημιουργήσαμε και να είναι η προεπιλεγμένη για τις επακόλουθες εντολές SQL που θα αποστείλουμε για εκτέλεση στο ΣΔΒΔ πρέπει να χρησιμοποιήσουμε την εντολή use ως εξής:

```
USE `υπουργείο`;
```

Είναι σημαντικό όταν εργαζόμαστε με εντολές SQL να έχουμε κατά νου σε ποια ΒΔ εργαζόμαστε κάθε στιγμή.

4.5.Τροποποίηση ΒΔ (Altering Databases)

Η εντολή ALTER DATABASE τροποποιεί τις επιλογές για μία υπάρχουσα βάση δεδομένων. Οι επιτρεπόμενες επιλογές είναι οι ίδιες με αυτές της εντολής CREATE DATABASE, δηλαδή το CHARACTER SET και COLLATE. Η ακόλουθη δήλωση αλλάζει το προκαθορισμένο collation της βάσης δεδομένων `υπουργείο` σε utf8_general_ci:

```
ALTER DATABASE `υπουργείο` COLLATE utf8_general_ci;
```

4.6.Διαγραφή ΒΔ (Dropping Databases)

Όταν μια βάση δεδομένων δεν είναι πλέον απαραίτητη, μπορεί να αφαιρεθεί με την εντολή DROP DATABASE:

```
DROP DATABASE `υπουργείο`;
```

Αν η βάση δεδομένων δεν υπάρχει, αυτό είναι ένα σφάλμα. Για να προκαλέσουμε μια προειδοποίηση αντί για σφάλμα, μπορούμε να συμπεριλάβουμε μια πρόταση IF

EXISTS:

```
DROP DATABASE IF EXISTS `υπουργείο`;
```

Η εντολή IF EXISTS μπορεί να χρησιμοποιηθεί επίσης και κατά τη δημιουργία αντικειμένων όπως ΒΔ και πινάκων, αλλά και για την κατάργηση πινάκων.

4.7. Επισκόπηση τύπων δεδομένων

Στην ενότητα αυτή γίνεται επισκόπηση των παρακάτω τύπων δεδομένων: Αριθμητικοί (Numeric Data Types), Χαρακτήρων (Character String Data Types), Διάδικοι (Binary String Data Types), Χρονικοί (Temporal Data Types), Χωρικοί (Spatial Data Types), και η έννοια των NULL τιμών.

Οι αριθμητικές τιμές οργανώνονται σε έναν αριθμό από τύπους δεδομένων που περιλαμβάνουν ακεραίους (integer), αριθμούς με δεκαδικά ψηφία (decimal) – χρήσιμα για τιμές-χρηματικά ποσά - ή αριθμούς κινητής υποδιαστολής (float ή double), αριθμούς με ή χωρίς πρόσημο (unsigned) και δυαδικούς αριθμούς (bit). Κάθε τύπος αριθμητικών τιμών είναι δυνατό να ορίζεται σε συγκεκριμένο πεδίο το οποίο με τη σειρά του ουσιαστικά ορίζει και τον χώρο που καταλαμβάνει στη ΒΔ π.χ. ο Int απαιτεί τέσσερα byte ενώ ο bigint πολύ περισσότερα με πολύ μεγαλύτερο βεβαίως εύρος τιμών. Επιπλέον, κάθε τύπος τιμών επιτρέπει διαφορετική ακρίβεια αλλά και παρουσιάζει διαφορετικό αριθμό ψηφίων.

Οι τιμές συμβολοσειρών (string) περιλαμβάνουν περιπτώσεις τύπων για χαρακτήρες ή σειρά bytes που αντιπροσωπεύουν χαρακτήρες ή δεδομένα. Οι συμβολοσειρές είναι δυνατό να είναι εξαρτώμενες από πεζά-κεφαλαία (case sensitive) ή όχι (case insensitive) και αντίστοιχα τα collation τους καταλήγουν σε _ci, ή _cs. Οι συμβολοσειρές δηλώνονται εντός μονών εισαγωγικών «» και μπορεί να είναι μεταβλητού μήκους (varchar). Ο πλέον δημοφιλής τύπος συμβολοσειρών είναι η μεταβλητή string τύπου varchar με έως 64K χαρακτήρες, και η blob που αποτελεί τύπο μεταβλητού μήκους δυαδικών δεδομένων για την αποθήκευση δεδομένων και αρχείων. Άλλοι τύποι περιλαμβάνουν σταθερού μήκους χαρακτήρες char αλλά και μεγάλου μήκους χαρακτήρες τύπου text που όμως δεν μπορούν να περιληφθούν σε ευρετηριοποίηση. Σημειώνουμε ότι οι δυαδικές μορφές/τύποι string περιλαμβάνουν μόνο bytes και απλώς διαφέρουν στο μέγεθος του string που μπορούν να

αποθηκεύσουν.

Οι τιμές ημερομηνίας και οι χρονικές τιμές γενικότερα αποτελούν τύπους της MySQL (date, time, datetime) με ημερομηνία, χρόνο, ή και τα δύο μαζί. Οι ημερομηνίες γενικότερα μπορούν να γράφονται εντός μονών εισαγωγικών «'». Η μορφοποίηση των τύπων χρόνου περιλαμβάνει διακριτοποίηση του έτους (YYYY), μήνα (MM), ημέρας (DD), ώρας (hh), λεπτού (mm), και δευτερολέπτου (ss). Οι ημερομηνίες υποστηρίζουν το διάστημα 1000-01-01 έως 9999-12-31 ενώ οι χρονικοί τύποι από -838:59:59 έως 839:59:59 καθώς είναι δυνατό να καταγράφουν χρονικές περιόδους πέραν του 24ώρου. Ο τύπος datetime ακολουθεί μορφοποίηση 'YYYY-MM-DD hh:mm:ss'. Υπάρχει επίσης ο ειδικός τύπος Timestamp που αναπαριστά τα δευτερόλεπτα που έχουν περάσει από την 1-1-1970 σε ώρα Συγχρονισμένου Παγκόσμιου Χρόνου (Coordinated Universal Time, UTC), ενώ την προσαρμόζει στην ώρα της ζώνης του υπολογιστή που φιλοξενεί το ΣΔΒΔ.

4.8.Περιγραφή Πινάκων (Describe Table)

Είναι σημαντικό να γίνει κατανοητό πώς ερμηνεύεται η έξοδος της εντολής DESCRIBE <όνομα_πίνακα>. Όταν χρειαζόμαστε να ξέρουμε τη δομή ενός πίνακα, μπορούμε να χρησιμοποιήσουμε την DESCRIBE καθώς η έξοδος της δήλωσης περιέχει μία γραμμή για κάθε στήλη του πίνακα. Τα πιο σημαντικά χαρακτηριστικά που παρουσιάζονται με την εκτέλεση της εντολής περιλαμβάνουν:

- Το πεδίο της "τιμής" (value) δηλώνει το όνομα της στήλης.
- Το πεδίο "τύπος" (type) εμφανίζει τον τύπο δεδομένων της στήλης.
- Περιλαμβάνει σημείωση αν η στήλη μπορεί να περιέχει τιμές NULL ή αν δεν μπορεί.
- Το πεδίο "κλειδί" (key) μπορεί να είναι: άδειο πεδίο το οποίο υποδεικνύει ότι η εν λόγω στήλη, είτε δεν δεικτοδοτείται ή δεικτοδοτείται μόνο ως δευτερεύουσα στήλη σε ένα ευρετήριο πολλαπλών στηλών, μη μοναδικών. Αν το πεδίο κλειδί είναι η λέξη-κλειδί PRI αυτό υποδεικνύει πως η παρούσα στήλη είναι ένα πρωτεύον κλειδί ή είναι μία από τις στήλες σε ένα πρωτεύον κλειδί πολλαπλών στηλών. Αν η τιμή κλειδιού είναι η λέξη-κλειδί UNI, αυτό δείχνει ότι η στήλη είναι η πρώτη στήλη ενός ευρετηρίου μοναδικής τιμής (unique valued) που δεν μπορεί να περιέχει NULL τιμές. Τέλος Αν η τιμή κλειδιού είναι η λέξη-κλειδί MUL, αυτό δείχνει ότι η στήλη είναι η πρώτη

στήλη ενός μη μοναδικού ευρετηρίου ή ενός ευρετηρίου μοναδικής τιμής που μπορεί να περιέχει τιμές NULL.

Σημειώνεται ότι ως προς το κλειδί (key), υπάρχουν περιπτώσεις όπου μία στήλη λαμβάνει περισσότερες από μία τιμές. Για παράδειγμα, μια στήλη που είναι ένα πρωτεύον κλειδί θα μπορούσε επίσης να είναι μέρος και άλλων ευρετηρίων.

4.9.Δημιουργία Πινάκων (Creating Tables)

Ένας νέος πίνακας μπορεί να δημιουργηθεί χρησιμοποιώντας μια δήλωση CREATE TABLE, που περιλαμβάνει το όνομα του πίνακα και μια λίστα των στηλών με τους αντίστοιχους τύπους δεδομένων. Ο ορισμός του πίνακα μπορεί επίσης να περιλαμβάνει ορισμούς ευρετηρίου. Για τη δημιουργία πίνακα, χρησιμοποιείται το όνομά του ακολουθούμενο από μια λίστα με τους ορισμούς των στηλών εντός παρενθέσεων όπως στο ακόλουθο παράδειγμα.

```
CREATE TABLE `υπουργείο`.`ΥΠΑΛΛΗΛΟΣ` (
  id INT NOT NULL,
  name VARCHAR(45) NOT NULL,
  hire_date DATE NOT NULL);
```

Ένας πίνακας πρέπει να περιλαμβάνει τουλάχιστο μία στήλη/πεδίο και να ανήκει σε μία ΒΔ. Όταν δημιουργείται ένας πίνακας παραμένει στη ΒΔ έως ότου διαγραφεί με την εντολή DROP TABLE <table_name>.

Είναι δυνατό να δημιουργηθούν προσωρινοί πίνακες κατά τον προγραμματισμό με την SQL για να αποθηκεύουν προσωρινά στοιχεία που διατηρούνται κατά τη διάρκεια σύνδεσης του πελάτη που εκτελεί το πρόγραμμα SQL. Για τη δημιουργία προσωρινού πίνακα προσθέτουμε τη δήλωση TEMPORARY μετά τη δήλωση CREATE οπότε έχουμε:

```
CREATE TEMPORARY TABLE `υπουργείο`.`ΥΠΑΛΛΗΛΟΣ1` (
  id INT NOT NULL,
  name VARCHAR(45) NOT NULL);
```

Σε αντίθεση με τους temporary πίνακες, οι πίνακες MEMORY διατηρούνται στη μνήμη RAM οπότε διατηρούνται έως ότου πραγματοποιηθεί επανεκκίνηση της

μηχανής εκτέλεσης του ΣΔΒΔ. Επιπλέον ένας πίνακας MEMORY μπορεί να χρησιμοποιηθεί από οποιοδήποτε πελάτη όπως ένας κανονικός – τυπικός πίνακας παρόλο που είναι «προσωρινά αποθηκευμένος στη RAM».

4.10. Επιλογές Στηλών (Column Options)

Το τελευταίο κομμάτι του ορισμού στήλης (που ακολουθεί τον τύπο δεδομένων) μπορεί να περιλαμβάνει προαιρετικά χαρακτηριστικά που τροποποιούν τον τρόπο που η MySQL χειρίζεται τη στήλη.

Αριθμητικοί τύποι δεδομένων, εκτός του BIT μπορεί να έχουν τα εξής χαρακτηριστικά:

- UNSIGNED: προκαλεί την άρση των αρνητικών τιμών.
- ZEROFILL: προκαλεί στις τιμές που ανακτώνται να γεμίζουν με μηδενικά αριστερά μέχρι να συμπληρωθεί το πλάτος της στήλης. Για παράδειγμα, αν έχουμε αποθηκεύσει τις τιμές 0, 14, και 1234 σε μια στήλη που είναι ορίζεται ως INT(5) ZEROFILL, η MySQL τις εμφανίζει ως 00000, 00014, και 01234 και όταν τις ανακτάμε. Χρησιμοποιώντας το χαρακτηριστικό ZEROFILL για μια στήλη την αναγκάζουμε να είναι UNSIGNED, επίσης.
- AUTO_INCREMENT: εφαρμόζεται σε ακέραιους τύπους δεδομένων. Χρησιμοποιείται για να παράγει ακολουθίες διαδοχικών μοναδικών τιμών.
- CHARACTER SET: καθορίζει το σύνολο χαρακτήρων που θα χρησιμοποιηθούν για τη στήλη. Το CHARSET είναι συνώνυμο για το CHARACTER SET.
- COLLATE: καθορίζει το collation για το σετ χαρακτήρων (CHARACTER SET).
- BINARY: είναι η συντομογραφία για τον καθορισμό της δυαδικής αντιπαραβολής (binary collation) για το σετ χαρακτήρων της στήλης.
- NULL και NOT NULL: ισχύουν για όλους τους τύπους των στηλών. Υποδεικνύουν αν μια στήλη μπορεί να περιέχει ή όχι NULL τιμές. Εάν δεν καθοριστεί καμία ιδιότητα, η προεπιλογή είναι να επιτρέπονται NULL τιμές στη στήλη. Οι εξαιρέσεις είναι ότι NULL δεν μπορεί να αποθηκευτεί σε AUTO_INCREMENT στήλες ή σε στήλες που είναι TIMESTAMP που

ορίζεται να ενημερώνονται αυτόματα με το τρέχων TIMESTAMP όταν οριστεί στην τιμή NULL.

- Η τιμή DEFAULT παρέχει σε μια στήλη με μια προκαθορισμένη τιμή που πρέπει να χρησιμοποιηθεί όταν δημιουργείται μία νέα εγγραφή, αλλά δεν διευκρινίζεται ρητά μια τιμή για τη στήλη. Για παράδειγμα, προκαθορισμένες τιμές χρησιμοποιούνται όταν εκτελέσει μια εντολή INSERT που δεν παρέχει τιμές για όλες τις στήλες του πίνακα.

4.11. Δημιουργία πινάκων βάσει υπαρχόντων πινάκων (Creating Tables Based on Existing Tables)

Είναι δυνατό να δημιουργήσουμε έναν πίνακα ως αποτέλεσμα ενός ερωτήματος ή κατ' ομοίωση ενός άλλου πίνακα:

- Η CREATE TABLE ... SELECT δημιουργεί έναν πίνακα με βάση το αποτέλεσμα που θα επιστραφεί από ένα ερώτημα SELECT. Θυμίζουμε ότι κάθε ερώτημα SQL που επιστρέφει αποτελέσματα ουσιαστικά επιστρέφει έναν πίνακα – προσωρινό – ακόμη και όταν αυτός είναι κενός περιεχομένου. Σε αυτή την περίπτωση, δημιουργείται ένας νέος πίνακας που περιλαμβάνει το σύνολο των γραμμών και των στηλών που επιστρέφονται από την SELECT.
- Η CREATE TABLE ... LIKE δημιουργεί έναν άδειο πίνακα, χρησιμοποιώντας τον ορισμό άλλου υφιστάμενου πίνακα.

Κατά τη χρήση των παραπάνω τεχνικών δεν δημιουργούνται τα ξένα κλειδιά με αυτόματο τρόπο αλλά απαιτείται να κατασκευαστούν αναλυτικά. Η διαδικασία δημιουργίας ξένων κλειδιών περιγράφεται στην αμέσως επόμενη ενότητα.

4.12. Τροποποίηση Πινάκων (Altering Tables)

Σε έναν πίνακα που έχει δημιουργηθεί είναι δυνατό να τροποποιηθούν στοιχεία της δομής του με χρήση της εντολής ALTER TABLE. Είναι δυνατή η προσθαφαίρεση στηλών, η αλλαγή του τύπου ή του ονόματος ενός πεδίου/στήλης, η προσθαφαίρεση ευρετηρίων καθώς και η μετονομασία του πίνακα.

Αν υποθέτουμε ότι ένας πίνακας αρχικά είχε τον ακόλουθο ορισμό:

```
CREATE TABLE `υπουργείο`.`ΥΠΑΛΛΗΛΟΣ`  
( ID INT NOT NULL,  
 Name VARCHAR(45) NOT NULL );
```

Για να προστεθεί μια νέα στήλη σε έναν πίνακα, χρησιμοποιούμε την ALTER TABLE, με την πρόταση ADD που καθορίζει τον ορισμό της στήλης. Ένας ορισμός στήλης χρησιμοποιεί την ίδια σύνταξη για την ALTER TABLE όπως στην CREATE TABLE. Για παράδειγμα, για να προστεθεί μια στήλη ημερομηνίας με το όνομα Hire_Date, για την καταγραφή των ημερομηνιών πρόσληψης, μπορεί να χρησιμοποιηθεί η ακόλουθη εντόλη:

```
ALTER TABLE `υπουργείο`.`ΥΠΑΛΛΗΛΟΣ` ADD HIRE_DATE DATE  
NOT NULL;
```

Το πεδίο προστίθεται στο τέλος της λίστας του πίνακα. Μπορούμε να καθορίσουμε τη σειρά των πεδίων δηλώνοντας στο τέλος της δήλωσης ALTER τη δήλωση AFTER <column_name>.

Για να μπορέσει κανείς να αφαιρέσει μία στήλη από έναν πίνακα χρησιμοποιεί την παράμετρο DROP στην ALTER TABLE ως εξής:

```
ALTER TABLE table_name DROP column_name;
```

Επίσης είναι δυνατό να αντικατασταθεί μία στήλη από μία άλλη με ίδιο ή διαφορετικό όνομα και τύπο δεδομένων με την παράμετρο CHANGE ως εξής:

```
ALTER TABLE table_name CHANGE column_name_initial  
new_column_name DATATYPE;
```

Αν χρειαστεί να τροποποιήσουμε μία υπάρχουσα στήλη ως προς τον τύπο δεδομένων της χρησιμοποιούμε την παράμετρο MODIFY ακολουθούμενο από το όνομα της στήλης και το νέο τύπο δεδομένων που επιθυμούμε.

Τέλος μπορούμε να μετονομάσουμε έναν πίνακα:

```
ALTER TABLE table_initial RENAME TO table_new
```

4.13. Διαγραφή Πινάκων (Dropping Tables)

Για να γίνει διαγραφή πίνακα χρησιμοποιούμε την εντολή DROP TABLE. Μπορούμε να περιλάβουμε έναν ή περισσότερους πίνακες προς διαγραφή χωριζόμενους με κόμμα:

```
DROP TABLE table_1 {, table_2, .... Table_n};
```

Εάν διαγραφεί πίνακας κατά λάθος, θα πρέπει να ανακτηθεί από αντίγραφα ασφαλείας καθώς η εντολή δεν έχει οποιουδήποτε είδους undo, οπότε πρέπει κανείς να είναι ιδιαίτερα προσεκτικός κατά τη χρήση της. Για να αφαιρεθούν εγγραφές από έναν πίνακα χωρίς να αφαιρεθεί ο ίδιος ο πίνακας, χρησιμοποιείται η εντολή DELETE ή η TRUNCATE TABLE. Οποιαδήποτε από τις ακόλουθες δηλώσεις αδειάζει εντελώς από δεδομένα – εγγραφές τον αναφερόμενο πίνακα, αλλά δεν τον σβήνει από τον κατάλογο των πινάκων της ΒΔ στην οποία ανήκει:

```
DELETE FROM table_initial;
```

```
TRUNCATE TABLE table_initial;
```

Η εντολή DELETE παίρνει μια προαιρετική πρόταση WHERE, η οποία προσδιορίζει ποιες εγγραφές επιλέγουμε να διαγραφούν. Αυτή είναι και η πιο συνήθης χρήση της, δηλαδή η διαγραφή υποσυνόλου των εγγραφών ενός πίνακα.

4.14. Περιορισμοί Ακεραιότητας (Integrity Constraints) – Πρωτεύοντα κλειδιά

Μία στήλη ή περισσότερες μπορεί να περιέχουν δεδομένα που προσδιορίζουν μοναδικά κάθε εγγραφή ενός πίνακα. Έχουμε περιγράψει ότι στην περίπτωση αυτή μπορεί η στήλη αυτή να αποτελεί το πρωτεύον κλειδί του πίνακα. Ένας μοναδικός αριθμός αποτελεί συχνά την προσέγγιση καθορισμού ενός πρωτεύοντος κλειδιού καθώς αποτελεί μια εγγυημένη και απλή λύση. Συχνά ονομάζουμε τη στήλη αυτή με το όνομα ID. Για παράδειγμα:

```
CREATE TABLE `υπουργείο`.`ΥΠΑΛΛΗΛΟΣ`  
(  
    id INT NOT NULL,  
    name VARCHAR(45) NOT NULL,  
    PRIMARY KEY (id)  
);
```

Μια εναλλακτική δήλωση του παραπάνω θα ήταν αν απευθείας προσδιορίζαμε τη στήλη ID ως πρωτεύον κλειδί (primary key) στην ίδια ακριβώς γραμμής εντολής.

```
CREATE TABLE `υπουργείο`.`ΥΠΑΛΛΗΛΟΣ`  
(  
    id INT NOT NULL PRIMARY KEY,  
    name VARCHAR(45) NOT NULL);
```

Όταν το πρωτεύον κλειδί αποτελείται από πολλές στήλες τότε η σύνταξη της εντολής γίνεται όπως στο ακόλουθο παράδειγμα:

```
CREATE TABLE `υπουργείο`.`ΥΠΑΛΛΗΛΟΣ`  
(  
    last_name VARCHAR(45) NOT NULL,  
    first_name VARCHAR(45) NOT NULL,  
    PRIMARY KEY (last_name, first_name)  
);
```

4.15. Δημιουργία Πινάκων με Ξένο Κλειδί

Η μηχανή αποθήκευσης InnoDB της MySQL υποστηρίζει την υλοποίηση των ξένων κλειδιών όπως είδαμε μέσα από το εργαλείο επεξεργασίας των πινάκων σε προηγούμενη ενότητα. Στην παρούσα ενότητα θα περιγραφεί η SQL που υλοποιεί την αναφορική ακεραιότητα. Για να γίνει έλεγχος της ακεραιότητας αυτού του τύπου απαιτείται η δήλωση ενός ή περισσοτέρων κλειδιών, ώστε να είναι δυνατός ο

έλεγχος συσχέτισης ανάμεσα σε διαφορετικούς πίνακες. Η αναφορική ακεραιότητα όπως έχει επισημανθεί είναι υλοποιήσιμη στη μηχανή αποθήκευσης InnoDB και διασφαλίζει ότι δεν θα παραβιάζονται από εντολές SQL οι απαιτήσεις/περιορισμοί της. Για παράδειγμα, στην αναφορική ακεραιότητα σε έναν πίνακα αν περιλαμβάνεται στήλη με περιεχόμενο τις τιμές στήλης πρωτεύοντος κλειδιού άλλου πίνακα τότε κάθε τιμή που εμφανίζεται στον πρώτο πίνακα υπάρχει πάντοτε στον πίνακα αναφοράς (που διαθέτει το πρωτεύον κλειδί). Πρακτικά με τον τρόπο αυτό ένα στοιχείο δε θα δείχνει ποτέ σε μία ανύπαρκτη εγγραφή άλλου πίνακα εφόσον διασφαλίζεται η αναφορική ακεραιότητα.

Παράλληλα είναι δυνατές πράξεις όπως η διαγραφή των στοιχείων που αναφέρονται σε εγγραφή άλλου πίνακα όταν διαγράφεται η εγγραφή αυτή μέσα από την τιμή του πρωτεύοντος κλειδιού. Πρακτικά όταν διαγράφεται (ή αντίστοιχα όταν τροποποιείται) η εγγραφή με ένα συγκεκριμένο ID – τιμή πρωτεύοντος κλειδιού – σε έναν πίνακα τότε είναι δυνατό να διαγραφούν και όλες οι εγγραφές, οι οποίες περιλαμβάνουν ως στήλη αναφορά του συγκεκριμένου ID δηλαδή ως ξένο κλειδί.

Στη συνέχεια δίνονται οι δηλώσεις δημιουργίας της σχέσης ΥΠΑΛΛΗΛΟΣ η οποία περιλαμβάνει την αναφορική ακεραιότητα ΠΡΟΪΣΤΑΜΕΝΟΣ καθώς και η σχέση ΕΞΑΡΤΩΜΕΝΟ_ΜΕΛΟΣ με ξένο κλειδί το ΑΔΤ_ΥΠΑΛΛΗΛΟΥ στη στήλη ΑΔΤ της σχέσης ΥΠΑΛΛΗΛΟΣ.

```
CREATE TABLE IF NOT EXISTS `νπουργείο`.`ΥΠΑΛΛΗΛΟΣ` (
  `ΑΔΤ` INT NOT NULL,
  `ΟΝΟΜΑΤΕΠΩΝΥΜΟ` VARCHAR(45) NULL,
  `ΦΥΛΟ` BINARY NULL,
  `ΠΡΟΪΣΤΑΜΕΝΟΣ` INT NULL,
  PRIMARY KEY (`ΑΔΤ`),
  INDEX `ΠΡΟΪΣΤΑΜΕΝΟΣ_idx`(`ΠΡΟΪΣΤΑΜΕΝΟΣ` ASC),
  CONSTRAINT `ΠΡΟΪΣΤΑΜΕΝΟΣ`
    FOREIGN KEY(`ΠΡΟΪΣΤΑΜΕΝΟΣ`)
    REFERENCES `mydb`.`ΥΠΑΛΛΗΛΟΣ`(`ΑΔΤ`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
```

```

CREATE TABLE IF NOT EXISTS
`υπουργείο`.`ΕΞΑΡΤΩΜΕΝΟ_ΜΕΛΟΣ` (
`ΑΔΤ_ΥΠΑΛΛΗΛΟΥ` INT NOT NULL,
`ΟΝΟΜΑΤΕΠΩΝΥΜΟ` VARCHAR(45) NOT NULL,
`ΗΜ_ΓΕΝΝΗΣΗΣ` DATE NULL,
`ΧΩΡΑ` VARCHAR(45) NULL,
PRIMARY KEY (`ΑΔΤ_ΥΠΑΛΛΗΛΟΥ`, `ΟΝΟΜΑΤΕΠΩΝΥΜΟ`),
CONSTRAINT `ΑΔΤ_ΥΠΑΛΛΗΛΟΥΣ`
FOREIGN KEY(`ΑΔΤ_ΥΠΑΛΛΗΛΟΥ`)
REFERENCES `mydb`.`ΥΠΑΛΛΗΛΟΣ`(`ΑΔΤ`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB

```

Η πρόταση – εντολή FOREIGN KEY περιλαμβάνει τις ακόλουθες παραμέτρους:

- Δηλώνει τη στήλη ΑΔΤ_ΥΠΑΛΛΗΛΟΥ του πίνακα ΕΞΑΡΤΩΜΕΝΟ_ΜΕΛΟΣ ως στήλη που αναφέρεται στην στήλη ΑΔΤ του πίνακα ΥΠΑΛΛΗΛΟΣ.
- Δηλώνει τη στήλη ΑΔΤ του πίνακα ΥΠΑΛΛΗΛΟΣ ως στήλη που αναφέρεται και αποτελεί το ξένο κλειδί.
- Δηλώνει ποιες ενέργειες πρέπει να γίνουν όταν συμβεί διαγραφή ή ενημέρωση.

Σε μια συσχέτιση με ξένο κλειδί οι στήλες αναφοράς και ξένου κλειδιού πρέπει να έχουν τον ίδιο τύπο δεδομένων και να ευρετηριοποιηθούν και οι δύο. Αν η στήλη αναφοράς δεν έχει ευρετήριο αυτό δημιουργείται εμμέσως. Τα μέρη on update & on delete είναι προαιρετικά.

4.16. Ευρετήρια πινάκων (Indexes)

Οι πίνακες μιας ΒΔ ιδιαίτερα με το πέρασμα του χρόνου αλλά και ορισμένες φορές

λόγω σχεδιασμού μπορεί να γίνουν πολύ μεγάλοι. Όσο μεγαλύτεροι είναι οι πίνακες τόσο βραδύτερη είναι κάθε ενέργεια ανάκτησης δεδομένων σε αυτούς. Για να διατηρηθεί η καλή χρονική απόκριση όταν χρησιμοποιούνται ερωτήματα, είναι σημαντικό να χρησιμοποιηθούν ευρετήρια στους πίνακες. Τα ευρετήρια επιτρέπουν πιο αποτελεσματική εύρεση των τιμών στις στήλες, με αποτέλεσμα οι ανακτήσεις που βασίζονται σε ευρετήρια να είναι πιο γρήγορες από αυτές που δεν βασίζονται σε αυτά. Συμβαίνει δηλαδή ότι ακριβώς γνωρίζουμε και από την εμπειρία μας για τη χρήση ευρετηρίων στην καθημερινότητα μέσω των τηλεφωνικών καταλόγων. Για τους μεγάλους πίνακες, η παρουσία ενός ευρετηρίου μπορεί να κάνει τη διαφορά ανάμεσα σε ένα ερώτημα που εκτελείται γρήγορα και κάποιο που είναι απαράδεκτα αργό. Ένας άλλος λόγος για να χρησιμοποιηθούν τα ευρετήρια είναι ότι μπορούν να επιβάλλουν περιορισμούς μοναδικότητας για να διασφαλιστεί ότι δεν υπάρχουν διπλές τιμές και ότι κάθε γραμμή ενός πίνακα μπορεί να διακριθεί από κάθε άλλη γραμμή. Για το λόγο αυτό κάθε πρωτεύον κλειδί είναι εκ κατασκευής ευρετηριοποιημένο χωρίς να χρειάζεται περαιτέρω δήλωση. Στο παρακάτω παράδειγμα δημιουργούμε ένα ευρετήριο για τη στήλη phone.

```
CREATE TABLE `νπουργείο`.`ΥΠΑΛΛΗΛΟΣ`  
(  
    last_name VARCHAR(45) NOT NULL,  
    first_name VARCHAR(45) NOT NULL,  
    phone VARCHAR(10) NOT NULL  
    PRIMARY KEY (last_name, first_name),  
    INDEX (phone)  
);
```

Εφόσον έχουν δημιουργηθεί ένα ή παραπάνω ευρετήρια σε έναν πίνακα τότε μπορούμε να τα χρησιμοποιούμε για ταχύτερες ανακτήσεις δεδομένων. Κατά την ανάκτηση, δηλώνοντας ακριβώς μετά τον αντίστοιχο πίνακα την παράμετρο USE INDEX και τη στήλη που έχει ευρετηριοποιηθεί τότε προτείνουμε στη MySQL να χρησιμοποιήσει το ευρετήριο για να βελτιώσει την ταχύτητα ανάκτησης των δεδομένων. Για παράδειγμα:

```
SELECT * from `υπουργείο`.`ΥΠΑΛΛΗΛΟΣ` USE INDEX (phone)  
WHERE phone = 6900547854
```

Εάν θέλουμε να χρησιμοποιήσουμε οπωσδήποτε πρώτα τα διαθέσιμα ευρετήρια πριν ξεκινήσει η MySQL να πραγματοποιεί πλήρη σάρωση στα δεδομένα του πίνακα για να εντοπίσει αυτό που χρειαζόμαστε τότε μπορούμε να χρησιμοποιούμε την FORCE INDEX.

Είναι δυνατό να χρησιμοποιούμε πολλαπλά ευρετήρια σε μία εντολή SELECT καθώς και να συνδυάζουμε στήλες σε ένα ευρετήριο.

5. Ανάκτηση Δεδομένων – Απλά ερωτήματα

Μελετώντας το παρόν κεφάλαιο οι επιμορφωνόμενοι θα είναι σε θέση να:

- ανακτούν δεδομένα μέσω απλών ερωτημάτων και κάνοντας χρήση της εντολής SELECT
- χειρίζονται σφάλματα και προειδοποιήσεις του συστήματος

5.1.Η εντολή SELECT

Η εντολή SELECT αποτελεί την πλέον βασική εντολή της γλώσσας SQL καθώς αφορά στην ανάκτηση δεδομένων από μία βάση δεδομένων. Η βασική μορφή της εντολής είναι η ακόλουθη:

```
SELECT <λίστα πεδίων>
FROM <λίστα πινάκων>
WHERE <συνθήκη>;
```

Στη <λίστα πεδίων> δηλώνουμε τα πεδία που θα εμφανίζονται στο αποτέλεσμα χωριζόμενα με κόμμα. Αν θέλουμε να εμφανίζονται όλα τα πεδία θα πρέπει να χρησιμοποιήσουμε τον χαρακτήρα «*».

Η πρόταση FROM δηλώνει τους πίνακες στους οποίους θα γίνει η αναζήτηση. Η πρόταση WHERE είναι προαιρετική και περιέχει τη συνθήκη που θέλουμε να ικανοποιούν οι εγγραφές του αποτελέσματος. Στο παράδειγμα που ακολουθεί, η εντολή SELECT εμφανίζει το ΑΔΤ, το ονοματεπώνυμο και το φύλο για όλους τους υπαλλήλους.

```
SELECT `ΑΔΤ`, `ονοματεπώνυμο`, `φύλο` FROM `υπάλληλος`;
```

Η εντολή SELECT χρησιμοποιείται για να ανακτά πληροφορίες από έναν ή περισσότερους πίνακες. Στη συνέχεια θα δούμε γενικές οδηγίες σχετικά με το πώς γράφουμε εντολές SELECT και πώς χρησιμοποιούμε τα διάφορα μέρη της σύνταξής της για να έχουμε τα αποτελέσματα που θέλουμε. Μία πληρέστερη σύνταξη για την SELECT είναι η παρακάτω:

```
SELECT columns
FROM table_name
WHERE filtering_expression
GROUP BY how_to_group
HAVING filtering_expression
ORDER BY how_to_sort
LIMIT row_to_return;
```

Η σύνταξη που παρουσιάζεται εδώ είναι μια απλοποιημένη έκδοση σε σχέση με την πλήρη σύνταξη της SELECT. Όλες οι προτάσεις που ακολουθούν τη λίστα της στήλης εξόδου (columns) είναι προαιρετικές. Ωστόσο, οποιαδήποτε δήλωση που τελικά συμπεριλαμβάνεται θα πρέπει να αναφέρεται με τη σειρά εμφάνισης που δίνεται παραπάνω.

Για να επιλέξουμε πληροφορίες από έναν πίνακα, είναι απαραίτητο να προσδιοριστεί ο πίνακας με την προσθήκη μιας πρότασης FROM table_name.

Η MySQL επιστρέφει ένα σύνολο αποτελεσμάτων (result set), δηλαδή ένα σύνολο εγγραφών, που αποτελείται από μία σειρά εξόδου για κάθε γραμμή του πίνακα. Εάν ο πίνακας είναι άδειος, το αποτέλεσμα θα είναι άδειο. Θα επιστραφεί απλώς ένα άδειο σύνολο αποτελεσμάτων. Για μια πράξη SELECT που ανακτά κάθε στήλη από έναν πίνακα, μπορεί να χρησιμοποιηθεί η συντόμευση * για να καθορίσει τις στήλες εξόδου. Το * σημαίνει "όλες τις στήλες του πίνακα", οπότε για τον πίνακα `υπάλληλος`, οι ακόλουθες δηλώσεις είναι ισοδύναμες:

```
SELECT `ΑΔΤ`,`ονοματεπώνυμο`,`φύλο` FROM `υπάλληλος`;
```

```
SELECT * FROM `υπάλληλος`;
```

Η χρήση της συντόμευσης * είναι σαφώς πιο βολική από το να πληκτρολογήσουμε μια λίστα με τα ονόματα των στηλών. Ωστόσο, αν θέλουμε να ανακτήσουμε όλες τις στήλες από έναν πίνακα και θέλουμε να διασφαλίσουμε ότι οι στήλες εμφανίζονται από αριστερά προς τα δεξιά σε μια συγκεκριμένη σειρά, δεν μπορεί να χρησιμοποιηθεί το *, επειδή δεν παρέχει κανέναν έλεγχο στη σειρά με την οποία θα εμφανίζονται στήλες. Θα πρέπει να αναφερθούν οι στήλες ρητά με τη σειρά που επιθυμούμε να τις δούμε.

Επιπλέον δε θα πρέπει να χρησιμοποιείται ένα SELECT * για να βρεθεί η τρέχουσα σειρά εμφάνισης από αριστερά προς τα δεξιά για τις στήλες σε έναν πίνακα καθώς δεν είναι ορθό να υποθέτουμε ότι θα εμφανίζονται πάντα στην ίδια σειρά και σε επόμενα μελλοντικά ερωτήματα.

Τέλος αν ένα ερώτημα επιστρέφει πολλαπλά ίδια αποτελέσματα, μπορούμε να κρατήσουμε τα μοναδικά αποτελέσματα μόνο χρησιμοποιώντας την παράμετρο DISTINCT πριν από τη στήλη που περιλαμβάνει τα πολλαπλά αποτελέσματα.

Είναι επίσης δυνατό, όπως και σε κάθε γλώσσα προγραμματισμού, να προστίθενται σχόλια στον κώδικα SQL. Μπορεί κανείς να προσθέσει σχόλια ξεκινώντας με το γράμμα «#», τις παύλες «--» ή να ορίζει σχόλια σε πολλαπλές γραμμές /* */.

5.2.Η συνθήκη φιλτραρίσματος where

Μία δήλωση SELECT ανακτά κάθε εγγραφή του πίνακα στην απλή της μορφή. Στην περίπτωση που χρειαζόμαστε να περιορίσουμε το επιστρεφόμενο αποτέλεσμα της ανάκτησης στις εγγραφές εκείνες που ικανοποιούν κάποια κριτήρια επιλογής/ συνθήκες φιλτραρίσματος τότε χρησιμοποιούμε τη δήλωση WHERE. Μια δήλωση WHERE μπορεί να είναι όσο απλή ή όσο σύνθετη χρειάζεται, για να προσδιορίσει τις γραμμές που είναι σχετικές με το σκοπό μας. Για παράδειγμα, για να ανακτήσουμε τις εγγραφές από τον πίνακα υπάλληλος για όσους υπαλλήλους είναι άντρες ('φύλο' = 'Α'), είναι επαρκές να χρησιμοποιήσουμε μια πρόταση WHERE που να καθορίζει μία και μοναδική προϋπόθεση:

```
SELECT `ΑΔΤ`, `ονοματεπώνυμο`, `φύλο` FROM `υπάλληλος`
WHERE `φύλο` = 'Α';
```

Πιο σύνθετες προτάσεις WHERE καθορίζουν πολλές συνθήκες, που μπορούν να συνδυαστούν με λογικούς τελεστές όπως AND και OR. Η ακόλουθη δήλωση επιστρέφει τις εγγραφές με απαίτηση η στήλη όνομα και η στήλη επώνυμο να περιλαμβάνουν το στοιχείο Μάριος Συρίγος αντίστοιχα.:

```
SELECT `ΑΔΤ`, `φύλο` FROM `υπάλληλος`
WHERE `ονομα` = 'Μάριος' AND `επώνυμο` = 'Συρίγος';
```

Στην περίπτωση που οι συνθήκες που επιθυμούμε να χρησιμοποιήσουμε περιλαμβάνουν διάστημα τιμών μπορούμε να κάνουμε χρήση της παραμέτρου BETWEEN.

Επιπλέον στην περίπτωση που χρειαζόμαστε να εντοπίσουμε με μία συνθήκη κάποιο πρότυπο τότε χρησιμοποιούμε την παράμετρο LIKE. Πρόκειται για μία παράμετρο που δίνει τεράστια ευελιξία στον ορισμό συνθηκών φιλτραρίσματος. Με το LIKE είναι δυνατό να γίνεται χρήση μεταχαρακτήρων (wildcards) για αναζήτηση. Πρόκειται για ειδικούς χαρακτήρες που αντιπροσωπεύουν είτε οποιοδήποτε χαρακτήρα «_» είτε μια σειρά από οποιουσδήποτε χαρακτήρες «%». Για παράδειγμα αν θέλουμε να βρούμε όσα ονόματα ξεκινούν από Γ θα γράφαμε ‘Γ%’. Ένα πρότυπο αναζήτησης σε μια συνθήκη μπορεί να περιλαμβάνει συνδυασμούς των μεταχαρακτήρων.

```
SELECT `ΑΔΤ`, `ονομα`, `επώνυμο` ,`φύλο` FROM `υπάλληλος`  
WHERE `ονομα` = 'ΙΩΑΝΝΗΣ' AND `επώνυμο` LIKE 'Γ%';
```

5.3. Ταξινόμηση – Order by

Όταν εκτελείται ένα ερώτημα, ο εξυπηρετητής μπορεί να επιστρέψει τις γραμμές με οποιαδήποτε σειρά χωρίς κάποια ταξινόμηση. Αυτή η σειρά μπορεί να επηρεαστεί από παράγοντες όπως η σειρά με την οποία γραμμές είναι στην πραγματικότητα αποθηκευμένες στον πίνακα ή από τα ευρετήρια που χρησιμοποιούνται για την βελτιστοποίηση του ερωτήματος. Αν χρειαζόμαστε οι εγγραφές-αποτελέσματα να επιστρέφουν με μια συγκεκριμένη σειρά, περιλαμβάνουμε την δήλωση ORDER BY, που υποδεικνύει πώς να ταξινομηθούν τα αποτελέσματα.

Η ORDER BY επιτρέπει να δηλωθούν μία ή περισσότερες στήλες, διαχωρισμένες με κόμμα, για να χρησιμοποιηθούν για την ταξινόμηση:

```
SELECT `ΑΔΤ`, `ονομα`, `επώνυμο` ,`φύλο` FROM `υπάλληλος`  
ORDER BY `επώνυμο`;
```

Από προεπιλογή, η ORDER BY ταξινομεί τις τιμές σε αύξουσα σειρά (από το μικρότερο στο μεγαλύτερο). Κάθε στήλη ταξινόμησης μπορεί να δηλωθεί με την παράμετρο ASC αν θέλουμε να έχουμε τα αποτελέσματα σε αύξουσα σειρά ρητά ή με DESC αν θέλουμε τα αποτελέσματα να επιστρέφονται σε φθίνουσα σειρά.

5.4. Τελεστές (Αριθμητικοί, Λογικοί κλπ)

Υπάρχουν δύο είδη τελεστών, οι τελεστές σύγκρισης και οι λογικοί τελεστές. Οι τελεστές αυτοί χρησιμοποιούνται κυρίως στην πρόταση WHERE, HAVING για να φιλτράρονται τα δεδομένα που πρέπει να επιλεγούν. Ο παρακάτω πίνακας περιγράφει κάθε τελεστή σύγκρισης

ΤΕΛΕΣΤΕΣ ΣΥΓΚΡΙΣΗΣ	ΠΕΡΙΓΡΑΦΗ
=	ίσο με
<>, !=	δεν είναι ίσο με
<	μικρότερο από
>	μεγαλύτερο από
>=	μεγαλύτερο ή ίσο από
<=	Μικρότερο ή ίσο από

Υπάρχουν τρεις Λογικοί Τελεστές και συγκεκριμένα οι AND, OR και NOT. Οι τελεστές αυτοί συγκρίνουν δύο προϋποθέσεις σε μια δεδομένη χρονική στιγμή για να επιστραφούν τα κατάλληλα δεδομένα γραμμή γραμμή. Κατά την ανάκτηση δεδομένων με δήλωση SELECT, μπορεί να χρησιμοποιηθούν λογικοί τελεστές στην πρόταση WHERE, για να συνδυαστούν περισσότερες από μία συνθήκες.

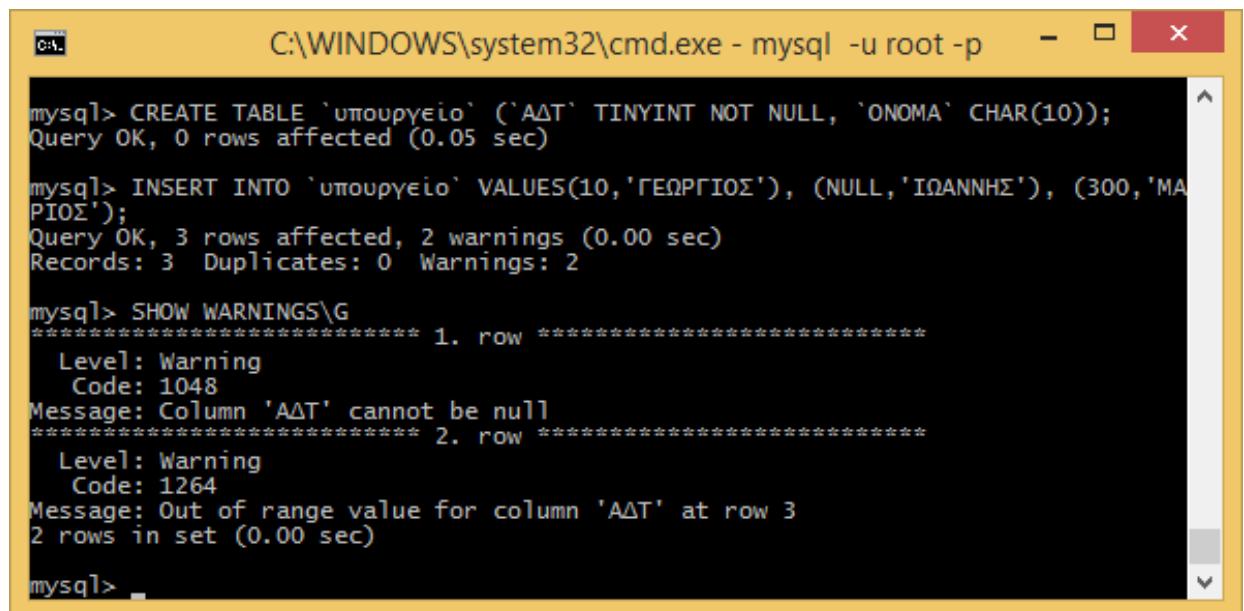
ΛΟΓΙΚΟΙ ΤΕΛΕΣΤΕΣ	ΠΕΡΙΓΡΑΦΗ
OR	Για να επιλεγεί η γραμμή θα πρέπει τουλάχιστο μία από τις συνθήκες να ισχύει.
AND	Για να επιλεγεί η γραμμή θα πρέπει όλες οι συνθήκες να ισχύουν.
NOT	Για να επιλεγεί η γραμμή θα πρέπει συνθήκη που προσδιορίζεται με NOT να μην ισχύει.

5.5. Χειρισμός Σφαλμάτων και Προειδοποιήσεων / Αντιμετώπιση προβλημάτων

Το *SHOW WARNINGS* είναι μία διαγνωστική εντολή που εμφανίζει πληροφορίες σχετικά με τις συνθήκες (σφάλματα, προειδοποιήσεις και σημειώσεις) που προκύπτουν από την εκτέλεση μια εντολής κατά την τρέχουσα περίοδο. Οι προειδοποιήσεις δημιουργούνται για δηλώσεις ΓΧΔ όπως INSERT, UPDATE και LOAD DATA INFILE καθώς και δηλώσεις ΓΟΔ, όπως οι CREATE TABLE και ALTER TABLE.

```
SHOW WARNINGS [LIMIT [offset,] row_count]
SHOW COUNT(*) WARNINGS
```

Στη συνέχεια παρουσιάζεται ένα απλό παράδειγμα που δείχνει προειδοποιήσεις μετατροπής δεδομένων για την INSERT:



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> CREATE TABLE `upourgygio` (`ADT` TINYINT NOT NULL, `ONOMA` CHAR(10));
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO `upourgygio` VALUES(10, 'ΓΕΩΡΓΙΟΣ'), (NULL, 'ΙΩΑΝΝΗΣ'), (300, 'ΜΑΡΙΟΣ');
Query OK, 3 rows affected, 2 warnings (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 2

mysql> SHOW WARNINGS\G
***** 1. row *****
  Level: Warning
  Code: 1048
Message: Column 'ADT' cannot be null
***** 2. row *****
  Level: Warning
  Code: 1264
Message: Out of range value for column 'ADT' at row 3
2 rows in set (0.00 sec)

mysql>
```

Εικόνα 30 Αποτελέσματα

Η *SHOW ERRORS* είναι μία ακόμη διαγνωστική εντολή, που εμφανίζει πληροφορίες μόνο για λάθη.

```
SHOW ERRORS [LIMIT [offset,] row_count]
SHOW COUNT(*) ERRORS
```

6. Γλώσσα Χειρισμού Δεδομένων – Data Manipulation Language (DML)

Μελετώντας το παρόν κεφάλαιο οι επιμορφωνόμενοι θα είναι σε θέση να:

- διαχειρίζονται τα δεδομένα ενός πίνακα. Συγκεκριμένα να εισάγουν δεδομένα σε πίνακα, να τροποποιούν δεδομένα, να αντικαθιστούν δεδομένα και να διαγράφουν μέρος των δεδομένων ή όλα τα δεδομένα ενός πίνακα.

Στα προηγούμενα κεφάλαια περιγράφηκε η δημιουργία μιας βάσης δεδομένων, η προσθήκη πινάκων σε μια βάση δεδομένων και αναλύθηκε η ανάκτηση – προσπέλαση των δεδομένων. Στο κεφάλαιο αυτό θα εξετάσουμε την προσθήκη και ενημέρωση των δεδομένων μιας ΒΔ. Τα περισσότερα ΣΔΒΔ παρέχουν εργαλεία διαχείρισης δεδομένων που επιτρέπουν την ανάκτηση δεδομένων ολόκληρων των πινάκων μιας βάσης, την προσθήκη, ενημέρωση και διαγραφή δεδομένων. Τα εργαλεία αυτά όμως είναι κατάλληλα για την επεξεργασία μικρής ποσότητας δεδομένων και χρησιμοποιούνται κυρίως για τον έλεγχο μιας ΒΔ. Για την διαχείριση μεγάλης ποσότητας δεδομένων η SQL παρέχει την γλώσσα χειρισμού δεδομένων - Data Manipulation Language (DML).

Η γλώσσα χειρισμού δεδομένων αποτελεί ένα από τα σημαντικότερα και βασικότερα κομμάτια της SQL. Συγκεκριμένα, η γλώσσα χειρισμού δεδομένων παρέχει την δυνατότητα στους χρήστες μιας βάσης δεδομένων α) να εισάγουν νέα δεδομένα, β) να τροποποιούν ήδη αποθηκευμένα δεδομένα, γ) να διαγράφουν δεδομένα και δ) να προσπελαύνουν τα δεδομένα της βάσης. Το δ, όπου ουσιαστικά αποτελεί το τμήμα της γλώσσας όπου χρησιμοποιείται μόνο για την ανάκτηση – προσπέλαση δεδομένων ονομάζεται γλώσσα ερωτημάτων (Query language). Η γλώσσα ερωτημάτων περιγράφηκε αναλυτικά στο προηγούμενο κεφάλαιο.

Η γλώσσα χειρισμού δεδομένων περιλαμβάνει τις εντολές:

- INSERT για την προσθήκη δεδομένων
- UPDATE για την τροποποίηση ήδη αποθηκευμένων δεδομένων
- DELETE για την διαγραφή δεδομένων
- SELECT για την ανάκτηση δεδομένων

Στο κεφάλαιο αυτό θα εστιάσουμε στις εντολές INSERT, UPDATE, DELETE και θα περιγράψουμε την σύνταξή τους στην MySQL. Επίσης θα εξετάσουμε τις εντολές REPLACE και TRUNCATE που υποστηρίζονται από την MySQL. Η REPLACE αποτελεί εντολή της γλώσσας χειρισμού δεδομένων. Παρόλο που η TRUNCATE αποτελεί εντολή της γλώσσας ορισμού δεδομένων θα περιγραφεί σε αυτό το κεφάλαιο προκειμένου να αναλυθούν οι διαφορές της με την εντολή DELETE.

6.1. Εισαγωγή Δεδομένων

6.1.1. Εντολή INSERT

Για να εισάγουμε δεδομένα σε έναν πίνακα χρησιμοποιούμε την εντολή INSERT. Η εντολή INSERT δίνει την δυνατότητα να ορίσουμε εμείς τα γνωρίσματα-στήλες του πίνακα που θέλουμε να εισάγουμε σε μια πλειάδα-εγγραφή. Επιπλέον με την εντολή INSERT μπορούμε να εισάγουμε σε έναν πίνακα πολλές πλειάδες-εγγραφές ταυτόχρονα. Για να χρησιμοποιήσουμε την εντολή INSERT πρέπει να έχουμε τουλάχιστον INSERT δικαιώματα. Κατά την εισαγωγή δεδομένων το ΣΔΒΔ ελέγχει τους περιορισμούς ακεραιότητας που καθορίστηκαν κατά την δημιουργία των πινάκων της ΒΔ μας. Αν κάποιος από τους περιορισμούς ακεραιότητας δεν ικανοποιείται η εισαγωγή δεδομένων αποτυγχάνει.

6.1.1.1. INSERT με χρήση του όρου <values>

Το γενικό πρότυπο σύνταξης που χρησιμοποιείται στην εντολή INSERT με χρήση του όρου <values> είναι:

```
INSERT INTO tbl_name [(column 1 [, ... column n] )] VALUES (value 1 [, ...
value n]);
```

- *tbl_name*: Το όνομα του πίνακα
- *column 1* ... *column n* : Τα ονόματα των γνωρισμάτων-στηλών του πίνακα
- *value 1* ... *value n*: είναι οι τιμές των αντίστοιχων γνωρισμάτων-στηλών

Τα ονόματα των γνωρισμάτων-στηλών που προσδιορίζονται αμέσως μετά το όνομα του

πίνακα έχουν ένα-προς-ένα αντιστοιχία με τις τιμές που εισάγονται μετά τον τελεστή VALUES. Έχουμε την δυνατότητα να εισάγουμε τιμές σε ορισμένα γνωρίσματα του πίνακα. Τα υπόλοιπα γνωρίσματα που δεν εμφανίζονται στην λίστα της εντολής INSERT αρχικοποιούνται με την τιμή που τους έχει προσδιορισθεί ως αρχική τιμή (default value) κατά την δημιουργία του πίνακα. Αν δεν έχει προσδιορισθεί κάποια αρχική τιμή τότε εισάγεται σε αυτά η κενή τιμή (NULL). Πριν την εκτέλεση της εντολής INSERT το ΣΔΒΔ ελέγχει όλους τους περιορισμούς ακεραιότητας που έχουν ορισθεί στον πίνακα. Σε περίπτωση που κάποιος περιορισμός δεν ικανοποιείται τότε η εντολή INSERT αποτυγχάνει εμφανίζοντας σχετικό μήνυμα λάθους. Επίσης αν ένα γνώρισμα-στήλη κατά την δημιουργία του πίνακα είχε ορισθεί ως NOT NULL και το γνώρισμά αυτό δεν εμφανίζεται στην λίστα γνωρισμάτων-στηλών της εντολής INSERT το ΣΔΒΔ που επιβάλλει τον περιορισμό NOT NULL απορρίπτει την εκτέλεση της εντολής INSERT αφού το γνώρισμα-στήλη που έχει ορισθεί ως NOT NULL έχει τιμή την κενή τιμή (NULL).

Μια δεύτερη μορφή της εντολής INSERT είναι:

INSERT INTO *tbl_name* VALUES (*value 1* [, ... *value n*]);

- ***tbl_name*: Το όνομα του πίνακα**
- ***value 1* ... *value n*: είναι οι τιμές των αντίστοιχων γνωρισμάτων-στηλών**

Στη δεύτερη αυτή σύνταξη της εντολής INSERT η λίστα με τα γνωρίσματα-στήλες του πίνακα δεν υπάρχει, εμφανίζονται μόνο οι τιμές των γνωρισμάτων αυτών. Στην σύνταξη αυτή είναι απαραίτητη η διατήρηση της σειράς των τιμών (values) αντίστοιχα για κάθε γνώρισμα σύμφωνα με την σειρά που τα γνωρίσματα αυτά ορίστηκαν κατά την δημιουργία του πίνακα (στην εντολή CREATE TABLE). Αρχικά η τιμή του πρώτου γνωρίσματος, κατόπιν η τιμή του δεύτερου και ούτω καθ' εξής.

Παράδειγμα

Για παράδειγμα ας θεωρήσουμε τον πίνακα Departments που περιλαμβάνει τα Τμήματα του Οργανισμού ενός Δημόσιου Φορέα (Εικόνα 31 Πίνακας DEPARTMENTS).

	id	department_name	building_address	employee_number	directorate_id
	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	KΤΙΡΙΟ Α	10	1
	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	KΤΙΡΙΟ Α	12	2
	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ	KΤΙΡΙΟ Α	1	2
	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ	KΤΙΡΙΟ Α	0	1
	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΕΩΝ	KΤΙΡΙΟ Α	0	2
	NULL	NULL	NULL	NULL	NULL

Εικόνα 31 Πίνακας DEPARTMENTS

Στα πλαίσια αλλαγών και μεταρρυθμίσεων ο Φορέας πρόσθεσε ένα νέο Τμήμα στον Οργανισμό του το Τμήμα Πρωτοκόλλου που ανήκει στην Διεύθυνση Διοικητικού–Οικονομικού (directorate_id=2). Ο πίνακας DEPARTMENTS της ΒΔ μας πρέπει να ενημερωθεί με την ακόλουθη πλειάδα-εγγραφή (Εικόνα 32 Προσθήκη Πλειάδας-Εγγραφής).

	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ	KΤΙΡΙΟ Α	5	2
--	---	-------------------	----------	---	---

Εικόνα 32 Προσθήκη Πλειάδας-Εγγραφής

Ο πίνακας DEPARTMENTS μετά την ενημέρωση πρέπει να έχει την ακόλουθη μορφή (Εικόνα 33 Πίνακας DEPARTMENTS μετά την προσθήκη):

	id	department_name	building_address	employee_number	directorate_id
▶	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	KΤΙΡΙΟ Α	10	1
	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	KΤΙΡΙΟ Α	12	2
	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ	KΤΙΡΙΟ Α	1	2
	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ	KΤΙΡΙΟ Α	0	1
	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΕΩΝ	KΤΙΡΙΟ Α	0	2
	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ	KΤΙΡΙΟ Α	5	2
	NULL	NULL	NULL	NULL	NULL

Εικόνα 33 Πίνακας DEPARTMENTS μετά την προσθήκη

Η εντολή INSERT που θα πραγματοποιήσει την προσθήκη της πλειάδας-εγγραφής είναι:

```
INSERT INTO DEPARTMENTS (id, department_name, building_address,
employee_number, directorate_id) VALUES (6, 'ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ',
'ΚΤΙΡΙΟ Α', 5, 2)
```

ή πιο απλά χωρίς την λίστα γνωρισμάτων – στηλών (ακολουθώντας όμως οι τιμές την σειρά των γνωρισμάτων όπως ορίστηκαν στην δημιουργία του πίνακα)

**INSERT INTO DEPARTMENTS VALUES (6, 'ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ',
'ΚΤΙΡΙΟ Α', 5, 2)**

Σημείωση

Για την εισαγωγή αλφαριθμητικών δεδομένων είναι απαραίτητη η χρήση αποστρόφου ('<αλφαριθμητικό>') στην αρχή και στο τέλος του αλφαριθμητικού. Χρησιμοποιούμε την απόστροφο και στην εισαγωγή ημερομηνιών που θα δούμε παρακάτω. Τα αριθμητικά δεδομένα χρησιμοποιούνται χωρίς τις αποστρόφους.

Επίσης τα αλφαριθμητικά δεδομένα είναι *case sensitive* δηλαδή υπάρχει διάκριση μεταξύ κεφαλαίων και πεζών χαρακτήρων.

Στην περίπτωση γνωρίσματος που κατά τον ορισμό του πίνακα που περιέχεται έχει οριστεί ως *autoincrement* γνώρισμα, το γνώρισμα αυτό παραλείπεται από την σύνταξη της INSERT. Το ΣΔΒΔ αναλαμβάνει να αποδώσει τιμή σε τέτοιου είδους γνωρίσματα.

Για να προσθέσουμε σε έναν πίνακα περισσότερες πλειάδες-εγγραφές δεδομένων χρησιμοποιούμε την μορφή της εντολής INSERT ως εξής:

**INSERT INTO *tbl_name* [(*column 1* [, ... *column n*])] VALUES (*value 1_1* [, ...
value 1_n]) [, ... (*value n_1* [, ... *value n_n*])] ;**

- *tbl_name*: Το όνομα του πίνακα
- *column1* ... *column n*: Τα ονόματα των γνωρισμάτων-στηλών του πίνακα
- *value 1_1* ... *value 1_n*: είναι οι τιμές των αντίστοιχων γνωρισμάτων - στηλών της πρώτης πλειάδας
- *value n_1* ... *value n_n*: είναι οι τιμές των αντίστοιχων γνωρισμάτων - στηλών της n πλειάδας

Η λίστα των γνωρισμάτων στην σύνταξη της εντολής δεν είναι υποχρεωτική αρκεί όμως να ακολουθήσουμε την σειρά των τιμών των αντίστοιχων γνωρισμάτων όπως

αυτά ορίστηκαν κατά την δημιουργία του πίνακα

Παράδειγμα

Για παράδειγμα ας θεωρήσουμε τον πίνακα DEPARTMENTS και ας υποθέσουμε ότι θέλουμε να εισάγουμε τα τμήματα «Τμήμα Μέριμνας» και «Τμήμα Προϋπολογισμού» που ανήκουν στην Διεύθυνση Διοικητικού–Οικονομικού (directorate_id=2). Η σύνταξη της εντολής INSERT είναι:

```
INSERT INTO DEPARTMENTS (id, department_name, building_address,  
employee_number, directorate_id) VALUES (7, 'ΤΜΗΜΑ ΜΕΡΙΜΝΑΣ', 'ΚΤΙΡΙΟ  
Α', 6, 2), (8, 'ΤΜΗΜΑ ΠΡΟΥΠΟΛΟΓΙΣΜΟΥ', 'ΚΤΙΡΙΟ Α', 6, 2)
```

ή πιο απλά χωρίς την λίστα γνωρισμάτων – στηλών (ακολουθώντας όμως οι τιμές την σειρά των γνωρισμάτων όπως ορίστηκαν στην δημιουργία του πίνακα)

```
INSERT INTO DEPARTMENTS VALUES (7, 'ΤΜΗΜΑ ΜΕΡΙΜΝΑΣ', 'ΚΤΙΡΙΟ  
Α', 6, 2), (8, 'ΤΜΗΜΑ ΠΡΟΥΠΟΛΟΓΙΣΜΟΥ', 'ΚΤΙΡΙΟ Α', 6, 2)
```

Στην εντολή INSERT η εισαγωγή κενής (NULL) τιμής σε ένα γνώρισμα – στήλη πραγματοποιείται:

- είτε έμμεσα παραλείποντας το γνώρισμα - στήλη στην INSERT π.χ.

```
INSERT INTO DEPARTMENTS (id, department_name, directorate_id)  
VALUES (7, 'ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ', 2)
```

(παραλείψαμε τα γνωρίσματα building_address και employee_number)

- είτε άμεσα εισάγοντας NULL τιμή στο γνώρισμα - στήλη στην INSERT π.χ.:

```
INSERT INTO DEPARTMENTS (id, department_name, building_address,  
employee_number, directorate_id) VALUES (7, 'ΤΜΗΜΑ
```

ΠΡΟΜΗΘΕΙΩΝ', NULL, NULL, 2)

(εισάγαμε NULL τιμές στα γνώρισματα building_address και employee_number)

Απαραίτητη προϋπόθεση για την εισαγωγή κενών (NULL) τιμών είναι τα γνωρίσματα – στήλες του πίνακα να έχουν ορισθεί ως NULLABLE πεδία στον ορισμό του πίνακα.

Συνήθη Σφάλματα

Τα συνηθέστερα σφάλματα που μπορεί να συμβούν κατά την εισαγωγή δεδομένων είναι:

- Να λείπουν υποχρεωτικές τιμές σε γνωρίσματα-στήλες που ορίστηκαν ως NOT NULL
- Η ύπαρξη διπλότυπων (duplicate) τιμών σε γνωρίσματα-στήλες που ορίστηκαν μοναδικά ώστε να παραβιάζονται περιορισμοί μοναδικότητας (unique constraints)
- Να παραβιάζονται περιορισμοί ζένων κλειδιών ή άλλου τύπου περιορισμού.
- Να εισάγονται τιμές διαφορετικού τύπου από τον τύπο τιμών που ορίστηκε το γνώρισμα-στήλη
- Να εισάγονται τιμές που ζεπερνούν τον μέγιστο αριθμό χαρακτήρων συγκρινόμενες με τον τύπο του γνωρίσματος-στήλης όπως φαίνεται στον ορισμό του πίνακα.

6.1.1.2. INSERT με χρήση φωλιασμένης ερώτησης

Μια παραλλαγή της εντολής INSERT είναι αυτή με τη χρήση φωλιασμένης ερώτησης. Η εντολή αυτή χρησιμοποιείται συνήθως για να εισάγει πολλαπλές πλειάδες-εγγραφές δεδομένων σε έναν πίνακα. Οι πλειάδες αυτές προκύπτουν ως αποτέλεσμα ερωτήματος σε άλλον πίνακα. Η INSERT έχει την μορφή:

INSERT INTO *tbl_name1* [(*column 1* [, ... *column n*])] <ΕΡΩΤΗΣΗ>

- *tbl_name1*: Το όνομα του πρώτου πίνακα

- *column 1 ... column n* : Το όνομα του γνωρίσματος του πρώτου πίνακα

ΕΡΩΤΗΣΗ:

SELECT [(*column 1* [, ... *column n*])]

FROM *tbl_name2*

[WHERE <condition(s)>]

- *tbl_name2*: Το όνομα του δεύτερου πίνακα
- *column 1 ... column n* : Το όνομα του γνωρίσματος του δεύτερου πίνακα

Ο αριθμός καθώς και ο τύπος των γνωρισμάτων –στήλες που αναφέρονται στην INSERT καθώς και ο αριθμός και ο τύπος των γνωρισμάτων που αναφέρονται στην SELECT πρέπει να είναι ίδιος (ένα προς ένα).

Παραδειγμάτων

Ας θεωρήσουμε ότι ορίζουμε τον πίνακα OLD_DEPARTMENTS ως εξής:

```
CREATE TABLE OLD_DEPARTMENTS (
  Id      int not null,
  department_name    varchar(100) not null);
```

Κατόπιν θα γεμίσουμε τον πίνακα OLD_DEPARTMENTS με όλες τις εγγραφές του πίνακα DEPARTMENTS, εκτελώντας ερώτημα στον πίνακα DEPARTMENTS. Η εντολή INSERT γράφεται ως εξής:

```
INSERT INTO OLD_DEPARTMENTS (id, department_name)
SELECT id, department_name
FROM DEPARTMENTS;
```

Αν θέλουμε να εισάγουμε στον πίνακα OLD_DEPARTMENTS μόνο τα τμήματα του πίνακα DEPARTMENTS που ανήκουν στην Διεύθυνση Διοικητικού-Οικονομικού (directorate_id=2) προσθέτουμε την συνθήκη με την χρήση του όρου WHERE, τότε η εντολή INSERT γίνεται:

```
INSERT INTO OLD_DEPARTMENTS (id, department_name)  
SELECT id, department_name  
FROM DEPARTMENTS  
WHERE directorate_id=2;
```

6.1.1.3. INSERT με χρήση του όρου <set>

Παραλλαγή στην σύνταξη της INSERT είναι η μορφή της εντολής με την χρήση του όρου set. Η μορφή αυτή της εντολής INSERT υποστηρίζεται μόνο από την MySQL - δεν περιλαμβάνεται στο SQL standard. Η γενική μορφή της εντολής είναι:

```
INSERT INTO tbl_name  
SET column 1 = value 1 [, ... column n = value n]  
• tbl_name: Το όνομα του πίνακα  
• column 1 ... column n : Τα ονόματα των γνωρισμάτων-στηλών του πίνακα  
• value 1 ... value n: είναι οι τιμές των αντίστοιχων γνωρισμάτων-στηλών
```

Δεν υπάρχει WHERE στην INSERT.

Σε αυτή την μορφή της εντολής τα γνωρίσματα-στήλες αντιστοιχίζονται ένα προς ένα με τις τιμές τους με την χρήση του όρου set. Δεν χρειάζεται να περικλείονται τα γνωρίσματα-στήλες σε παρενθέσεις όμως η κάθε ανάθεση τιμής σε γνώρισμα-στήλη χωρίζεται με κόμμα (,). Το κύριο πλεονέκτημα της χρήσης αυτή της μορφής του INSERT είναι ότι απλοποιεί την αντιστοίχιση γνωρίσματος-στήλης με την τιμή της χωρίς να χρειάζεται να ελέγχουμε κάθε φορά τις παρενθέσεις και την ορθή σειρά των γνωρισμάτων και τιμών τους. Το μειονέκτημα αυτή της μορφής είναι ότι δεν παρέχεται η δυνατότητα να εισάγουμε περισσότερες της μιας εγγραφές στον πίνακα μας.

Παράδειγμα

Για παράδειγμα ας θεωρήσουμε τον πίνακα DEPARTMENTS και ας εκτελέσουμε το πρώτο παράδειγμα (**Error! Reference source not found.**) με την νέα σύνταξη της εντολής INSERT. Υπενθυμίζουμε ότι στο παράδειγμα αυτό εισάγουμε το Τμήμα

Πρωτοκόλλου που ανήκει στην Διεύθυνση Διοικητικού –Οικονομικού (directorate_id=2). Η εντολή INSERT γράφεται:

```
INSERT INTO DEPARTMENTS
SET id = 6,
department_name = 'ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ',
building_address = 'ΚΤΙΡΙΟ Α',
employee_number = 5,
directorate_id = 2;
```

6.1.2. Εντολή REPLACE

Επιπρόσθετα με την εντολή INSERT για την εισαγωγή δεδομένων η MySQL υποστηρίζει και την εντολή REPLACE. Η εντολή REPLACE της MySQL αποτελεί επέκταση του SQL standard. (Η SQL δεν υποστηρίζει την εντολή REPLACE). Η εντολή REPLACE μοιάζει με την εντολή INSERT. Η μόνη κύρια διαφορά τους είναι στο πως συμπεριφέρονται στην ύπαρξη πρωτεύοντος κλειδιού (primary key) ή στην ύπαρξη μοναδικών κλειδιών (unique keys). Στην INSERT αν προσπαθήσουμε να εισάγουμε μια πλειάδα δεδομένων που περιέχει πρωτεύον κλειδί ή μοναδικό κλειδί το οποίο ήδη περιέχεται στον πίνακα, η εισαγωγή αυτών των δεδομένων δεν θα πραγματοποιηθεί. Το ΣΔΒΔ θα αποτρέψει μια τέτοια εισαγωγή με την εμφάνιση κατάλληλου μηνύματος λάθους. Αντίθετα η εντολή REPLACE διαγράφει τα δεδομένα του πίνακα με αυτό το πρωτεύον κλειδί ή μοναδικό κλειδί και εισάγει την νέα πλειάδα δεδομένων στο ίδιο πρωτεύον κλειδί ή μοναδικό κλειδί. Για να χρησιμοποιήσουμε την εντολή REPLACE πρέπει να έχουμε τουλάχιστον INSERT και DELETE δικαιώματα.

6.1.2.1. REPLACE με χρήση του όρου <values>

Η εντολή REPLACE ακολουθεί την ίδια μορφή σύνταξης με την INSERT. Αρχικά θα δούμε την σύνταξη της REPLACE με την χρήση του όρου <values>:

```
REPLACE INTO tbl_name [(column 1 [, ... column n])] VALUES (value 1 [, ...
value n] );
```

- *tbl_name*: Το όνομα του πίνακα
- *column 1 ... column n* : Τα ονόματα των γνωρισμάτων-στηλών του πίνακα
- *value 1 ... value n*: είναι οι τιμές των αντίστοιχων γνωρισμάτων-στηλών

Τα γνωρίσματα-στήλες του πίνακα που δεν εμφανίζονται στην REPLACE παίρνουν είτε την προκαθορισμένη τιμή τους (default value) αν κατά την δήλωση τους στην δημιουργία του πίνακα ορίστηκαν προκαθορισμένες τιμές γι αυτά, είτε παίρνουν την κενή (NULL) τιμή.

Όμοια με την INSERT στην εντολή REPLACE η λίστα των γνωρισμάτων-στηλών στην παραπάνω σύνταξη μπορεί να μην χρησιμοποιηθεί. Αρκεί όμως η σειρά των τιμών που παρατίθενται να είναι ακριβώς η σειρά των αντίστοιχων γνωρισμάτων όπως αυτά δηλώθηκαν στην δημιουργία του πίνακα. Η σύνταξη της REPLACE χωρίς την λίστα των γνωρισμάτων-στηλών είναι:

REPLACE INTO *tbl_name* VALUES (*value 1* [, ... *value n*]);

- *tbl_name*: Το όνομα του πίνακα
- *value 1 ... value n*: είναι η τιμή των αντίστοιχων γνωρισμάτων – στήλη

Παράδειγμα

Ας θεωρήσουμε τον πίνακα DEPARTMENTS. Επιθυμούμε να εισαγάγουμε ένα νέο τμήμα με την επωνυμία Τμήμα Μισθοδοσίας με την χρήση της εντολής REPLACE.

**REPLACE INTO *DEPARTMENTS* VALUES (*id*, *department_name*,
directorate_id) VALUES(9, ‘ΤΜΗΜΑ ΜΙΣΘΟΔΟΣΙΑΣ’, 2)**

Το id αποτελεί το πρωτεύον κλειδί του πίνακα DEPARTMENTS. Αν εφαρμόσουμε την παραπάνω εντολή και το id με τιμή 9 δεν υπάρχει στον πίνακα, τότε η εντολή REPLACE θα εισάγει την πλειάδα που ορίζεται στην εντολή. Αν όμως το id με τιμή 9 υπάρχει ήδη στον πίνακα DEPARTMENTS τότε η εφαρμογή της παραπάνω εντολής REPLACE αφού διαγράψει πρώτα την εγγραφή του πίνακα με id 9 κατόπιν θα εισαγάγει τη νέα πλειάδα δεδομένων στο ίδιο πρωτεύον κλειδί δηλαδή στο id 9.

Αποτέλεσμα αυτού είναι να επανεγγραφούν τα νέα δεδομένα του πίνακα στο ίδιο πρωτεύον κλειδί και να χαθούν τα παλιά δεδομένα.

6.1.2.2. REPLACE με χρήση του όρου <set>

Η εντολή REPLACE με χρήση του όρου <set> ακολουθεί την ίδια μορφή σύνταξης με την INSERT και την χρήση του όρου <set>. Η χρήση της μορφής αυτής έχει τα ίδια πλεονεκτήματα και μειονεκτήματα που περιγράψαμε στην εντολή INSERT με την χρήση του ίδιου όρου <set>. Η γενική της μορφή είναι:

REPLACE INTO tbl_name

SET column 1 = value 1 [, ... column n = value n]

- ***tbl_name: Το όνομα του πίνακα***
- ***column 1 ... column n : Τα ονόματα των γνωρισμάτων-στηλών του πίνακα***
- ***value 1 ... value n: είναι οι τιμές των αντίστοιχων γνωρισμάτων-στηλών***

Τα γνωρίσματα-στήλες του πίνακα που δεν εμφανίζονται σε αυτή τη μορφή της REPLACE παίρνουν είτε την προκαθορισμένη τιμή τους (default value) αν κατά την δήλωση τους στην δημιουργία του πίνακα ορίστηκαν προκαθορισμένες τιμές γι αυτά, είτε παίρνουν την κενή τιμή δηλαδή NULL.

Ας εκτελέσουμε το ίδιο παράδειγμα με αυτό της προηγούμενης παραγράφου προκειμένου να δούμε τις διαφορές στις δύο εναλλακτικές συντάξεις της εντολής REPLACE.

Παράδειγμα

Θεωρούμε τον πίνακα DEPARTMENTS και επιθυμούμε να εισαγάγουμε το νέο τμήμα με την επωνυμία Τμήμα Μισθοδοσίας με την χρήση της εντολής REPLACE και της επιλογής <set>.

REPLACE INTO DEPARTMENTS

SET building_address = 'KTIPIO A',

```
employee_number = 10 ,  
department_name='ΤΜΗΜΑ ΜΙΣΘΟΔΟΣΙΑΣ',  
id=10;
```

Όμοια με την προηγούμενη μορφή σύνταξης, αφού το id αποτελεί το πρωτεύον κλειδί του πίνακα DEPARTMENTS το ΣΔΒΔ κατά την εκτέλεση της εντολής REPLACE, ελέγχει αν υπάρχει εγγραφή στον πίνακα με id=10, αν δεν υπάρχει θα εισαγάγει το νέο Τμήμα Μισθοδοσίας στον πίνακα DEPARTMENTS. Διαφορετικά, εάν δηλαδή υπάρχει εγγραφή στον πίνακα με id=10, η εκτέλεση της εντολής REPLACE θα διαγράψει πρώτα την εγγραφή με id=10 και κατόπιν θα εισαγάγει στο ίδιο πρωτεύον κλειδί (δηλαδή id=10) την νέα πλειάδα δεδομένων όπως αυτή ορίζεται στον όρο <set>.

Παραλλάσσοντας την προηγούμενη εντολή REPLACE, διαγράφοντας δηλαδή το γνώρισμα-στήλη department_name='ΤΜΗΜΑ ΜΙΣΘΟΔΟΣΙΑΣ', στον όρο <set>, και εκτελώντας την εντολή στον MySQL Client:

REPLACE INTO DEPARTMENTS

```
SET building_address = 'ΚΤΙΡΙΟ Α',  
employee_number = 10 ,  
id=10;
```

Θα λάβουμε το μήνυμα λάθους:

Error Code 1364. Field 'department_name' doesn't have a default value.

Κατά την δημιουργία του πίνακα DEPARTMENTS το γνώρισμα-στήλη department_name στον πίνακα έχει δηλωθεί ως γνώρισμα NOT NULL αλλά δεν έχει αποδοθεί προκαθορισμένη αρχική τιμή. Αφού το γνώρισμα department_name δεν εμφανίζεται στην εντολή REPLACE του παραδείγματος, το ΣΔΒΔ ελέγχει αν έχει καθοριστεί σε αυτό προκαθορισμένη τιμή (default value). Αν δεν βρει μια προκαθορισμένη τιμή το ΣΔΒΔ εμφανίζει το μήνυμα λάθους που είδαμε.

Συμβουλή

- Αν αναπτύσσετε μια εφαρμογή η οποία θέλετε δυναμικά να υποστηρίζει και άλλες ΒΔ εκτός από την MySQL τότε αποφύγετε την χρήση της REPLACE αφού δεν υποστηρίζεται από άλλα ΣΔΒΔ. Αντί γι αυτήν χρησιμοποιείστε συνδυασμό

INSERT και DELETE εντολών.

- Στην περίπτωση που θέλετε να ενημερώσετε δεδομένα χρησιμοποιείστε την εντολή UPDATE (περιγράφεται στην επόμενη παράγραφο) γιατί εκτελείται γρηγορότερα από ότι η εντολή REPLACE.

6.2. Ενημέρωση Δεδομένων

6.2.1. Εντολή Update

Εκτός από την εισαγωγή δεδομένων πολλές φορές είναι απαραίτητη η τροποποίηση των δεδομένων μας, δηλαδή η αλλαγή της τιμής μιας ή περισσότερων εγγραφών ενός πίνακα ή περισσότερων πινάκων σε μια ΒΔ. Για το σκοπό αυτό χρησιμοποιούμε την εντολή UPDATE της SQL. Όπως και στην περίπτωση εισαγωγής δεδομένων έτσι και στην ενημέρωση δεδομένων το ΣΔΒΔ ελέγχει την ικανοποίηση των περιορισμών ακεραιότητας που έχουν καθορισθεί στην δημιουργία των πινάκων της ΒΔ. Σε περίπτωση που κάποιος περιορισμός δεν ικανοποιείται η ενημέρωση δεδομένων αποτυγχάνει. Για να χρησιμοποιήσουμε την εντολή UPDATE πρέπει να έχουμε τουλάχιστον UPDATE δικαιώματα. Η σύνταξη της εντολής UPDATE στην γενική της μορφή είναι:

UPDATE *tbl_name*

SET column 1 = value 1 [, ... column n = value n]

[WHERE condition];

- ***tbl_name: Το όνομα του πίνακα***
- ***column 1 ... column n : Τα ονόματα των γνωρισμάτων-στηλών του πίνακα***
- ***value 1 ... value n: είναι οι τιμές των αντίστοιχων γνωρισμάτων – στηλών***
- ***condition: συνθήκη τροποποίησης***

Η εντολή UPDATE μπορεί να ενημερώσει περισσότερες της μιας εγγραφές σε έναν πίνακα. Το πόσες εγγραφές ενημερώνει η εντολή καθορίζεται από την συνθήκη τροποποίησης (WHERE condition). Συγκεκριμένα, στην περίπτωση που ο τελεστής WHERE υπάρχει στην εντολή, η UPDATE βρίσκει πρώτα όλες τις εγγραφές του πίνακα

tbl_name για τις οποίες η συνθήκη «condition» είναι αληθής και μετά ενημερώνει τα γνωρίσματα column 1 ... column n με τις τιμές που αναφέρονται στην εντολή, value 1 ... value n, αντίστοιχα. Η παράλειψη του τελεστή WHERE επιφέρει τροποποίηση σε όλες τις έγγραφές του πίνακα.

Παράδειγμα

Λόγω οργανωτικών αλλαγών σε έναν δημόσιο φορέα ο υπάλληλος ΜΑΝΩΛΗΣ ΠΑΠΑΔΟΠΟΥΛΟΣ που υπηρετούσε στο Τμήμα Πληροφορικής (department_id=1) μετατέθηκε στο Τμήμα Προσωπικού (department_id=2). Για να πραγματοποιήσουμε αυτή την ενημέρωση θα χρησιμοποιήσουμε την εντολή UPDATE. Αν λάβουμε υπόψη ότι οι υπάλληλοι του φορέα διατηρούνται στον πίνακα EMPLOYEES της ΒΔ μας, η ενημέρωση θα πρέπει να πραγματοποιηθεί πάνω στον πίνακα EMPLOYEES (Εικόνα 34 Ο πίνακας EMPLOYEES περιέχει τους υπαλλήλους του φορέα)

	id	firstname	lastname	age	salary	grade	department_id	manager_id	birthdate
▶	1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	38	1200	B	1	2	1976-02-01
	2	ΝΙΚΟΣ	ΤΑΣΟΠΟΥΛΟΣ	40	2000	A	1	NULL	1972-04-03
	3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2	NULL	1968-04-01
	4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	Δ	2	3	1978-08-09
	5	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3	NULL	1978-10-09
	6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2000	A	1	2	1968-11-23

Εικόνα 34 Ο πίνακας EMPLOYEES περιέχει τους υπαλλήλους του φορέα

Η μορφή της εντολής UPDATE είναι:

```
UPDATE      EMPLOYEES
SET          DEPARTMENT_ID=2
WHERE        FIRSTNAME='ΜΑΝΩΛΗΣ' AND
LASTNAME='ΠΑΠΑΔΟΠΟΥΛΟΣ';
```

Η χρήση του όρου WHERE και της συνθήκης πραγματοποιείται για να περιορίσουμε την ενημέρωση στις εγγραφές του πίνακα EMPLOYEES σε εκείνες μόνο που το FIRSTNAME είναι ΜΑΝΩΛΗΣ και το LASTNAME είναι ΠΑΠΑΔΟΠΟΥΛΟΣ. Στην περίπτωση που δεν χρησιμοποιηθεί η συνθήκη και γενικά ο τελεστής WHERE η εκτέλεση της παραπάνω εντολής θα εφαρμοσθεί σε όλες τις εγγραφές του πίνακα EMPLOYEES τροποποιώντας την τιμή του γνωρίσματος-στήλη department_id στην τιμή 2 σε όλες τις εγγραφές του πίνακα EMPLOYEES. Ουσιαστικά θα ενημερώσει

όλους των υπαλλήλους του Φορέα ότι ανήκουν στο Τμήμα Προσωπικού. Η πράξη αυτή θα επέφερε μεγάλη αλλοίωση των δεδομένων. Επομένως πρέπει να είμαστε ιδιαίτερα προσεκτικοί στην χρήση της εντολής UPDATE ώστε η συνθήκη που εισάγουμε στην εντολή να είναι η ορθή κάθε φορά.

Παρακάτω θα δούμε παράδειγμα ενημέρωσης περισσότερων γνωρισμάτων—στηλών σε έναν πίνακα.

Παράδειγμα

Ενημερώστε τον υπάλληλο ΜΑΝΩΛΗ ΠΑΠΑΔΟΠΟΥΛΟ ώστε να ανήκει στο Τμήμα Πληροφορικής (department_id=1) και ο μισθός του (salary) να γίνει 2500 ευρώ. Σημειώστε το id του Τμήματος Πληροφορικής είναι 1. Η εντολή UPDATE που θα ενημερώσει τα γνωρίσματα-στήλες (department_id, salary) του πίνακα EMPLOYEES είναι:

```
UPDATE      EMPLOYEES
SET          DEPARTMENT_ID=1,
            SALARY=2500
WHERE        FIRSTNAME='ΜΑΝΩΛΗΣ' AND
            LASTNAME='ΠΑΠΑΔΟΠΟΥΛΟΣ';
```

Τα γνωρίσματα στον όρο SET που ενημερώνονται χωρίζονται με κόμμα (,).

6.3. Διαγραφή Δεδομένων

Πολλές φορές στην διαχείριση των δεδομένων μιας ΒΔ είναι απαραίτητη η μερική ή η ολική διαγραφή δεδομένων σε έναν πίνακα. Η διαγραφή όμως μπορεί να διαδοθεί και σε άλλους πίνακες αν έχουν οριστεί κατάλληλοι περιορισμοί ακεραιότητας. Οι εντολές που χρησιμοποιούνται για τον σκοπό αυτό είναι η εντολή DELETE και η εντολή TRUNCATE.

6.3.1. Εντολή DELETE

Η εντολή DELETE διαγράφει εγγραφές από τον πίνακα που εμείς ορίζουμε στην εντολή. Εγγραφές διαγράφονται ρητά από έναν μόνο πίνακα κάθε φορά με την εκτέλεση

της εντολής. Αν θέλουμε να διαγράψουμε δεδομένα σε περισσότερους πίνακες τότε θα πρέπει να εφαρμόσουμε την εντολή σε κάθε πίνακα ξεχωριστά. Με την εντολή DELETE μπορούμε να διαγράψουμε μόνο ολόκληρη εγγραφή και δεν μπορούμε να διαγράψουμε τιμές μόνο από συγκεκριμένα γνωρίσματα-στήλες. Η γενική μορφή της εντολής DELETE στην SQL είναι:

DELETE FROM tbl_name

[WHERE condition];

- *tbl_name: Το όνομα του πίνακα*
- *condition: συνθήκη διαγραφής, καθορίζει τον αριθμό γραμμών που θα διαγραφούν*

Στην περίπτωση που ο τελεστής WHERE υπάρχει στην εντολή, η DELETE βρίσκει πρώτα όλες τις εγγραφές του πίνακα *tbl_name* για τις οποίες η συνθήκη «*condition*» είναι αληθής και μετά τις διαγράφει. Η παράλειψη του τελεστή WHERE οδηγεί στην διαγραφή όλων των εγγραφών του πίνακα *tbl_name*.

Παράδειγμα

Ο πίνακας DEPARTMENTS περιλαμβάνει τα τμήματα του οργανισμού ενός φορέα (Εικόνα 35 Πίνακας DEPARTMENTS).

	id	department_name	building_address	employee_number	directorate_id	inseitdate
▶	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	KΤΙΡΙΟ Α	10	1	NULL
▶	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	KΤΙΡΙΟ Α	12	2	NULL
▶	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ	KΤΙΡΙΟ Α	1	2	NULL
▶	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ	KΤΙΡΙΟ Α	0	1	NULL
▶	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΕΩΝ	KΤΙΡΙΟ Α	0	2	NULL
▶	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ	KΤΙΡΙΟ Α	5	2	NULL
▶	7	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ	NULL	NULL	2	NULL

Εικόνα 35 Πίνακας DEPARTMENTS

Ας υποθέσουμε ότι ο οργανισμός αλλάζει και το Τμήμα Προμηθειών καταργείται. Για να ενημερώσουμε τον πίνακα DEPARTMENTS ώστε να περιέχει τα σωστά τμήματα πρέπει να εκτελέσουμε την εντολή DELETE:

DELETE FROM DEPARTMENTS

WHERE department_name='ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ';

Επιτρόσθετα, σε πολλές περιπτώσεις η διαγραφή μιας εγγραφής σε έναν πίνακα μας δεν επιτρέπεται αν υπάρχει ξένο κλειδί (foreign key) σε άλλον πίνακα που αναφέρεται στην εγγραφή αυτή. Στην περίπτωση αυτή θα πρέπει πρώτα να διαγραφεί ή να τροποποιηθεί η εγγραφή που περιέχει την αναφερόμενη τιμή στον άλλον πίνακα και κατόπιν να διαγραφεί η εγγραφή στον πίνακά μας. Η περίπτωση αυτή εμφανίζεται αν το ξένο κλειδί (foreign key) στον άλλον πίνακα που περιέχει την αναφερόμενη τιμή έχει δημιουργηθεί χωρίς την επιλογή Διάδοση Διαγραφής (DELETE CASCADE). Σε περίπτωση που στο ξένο κλειδί (foreign key) έχει ορισθεί Διάδοση Διαγραφής (DELETE CASCADE) τότε διαγράφοντας την εγγραφή στον πίνακά μας αυτόματα θα πραγματοποιηθούν διαγραφές και στις εγγραφές του άλλου πίνακα (στον οποίο έχει οριστεί το ξένο κλειδί (foreign key)) οι οποίες έχουν τιμή ίδια με αυτή που διαγράφουμε στον πίνακά μας.

Παραδείγμα

Ας υποθέσουμε ότι θέλουμε να καταργήσουμε το Τμήμα Πληροφορικής από τον Οργανισμό του Φορέα μας. Αυτό που θα πρέπει να κάνουμε ώστε να ενημερωθεί η ΒΔ μας είναι να διαγράψουμε την εγγραφή με id =1 του πίνακα DEPARTMENTS (Εικόνα 35 Πίνακας DEPARTMENTS). Ουσιαστικά θα πρέπει να εκτελέσουμε την εντολή:

DELETE FROM DEPARTMENTS

WHERE department_name='ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ';

Επίσης σημειώστε ότι ο πίνακας EMPLOYEES (**Error! Reference source not found.**) περιέχει το γνώρισμα-στήλη department_id. Κατά την δημιουργία του πίνακα ορίστηκε ξένο κλειδί (foreign key) στο γνώρισμα-στήλη department_id προς τον πίνακα DEPARTMENTS χωρίς πράξη Διαγραφής (DELETE action). Ελέγχοντας τα δεδομένα του πίνακα EMPLOYEES παρατηρούμε ότι περιλαμβάνει υπαλλήλους που ανήκουν στο Τμήμα Πληροφορικής (department_id=1). Η εκτέλεση της παραπάνω εντολής θα αποτύχει αφού το id = 1 που απεικονίζει το Τμήμα Πληροφορικής υπάρχει ως

αναφερόμενη τιμή στον πίνακα EMPLOYEES (department_id=1). Εκτελώντας την παραπάνω εντολή στον MySQL Client μας εμφανίζει το ακόλουθο μήνυμα λάθους:

```
Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails  
(`employees`, CONSTRAINT `fk_department_id` FOREIGN KEY (`department_id`)  
REFERENCES `departments` (`id`) ON DELETE NO ACTION ON UPDATE NO  
ACTION)0.000 sec.
```

Αν κατά την δημιουργία του πίνακα EMPLOYEES είχαμε ορίσει το ξένο κλειδί (foreign key) προς τον πίνακα DEPARTMENTS με Διάδοση Διαγραφής (DELETE CASCADE action) τότε η παραπάνω εντολή δεν θα αποτύχει και μαζί με το Τμήμα Πληροφορικής του πίνακα DEPARTMENTS θα διέγραψε και τους υπαλλήλους στον πίνακα EMPLOYEES που άνηκαν στο Τμήμα Πληροφορικής εκείνους δηλαδή με department_id=1.

6.3.2. Εντολή TRUNCATE

Η τελευταία εντολή αυτής της ενότητας που θα περιγράψουμε είναι η εντολή TRUNCATE. Η TRUNCATE διαγράφει όλες τις εγγραφές ενός πίνακα. Η εντολή TRUNCATE παρόλο που περιγράφεται σε αυτή την ενότητα δεν ανήκει στις DML εντολές. Η εντολή TRUNCATE είναι μία DDL εντολή. Η σύνταξή της είναι:

```
TRUNCATE TABLE tbl_name ;
```

- *tbl_name: Το όνομα του πίνακα του οποίου τα δεδομένα θα διαγραφούν*

Η εντολή TRUNCATE διαγράφει όλες τις γραμμές-εγγραφές ενός πίνακα, αφήνοντας τον πίνακα άδειο αλλά την δομή του ανέπαφη. Η εντολή TRUNCATE είναι πιο γρήγορη από την εντολή DELETE αλλά δεν είναι εύκολο να επαναφέρουμε τα δεδομένα που διαγράφηκαν με την εντολή αυτή. Επίσης αν ο πίνακας έχει γνώρισμα-στήλη AUTO_INCREMENT η εντολή TRUNCATE επαναφέρει την τιμή του auto_increment στο 0 αντίθετα από την DELETE η οποία δεν επηρεάζει την τιμή του auto_increment.

Αν έχουν δηλωθεί ξένα κλειδιά (foreign keys) που αναφέρονται στον πίνακα (A) που εκτελούμε την εντολή TRUNCATE, η εντολή ενεργεί ως εξής:

- Αν τα ξένα κλειδιά (foreign keys) στους άλλους πίνακες της ΒΔ μας έχουν ορισθεί με Διάδοση Διαγραφής (DELETE CASCADE action) τότε διαγράφονται και οι εγγραφές στους πίνακες αυτούς με τιμή γνωρίσματος την αναφερόμενη τιμή του γνωρίσματος στον πίνακα (A).
- Αν στον ορισμό των ξένων κλειδιών (foreign keys) στους άλλους πίνακες της ΒΔ μας δεν δηλωθεί Διάδοση Διαγραφής (DELETE CASCADE action), τότε η TRUNCATE διαγράφει όλες τις εγγραφές του πίνακα (A) στον οποίο αναφέρονται τα ξένα κλειδιά και εμφανίζει μήνυμα λάθους όταν υπάρξει εγγραφή του (A) με τιμή γνωρίσματος-στήλη η οποία εμφανίζεται σε τουλάχιστον έναν από τους άλλους πίνακες.

Στην MySQL από την έκδοση 5.0.3 και μετά στις ΒΔ που έχουν υλοποιηθεί με χρήση της μηχανής αποθήκευσης InnoDB, αν δεν έχουν δηλωθεί ξένα κλειδιά (foreign keys) που να αναφέρονται στον πίνακα, η TRUNCATE κάνει drop τον πίνακα δηλαδή διαγράφει όχι μόνο τα δεδομένα του πίνακα αλλά και ολόκληρη την δομή του και δημιουργεί έναν νέο πίνακα με την ίδια δομή (εκ νέου CREATE TABLE).

7. Συναλλαγές (Transactions)

Μελετώντας το παρόν κεφάλαιο οι επιμορφωνόμενοι θα είναι σε θέση να:

- κατανοήσουν την έννοια της Συναλλαγής (Transaction) και τις ιδιότητές της.
- εξοικειωθούν με τις εντολές μιας συναλλαγής (transaction).
- γνωρίσουν τα πιθανά Επίπεδα Απομόνωσης (Isolation Levels) μιας συναλλαγής (transaction) και να κατανοήσουν τις διαφορές τους.
- κατανοήσουν το κλείδωμα σε επίπεδο γραμμών – rows (Row Level Locking) ενός πίνακα.

Στο προηγούμενο κεφάλαιο περιγράφηκε η γλώσσα διαχείρισης δεδομένων (DML) και παρουσιάστηκαν οι εντολές που περιέχονται στην γλώσσα και υποστηρίζονται από το SQL standard. Στο κεφάλαιο αυτό θα εξετάσουμε τις βασικές έννοιες διαχείρισης συναλλαγών.

7.1. Τι είναι η συναλλαγή (Transaction)

Μια συναλλαγή γενικά είναι μια συλλογή από διάφορες λειτουργίες όπου η μια ακολουθεί την άλλη και εκτελούνται σαν μια μονάδα. Μια συναλλαγή θα ολοκληρωθεί αν όλες οι λειτουργίες που περιλαμβάνει ολοκληρωθούν επιτυχώς. Αν μια λειτουργία δεν ολοκληρωθεί επιτυχώς τότε ολόκληρη η συναλλαγή αποτυγχάνει.

Για παράδειγμα στον τραπεζικό χώρο, η μεταφορά χρημάτων από έναν τραπεζικό λογαριασμό A σε έναν άλλο τραπεζικό λογαριασμό B είναι μια συναλλαγή. Η συναλλαγή αυτή περιλαμβάνει τρείς ξεχωριστές λειτουργίες:

- Την αφαίρεση χρημάτων που μεταφέρονται από τον τραπεζικό λογαριασμό A
- Την αύξηση των χρημάτων του τραπεζικού λογαριασμού B. Η αύξηση είναι ίση με το ποσό των χρημάτων που μεταφέρονται.
- Την εγγραφή της συναλλαγής στο Αρχείο συναλλαγών της τράπεζας.

Για την πραγματοποίηση της συναλλαγής «Μεταφορά χρημάτων από το λογαριασμό A στον λογαριασμό B» πρέπει να εκτελεστούν και οι τρεις λειτουργίες. Αν κάποια από τις λειτουργίες αποτύχει τότε αποτυγχάνει όλη η συναλλαγή και πρέπει να ακυρωθούν όλες οι προηγούμενες λειτουργίες που έχουν εκτελεστεί.

Στην SQL μια συναλλαγή είναι μια λειτουργική μονάδα εκτέλεσης μιας ακολουθίας εντολών διαχείρισης δεδομένων (DML εντολών) που εκτελείται σε μια βάση δεδομένων.

Στην MySQL μια συναλλαγή ξεκινά με:

- **START TRANSACTION;**
- ή **BEGIN WORK;**

Περιλαμβάνει μια λίστα από SQL DML εντολές και καταλήγει με:

- **COMMIT:** ολοκληρώνει την τρέχουσα συναλλαγή, δηλαδή κάνει μόνιμες τις ενημερώσεις που εκτελούνται από την συναλλαγή στην ΒΔ.
- ή **ROLLBACK:** αναιρεί την τρέχουσα συναλλαγή, δηλαδή ακυρώνει όλες τις ενημερώσεις που έγιναν από τις SQL εντολές της συναλλαγής. Έτσι η βάση δεδομένων επανέρχεται στην κατάσταση που βρισκόταν πριν την εκτέλεση της συναλλαγής.

START TRANSACTION; | BEGIN WORK;

DML εντολή; ... ;DML εντολή;

COMMIT; | ROLLBACK;

Όταν μια συναλλαγή εκτελεστεί με το COMMIT τα αποτελέσματά της δεν αναιρούνται πλέον με το ROLLBACK. Το ΣΔΒΔ διασφαλίζει ότι αν συμβεί μια αποτυχία όπως ένα λάθος σε μια DML εντολή, μια διακοπή ρεύματος, μια γενικότερη αστάθεια ή βλάβη του συστήματος, θα ακυρωθούν τα αποτελέσματα μιας συναλλαγής αν δεν έχει εκτελεστεί ακόμα το COMMIT της. Στην περίπτωση διακοπής ρεύματος ή άλλων προβλημάτων του συστήματος η ROLLBACK θα εκτελεστεί όταν πραγματοποιηθεί επανεκκίνηση στο σύστημα.

Η διαχείριση συναλλαγών αποτελεί ιδιαίτερα δύσκολη λειτουργία αφού πολλοί χρήστες ταυτόχρονα προσπαθούν να δουν ή να τροποποιήσουν τα δεδομένα μιας ΒΔ. Το αποτέλεσμα μπορεί να είναι ασυνεπή και ανακριβή δεδομένα. Για την αποφυγή τέτοιων προβλημάτων η διαχείριση συναλλαγών που υποστηρίζουν τα σύγχρονα ΣΔΒΔ απομονώνουν την κάθε λειτουργία πάνω στα δεδομένα της ΒΔ και βεβαιώνουν ακρίβεια και συνέπεια σε όλη την ΒΔ.

7.2.Εντολές συναλλαγών

Όπως ήδη αναφέραμε η σύνταξη μιας συναλλαγής στην MySQL είναι:

Μια συναλλαγή ξεκινά με START TRANSACTION; ή BEGIN WORK; Κατόπιν περιλαμβάνει μια λίστα από SQL DML εντολές και τέλος καταλήγει με:

- COMMIT : ολοκληρώνει την τρέχουσα συναλλαγή
- ή ROLLBACK: αναιρεί την τρέχουσα συναλλαγή

Παράδειγμα

Ας δούμε μια συναλλαγή που καταλήγει με COMMIT:

```
START TRANSACTION;
INSERT INTO EMPLOYEES (firstname, lastname, age, salary, department_id)
VALUES ('ΦΙΛΙΠΠΟΣ', 'ΝΙΚΟΛΑΟΥ' , 25, 1000, 1);

UPDATE EMPLOYEES SET salary=2500 WHERE id=6;
COMMIT;
```

Η συναλλαγή ξεκινά με START TRANSACTION. Περιλαμβάνει δύο (2) DML εντολές μία INSERT και μία UPDATE. Και οι δύο DML εντολές εφαρμόζονται στον πίνακα EMPLOYEES. Αν υποθέσουμε ότι και οι δύο εντολές INSERT και UPDATE εκτελούνται επιτυχώς τότε η εντολή COMMIT που ακολουθεί αποθηκεύει όλες τις αλλαγές στην ΒΔ. Συγκεκριμένα στον πίνακα EMPLOYEES εισάγεται εγγραφή με τον υπάλληλο ΦΙΛΙΠΠΟ ΝΙΚΟΛΑΟΥ και ενημερώνεται ο μισθός σε 2500 ευρώ του υπαλλήλου με id=6. Εκτελώντας την εντολή SELECT στον πίνακα EMPLOYEES θα δούμε ότι όλες οι ενημερώσεις της συναλλαγής αποθηκεύτηκαν στον πίνακα EMPLOYEES.

```
SELECT * FROM EMPLOYEES;
```

Παράδειγμα

Ας δούμε μια συναλλαγή που καταλήγει με ROLLBACK.

BEGIN WORK;

```
INSERT INTO EMPLOYEES (firstname, lastname, age, salary, department_id)
VALUES ('ΓΕΩΡΓΙΟΣ', 'ΓΑΛΑΝΗΣ', 35, 1100, 1);
```

```
UPDATE EMPLOYEES SET salary=3500 WHERE id=6;
```

ROLLBACK;

Η συναλλαγή ξεκινά με BEGIN WORK. Θα μπορούσε όμως να ξεκινήσει με START TRANSACTION. Η MySQL υποστηρίζει και τις δύο επιλογές για την εκκίνηση της συναλλαγής δηλαδή START TRANSACTION; και BEGIN WORK;. Η συναλλαγή περιλαμβάνει δύο (2) DML εντολές μία INSERT και μία UPDATE. Και οι δύο DML εντολές εφαρμόζονται στον πίνακα EMPLOYEES. Αρχικά εισάγεται εγγραφή με τον υπάλληλο ΓΕΩΡΓΙΟ ΓΑΛΑΝΗ (INSERT εντολή) και κατόπιν με την εκτέλεση της δεύτερης εντολής UPDATE ενημερώνεται ο μισθός του υπαλλήλου με id=6 σε 3500 ευρώ. Αν υποθέσουμε ότι και οι δύο εντολές INSERT και UPDATE εκτελεστούν επιτυχώς, η εντολή ROLLBACK θα ακυρώσει όλες τις ενημερώσεις που έγιναν από τις SQL DML εντολές της συναλλαγής. Δηλαδή, η εγγραφή με τον ΓΕΩΡΓΙΟ ΓΑΛΑΝΗ θα διαγραφεί και η ενημέρωση του μισθού του υπαλλήλου με id=6 θα επανέλθει στην τιμή που είχε πριν την εκτέλεση της UPDATE. Εκτελώντας την εντολή SELECT πάνω στον πίνακα EMPLOYEES θα δούμε ότι ο πίνακας παρέμεινε ανέπαφος από την εκτέλεση της συναλλαγής.

```
SELECT * FROM EMPLOYEES;
```

Μια συναλλαγή ολοκληρώνεται όταν συμβεί κάτι από τα παρακάτω:

- η εκτέλεση των εντολών COMMIT ή ROLLBACK
- η εκτέλεση μιας DDL ή DCL εντολής. Οι εντολές DDL και DCL εκτελούν αυτόματα με το πέρας της εκτέλεσής του την εντολή COMMIT. Επομένως όλες οι ενημερώσεις που είχαν επιφέρει οι DML εντολές που περιλάμβανε η συναλλαγή πριν την εκτέλεση της εντολής DDL ή DCL αποθηκεύονται στην ΒΔ. Τέτοιες DDL ή DCL στην MySQL εντολές είναι:
 - ALTER TABLE: Τροποποιεί τον ορισμό του πίνακα.
 - CREATE INDEX: Δημιουργεί index σε έναν πίνακα.
 - DROP DATABASE: Διαγράφει μια ΒΔ από την MySQL.

- DROP INDEX: Διαγράφει έναν index από έναν πίνακα
- DROP TABLE: Διαγράφει ένα πίνακα από την ΒΔ.
- LOCK TABLES: Εμποδίζει την ταυτόχρονη πρόσβαση στους πίνακες.
- RENAME TABLES: Μετονομάζει τους πίνακες.
- SET AUTOCOMMIT=1: Θέτει το autocommit ενεργό (θα περιγραφεί σε επόμενη παράγραφο) .
- START TRANSACTION: Ξεκινά μια νέα συναλλαγή.
- TRUNCATE TABLE: Διαγράφει τα δεδομένα ενός πίνακα.
- UNLOCK TABLES: Ξεκλειδώνει τους κλειδωμένους πίνακες.

Επιπλέον καμία από τις παραπάνω εντολές δεν μπορεί να ανακληθεί.

- Ο χρήστης κλείνει τον MySQL Client. Στην περίπτωση αυτή η συναλλαγή δεν ολοκληρώνεται. Αυτόματα εκτελείται η εντολή ROLLBACK και όλες οι ενημερώσεις που είχαν πραγματοποιηθεί μέσα στην συναλλαγή ακυρώνονται.
- Το ΣΔΒΔ αποτυγχάνει και τερματίζει. Στην επανεκκίνηση του συστήματος εκτελείται η εντολή ROLLBACK και όλες οι ενημερώσεις που είχαν πραγματοποιηθεί από τις DML εντολές της συναλλαγής ακυρώνονται.

Τα πλεονεκτήματα της χρήσης των εντολών COMMIT και ROLLBACK είναι ότι μετά την εκτέλεση των DML εντολών μέσα στην συναλλαγή μας έχουμε την δυνατότητα καταρχήν να επιβεβαιώσουμε την συνέπεια των δεδομένων καθώς και επιπλέον να δούμε ακριβώς τις ενημερώσεις στα δεδομένα πριν την οριστική και μόνιμη αποθήκευσή τους στην ΒΔ.

7.2.1. Πρόσθεση SAVEPOINTS στις συναλλαγές

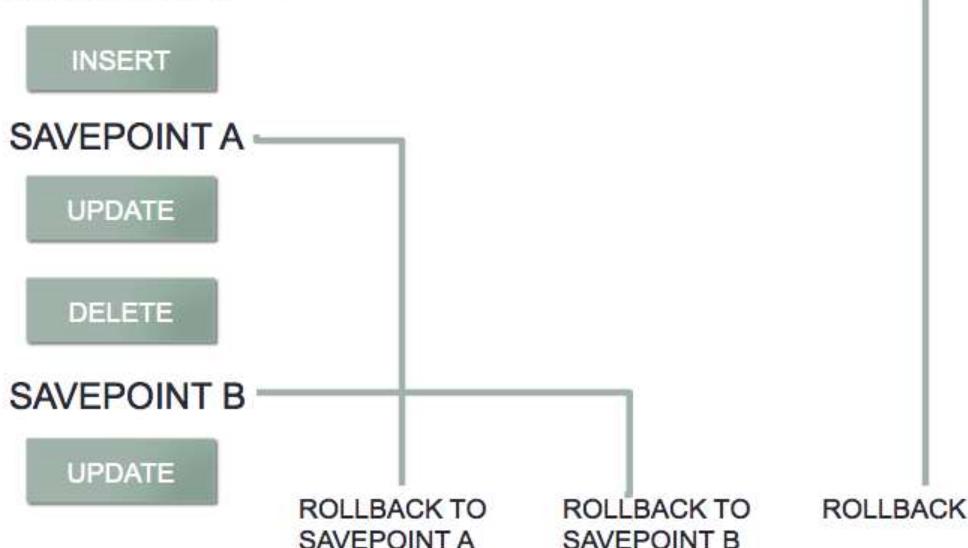
Στην προηγούμενη παράγραφο περιγράψαμε πως δημιουργείται μια συναλλαγή και πως ακριβώς χρησιμοποιούνται οι εντολές START TRANSACTION, COMMIT και ROLLBACK, στην παράγραφο αυτή θα δούμε τις εντολές SAVEPOINT, ROLLBACK TO SAVEPOINT και RELEASE SAVEPOINT που χρησιμοποιούνται για να απομονώσουμε μέρη της συναλλαγής μας. Η εντολή SAVEPOINT μας επιτρέπει να ορίσουμε ένα σημείο επαναφοράς στην συναλλαγή μας ώστε να έχουμε την δυνατότητα να επανέλθουμε στο σημείο αυτό. Η εντολή ROLLBACK TO SAVEPOINT μας επιτρέπει να ανακαλέσουμε όλες τις ενημερώσεις που έχουν πραγματοποιηθεί στην ΒΔ μας από τις DML εντολές που υπήρχαν στην συναλλαγή μας μέχρι το σημείο

επαναφοράς που δείχνει το SAVEPOINT της εντολής. Η εντολή RELEASE SAVEPOINT απελευθερώνει τον ορισμό του SAVEPOINT.

Το SAVEPOINT μέσα στην συναλλαγή μας είναι ουσιαστικά ένα σημάδι (marker) που δείχνει το σημείο στο οποίο μπορούμε να επανέλθουμε ακυρώνοντας την συναλλαγή μας μέχρι εκείνο το σημείο. Σε μεγάλες συναλλαγές με πολλαπλές DML εντολές η ικανότητα χρήσης τέτοιων δυνατοτήτων είναι πολλές φορές πολύτιμη, αφού παρέχουν ευελιξία και δυνατότητα ελέγχου της συνέπειας των δεδομένων που ενημερώνονται μέσα στην συναλλαγή μας. Οι συναλλαγές γίνονται καλύτερα διαχειρίσιμες.

Στο παρακάτω σχήμα (**Error! Reference source not found.**) βλέπουμε ένα παράδειγμα μιας συναλλαγής που περιλαμβάνει τις DML εντολές με την ακόλουθη σειρά INSERT, UPDATE, DELETE, UPDATE. Μεταξύ της INSERT και UPDATE έχει οριστεί το σημείο επαναφοράς SAVEPOINT A, ενώ μεταξύ της DELETE και τελευταίας UPDATE έχει οριστεί το σημείο επαναφοράς SAVEPOINT B. Στην περίπτωση που επιθυμούμε να επανέλθουμε στη συναλλαγή μας στο σημείο επαναφοράς SAVEPOINT B και να ακυρώσουμε την συναλλαγή μας μέχρι εκείνο το σημείο (ουσιαστικά ακυρώνεται η τελευταία εντολή UPDATE), εκτελούμε το ROLLBACK TO SAVEPOINT B. Αν θέλουμε να επαναφέρουμε την συναλλαγή μας μέχρι το σημείο SAVEPOINT A διατηρώντας μόνο τις ενημερώσεις της εντολής INSERT, εκτελούμε το ROLLBACK TO SAVEPOINT A. Ενώ αν επιθυμούμε να ακυρώσουμε ολόκληρη της συναλλαγή μας χρησιμοποιούμε το ROLLBACK.

ΣΥΝΑΛΛΑΓΗ



Εικόνα 36 Ορισμός SAVEPOINTS σε μια συναλλαγή

Για να ορίσουμε ένα SAVEPOINT ένα δηλαδή σημείο επαναφοράς μέσα στην συναλλαγή μας εκτελούμε την εντολή:

SAVEPOINT *identifier*;

- *identifier*: η ετικέτα του σημείου επαναφοράς

Για να ακυρώσουμε τις ενημερώσεις που πραγματοποιήθηκαν στην συναλλαγή μας μέχρι ένα SAVEPOINT εκτελούμε την εντολή:

ROLLBACK [WORK] TO [SAVEPOINT] *identifier*;

- *identifier*: η ετικέτα του σημείου επαναφοράς

Ενώ για να απελευθερώσουμε ένα SAVEPOINT ώστε να μην δείχνει πια κάποιο σημείο στην συναλλαγή μας εκτελούμε την εντολή:

RELEASE SAVEPOINT *identifier*;

- *identifier*: η ετικέτα του σημείου επαναφοράς

Όλα τα σημεία επαναφοράς - SAVEPOINTS διαγράφονται μετά από την εκτέλεση ROLLBACK ή COMMIT στην συναλλαγή μας.

Παράδειγμα

Ας δούμε ένα παράδειγμα συναλλαγής στην οποία έχουμε προσθέσει δύο σημεία επαναφοράς - SAVEPOINTS τα POINTA και POINTB. Τα δύο αυτά σημεία επαναφοράς – SAVEPOINTS παρέχουν δύο σημάδια που δείχνουν δύο σημεία της συναλλαγής μέχρι τα οποία μπορούμε να ανακαλέσουμε την συναλλαγή μας.

START TRANSACTION;

INSERT INTO EMPLOYEES (firstname, lastname, age, salary, department_id)

```
VALUES ('ΙΩΑΝΝΗΣ', 'ΠΕΤΡΑΚΟΣ', 45, 1600, 1);
```

```
SAVEPOINT POINTA;
```

```
DELETE FROM EMPLOYEES WHERE id=1;
```

```
SAVEPOINT POINTB;
```

```
UPDATE EMPLOYEES SET salary=2500 WHERE id=5;
```

Η συναλλαγή αποτελείται από τρεις (3) DML εντολές. Μία INSERT, η οποία εισάγει στον πίνακα EMPLOYEES τον ΙΩΑΝΝΗ ΠΕΤΡΑΚΟ, μία DELETE όπου διαγράφει την εγγραφή με id=1 την υπάλληλο ΜΑΡΙΑ ΜΠΕΚΙΑΡΗ και μία UPDATE όπου ενημερώνει τον μισθό της εγγραφής id=5 ΑΝΝΑΣ ΠΑΠΑ από 3500 σε 2500 ευρώ.

Αν εκτελέσουμε την παραπάνω συναλλαγή και δούμε τα δεδομένα του πίνακα EMPLOYEES (για να δούμε τα δεδομένα του πίνακα εκτελούμε: SELECT * FROM EMPLOYEES) θα παρατηρήσουμε ότι ο ΙΩΑΝΝΗΣ ΠΕΤΡΑΚΟΣ εισήχθη στον πίνακα, η εγγραφή με id=1 ΜΑΡΙΑ ΜΠΕΚΙΑΡΗ διαγράφηκε και στην εγγραφή με id=5 ΑΝΝΑ ΠΑΠΑ ενημερώθηκε ο μισθός της σε 2500 ευρώ (**Error! Reference source not found.**).

	id	firstname	lastname	age	salary	department_id	
	1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	38	1200	1	
	2	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	40	2000	1	
	3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	2	
	4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	2	
	5	ΑΝΝΑ	ΠΑΠΑ	35	3500	3	
	6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2000	1	
	8	ΦΙΛΙΠΠΟΣ	ΝΙΚΟΛΑΟΥ	25	1000	1	

	id	firstname	lastname	age	salary	departmer	
	2	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	40	2000	1	
	3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	2	
	4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	2	
	5	ΑΝΝΑ	ΠΑΠΑ	35	2500	3	
	6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2000	1	
	8	ΦΙΛΙΠΠΟΣ	ΝΙΚΟΛΑΟΥ	25	1000	1	
	9	ΙΩΑΝΝΗΣ	ΠΕΤΡΑΚΟΣ	45	1600	1	

Εικόνα 37 Πίνακας EMPLOYEES μετά την εκτέλεση της συναλλαγής

Στην συνέχεια εκτελούμε την εντολή ανάκλησης - ROLLBACK έως το σημείο επαναφοράς POINTB, γράφοντας:

ROLLBACK TO SAVEPOINT POINTB;

Η εντολή αυτή θα ακυρώσει την τελευταία από τις DML εντολές την UPDATE. Ελέγχοντας τα δεδομένα του πίνακα EMPLOYEES εκτελώντας SELECT * from EMPLOYEES παρατηρούμε ότι ο Μισθός (salary) της ΑΝΝΑΣ ΠΑΠΑ έγινε 3500 ευρώ που σημαίνει ότι η UPDATE εντολή ακυρώθηκε.

Κατόπιν εκτελούμε την εντολή ανάκλησης - ROLLBACK έως το σημείο επαναφοράς POINTA. Η εντολή ανάκλησης είναι:

ROLLBACK TO SAVEPOINT POINTA;

Η εντολή αυτή θα ακυρώσει την εντολή DELETE. Ελέγχοντας τα δεδομένα του πίνακα EMPLOYEES εκτελώντας SELECT * from EMPLOYEES παρατηρούμε ότι εμφανίστηκε η εγγραφή με id=1 η ΜΑΡΙΑ ΜΠΕΚΙΑΡΗ, η οποία είχε διαγραφεί.

Τέλος, αν εκτελέσουμε ανάκληση όλης της συναλλαγής με την εντολή:

ROLLBACK ;

και δούμε τα δεδομένα του πίνακα EMPLOYEES, ο ΙΩΑΝΝΗΣ ΠΕΤΡΑΚΟΣ εξαφανίστηκε. Αυτό σημαίνει ότι η εντολή INSERT της συναλλαγής, που τον εισήγαγε, ακυρώθηκε.

7.2.2. Αυτόματο COMMIT

Στην MySQL, λόγω προεπιλογής της (by default), όταν εκτελείται μία SQL εντολή, αυτόματα οι ενημερώσεις της αποθηκεύονται στην ΒΔ. Κάθε SQL εντολή μπορεί να θεωρηθεί ως μία συναλλαγή δίχως να χρειάζεται για την εκτέλεση της η εισαγωγή της πρότασης START TRANSACTION πριν από αυτήν και η εισαγωγή της πρότασης COMMIT μετά από αυτήν.

Η MySQL παρέχει την εντολή set autocommit για να ενεργοποιεί ή απενεργοποιεί την προεπιλογή της για αυτόματη εκτέλεση του COMMIT σε όλες τις εντολές της.

```
SET AUTOCOMMIT = 0; (απενεργοποιεί το autocommit mode)  
SET AUTOCOMMIT = 1; (ενεργοποιεί το autocommit mode)
```

Για να ελέγχουμε αν είναι ή όχι ενεργοποιημένο το autocommit mode στην MySQL εκτελούμε την εντολή:

```
SELECT @@autocommit;
```

- *An επιστρέψει 0 τότε το autocommit mode είναι απενεργοποιημένο*
- *An επιστρέψει 1 τότε autocommit mode είναι ενεργοποιημένο*

7.3. Οι Ιδιότητες των συναλλαγών

Σύμφωνα με την θεωρία των συναλλαγών για να διασφαλιστεί ή ακεραιότητα των δεδομένων απαιτείται η βάση δεδομένων να διατηρεί τις παρακάτω 4 ιδιότητες των συναλλαγών. Οι ιδιότητες αυτές αναφέρονται με το ακρωνύμιο ACID (*Atomicity, Consistency, Isolation, Durability*):

Atomicity - Ατομικότητα: Όλες οι λειτουργίες εντός της μονάδας της συναλλαγής θα πρέπει είτε να ολοκληρώνονται με επιτυχία στη βάση δεδομένων είτε αν αποτύχει κάποια λειτουργία από αυτές ολόκληρη η συναλλαγή να ακυρώνεται επανέρχοντας στην κατάσταση πριν την εκτέλεσή της.

Consistency - Συνέπεια: εξασφαλίζεται ότι η ΒΔ βρίσκεται σε σταθερή κατάσταση μετά την ολοκλήρωση κάθε συναλλαγής. Η συναλλαγή πρέπει να μετατρέπει την ΒΔ από μια κατάσταση συνέπειας σε μια άλλη. Η συνέπεια της ΒΔ σχετίζεται με την ορθότητα και ακεραιότητα των δεδομένων της. Στο παράδειγμα που αναφέραμε στην προηγούμενη ενότητα με την μεταφορά χρημάτων από έναν τραπεζικό λογαριασμό σε έναν άλλο τραπεζικό λογαριασμό αφαίρεση των χρημάτων από τον πρώτο λογαριασμό πρέπει να συνοδεύεται με τη πρόσθεση των χρημάτων στον άλλο τραπεζικό λογαριασμό έτσι ώστε το άθροισμα των χρημάτων των δύο τραπεζικών λογαριασμών

πριν και μετά την συναλλαγή να είναι ίδιο. Αυτό ακριβώς σημαίνει συνέπεια στα δεδομένα μιας ΒΔ.

Isolation - Απομόνωση: Κατά την διάρκεια εκτέλεσης μιας συναλλαγής Α δεν επιτρέπονται άλλες συναλλαγές να έχουν πρόσβαση στα δεδομένα που τροποποιούνται στην συναλλαγή Α μέχρι την ολοκλήρωσή της. Κάθε συναλλαγή εκτελείται ανεξάρτητα και απομονωμένα από τις άλλες συναλλαγές.

Durability - Ανθεκτικότητα: Σε κάθε συναλλαγή που εκτελείται με επιτυχία, οι αλλαγές που αυτή έχει πραγματοποιήσει στην ΒΔ παραμένουν μόνιμα ακόμα και μετά από οποιαδήποτε αποτυχία ή βλάβη του συστήματος και δεν μπορούν να ανακληθούν.

7.4. Επίπεδα Απομόνωσης – Isolation Levels

Τα ΣΔΒΔ όταν επεξεργάζονται συναλλαγές συνήθως επιτρέπουν την ταυτόχρονη εκτέλεση πολλαπλών συναλλαγών. Η ταυτόχρονη όμως εκτέλεση των συναλλαγών μπορεί να προκαλέσει προβλήματα συνέπειας στα δεδομένα που αυτές επεξεργάζονται λόγω της ταυτόχρονης πρόσβασης και ενημέρωσης στα ίδια δεδομένα. Για να διασφαλίσουμε την απομόνωση των συναλλαγών και κατ' επέκταση και την συνέπεια των δεδομένων τα ΣΔΒΔ παρέχουν τέσσερα (4) επίπεδα απομόνωσης τα οποία διαφέρουν στον βαθμό ικανοποίησης της απομόνωσης ταυτόχρονων εκτελούμενων συναλλαγών.

Πριν παρουσιάσουμε τα επίπεδα απομόνωσης και για να κατανοήσουμε τις διαφορές μεταξύ αυτών των επιπέδων, θα πρέπει πρώτα να αποκτήσουμε μια γενική εικόνα ορισμένων από τα προβλήματα που μπορεί να προκύψουν σε μια συναλλαγή, ανάλογα με το πόσο απομονωμένη είναι αυτή η συναλλαγή από άλλες συναλλαγές οι οποίες εκτελούνται την ίδια χρονική στιγμή. Τα προβλήματα αυτά ονομάζονται: dirty reads, nonrepeatable reads και phantom reads.

- **Dirty reads:** Τέτοιου είδους προβλήματα μπορεί να συμβούν όταν πολλαπλές συναλλαγές προσπαθούν να έχουν πρόσβαση στα δεδομένα ενός πίνακα την ίδια στιγμή ή κοντά στην ίδια στιγμή. Ένα Dirty Read μπορεί να συμβεί όταν μία συναλλαγή τροποποιεί τα δεδομένα σε έναν πίνακα, μια δεύτερη συναλλαγή διαβάζει τον πίνακα πριν οι εν λόγω τροποποιήσεις αποθηκευτούν μόνιμα (committed) στην ΒΔ και στη συνέχεια η πρώτη συναλλαγή ακυρώσει την τροποποίηση, επιστρέφοντας τη βάση δεδομένων στην αρχική τους τιμή. Η δεύτερη συναλλαγή θα μπορούσε στη συνέχεια να

προσπαθήσει να τροποποιήσει τον πίνακα με βάση την αρχική της ανάγνωση, η οποία δεν είναι πλέον ακριβής (αφού έχουν ακυρωθεί από την πρώτη συναλλαγή).

Συναλλαγή 1	Συναλλαγή 2
SELECT age FROM users WHERE id = 1; <i>(Η age είναι 20)</i>	
UPDATE users SET age = 21 WHERE id = 1; <i>(χωρίς COMMIT)</i>	
	SELECT age FROM users WHERE id = 1; <i>(Η age είναι 21)</i>
ROLLBACK; <i>(Η age γίνεται 20)</i>	
Δεν υπάρχει id=1 και age = 21. Η Συναλλαγή 2 δεν έχει σωστά δεδομένα.	

- **Nonrepeatable reads:** Εκτός από Dirty Reads, πολλαπλές ταυτόχρονες ή σχεδόν ταυτόχρονες συναλλαγές μπορεί να προκαλέσουν το πρόβλημα που λέγεται Nonrepeatable reads. Ένα nonrepeatable read πρόβλημα μπορεί να συμβεί όταν μία συναλλαγή διαβάζει από έναν πίνακα, και στη συνέχεια μια δεύτερη συναλλαγή ενημερώνει τον πίνακα. Εάν η πρώτη συναλλαγή, τότε προσπαθεί να διαβάσει τον πίνακα και πάλι - μετά την ενημέρωση - η συναλλαγή βιώνει ένα nonrepeatable read πρόβλημα. Με άλλα λόγια, τα δεδομένα είναι πλέον διαφορετικά από ό, τι είχε αρχικά διαβάσει η συναλλαγή.

Συναλλαγή 1	Συναλλαγή 2
SELECT age FROM users WHERE id = 1; <i>(Η age είναι 20)</i>	
	UPDATE users SET age = 21 WHERE id = 1; COMMIT;
SELECT age FROM users WHERE id = 1; <i>(Η age είναι 21)</i>	
Η Συναλλαγή 1 πρώτα πήρε ως αποτέλεσμα age=20 και μετά από λίγο πήρε age=21.	

- Phantom reads:** Ένα πρόβλημα Phantom read είναι παρόμοιο με ένα πρόβλημα nonrepeatable read, εκτός από το ότι είναι πιο συγκεκριμένο σε σχέση με τα δεδομένα που ανακτώνται από έναν πίνακα. Η συναλλαγή μας μπορεί να αντιμετωπίσει ένα πρόβλημα Phantom read, όταν η πρώτη συναλλαγή διαβάζει τα δεδομένα που βασίζονται σε μια συγκεκριμένη κατάσταση αναζήτησης, μια δεύτερη συναλλαγή, ενημερώνει τον πίνακα, και στη συνέχεια η πρώτη συναλλαγή διαβάζει από τον πίνακα για μια ακόμη φορά, χρησιμοποιώντας τον ίδιο όρο αναζήτησης. Επειδή ο πίνακας έχει αλλάξει, τα διαφορετικά αποτελέσματα επιστρέφονται στην πρώτη συναλλαγή.

Συναλλαγή 1	Συναλλαγή 2
SELECT * FROM users WHERE age BETWEEN 10 AND 30;	
	INSERT INTO users VALUES (3, 'ΙΩΑΝΝΗΣ', 27); COMMIT;
SELECT * FROM users WHERE age BETWEEN 10 AND 30;	
Η Συναλλαγή 1 πήρε αρχικά διαφορετικά αποτέλεσμα στην αναζήτηση από ότι στην τελευταία ίδια αναζήτηση.	

Τώρα που περιγράφηκαν οι διάφοροι τύποι των προβλημάτων που μπορεί να εμφανιστούν στα δεδομένα όταν οι συναλλαγές εκτελούνται την ίδια στιγμή, θα μελετήσουμε τα είδη των επιπέδων απομόνωσης των συναλλαγών που υπάρχουν έτσι ώστε να είμαστε σε θέση να ρυθμίζουμε σε κάθε συναλλαγή το επίπεδο απομόνωσης της και να εμποδίζονται με τον τρόπο αυτό τυχόν ανωμαλίες στα δεδομένα. Τα επίπεδα απομόνωσης είναι:

Read Uncommitted: Το λιγότερο περιοριστικό των επιπέδων απομόνωσης, το επίπεδο READ UNCOMMITTED επιτρέπει dirty reads, nonrepeatable reads, και phantom reads. Το επίπεδο αυτό θα πρέπει να χρησιμοποιείται μόνο για συναλλαγές που επιστρέφουν γενικές πληροφορίες, όπως στατιστικά στοιχεία τα οποία δεν χρειάζεται να είναι λεπτομερή και ακριβή.

Read Committed: Το επίπεδο αυτό απομόνωσης είναι λίγο περισσότερο περιοριστικό από ότι το Read Uncommitted. Το Read Committed επιτρέπει nonrepeatable reads και phantom reads, αλλά εμποδίζει τα dirty reads. Οι εγγραφές που επιστρέφονται ως αποτελέσματα των αναζητήσεων – ερωτήσεων SELECT που περιλαμβάνονται στις συναλλαγές, είναι μόνο οι εγγραφές των οποίων οι ενημερώσεις έχουν γίνει ήδη COMMIT στην ΒΔ μέχρι την στιγμή που θα ξεκινήσει η εκτέλεση της αναζήτησης – ερώτησης SELECT.

Repeatable Read: Το επίπεδο αυτό απομόνωσης είναι πιο περιοριστικό από τα προηγούμενα δύο επίπεδα. Το Repeatable Reads επιτρέπει phantom reads, αλλά εμποδίζει dirty reads και nonrepeatable reads. Οι εγγραφές που επιστρέφονται ως αποτέλεσματα των αναζητήσεων – ερωτήσεων SELECT που περιλαμβάνονται στις συναλλαγές, είναι μόνο οι εγγραφές των οποίων οι ενημερώσεις έχουν γίνει ήδη COMMIT στην ΒΔ μέχρι την στιγμή που θα ξεκινήσει η εκτέλεση της συναλλαγής και όχι της ερώτησης όπως ορίζεται στο προηγούμενο επίπεδο απομόνωσης. Το Repeatable Read είναι και το επίπεδο απομόνωσης που χρησιμοποιεί και η MySQL ως προεπιλογή (by default) στην InnoDB storage engine.

Serializable: Το πιο περιοριστικό από όλα τα επίπεδα απομόνωσης. Το Serializable εμποδίζει dirty reads, nonrepeatable reads, και phantom reads. Όταν χρησιμοποιείται το επίπεδο απομόνωσης Serializable οι συναλλαγές είναι πλήρως απομονωμένες η μία από την άλλη και το ΣΔΒΔ τις επεξεργάζεται σειριακά. Μια συναλλαγή A σε αυτό το επίπεδο απομόνωσης μπορεί να τροποποιήσει τα δεδομένα μιας εγγραφής μόνο αν προηγούμενες αλλαγές στα δεδομένα αυτά από άλλες συναλλαγές έχουν γίνει COMMIT και έχουν αποθηκευτεί στην ΒΔ μόνιμα αφότου η συναλλαγή A ξεκινήσει.

Ο παρακάτω πίνακας περιγράφει περιληπτικά για κάθε επίπεδο απομόνωσης ποιος τύπος προβλημάτων στα δεδομένα μπορεί να εμφανιστεί και ποιος όχι.

Isolation level	Dirty reads	Nonrepeatable reads	Phantom reads
READ UNCOMMITTED	NAI	NAI	NAI
READ COMMITTED	OXI	NAI	NAI
REPEATABLE READ	OXI	OXI	NAI
SERIALIZABLE	OXI	OXI	OXI

Για να προσδιορίσουμε κάθε φορά το επίπεδο απομόνωσης μιας συναλλαγής θα πρέπει να εξισορροπήσουμε από τη μια την ανάγκη για ακρίβεια των δεδομένων που ανακτώνται και από την άλλη την χρονική απόδοση εκτέλεσης των συναλλαγών. Το πιο περιοριστικό επίπεδο απομόνωσης έχει και την μεγαλύτερη επίδραση στην απόδοση εκτέλεσης των συναλλαγών. Για δεδομένα των οποίων η ακρίβεια είναι πάντα ζωτικής σημασίας, όπως στις χρηματοπιστωτικές συναλλαγές, θα πρέπει να χρησιμοποιούμε το Serializable επίπεδο απομόνωσης.

Η εντολή που καθορίζει το επίπεδο απομόνωσης μιας συναλλαγής είναι:

SET TRANSACTION ISOLATION LEVEL *isolation_level*;

- *isolation_level* μπορεί να είναι **READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE**

Για να δούμε το επίπεδο απομόνωσης που χρησιμοποιείται εκτελούμε την εντολή:

```
SELECT @@global.tx_isolation;
```

7.5. Κλείδωμα πινάκων

Η τελευταίες εκδόσεις της MySQL χρησιμοποιούν την μηχανή αποθήκευσης InnoDB που υποστηρίζει συναλλαγές και κλείδωμα σε επίπεδο εγγραφής. Η εγγραφή δηλαδή που είτε εισάγεται είτε ενημερώνεται είτε διαγράφεται κλειδώνεται και δεν είναι προσβάσιμη μέχρι την εκτέλεση της εντολής COMMIT.

Στις παλαιότερες μηχανές αποθήκευσης που υποστηρίζονταν στις προηγούμενες εκδόσεις της MySQL όπως η MyISAM, δεν υπήρχε η έννοια της συναλλαγής κάθε ενημέρωση σε εγγραφή ενός πίνακα σήμαινε και κλείδωμα ολόκληρου του πίνακα. Ο πίνακας δηλαδή δεν ήταν προσβάσιμος όταν μια εγγραφή του ενημερωνόταν ή εισαγόταν ή διαγραφόταν μέχρι την εκτέλεση του COMMIT.

8. Συνενώσεις

Μελετώντας το παρόν κεφάλαιο οι επιμορφωνόμενοι θα είναι σε θέση να:

- μπορούν να περιγράψουν την έννοια της συνένωσης (join)
- κατανοήσουν και να χρησιμοποιήσουν τους διαφορετικούς τύπους της συνένωσης (join)
- εξοικειωθούν με τη συνένωση (join) στις Update, Delete εντολές

Σε μια κανονικοποιημένη βάση δεδομένων, οι ομάδες των δεδομένων αποθηκεύονται σε επιμέρους πίνακες, και ορίζονται σχέσεις μεταξύ αυτών των πινάκων για τη σύνδεση των σχετικών δεδομένων. Για να επιλέξουμε δεδομένα πρέπει να συσχετίσουμε τους επιμέρους πίνακες βάσει των σχέσεων που έχουν οριστεί μεταξύ τους. Για την συσχέτιση των πινάκων χρησιμοποιούνται τα κοινά τους γνωρίσματα-στήλες. Παρακάτω θα δούμε τρόπους με τους οποίους συνενώνονται οι πίνακες μεταξύ τους.

8.1. Καρτεσιανό γινόμενο πινάκων

Μέχρι τώρα όλες οι εντολές της SQL και κατ' επέκταση οι πράξεις πάνω στα δεδομένα εφαρμόζονταν σε ένα πίνακα. Το Καρτεσιανό γινόμενο συμβολίζεται με το **x** και εφαρμόζεται σε δύο πίνακες. Γράφουμε το Καρτεσιανό γινόμενο δύο πινάκων R1(a, b, c) και R2(d, e, f), R1 x R2. Αποτέλεσμα του Καρτεσιανού γινομένου είναι όλοι οι δυνατοί συνδυασμοί των δύο πινάκων. Κάθε γραμμή του ενός πίνακα συνδυάζεται με όλες τις γραμμές του άλλου. Το Καρτεσιανό γινόμενο συντάσσεται ως:

SELECT * FROM R1, R2; όπου R1 και R2 οι πίνακες μας

Δηλαδή παραθέτουμε μέσα στο FROM τους πίνακες των οποίων θέλουμε να υπολογίσουμε το καρτεσιανό γινόμενο διαχωρισμένους με κόμμα.

Παράδειγμα

Ας θεωρήσουμε το παράδειγμά με τους υπαλλήλους ενός δημόσιου φορέα και τα τμήματα στα οποία έχουν τοποθετηθεί οργανικά. Ο πίνακας EMPLOYEES με τους υπαλλήλους του φορέα είναι (Εικόνα 38 Πίνακας employees (καρτεσιανό γινόμενο)):

	id	firstname	lastname	age	salary	grade	department_id
	1	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2
	2	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	Δ	2
	3	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3
	4	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2000	A	1

Εικόνα 38 Πίνακας employees (καρτεσιανό γινόμενο)

Ο πίνακας DEPARTMENTS με τα τμήματα του οργανισμού του φορέα είναι (Εικόνα 39 Πίνακας DEPARTMENTS (καρτεσιανό γινόμενο)):

	id	department_name
	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ

Εικόνα 39 Πίνακας DEPARTMENTS (καρτεσιανό γινόμενο)

Το καρτεσιανό γινόμενο αυτών των δύο πινάκων είναι ένας συνδυασμένος πίνακας που έχει το σύνολο των γνωρισμάτων-στηλών των δύο πινάκων. Κάθε γραμμή του ενός πίνακα συνδυάζεται με όλες τις γραμμές του άλλου πίνακα. Δηλαδή στην προκειμένη περίπτωση το καρτεσιανό γινόμενο των πινάκων employees και departments παρουσιάζεται στην ακόλουθη εικόνα (Εικόνα 40 Το Καρτεσιανό γινόμενο των πινάκων employees και departments).

	id	firstname	lastname	age	salary	grade	department_id	id	department_name
	1	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
	1	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
	1	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ
	2	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	Δ	2	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
	2	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	Δ	2	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
	2	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	Δ	2	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ
	3	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
	3	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
	3	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ
	4	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2000	A	1	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
	4	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2000	A	1	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
	4	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2000	A	1	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ

Εικόνα 40 Το Καρτεσιανό γινόμενο των πινάκων employees και departments

Το Καρτεσιανό γινόμενο της Εικόνα 40 Το Καρτεσιανό γινόμενο των πινάκων employees και departments το παίρνουμε με την εντολή:

```
SELECT * FROM EMPLOYEES, DEPARTMENTS;
```

Αν οι δύο πίνακες διαθέτουν γνωρίσματα-στήλες με διαφορετικά ονόματα, τότε για να ανακτήσουμε και να χρησιμοποιήσουμε τα γνωρίσματα-στήλες, τα καλούμε με τα ονόματα των γνωρισμάτων αυτών. Δηλαδή στο προηγούμενο παράδειγμα αν θέλαμε να ανακτήσουμε τα γνωρίσματα – στήλες *firstname* (Όνομα), *lastname* (Επώνυμο), *age* (Ηλικία) από τον πίνακα EMPLOYEES και το γνώρισμα-στήλη *department_name* από τον πίνακα DEPARTMENTS θα χρησιμοποιούσαμε τα ονόματα των γνωρισμάτων-στηλών στην εντολή SELECT. Δηλαδή θα γράφαμε:

```
SELECT firstname, lastname, age, department_name  
FROM EMPLOYEES, DEPARTMENTS;
```

Στη περίπτωση όμως που οι πίνακες διαθέτουν γνωρίσματα-στήλες με κοινά ονόματα τότε πρέπει να βρούμε ένα συνδυασμό ονοματολογίας για να ξεχωρίζουμε τα όμοια αυτά γνωρίσματα-στήλες. Το κάθε όνομα του κοινού γνωρίσματος προκύπτει από <το όνομα του πίνακα>. (τελεία)<το όνομα του γνωρίσματος-στήλη>. Στο παράδειγμά μας ο πίνακας EMPLOYEES διαθέτει το γνώρισμα-στήλη *id* που αποτελεί και το πρωτεύον κλειδί του πίνακα και ο πίνακας DEPARTMENTS διαθέτει γνώρισμα-στήλη με το ίδιο όνομα (*id*) που και γι αυτόν αποτελεί το πρωτεύον κλειδί του. Για να ανακτήσουμε το γνώρισμα-στήλη *id* του πίνακα EMPLOYEES γράφουμε *EMPLOYEES.id*. Όμοια για να ανακτήσουμε το γνώρισμα-στήλη του πίνακα DEPARTMENTS γράφουμε *DEPARTMENTS.id*. Η προηγούμενη εντολή SELECT με επιπλέον τα γνωρίσματα-στήλες *id* των δύο πινάκων γίνεται:

```
SELECT EMPLOYEES.id, firstname, lastname, age, DEPARTMENTS.id,  
department_name  
FROM EMPLOYEES, DEPARTMENTS;
```

Το Καρτεσιανό γινόμενο λέγεται αλλιώς και CROSS JOIN.

8.2.Συνένωση (JOIN) πινάκων

Το Καρτεσιανό γινόμενο δύο πινάκων επιστρέφει όλους τους δυνατούς συνδυασμούς των γραμμών των δύο πινάκων. Σε πολλές περιπτώσεις όμως χρειαζόμαστε μόνο ένα υποσύνολο του καρτεσιανού γινομένου, ανάλογα με την τιμή που έχουν τα γνωρίσματα-στήλες των συνδυασμένων πινάκων. Για το σκοπό αυτό έχουμε την πράξη της συνένωσης πινάκων. Η συνένωση συνδυάζει δύο ή περισσότερους πίνακες βασισμένη σε κοινές τιμές των γνωρίσματων-στήλων των πινάκων, προκειμένου να ανακτήσει τα δεδομένα που απαιτούνται.

Οι τύποι των συνενώσεων στην SQL είναι:

- Φυσική Συνένωση - Natural join,
- Εσωτερική Συνένωση - Inner join ή Equijoin,
- Εξωτερική Συνένωση - Outer join (full outer join, left outer join, right outer join),
- Self join,
- Nonequi join.

8.2.1. Φυσική Συνένωση πινάκων (Natural Join)

Η φυσική συνένωση δύο πινάκων συσχετίζει τα γνωρίσματα-στήλες των πινάκων που έχουν το ίδιο όνομα. Επιστρέφει τις γραμμές των δύο πινάκων, οι οποίες έχουν την ίδια τιμή στα κοινά γνωρίσματα-στήλες. Για παράδειγμα έστω ότι έχουμε τον πίνακα T1(a,b,c) και τον πίνακα T2(a, d,e):

T1			T2			T1 NATURAL JOIN T2				
a	b	c	a	d	e	a	b	c	d	e
1	20	10	2	10	30	2	30	20	10	30
2	30	20	3	50	60	3	40	10	50	60
3	40	10	4	80	90					

Το αποτέλεσμα της φυσικής συνένωσης φαίνεται στον τρίτο πίνακα δεξιά. Στο παράδειγμά μας, το μοναδικό κοινό γνώρισμα-στήλη των πινάκων T1 και T2 είναι το a. Στην φυσική συνένωση επιστρέφονται οι γραμμές των δύο πινάκων όπου η τιμή του γνωρίσματος a στον πίνακα T1 ισούται με την τιμή του γνωρίσματος-στήλη a του πίνακα T2 δηλαδή $T1.a=T2.a$.

Για να εφαρμοστεί η φυσική συνένωση σε δύο πίνακες πρέπει να ικανοποιούνται τα εξής:

1. Οι πίνακες να έχουν γνωρίσματα-στήλες με ίδια ονόματα.
2. Τα γνωρίσματα-στήλες με τα ίδια ονόματα θα πρέπει να είναι του ιδίου τύπου δεδομένων. Δηλαδή αν το γνώρισμα-στήλη a του πίνακα T1 είναι αριθμητικού τύπου και το γνώρισμα-στήλη a του T2 είναι τύπου string, τότε δεν μπορεί να εφαρμοστεί η φυσική συνένωση και επιστρέφεται μήνυμα σφάλματος κατά την εκτέλεση της.

Η φυσική συνένωση ή natural join συντάσσεται:

```
SELECT * FROM table1 NATURAL JOIN table2;
```

- *table1, table2: πίνακες*

Αρχικά ορίζουμε στο SELECT τα γνωρίσματα-στήλες των δύο πινάκων που θέλουμε να εμφανιστούν. Αν επιθυμούμε να εμφανίζονται όλα τα γνωρίσματα-στήλες τότε γράφουμε * (όπως φαίνεται στην παραπάνω σύνταξη). Κατόπιν ακολουθεί το FROM όπου ορίζουμε τους πίνακες καθώς και το είδος της συνένωσης μεταξύ τους. Η φυσική συνένωση δηλώνεται ως NATURAL JOIN.

Παράδειγμα

Ας δούμε ένα παράδειγμα με την χρήση της φυσικής συνένωσης. Έστω ότι έχουμε τον πίνακα DEPARTMENTS όπου είναι αποθηκευμένα τα τμήματα του οργανισμού ενός φορέα. Το κάθε τμήμα βρίσκεται σε μια ταχυδρομική διεύθυνση (πίνακας ADDRESS) (**Error! Reference source not found.**).

DEPARTMENTS

	id	department_name	employee_number	directorate_id	address_id
	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	10	1	1
	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	12	2	1
	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ	1	2	2
	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ	0	1	2
	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΩΝ	0	2	2
	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ	5	2	2
	7	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ	NULL	2	2

ADDRESS

	address_id	street	city_name	number
	1	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ	211
	2	ΘΡΑΚΗΣ	ΚΑΛΛΙΘΕΑ	10

Εικόνα 41 Φυσική Συνένωση - Πίνακες DEPARTMENTS, ADDRESS

Στο ερώτημα «Φέρε μου τα τμήματα καθώς και την ταχυδρομική τους διεύθυνση» απαιτείται η χρήση της συνένωσης των δύο πινάκων DEPARTMENTS και ADDRESS. Με μια καλύτερη ματιά στα γνωρίσματα-στήλες των πινάκων παρατηρούμε ότι οι δύο πίνακες διαθέτουν το κοινό γνώρισμα-στήλη address_id όπου αποτελεί και το κλειδί της σύνδεσης των δύο πινάκων. Επομένως η χρήση της φυσικής συνένωσης προκειμένου να ανακτήσουμε τα τμήματα και τις διευθύνσεις τους είναι εφικτή. Η εντολή της φυσικής συνένωσης που πρέπει να εκτελέσουμε είναι:

```
SELECT * FROM DEPARTMENTS NATURAL JOIN ADDRESS;
```

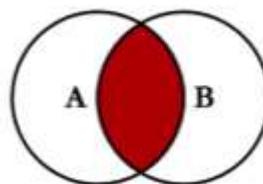
Τα αποτελέσματα της εκτέλεσης της φυσικής συνένωσης είναι:

address_id	id	department_name	employee_number	directorate_id	street	city_name	number
1	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	10	1	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ	211
1	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	12	2	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ	211
2	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ	1	2	ΘΡΑΚΗΣ	ΚΑΛΛΙΒΕΑ	10
2	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ	0	1	ΘΡΑΚΗΣ	ΚΑΛΛΙΒΕΑ	10
2	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΕΩΝ	0	2	ΘΡΑΚΗΣ	ΚΑΛΛΙΒΕΑ	10
2	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ	5	2	ΘΡΑΚΗΣ	ΚΑΛΛΙΒΕΑ	10
2	7	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ	NULL	2	ΘΡΑΚΗΣ	ΚΑΛΛΙΒΕΑ	10
1	8	ΤΜΗΜΑ ΜΙΣΘΩΔΟΣΙΑΣ	NULL	2	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ	211

Εικόνα 42 Το αποτέλεσμα της φυσικής συνένωσης στους πίνακες DEPARTMENTS και ADDRESS

8.2.2. Εσωτερική Συνένωση πινάκων (Inner Join)

Η εσωτερική συνένωση αποτελεί την πιο συνηθισμένη μορφή συνένωσης. Όπως και η φυσική συνένωση δύο πινάκων έτσι και η εσωτερική συνένωση βασίζεται στην ισότητα των γνωρισμάτων-στηλών των δύο πινάκων για τον προσδιορισμό των εγγραφών που θα συμπεριληφθούν στο αποτέλεσμα της συνένωσης. Η διαφορά τους έγκειται στο γεγονός ότι στην εσωτερική συνένωση ορίζουμε κάθε φορά την συνθήκη συνένωσης των πινάκων. Στην **Error! Reference source not found.** φαίνεται σχηματικά η πράξη της συνένωσης δύο πινάκων A και B. Η εσωτερική συνένωση επιστρέφει τις εγγραφές εκείνες από τους δύο πίνακες A και B που έχουν κοινές τιμές στα γνωρίσματα-στήλες όπως ορίζονται στην συνθήκη της συνένωσης.



Εικόνα 43 Εσωτερική Συνένωση - Inner Join

Η εσωτερική συνένωση συντάσσεται:

```
SELECT * FROM table1 [INNER] JOIN table2
ON (table1.column1=table2.column2);
•      table1, table2: πίνακες
•      column1 είναι το γνώρισμα-στήλη του πίνακα table1 και column2 είναι το
```

γνώρισμα του πίνακα *table2* για τα οποία εφαρμόζεται η συνθήκη της συνένωσης

Αρχικά ορίζουμε στο SELECT τα γνωρίσματα-στήλες των δύο πινάκων που θέλουμε να εμφανιστούν. Για να δηλώσουμε κάποιο γνώρισμα-στήλη column1 του πίνακα table1 γράφουμε table1.column1. Αν το όνομα του γνωρίσματος υπάρχει μόνο στον πίνακα table1 τότε μπορούμε να γράψουμε απλά column1. Αν επιθυμούμε να εμφανίζονται όλα τα γνωρίσματα-στήλες τότε μετά το SELECT γράφουμε * (όπως φαίνεται στην παραπάνω σύνταξη). Κατόπιν ακολουθεί το FROM όπου ορίζουμε τους πίνακες καθώς και το είδος της συνένωσης μεταξύ τους. Η εσωτερική συνένωση δηλώνεται ως INNER JOIN είτε απλά JOIN. Έπειτα γράφεται ο τελεστής ON και δηλώνεται η συνθήκη της συνένωσης όπου ουσιαστικά καθορίζει τα γνωρίσματα-στήλες των δύο πινάκων που θα αντιστοιχιστούν.

Η εσωτερική συνένωση μπορεί να γραφεί ισοδύναμα και ως:

SELECT * FROM *table1*, *table2*

WHERE *table1.column1*=*table2.column2*;

- ***table1, table2: πίνακες***
- ***column1 είναι το γνώρισμα-στήλη του πίνακα *table1* και column2 είναι το γνώρισμα του πίνακα *table2* για τα οποία εφαρμόζεται η συνθήκη της συνένωσης***

Στην παραπάνω σύνταξη η συνθήκη της συνένωσης γίνεται συνθήκη στον τελεστή WHERE. Με την παραπάνω σύνταξη καταλαβαίνουμε ότι η συνένωση αποτελεί περικοπή των αποτελεσμάτων του καρτεσιανού γινομένου εφαρμόζοντας την συνθήκη συνένωσης.

Παράδειγμα

Ας θεωρήσουμε το παράδειγμα ενός Δημοσίου Φορέα, όπου οι υπάλληλοί του είναι αποθηκευμένοι στον πίνακα EMPLOYEES και τα τμήματα του Φορέα βρίσκονται στον πίνακα DEPARTMENTS της βάσης δεδομένων μας. Στο ερώτημα «Φέρε μου τους υπαλλήλους του Φορέα που είναι τοποθετημένοι σε τμήματα καθώς και τα τμήματα αυτά» καταλαβαίνουμε ότι θα πρέπει οι πίνακες EMPLOYEES και DEPARTMENTS να συνενωθούν ώστε να λάβουμε τα τμήματα που είναι τοποθετημένοι οι υπάλληλοι.

Στην παρακάτω εικόνα (**Error! Reference source not found.**) φαίνεται ο τρόπος της συνένωσης των πινάκων. Το γνώρισμα-στήλη department_id του πίνακα EMPLOYEES θα αντιστοιχιστεί με το γνώρισμα-στήλη id του πίνακα DEPARTMENTS. Το γνώρισμα-στήλη department_id αποτελεί ξένο κλειδί (foreign key) και αναφέρεται στον πίνακα DEPARTMENTS.

Employees

	id	firstname	lastname	age	salary	grade	department_id
1	1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	38	1200	B	1
2	2	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	40	2000	A	1
3	3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2
4	4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	Δ	2
5	5	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3
6	6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2500	A	1
7	7	ΦΙΛΙΠΠΟΣ	ΝΙΚΟΛΑΟΥ	40	2000	A	NULL
8	8	ΒΑΣΙΛΙΚΗ	ΑΝΔΡΙΩΤΗ	34	1100	Γ	NULL

Departments

	id	department_name	employee_number
▶	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	10
	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	12
	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ	1
	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ	0
	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΩΝ	0
	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ	5
	7	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ	NULL
	8	ΤΜΗΜΑ ΜΙΣΘΩΔΟΣΙΑΣ	NULL

Εικόνα 44 Παράδειγμα εσωτερικής συνένωσης

Η εντολή συνένωσης που θα εκτελεστεί για να ανακτήσουμε τους υπαλλήλους που είναι τοποθετημένοι στα τμήματα του Φορέα είναι:

```
SELECT firstname, lastname, department_name FROM EMPLOYEES
INNER JOIN DEPARTMENTS ON
(EMPLOYEES.department_id=DEPARTMENTS.id);
```

Το αποτέλεσμα της συνένωσης είναι:

	firstname	lastname	department_name
1.	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
2.	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
3.	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
4.	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
5.	ΑΝΝΑ	ΠΑΠΑ	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ
6.	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Εικόνα 45 Το αποτέλεσμα της εσωτερικής συνένωσης

8.2.2.1. Εσωτερική συνένωση με την χρήση του τελεστή USING

Στην περίπτωση που τα αντίστοιχα γνωρίσματα-στήλες των δύο πινάκων στους οποίους θα εφαρμόσουμε την πράξη της συνένωσης έχουν το ίδιο όνομα για παράδειγμα column1 τότε η εσωτερική συνένωση μπορεί να έχει και την ακόλουθη σύνταξη:

```
SELECT * FROM table1
INNER JOIN table2 USING (column1);
```

- *table1, table2: πίνακες*
- *column1 είναι το κοινό γνώρισμα-στήλη των πίνακα table1 και table2 το οποίο χρησιμοποιείται για τη συνένωση των πινάκων*

Στην σύνταξη αυτή χρησιμοποιείται ο τελεστής USING όπου μέσα σε παρενθέσεις δηλώνουμε το κοινό όνομα του γνωρίσματος-στήλης των δύο πινάκων με την χρήση του οποίου θα πραγματοποιηθεί η συνένωση των δύο πινάκων.

Παράδειγμα

Αν θεωρήσουμε τους πίνακες DEPARTMENTS όπου είναι αποθηκευμένα όλα τα τμήματα ενός οργανισμού και ADDRESS όπου βρίσκονται όλες οι ταχυδρομικές διευθύνσεις των τμημάτων. Η ερώτηση «Φέρε με του τα τμήματα καθώς και την ταχυδρομική τους διεύθυνση» θα μπορούσε να απαντηθεί συνενώνοντας τους πίνακες DEPARTMENTS και ADDRESS με την χρήση του κοινού τους γνωρίσματος-στήλης address_id (**Error! Reference source not found.**).

DEPARTMENTS

	id	department_name	employee_number	directorate_id	address_id
	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	10	1	1
	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	12	2	1
	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ	1	2	2
	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ	0	1	2
	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΕΩΝ	0	2	2
	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ	5	2	2
	7	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ	NULL	2	2

ADDRESS

	address_id	street	city_name	number
	1	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ	211
	2	ΘΡΑΚΗΣ	ΚΑΛΛΙΘΕΑ	10

Εικόνα 46 Εσωτερική συνένωση με την χρήση του τελεστή USING

Η πράξη της συνένωσης θα γραφεί ως:

```
SELECT DEPARTMENTS.department_name, DEPARTMENTS.id,
ADDRESS.address_id, ADDRESS.street, ADDRESS.city_name
FROM DEPARTMENTS JOIN ADDRESS USING(address_id);
```

Το αποτέλεσμα φαίνεται στην παρακάτω εικόνα (Error! Reference source not found.).

	department_name	id	address_id	street	city_name
	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙ..	1	1	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ
	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	2	1	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ
	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ	3	2	ΘΡΑΚΗΣ	ΚΑΛΛΙΘΕΑ
	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗ...	4	2	ΘΡΑΚΗΣ	ΚΑΛΛΙΘΕΑ
	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ...	5	2	ΘΡΑΚΗΣ	ΚΑΛΛΙΘΕΑ
	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛ...	6	2	ΘΡΑΚΗΣ	ΚΑΛΛΙΘΕΑ
	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ	7	2	ΘΡΑΚΗΣ	ΚΑΛΛΙΘΕΑ

Εικόνα 47 Το αποτέλεσμα της εσωτερικής συνένωσης με την χρήση του USING

8.2.2.2. Εσωτερική συνένωση και επιπλέον συνθήκη

Στην περίπτωση που επιθυμούμε να εφαρμόσουμε επιπλέον συνθήκη στα αποτελέσματα της εσωτερικής συνένωσης δύο πινάκων χρησιμοποιούμε: είτε τον τελεστή AND και την επιπλέον συνθήκη ακριβώς μετά την συνθήκη της εσωτερικής συνένωσης είτε τον τελεστή WHERE και την επιπλέον συνθήκη ακριβώς μετά την συνθήκη της εσωτερικής συνένωσης. Και οι δύο τρόποι είναι ανάλογοι. Δηλαδή:

```
SELECT * FROM table1 [INNER] JOIN table2
ON (table1.column1=table2.column2)
AND (επιπλέον συνθήκη);
```

Είτε

```
SELECT * FROM table1 [INNER] JOIN table2
ON (table1.column1=table2.column2)
WHERE (επιπλέον συνθήκη);
```

Παράδειγμα

Αν στο προηγούμενο παράδειγμα, της παραγράφου Εσωτερική συνένωση με χρήση του τελεστή Using, μας ζητηθεί να απαντήσουμε στο ερώτημα «Φέρε μου τα τμήματα καθώς και την ακριβή τους ταχυδρομική διεύθυνση που στεγάζονται στον ΤΑΥΡΟ», το μέρος της ερώτησης «που στεγάζονται στον Ταύρο» αποτελεί την επιπλέον συνθήκη.

DEPARTMENTS

	id	department_name	employee_number	directorate_id	address_id
▶	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	10	1	1
	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	12	2	1
	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ	1	2	2
	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ	0	1	2
	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΕΩΝ	0	2	2
	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ	5	2	2
	7	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ	NULL	2	2
	8	ΤΜΗΜΑ ΜΙΣΘΩΔΟΣΙΑΣ	NULL	2	1

ADDRESS

	address_id	street	city_name	number
▶	1	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ	211
	2	ΘΡΑΚΗΣ	ΚΑΛΛΙΒΕΑ	10

Εικόνα 48 Εσωτερική συνένωση και επιπλέον συνθήκη

Η εντολή που θα εκτελέσουμε προκειμένου να ανακτήσουμε τα δεδομένα που επιθυμούμε είναι:

```
SELECT DEPARTMENTS.department_name, DEPARTMENTS.id,  
ADDRESS.address_id, ADDRESS.street, ADDRESS.city_name  
FROM DEPARTMENTS JOIN ADDRESS  
ON (DEPARTMENTS.address_id = ADDRESS.address_id)  
AND ADDRESS.city_name='ΤΑΥΡΟΣ' ;
```

Είτε

```
SELECT DEPARTMENTS.department_name, DEPARTMENTS.id,  
ADDRESS.address_id, ADDRESS.street, ADDRESS.city_name  
FROM DEPARTMENTS JOIN ADDRESS  
ON (DEPARTMENTS.address_id = ADDRESS.address_id)  
WHERE ADDRESS.city_name='ΤΑΥΡΟΣ' ;
```

Το αποτέλεσμα της παραπάνω εντολής είτε με την μορφή του AND είτε με το WHERE είναι:

	department_name	id	address_id	street	city_name
	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	1	1	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ
	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	2	1	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ
	ΤΜΗΜΑ ΜΙΣΘΩΔΟΣΙΑΣ	8	1	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ

Εικόνα 49 Αποτέλεσμα Εσωτερικής συνένωσης με επιπλέον συνθήκη

8.2.3. Εσωτερική συνένωση σε περισσότερους από δύο πίνακες

Μέχρι τώρα αναλύσαμε και παρουσιάσαμε συνενώσεις σε δύο πίνακες. Πολλές φορές όμως για να απαντήσουμε πολύπλοκα ερωτήματα απαιτείται η συνένωση περισσότερων πινάκων. Η συνένωση τριών πινάκων γράφεται γενικά:

```
SELECT * FROM table1  
JOIN table2 ON (table1.column1=table2.column2)
```

```
JOIN table3 ON (table3.column3=table2.column2);
```

Οι εσωτερικές συνενώσεις εκτελούνται από αριστερά προς τα δεξιά. Πρώτα θα εκτελεστεί η εσωτερική συνένωση μεταξύ των πινάκων table1 και table2 και κατόπιν η εσωτερική συνένωση με τον τρίτο πίνακα table3. Η πρώτη συνένωση μπορεί να αναφερθεί σε γνωρίσματα-στήλες των πινάκων table1 και table2 αλλά δεν μπορεί να αναφερθεί σε γνωρίσματα-στήλες του τρίτου πίνακα table3. Η δεύτερη συνένωση μπορεί να αναφερθεί σε γνωρίσματα-στήλες και των τριών πινάκων table1, table2, table3.

Παράδειγμα

Για παράδειγμα για να απαντήσουμε το ερώτημα «Φέρε μου τους υπαλλήλους τα τμήματα στα οποία ανήκουν και τις ταχυδρομικές διευθύνσεις των τμημάτων αυτών.» απαιτείται η εφαρμογή συνένωσεων σε τρεις πίνακες, στον πίνακα EMPLOYEES που περιέχει τους υπαλλήλους μας, στον πίνακα DEPARTMENTS που περιέχει τα τμήματα του Φορέα μας και στον πίνακα ADDRESS που περιλαμβάνει τις ταχυδρομικές διευθύνσεις των τμημάτων του Φορέα (**Error! Reference source not found.**).

Employees

	id	firstname	lastname	age	salary	grade	department_id	manager_id	birthdate
▶	1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	38	1200	B	1	2	1976-02-01
	2	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	40	2000	A	1	HULL	1972-04-03
	3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2	HULL	1968-04-01
	4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	Δ	2	3	1978-08-09
	5	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3	HULL	1978-10-09
	6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2500	A	1	2	1968-11-23
	7	ΦΩΤΙΟΣ	ΝΙΚΟΛΑΟΥ	40	2000	A	HULL	HULL	1974-11-29
	8	ΒΑΣΙΚΗ	ΑΝΔΡΙΩΤΗ	34	1100	Γ	HULL	HULL	1987-11-09

Departments

	id	department_name	employee_number	directorate_id	address_id
▶	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	10	1	1
	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	12	2	1
	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ	1	2	2
	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ	0	1	2
	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΩΝ	0	2	2
	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ	5	2	2
	7	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ	HULL	2	2
	8	ΤΜΗΜΑ ΜΙΣΘΩΔΟΣΙΑΣ	HULL	2	1

Address

	address_id	street	city_name	number
▶	1	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ	211
	2	ΘΡΑΚΗΣ	ΚΑΛΛΙΘΕΑ	10

Εικόνα 50 Παράδειγμα συνένωσης σε περισσότερους από δύο πίνακες

Η εντολή της συνένωσης που πρέπει να εκτελέσουμε είναι:

```

SELECT EMPLOYEES.firstname, EMPLOYEES.lastname,
DEPARTMENTS.department_name,
ADDRESS.street, ADDRESS.city_name
FROM EMPLOYEES JOIN DEPARTMENTS
ON (EMPLOYEES.department_id= DEPARTMENTS. id)
JOIN ADDRESS ON (DEPARTMENTS.address_id= ADDRESS.address_id)

```

Το αποτέλεσμα είναι:

	firstname	lastname	department_name	street	city_name
▶	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ
▶	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ
▶	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ
▶	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ
▶	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	ΠΕΙΡΑΙΩΣ	ΤΑΥΡΟΣ
▶	ΑΝΝΑ	ΠΑΠΑ	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ	ΘΡΑΚΗΣ	ΚΑΛΛΙΘΕΑ

Εικόνα 51 Το αποτέλεσμα συνένωσης σε τρεις πίνακες

8.2.4. Μετονομασία Πινάκων (tables aliases)

Η SQL μας δίνει την δυνατότητα στις ερωτήσεις για ανάκτηση δεδομένων να μετονομάζουμε τους πίνακές μας δίνοντάς τους ένα ψευδώνυμο (table alias). Η μετονομασία του πίνακα είναι προσωρινή μόνο στην σύνταξη και εκτέλεση της εντολής ανάκτησης (SELECT). Τα ψευδώνυμα των πινάκων βοηθούν στο να περιορίζουμε σε μέγεθος τον κώδικα SQL που γράφουμε. Το ψευδώνυμο ενός πίνακα ορίζεται στο FROM της ερώτησής μας. Μετά το FROM γράφουμε το όνομα του πίνακα κατόπιν κενό και μετά ακολουθεί το ψευδώνυμο (FROM <όνομα πίνακα><κενό><ψευδώνυμο>). Εναλλακτικά μετά το FROM γράφουμε το όνομα του πίνακα κατόπιν κενό τον τελεστή as και μετά ακολουθεί το ψευδώνυμο (FROM <όνομα πίνακα><κενό><as><κενό><ψευδώνυμο>). Όταν χρησιμοποιήσουμε ψευδώνυμα-table alias πρέπει να αντικαταστήσουμε σε ολόκληρη τη σύνταξη της ερώτησης το όνομα του πίνακα με το ψευδώνυμό του.

Για παράδειγμα αν θεωρήσουμε την εντολή ερώτησης:

```
SELECT emp.firstname, emp.lastname FROM EMPLOYEES emp
```

```
WHERE emp.id = 10;
```

Είτε

```
SELECT emp.firstname, emp.lastname FROM EMPLOYEES as emp
```

```
WHERE emp.id = 10;
```

Ο πίνακας EMPLOYEES μετονομάστηκε σε emp. Η μετονομασία ορίζεται στον τελεστή FROM. Κατόπιν όταν θέλουμε να αναφερθούμε στον πίνακα χρησιμοποιούμε το ψευδώνυμο του emp. Για παράδειγμα στην συνθήκη του WHERE όταν θέλουμε να

αναφερθούμε στην στήλη id του πίνακα γράφουμε emp.id. Επίσης για να ανακτήσουμε τα γνωρίσματα-στήλες firstname, lastname του πίνακα EMPLOYEES χρησιμοποιούμε το ψευδώνυμο του πίνακα και γράφουμε emp.firstname, emp.lastname.

Στην παρακάτω εντολή ανάκτησης δεδομένων ο πίνακας EMPLOYEES με μετονομάστηκε σε **a** και ο πίνακας DEPARTMENTS σε **d**.

```
SELECT a.firstname, a.lastname, d.department_name  
FROM EMPLOYEES a, DEPARTMENTS d  
WHERE a.department_id=d.id
```

8.2.5. Μετονομασία γνωρισμάτων-στηλών (columns aliases)

Όμοια με την μετονομασία πινάκων, η SQL μας δίνει την δυνατότητα στις ερωτήσεις για ανάκτηση δεδομένων να μετονομάζουμε τα γνωρίσματα-στήλες που ανακτούμε δίνοντάς τους ένα ψευδώνυμο (column alias). Η μετονομασία των γνωρισμάτων-στηλών είναι προσωρινή μόνο στην σύνταξη και εκτέλεση της εντολής. Τα ψευδώνυμα των γνωρισμάτων-στηλών εμφανίζονται ως τίτλοι στο αποτέλεσμα των εντολών ανάκτησης δεδομένων. Συνήθως η μετονομασία γνωρισμάτων εφαρμόζεται σε γνωρίσματα-στήλες που προκύπτουν από υπολογισμούς σε γνωρίσματα-στήλες των πινάκων. Το ψευδώνυμο ενός γνωρίσματος στήλης ορίζεται στον τελεστή SELECT της ερώτησής μας. Μετά το SELECT γράφουμε το όνομα του γνωρίσματος-στήλης ή το υπολογιζόμενο γνώρισμα από τα γνωρίσματα-στήλες των πινάκων, κατόπιν κενό και μετά ακολουθεί το ψευδώνυμο ή ο τελεστής **as** και μετά το ψευδώνυμο (SELECT<κενό><όνομα στήλης><κενό><as><κενό><ψευδώνυμο> ή SELECT<κενό><όνομα στήλης><κενό><ψευδώνυμο> (με απουσία δηλαδή του τελεστή as)). Για παράδειγμα στην παρακάτω ερώτηση ανάκτησης δεδομένων το γνώρισμα a.firstname του πίνακα μετονομάστηκε σε fname, το a.lastname μετονομάστηκε σε lname και το d.department_name σε depname.

```
SELECT a.firstname fname, a.lastname lname, d.department_name depname  
FROM EMPLOYEES a, DEPARTMENTS d
```

WHERE a.department_id=d.id

Στο επόμενο παράδειγμα βλέπουμε την μετονομασία γνωρίσματος-στήλης που προκύπτει από υπολογισμούς σε γνωρίσματα-στήλες του πίνακα EMPLOYEES. Συγκεκριμένα ορίζουμε το γνώρισμα-στήλη annual_salary που προκύπτει από το γνώρισμα-στήλη salary του πίνακα EMPLOYEES πολλαπλασιασμένο με 12 προκειμένου να υπολογίσουμε τον ετήσιο μισθό κάθε υπαλλήλου.

```
SELECT lastname lname, 12*salary as annual_salary  
FROM EMPLOYEES
```

Το αποτέλεσμα του ερωτήματος φαίνεται στην **Error! Reference source not found.**

	lname	annual_salary
.	ΜΠΕΚΙΑΡΗ	14400
.	ΤΑΣΟΠΟΥΛΟΣ	24000
.	ΠΑΠΑΓΕΩΡΓΙΟΥ	36000
.	ΚΑΡΑΝΙΚΟΛΑΣ	12000
.	ΠΑΠΑ	42000
.	ΠΑΠΑΔΟΠΟΥΛΟΣ	24000

Εικόνα 52 Μετονομασία γνωρισμάτων-στηλών πίνακα

8.2.6. Συνένωση (Self Join)

Πολλές φορές εμφανίζεται απαραίτητη η συνένωση ενός πίνακα με τον εαυτό του. Συνήθως η ανάγκη εμφανίζεται σε πίνακες που έχουν κάποιο γνώρισμα-στήλη που αναφέρεται στο πρωτεύον κλειδί του ίδιου του πίνακα. Πρόκειται δηλαδή για ξένο κλειδί (Foreign key) του πίνακα που αναφέρεται στο πρωτεύον κλειδί του. Η συνένωση αυτή ονομάζεται self-join. Η συνένωση ενός πίνακα με τον εαυτό του σημαίνει ότι κάθε εγγραφή του πίνακα συνενώνεται με τον εαυτό της και με κάθε άλλη εγγραφή του πίνακα. Στο self join θεωρούμε ότι έχουμε δύο αντίγραφα του ιδίου πίνακα τα οποία τα χειριζόμαστε ως δύο ξεχωριστούς πίνακες. Στην πραγματικότητα δεν δημιουργούνται αντίγραφα του πίνακα, η εντολή SQL εκτελείται σαν να είχαμε δύο αντίγραφα. Εφαρμόζουμε την συνένωση στα δύο αυτά ξεχωριστά αντίγραφα. Η self join συνένωση συντάσσεται ως εξής:

```
SELECT * FROM TABLE1 A, TABLE1 B  
WHERE A.column1=B.column2  
Eίτε  
SELECT * FROM TABLE1 A  
JOIN TABLE1 B ON (A.column1=B.column2)
```

Το πρώτο «αντίγραφο» του πίνακα μετονομάζεται σε Α και το δεύτερο «αντίγραφο» του πίνακα σε Β. Οι δύο μορφές συνένωσης που φαίνονται παραπάνω είναι αντίστοιχες. Στην πρώτη μορφή η συνθήκη της συνένωσης εμφανίζεται στον τελεστή WHERE ενώ στην δεύτερη μορφή η συνθήκη της συνένωσης εμφανίζεται στον τελεστή ON. Οι δύο αυτές μορφές είναι ισοδύναμες.

Παράδειγμα

Ας χρησιμοποιήσουμε τον πίνακα EMPLOYEES που περιλαμβάνει όλους τους υπαλλήλους ενός Δημοσίου Φορέα. Ο πίνακας EMPLOYEES έχει το γνώρισμα-στήλη manager_id όπου σε αυτό αποθηκεύουμε για κάθε υπάλληλο τον προϊστάμενό του. Ο προϊστάμενος είναι και αυτός υπάλληλος του Δημόσιου Φορέα, και υπάρχει εγγραφή στον πίνακα EMPLOYEES που περιέχει τα δικά του στοιχεία. Ουσιαστικά το γνώρισμα-στήλη manager_id αποτελεί ξένο κλειδί (foreign key) του πίνακα EMPLOYEES που αναφέρεται στον ίδιο τον πίνακα EMPLOYEES. Το γνώρισμα-στήλη manager_id αναφέρεται στο γνώρισμα-στήλη id που αποτελεί το πρωτεύον κλειδί του πίνακα EMPLOYEES.

	employees
id	INT(11)
firstname	VARCHAR(200)
lastname	VARCHAR(200)
age	INT(11)
salary	DOUBLE
grade	VARCHAR(45)
department_id	INT(11)
manager_id	INT(11)
birthdate	DATE
Indexes	

Εικόνα 53 Σχήμα Πίνακα

Στο ερώτημα «Ποιοι υπάλληλοι του Φορέα είναι managers» μπορούμε να απαντήσουμε κάνοντας χρήση της συνένωσης self join αφού το στοιχείο που μας δείχνει τους managers του Φορέα είναι το γνώρισμα-στήλη manager_id που αναφέρεται στον ίδιο τον πίνακα EMPLOYEES. Ας θεωρήσουμε δύο αντίγραφα του πίνακα EMPLOYEES τον πίνακα **e** και τον πίνακα **m** (Εικόνα 54 Self-join δύο αντίγραφα του πίνακα EMPLOYEES). Οι managers (κόκκινο περίβλημα στον πίνακα **m**) είναι οι υπάλληλοι με id 2 και 3. Εξετάζοντας τον πίνακα **e** οι υπάλληλοι με id 2 και 3 είναι ο ΝΙΚΟΣ ΚΟΥΡΤΗΣ και ο ΓΕΩΡΓΙΟΣ ΠΑΠΑΓΕΩΡΓΙΟΥ.

Employees e

id	firstname	lastname	age	salary	grade	department_id	manager_id	birthdate
1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	38	1200	B	1	2	1976-02-01
2	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	40	2000	A	1	NULL	1972-04-03
3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2	NULL	1968-04-01
4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	Δ	2	3	1978-08-09
5	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3	NULL	1978-10-09
6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2500	A	1	2	1968-11-23
7	ΦΙΛΙΠΠΟΣ	ΝΙΚΟΛΑΟΥ	40	2000	A	NULL	NULL	1974-11-29
8	ΒΑΣΙΚΗ	ΑΝΔΡΙΩΤΗ	34	1100	Γ	NULL	NULL	1987-11-09

Employees m

id	firstname	lastname	age	salary	grade	department_id	manager_id	birthdate
1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	38	1200	B	1	2	1976-02-01
2	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	40	2000	A	1	NULL	1972-04-03
3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2	NULL	1968-04-01
4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	Δ	2	3	1978-08-09
5	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3	NULL	1978-10-09
6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2500	A	1	2	1968-11-23
7	ΦΙΛΙΠΠΟΣ	ΝΙΚΟΛΑΟΥ	40	2000	A	NULL	NULL	1974-11-29
8	ΒΑΣΙΚΗ	ΑΝΔΡΙΩΤΗ	34	1100	Γ	NULL	NULL	1987-11-09

Εικόνα 54 Self-join δύο αντίγραφα του πίνακα EMPLOYEES

Αν εφαρμόσουμε την συνένωση self join στον EMPLOYEES, θα ανακτήσουμε τους managers του Φορέα. Η εντολή ανάκτησης γράφεται:

```
SELECT e.firstname, e.lastname
FROM EMPLOYEES e, EMPLOYEES m
WHERE e.id=m.manager_id;

Eίτε

SELECT e.firstname, e.lastname
FROM EMPLOYEES e JOIN EMPLOYEES m
ON (e.id=m.manager_id);
```

Το αποτέλεσμα της εντολής φαίνεται στην **Error! Reference source not found.** είναι:

firstname	lastname
ΝΙΚΟΣ	ΚΟΥΡΤΗΣ
ΝΙΚΟΣ	ΚΟΥΡΤΗΣ
ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ

Εικόνα 55 Αποτέλεσμα self-join συνένωσης στον πίνακα EMPLOYEES

Η εγγραφή ΝΙΚΟΣ ΚΟΥΡΤΗΣ εμφανίζεται δύο φορές αφού δύο είναι οι υπάλληλοι που τον έχουν προϊστάμενο. Αν θέλαμε να πάρουμε μοναδική φορά τον καθένα θα έπρεπε να χρησιμοποιήσουμε τον τελεστή DISTINCT .

8.2.7. Nonequi Join Συνένωση

Η συνένωση nonequijoin διαφέρει από τους υπόλοιπους τύπους συνενώσεων. Στην συνθήκη συνένωσης της nonequijoin χρησιμοποιείται οποιοσδήποτε άλλος τελεστής σύγκρισης πλην του τελεστή της isότητας. Μπορούν να χρησιμοποιηθούν οι τελεστές !=, <, >, <=, >=, BETWEEN κα. Η συνένωση nonequijoin συντάσσεται:

```
SELECT * FROM TABLE1, TABLE2  
WHERE TABLE1.column1 OPERATOR TABLE2.columns2;  
Eίτε  
SELECT * FROM TABLE1 JOIN TABLE2  
ON (TABLE1.column1 OPERATOR TABLE2.columns2);  
• OPERATOR είναι οποιοσδήποτε τελεστής σύγκρισης πλην της isότητας
```

8.2.8. Εξωτερική Συνένωση (Outer Join)

Η εσωτερική συνένωση ακόμα και η φυσική συνένωση επιστρέφουν όλες τις εγγραφές των πινάκων για τις οποίες υπάρχει αντιστοίχιση στις τιμές των γνωρισμάτων-στηλών που περιλαμβάνονται στην συνθήκη της συνένωσης. Οποιαδήποτε εγγραφή στους δύο πίνακες που περιλαμβάνει τιμές στα γνωρίσματα-στήλες της συνθήκη της συνένωσης που δεν ταυτίζονται δεν περιλαμβάνεται στα αποτελέσματα της συνένωσης. Πολλές φορές όμως χρειάζεται να ανακτήσουμε και τις εγγραφές εκείνες των δύο πινάκων που δεν ταυτίζονται. Για το σκοπό αυτό χρησιμοποιούμε την Εξωτερική συνένωση ή αλλιώς OUTER JOIN. Η εξωτερική συνένωση επιστρέφει όλες τις εγγραφές του αριστερού ή/και δεξιού πίνακα χωρίς κατ' ανάγκη να ικανοποιείται η συνθήκη της συνένωσης.

Η εξωτερική συνένωση διακρίνεται σε:

- Αριστερή εξωτερική συνένωση ή LEFT OUTER JOIN ή LEFT JOIN
- Δεξιά εξωτερική συνένωση ή RIGHT OUTER JOIN ή RIGHT JOIN
- και σε πλήρης εξωτερική συνένωση FULL OUTER JOIN ή OUTER JOIN

Για να κατανοήσουμε καλύτερα την έννοια της εξωτερικής συνένωσης ας δούμε το παράδειγμα που παρουσιάζεται στην παρακάτω εικόνα (**Error! Reference source not found.**).

EMPLOYEES

	id	firstname	lastname	age	salary	grade	department_id
	1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	38	1200	B	1
	2	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	40	2000	A	1
	3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2
	4	ΒΑΣΙΛΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	A	2
	5	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3
	6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2500	A	1
	7	ΦΩΤΙΟΣ	ΝΙΚΟΛΑΟΥ	40	2000	A	HULL
	8	ΒΑΣΙΛΗ	ΑΝΔΡΙΩΤΗ	34	1100	Γ	HULL

Employees που δεν είναι
Τοποθετημένοι σε Τμήμα

DEPARTMENTS

id	department_name
1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ
4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ
5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΕΩΝ
6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ
7	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ
8	ΤΜΗΜΑ ΜΙΣΘΟΔΟΣΙΑΣ

Εικόνα 56 Εξωτερική συνένωση ή OUTER JOIN

Ο πίνακας EMPLOYEES περιλαμβάνει όλους τους υπαλλήλους ενός Φορέα είτε αυτοί είναι τοποθετημένοι σε Τμήμα είτε όχι όπως τους υπαλλήλους με το μπλε περίγραμμα στην Εικόνα 56 Εξωτερική συνένωση ή OUTER JOIN. Ο Πίνακας DEPARTMENTS περιέχει όλα τα τμήματα του Φορέα ακόμα και αυτά τα οποία δεν είναι στελεχωμένα (πράσινο περίγραμμα στην Εικόνα 56 Εξωτερική συνένωση ή OUTER JOIN). Η εσωτερική συνένωση επιστρέφει τις εγγραφές του EMPLOYEES για τις οποίες υπάρχει αντιστοίχιση με τις εγγραφές του DEPARTMENTS. Δηλαδή όλους τους υπαλλήλους πλην αυτούς που δεν είναι τοποθετημένοι σε τμήματα και όλα τα τμήματα πλην αυτά που δεν είναι στελεχωμένα. Η εσωτερική συνένωση δεν αρκεί για να απαντήσουμε στις ερωτήσεις:

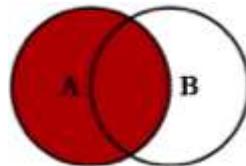
- «Ποιοι είναι όλοι οι υπάλληλοι του Φορέα και ποια τα τμήματα στα οποία είναι τοποθετημένοι»
- ή στην ερώτηση «Ποιά είναι όλα τα τμήματα του Φορέα και ποιοι υπάλληλοι υπηρετούν σε κάθε τμήμα»

- ή ακόμα και στην ερώτηση «Ποια είναι όλα τα τμήματα του Φορέα και ποιοι υπάλληλοι υπηρετούν σε κάθε τμήμα αλλά και ποιοι υπάλληλοι δεν είναι τοποθετημένοι σε τμήματα»

Τις ερωτήσεις αυτές καλείται να τις απαντήσει η πράξη της εξωτερικής συνένωσης.

8.2.9. Αριστερή εξωτερική συνένωση (LEFT OUTER JOIN)

Η αριστερή εξωτερική συνένωση ή αλλιώς left outer join ή left join μεταξύ δύο πινάκων A και B (**Error! Reference source not found.**) επιστρέφει όλες τις εγγραφές του πίνακα A που είτε υπάρχει ταύτιση των τιμών των κοινών γνωρισμάτων (που ορίζονται στην συνθήκη της συνένωσης) με τον πίνακα B είτε όχι. Στο αποτέλεσμα της αριστερής εξωτερικής συνένωσης εμφανίζονται όλες οι εγγραφές του A αλλά μόνο οι εγγραφές του B στις οποίες οι τιμές των γνωρισμάτων των δύο πινάκων ταυτίζονται. Οι εγγραφές του A που δεν έχουν ταυτιστεί με εγγραφές του B, οι εγγραφές δηλαδή του A που έχουν κενές (NULL) τιμές στα γνωρίσματα-στήλες του B περιέχονται στο αποτέλεσμα της αριστερής εξωτερικής συνένωσης.



Εικόνα 57 Αριστερή εξωτερική συνένωση (Left Outer Join)

Η αριστερή εξωτερική συνένωση συντάσσεται ως εξής:

```
SELECT * FROM table1
```

```
LEFT [OUTER] JOIN table2 ON (table1.column1=table2.column2);
```

- όπου table1 και table2 οι πίνακες στους οποίους εφαρμόζεται η συνένωση
- table1.column1 και table2.column2 τα γνωρίσματα-στήλες των δύο πινάκων όπου οι τιμές τωνς ταυτίζονται

Ο τελεστής OUTER μπορεί να παραληφθεί από την σύνταξη της συνένωσης.

Παράδειγμα

Για παράδειγμα η ερώτηση «Ποιοι είναι όλοι οι υπάλληλοι του Φορέα και ποια τα τμήματα στα οποία είναι τοποθετημένοι.» για να απαντηθεί χρησιμοποιούμε την αριστερή εξωτερική συνένωση μεταξύ των πινάκων EMPLOYEES (αριστερός πίνακας αφού η ερώτηση ζητά όλους τους υπαλλήλους) και DEPARTMENTS (δεξιός πίνακας).

EMPLOYEES

	id	firstname	lastname	age	salary	grade	department_id
	1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	38	1200	B	1
	2	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	40	2000	A	1
	3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2
	4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	Δ	2
	5	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3
	6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2500	A	1
	7	ΦΙΛΙΠΠΟΣ	ΝΙΚΟΛΑΟΥ	40	2000	A	NULL
	8	ΒΑΣΙΛΗ	ΑΝΔΡΙΩΤΗ	34	1100	Γ	NULL

Εμπολεμένες που δεν είναι
Τοποθετημένοι σε Τμήμα

DEPARTMENTS

	id	department_name
	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ
	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ
	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΕΩΝ
	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΟΥ
	7	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ
	8	ΤΜΗΜΑ ΜΙΣΘΟΔΟΣΙΑΣ

Εικόνα 58 Παράδειγμα Αριστερής εξωτερικής συνένωσης

Η πράξη της συνένωσης που θα απαντήσει στην ερώτηση είναι:

```
SELECT * FROM EMPLOYEES
LEFT [OUTER] JOIN DEPARTMENTS ON
(EMPLOYEES.department_id=DEPARTMENTS.id);
```

Το αποτέλεσμα της συνένωσης είναι:

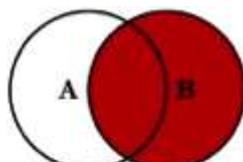
	id	firstname	lastname	id	department_name
1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	
2	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	
3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	
4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	
5	ΑΝΝΑ	ΠΑΠΑ	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ	
6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	
7	ΦΙΛΙΠΠΟΣ	ΝΙΚΟΛΑΟΥ	NULL	NULL	
8	ΒΑΣΙΛΙΚΗ	ΑΝΔΡΙΩΤΗ	NULL	NULL	

Εικόνα 59 Αποτέλεσμα παραδείγματος αριστερής εξωτερικής συνένωσης

Στο αποτέλεσμα (Εικόνα 59 Αποτέλεσμα παραδείγματος αριστερής εξωτερικής συνένωσης) της αριστερής εξωτερικής συνένωσης βλέπουμε όλες τις εγγραφές του πίνακα EMPLOYEES και εκείνες μόνο τις εγγραφές του DEPARTMENTS για τις οποίες υπάρχει ταύτιση τιμών μεταξύ των κοινών γνωρισμάτων-στηλών. Στο αποτέλεσμα της συνένωσης υπάρχουν και οι εγγραφές του EMPLOYEES που έχουν NULL τιμές στα γνωρίσματα-στήλες του DEPARTMENTS.

8.2.10. Δεξιά εξωτερική συνένωση (RIGHT OUTER JOIN)

Η δεξιά εξωτερική συνένωση ή αλλιώς right outer join ή right join μεταξύ δύο πινάκων A και B (Εικόνα 60 Δεξιά εξωτερική συνένωση (Right Outer Join)) επιστρέφει όλες τις εγγραφές του πίνακα B που είτε υπάρχει ταύτιση των τιμών των κοινών γνωρισμάτων (που ορίζονται στην συνθήκη της συνένωσης) με τον πίνακα A είτε όχι. Στο αποτέλεσμα της δεξιάς εξωτερικής συνένωσης εμφανίζονται όλες οι εγγραφές του B αλλά μόνο οι εγγραφές του A στις οποίες οι τιμές των γνωρισμάτων των δυο πινάκων ταυτίζονται. Η δεξιά εξωτερική συνένωση φαίνεται στην παρακάτω εικόνα (**Error! Reference source not found.**)



Εικόνα 60 Δεξιά εξωτερική συνένωση (Right Outer Join)

Η δεξιά εξωτερική συνένωση συντάσσεται ως εξής:

SELECT * FROM *table1*

RIGHT [OUTER] JOIN *table2* ON (*table1.column1*=*table2.column2*);

- *όπου *table1* και *table2* οι πίνακες στον οποίον εφαρμόζεται η συνένωση*
- **table1.column1* και *table2.column2* τα γνωρίσματα-στήλες των δύο πινάκων όπου οι τιμές των ταυτίζονται*

Ο τελεστής OUTER μπορεί να παραληφθεί από την σύνταξη της συνένωσης.

Παράδειγμα

Για παράδειγμα στην ερώτηση «Ποιά είναι όλα τα τμήματα του Φορέα και ποιοι υπάλληλοι υπηρετούν σε κάθε τμήμα» χρησιμοποιούμε την δεξιά εξωτερική συνένωση μεταξύ των πινάκων EMPLOYEES (αριστερός πίνακας) και DEPARTMENTS (δεξιός πίνακας αφού η ερώτηση ζητά όλα τα τμήματα).

EMPLOYEES

	id	firstname	lastname	age	salary	grade	department_id
	1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	38	1200	B	1
	2	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	40	2000	A	1
	3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2
	4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	Δ	2
	5	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3
	6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2500	A	1
	7	ΦΙΛΙΠΠΟΣ	ΝΙΚΟΛΑΟΥ	40	2000	A	NULL
	8	ΒΑΣΙΛΙΚΗ	ΑΝΔΡΙΩΤΗ	34	1100	Γ	NULL

DEPARTMENTS

	id	department_name
	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ
	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ
	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΕΩΝ
	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΛΟΥ
	7	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ
	8	ΤΜΗΜΑ ΜΙΣΘΟΔΟΣΙΑΣ

Τμήματα που δεν
είναι στελεχωμένα

Εικόνα 61 Παράδειγμα δεξιάς εξωτερικής συνένωσης

Η πράξη της συνένωσης που θα απαντήσει στην ερώτηση είναι:

```
SELECT * FROM EMPLOYEES
RIGHT [OUTER] JOIN DEPARTMENTS ON
(EMPLOYEES.department_id=DEPARTMENTS.id);
```

Το αποτέλεσμα της συνένωσης είναι:

	id	firstname	lastname	id	department_name
	1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
	2	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
	6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
	3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
	4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
	5	ΑΝΝΑ	ΠΑΠΑ	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ
	NULL	NULL	NULL	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ
	NULL	NULL	NULL	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΕΩΝ
	NULL	NULL	NULL	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΟΥ
	NULL	NULL	NULL	7	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ
	NULL	NULL	NULL	8	ΤΜΗΜΑ ΜΙΣΘΩΔΟΣΙΑΣ

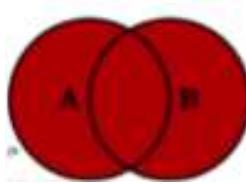
Εικόνα 62 Αποτέλεσμα παραδείγματος δεξιάς εξωτερικής συνένωσης

Στο αποτέλεσμα (Εικόνα 62 Αποτέλεσμα παραδείγματος δεξιάς εξωτερικής συνένωσης) της δεξιάς εξωτερικής συνένωσης βλέπουμε όλες τις εγγραφές του πίνακα DEPARTMENTS και εκείνες μόνο τις εγγραφές του EMPLOYEES για τις οποίες υπάρχει ταύτιση τιμών μεταξύ των κοινών γνωρισμάτων-στηλών. Οι υπόλοιπες εγγραφές του DEPARTMENTS έχουν NULL τιμές στα γνωρίσματα-στήλες του EMPLOYEES.

8.2.11. Πλήρης εξωτερική συνένωση (FULL OUTER JOIN)

Η πλήρης εξωτερική συνένωση ή αλλιώς full outer join μεταξύ δύο πινάκων A και B (Εικόνα 63 Πλήρη εξωτερική συνένωση (Full Outer Join)) επιστρέφει όλες τις εγγραφές του πίνακα A και όλες τις εγγραφές του πίνακα B είτε υπάρχει ταύτιση των τιμών των κοινών γνωρισμάτων (που ορίζονται στην συνθήκη της συνένωσης) τους είτε όχι. Στο αποτέλεσμα της πλήρους εξωτερικής συνένωσης οι εγγραφές του A που δεν έχουν ταυτιστεί με τις εγγραφές του πίνακα B έχουν κενές (NULL) τιμές στα γνωρίσματα-στήλες του B. Το ίδιο συμβαίνει αντίστοιχα και στις εγγραφές του B που δεν έχουν

ταυτιστεί με εγγραφές του Α. Η πλήρης εξωτερική συνένωση φαίνεται στην παρακάτω εικόνα (Εικόνα 63 Πλήρη εξωτερική συνένωση (Full Outer Join)).



Εικόνα 63 Πλήρη εξωτερική συνένωση (Full Outer Join)

Η πλήρης εξωτερική συνένωση δεν υποστηρίζεται ακόμα στην MySQL. Μπορεί όμως να προσημειωθεί με:

```
SELECT * FROM table1
LEFT JOIN table2 ON table1.column1 = table2.column2
UNION
SELECT * FROM table1
RIGHT JOIN table2 ON table1.column1 = table2.column2
•      ópon table1 και table2 oi pínakes stonç opoiouç eφaprodzetai η συνένωση
•      table1.column1 και table2.column2ta γνωρísmata-stíloues twn dñou pínakow
ópon oi tímēs tonç tautízontai
```

Ο τελεστής UNION είναι ο τελεστής της ένωσης δύο συνόλων, της αριστερής εξωτερικής συνένωσης και της δεξιάς εξωτερικής συνένωσης. Σε επόμενο κεφάλαιο θα παρουσιάσουμε την πράξη της ένωσης δύο πινάκων.

Παράδειγμα

Για παράδειγμα στην ερώτηση «Ποια είναι όλα τα τμήματα του Φορέα και ποιοι υπάλληλοι υπηρετούν σε κάθε τμήμα αλλά και ποιοι υπάλληλοι δεν είναι τοποθετημένοι σε τμήματα» χρησιμοποιούμε την πλήρη εξωτερική συνένωση μεταξύ των πινάκων EMPLOYEES (αριστερός πίνακας - η ερώτηση ζητά όλουç τους υπαλλήλους ακόμα και εκείνους που δεν είναι τοποθετημένοι σε τμήμα του Φορέα) και DEPARTMENTS (δεξιός πίνακας-η ερώτηση ζητά όλα τα τμήματα).

EMPLOYEES

	id	firstname	lastname	age	salary	grade	department_id
1	1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	38	1200	B	1
2	2	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	40	2000	A	1
3	3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2
4	4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	Δ	2
5	5	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3
6	6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2500	A	1
7	7	ΦΙΛΙΠΠΟΣ	ΝΙΚΟΛΑΟΥ	40	2000	A	NULL
8	8	ΒΑΣΙΛΙΚΗ	ΑΝΔΡΙΩΤΗ	34	1100	Γ	NULL

Employees που δεν είναι
Τοποθετημένοι σε Τμήμα

DEPARTMENTS

	id	department_name
1	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
2	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
3	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ
4	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ
5	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΕΩΝ
6	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛΟΥ
7	7	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ
8	8	ΤΜΗΜΑ ΜΙΣΘΩΔΟΣΙΑΣ

Εικόνα 64 Παράδειγμα πλήρης εξωτερικής συνένωσης

Η πράξη της συνένωσης που θα απαντήσει στην ερώτηση είναι:

```

SELECT * FROM EMPLOYEES
LEFT [OUTER] JOIN DEPARTMENTS ON
(EMPLOYEES.department_id=DEPARTMENTS.id)
UNION
SELECT * FROM EMPLOYEES
RIGHT [OUTER] JOIN DEPARTMENTS ON
(EMPLOYEES.department_id=DEPARTMENTS.id)
    
```

Το αποτέλεσμα της συνένωσης είναι:

	id	firstname	lastname	id	department_name
	1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
	2	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
	3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
	4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ
	5	ΑΝΝΑ	ΠΑΠΑ	3	ΤΜΗΜΑ ΛΟΓΙΣΤΗΡΙΟΥ
	6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
	7	ΦΙΛΙΠΠΟΣ	ΝΙΚΟΛΑΟΥ	NULL	NULL
	8	ΒΑΣΙΛΙΚΗ	ΑΝΔΡΙΩΤΗ	NULL	NULL
	NULL	NULL	NULL	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ Κ...
	NULL	NULL	NULL	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧ...
	NULL	NULL	NULL	6	ΤΜΗΜΑ ΠΡΩΤΟΚΟΛ...
	NULL	NULL	NULL	7	ΤΜΗΜΑ ΠΡΟΜΗΘΕΙΩΝ
	NULL	NULL	NULL	8	ΤΜΗΜΑ ΜΙΣΘΩΔΟΣΙΑΣ

Εικόνα 65 Αποτέλεσμα παραδείγματος πλήρους εξωτερικής συνένωσης

Στο αποτέλεσμα (Εικόνα 65 Αποτέλεσμα παραδείγματος πλήρους εξωτερικής συνένωσης) της πλήρους εξωτερικής συνένωσης βλέπουμε όλες τις εγγραφές του πίνακα DEPARTMENTS και όλες τις εγγραφές του πίνακα EMPLOYEES. Οι εγγραφές οι οποίες δεν έχουν ταυτιστεί ως προς τα κοινά τους γνωρίσματα-στήλες του DEPARTMENTS και του EMPLOYEES έχουν NULL τιμές στα γνωρίσματα-στήλες των δύο πινάκων.

9. Συναρτήσεις

Μελετώντας το παρόν κεφάλαιο οι επιμορφωνόμενοι θα είναι σε θέση να εξοικειωθούν με τις sql συναρτήσεις:

- Συναθροιστικές συναρτήσεις (Aggregate functions),
- Συναρτήσεις String (String functions),
- Αριθμητικές Συναρτήσεις (Numeric functions),
- Συναρτήσεις Ημερομηνίας και ώρας (Date and Time functions)
- να κατανοήσουν την εντολή ομαδοποίησης group by και την εντολή Having

Στο κεφάλαιο αυτό θα αναφερθούμε στις συναρτήσεις που υποστηρίζονται από το standard της SQL. Θα παρουσιάσουμε τις συναρτήσεις που χρησιμοποιούνται στην MySQL και θα δούμε την σύνταξη τους. Οι συναρτήσεις είναι εντολές που ποικίλουν είτε παίρνουν ως παραμέτρους γνωρίσματα-στήλες του πίνακα είτε δεν έχουν παραμέτρους. Οι συναρτήσεις επιστρέφουν ως αποτέλεσμα τιμές που είτε προκύπτουν από επεξεργασία των τιμών των γνωρισμάτων-στηλών του πίνακα είτε επιστρέφουν κάποια συγκεκριμένη τιμή του πίνακα.

9.1. Συναθροιστικές Συναρτήσεις (Aggregate functions)

Οι Συναθροιστικές συναρτήσεις είναι συναρτήσεις που παίρνουν μια συλλογή (ένα σύνολο ή πολλαπλά σύνολα) από τιμές ως είσοδο-παράμετρο και επιστρέφουν μια μόνο τιμή. Η SQL παρέχει πέντε (5) συναθροιστικές συναρτήσεις:

- **AVG:** Μέσος Όρος. Η συνάρτηση αυτή επιστρέφει το μέσο όρο ενός συνόλου αριθμητικών δεδομένων. Αν επιθυμούμε να υπολογίσουμε τον μέσο όρο μόνο των διακριτών τιμών του συνόλου τότε μπορεί να χρησιμοποιηθεί ο τελεστής DISTINCT δηλαδή AVG(DISTINCT expr), όπου expr αντιστοιχεί στο σύνολο τιμών. Το expr μπορεί να είναι ένα γνώρισμα-στήλη του πίνακα που επιθυμούμε να υπολογίσουμε την μέση τιμή των τιμών του ή μπορεί να είναι μια έκφραση της οποίας επιθυμούμε να υπολογίσουμε την μέση τιμή.
- **SUM:** Άθροισμα. Η συνάρτηση αυτή επιστρέφει το άθροισμα ενός συνόλου μόνο αριθμητικών δεδομένων. Αν επιθυμούμε να υπολογίσουμε το άθροισμα μόνο των διακριτών τιμών του συνόλου τότε μπορεί να χρησιμοποιηθεί ο τελεστής DISTINCT

δηλαδή $\text{SUM}(\text{DISTINCT expr})$, όπου expr αντιστοιχεί στο σύνολο τιμών (όμοια με την AVG).

- **MAX:** Μέγιστη τιμή. Η συνάρτηση αυτή επιστρέφει την μέγιστη τιμή ενός συνόλου αριθμητικών και μη δεδομένων.
- **MIN:** Ελάχιστη τιμή. Η συνάρτηση αυτή επιστρέφει την ελάχιστη τιμή ενός συνόλου αριθμητικών και μη δεδομένων.
- **COUNT:** Καταμέτρηση. Η συνάρτηση αυτή επιστρέφει το πλήθος των εγγραφών του πίνακα και συντάσσεται $\text{COUNT}(*)$ ή το πλήθος των τιμών ενός γνωρίσματος-στήλης που η τιμή του δεν είναι `NULL` και συντάσσεται $\text{COUNT}(\text{column})$. Στην περίπτωση που επιθυμούμε να μετρήσουμε μόνο τις μοναδικές τιμές ενός γνωρίσματος-στήλης του πίνακα μπορούμε να χρησιμοποιήσουμε τον τελεστή DISTINCT και η συνάρτηση να γραφεί $\text{COUNT}(\text{DISTINCT column})$.

Παράδειγμα

Φανταστείτε το ερώτημα «Βρείτε τον μέγιστο μισθό των υπαλλήλων του Φορέα». Ο μέγιστος μισθός των υπαλλήλων του Φορέα είναι 3500 ευρώ (Εικόνα 66 Παράδειγμα Συναθροιστικής συνάρτησης MAX - Μέγιστος Μισθός του Φορέα)

Employees

	id	firstname	lastname	age	salary
1	MARIA	ΜΠΕΚΙΑΡΗ	38	1200	
2	NIKOS	ΤΑΣΟΠΟΥΛΟΣ	40	2000	
3	GEORGIOS	ΠΑΠΑΓΕΩΡΠΟΥ	45	3000	
4	VASILIOS	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	
5	ANNA	ΠΑΠΑ	35	3500	
6	MANOULIS	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2000	
7	PHILIPPOS	ΝΙΚΟΛΑΟΥ	25	1000	

Ο μεγαλύτερος μισθός της Εταιρείας
Max(salary) = 3500

max(salary)
3500

Εικόνα 66 Παράδειγμα Συναθροιστικής συνάρτησης MAX - Μέγιστος Μισθός του Φορέα

Για να απαντήσουμε στο παραπάνω ερώτημα πρέπει να χρησιμοποιήσουμε την συναθροιστική συνάρτηση MAX και να εκτελέσουμε την εντολή:

SELECT $\text{MAX}(\text{salary})$ FROM *EMPLOYEES*;

Αν θέλουμε να περιορίσουμε το σύνολο τιμών μας σε συγκεκριμένες εγγραφές του πίνακα μας και όχι σε όλο τον πίνακα όπως η προηγούμενη εντολή χρησιμοποιούμε τον τελεστή WHERE .

Παράδειγμα

Για παράδειγμα αν μας ζητήσουν να απαντήσουμε στο ερώτημα «Βρείτε το μέγιστο, το ελάχιστο, το μέσο μισθό και το άθροισμα των μηνιαίων μισθών του ΤΜΗΜΑΤΟΣ ΠΛΗΡΟΦΟΡΙΚΗΣ» όπου το ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ έχει department_id=1. Θα πρέπει να περιορίσουμε το σύνολο τιμών μόνο στους υπαλλήλους του τμήματος Πληροφορικής. Η εντολή ανάκτησης δεδομένων θα είναι:

```
SELECT MAX(salary), MIN(salary), AVG(salary), SUM(salary)
FROM employees
WHERE department_id=1;
```

Το αποτέλεσμα της εντολής θα είναι:

	MAX(salary)	MIN(salary)	AVG(salary)	SUM(salary)
►	2000	1000	1550	6200

Εικόνα 67 Αποτέλεσμα συναθροιστικών συναρτήσεων με χρήση του τελεστή WHERE

Παράδειγμα

Φανταστείτε ότι μας ζητούν να απαντήσουμε στο ερώτημα «Πόσοι είναι οι υπάλληλοι του ΤΜΗΜΑΤΟΣ ΠΛΗΡΟΦΟΡΙΚΗΣ» όπου το ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ έχει department_id=1. Θα πρέπει να περιορίσουμε το σύνολο τιμών μόνο στους υπαλλήλους του τμήματος Πληροφορικής. Η εντολή ανάκτησης δεδομένων θα είναι:

```
SELECT COUNT(*)
FROM employees
WHERE department_id=1;
```

Το αποτέλεσμα της εντολής θα είναι:

	count(*)
►	3

Εικόνα 68 Αποτέλεσμα COUNT, επιστρέφει τον αριθμό των υπαλλήλων του Τμήματος Πληροφορικής

Υπάρχουν περιπτώσεις όπου πρέπει να απαλείψουμε τα διπλότυπα πριν εφαρμόσουμε την συναθροιστική συνάρτηση. Για να απαλείψουμε τα διπλότυπα χρησιμοποιούμε τον τελεστή DISTINCT.

Παράδειγμα

Για παράδειγμα στο ερώτημα «Να βρεθεί το πλήθος των διαφορετικών τιμών μισθών στη βάση δεδομένων». Σε αυτή την περίπτωση η εντολή ανάκτησης είναι:

```
SELECT COUNT(DISTINCT salary)  
FROM EMPLOYEES;
```

9.1.1.1. Ομαδοποίηση Δεδομένων τελεστής group by

Υπάρχουν περιπτώσεις που θα θέλαμε να εφαρμόσουμε τις Συναθροιστικές συναρτήσεις όχι μόνο σε ένα σύνολο εγγραφών αλλά και σε ομάδες από σύνολα εγγραφών. Οι ομάδες από εγγραφές καθορίζονται από τον τελεστή group by. Τα γνωρίσματα-στήλες που δίνονται στο group by χρησιμοποιούνται για να σχηματίσουν τις ομάδες. Οι εγγραφές με την ίδια τιμή γνωρισμάτων-στηλών του group by τοποθετούνται στη ίδια ομάδα.

Παράδειγμα

Για παράδειγμα αν ζητηθεί να απαντήσουμε στο ερώτημα «Βρείτε το μέσο μισθό υπαλλήλων του Φορέα ανά τμήμα», εκτελούμε την εντολή:

```
SELECT department_id, AVG(salary)  
FROM EMPLOYEES  
GROUP BY department_id;
```

Χωρίζουμε τον πίνακα με τους υπαλλήλους του Φορέα σε ομάδες βάση του τμήματος στο οποίο είναι τοποθετημένοι. Κατόπιν υπολογίζουμε τον μέσο όρο μισθών των υπαλλήλων ανά τμήμα.

	id	firstname	lastname	age	salary	grade	department_id
	1	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	38	1200	A	1
	2	ΝΙΚΟΣ	ΤΑΣΟΠΟΥΛΟΣ	40	2000	A	1
	6	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	45	2000	A	1
	3	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	45	3000	A	2
	4	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	35	1000	B	2
	5	ΑΝΝΑ	ΠΑΠΑ	35	3500	B	3

Εικόνα 69 Ο πίνακας EMPLOYEES χωρίστηκε σε τρεις ομάδες βάσει του τμήματος που έχουν τοποθετηθεί οι υπάλληλοι

Παράδειγμα

Φανταστείτε την ερώτηση «Για κάθε τμήμα να ανακτηθεί ο κωδικός τμήματος, το όνομα του τμήματος καθώς και το πλήθος των υπαλλήλων που είναι τοποθετημένοι σε αυτό». Για να την απαντήσουμε γράφουμε:

```
SELECT department_id, department_name, COUNT(*)
FROM EMPLOYEES, DEPARTMENTS
WHERE EMPLOYEES.department_id=DEPARTMENTS.id
GROUP BY department_id, department_name;
```

Στην παραπάνω εντολή βλέπουμε πως μπορούμε να χρησιμοποιήσουμε μια συνθήκη συνένωσης σε συνδυασμό με την GROUP BY. Στην περίπτωση αυτή η ομαδοποίηση και οι συναρτήσεις εφαρμόζονται μετά την συνένωση των δύο σχέσεων.

Μερικές φορές θέλουμε να ανακτήσουμε τις τιμές αυτών των συναρτήσεων μόνο για ομάδες που ικανοποιούν ορισμένες συνθήκες. Για παράδειγμα, υποθέστε ότι θέλουμε να τροποποιήσουμε την προηγούμενη ερώτηση έτσι ώστε στο αποτέλεσμα να εμφανίζονται μόνο τα τμήματα με περισσότερους από δύο υπαλλήλους. Για το σκοπό αυτό η SQL παρέχει τον τελεστή HAVING που μπορεί να εμφανιστεί μαζί με τον τελεστή GROUP BY. Ο τελεστής HAVING παρέχει μια συνθήκη πάνω σε κάθε ομάδα εγγραφών, έτσι ώστε μόνο οι ομάδες εγγραφών που ικανοποιούν την συνθήκη να εμφανίζονται στο αποτέλεσμα της ερώτησης. Η σύνταξη των εντολών ανάκτησης δεδομένων είναι:

```
SELECT column, group_function  
FROM table  
[WHERE condition]  
[GROUP BY group_by_expression]  
[HAVING group_condition]  
[ORDER BY column];
```

Παράδειγμα

Στην ερώτηση «Για κάθε τμήμα στο οποίο είναι τοποθετημένοι πάνω από 2 υπάλληλοι να ανακτηθεί ο κωδικός τμήματος, το όνομα του τμήματος καθώς και το πλήθος των υπαλλήλων που είναι τοποθετημένοι σε αυτό». Για να την απαντήσουμε γράφουμε:

```
SELECT department_id, department_name, COUNT(*)  
FROM EMPLOYEES, DEPARTMENTS  
WHERE EMPLOYEES.department_id=DEPARTMENTS.id  
GROUP BY department_id, department_name  
HAVING COUNT(*)>2;
```

Σημειώστε ότι ενώ οι συνθήκες στο WHERE περιορίζουν τις εγγραφές στις οποίες εφαρμόζονται οι συναρτήσεις, ο τελεστής HAVING περιορίζει ολόκληρες ομάδες.

Παράδειγμα

Φανταστείτε την ερώτηση «Φέρε μου τα τμήματα (id τμημάτων) και το μέσο μισθό υπαλλήλων τους που έχουν μέσο όρο μισθό υπαλλήλων μεγαλύτερο ή ίσο με 2000 Ευρώ». Η εντολή ανάκτησης γράφεται:

```
SELECT department_id, AVG(salary)  
FROM EMPLOYEES  
GROUP BY department_id  
HAVING AVG(salary) >= 2000;
```

Στην ερώτηση αυτή περιορίζουμε τις ομάδες των τμημάτων σε αυτές με μέσο μισθό

υπαλλήλων μεγαλύτερο ή ίσο του 2000 Ευρώ.

Παράδειγμα

Στην ερώτηση «Για κάθε τμήμα να ανακτηθεί ο κωδικός τμήματος, το όνομα του τμήματος καθώς και το πλήθος των υπαλλήλων που είναι τοποθετημένοι σε αυτό ηλικίας πάνω από 40». Για να την απαντήσουμε γράφουμε:

```
SELECT department_id, department_name, COUNT(*)
FROM EMPLOYEES, DEPARTMENTS
WHERE EMPLOYEES.department_id=DEPARTMENTS.id and
EMPLOYEES.age>40
GROUP BY department_id, department_name;
```

Στην ερώτηση αυτή περιορίζουμε τις εγγραφές κάθε ομάδας σε αυτές που ικανοποιούν την συνθήκη του WHERE, δηλαδή τους υπαλλήλους ηλικίας μεγαλύτερης του 40.

Πρέπει να είμαστε ιδιαίτερα προσεκτικοί όταν εφαρμόζονται δύο διαφορετικές συνθήκες μία στον τελεστή WHERE της SELECT και μία στον τελεστή HAVING.

Παράδειγμα

Για παράδειγμα στην ερώτηση «Για κάθε τμήμα με περισσότερους από πέντε (5) υπαλλήλους να ανακτηθεί ο κωδικός τμήματος, το όνομα του τμήματος και το πλήθος των υπαλλήλων με μισθό μεγαλύτερο 1000 Ευρώ». Υποθέστε ότι απαντάμε λανθασμένα όπως:

```
SELECT department_id, department_name, COUNT(*)
FROM EMPLOYEES, DEPARTMENTS
WHERE EMPLOYEES.department_id=DEPARTMENTS.id and
EMPLOYEES.salary>1000;
GROUP BY department_id, department_name
HAVING COUNT(*) > 5;
```

Η εντολή αυτή ανάκτησης είναι λανθασμένη αφού θα επιλέξει μόνο τα τμήματα με

περισσότερους από πέντε υπαλλήλους που όλοι κερδίζουν πάνω από 1000 Ευρώ. Όταν στην εντολή υπάρχει WHERE αυτή εκτελείται πρώτα, για να επιλεγούν συγκεκριμένες εγγραφές και κατόπιν εκτελείται η HAVING για να επιλεγούν συγκεκριμένες ομάδες εγγραφών. Οι εγγραφές έχουν ήδη περιοριστεί στους υπαλλήλους με μισθό μεγαλύτερο από 1000 Ευρώ πριν εφαρμοστεί η HAVING. Για να γραφεί σωστά η παραπάνω εντολή ανάκτησης θα χρησιμοποιήσουμε τα υποερωτήματα-subqueries τα οποία θα περιγράψουμε σε επόμενο κεφάλαιο.

Στην εντολή αυτή πρώτα θα εκτελεστεί το υποερώτημα, πρώτα δηλαδή θα επιλεχθούν τα τμήματα με περισσότερους από πέντε υπαλλήλους και μετά θα εφαρμοστεί η συνθήκη ότι ο μισθός πρέπει να είναι μεγαλύτερος του 1000 Ευρώ.

```
SELECT department_id, department_name, COUNT(*)
FROM EMPLOYEES, DEPARTMENTS
WHERE EMPLOYEES.department_id=DEPARTMENTS.id
and EMPLOYEES.salary>1000
and DEPARTMENTS.id IN ( SELECT department_id
                           FROM EMPLOYEES
                           GROUP BY department_id
                           HAVING COUNT(*)>5)
GROUP BY department_id, department_name;
```

9.1.2. Συναρτήσεις Αλφαριθμητικών (String functions)

Η SQL και κατ επέκταση η MySQL περιλαμβάνουν ένα σύνολο συναρτήσεων για την διαχείριση των αλφαριθμητικών.

- **CONCAT:** Η συνάρτηση αυτή συνενώνει δύο ή περισσότερα αλφαριθμητικά (strings) σε ένα. Η σύνταξή της είναι η ακόλουθη:

CONCAT (str1, str2 ...[, strN])

Για παράδειγμα:

SELECT CONCAT ('H', ' ', 'My','SQ', 'L') ;

Συνενώνει πέντε (5) αλφαριθμητικά σε ένα. Θα επιστρέψει το αλφαριθμητικό:

'H MySQL'

Παρατηρήστε ότι το δεύτερο αλφαριθμητικό είναι το κενό. Επίσης αν κάποιο από τα

str1, str2, str3, ... είναι NULL τότε η CONCAT επιστρέφει NULL. Για παράδειγμα:

SELECT CONCAT ('My',NULL, 'L');

Θα επιστρέψει NULL.

- **CONCAT_WS:** Η συνάρτηση αυτή όπως και η CONCAT συνενώνει δύο ή περισσότερα αλφαριθμητικά (strings) σε ένα. Η διαφορά της έγκειται στο ότι μας επιτρέπει να ορίσουμε ένα διαχωριστή ως πρώτο όρισμα της συνάρτησης που θα εισάγεται αυτόματα ανάμεσα στα αλφαριθμητικά που θα συνενώσει. Εάν ένα από τα αλφαριθμητικά είναι NULL, ο διαχωριστής δεν χρησιμοποιείται. Η σύνταξή της είναι:

CONCAT_WS (separator, str1, str2 ... [, strN])

Για παράδειγμα η:

```
SELECT      CONCAT_WS('      ',      'H','MySQL','παρέχει','ένα','σύνολο',
'συναρτήσεων');
```

Θα επιστρέψει: '**H MySQL παρέχει ένα σύνολο συναρτήσεων**'

Στο παράδειγμα αυτό ο διαχωριστής είναι ο χαρακτήρας κενό και ορίζεται ως πρώτο όρισμα της συνάρτησης CONCAT_WS. Κατόπιν όλα τα υπόλοιπα αλφαριθμητικά συνενώνονται εισάγοντας μεταξύ τους τον διαχωριστή δηλαδή το κενό.

- **LOWER:** Η MySQL περιλαμβάνει την συνάρτηση LOWER που μας επιτρέπει να αλλάξουμε τα αλφαριθμητικά μας σε πεζά. Η σύνταξή της είναι:

LOWER (str)

Για παράδειγμα, η ακόλουθη δήλωση SELECT χρησιμοποιεί τη συνάρτηση LOWER () για να αφαιρέσουμε τα κεφαλαία γράμματα:

SELECT LOWER ('H MySQL')

Θα επιστρέψει: '**η mysql**'

- **UPPER:** Όμοια με την LOWER η MySQL περιλαμβάνει την συνάρτηση UPPER που μας επιτρέπει να αλλάξουμε τα αλφαριθμητικά μας σε κεφαλαία. Η σύνταξή της είναι:

UPPER (str)

Για παράδειγμα, η ακόλουθη δήλωση SELECT χρησιμοποιεί τη συνάρτηση UPPER () για να μετατρέψουμε όλο το αλφαριθμητικό σε κεφαλαία γράμματα:

SELECT UPPER ('η MySQL')

Θα επιστρέψει: '**H MYSQL**'

- **REPLACE:** Η συνάρτηση αυτή αντικαθιστά το δεύτερο όρισμα με την τιμή του τρίτου ορίσματος στο αλφαριθμητικό του πρώτου ορίσματος. Η σύνταξή της είναι:

REPLACE (str, from_str, to_str)

Για παράδειγμα:

SELECT REPLACE('www.mysql.com', 'w', 'Ww');

Επιστρέφει : 'WwWwWw.mysql.com'

Αντικαθιστά το w με το Ww στο αλφαριθμητικό του πρώτου ορίσματος.

- **REVERSE:** Η συνάρτηση αυτή επιστρέφει τους χαρακτήρες του αλφαριθμητικού που βρίσκεται στο όρισμα με αντίστροφη σειρά. Η σύνταξή της είναι:

REVERSE(str)

Για παράδειγμα:

SELECT REVERSE ('MySQL');

Επιστρέφει : 'LQSyM'

- **REPEAT:** Η συνάρτηση αυτή επαναλαμβάνει το πρώτο όρισμα που είναι ένα αλφαριθμητικό τόσες φορές όσες αναφέρονται στο δεύτερο όρισμα. Η σύνταξή της είναι:

REPEAT(str, count)

Αν το δεύτερο όρισμα δηλαδή στην παραπάνω σύνταξη το count είναι μικρότερο του 1, η REPEAT επιστρέφει το κενό. Ενώ αν το αλφαριθμητικό str ή το count είναι NULL τότε επιστρέφεται NULL. Για παραδειγμα:

REPEAT('MySQL', 3)

Θα επιστρέψει: MySQLMySQLMySQL

- **LENGTH:** Η συνάρτηση αυτή χρησιμοποιείται για την εύρεση του μήκους σε bytes του αλφαριθμητικού που περιλαμβάνεται στο όρισμα. Η σύνταξή της είναι:

LENGTH(str)

Για παράδειγμα:

SELECT LENGTH('MySQL')

Θα επιστρέψει: 5

- **CHAR_LENGTH ή CHARACTER_LENGTH:** Η συνάρτηση αυτή επιστρέφει τον αριθμό των χαρακτήρων του αλφαριθμητικού που περιλαμβάνεται στο όρισμα. Η σύνταξή της είναι:

CHAR_LENGTH(str)

Για παράδειγμα:

SELECT CHAR_LENGTH('MySQL')

Θα επιστρέψει: 5

Αυτό σημαίνει ότι για ένα αλφαριθμητικό που περιέχει πέντε χαρακτήρες των 2-byte, η συνάρτηση LENGTH() επιστρέφει 10, ενώ CHAR_LENGTH () επιστρέφει 5.

- **SUBSTRING:** Επιστρέφει ένα κομμάτι του αλφαριθμητικού που είναι στο όρισμα ανάλογα με την σύνταξη της συνάρτησης που χρησιμοποιείται.

SUBSTRING(str, pos): Επιστρέφει το κομμάτι του αλφαριθμητικού str από την θέση pos και μετά.

SUBSTRING(str FROM pos): Επιστρέφει το κομμάτι του αλφαριθμητικού του str από την θέση pos και μετά.

SUBSTRING(str, pos, len): Επιστρέφει το κομμάτι του αλφαριθμητικού του str από την θέση pos και μετά len χαρακτήρες.

SUBSTRING(str FROM pos FOR len): Επιστρέφει το κομμάτι του αλφαριθμητικού του str από την θέση pos και μετά len χαρακτήρες.

Στην περίπτωση που το pos είναι αρνητικός αριθμός ξεκινάμε την αρίθμηση από το τέλος του str για pos θέσεις.

Για παράδειγμα:

SELECT SUBSTRING('Quadratically',5);

Επιστρέφει: **ratically**

SELECT SUBSTRING('foobarbar' FROM 4);

Επιστρέφει: **barbar**

SELECT SUBSTRING('Quadratically',5,6);

Επιστρέφει: **ratica**

SELECT SUBSTRING('Sakila', -3);

Επιστρέφει: **ila**

SELECT SUBSTRING('Sakila', -5, 3);

Επιστρέφει: **aki**

SELECT SUBSTRING('Sakila' FROM -4 FOR 2);

Επιστρέφει: **ki**

- **LOCATE:** Η συνάρτηση αυτή επιστρέφει τη θέση της πρώτης εμφάνισης του πρώτου ορίσματος μέσα στο αλφαριθμητικό του δεύτερου ορίσματος. Η σύνταξή της είναι :

LOCATE(substr,str)

LOCATE(substr,str,pos)

Επιστρέφει τη θέση του κομματιού *substr* που εμφανίζεται μέσα στο αλφαριθμητικό *str*. Η δεύτερη σύνταξη επιστρέφει τη θέση της πρώτης εμφάνισης του κομματιού *substr* μέσα στο *str* ξεκινώντας την αναζήτηση από τη θέση *pos*. Αν το *substr* δεν υπάρχει στο *str*, το αποτέλεσμα είναι 0.

- **INSTR:** Επιστρέφει τη θέση στην οποία το δεύτερο όρισμα εμφανίζεται μέσα στο πρώτο όρισμα. Η σύνταξή της είναι:

INSTR(str,substr)

Η συνάρτηση αυτή θα επιστρέψει την θέση του *str* όπου εμφανίζεται το *substr* για πρώτη φορά.

- **TRIM:** Η συνάρτηση αυτή χρησιμοποιείται για να διαγράφει τα προθέματα ή επιθέματα ενός αλφαριθμητικού *str* ανάλογα με την σύνταξή της. Η συνάρτηση συντάσσεται:

TRIM ([{BOTH | LEADING | TRAILING} [remstr] FROM] str)

ή

TRIM([remstr FROM] str)

Επιστρέφει το αλφαριθμητικό *str* χωρίς τα κομμάτια *remstr* που υπάρχουν σαν προθέματα ή επιθέματα. Το *LEADING* διαγράφει τα κομμάτια του *remstr* που υπάρχουν ως προθέματα στο *str*. Το *TRAILING* διαγράφει τα κομμάτια του *remstr* που υπάρχουν ως επιθέματα από το *str*. Το *BOTH* διαγράφει τα κομμάτια *remstr* που υπάρχουν ως προθέματα ή επιθέματα στο *str*. Αν δεν οριστεί κάποια παράμετρος από τα BOTH, LEADING, ή TRAILING θεωρείται ότι ισχύει η BOTH. Το *remstr* είναι προαιρετικό και, αν δεν οριστεί, αφαιρούνται χαρακτήρες κενά που βρίσκονται ως προθέματα ή επιθέματα στο *str*. Για παράδειγμα:

SELECT TRIM(' bar ');

Επιστρέφει: 'bar'

SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx');

Επιστρέφει: 'barxxx'

SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx');

Επιστρέφει: 'bar'

SELECT TRIM(TRAILING 'xyz' FROM 'barxxz');

Επιστρέφει: 'barx'

- **RTRIM:** Επιστρέφει το αλφαριθμητικό που δηλώνεται ως όρισμα αφαιρώντας τους κενούς χαρακτήρες που πιθανόν να υπάρχουν ως επίθεμα στο αλφαριθμητικό *str*. Η σύνταξή της συνάρτησης είναι:

RTRIM(str)

Για παράδειγμα:

SELECT RTRIM('barbar ');

Επιστρέφει: 'barbar'

- **LTRIM:** Επιστρέφει το αλφαριθμητικό που δηλώνεται ως όρισμα αφαιρώντας τους κενούς χαρακτήρες που πιθανόν να υπάρχουν ως πρόθεμα στο αλφαριθμητικό *str*. Η σύνταξή της συνάρτησης είναι:

LTRIM(str)

Για παράδειγμα:

SELECT LTRIM(' barbar');

Επιστρέφει: 'barbar'

9.1.3. Αριθμητικές συναρτήσεις (Mathematical functions)

Η MySQL παρέχει ένα σύνολο μαθηματικών συναρτήσεων που χρησιμοποιούνται για τη διαχείριση αριθμητικών δεδομένων. Οι πιο συχνά χρησιμοποιούμενες συναρτήσεις είναι:

- **ABS:** Επιστρέφει την απόλυτη τιμή του ορίσματος. Η σύνταξή της συνάρτησης είναι:

ABS(X)

Για παράδειγμα:

SELECT ABS(2);

Επιστρέφει: 2

SELECT ABS(-32);

Επιστρέφει: 32

- **CEIL** ή **CEILING:** Επιστρέφει τον κοντινότερο ακέραιο αριθμό που είναι μεγαλύτερος από την τιμή του ορίσματος. Η σύνταξή της συνάρτησης είναι:

CEIL(X) ή CEILING(X)

Για παράδειγμα:

SELECT CEILING(1.23);

Επιστρέφει: **2**

SELECT CEILING(-1.23);

Επιστρέφει: **-1**

- **FLOOR:** Επιστρέφει τον κοντινότερο ακέραιο αριθμό που είναι μικρότερος από την τιμή. Η σύνταξή της συνάρτησης είναι:

FLOOR (X)

Για παράδειγμα:

SELECT FLOOR(1.23);

Επιστρέφει: **1**

SELECT FLOOR(-1.23);

Επιστρέφει: **-2**

- **ROUND:** Στρογγυλοποιεί το πρώτο όρισμα σε τόσα δεκαδικά ψηφία όσο το δεύτερο όρισμα. Αν δεν ορίζεται το δεύτερο όρισμα τότε ισοδυναμεί με μηδέν (0) δεκαδικά ψηφία. Η σύνταξή της συνάρτησης είναι:

ROUND (X, D) όπου X ο αριθμός που στρογγυλοποιείται και D ο αριθμός των δεκαδικών ψηφίων

ή

ROUND (X) , όπου X ο αριθμός που στρογγυλοποιείται

Για παράδειγμα:

SELECT ROUND(-1.23);

Επιστρέφει: **-1**

SELECT ROUND(-1.58);

Επιστρέφει: **-2**

SELECT ROUND(1.58);

Επιστρέφει: **2**

SELECT ROUND(1.298, 1);

Επιστρέφει: **1.3**

SELECT ROUND(1.298, 0);

Επιστρέφει: **1**

- **MOD:** Επιστρέφει το υπόλοιπο της διαίρεσης του πρώτου ορίσματος / του δεύτερου ορίσματος. Η σύνταξή της συνάρτησης είναι:

MOD (N, M) , το υπόλοιπο της διαίρεσης N/M

Για παράδειγμα:

SELECT MOD(34.5, 3);

Επιστρέφει: **1,5**

- **SQRT:** Επιστρέφει την τετραγωνική ρίζα του ορίσματος. Η σύνταξη της συνάρτησης είναι:

SQRT (X)

Για παράδειγμα:

SELECT SQRT (4);

Επιστρέφει: **2**

- **POW:** Επιστρέφει την τιμή του πρώτου ορίσματος υψωμένη στην τιμή του δεύτερου ορίσματος. Η σύνταξη της συνάρτησης είναι:

POW (X, Y), επιστρέφει X^Y

Για παράδειγμα:

SELECT POW (4, 2);

Επιστρέφει: **16**

- **TRUNCATE:** Επιστρέφει το πρώτο όρισμα με τόσα δεκαδικά ψηφία όσο το δεύτερο όρισμα. Αν το δεύτερο όρισμα είναι 0 επιστρέφει το πρώτο όρισμα χωρίς δεκαδικά ψηφία. Η σύνταξη της συνάρτησης είναι:

TRUNCATE (X, D), όπου X είναι ο αριθμός και ο D είναι το πλήθος των δεκαδικών ψηφίων που πρέπει ο X να έχει.

Για παράδειγμα:

SELECT TRUNCATE(1.223,1);

Επιστρέφει: **1.2**

SELECT TRUNCATE(1.999,1);

Επιστρέφει: **1.9**

SELECT TRUNCATE(1.999,0);

Επιστρέφει: **1**

SELECT TRUNCATE(-1.999,1);

Επιστρέφει: **-1.9**

SELECT TRUNCATE(10.28*100,0);

Επιστρέφει: **1028**

- **SIGN:** Επιστρέφει το πρόσημο του ορίσματος σαν -1, 0, 1. Αν το όρισμα είναι αρνητικός αριθμός επιστρέφει -1, αν το όρισμα είναι 0 επιστρέφει 0 και αν το όρισμα είναι θετικός αριθμός επιστρέφει 1. Η σύνταξη της συνάρτησης είναι:

SIGN(X)

Για παράδειγμα:

SELECT SIGN(-32);

Επιστρέφει: **-1**

SELECT SIGN(0);

Επιστρέφει: **0**

SELECT SIGN(234);

Επιστρέφει: **1**

9.1.4. Συναρτήσεις ημερομηνίας / ώρας (Date and time functions)

Η MySQL παρέχει ένα σύνολο συναρτήσεων που χρησιμοποιούνται για τη διαχείριση δεδομένων τύπου ημερομηνίας και ώρας. Οι συναρτήσεις αυτές διαφέρουν στις διάφορες υλοποιήσεις της SQL. Στην MySQL οι πιο συχνά χρησιμοποιούμενες συναρτήσεις είναι:

- **NOW()**: Επιστρέφει την τρέχουσα ημερομηνία και ώρα στην μορφή ‘YYYY-MM-DD HH:MM:SS’ όπου YYYY αναπαριστά το έτος με τέσσερις χαρακτήρες π.χ. 2014, το MM αναπαριστά τον μήνα με δύο πάντα χαρακτήρες δηλαδή 01, το DD αναπαριστά την ημέρα με δύο χαρακτήρες π.χ. 08, το HH αναπαριστά την ώρα με δύο χαρακτήρες 09 το MM αναπαριστά τα λεπτά με δύο χαρακτήρες π.χ. 30 και τέλος το SS αναπαριστά τα δευτερόλεπτα με δύο χαρακτήρες π.χ. 45. Το ίδιο αποτέλεσμα μας δίνει και η συνάρτηση **SYSDATE** και **CURRENT_TIMESTAMP**. Η σύνταξή των δύο συναρτήσεων που επιστρέφουν την τρέχουσα ώρα και ημερομηνία είναι:

NOW()

SYSDATE()

CURRENT_TIMESTAMP()

Για παράδειγμα:

SELECT NOW()

Επιστρέφει: **2014-03-12 13:47:36**

SELECT SYSDATE ()

Επιστρέφει: **2014-03-12 13:50:36**

- **DATE_FORMAT**: Μορφοποιεί το πρώτο όρισμα που είναι τύπου ημερομηνίας με τον τρόπο που αναφέρεται στο δεύτερο όρισμα. Η σύνταξη της συνάρτησης είναι:

DATE_FORMAT(date, format), όπου *date* είναι μια ημερομηνία και *format* είναι ο

τρόπος μορφοποίησης αυτής.

Η μορφοποίηση γίνεται με τη χρήση του χαρακτήρα % και των παραμέτρων μορφοποίησης.

Παράμετρος Μορφοποίησης	Περιγραφή
% a	Όνομα ημέρας της εβδομάδας (Sun..Sat)
% b	Όνομα του μήνα (Jan..Dec)
% c	Ο Μήνας, αριθμητικά (0 .. 12)
% D	Η ημέρα του μήνα με την αγγλική κατάληξη (0th, 1st, 2nd, 3rd, ...)
% d	Η Ημέρα του μήνα , αριθμητικά (00 .. 31)
% f	Τα μικροδευτερόλεπτα (000000 .. 999999)
% H	Η ώρα (00 .. 23)
% h	Η ώρα (01 .. 12)
% i	Τα Λεπτά, αριθμητικά (00 .. 59)
% j	Η Ημέρα του έτους (001 .. 366)
% k	Η ώρα (0 .. 23)
% l	Η ώρα (1 .. 12)
%M	Το Όνομα του Μήνα (January..December)
% m	Ο Μήνας , αριθμητικά (00 .. 12)
% p	AM ή PM
% r	Η ώρα , 12 - ωρη της μορφής: hh:mm:ss ακολουθεί AM ή PM
% S	Τα δευτερόλεπτα (00 .. 59)
% s	Τα δευτερόλεπτα (00 .. 59)
% T	Η ώρα , 24 – ωρη της μορφής: hh:mm:ss
% U	Η Εβδομάδα (00 .. 53), όπου η Κυριακή είναι η πρώτη ημέρα της εβδομάδας
% u	Η Εβδομάδα (00 .. 53), όπου η Δευτέρα είναι η πρώτη ημέρα της εβδομάδας
% V	Η Εβδομάδα (01 .. 53), όπου η Κυριακή είναι η πρώτη ημέρα της εβδομάδας Χρησιμοποιείται με %X
% v	Η Εβδομάδα (01 .. 53), όπου η Δευτέρα είναι η πρώτη ημέρα της εβδομάδας. Χρησιμοποιείται με % x

% W	Το Όνομα της Ημέρας της εβδομάδας (Sunday..Saturday)
% w	Η Ημέρα της εβδομάδας (0=Sunday..6=Saturday)
% X	Το Έτος για την εβδομάδα όπου η Κυριακή είναι η πρώτη ημέρα της εβδομάδας. Το έτος αναπαριστάται με τέσσερα ψηφία. Χρησιμοποιείται με την %V
% x	Το Έτος για την εβδομάδα όπου η Δευτέρα είναι η πρώτη ημέρα της εβδομάδας. Το έτος αναπαριστάται με τέσσερα ψηφία. Χρησιμοποιείται με την %v
% Y	Το έτος με τέσσερα ψηφία
% y	Το έτος με δύο ψηφία

Για παράδειγμα

SELECT DATE_FORMAT('2009-10-04 22:23:00', '%W %M %Y');

Επιστρέφει: 'Sunday October 2009'

SELECT DATE_FORMAT('2007-10-04 22:23:00', '%H:%i:%s');

Επιστρέφει: '22:23:00'

SELECT DATE_FORMAT('1900-10-04 22:23:00', '%D %y %a %d %m %b %j');

Επιστρέφει: '4th 00 Thu 04 10 Oct 277'

SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H %k %I %r %T %S %w');

Επιστρέφει: '22 22 10 10:23:00 PM 22:23:00 00 6'

SELECT DATE_FORMAT('1999-01-01', '%X %V');

Επιστρέφει: '1998 52'

SELECT DATE_FORMAT('2006-06-00', '%d');

Επιστρέφει: '00'

- **STR_TO_DATE:** Η συνάρτηση αυτή μετατρέπει το πρώτο όρισμα που είναι ένα αλφαριθμητικό σε ημερομηνία ανάλογα με το δεύτερο όρισμα. Η STR_TO_DATE επιστρέφει μια τιμή τύπου DATETIME αν το αλφαριθμητικό περιέχει τόσο την ημερομηνία όσο και τα μέρη του χρόνου, ή μια τιμή τύπου ημερομηνίας DATE ή μια τιμή τύπου TIME αν το αλφαριθμητικό περιέχει μόνο την ημερομηνία ή τμήματα του χρόνου. Εάν η ημερομηνία, η ώρα, ή η τιμή datetime που εξάγεται από το αλφαριθμητικό είναι παράνομη, τότε η STR_TO_DATE () επιστρέφει NULL. Η σύνταξη της συνάρτησης είναι:

STR_TO_DATE (str, format), όπου *str* είναι το αλφαριθμητικό που αντιστοιχεί σε ημερομηνία και *format* είναι η μορφή ημερομηνίας που ακολουθεί το αλφαριθμητικό. Το *format* αποτελείται από παραμέτρους μορφοποίησης οι οποίες καθορίζουν την μορφή της ημερομηνίας.

Για παράδειγμα:

SELECT STR_TO_DATE('01,5,2013','%d,%m,%Y');

Επιστρέφει: **'2013-05-01'**

SELECT STR_TO_DATE('May 1, 2013','%M %d,%Y');

Επιστρέφει: **'2013-05-01'**

- **DAYNAME**: Η συνάρτηση αυτή επιστρέφει το όνομα της ημέρας του ορίσματος τύπου ημερομηνίας. Η σύνταξη της συνάρτησης είναι:

DAYNAME(date), όπου *date* είναι η ημερομηνία της οποίας τα όνομα της ημέρας επιστρέφεται.

Για παράδειγμα:

SELECT DAYNAME('2007-02-03');

Επιστρέφει: **'Saturday'**

- **DAYOFMONTH**: Η συνάρτηση αυτή επιστρέφει τη μέρα του μήνα από 1 έως 31 του ορίσματος που είναι τύπου ημερομηνία. Η σύνταξη της συνάρτησης είναι:

DAYOFMONTH (date), όπου *date* είναι η ημερομηνία.

Για παράδειγμα:

SELECT DAYOFMONTH ('2007-02-03');

Επιστρέφει: **'3'**

- **DAYOFWEEK**: Η συνάρτηση αυτή επιστρέφει τη μέρα της εβδομάδας από 1...7 (1=Κυριακή,...,7=Σάββατο) του ορίσματος που είναι τύπου ημερομηνία. Η σύνταξη της συνάρτησης είναι:

DAYOFWEEK (date), όπου *date* είναι η ημερομηνία.

Για παράδειγμα:

SELECT DAYOFWEEK ('2007-02-03');

Επιστρέφει: **'7'**

- **DAYOFYEAR**: Η συνάρτηση αυτή επιστρέφει τη μέρα του έτους από 1...366 του ορίσματος που είναι τύπου ημερομηνία. Η σύνταξη της συνάρτησης είναι:

DAYOFYEAR (date), όπου *date* είναι η ημερομηνία.

Για παράδειγμα:

SELECT DAYOFYEAR ('2007-02-03');

Επιστρέφει: '34'

- **DATE_DIFF:** Η συνάρτηση αυτή αφαιρεί δύο ημερομηνίες που αποτελούν τα δύο ορίσματα της συνάρτησης. Αν τα ορίσματα είναι τύπου ημερομηνίας και χρόνου, μόνο το κομμάτι της ημερομηνίας χρησιμοποιείται. Η σύνταξη της συνάρτησης είναι:

DATE_DIFF (expr1, expr2), όπου *expr1* και *expr2* είναι οι ημερομηνίες που αναζητούμε την διαφορά τους.

Για παράδειγμα:

SELECT DATEDIFF('2007-12-31 23:59:59','2007-12-30');

Επιστρέφει: 1

SELECT DATEDIFF('2010-11-30 23:59:59','2010-12-31');

Επιστρέφει: '-31'

- **DATE_ADD και DATE_SUB:** Η συνάρτηση DATE_ADD προσθέτει στο πρώτο όρισμα που είναι τύπου ημερομηνίας το δεύτερο όρισμα *expr*, όπου στο δεύτερο όρισμα ορίζεται η μονάδα μέτρησης του δεύτερου ορίσματος που προστίθεται (*unit*). Η σύνταξη της συνάρτησης είναι:

DATE_ADD(date, INTERVAL expr unit), όπου *date* είναι η αρχική ημερομηνία, *expr* είναι το διάστημα που προσθέτουμε στην αρχική ημερομηνία και *unit* είναι η μονάδα μέτρησης του ορίσματος *expr*.

Η συνάρτηση DATE_SUB αφαιρεί από το πρώτο όρισμα που είναι τύπου ημερομηνίας το δεύτερο όρισμα *expr*, όπου στο δεύτερο όρισμα ορίζεται η μονάδα μέτρησης του δεύτερου ορίσματος που προσθέτεται (*unit*). Η σύνταξη της συνάρτησης είναι:

DATE_SUB (date, INTERVAL expr unit), όπου *date* είναι η αρχική ημερομηνία, *expr* είναι το διάστημα που αφαιρούμε από την αρχική ημερομηνία και *unit* είναι η μονάδα μέτρησης του ορίσματος *expr*.

Το *unit* είναι της μορφής π.χ.:

UNIT	Μορφή expr
MICROSECOND	MICROSECONDS
SECOND	SECONDS
MINUTE	MINUTES
HOUR	HOURS
DAY	DAYS
WEEK	WEEKS

MONTH	MONTHS
QUARTER	QUARTERS
YEAR	YEARS
SECOND_MICROSECOND	'SECONDS.MICROSECONDS'
MINUTE_MICROSECOND	'MINUTES:SECONDS.MICROSECONDS'
MINUTE_SECOND	'MINUTES:SECONDS'
HOUR_MICROSECOND	'HOURS:MINUTES:SECONDS.MICROSECONDS'
HOUR_SECOND	'HOURS:MINUTES:SECONDS'
HOUR_MINUTE	'HOURS:MINUTES'
DAY_MICROSECOND	'DAYS HOURS:MINUTES:SECONDS.MICROSECONDS'
DAY_SECOND	'DAYS HOURS:MINUTES:SECONDS'
DAY_MINUTE	'DAYS HOURS:MINUTES'
DAY_HOUR	'DAYS HOURS'
YEAR_MONTH	'YEARS-MONTHS'

Για παράδειγμα:

SELECT DATE_ADD('2000-12-31 23:59:59', INTERVAL 1 SECOND);

Επιστρέφει: **'2001-01-01 00:00:00'**

SELECT DATE_ADD('2010-12-31 23:59:59', INTERVAL 1 DAY);

Επιστρέφει: **'2011-01-01 23:59:59'**

**SELECT DATE_ADD('2100-12-31 23:59:59', INTERVAL '1:1:1'
MINUTE_SECOND);**

Επιστρέφει: **'2101-01-01 00:01:00'**

**SELECT DATE_SUB('2005-01-01 00:00:00', INTERVAL '1 1:1:1'
DAY_SECOND);**

Επιστρέφει: **'2004-12-30 22:58:59'**

SELECT DATE_ADD('1900-01-01 00:00:00', INTERVAL '-1 10' DAY_HOUR);

Επιστρέφει: **'1899-12-30 14:00:00'**

SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);

Επιστρέφει: **'1997-12-02'**

**SELECT DATE_ADD('1992-12-31 23:59:59.000002', INTERVAL '1.999999'
SECOND_MICROSECOND);**

Επιστρέφει: **'1993-01-01 00:00:01.000001'**

- **CURDATE και CURRENT_DATE.** Οι συναρτήσεις αυτές επιστρέφουν τη

τρέχουνσα ημερομηνία. Η σύνταξη των συναρτήσεων είναι:

CURDATE ()

CURRENT_DATE ()

- **CURTIME και CURRENT_TIME.** Οι συναρτήσεις αυτές επιστρέφουν τη τρέχουνσα ώρα ακολουθώντας τη μορφή 'HH:MM:SS' ή HHMMSS, ανάλογα αν χρησιμοποιείται η συνάρτηση σαν string ή σαν αριθμητικό δεδομένο. Η τιμή αναφέρεται στην τρέχουνσα time zone. Η σύνταξη των συναρτήσεων είναι:

CURTIME ()

CURRENT_TIME ()

9.1.5. Άλλες Συναρτήσεις

Η SQL και κατ επέκταση η MySQL περιλαμβάνουν ένα σύνολο συναρτήσεων για την διαχείριση των αλφαριθμητικών.

- **IFNULL:** Η συνάρτηση αυτή επιστρέφει το πρώτο όρισμα αν δεν είναι NULL ή το δεύτερο αν το πρώτο είναι NULL. Οι τύποι των δύο ορισμάτων πρέπει να είναι ίδιοι. Η σύνταξή της είναι η ακόλουθη:

IFNULL (expr1, expr2)

- **CURRENT_USER:** Η συνάρτηση αυτή επιστρέφει το όνομα του τρέχοντος χρήστη. Η σύνταξή της είναι η ακόλουθη:

CURRENT_USER ()

- **DATABASE:** Η συνάρτηση αυτή επιστρέφει το όνομα της τρέχουνσας βάσης δεδομένων. Η σύνταξή της είναι η ακόλουθη:

DATABASE ().

10. Ανάκτηση Δεδομένων - Σύνθετα ερωτήματα

Μελετώντας το παρόν κεφάλαιο οι επιμορφωνόμενοι θα είναι σε θέση να:

- Ανακτούν δεδομένα εκτελώντας σύνθετα ερωτήματα και ερωτήματα με πράξεις συνόλων.

Στο κεφάλαιο αυτό θα προσεγγίσουμε την ανάκτηση δεδομένων χρησιμοποιώντας σύνθετα ερωτήματα. Συγκεκριμένα θα αναφερθούμε σε ερωτήματα ελέγχου μέλους ενός συνόλου και θα μελετήσουμε τις πράξεις συνόλων της σχεσιακής άλγεβρας και όπως αυτές έχουν ενσωματωθεί στην SQL. Τέλος θα δούμε τις ιδιαιτερότητες που παρουσιάζει η MySQL.

10.1. Μέλος Συνόλου

Η SQL για να ελέγξει αν μια εγγραφή είναι μέλος ενός συνόλου τιμών χρησιμοποιεί τον τελεστή **in**. Με το **not in** ελέγχουμε αν μια εγγραφή δεν είναι μέλος ενός συνόλου τιμών.

Παράδειγμα

Για παράδειγμα για να αναζητήσουμε τους εργαζόμενους ενός Φορέα με μισθούς 1000, 1500, 2000 Ευρώ θα γράψουμε το ακόλουθο ερώτημα:

```
SELECT firstname, lastname, salary
FROM EMPLOYEES
WHERE salary IN (1000, 2000, 1500);
```

Το αποτέλεσμα θα ήταν εγγραφές με εργαζόμενους και μισθό 1000, 1500, 2000 Ευρώ.

<i>firstname</i>	<i>lastname</i>	<i>salary</i>
ΝΙΚΟΣ	ΚΟΥΡΤΗΣ	2000
ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ	1000
ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ	2000

Εικόνα 70 Οι εργαζόμενοι με μισθό 1000, 1500, 2000 Ευρώ

Για να ελέγξουμε αν μια εγγραφή δεν ανήκει σε ένα σύνολο τιμών όπως ήδη παρουσιάσαμε χρησιμοποιούμε τον τελεστή **not in**.

Παράδειγμα

Για παράδειγμα για να αναζητήσουμε τους εργαζόμενους ενός Φορέα με μισθό που δεν περιλαμβάνεται στις τιμές 1000, 1500, 2000 Ευρώ, θα γράψουμε το ακόλουθο ερώτημα:

```
SELECT firstname, lastname, salary
FROM EMPLOYEES
WHERE salary NOT IN (1000, 2000, 1500);
```

Το αποτέλεσμα θα είναι εργαζόμενοι με διαφορετικούς μισθούς από τους: 1000, 1500 και 2000 Ευρώ.

	<i>firstname</i>	<i>lastname</i>	<i>salary</i>
▶	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ	1200
	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ	3000
	ΑΝΝΑ	ΠΑΠΑ	3500

Εικόνα 71

10.2. Κενές Τιμές

Η SQL επιτρέπει την χρήση κενών τιμών για να δείξει την απουσία πληροφοριών στην τιμή ενός γνωρίσματος-στήλης. Η κενή τιμή αναπαρίσταται με την λέξη κλειδί **null**. Ο έλεγχος για την ύπαρξη κενής τιμής γράφεται: **is null**. Έτσι για να βρούμε όλους τους εργαζόμενους που δεν είναι τοποθετημένοι σε ένα τμήμα γράφουμε:

```
SELECT firstname, lastname
FROM EMPLOYEES
WHERE department_id is null;
```

Ο τελεστής **is not null** ελέγχει την απουσία της κενής τιμής.

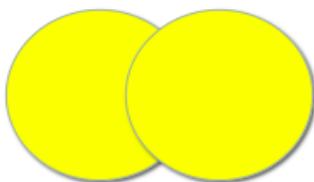
10.3. Πράξεις Συνόλων

Η SQL έχει ενσωματώσει τις βασικότερες από τις πράξεις συνόλων της σχεσιακής άλγεβρας. Οι SQL τελεστές union, intersect, και except λειτουργούν πάνω σε σχέσεις και αντιστοιχούν με τους τελεστές της σχεσιακής άλγεβρας ένωση (v), τομή (Π) και διαφορά (-). Όπως η ένωση, η τομή και η διαφορά στη σχεσιακή άλγεβρα, έτσι και οι

σχέσεις-οι πίνακες-τα ερωτήματα που συμμετέχουν στις πράξεις πρέπει να είναι συμβατές, δηλαδή, πρέπει να έχουν το ίδιο σύνολο γνωρισμάτων-στηλών.

10.3.1. Ο τελεστής Ένωσης

Η πράξη της ένωσης (union) είναι γνωστή από τη σχεσιακή άλγεβρα. Η πράξη της ένωσης μπορεί να εφαρμοστεί σε δύο σύνολα. Τα σύνολα αυτά μπορεί να είναι δύο πίνακες ή τα αποτελέσματα δύο ερωτημάτων. Στην περίπτωση αυτή το παραγόμενο σύνολο περιέχει τις εγγραφές των δύο πινάκων ή τις εγγραφές των αποτελεσμάτων των δύο ερωτημάτων.



Εικόνα 72 Η πράξη της ένωσης δύο συνόλων

Στην πράξη της ένωσης θα πρέπει να ισχύουν οι εξής προϋποθέσεις:

- Το πλήθος των γνωρισμάτων-στηλών των δύο συνόλων-πινάκων-ερωτημάτων πρέπει να είναι το ίδιο.
- Τα γνωρίσματα-στήλες των δύο συνόλων-πινάκων-ερωτημάτων πρέπει να είναι του ίδιου τύπου.

Στην SQL υπάρχουν δύο τύποι ενώσεων:

- Η ένωση union στην οποία το παραγόμενο σύνολο περιέχει τις εγγραφές και των δύο συνόλων-πινάκων ή τα αποτελέσματα και των δύο ερωτημάτων χωρίς διπλότυπες εγγραφές.
- Η ένωση union all στην οποία το παραγόμενο σύνολο περιέχει όλες τις εγγραφές και των δύο συνόλων. Η ένωση αυτή γράφεται ως union all και διατηρεί όλα τα διπλότυπα των δύο συνόλων ή όλα τα διπλότυπα των αποτελεσμάτων των δύο ερωτημάτων (στην περίπτωση της ένωσης δύο ερωτημάτων).

Η υλοποίηση της SQL στην MySQL υποστηρίζει και τους δύο τύπους ενώσεων union και union all.

Παράδειγμα

Φανταστείτε το παράδειγμα των ασφαλιζόμενων στους Ασφαλιστικούς Φορείς ΙΚΑ και Δημοσίου. Στο παράδειγμα αυτό θεωρούμε ότι υπάρχουν ασφαλισμένοι στο ΙΚΑ, ασφαλισμένοι στο Δημόσιο και ασφαλισμένοι και στα δύο ταμεία ΙΚΑ και Δημοσίου. Θεωρείστε ότι μας ζητείται να απαντήσουμε στο ερώτημα «Βρείτε όλους τους ασφαλιζόμενους του ΙΚΑ και του Δημοσίου».

Ασφαλισμένοι ΙΚΑ

	firstname	lastname
▶	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ
	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ
	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ
	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ
	ΑΝΝΑ	ΠΑΠΑ
	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ

UNION

Ασφαλισμένοι Δημοσίου

	firstname	lastname
▶	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ
	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ
	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ
	ΒΑΣΙΛΕΙΟΣ	ΜΑΡΓΑΡΙΤΗΣ

=

Ασφαλισμένοι ΙΚΑ και Δημοσίου

	firstname	lastname
▶	ΜΑΡΙΑ	ΜΠΕΚΙΑΡΗ
	ΝΙΚΟΣ	ΚΟΥΡΤΗΣ
	ΓΕΩΡΓΙΟΣ	ΠΑΠΑΓΕΩΡΓΙΟΥ
	ΒΑΣΙΛΕΙΟΣ	ΚΑΡΑΝΙΚΟΛΑΣ
	ΑΝΝΑ	ΠΑΠΑ
	ΜΑΝΩΛΗΣ	ΠΑΠΑΔΟΠΟΥΛΟΣ
	ΒΑΣΙΛΕΙΟΣ	ΜΑΡΓΑΡΙΤΗΣ

Εικόνα 73 Η πράξη της ένωσης

Για να απαντήσουμε στο ερώτημα αυτό πρέπει να εκτελέσουμε την πράξη της ένωσης στους δύο σχετιζόμενους πίνακες: INSURED_IKA που αντιστοιχεί στους Ασφαλιζόμενους του ΙΚΑ και INSURED_DHMOΣΙΟ που απεικονίζει τους Ασφαλιζόμενους του Δημοσίου. Η εντολή γράφεται:

```
SELECT firstname, lastname FROM INSURED_IKA
UNION
SELECT firstname, lastname FROM INSURED_DHMOΣΙΟ;
```

Η παραπάνω εντολή union αυτόματα απαλείφει τα διπλότυπα. Έτσι στο παραπάνω παράδειγμα αν ένας ασφαλιζόμενος είναι ασφαλισμένος στο ΙΚΑ και ασφαλισμένος στο Δημόσιο θα εμφανιστεί μόνο μία φορά στο αποτέλεσμα (δείτε το παράδειγμα του ΓΕΩΡΓΙΟΥ ΠΑΠΑΓΕΩΡΓΙΟΥ).

Αν θέλουμε να διατηρήσουμε όλα τα διπλότυπα, πρέπει να γράψουμε τον τελεστή union all στη θέση του union, δηλαδή:

```
SELECT firstname, lastname FROM INSURED_IKA  
UNION ALL  
SELECT firstname, lastname FROM INSURED_DHMOΣΙΟ;
```

Η χρήση του τελεστή union all θα εμφανίσει στο αποτέλεσμα όλες τις διπλότυπες εγγραφές. Στην περίπτωση του παραπάνω παραδείγματος, στο αποτέλεσμα θα εμφανιζόταν ο ΓΕΩΡΓΙΟΣ ΠΑΠΑΓΕΩΡΓΙΟΥ δύο φορές, την μία ως ασφαλιζόμενος του ΙΚΑ και την άλλη ως ασφαλιζόμενος του Δημοσίου.

10.3.2. Ο τελεστής Τομής

Η πράξη της τομής (intersect) δύο συνόλων-πινάκων-ερωτημάτων συμβολίζεται με Π και επιστρέφει ένα σύνολο εγγραφών που υπάρχουν και στα δύο σύνολα-πίνακες-ερωτήματα. Και στην περίπτωση αυτή πρέπει να ισχύουν οι ίδιες προϋποθέσεις όπως στην πράξη της ένωσης. Η πράξη της τομής φαίνεται στην Εικόνα 74 Η πράξη της τομής δύο συνόλων

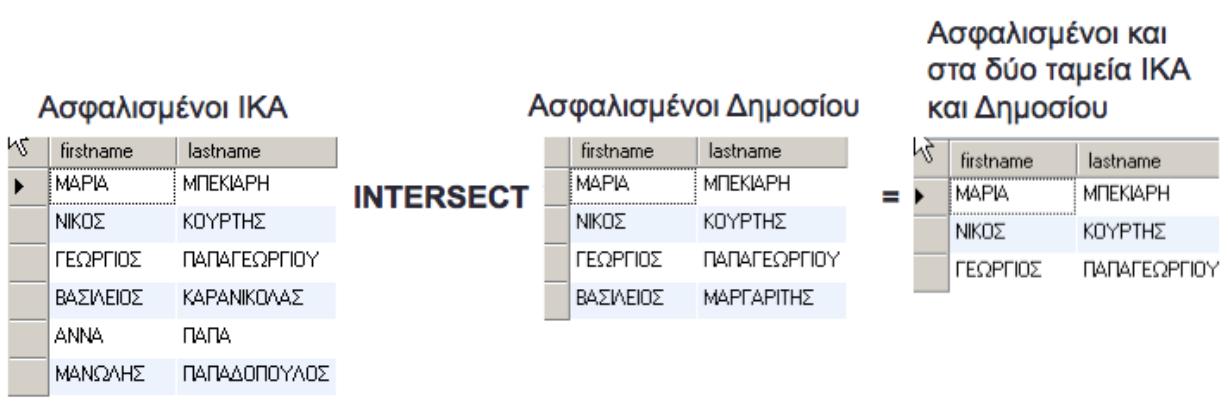


Εικόνα 74 Η πράξη της τομής δύο συνόλων

Η MySQL δεν υποστηρίζει άμεσα την πράξη της τομής δύο συνόλων-πινάκων (intersect). Η τομή δύο πινάκων μπορεί προσημειωθεί με την εσωτερική συνένωση (inner join) δύο πινάκων. Στην εσωτερική συνένωση (inner join) συγκρίνουμε τις εγγραφές του ενός πίνακα με τις εγγραφές του άλλου πίνακα ως προς κάποια γνωρίσματα και επιλέγουμε αυτές που έχουν κοινά γνωρίσματα ενώ παράλληλα εξαιρούνται οι διπλότυπες εγγραφές. Η ίδια λειτουργία ακολουθείται και στην τομή δύο συνόλων. Η τομή δύο συνόλων είναι το σύνολο με τις κοινές τιμές των δύο συνόλων. Επομένως η τομή δύο πινάκων μπορεί να γραφεί και ως η εσωτερική συνένωση (inner join) των δύο πινάκων.

Παράδειγμα

Φανταστείτε το προηγούμενο παράδειγμα, των ασφαλιζόμενων του ΙΚΑ και των ασφαλιζόμενων του Δημοσίου. Θεωρείστε το ερώτημα «Βρείτε τους ασφαλιζόμενους του ΙΚΑ που είναι παράλληλα και ασφαλιζόμενοι του Δημοσίου».



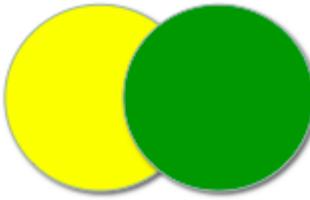
Εικόνα 75 Η πράξη της τομής

Για να απαντήσουμε στο ερώτημα αυτό πρέπει να εκτελέσουμε την πράξη της τομής στους δύο σχετιζόμενους πίνακες: INSURED_IKA και INSURED_DHMOSIO. Θεωρούμε ότι το αφμ των ασφαλιζόμενων (αντιστοιχεί στο γνώρισμα-στήλη afm και των δύο πινάκων) είναι το κοινό γνώρισμα των δύο πινάκων. Επίσης το αφμ χαρακτηρίζει μοναδικά τον κάθε ασφαλιζόμενο. Η εντολή SELECT γράφεται:

```
SELECT firstname, lastname  
FROM INSURED_IKA INNER JOIN INSURED_DHMOSIO  
ON INSURED_IKA.afm=INSURED_DHMOSIO.afm;
```

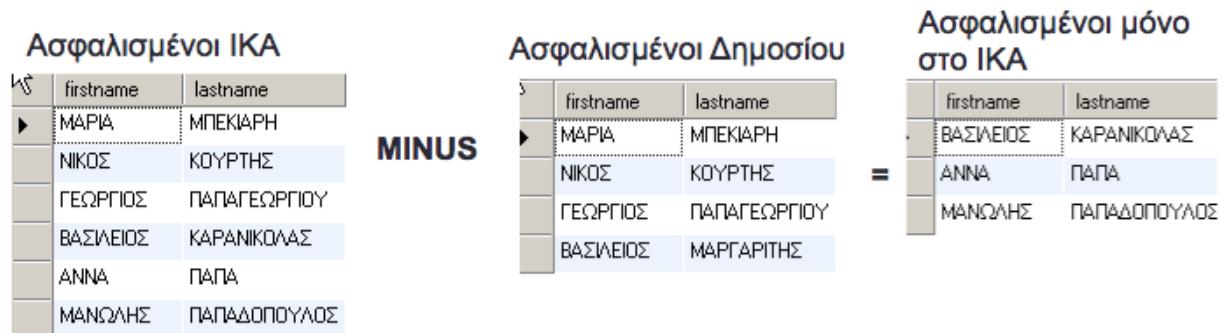
10.3.3. Ο τελεστής της Διαφοράς

Η πράξη της διαφοράς (minus) δύο συνόλων-πινάκων-ερωτημάτων συμβολίζεται με – και χρησιμοποιείται για να απομονώσει τις εγγραφές ενός συνόλου-πίνακα-ερωτήματος, οι οποίες δεν ανήκουν σε κάποιο άλλο σύνολο-πίνακα-ερώτημα. Για την πράξη αυτή ισχύουν επίσης οι ίδιες προϋποθέσεις που ισχύουν στην πράξη της ένωσης. Η πράξη της διαφοράς φαίνεται στην Εικόνα 76 Η πράξη της διαφοράς δύο συνόλων.



Εικόνα 76 Η πράξη της διαφοράς δύο συνόλων

Η MySQL δεν υποστηρίζει άμεσα την πράξη της διαφοράς δύο συνόλων (minus). Αν θεωρήσουμε το παράδειγμα με τους ασφαλιζόμενους του ΙΚΑ και τους ασφαλιζόμενους του Δημοσίου, η διαφορά μεταξύ των πινάκων ασφαλιζόμενοι ΙΚΑ και ασφαλιζόμενοι Δημοσίου περιλαμβάνει τους ασφαλιζόμενους του ΙΚΑ που δεν είναι ασφαλισμένοι και σε άλλο ταμείο.



Εικόνα 77 Η πράξη της διαφοράς

Η διαφορά δύο συνόλων-πινάκων στην MySQL υλοποιείται με δύο τρόπους:

Πρώτος τρόπος: η διαφορά υλοποιείται με την χρήση της αριστερής συνένωσης.

Δηλαδή:

```
SELECT DISTINCT afm, firstname, lastname
FROM INSURED_IKA ika LEFT JOIN INSURED_DHMOSIO dhm
ON ika.afm=dhm.afm
WHERE dhm.afm is null;
```

Δεύτερος τρόπος: η διαφορά υλοποιείται με την χρήση υποερωτημάτων. Τα υποερωτήματα θα παρουσιασθούν στο επόμενο κεφάλαιο. Δηλαδή:

```
SELECT afm, firstname, lastname
FROM INSURED_IKA
WHERE afm NOT IN (SELECT afm FROM INSURED_DHMOSIO);
```

11. Υποερωτήματα – Subqueries

Μελετώντας το παρόν κεφάλαιο οι επιμορφωνόμενοι θα είναι σε θέση να:

- κατανοήσουν τι είναι τα υποερωτήματα (subqueries)
- εξοικειωθούν και να χρησιμοποιήσουν όλους τους τύπους υποερωτήματων (subqueries)
- να κατανοήσουν τα συσχετιζόμενα υποερωτήματα (correlated subqueries) και να αποσαφηνίσουν την διαφορά correlated και uncorrelated υποερωτήματα (subqueries)
- να μετατρέπουν υποερωτήματα (subqueries) σε ερωτήματα με συνενώσεις (joins)

Στο κεφάλαιο αυτό θα αναφερθούμε στην έννοια των υποερωτημάτων. Ένα υποερωτημα είναι μια παράσταση select-from-where που είναι ένθετη μέσα σε ένα άλλο ερώτημα. Επιπρόσθετα, θα παρουσιάσουμε τους διαφορετικούς τύπους υποερωτημάτων και θα μελετήσουμε τις διαφορετικές χρήσεις τους.

11.1. Τι είναι τα υποερωτήματα (subqueries)

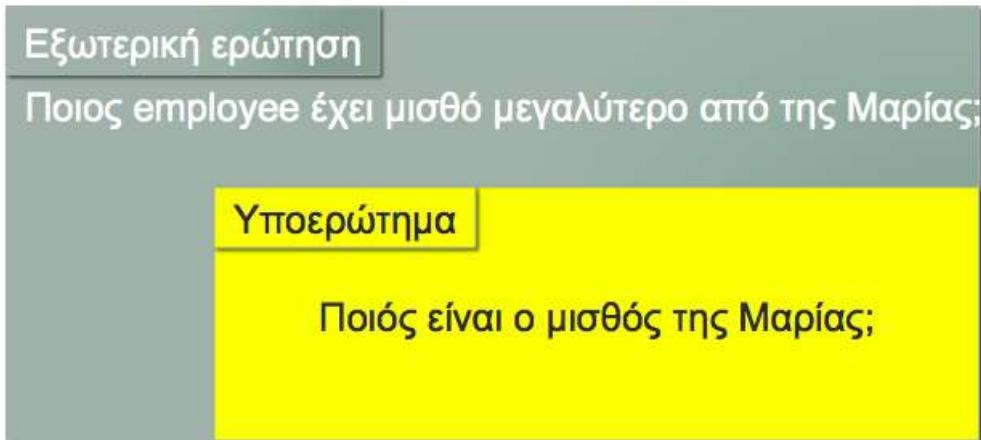
Πολλές φορές παρουσιάζονται ερωτήματα τα οποία προκειμένου να απαντηθούν απαιτούν να ανακτηθούν πρώτα κάποιες τιμές από την βάση δεδομένων και στη συνέχεια να χρησιμοποιηθούν αυτές οι τιμές σε μια συνθήκη σύγκρισης. Οι ερωτήσεις που χρησιμοποιούνται για την ανάκτηση των τιμών αυτών εκφράζονται ως υποερωτήματα ή ένθετα ερωτήματα. Τα υποερωτήματα αυτά είναι πλήρεις προτάσεις SELECT μέσα σε μια πρόταση WHERE ή πρόταση HAVING ή ακόμα και στο FROM κάποιας άλλης ερώτησης που ονομάζεται εξωτερική ερώτηση (outer query) ή κύρια ερώτηση.

Για παράδειγμα ας θεωρήσουμε το ερώτημα: **Ποιος έχει μισθό μεγαλύτερο από αυτόν της Μαρίας;** Για να λυθεί αυτό το πρόβλημα, θα πρέπει να εκτελέσουμε δύο ερωτήματα:

- το ένα για να βρούμε πόσα κερδίζει η Μαρία (ο μισθός της Μαρίας)

- και το δεύτερο ερώτημα για να βρούμε ποιος κερδίζει περισσότερα από το ποσό που κερδίζει η Μαρία.

Τα δύο ερωτήματα φαίνονται στην παρακάτω εικόνα:



Εικόνα 78 Τι είναι το υποερώτημα

Για να λύσουμε αυτό το πρόβλημα τοποθετούμε το ένα ερώτημα μέσα στο άλλο ερώτημα. Το εσωτερικό ερώτημα ή υποερώτημα (Ποιός είναι ο μισθός της Μαρίας) επιστρέφει μια τιμή που χρησιμοποιείται από το εξωτερικό ερώτημα. Η χρήση υποερωτήματος είναι ισοδύναμη με την πραγματοποίηση δύο διαδοχικών ερωτημάτων. Χρησιμοποιούμε το αποτέλεσμα του πρώτου ερωτήματος-υποερωτήματος ως τιμή αναζήτησης στο δεύτερο ερώτημα-κύριο ερώτημα. Η σύνταξη υποερωτημάτων είναι της μορφής:

Εξωτερικό/Κύριο ερώτημα

SELECT select_list_of_columns

FROM tableA

**WHERE expr operator (SELECT select_list_of_columns
FROM tableB)**

Υποερώτημα

Πρώτα εκτελείται το υποερώτημα και το αποτέλεσμα του χρησιμοποιείται στο Εξωτερικό ερώτημα.

Το ερώτημα του παραδείγματος γράφεται:

```
SELECT      lastname, firstname, salary
FROM       EMPLOYEES          1200
```

```

WHERE      salary > (SELECT salary
FROM        EMPLOYEES
WHERE        firstname='MARIA')

```

Το αποτέλεσμα του υποερωτήματος δηλαδή ο μισθός της Μαρίας είναι 1200 Ευρώ. Το ποσό αυτό χρησιμοποιείται στο εξωτερικό ερώτημα. Το αποτέλεσμα του ερωτήματος φαίνεται στην παρακάτω εικόνα.

	lastname	firstname	salary
▶	ΠΑΠΑΓΕΩΡΓΙΟΥ	ΓΕΩΡΓΙΟΣ	3000
	ΚΟΥΡΤΗΣ	ΝΙΚΟΣ	2000
	ΠΑΠΑΔΟΠΟΥΛΟΣ	ΜΑΝΩΛΗΣ	2000
	ΠΑΠΑ	ΑΝΝΑ	1300

Εικόνα 79

11.2. Κατηγορίες υποερωτημάτων

Τα υποερωτήματα (subqueries) τα συναντάμε στην συνθήκη του τελεστή WHERE, στην συνθήκη του τελεστή HAVING, ως οντότητα πίνακα στον τελεστή FROM καθώς και ως παράμετρο μέσα σε συναρτήσεις. Ανάλογα με τις τιμές που επιστρέφουν τα υποερωτήματα στην MySQL χωρίζονται στις εξής κατηγορίες:

- Scalar subqueries : είναι τα υποερωτήματα που επιστρέφουν μια μόνο τιμή, δηλαδή μία εγγραφή με ένα γνώρισμα-στήλη
- Column Subqueries ή Predicate Subqueries: είναι τα υποερωτήματα που επιστρέφουν μία ή περισσότερες εγγραφές ενός γνωρίσματος-στήλης
- Row Subqueries: είναι τα υποερωτήματα που επιστρέφουν μια εγγραφή με ένα ή περισσότερα γνωρίσματα-στήλες.
- Table Subqueries: είναι τα υποερωτήματα που επιστρέφουν ένα πίνακα από τιμές.

Τα table subqueries τοποθετούνται στον τελεστή FROM. Τα υποερωτήματα αυτά επιστρέφουν μια ή περισσότερες εγγραφές με ένα ή περισσότερα γνωρίσματα-στήλες.

Μια πιο γενική κατηγοριοποίηση είναι ο διαχωρισμός των υποερωτημάτων σε δύο κατηγορίες στα υποερωτήματα που επιστρέφουν μια εγγραφή (Single-row subqueries) και στα υποερωτημάτα που επιστρέφουν πολλαπλές εγγραφές (Multi-row subqueries). Στην πρώτη κατηγορία στα Single-row subqueries ανήκουν τα Scalar subqueries και τα Row subqueries. Στην δεύτερη κατηγορία στα Multi-row subqueries ανήκουν τα Column subqueries και τα Table subqueries. Έτσι ανάλογα με τον αριθμό εγγραφών που

επιστρέφουν τα υποερωτήματα διακρίνονται σε:

- Single-row Subqueries που περιλαμβάνουν τα:
 - Scalar subqueries
 - Row subqueries
- Multi-row subqueries που περιλαμβάνουν τα:
 - Column subqueries
 - Table subqueries.

11.3. Single-row Subqueries

Τα single-row subqueries είναι όπως ήδη αναφέραμε τα υποερωτήματα που επιστρέφουν μία μόνο εγγραφή. Στην κατηγορία αυτή ανήκουν τα Scalar Subqueries και τα Row Subqueries. Τα υποερωτήματα αυτά βρίσκονται συνήθως (εκτός από τις περιπτώσεις του Scalar subquery που θα δούμε στην επόμενη παράγραφο) στις προτάσεις με τους τελεστές WHERE και HAVING. Στις περιπτώσεις αυτές για να συγκρίνουμε τις τιμές της εξωτερικής ερώτησης (outer query) με τιμές που επιστρέφουν αυτά τα υποερωτήματα χρησιμοποιούμε μόνο τους παρακάτω τελεστές σύγκρισης:

Τελεστές Σύγκρισης	
=	Ίσος
>	Μεγαλύτερος
>=	Μεγαλύτερος ίσος
<	Μικρότερος
<=	Μικρότερος ίσος
<>	Διαφορετικός

Παράδειγμα

Ας θεωρήσουμε το παράδειγμα στο οποίο αναζητούμε τους υπαλλήλους που εργάζονται στο «ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ» και ο μισθός τους είναι μεγαλύτερος του ΠΑΠΑΓΕΩΡΓΙΟΥ ΙΩΑΝΝΗ. Το ερώτημα γράφεται:

```

SELECT lastname, firstname
FROM employees
WHERE departments_id = (SELECT id
                           FROM departments
                           WHERE department_name =
                                 'ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ')
                           AND salary > (SELECT salary
                           FROM employees
                           WHERE lastname = 'ΠΑΠΑΓΕΩΡΓΙΟΥ' AND
                                 firstname='ΙΩΑΝΝΗΣ');

```

Στο παράδειγμα αυτό υλοποιούνται ένα εξωτερικό ερώτημα και 2 υποερωτήματα, το ένα επιστρέφει το γνώρισμα-στήλη id του Τμήματος Πληροφορικής και το δεύτερο ανακτά τον μισθό του ΙΩΑΝΝΗ ΠΑΠΑΓΕΩΡΓΙΟΥ προκείμενου να συγκριθεί ο μισθός του με τους μισθούς των υπαλλήλων του Τμήματος Πληροφορικής.

Το αποτέλεσμα του ερωτήματος είναι:

	lastname	firstname
▶	ΚΟΥΡΤΗΣ	ΝΙΚΟΣ
	ΠΑΠΑΔΟΠΟΥΛΟΣ	ΜΑΝΩΛΗΣ

Εικόνα 80

Παράδειγμα

Για παράδειγμα φανταστείτε ότι απαιτείται να αναζητήσετε τους εργαζόμενους με τον μεγαλύτερο μισθό στην Υπηρεσία. Το ερώτημα γράφεται:

```

SELECT lastname, firstname, salary
FROM employees
WHERE salary = (SELECT max(salary)
                           FROM employees);

```

Στο παράδειγμα αυτό έχει υλοποιηθεί μία εξωτερική ερώτηση και ένα υποερώτημα. Το υποερώτημα χρησιμοποιεί τη συναθροιστική συνάρτηση (aggregate function) *max* για να βρούμε το μεγαλύτερο μισθό που έχει εργαζόμενος στην Υπηρεσία. Το αποτέλεσμα είναι:

	lastname	firstname	salary
▶	ΠΑΠΑΓΕΩΡΓΙΟΥ	ΓΕΩΡΓΙΟΣ	3000

Εικόνα 81

Όπως αναφέραμε αρχικά τα υποερωτήματα αυτά τα συναντάμε εκτός από τον τελεστή WHERE και στον τελεστή HAVING. Παρακάτω θα δούμε ένα παράδειγμα με υποερώτημα στον τελεστή HAVING.

Παράδειγμα

Φανταστείτε το παράδειγμα στο οποίο αναζητάμε τα τμήματα της Υπηρεσίας (departments_id) που έχουν ελάχιστο μισθό μεγαλύτερο από τον ελάχιστο μισθό του Τμήματος Πληροφορικής. Ας θεωρήσουμε ότι γνωρίζουμε ότι το department_id=1 αντιστοιχεί στο Τμήμα Πληροφορικής.

```
SELECT      department_id , min(salary)
FROM        employees
GROUP BY    department_id
HAVING min(salary) > ( SELECT  min(salary)
                          FROM   employees
                          ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
                          WHERE  department_id=1 );
```

Στο παράδειγμα αυτό το υποερώτημα βρίσκεται στον τελεστή HAVING και συγκρίνεται εκεί με τον τελεστή μεγαλύτερο. Το υποερώτημα επιστρέφει τον ελάχιστο μισθό του τμήματος Πληροφορικής, ενώ η εξωτερική ερώτηση ομαδοποιεί τους εργαζόμενους ανά τμήμα, υπολογίζει τον ελάχιστο μισθό του τμήματος και κατόπιν ελέγχει αν ο ελάχιστος μισθός είναι μεγαλύτερος από το αποτέλεσμα του υποερωτήματος. Αν η συνθήκη είναι αληθής εμφανίζει το τμήμα στα αποτελέσματα του ερωτήματος διαφορετικά δεν τον εμφανίζει. Το ΣΔΒΔ εκτελεί πρώτα το υποερώτημα και κατόπιν επιστρέφει το αποτέλεσμα στο HAVING της εξωτερικής ερώτησης.

Παράδειγμα

Ας υποθέσουμε ότι αναζητούμε τους υπαλλήλους της Υπηρεσίας που έχουν μισθό ίσο με τους μεγαλύτερους μισθούς κάθε τμήματος της Υπηρεσίας.

```

SELECT lastname, firstname , salary
FROM employees
WHERE salary = ( SELECT max(salary)
                  FROM employees
                  GROUP BY department_id);

```

Το ερώτημα αυτό αν εκτελεστεί αποτυγχάνει. Στην MySQL θα επιστρέψει το:

Error Code: 1242 Subquery returns more than 1 row

Ο λόγος της αποτυχίας είναι διότι το υποερώτημα επιστρέφει περισσότερες της μιας εγγραφές. Το υποερώτημα επιστρέφει εγγραφές με τους μεγαλύτερους μισθούς κάθε τμήματος της Υπηρεσίας. Ο τελεστής σύγκρισης = δεν είναι κατάλληλος. Θα δούμε στην συνέχεια ότι το παραπάνω ερώτημα για να εκτελεστεί χρειάζεται, αντί του τελεστή σύγκρισης =, τον τελεστή IN.

11.3.1. Scalar Subqueries

Tα scalar subqueries μπορούν να χρησιμοποιηθούν σχεδόν οπουδήποτε μέσα σε μια πρόταση SELECT. Ουσιαστικά σε οποιοδήποτε σημείο που επιτρέπεται μια τιμή από το συντακτικό της SQL. Έτσι μπορούμε να δούμε υποερωτήματα ως παράμετροι συναρτήσεων ή μπορούμε να εφαρμόσουμε μαθηματικές πράξεις πάνω σε scalar subqueries που επιστρέφουν αριθμητικές τιμές.

Παράδειγμα

```

SELECT CONCAT('Ο μισθός του κ. Παπαγεωργίου είναι: ',
(SELECT salary FROM EMPLOYEES
WHERE lastname='ΠΑΠΑΓΕΩΡΓΙΟΥ') as S1;

```

Το αποτέλεσμα θα είναι:

S1

Ο μισθός του κ. Παπαγεωργίου είναι:1500

Στο παράδειγμα αυτό το υποερώτημα αποτελεί την δεύτερη παράμετρο της συνάρτησης CONCAT.

Παράδειγμα

Φανταστείτε το παράδειγμα στο οποίο αναζητάμε την αναλογία των εργαζόμενων που είναι τοποθετημένοι στο Τμήμα Πληροφορικής (με department_id=1) σε σύγκριση με το σύνολο των εργαζόμενων του Φορέα.

```
SELECT (SELECT COUNT(*) FROM EMPLOYEES WHERE  
department_id='1') / (SELECT COUNT(*) FROM EMPLOYEES ) as RATIO;
```

Το αποτέλεσμα θα είναι:

RATIO
0,24

Στο παράδειγμα αυτό εκτελέστηκε η πράξη της διαίρεσης με διαιρέτη και διαιρετέο υποερωτήματα.

11.3.2. Row Subqueries

Τα Row subqueries όπως έχουμε ήδη αναφέρει είναι τα υποερωτήματα που επιστρέφουν μία και μόνο εγγραφή με πολλαπλά όμως γνωρίσματα στήλες. Στα Row subqueries εκτελούμε τον τελεστή σύγκρισης = σε όλα τα γνωρίσματα-στήλες της εγγραφής.

```
SELECT * FROM employees  
WHERE (lastname,firstname) = (SELECT lastname,firstname  
                                FROM employees  
                                WHERE afm='099999999');
```

Το παραπάνω ερώτημα ανήκει στην κατηγορία Row Subqueries, αφού καταρχήν επιστρέφει μία μόνο εγγραφή, δηλαδή τον εργαζόμενο με αφμ 099999999 και κατά δεύτερον στην συνθήκη WHERE συγκρίνεται η πλειάδα (lastname, fitstname) του πίνακα της εξωτερικής ερώτησης με την ομοίου τύπου πλειάδα που επιστρέφεται από

τον πίνακα του υποερωτήματος. Ουσιαστικά μέσω των πλειάδων συγκρίνουμε το employees.lastname με το tempemployees.lastname και το employees.firstname με το tempemployees.firstname. Επίσης, αν στο υποερώτημα απουσίαζε η συνθήκη του WHERE, το υποερώτημα θα επέστρεφε πολλές εγγραφές και επομένως δεν θα άνηκε στην κατηγορία Row Subqueries.

Η πλειάδα (lastname, firstname) μπορεί να γραφεί και ROW(lastname, firstname). Η πρόταση Row(column1, column2, ..., columnN) δημιουργεί μια πλειάδα τιμών και είναι γνωστή ως “row constructor”. Το παραπάνω ερώτημα μπορεί να γραφεί εναλλακτικά και ως:

```
SELECT * FROM employees
WHERE ROW(lastname,firstname) = (SELECT lastname,firstname
FROM tempemployees
WHERE afm='099999999');
```

11.4. Multiple-row Subqueries

Τα multiple-row subqueries είναι όπως ήδη αναφέραμε τα υποερωτήματα που επιστρέφουν πολλαπλές εγγραφές. Στην κατηγορία αυτή ανήκουν τα Column Subqueries και τα Table Subqueries. Τα υποερωτήματα αυτά βρίσκονται στους τελεστές WHERE, HAVING (Column Subqueries) και στον τελεστή FROM (Table Subqueries).

Στην περίπτωση που τα υποερωτήματα βρίσκονται στην συνθήκη του WHERE και HAVING, για να συγκρίνουμε τις τιμές της εξωτερικής ερώτησης (outer query) με τις τιμές που επιστρέφουν τα υποερωτήματα χρησιμοποιούμε τους παρακάτω τελεστές σύγκρισης:

Τελεστές Σύγκρισης	
IN	Ισος με οποιοδήποτε μέλος της λίστας
ANY	Συγκρίνεται με κάθε τιμή που επιστρέφει το subquery
ALL	Συγκρίνεται με όλες τις τιμές που επιστρέφει το subquery
SOME	Ομοίως όπως ο τελεστής ANY
EXISTS, NOT EXISTS	Ελέγχεται αν το υποερώτημα

11.4.1. Με χρήση του τελεστή ALL

Ο τελεστής ALL τοποθετείται στην συνθήκη του WHERE της εξωτερικής ερώτησης πριν το υποερώτημα. Η SQL επιτρέπει τις συγκρίσεις > ALL, < ALL, <= ALL, >= ALL, = ALL και <> ALL. Χρησιμοποιώντας τον τελεστή ALL περιορίζουμε το σύνολο των αποτελεσμάτων μόνο σε εκείνες τις εγγραφές όπου η σύγκριση ισχύει για **όλες** τις τιμές που παράγονται από το υποερώτημα.

Παράδειγμα

Ας θεωρήσουμε ότι μας ζητείται να εμφανίσουμε όλους τους υπαλλήλους μιας Υπηρεσίας με μισθό μεγαλύτερο από το μισθό **όλων** των υπαλλήλων του τμήματος πληροφορικής (department_id = 1). Το ερώτημα με την χρήση υποερωτημάτων μπορεί να γραφεί:

```
SELECT lastname, firstname, department_id, salary FROM employees
WHERE salary > ALL (SELECT salary
FROM employees
WHERE department_id=1)
AND department_id<>1;
```

Το υποερώτημα επιστρέφει όλους τους μισθούς των εργαζόμενων που είναι τοποθετημένοι στο Τμήμα Πληροφορικής. Ο τελεστής > ALL στην εξωτερική ερώτηση περιορίζει την εμφάνιση των υπαλλήλων σε εκείνους που έχουν μισθό μεγαλύτερο από το όλους τους μισθούς των εργαζόμενων του Τμήματος Πληροφορικής. Η δεύτερη συνθήκη στο WHERE της εξωτερικής ερώτησης (department_id <> 1) χρησιμοποιείται για να εξαιρέσουμε τους εργαζόμενους του Τμήματος Πληροφορικής.

Παράδειγμα

Ας φανταστούμε ότι μας ζητείτε να εμφανίσουμε το τμήμα ή τα τμήματα με τους περισσότερους υπαλλήλους. Το ερώτημα γράφεται:

```
SELECT department_id  
FROM employees  
GROUP BY department_id  
HAVING COUNT(*) >= ALL (SELECT COUNT(*)  
                           FROM employees  
                           GROUP BY department_id);
```

Το υποερώτημα ομαδοποιεί τους εργαζόμενους της Υπηρεσίας ανά τμήμα και κατόπιν υπολογίζει το πλήθος των εγγραφών ανά ομάδα δηλαδή ανά τμήμα. Έτσι επιστρέφεται ένα σύνολο τιμών που αντιστοιχεί στο πλήθος των εργαζομένων ανά τμήμα. Η εξωτερική ερώτηση ομαδοποιεί ομοίως τους εργαζόμενους ανά τμήμα, κατόπιν υπολογίζει τον πλήθος των εργαζόμενων ανά τμήμα και εμφανίζει τα τμήματα εκείνα που το πλήθος των εργαζόμενών τους είναι μεγαλύτερο ή ίσο από όλες τις τιμές του υποερωτήματος.

11.4.2. Με χρήση του τελεστή ANY

Ο τελεστής ANY, ομοίως με τον τελεστή ALL που αναφέραμε, τοποθετείται στην συνθήκη του WHERE της εξωτερικής ερώτησης πριν το υποερώτημα. Η SQL επιτρέπει τις συγκρίσεις > ANY, < ANY, >= ANY, <= ANY, = ANY και <> ANY. Ο τελεστής ANY ακολουθεί πάντα τον τελεστή σύγκρισης (δηλαδή >, <, >=, <=, =, <>) και επιστρέφει «TRUE» αν η σύγκριση είναι αληθής για οποιαδήποτε τιμή του συνόλου τιμών – για τουλάχιστον μία τιμή που επιστρέφει το υποερώτημα.

Παράδειγμα

Ας θεωρήσουμε το παράδειγμα που μας ζητείται να εμφανίσουμε τους εργαζόμενους με μισθό μεγαλύτερο από το μικρότερο μισθό των εργαζόμενων του Τμήματος Πληροφορικής (department_id=1). Το ερώτημα γράφεται:

```
SELECT lastname, firstname, department_id, salary FROM employees
```

```

WHERE salary> ANY (SELECT salary
                     FROM employees
                     WHERE department_id=1)

                     AND department_id<>1;

```

Το υποερώτημα επιστρέφει τους μισθούς των εργαζόμενων του Τμήματος Πληροφορικής που είναι 1200 Ευρώ, 2000 Ευρώ, 2000 Ευρώ. Η εξωτερική ερώτηση επιστρέφει τους εργαζόμενους με μισθό μεγαλύτερο από οποιοδήποτε μισθό των εργαζομένων του Τμήματος Πληροφορικής. Για να εμφανιστεί στο αποτέλεσμα του ερωτήματός μας ένας εργαζόμενος, αρκεί ο μισθός του να είναι μεγαλύτερος από τον μικρότερο μισθό του Τμήματος Πληροφορικής, δηλαδή μεγαλύτερος από 1200 Ευρώ. Αρχικά η SQL επέτρεπε μόνο την χρήση του τελεστή ANY. Οι μετέπειτα εκδόσεις της πρόσθεσαν την εναλλακτική χρήση του SOME για να αποφεύγουμε την ασάφεια της λέξης ANY (που σημαίνει οποιοδήποτε) στα αγγλικά. Η SQL υποστηρίζει ομοίως τις συγκρίσεις >SOME, < SOME, >= SOME, <= SOME, <> SOME.

11.4.3. Με χρήση των τελεστών IN και NOT IN

Ο τελεστής IN, όπως αναφέραμε σε προηγούμενη ενότητα ελέγχει αν μια τιμή περιλαμβάνεται σε ένα σύνολο τιμών. Ενώ ο τελεστής NOT IN επιστρέφει TRUE αν η τιμή δεν περιλαμβάνεται στο σύνολο τιμών. Ακριβώς η ίδια λογική ακολουθείται και στα υποερωτήματα. Ένα υποερώτημα επιστρέφει ένα σύνολο τιμών. Ο τελεστής IN καθώς και ο τελεστής NOT IN βρίσκονται στην συνθήκη WHERE της εξωτερικής ερώτησης και περιορίζουν τα αποτελέσματα της εξωτερικής ερώτησης με το να ελέγχεται αν η τιμή ενός γνωρίσματος-στήλης του πίνακα της εξωτερικής ερώτησης περιλαμβάνεται (IN) ή δεν περιλαμβάνεται (NOT IN) στο σύνολο τιμών που επιστρέφει το υποερώτημα. Ο τελεστής IN είναι ισοδύναμος με τον τελεστή =ANY και τον =SOME. Ο =ANY και ο =SOME σημαίνουν ίσος με οποιονδήποτε τιμή ενός συνόλου τιμών κάτι που ισοδυναμεί με την λειτουργία του τελεστή IN.

Παράδειγμα

Βρείτε τους εργαζόμενους που αμείβονται με μισθό όσο οι μέγιστοι μισθοί κάθε τμήματος.

```
SELECT lastname, firstname , department_id, salary  
FROM employees  
WHERE salary IN ( SELECT max(salary)  
                  FROM employees  
                  GROUP BY department_id);
```

Το υποερώτημα επιστρέφει τους μέγιστους μισθούς κάθε τμήματος της Υπηρεσίας. Επιστρέφει δηλαδή ένα σύνολο τιμών της μορφής (2000, 3000, 1800, 2500). Η εξωτερική ερώτηση εμφανίζει τους εργαζόμενους που έχουν μισθό ίσο με κάποια τιμή από το σύνολο τιμών που επιστρέφει το υποερώτημα, δηλαδή τους εργαζόμενους με μισθούς 2000 Ευρώ, ή 3000 Ευρώ, ή 1800 Ευρώ, ή 2500 Ευρώ.

11.4.4. Με χρήση των τελεστών EXISTS και NOT EXISTS

Ο τελεστής EXISTS τοποθετείται στην συνθήκη του WHERE της εξωτερικής ερώτησης πριν το υποερώτημα. Η SQL περιλαμβάνει το EXISTS για να ελέγχεται αν το υποερώτημα επιστρέφει εγγραφές στο αποτέλεσμά του, χωρίς όμως να ενδιαφερόμαστε για το ποιες είναι οι τιμές αυτές. Έτσι, το EXISTS επιστρέφει TRUE αν το υποερώτημα επιστρέφει εγγραφές και FALSE αν το αποτέλεσμα του υποερωτήματος είναι κενό. Για να ελέγξουμε την μη ύπαρξη εγγραφών ενός υποερωτήματος χρησιμοποιούμε τον τελεστή NOT EXISTS. Συνήθως, το υποερώτημα μετά το EXISTS ξεκινά με SELECT *, αλλά μπορεί να ξεκινά και με SELECT column1 όπου column1 είναι το όνομα ενός γνωρίσματος-στήλης του πίνακα που βρίσκεται στο υποερώτημα. Η MySQL αγνοεί την λίστα γνωρισμάτων-στηλών του SELECT στο υποερώτημα, το μόνο που ελέγχεται είναι αν επιστρέφονται εγγραφές.

Παράδειγμα

Ας θεωρήσουμε το παράδειγμα όπου μας ζητούν να εμφανίσουμε όλα τα τμήματα που έχουν τοποθετημένους εργαζόμενους. Το ερώτημα με την χρήση του EXISTS γράφεται:

```
SELECT * FROM departments d  
WHERE EXISTS (SELECT *  
               FROM employees e  
               WHERE e.department_id = d.id );
```

Το υποερώτημα επιστρέφει τους εργαζόμενους που είναι τοποθετημένοι στο τμήμα με department_id = d.id, όπου d.id είναι το τμήμα που επιστρέφει η εξωτερική ερώτηση. Αναλυτικότερα, το d.id είναι αναφορά στον πίνακα departments ή αλλιώς d της εξωτερικής ερώτησης. Δηλαδή το παραπάνω ερώτημα, για όλα τα τμήματα του πίνακα departments, ελέγχει αν για κάθε τμήμα A υπάρχουν εγγραφές στον πίνακα employees με department_id ίσο με το id του τμήματος A.

Το αντίστροφο του παραπάνω ερωτήματος είναι:

Παράδειγμα

Ας θεωρήσουμε το παράδειγμα όπου μας ζητούν να εμφανίσουμε όλα τα τμήματα που δεν έχουν τοποθετημένους εργαζόμενους. Το ερώτημα με την χρήση του NOT EXISTS γράφεται:

```
SELECT * FROM departments d  
WHERE NOT EXISTS (SELECT *  
                   FROM employees e  
                   WHERE e.department_id = d.id );
```

Το παραπάνω ερώτημα για όλα τα τμήματα του πίνακα departments, ελέγχει αν για το κάθε τμήμα A δεν υπάρχουν εγγραφές στον πίνακα employees με department_id το id του τμήματος A. Το ερώτημα αυτό επιστρέφει τα τμήματα εκείνα για τα οποία δεν υπάρχουν τοποθετημένοι εργαζόμενοι στο πίνακα employees.

11.4.5. Single – Column Subqueries

Όλα τα παραδείγματα που αναφέρθηκαν στην ενότητα Multiple Row Subqueries έως τώρα ήταν παραδείγματα υποερωτημάτων που επιστρέφουν πολλές εγγραφές ενός μόνο γνωρίσματος-στήλης του πίνακα του υποερωτήματος (πλην της παραγράφου «Με χρήση των τελεστών EXISTS και NOT EXISTS»). Τα υποερωτήματα αυτά που επιστρέφουν πολλαπλές εγγραφές ενός μόνο γνωρίσματος-στήλης του πίνακα του υποερωτήματος, ονομάζονται Single-column Subqueries.

Παράδειγμα

Ας θεωρήσουμε ότι μας ζητούν να βρούμε τους εργαζόμενους που αμείβονται με μισθό όσο οι ελάχιστοι μισθοί κάθε τμήματος. Το ερώτημα γράφεται:

```
SELECT lastname, firstname , department_id, salary  
FROM employees  
WHERE salary IN ( SELECT min(salary)  
                   FROM employees  
                   GROUP BY department_id);
```

Το υποερώτημα επιστρέφει τους ελάχιστους μισθούς κάθε τμήματος. Η εξωτερική ερώτηση εμφανίζει τους εργαζόμενους που έχουν μισθό ίσο με κάποια τιμή από το σύνολο τιμών που επιστρέφει το υποερώτημα, δηλαδή τους ελάχιστους μισθούς των τμημάτων της Υπηρεσίας.

11.4.6. Correlated Subqueries

Το υποερώτημα των παραδειγμάτων της ενότητας «Με την χρήση των τελεστών EXISTS και NOT EXISTS» στην συνθήκη του τελεστή WHERE παρουσίαζε μια ιδιομορφία σε σύγκριση με άλλες συνθήκες άλλων υποερωτημάτων. Συγκεκριμένα, στην συνθήκη του υποερωτήματος το γνώρισμα-στήλη του πίνακα υποερωτήματος συγκρινόταν με το γνώρισμα-στήλη του πίνακα της εξωτερικής ερώτησης. Υπήρχε δηλαδή μέσα στο υποερώτημα αναφορά σε γνώρισμα-στήλη εκτός υποερωτήματος. Correlated Subqueries είναι τα υποερωτήματα που περιέχουν τουλάχιστον μια αναφορά σε γνωρίσματα-στήλες πινάκων που εμφανίζονται στην εξωτερική ερώτηση.

Παράδειγμα

Αν για παράδειγμα μας ζητηθεί να βρούμε τα τμήματα που έχουν τουλάχιστον από 2 και πάνω υπαλλήλους. Το ερώτημα γράφεται ως εξής:

```
SELECT department_name  
FROM departments d  
WHERE 2 <= (SELECT count(*)  
             FROM employees  
            WHERE department_id = d.id);
```

Το υποερώτημα απαριθμεί τους εργαζόμενους που είναι τοποθετημένοι στο τμήμα που αναφέρεται στην εξωτερική ερώτηση. Στο ερώτημα αυτό εμφανίζονται τα τμήματα που έχουν πάνω από 2 εργαζόμενους.

11.4.7. Table Subqueries

Τα Table Subqueries είναι τα υποερωτήματα που εμφανίζουν πολλαπλές εγγραφές πολλαπλών γνωρισμάτων-στηλών του πίνακα του υποερωτήματος. Η επιπλέον διαφορά με τα υπόλοιπα υποερωτήματα είναι ότι τα Table Subqueries βρίσκονται όχι στην συνθήκη του WHERE και HAVING αλλά στο FROM της εξωτερικής ερώτησης. Τα υποερωτήματα αυτά χρησιμοποιούνται ως πίνακες στο FROM και πραγματοποιείται αναζήτηση από την εξωτερική ερώτηση πάνω στα αποτελέσματα-εγγραφές αυτών των υποερωτημάτων.

Παράδειγμα

Για παράδειγμα, θεωρείστε ότι μας ζητείται να βρούμε τα ονόματα των τμημάτων που έχουν προϊσταμένους καθώς και το όνομα της διεύθυνσης που αυτά ανήκουν. Ας θεωρήσουμε την υλοποίηση που ο πίνακας EMPLOYEES περιλαμβάνει το γνώρισμα-στήλη manager_id στο οποίο δηλώνεται για κάθε εργαζόμενο ο προϊστάμενός του. Το ερώτημα θα γραφεί:

```
SELECT department_name, directorate_name  
FROM departments dep, directorates dir,
```

```

(SELECT distinct department_id
FROM employees
WHERE manager_id IS NOT null) as emp
WHERE emp.department_id=dep.id AND dep.directorate_id=dir.id;

```

Το υποερώτημα (κόκκινα γράμματα) είναι της κατηγορίας Table Subquery. Το υποερώτημα αυτό βρίσκεται στο FROM της εξωτερικής ερώτησης και επιστρέφει όλα τα department_id στα οποία είναι τοποθετημένοι εργαζόμενοι και αυτοί οι εργαζόμενοι έχουν προϊσταμένους. Η συνθήκη της ύπαρξης προϊσταμένου βρίσκεται στο WHERE του υποερωτήματος και είναι η: «**manager_id IS NOT null**». Το υποερώτημα λειτουργεί ως πίνακας και είναι απαραίτητο ως πίνακας στο FROM να έχει κάποιο όνομα. Το όνομα του πίνακα, που προκύπτει από τα αποτελέσματα του υποερωτήματος, ακολουθεί το υποερώτημα και είναι το **emp**. Στην συνθήκη του WHERE της εξωτερικής ερώτησης πραγματοποιείται η συνένωση των κοινών γνωρισμάτων των τριών πινάκων που χρησιμοποιούνται, δηλαδή του πίνακα emp που αναφέρεται στο υποερώτημα, του πίνακα dep που αναφέρεται στον πίνακα DEPARTMENTS και του πίνακα dir που αναφέρεται στον πίνακα DIRECTORATES, ο οποίος αντιστοιχεί στον πίνακα με τις Διευθύνσεις του Φορέα.

11.5. Μετατροπή υποερωτημάτων σε ερωτήματα με τη χρήση συνενώσεων (JOIN)

Τα υποερωτήματα πολλές φορές είναι ιδιαίτερα χρονοβόρα και έχουν μεγάλο υπολογιστικό κόστος, αφού πολλές φορές χρειάζεται να επαναϋπολογίσουμε το υποερώτημα για κάθε εγγραφή της εξωτερικής ερώτησης. Αρκετές φορές είναι δυνατή η μετατροπή των υποερωτημάτων σε ερωτήματα με την χρήση συνενώσεων (JOIN) που είδαμε σε προηγούμενα κεφάλαια. Τα ερωτήματα με την χρήση συνενώσεων είναι λιγότερο χρονοβόρα και με μικρότερο υπολογιστικό κόστος από αυτό των υποερωτημάτων. Σε άλλες όμως περιπτώσεις δεν είναι δυνατό να ξαναγραφεί το ερώτημα χωρίς την χρήση υποερωτημάτων.

Για παράδειγμα, αν θέλουμε να ελέγξουμε αν κάποιες τιμές είναι μέρη ενός συνόλου τιμών δεν χρειάζεται να χρησιμοποιήσουμε υποερωτήματα με το IN. Το ερώτημα αυτό μπορεί να γραφεί με την χρήση της εσωτερικής συνένωσης. Επομένως:

```
SELECT * FROM t1 WHERE id IN (SELECT id FROM t2);
```

Γράφεται με την χρήση εσωτερικής συνένωσης ως:

```
SELECT t1.* FROM t1 INNER JOIN t2 ON t1.id=t2.id;
```

Παράδειγμα

Ας θεωρήσουμε ότι μας ζητούν να εμφανίσουμε όλα τα τμήματα που έχουν τοποθετημένους εργαζόμενους. Το ερώτημα γράφεται:

Με την χρήση του τελεστή IN:

```
SELECT * FROM DEPARTMENTS  
WHERE id IN (SELECT DISTINCT department_id  
FROM EMPLOYEES  
WHERE department_id is not null);
```

Με την χρήση εσωτερικής συνένωσης:

```
SELECT DISTINCT DEPARTMENTS.* FROM DEPARTMENTS  
INNER JOIN EMPLOYEES ON  
DEPARTMENTS.id= EMPLOYEES.department_id;
```

Επίσης αν θέλουμε να ελέγξουμε αν κάποιες τιμές δεν είναι μέρη ενός συνόλου τιμών δεν χρειάζεται να χρησιμοποιήσουμε υποερωτήματα με το NOT IN ή το NOT EXISTS αλλά τα ερωτήματα αυτά μπορούν να γραφούν με την χρήση της εξωτερικής συνένωσης. Επομένως:

```
SELECT * FROM t1 WHERE id NOT IN (SELECT id FROM t2);
```

```
SELECT * FROM t1 WHERE NOT EXISTS (SELECT id FROM t2 WHERE  
t1.id=t2.id);
```

Γράφονται με την χρήση εξωτερικής συνένωσης ως:

```
SELECT table1.* FROM table1 LEFT JOIN table2 ON table1.id=table2.id  
WHERE table2.id IS NULL;
```

Παράδειγμα

Ας θεωρήσουμε το ίδιο παράδειγμα που αναφέραμε στην ενότητα «Με χρήση των τελεστών EXISTS και NOT EXISTS» όταν παρουσιάζαμε τον τελεστή NOT EXISTS. Δηλαδή: μας ζητούν να εμφανίσουμε όλα τα τμήματα που δεν έχουν τοποθετημένους εργαζόμενους. Το ερώτημα γράφεται:

Με την χρήση τελεστή NOT EXISTS

```
SELECT * FROM departments d  
WHERE NOT EXISTS (SELECT *  
                   FROM employees e  
                   WHERE d.id=e.department_id )
```

Με την χρήση εξωτερικής συνένωσης:

```
SELECT departments.* FROM departments LEFT JOIN employees ON  
departments.id=employees.department_id  
WHERE employees.department_id IS NULL;
```

12. Όψεις (Views)

Μελετώντας το παρόν κεφάλαιο οι επιμορφωνόμενοι θα είναι σε θέση να:

- κατανοήσουν τι είναι τα αντικείμενα Όψεις (views).
- εξοικειωθούν με την δημιουργία, τροποποίηση, διαγραφή των Όψεων (views)

12.1. Τι είναι οι Όψεις (Views)

Μια όψη (view) είναι ένα αντικείμενο βάσης δεδομένων που ορίζεται με μια δήλωση SELECT με την οποία ανακτώνται τα δεδομένα που θέλουμε και μόνο σύμφωνα με την εντολή SELECT. Οι Όψεις ονομάζονται και εικονικοί πίνακες. Μια όψη επιλέγει τα δεδομένα από πίνακες ή άλλες όψεις. Σε ορισμένες περιπτώσεις, μια όψη μπορεί να ανανεώνεται χρησιμοποιώντας δηλώσεις όπως UPDATE, DELETE, INSERT ώστε να τροποποιεί ακόμα και ένα πίνακα. Στην ουσία μία όψη δεν περιέχει από μόνη της δεδομένα αλλά είναι ένας καθρέφτης μέσω του οποίου είναι ορατά τα δεδομένα άλλων πινάκων ή/και όψεων όπου μπορούμε να τα δούμε ή να τα αλλάξουμε.

Τι προσφέρουν οι όψεις;

- Περιορίζουν την πρόσβαση στα δεδομένα πινάκων,
- Εμφανίζουν μόνο συγκεκριμένες στήλες πίνακα,
- Βοηθούν στο να εκτελέσουμε απλά ερωτήματα, ενώ συνήθως πίσω από μια όψη υπάρχουν πολύπλοκα ερωτήματα,
- Δίνουν την δυνατότητα πρόσβασης σε συγκεκριμένες ομάδες χρηστών σύμφωνα με κριτήρια που μπορούμε να θέσουμε,
- Μπορούν να χρησιμοποιηθούν για να εκτελέσουν υπολογισμούς και την εμφάνιση των αποτελεσμάτων. Για παράδειγμα μια όψη μπορεί να περιέχει τα συγκεντρωτικά αποτελέσματα τα οποία προκύπτουν από χρήση συναρτήσεων,
- Μπορούν να χρησιμοποιηθούν για την επιλογή ενός περιορισμένου συνόλου εγγραφών με τη βοήθεια ενός κατάλληλου WHERE, ή να επιλεγεί μόνο ένα υποσύνολο των στηλών ενός πίνακα. Αυτό μπορεί να φανεί χρήσιμο ακόμα και στο να τροποποιείται η δομή των πινάκων της βάσης,
- Μπορούν να χρησιμοποιηθούν για την επιλογή δεδομένων από πολλούς πίνακες χρησιμοποιώντας μια σύνδεση (join) ή ένωση (union). Αυτή η διαδικασία

γίνεται αυτόματα από τη στιγμή που θα ορίσουμε την όψη και δεν χρειάζεται να εκτελείται κάθε φορά κάποιος πολύπλοκος συνδυασμός εντολών.

12.2. Κατηγορίες Όψεων

Οι όψεις μπορούν να διαχωριστούν στις εξής κατηγορίες:

Απλές όψεις

- Προέρχονται από ένα και μόνο πίνακα,
- Δεν περιέχουν συναρτήσεις και group από data (π.χ. Group by),
- Επιτρέπονται DML λειτουργίες μέσω της όψης (insert, update, delete) στον πίνακα που καθρεφτίζουν.

Σύνθετες όψεις:

- Προέρχονται από περισσότερους τους ενός πίνακες,
- Περιέχουν συναρτήσεις ή group από data (π.χ. Group by),
- Δεν επιτρέπονται DML λειτουργίες μέσω της όψης στους πίνακες που καθρεφτίζουν.

12.3. Δημιουργία Όψης

Για να δημιουργηθεί μια όψη χρησιμοποιείται η εντολή CREATE VIEW. Η πλήρης σύνταξη της είναι η παρακάτω.

```
CREATE [OR REPLACE] [ALGORITHM = algorithm_type]
VIEW view_name [(column_list)]
AS select_statement
[WITH [CASCADED | LOCAL] CHECK OPTION];
```

view_name, είναι το όνομα της όψης, οι όψεις και οι πίνακες έχουν ονόματα από το ίδιο σύνολο γι' αυτό τα ονόματα των όψεων δεν πρέπει να υπάρχουν ξανά στην βάση ούτε σαν όψη αλλά ούτε και σαν πίνακας.

select_statement είναι το σύνολο της εντολής select που συγκεντρώνει τα στοιχεία που θέλουμε. Εξ ορισμού τα ονόματα των στηλών στην όψη είναι ίδια με τα ονόματα των στηλών στον πίνακα ή των πινάκων που ανακτά η όψη. Με χρήση του χαρακτήρα μπαλαντέρ * ανακτώνται όλες οι στήλες.

Οι ελάχιστες παράμετροι που χρειάζεται μια εντολή δημιουργίας όψης είναι:

```
CREATE VIEW view_name AS select_statement;
```

Παραδειγμα

Για να δημιουργήσουμε μια όψη η οποία θα δείχνει τα στοιχεία του πίνακα departments εκτελούμε την εντολή:

```
CREATE OR REPLACE  
VIEW departments_v  
AS SELECT * FROM departments;
```

Δραστηριότητα

Να δημιουργήσετε το view departments_sal_vw που να φέρνει τον ελάχιστο και τον μέγιστο μισθό κάθε τμήματος της εταιρείας, σημειώνεται ότι στη λίστα πρέπει να αναφέρεται το όνομα του τμήματος, ο μέγιστος μισθός, και ο ελάχιστος μισθός.

```
CREATE VIEW departments_sal_vw  
(department_name, max_salary, min_salary)  
AS SELECT d.department_name, max(e.salary), min(e.salary)  
FROM employees e INNER JOIN departments d ON d.id=e.department_id  
GROUP BY e.department_id;
```

Συμβουλή

Είναι καλό να δίνονται νέα ονόματα στις στήλες της όψης ώστε να διακρίνονται καλύτερα.

Δραστηριότητα

Δημιουργήστε όψη που να αναπαριστά τους υπαλλήλους του Τμήματος Προσωπικού. Το όνομα της όψης να είναι pers_employees_vw. Η όψη να φέρνει το id, όνομα, επώνυμο, βαθμό και τον ετήσιο μισθό του κάθε υπαλλήλου.

```
CREATE VIEW pers_employees_vw
AS SELECT id, firstname, lastname, grade, 12 * salary
FROM employees
WHERE department_id = 2;
```

Συμβούλη

Εάν στον ορισμό μιας όψης περιλαμβάνεται μια παράμετρος *WHERE* και μια εντολή που αναφέρεται στην όψη περιλαμβάνει επίσης μια παράμετρο *WHERE*, τότε το αποτέλεσμα χρησιμοποιεί τις συνθήκες και των δύο *where* παραμέτρων.

Εάν στον ορισμό μιας όψης περιλαμβάνεται μια παράμετρος *order by* και μια εντολή που αναφέρεται στην όψη περιλαμβάνει επίσης μια παράμετρο *order by clause*, τότε το αποτέλεσμα ταξινομείται με βάση την *order by* της εντολής και όχι του ορισμού.

Δραστηριότητα

Δημιουργήστε όψη που να αναπαριστά τους υπαλλήλους του Τμήματος Πληροφορικής.

```
CREATE OR REPLACE VIEW it_employees_vw (name, sal, department_id)
AS SELECT concat(firstname, ' ', lastname), salary, department_id
FROM employees WHERE department_id = 1;
```

Δραστηριότητα

Να τροποποιήσετε την όψη *it_employees_vw* και να φέρνει το *id*, το ονοματεπώνυμο του υπαλλήλου, το μισθό του και το *id* του τμήματος Πληροφορικής.

```
CREATE OR REPLACE VIEW it_employees_vw (id, name, sal, department_id)
AS SELECT id, concat(firstname, ' ', lastname), salary, department_id
FROM employees WHERE department_id = 1;
```

Για να αντικατασταθούν τα εξ ορισμού εμφανιζόμενα ονόματα στηλών με όποια ονόματα χρειάζεται αρκεί να χρησιμοποιηθεί η παράμετρος *column_list*. Ένας άλλος τρόπος για να μετονομαστούν οι στήλες μιας όψης είναι να χρησιμοποιηθούν ονόματα

αντιστοίχου (aliases) στην εντολή select. Αυτό είναι χρήσιμο μόνο αν μετονομάζεται μέρος των στηλών.

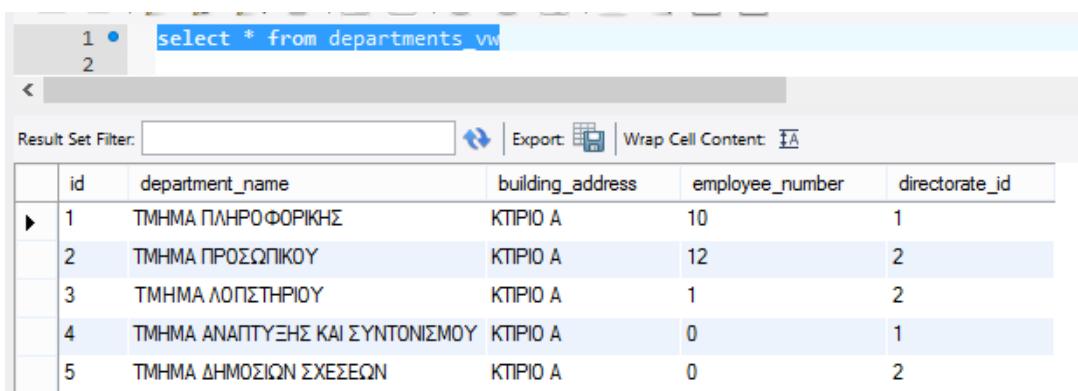
12.4. Ανάκτηση δεδομένων από Όψη

Για να ανακτήσουμε δεδομένα από μια όψη αρκεί να εκτελέσουμε μια απλή εντολή SELECT. Στην Εικόνα 82 Αποτέλεσμα αναζήτησης στο view φαίνεται το αποτέλεσμα της εκτέλεσης της εντολής.

```
SELECT * FROM departments_sal_vw;
```

Επειδή η όψη departments_sal_vw εμφανίζει τα στοιχεία του πίνακα departments είναι σαν να έχουμε εκτελέσει την εντολή

```
SELECT * FROM departments;
```

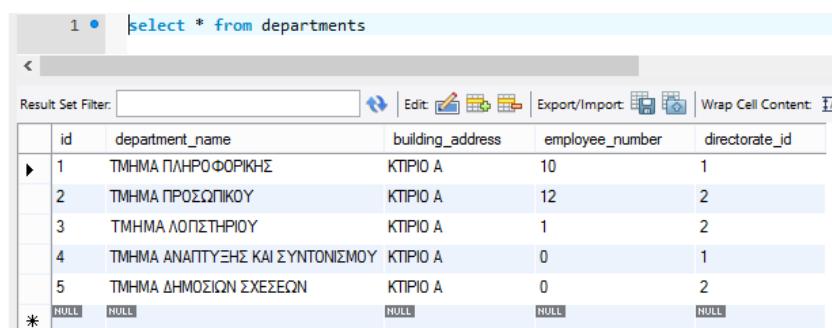


The screenshot shows the Oracle SQL Developer interface with the following details:

- SQL Editor pane: Shows the query `select * from departments vw`.
- Output pane: Displays the results of the query, which is identical to the output shown in Eikona 83 below.

	id	department_name	building_address	employee_number	directorate_id
▶	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	ΚΤΠΡΙΟ Α	10	1
	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	ΚΤΠΡΙΟ Α	12	2
	3	ΤΜΗΜΑ ΛΟΠΣΤΗΡΙΟΥ	ΚΤΠΡΙΟ Α	1	2
	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ	ΚΤΠΡΙΟ Α	0	1
	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΕΩΝ	ΚΤΠΡΙΟ Α	0	2

Εικόνα 82 Αποτέλεσμα αναζήτησης στο view



The screenshot shows the Oracle SQL Developer interface with the following details:

- SQL Editor pane: Shows the query `select * from departments`.
- Output pane: Displays the results of the query, which is identical to the output shown in Eikona 82 above.

	id	department_name	building_address	employee_number	directorate_id
▶	1	ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ	ΚΤΠΡΙΟ Α	10	1
	2	ΤΜΗΜΑ ΠΡΟΣΩΠΙΚΟΥ	ΚΤΠΡΙΟ Α	12	2
	3	ΤΜΗΜΑ ΛΟΠΣΤΗΡΙΟΥ	ΚΤΠΡΙΟ Α	1	2
	4	ΤΜΗΜΑ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΥΝΤΟΝΙΣΜΟΥ	ΚΤΠΡΙΟ Α	0	1
	5	ΤΜΗΜΑ ΔΗΜΟΣΙΩΝ ΣΧΕΣΕΩΝ	ΚΤΠΡΙΟ Α	0	2
*	NULL	NULL	NULL	NULL	NULL

Εικόνα 83

12.5. Περιορισμοί όψεων

Ορίζονταις μια όψη μπορούν να συμπεριληφθούν οι περισσότερες από τις παραμέτρους

που επιτρέπονται στις εντολές, όπως WHERE, GROUP BY, και ούτω καθεξής. Ωστόσο, οι όψεις στη MySQL έχουν κάποιους περιορισμούς που δεν ισχύουν για τους πίνακες:

- ✓ Δεν μπορείτε να δημιουργήσετε μια προσωρινή όψη,
- ✓ Δεν μπορείτε να συσχετίσετε ένα έναυσμα (trigger) με όψη,
- ✓ Πρέπει να υπάρχουν ήδη οι πίνακες στους οποίους βασίζεται μια όψη,
- ✓ Η πρόταση SELECT στον ορισμό της όψης δεν μπορεί να περιέχει:
 - ✓ Δευτερεύοντα ερωτήματα στον όρο FROM,
 - ✓ Αναφορές σε προσωρινούς πίνακες,
 - ✓ Αναφορές σε μεταβλητές χρήστη,
 - ✓ Αναφορές σε παραμέτρους διαδικασίας, εάν ο ορισμός της όψης είναι μέσα σε μια αποθηκευμένη ρουτίνα,
 - ✓ Αναφορές σε παραμέτρους προετοιμασμένων εντολών.

12.6. Αλγόριθμοι όψεων

Αναφέρθηκε νωρίτερα ότι στην εντολή CREATE VIEW της MySQL υπάρχει η παράμετρος ALGORITHM, η οποία ορίζει τον αλγόριθμο που επεξεργάζεται την όψη. Όταν χρησιμοποιείται έχει την σύνταξη:

ALGORITHM = {UNDEFINED | MERGE | TEMPORARY}

UNDEFINED, είναι ο εξ ορισμού χρησιμοποιούμενος, η mysql αποφασίζει από μόνη της πως θα εκτελέσει την εντολή,

MERGE, εκτελεί μια συνένωση της εντολής που αναφέρεται στην όψη και στις εντολές δημιουργίας της όψης,

TEMPORARY, χρησιμοποιεί έναν ενδιάμεσο πίνακα για να αποθηκεύσει τα αποτελέσματα της εντολής δημιουργίας και μετά χρησιμοποιεί αυτόν τον πίνακα για να εκτελέσει την εντολή που αναφέρεται στην όψη. Με αυτό τον τρόπο δεν μπορούν να δημιουργηθούν ανανεούμενες όψεις.

12.7. Δημιουργία ανανεούμενης όψης (updatable view)

Οι όψεις είναι ανανεούμενες όταν μπορούν να χρησιμοποιηθούν με εντολές όπως η UPDATE και η DELETE για να τροποποιήσουν τους πίνακες που σχετίζονται. Δεν είναι όλες οι όψεις ανανεούμενες. Για παράδειγμα μπορεί ένας πίνακας να ανανεωθεί

αλλά μία όψη αυτού η οποία περιέχει στοιχεία από υπολογισμό εγγραφών του να μην μπορεί. Ο λόγος είναι ότι κάθε εγγραφή της όψης δεν αντιστοιχεί μοναδιαία και σε εγγραφή πίνακα και έτσι μια όψη δεν ξέρει τι πρέπει να ανανεώσει. Έτσι η βασική αρχή για να μπορεί να ανανεωθεί μια όψη είναι να έχει μια προς μια συσχέτιση των εγγραφών της με τις εγγραφές του πίνακα και οι στήλες της όψης να είναι απλές αναφορές στον πίνακα και όχι εκφράσεις. Στην περίπτωση που οι στήλες της όψης ονομάζονται ακριβώς με τον ίδιο τρόπο όπως ο πίνακας τότε μπορούμε ακόμα και να προσθέσουμε εγγραφές.

Παράδειγμα

```
UPDATE it_employees_vw SET sal ='5000' WHERE id_number = 1  
SELECT * FROM employees WHERE id=1;
```

Ο υπάλληλος με id=1 έχει πλέον sal='5000'

Με χρήση της παραμέτρου CHECK OPTION

- **LOCAL:** όταν εκτελούμε update στην όψη, η MySQL ελέγχει αν οι ενημερώσεις που πάμε να πραγματοποιήσουμε ακολουθούν τον ορισμό της όψης.
- **CASCDED:** όταν εκτελούμε update στην όψη, η MySQL ελέγχει την τρέχουσα όψη και αν η τρέχουσα όψη βασίζεται σε άλλη όψη ελέγχει και την επόμενη όψη κοκ, ότι οι ενημερώσεις που πάμε να πραγματοποιήσουμε τηρούν τον ορισμό της όψης.

Παράδειγμα

Διαφορά LOCAL – CASCDED

```
CREATE TABLE t1 (a INT);  
CREATE VIEW v1 AS SELECT * FROM t1 WHERE a < 2;  
  
CREATE VIEW v2 AS SELECT * FROM v1 WHERE a > 0  
WITH LOCAL CHECK OPTION;
```

```

CREATE VIEW v3 AS SELECT * FROM v1 WHERE a > 0
    WITH CASCADED CHECK OPTION;
INSERT INTO v2 VALUES (2);
Query OK, 1 row affected (0.00 sec)
INSERT INTO v3 VALUES (2);
ERROR 1369 (HY000): CHECK OPTION failed 'test.v3';

```

Μια όψη δεν είναι ανανεούμενη αν περιέχει:

- ✓ Aggregate functions (SUM(), MIN(), MAX(), COUNT(), κ.α.), <http://dev.mysql.com/doc/refman/5.0/en/group-by-functions.html>
- ✓ DISTINCT,
- ✓ GROUP BY,
- ✓ HAVING,
- ✓ UNION or UNION ALL,
- ✓ Subquery στη select,
- ✓ Joins,
- ✓ Nonupdatable view στο FROM
- ✓ Στήλες με expressions (όπως όψη με το ετήσιο μισθό 12 * salary)

Μια όψη είναι insertable όταν είναι updatable και:

- ✓ Δεν πρέπει να υπάρχουν διπλότυπα ονόματα στηλών στην όψη,
- ✓ Πρέπει να περιέχει όλες τις στήλες του πίνακα που βασίζεται και που δεν έχουν default τιμή,
- ✓ Πρέπει να περιέχει όλες τις NOT NULL στήλες του πίνακα που βασίζεται,
- ✓ Κάθε στήλη πρέπει να είναι μια απλή αναφορά σε στήλη πίνακα και όχι παραγόμενη στήλη, στήλη δηλαδή με expressions, όπως:
 - ✓ UPPER(column)
 - ✓ Subquery

12.8. Τροποποίηση όψης

Η διαφορά με τον ορισμό της όψης είναι ότι η σύνταξη της εντολής είναι με ALTER VIEW και δεν μπορεί να χρησιμοποιηθεί το τμήμα της εντολής OR REPLACE. Για να εκτελεστεί πρέπει να υπάρχει η όψη.

Παράδειγμα

```
ALTER VIEW it_employees_vw (id, name, sal, department_id)
AS SELECT id, concat(firstname, ' ', lastname), salary, department_id
FROM employees WHERE department_id = 2;
```

Πλέον η όψη it_employees_vw έχει τους ανθρώπους του τμήματος προσωπικού μόνο.

12.9. Διαγραφή Όψης

Η διαγραφή μιας όψης γίνεται με την εντολή drop. Η πλήρης σύνταξη της εντολής είναι:

```
DROP VIEW [IF EXISTS] view_name [, view_name] ... ;
```

Παράδειγμα

```
DROP VIEW IF EXISTS v1, v2;
```

Όταν διαγράφουμε μια όψη δεν χάνουμε τα δεδομένα αφού η όψη δεν περιέχει δεδομένα αλλά καθρεφτίζει πίνακα της βάσης δεδομένων που περιέχει δεδομένα.

Δραστηριότητα

Διαγράψτε την όψη departments_vw

12.10. Έλεγχος όψης

Μια όψη εξαρτάται από έναν άλλο πίνακα η όψη. Έτσι σε μια πιθανή τροποποίηση αυτών (μετονομασία, διαγραφή γραμμών κα.) επηρεάζει και την όψη χωρίς προειδοποίηση. Για να ελεγχθεί μια όψη για προβλήματα εκτελούμε την εντολή.

```
CHECK TABLE view_name;
```

Η οποία θα δώσει κάποιες πληροφορίες αν εμφανίζεται πρόβλημα.

Παράδειγμα

```
CHECK TABLE it_employees_vw;
```

The screenshot shows a database management interface with a results grid. The title bar says "check table it_employees_vw". The results grid has columns: Table, Op, Msg_type, and Msg_text. One row is shown: "inepdb.it_employees_vw check status OK".

	Table	Op	Msg_type	Msg_text
▶	inepdb.it_employees_vw	check	status	OK

Εικόνα 84

12.11. Μεταδεδομένα όψεων

Το σχήμα INFORMATION_SCHEMA έχει όψη του πίνακα, η οποία περιέχει όλα τα μεταδεδομένα των όψεων.

Παράδειγμα

```
SELECT * FROM INFORMATION_SCHEMA.VIEWS  
WHERE TABLE_NAME = 'it_employees_vw'  
AND TABLE_SCHEMA = 'inepdb';
```

The screenshot shows a database management interface with a results grid. The title bar says "SELECT * FROM INFORMATION_SCHEMA.VIEWS". The results grid has columns: TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and VIEW_DEFINITION. One row is shown: "inepdb it_employees_vw select inedb.employees AS E1 left join inedb.employees AS E2 on E1.employee_id = E2.employee_id select inedb.employees AS E1 left join inedb.departments AS D1 on E1.department_id = D1.department_id".

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	VIEW_DEFINITION
inefdb	inepdb	it_employees_vw	select inedb.employees AS E1 left join inedb.employees AS E2 on E1.employee_id = E2.employee_id select inedb.employees AS E1 left join inedb.departments AS D1 on E1.department_id = D1.department_id

Εικόνα 85

Δραστηριότητα

Πως θα βρείτε τα μεταδεδομένα της όψης departments_sal_vw και τη δήλωση δημιουργίας της;

12.12. Δικαιώματα για δημιουργία όψεων

Για να δημιουργήσουμε μια όψη θα πρέπει ο χρήστης να έχει δικαιώματα (privileges):

- ✓ **CREATE VIEW** δικαιώματα,
- ✓ **SELECT** δικαιώματα για όλες τις στήλες και πίνακες που υπάρχουν στο select
- ✓ Αν χρησιμοποιήσουμε στο CREATE VIEW το OR REPLACE πρέπει να έχουμε και **DROP** δικαιώματα για την όψη,
- ✓ Αν έχουμε επιπλέον δικαιώματα στους πίνακες της όψης όπως INSERT, UPDATE κ.α. αυτόματα έχουμε τα ίδια αυτά δικαιώματα και στην όψη.

13. Διαχείριση Βάσης Δεδομένων Mysql και Ασφάλεια (Data Control Language - DCL)

Μελετώντας το παρόν κεφάλαιο οι επιμορφωνόμενοι θα είναι σε θέση να:

- διαχειρίζονται μία βάση δεδομένων
- δημιουργούν χρήστες, να αλλάζουν κωδικούς πρόσβασης
- παρέχουν ή να ανακαλούν δικαιώματα λογαριασμών (Grant/Revoke privileges)

Η ακόλουθη ενότητα περιλαμβάνει θέματα που αφορούν θεμελιώδη στοιχεία ενός MySql Εξυπηρετητή (Server) όπως:

1. Τη δημιουργία και διαχείριση χρηστών και ασφάλειας,
2. Τη συντήρηση και υποστήριξη πινάκων,
3. Τη δημιουργία αντιγράφων ασφαλείας της Βάσης Δεδομένων (Backup Database) και την αποκατάσταση αυτών,
4. Την εξαγωγή και εισαγωγή μιας MySQL βάσης δεδομένων,
5. Την ανάκτηση χρήσιμων διαχειριστικών πληροφορίων που παρέχει ο MySQL Εξυπηρετητής

Όλα αυτά στον MySql Εξυπηρετητή επιτυγχάνονται με χρήση εντολών που ανήκουν στην κατηγορία των Data Control Language αλλά και μέσω προγραμμάτων διαχείρισης της βάσης.

13.1. Διαχείριση Λογαριασμών Χρηστών

Η παρούσα ενότητα περιγράφει την διαχείριση λογαριασμών για σύνδεση σε βάση δεδομένων MySQL και για πρόσβαση στα περιεχόμενα της. Καλύπτει τα ακόλουθα θέματα:

- ✓ Οι πίνακες εκχώρησης δικαιωμάτων που αποθηκεύουν τις πληροφορίες των λογαριασμών,
- ✓ Οι δηλώσεις SQL που χρησιμοποιούνται για τη διαχείριση των λογαριασμών,
- ✓ Τη χρήση των περιεχομένων του πίνακα εκχώρησης δικαιωμάτων για τον έλεγχο της πρόσβασης.

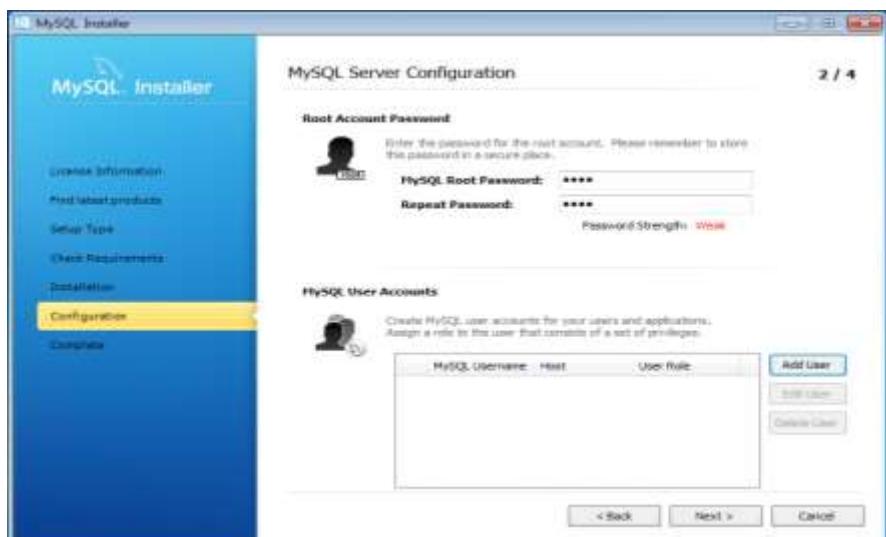
13.2. Κατά τη διάρκεια της εγκατάστασης

Κατά την διάρκεια της εγκατάστασης του MySQL εξυπηρετητή και πιο συγκεκριμένα στο δεύτερο βήμα δημιουργείται ο χρήστης root εξ ορισμού και ζητείται να επιλεγεί ένας κωδικός πρόσβασης (password) για αυτόν με περιορισμό να μην έχει μήκος μικρότερο των τεσσάρων (4) χαρακτήρων.



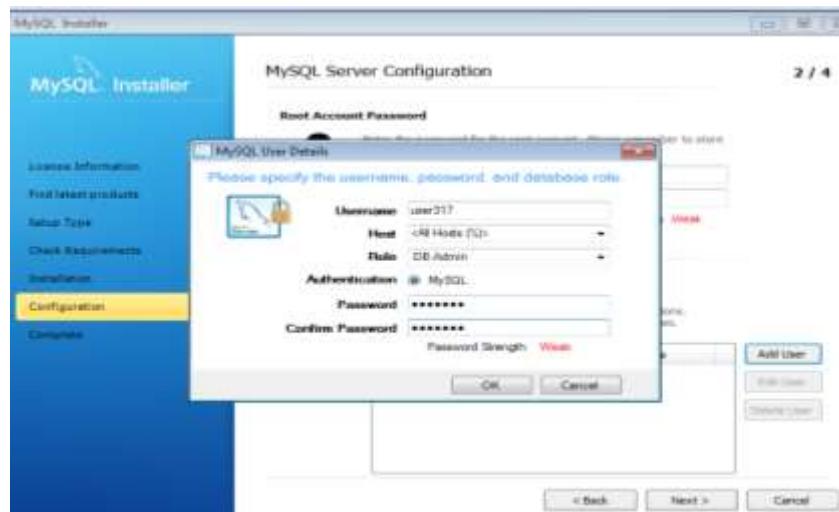
Εικόνα 86 Δημιουργία χρήστη root

Ανάλογα με το μήκος του κωδικού πρόσβασης θα μας ενημερώσει αν ο κωδικός είναι αδύναμος (weak με κόκκινο - Εικόνα 87 Αδύναμος κωδικός πρόσβασης)



Εικόνα 87 Αδύναμος κωδικός πρόσβασης

Επίσης κατά την διάρκεια της εγκατάστασης έχουμε την δυνατότητα να δημιουργήσουμε επιπλέον χρήστες με διάφορους ρόλους.



Εικόνα 88 Δημιουργία Χρήστη



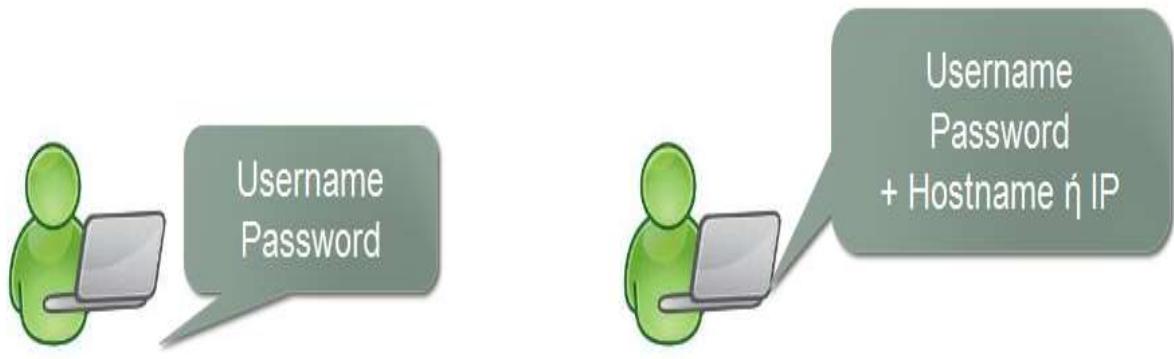
Εικόνα 89 Δημιουργία Χρήστη



Εικόνα 90 Πιθανοί ρόλοι χρήστη

13.3. Διαχείριση χρηστών, τι ισχύει;

Το σύστημα ελέγχου πρόσβασης της MySQL παρέχει τη δυνατότητα να δημιουργηθούν λογαριασμοί MySQL και να καθοριστεί τι μπορεί να κάνει κάθε λογαριασμός. Στην MySQL, η έννοια του λογαριασμού είναι συνδεδεμένη με τρία πράγματα: ένα όνομα χρήστη (username) ένα κωδικό πρόσβασης (password) και το όνομα του διακομιστή πελάτη (client host) από τον οποίον συνδέεται ο χρήστης. Δηλαδή, όταν ολοκληρώσετε τη σύνδεση με το διακομιστή της βάσης, ελέγχει όχι μόνο το username αλλά και τον client host από τον οποίο συνδέεται κάποιος. Σαν αποτέλεσμα αυτού είναι ότι μπορούμε να έχουμε δύο λογαριασμούς με το ίδιο username αλλά άλλο client host.



Εικόνα 91 Δημιουργία χρήστη - Διαφοροποίηση από άλλες Βάσεις

Στις SQL εντολές που απαιτούν ονόματα λογαριασμών, το όνομα δίνεται με την μορφή 'user_name'@'host_name'. Είναι επίσης δυνατό να καθοριστεί ένα πρότυπο για το δεύτερο τμήμα (client host), ώστε ο λογαριασμός να χρησιμοποιηθεί για σύνδεση με το διακομιστή MySQL από διάφορους πελάτες (client hosts). Για παράδειγμα, το όνομα ενός λογαριασμού που είναι 'xristis'@'%ekdd.gr' θα ισχύει για ένα χρήστη με όνομα xristis που συνδέεται από οποιοδήποτε υπολογιστή στον τομέα ekdd.gr.

Το τμήμα του χρήστη και του διακομιστή πελάτη απαιτείται να είναι σε quoted ξεχωριστά. Τα quotes είναι αναγκαία μόνο όταν περιέχονται ειδικοί χαρακτήρες όπως κάτω παύλες. Παρόλαυτά ακόμα και αν δεν υπάρχουν ειδικοί χαρακτήρες είναι αποδεκτά. Για να δημιουργηθεί ένας ανώνυμος χρήστης (είναι αυτός που ταιριάζει σε κάθε όνομα χρήστη) πρέπει να χρησιμοποιήσουμε το κενό για το μέρος του χρήστη *CREATE USER ''@'localhost'* (την εντολή *CREATE USER ''@'localhost'* θα την εξετάσουμε αργότερα). Φυσικά δεν είναι καλή πρακτική να δημιουργούμε ανώνυμους

χρήστες και ειδικά αν δεν έχουν κωδικό πρόσβασης γιατί ο οποιοσδήποτε θα έχει πρόσβαση στη βάση, κάτι το οποίο μπορεί να έχει τα όποια προβλήματα ασφαλείας.

Το τμήμα του διακομιστή πελάτη μπορεί να έχει τις εξής μορφές:

- Το όνομα localhost
- Ένα όνομα διακομιστή/πελάτη όπως myhost.ekdd.gr
- Ένα IP αριθμό π.χ. 192.168.1.20
- Μια μορφή που περιέχει '%' or '_'. Ο χαρακτήρας μπαλαντέρ είναι χρήσιμος όταν θέλουμε ο χρήστης να συνδέεται από κάθε διακομιστή/πελάτη ή δίκτυο.

Παράδειγμα

Ένα μέρος διακομιστή πελάτη με τιμή %.example.com ταιριάζει με κάθε διακομιστή στον τομέα example.com. Μια τιμή 192.168.% διακομιστή πελάτη ταιριάζει σε κάθε πελάτη στο υποδίκτυο 192.168. Οι τιμές 10.0.0.0/255.255.255.0 ταιριάζουν με κάθε διακομιστή πελάτη. Αυτή η μορφή είναι χρήσιμη όταν θέλουμε ο ένας χρήστης να μπορεί να συνδεθεί από κάθε διακομιστή πελάτη ή υποδίκτυο.

Το πλεονεκτήματα με αυτή την πολιτική διαχείρισης των χρηστών είναι ότι η MySQL περιορίζει επιπλέον την πρόσβαση των χρηστών στην βάση αφού οι χρήστες πρέπει να συνδέονται από συγκεκριμένο host ή hosts σε ένα domain. Ο χρήστης μπορεί να έχει διαφορετικά passwords και διαφορετικά δικαιώματα (privileges) ανάλογα με το host που συνδέεται. Ο χρήστης scott, που συνδέεται από abc.com μπορεί να είναι ή να μην είναι ίδιος με αυτόν που συνδέεται από το xyz.com

13.4. Δημιουργία Χρήστη

Η DCL εντολή δημιουργίας χρήστη είναι η CREATE USER η οποία δημιουργεί ένα νέο λογαριασμό και αποδίδει και κωδικό πρόσβασης αν απαιτείται. Αυτή η εντολή φτιάχνει μια εγγραφή στον πίνακα USER. Δεν δίνει κάποιο προνόμιο στο χρήστη, για να γίνει αυτό πρέπει να χρησιμοποιήσουμε την DCL εντολή GRANT που θα δούμε αργότερα.

Η πλήρης σύνταξη της εντολής είναι:

CREATE USER *user_specification*
[IDENTIFIED BY '*password*'];

Παράδειγμα

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

Δραστηριότητα: Δημιουργήστε τον χρήστη scott με κωδικό πρόσβασης scott χρησιμοποιώντας command line εντολή.

Απάντηση

CREATE USER 'scott'@'localhost' IDENTIFIED BY 'scott';

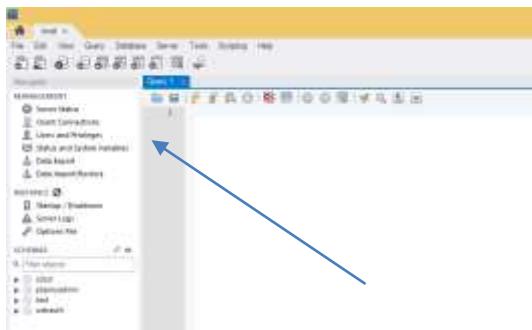
Για να επιτρέψουμε ο user να συνδέεται:

- Από οποιοδήποτε hostname γράφουμε: username@%
- Από localhost:
 - username@localhost
 - username@127.0.0.1
 - username@::1

To root@localhost δεν αρκεί γιατί υπάρχει λογαριασμός root@localhost και root@127.0.0.1 και root@::1. Επίσης αν ο Server τρέχει με το option –skip-name-resolve η σύνδεση root@localhost αποτυγχάνει. Τέλος ο localhost σε IPV6 διευθύνσεις είναι το ::1.

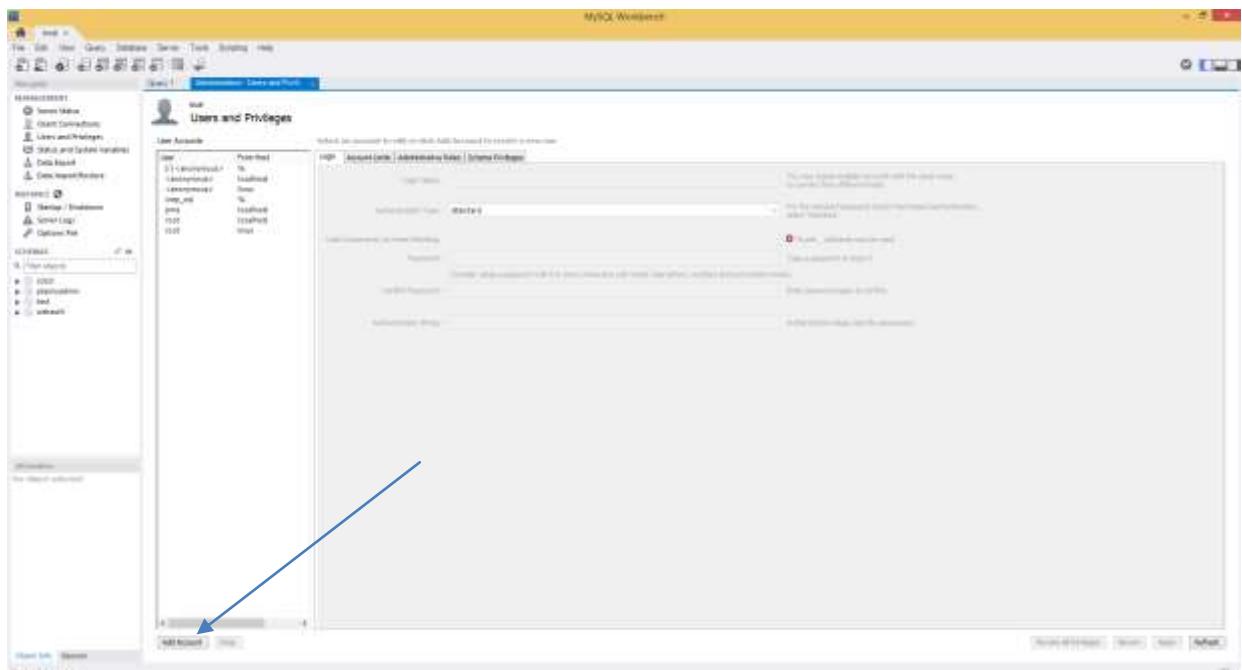
Με το Workbench η δημιουργία χρήστη γίνεται πιο απλά στα παρακάτω βήματα.

Από την αρχική οθόνη του MySQL WorkBench επιλέγουμε Servers, Users and Privileges και θα μεταβούμε στην οθόνη της Εικόνα 92 Users and Privileges.



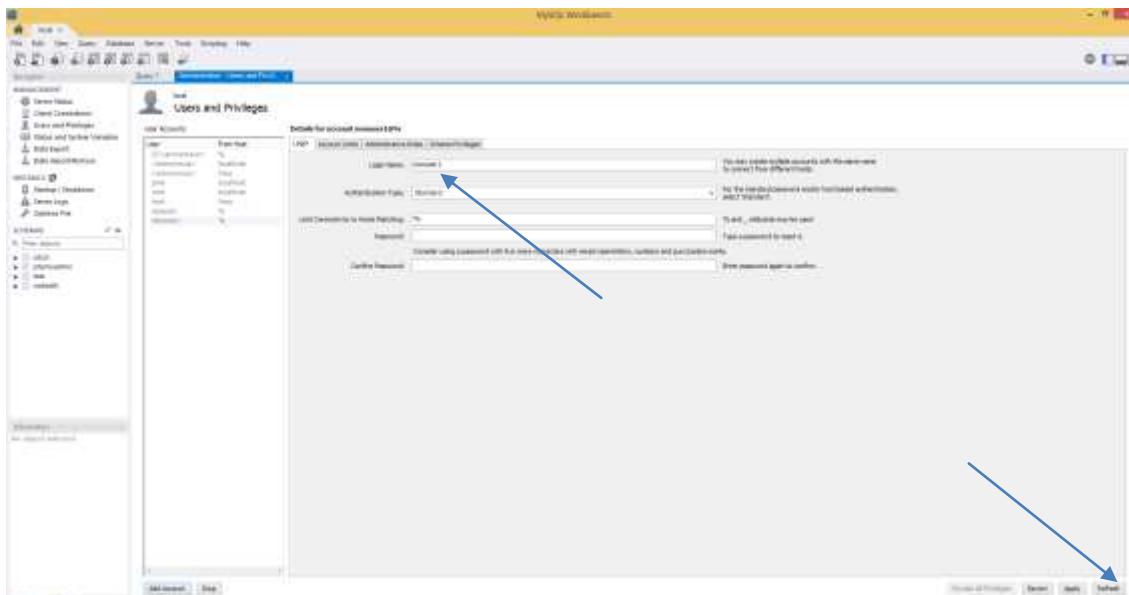
Εικόνα 92 Users and Privileges

Στη συνέχεια επιλέγουμε το κουμπί Add Account, κάτω αριστερά και μεταβαίνουμε στην οθόνη Εικόνα 93 Οθόνη Users and Privileges

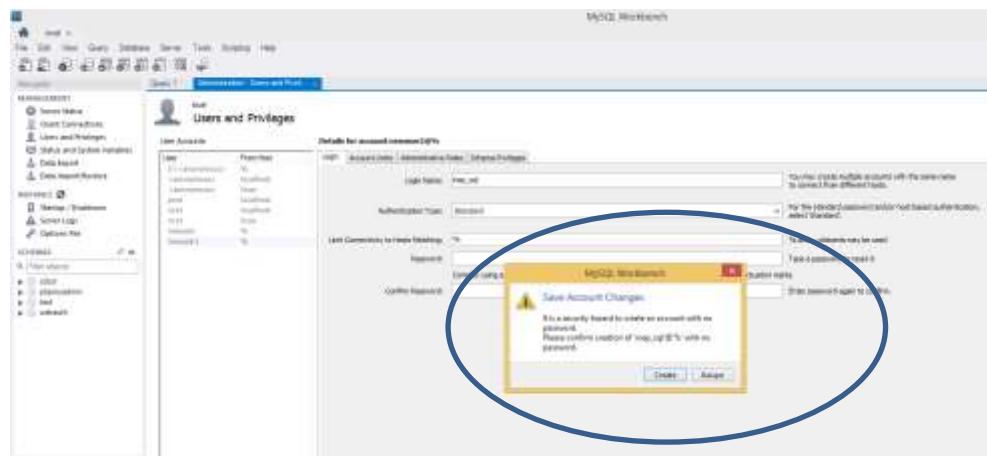


Εικόνα 93 Οθόνη Users and Privileges

Γράφουμε το όνομα του νέου χρήστη και επιλέγουμε κωδικό πρόσβασης. Αν δεν βάλουμε κωδικό πρόσβασης τότε θα εμφανιστεί το ενημερωτικό μήνυμα της Εικόνα 94 Δημιουργία νέου χρήστη, που μας λέει ότι είναι κίνδυνος ασφαλείας αν δεν θέσουμε κάποιο κωδικό πρόσβασης.

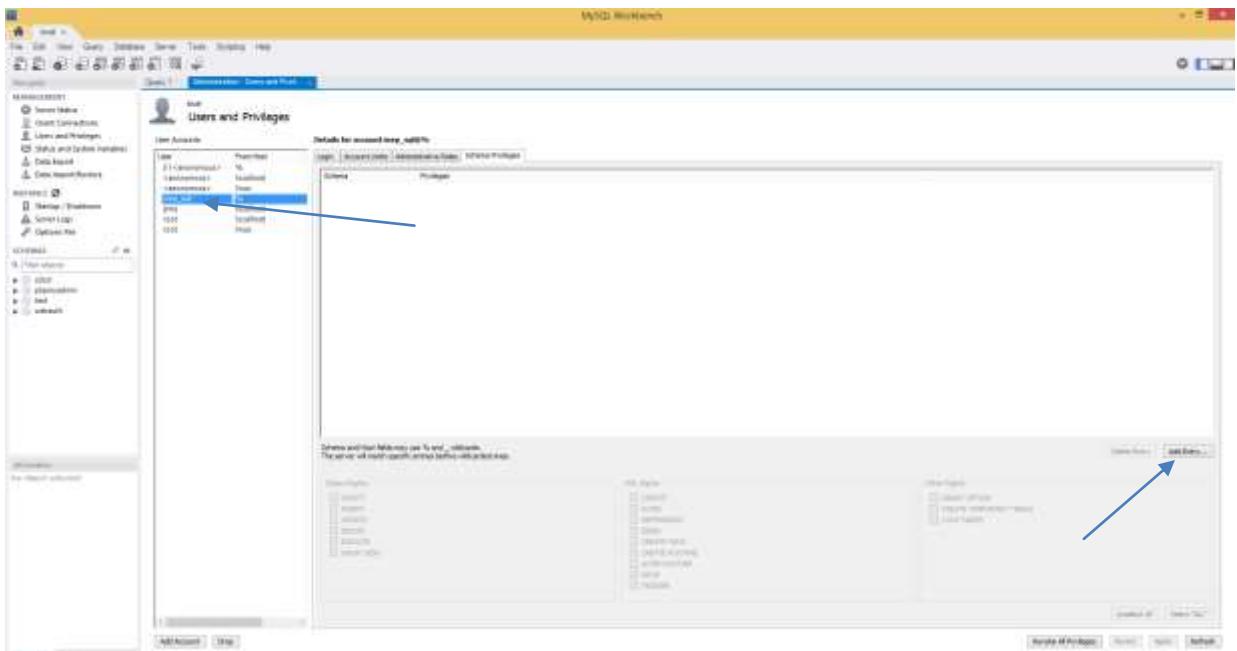


Εικόνα 94 Δημιουργία νέου χρήστη

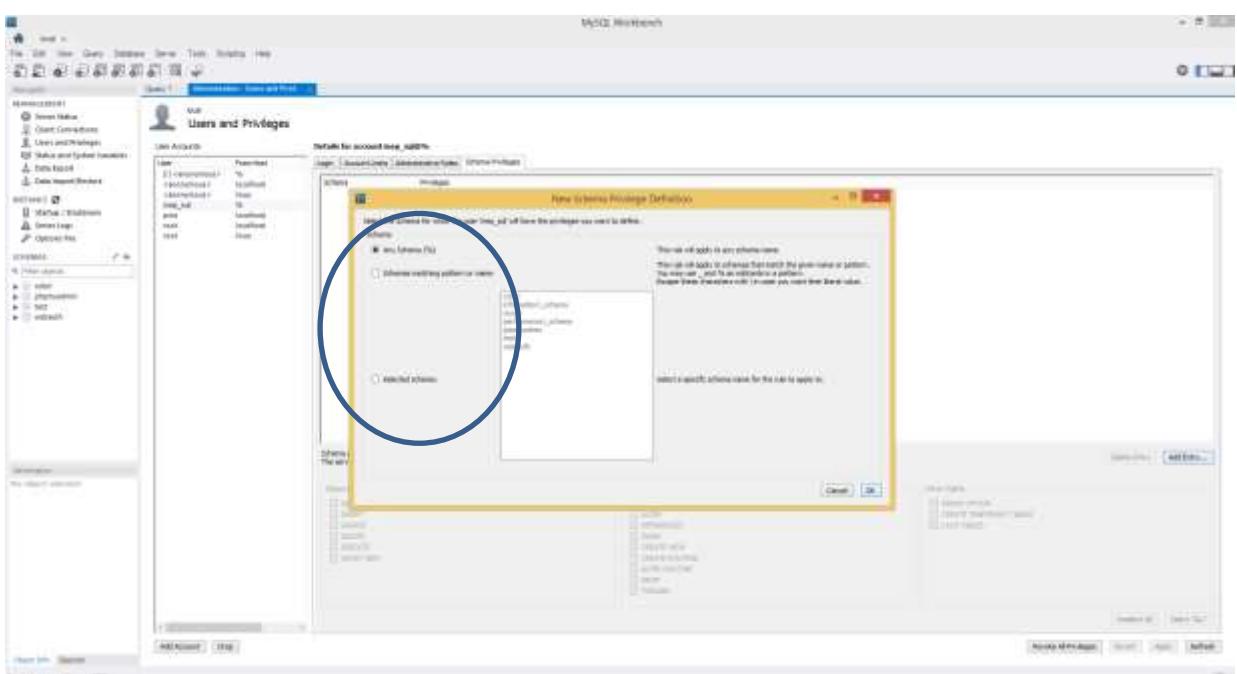


Εικόνα 95 Προειδοποίηση για password

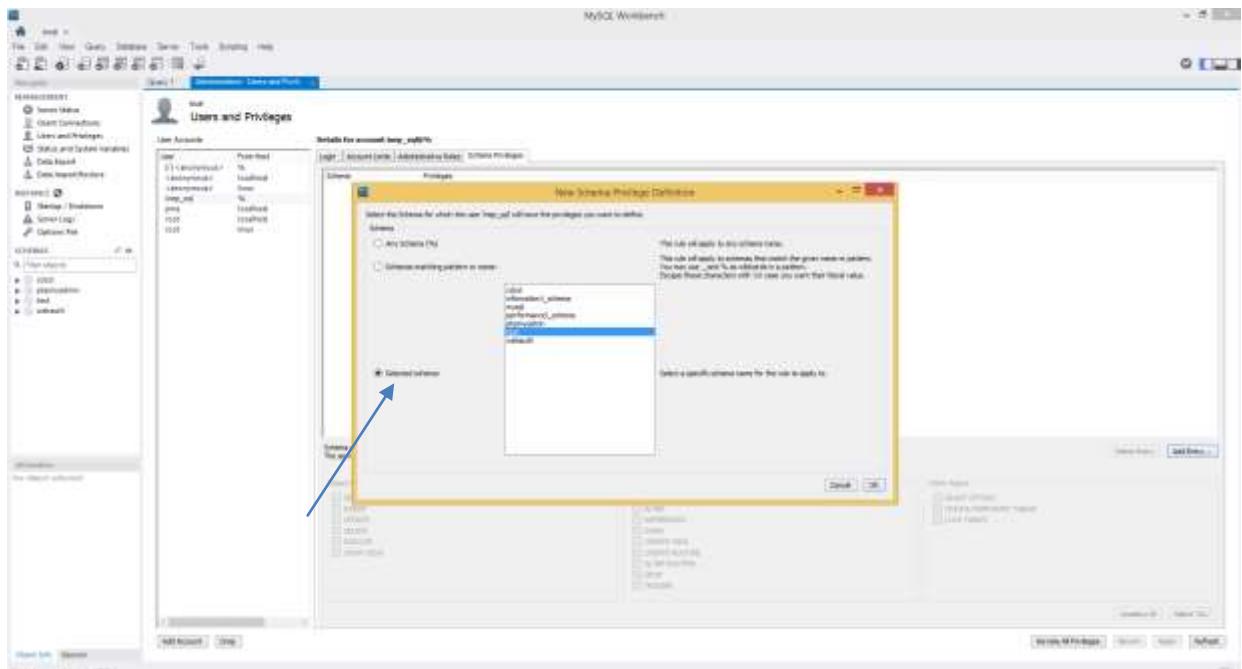
Στην συνέχεια βάζουμε τα δικαιώματα των χρηστών από την οθόνη όπως φαίνεται στην Εικόνα 95 Προειδοποίηση για password. Επιλέγουμε Add Entry... για να προσθέσουμε τα σχήματα στα οποία θα δώσουμε δικαιώματα και εμφανίζεται η οθόνη της Εικόνα 97 Δικαιώματα χρήστη σε όλα τα σχήματα. Επιλέγουμε το σχήμα, τα σχήματα ή και όλα τα σχήματα που θέλουμε να δώσουμε δικαιώματα στο χρήστη (Εικόνα 96 Δικαιώματα του χρήστη) και στη συνέχεια δίνουμε τα δικαιώματα (Εικόνα 98 Δικαιώματα χρήστη σε συγκεκριμένο σχήμα).



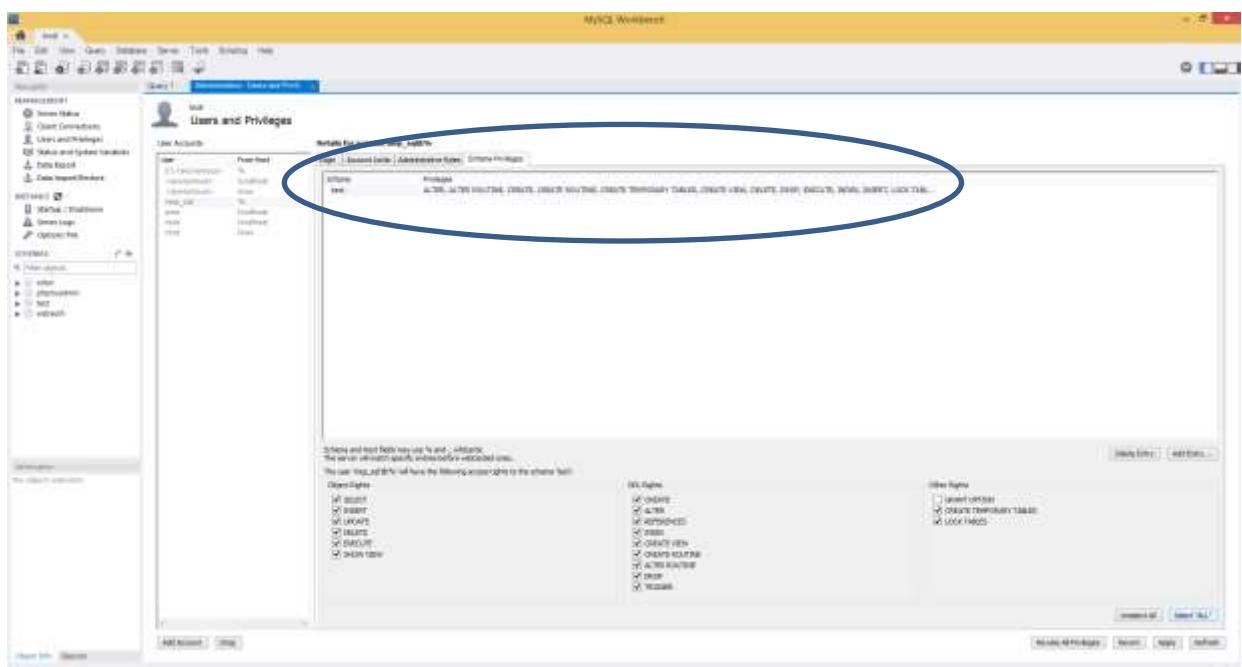
Εικόνα 96 Δικαιώματα του χρήστη



Εικόνα 97 Δικαιώματα χρήστη σε όλα τα σχήματα



Εικόνα 98 Δικαιώματα χρήστη σε συγκεκριμένο σχήμα



Εικόνα 99 Εμφάνιση δικαιωμάτων χρήστη

13.5. Μετονομασία χρήστη

Η εντολή RENAME USER μετονομάζει το όνομα χρήστη για έναν υπάρχοντα λογαριασμό. Μπορεί να τροποποιήσει το όνομα χρήστη (username) ή το hostname του λογαριασμού ή και τα δύο. Η σύνταξη της είναι:

RENAME USER old_user TO new_user;

Παράδειγμα

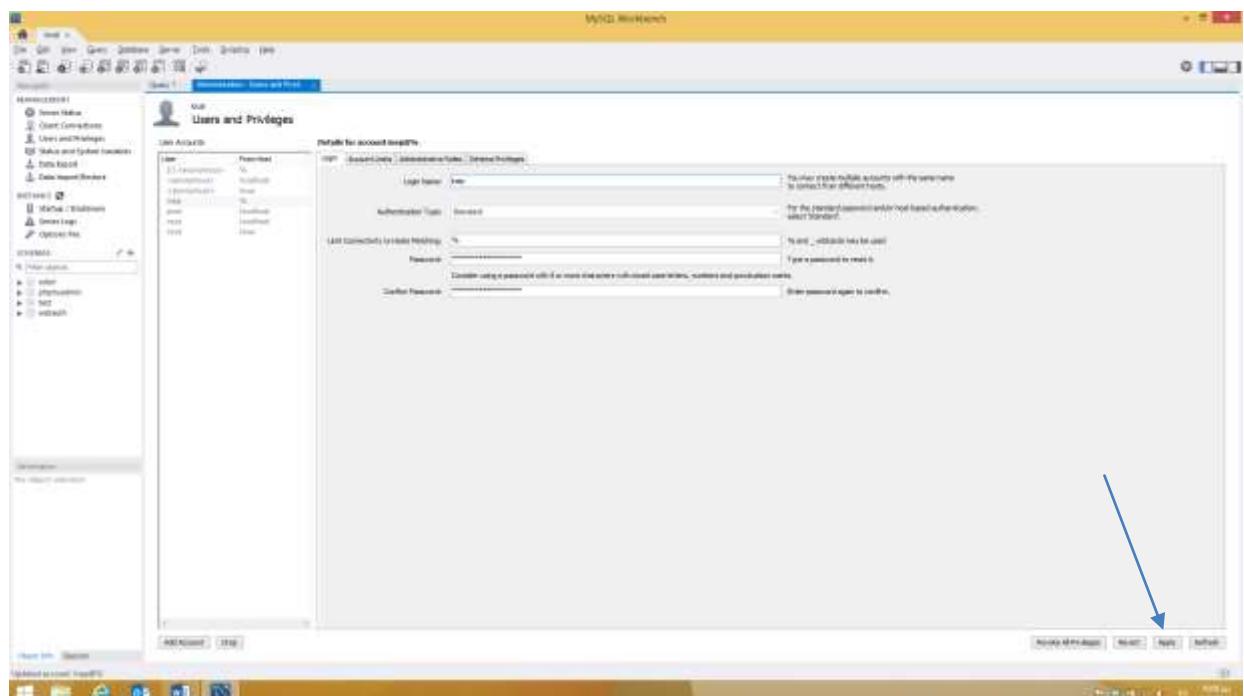
RENAME USER 'jim'@'localhost' TO 'john'@'localhost';

Δραστηριότητα: Μετονομάστε το χρήστη scott σε scotty χρησιμοποιώντας command line εντολή.

Απάντηση

RENAME USER 'scott'@'localhost' TO 'scotty'@'localhost';

Στο MySQL Workbench η αλλαγή ονόματος γίνεται απλά μέσα από την παρακάτω οθόνη και γράφοντας στο πεδίο Login Name το νέο όνομα και πατώντας Apply.



Εικόνα 100 Αλλαγή ονόματος χρήστη

13.6. Διαγραφή χρήστη

Η DROP USER ανακαλεί όλα τα προνόμια για έναν υπάρχοντα λογαριασμό και στη συνέχεια αφαιρεί το λογαριασμό. Καταργεί όλες τις εγγραφές για το λογαριασμό από οποιοδήποτε πίνακα προνομίων. Για να ανακληθούν μόνο τα προνόμια χωρίς να

αφαιρεθεί ο λογαριασμός υπάρχει η εντολή REVOKE που θα δούμε παρακάτω.

Η σύνταξη της είναι:

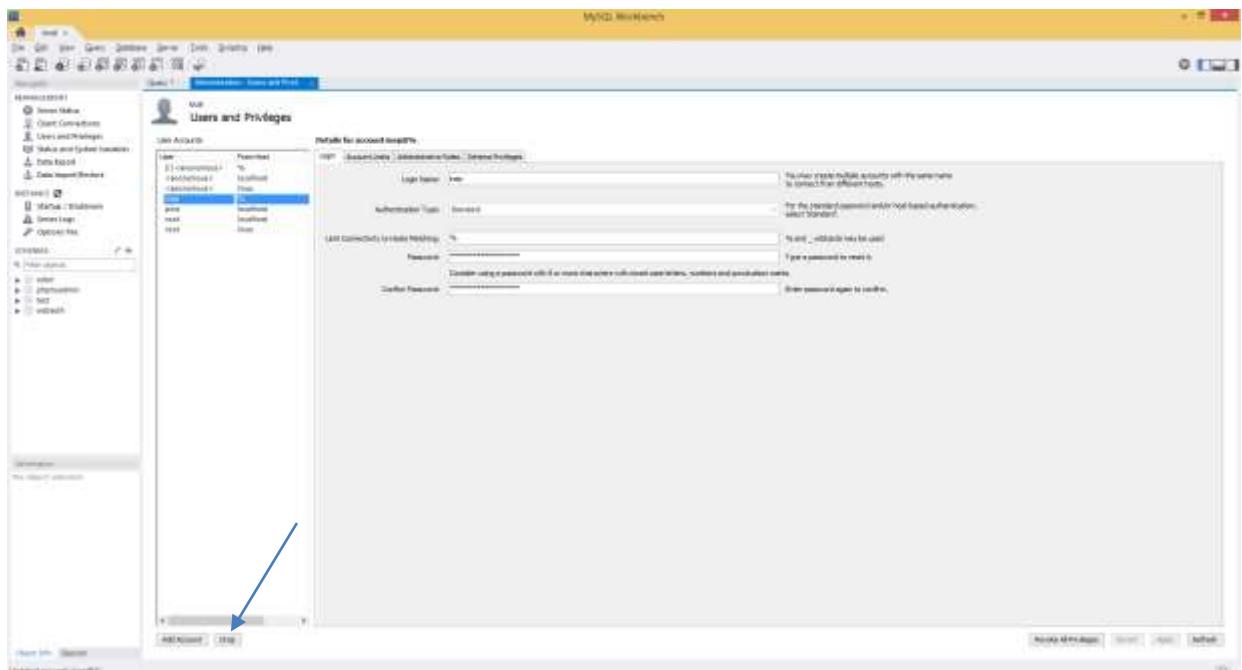
DROP USER user;

Παράδειγμα

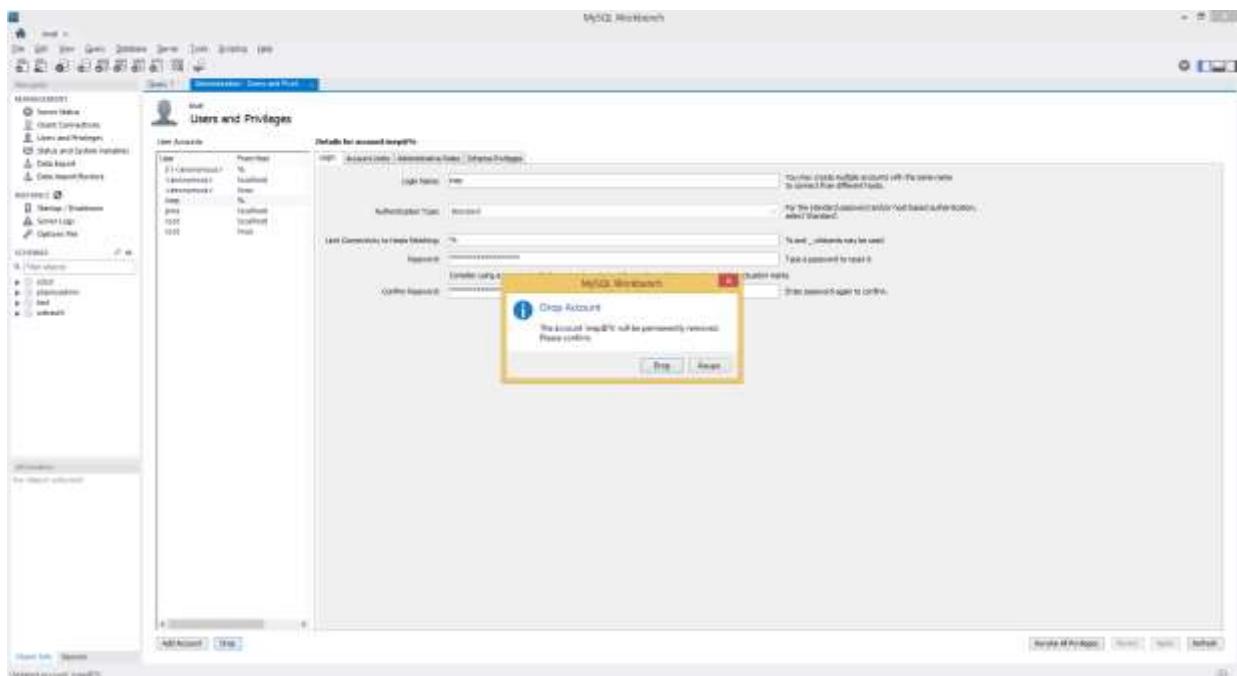
DROP USER 'scotty'@'localhost';

Δραστηριότητα: Διαγράψτε το χρήστη johh χρησιμοποιώντας command line εντολή.

Στο MySQL Workbench η διαγραφή χρήστη γίνεται απλά μέσα από την παρακάτω οθόνη επιλέγοντας το κουμπί Drop.



Εικόνα 101 Διαγραφή χρήστη



Εικόνα 102 Επιβεβαίωση διαγραφής χρήστη

13.7. Αλλαγή κωδικού πρόσβασης (password)

Όπως αναφέραμε παραπάνω μπορούμε να δώσουμε ένα κωδικό πρόσβασης για ένα λογαριασμό συμπεριλαμβάνοντας την επιλογή *IDENTIFIED BY* είτε στην εντολή *CREATE USER*, είτε στην *GRANT*. Με αυτό τον τρόπο με την *CREATE USER* τίθεται ένα αρχικός κωδικός πρόσβασης. Με την *GRANT* τίθεται αρχικός κωδικός πρόσβασης ή αλλάζει ο τρέχων ανάλογα με το αν ο λογαριασμός είναι νέος ή υπάρχει.

Για να τροποποιηθεί ο κωδικός πρόσβασης ενός υπάρχοντος λογαριασμού χωρίς να αλλαχθούν τα δικαιώματα υπάρχουν δύο επιλογές:

Χρήση της εντολής *SET PASSWORD* ορίζοντας το όνομα χρήστη και τον νέο κωδικό πρόσβασης. Η σύνταξη της εντολής είναι:

```
SET PASSWORD [FOR user] = PASSWORD('cleartext password');
```

Παράδειγμα

```
SET PASSWORD FOR 'scott'@'localhost' = PASSWORD('scott1');
```

Κάθε ανώνυμος χρήστης μπορεί να τροποποιήσει τον κωδικό πρόσβασης του κάνοντας χρήση της SET PASSWORD χωρίς FOR

Παράδειγμα

```
SET PASSWORD = PASSWORD('NewPass');
```

Η δεύτερη επιλογή είναι με χρήση της GRANT με δικαιώματα USAGE στο επίπεδο global και με χρήση του IDENTIFIED BY

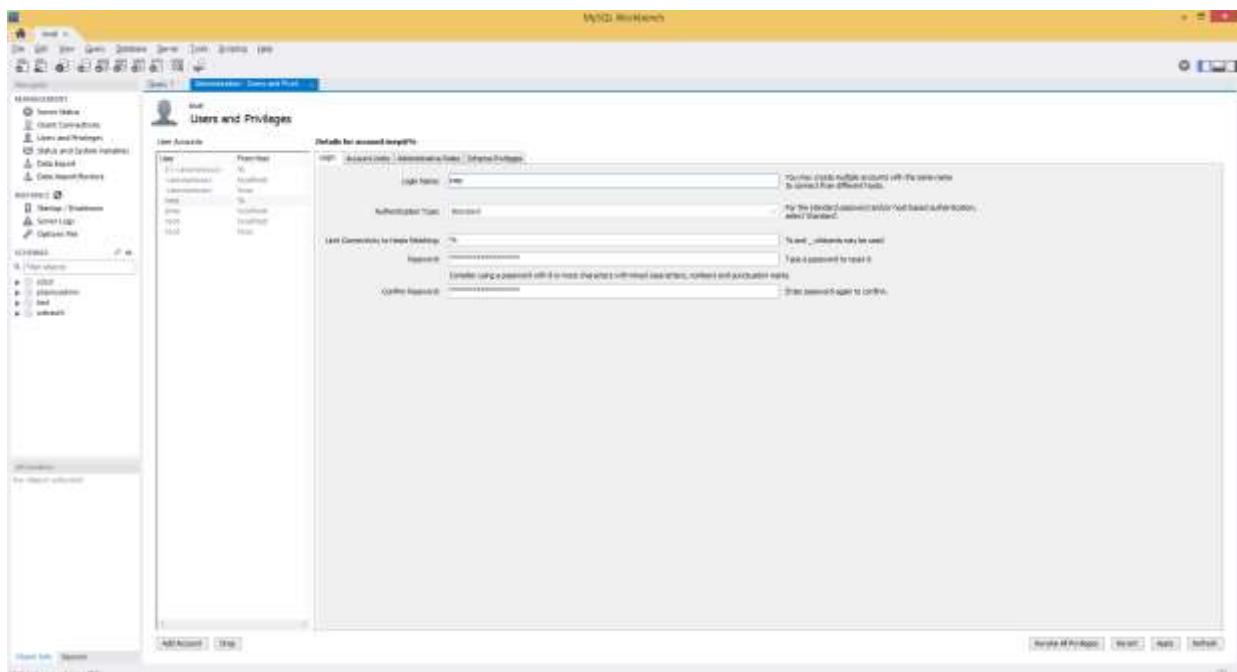
Παράδειγμα

```
GRANT USAGE ON *.* TO 'scott'@'localhost' IDENTIFIED BY 'NewPass';
```

Με τη χρήση του USAGE σημαίνει ότι δεν αλλάζουμε τα δικαιώματα, έτσι το μόνο που αλλάζει είναι ο κωδικός πρόσβασης.

Με την χρήση της SET PASSWORD η συνάρτηση PASSWORD() κωδικοποιεί τον κωδικό πρόσβασης ενώ η CREATE USER και GRANT δεν κωδικοποιεί τον κωδικό πρόσβασης. Μπορούν να οριστούν και λογαριασμοί χωρίς κωδικό πρόσβασης αλλά αυτό δεν είναι ασφαλές.

Στο MySQL Workbench η αλλαγή κωδικού πρόσβασης γίνεται απλά μέσα από την παρακάτω οθόνη αλλάζοντας τον κωδικό πρόσβασης στα ανάλογα πεδία και πατώντας Apply.



Εικόνα 103 Αλλαγή κωδικού πρόσβασης

13.8. Δικαιώματα (privileges) χρηστών

Διάφοροι τύποι προνομίων μπορούν να χορηγηθούν σε ένα λογαριασμό. Τα προνόμια παρέχονται ανάλογα με τον τρόπο που πρόκειται να χρησιμοποιηθεί ο λογαριασμός. Έτσι μπορεί να χρησιμοποιηθεί για να αποδοθούν δικαιώματα:

- Σε ένα λογαριασμό που χρειάζεται μόνο να διαβάζει η πρόσβαση που χρειάζεται στη βάση δεδομένων μπορεί να είναι μόνο με το προνόμιο SELECT.
- Σε ένα λογαριασμό που χρειάζεται να τροποποιεί τα δεδομένα μπορεί να δοθούν προνόμια DELETE, INSERT, UPDATE.
- Σε λογαριασμούς διαχειριστών μπορεί να δοθούν δικαιώματα PROCESS ή SUPER για να βλέπουν την δραστηριότητα των χρηστών.

Τα δικαιώματα που παρέχονται από την MySQL διακρίνονται σε:

- Administrative privileges δίνουν τη δυνατότητα στους χρήστες να εκτελούν λειτουργίες διαχείρισης του MySQL server. Αυτά τα δικαιώματα εφαρμόζονται global και δεν είναι συγκεκριμένα για μια βάση δεδομένων.
- Database privileges εφαρμόζονται σε μια βάση δεδομένων και στα αντικείμενα που περιέχει. Τα δικαιώματα αυτά αποδίδονται σε συγκεκριμένες βάσεις δεδομένων ή εφαρμόζονται σε όλες.

- Privileges σε database objects όπως tables, indexes, views, stored routines μπορούν να αποδοθούν σε συγκεκριμένα αντικείμενα της βάσης δεδομένων ή σε όλα τα αντικείμενα της βάσης δεδομένων ή σε περισσότερες βάσεις.

Η MySQL ελέγχει την πρόσβαση χρησιμοποιώντας τα περιεχόμενα των πινάκων χορήγησης δικαιωμάτων. Οι πίνακες αυτοί ορίζουν τους λογαριασμούς MySQL και τα προνόμια που κατέχουν. Για την διαχείριση του περιεχόμενου τους, χρησιμοποιούν τις δηλώσεις όπως CREATE USER, GRANT, και REVOKE. Οι δηλώσεις αυτές παρέχουν μια διεπαφή με τους πίνακες χορήγησης δικαιωμάτων μέσω δικαιωμάτων λογαριασμού διαχείρισης χωρίς να χρειάζεται να τροποποιηθούν άμεσα οι πίνακες.

13.9. Οι τύποι δικαιωμάτων που υποστηρίζει η MySQL

Σε ένα λογαριασμό MySQL μπορούν να εκχωρηθούν διάφοροι τύποι προνομίων και σε διαφορετικά επίπεδα (συνολικά ή μόνο για συγκεκριμένες βάσεις δεδομένων, πίνακες, ή στήλες). Για παράδειγμα, μπορεί να επιτραπεί σε ένα χρήστη να επιλέξει από οποιοδήποτε πίνακα, σε οποιαδήποτε βάση δεδομένων με τη χορήγηση προνομίου της SELECT σε όλο το φάσμα της βάσης. Τα προνόμια που υποστηρίζονται στη MySQL εμφανίζονται στους παρακάτω πίνακες.

Προνόμια Διαχειριστικά	Λειτουργία δικαιώματος
CREATE TEMPORARY TABLES	Χρήση TEMPORARY μεh CREATE TABLE
CREATE USER	Create, drop, rename κάποιο λογαριασμό
FILE	Διαβάζει ή γράφει αρχεία που βρίσκονται στον εξυπηρετητή
LOCK TABLES	Κλειδώνει τους πίνακες
PROCESS	Εμφάνιση του process
RELOAD	Χρήση με FLUSH και RESET
REPLICATION CLIENT	Ρωτάει τον εξυπηρετητή για πληροφορίες σχετικά με τους replication hosts
REPLICATION SLAVE	Λειτουργεί ως replication slave
SHOW DATABASES	Βλέπει όλες τις βάσεις δεδομένων
SHUTDOWN	Κάνει Shut down τον εξυπηρετητή

SUPER	Διάφορες διαχειριστικές δυνατότητες
-------	-------------------------------------

Δικαιώματα πρόσβασης στη βάση	Δικαιώματα διαχείρισης που επιτρέπονται με δικαιώματα
ALTER	Modify tables with ALTER TABLE
ALTER ROUTINE	Alter or drop stored routines
CREATE	Create databases and tables
CREATE ROUTINE	Create stored routines
CREATE VIEW	Create views
DELETE	Remove rows from tables
DROP	Drop databases and tables
EXECUTE	Execute stored routines
GRANT OPTION	Grant privileges to other accounts
INDEX	Create and drop indexes
INSERT	Add rows to tables
SELECT	Select records from tables
SHOW VIEW	Use SHOW CREATE VIEW
UPDATE	Modify records in tables

Υπάρχουν επιλογές και για ειδικά δικαιώματα όπως

- ALL και ALL PRIVILEGES τα οποία είναι συντομεύσεις των “all privileges except GRANT OPTION.”, το οποίο είναι πλήρη δικαιώματα εκτός του να δίνει δικαιώματα σε άλλους λογαριασμούς.
- USAGE, είναι το δικαίωμα του να έχει μόνο το δικαίωμα να συνδέεται στη βάση και κανένα άλλο δικαίωμα. Ο λογαριασμός μπορεί να έχει πρόσβαση στη βάση για περιορισμένους λόγους όπως SHOW VARIABLES ή SHOW STATUS αλλά όχι πρόσβαση σε περιεχόμενα.

Τα δικαιώματα μπορούν να εκχωρηθούν σε διαφορετικό επίπεδο:

- Κάθε προνόμιο μπορεί να αποδοθεί καθολικά. Συνήθως αποδίδεται σε λογαριασμούς διαχειριστών. Για παράδειγμα ένα προνόμιο τύπου DELETE καθολικής εφαρμογής δίνει δικαίωμα στο λογαριασμό να διαγράψει εγγραφές από έναν πίνακα σε κάθε βάση δεδομένων.

- Κάποια προνόμια μπορούν να αποδοθούν μόνο προς συγκεκριμένες βάσεις: ALTER, CREATE, CREATE TEMPORARY TABLES, CREATE VIEW, DELETE, DROP, GRANT OPTION, INDEX, INSERT, LOCK TABLES, SELECT, SHOW VIEW και UPDATE. Ένα προνόμιο σε επίπεδο βάσης αποδίδεται για όλα τα στοιχεία αυτής της βάσης
- Κάποια προνόμια μπορούν να αποδοθούν μόνο σε συγκεκριμένες στήλες πινάκων: INSERT, SELECT, and UPDATE.
- Κάποια προνόμια μπορούν να αποδοθούν μόνο σε συγκεκριμένες αποθηκευμένες ρουτίνες (stored routines): EXECUTE, ALTER ROUTINE, και GRANT OPTION.

13.10. Τροποποίηση δικαιωμάτων χρήστη

13.10.1. Grant Privileges

Για να χρησιμοποιήσουμε το GRANT ή το REVOKE πρέπει να έχουμε GRANT OPTION privilege.

Η σύνταξη για την απόδοση δικαιωμάτων με την Grant είναι:

```
GRANT priv_type [, priv_type] ...
ON priv_level
ON priv_level TO User_specification
[WITH with_option]
priv_level:
  *
  / *.* 
  / db_name.*
  / db_name.tbl_name
  / tbl_name
  / db_name.routine_name
user_specification: user [IDENTIFIED BY 'password']
with_option:
```

```
GRANT OPTION ( allows them to grant or remove other users' privileges);  
/MAX_QUERIES_PER_HOUR count  
/MAX_UPDATES_PER_HOUR count  
/MAX_CONNECTIONS_PER_HOUR count  
/MAX_USER_CONNECTIONS count;
```

Παραδείγματα

- ✓ **GRANT ALL ON *.* TO 'scott'@'localhost';** (*Θέτει όλα τα privileges πλην GRANT OPTION*)
- ✓ **GRANT ALL ON *.* TO 'scott'@'localhost' with GRANT OPTION;** (*SUPER USER με Grant Option: grant privileges*)
- ✓ **GRANT SELECT, INSERT ON *.* TO 'scott'@'localhost';**

13.10.2. Grant - Database Privileges

Τα database privileges εφαρμόζονται σε όλα τα αντικείμενα της database. Χρησιμοποιούμε την σύνταξη `ON db_name.*`

Παραδείγμα

- ✓ **GRANT ALL ON mydb.* TO 'scott'@'somehost';**
- ✓ **GRANT SELECT, INSERT ON mydb.* TO 'scott'@'somehost';**

Περιλαμβάνονται τα δικαιώματα: CREATE, DROP, GRANT OPTION, και LOCK TABLES. Η MySQL αποθηκεύει τα δικαιώματα database στον πίνακα mysql.db.

13.10.3. Grant - Table Privileges

Τα table privileges εφαρμόζονται σε όλες τις στήλες του πίνακα. Χρησιμοποιούμε την σύνταξη ON db_name.tbl_name.

Παράδειγμα

- ✓ **GRANT ALL ON mydb.mytbl TO 'scott'@'somehost';**
- ✓ **GRANT SELECT, INSERT ON mydb.mytbl TO 'scott'@'somehost';**

Αν στην εντολή καθορίσεις πίνακα tbl_name αντί για db_name.tbl_name, η εντολή θα ερμηνεύσει ως tbl_name τη default database.

Οι επιλογές για privileges σε έναν πίνακα είναι *ALTER, CREATE VIEW, CREATE, DELETE, DROP, GRANT OPTION, INDEX, INSERT, SELECT, SHOW VIEW, TRIGGER, και UPDATE*. Η MySQL αποθηκεύει τα δικαιώματα tables στον πίνακα mysql.tables_priv

13.10.4. Grant - Column privileges

Column privileges εφαρμόζονται σε καθορισμένες στήλες

Παράδειγμα

- ✓ **GRANT SELECT (col1), INSERT (col1,col2) ON mydb.mytbl TO 'scott'@'somehost';**
- ✓ **Ta privileges σε ένα πίνακα είναι: INSERT, SELECT, and UPDATE;**

Η MySQL αποθηκεύει τα δικαιώματα columns στον πίνακα στον πίνακα mysql.columns_priv.

13.10.5. Global privileges

Τα Global privileges είναι:

- τα administrative privileges
- τα privileges που εφαρμόζονται σε όλες τις βάσεις δεδομένων

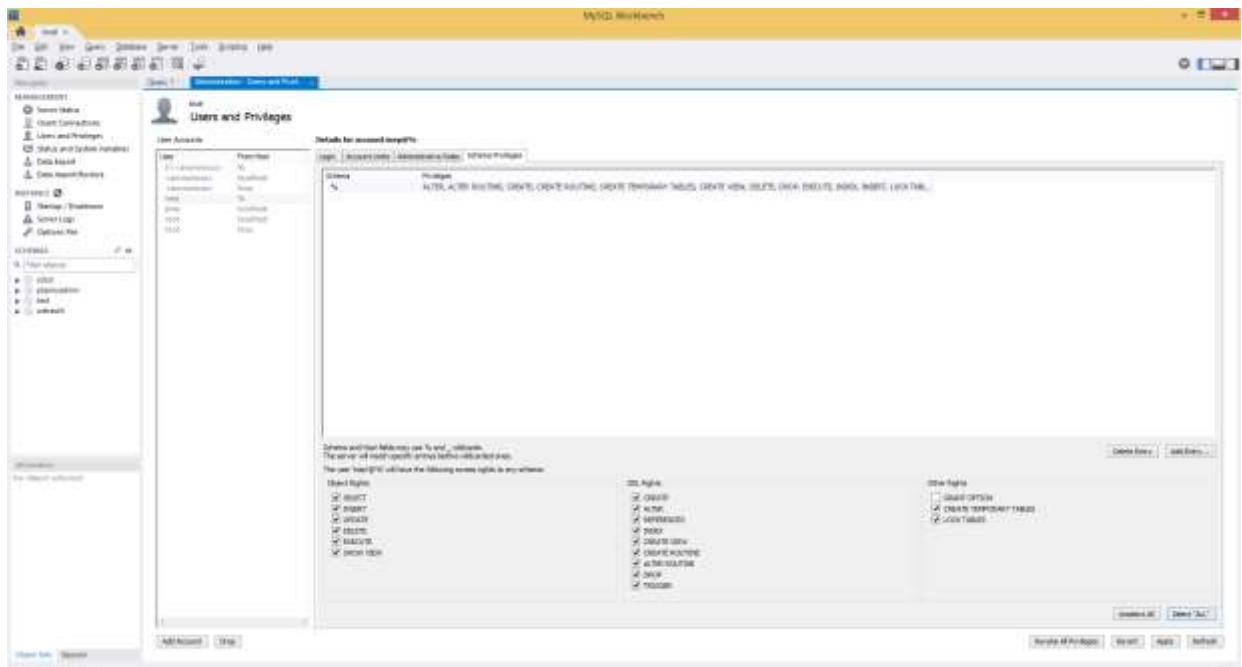
Για να αναθέσουμε Global privileges, χρησιμοποιούμε το ON *.*

Παράδειγμα

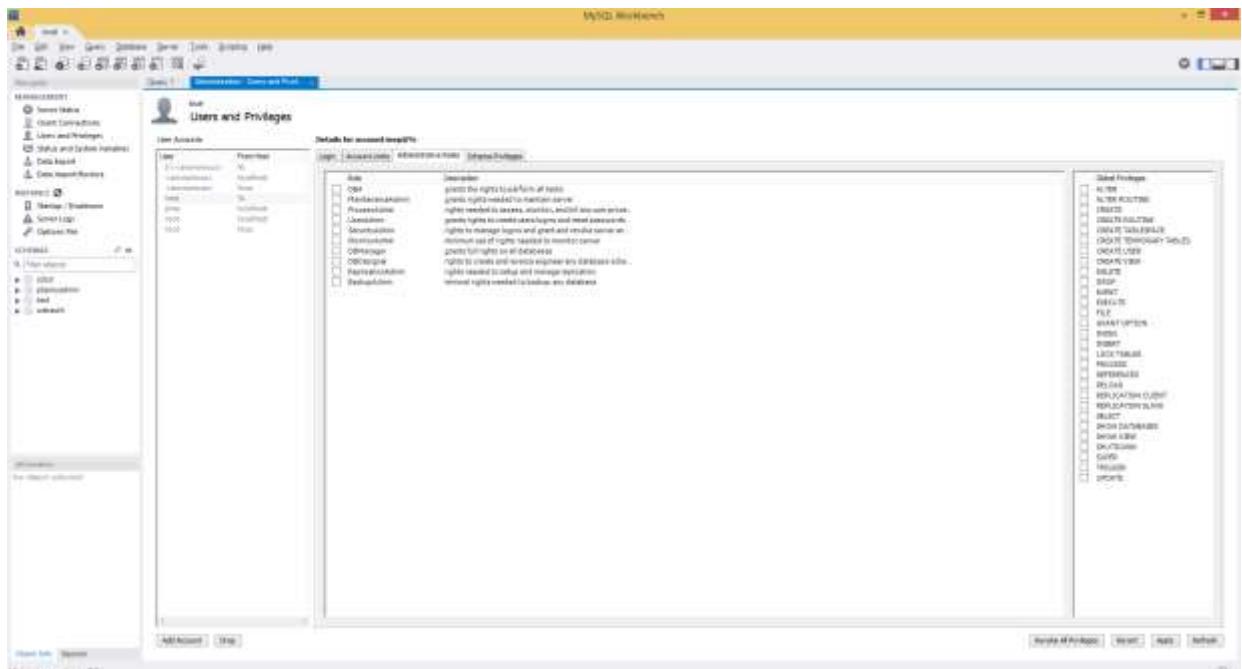
CREATE USER, FILE, PROCESS (show processlist), SHOW DATABASES, SHUTDOWN, and SUPER κ.α.;

H MySQL αποθηκεύει τις global privileges στον πίνακα mysql.user.

Στο MySQL Workbench η απόδοση και αλλαγή δικαιωμάτων (π.χ. ανάκληση) σε κάθε χρήστη γίνεται μέσα από τις παρακάτω δύο οθόνες.



Εικόνα 104 Απόδοση δικαιωμάτων στο χρήστη



Εικόνα 105 Απόδοση δικαιωμάτων στο χρήστη

13.11. Ανάκληση Δικαιωμάτων (Revoking Privileges)

Για την ανάκληση των δικαιωμάτων που έχουν αποδοθεί σε ένα χρήστη χρησιμοποιούμε την εντολή *REVOKE*. Η σύνταξη της έχει τα εξής μέρη:

- Ακολουθείται από μια λίστα δικαιωμάτων τα οποία ανακαλούνται,
- Μια παράμετρο *ON* που δηλώνει το επίπεδο που θα ανακληθούν τα δικαιώματα,
- Μια παράμετρο *FROM* που δηλώνει το όνομα χρήστη

Αν υποθέσουμε ότι ο χρήστης *scott* από τοπικό υπολογιστή έχει δικαιώματα *SELECT*, *DELETE*, *INSERT*, and *UPDATE* στην βάση με όνομα *Thursday* αλλά θέλουμε να αλλάξουμε τα δικαιώματα του λογαριασμού ώστε να έχει μόνο δικαιώματα *SELECT* αρκεί να εκτελέσουμε *REVOKE* στα δικαιώματα που του επιτρέπουν να κάνει όλα τα άλλα. Δηλαδή να εκτελέσουμε την εντολή:

***REVOKE DELETE, INSERT, UPDATE ON Thursday.* FROM 'scott'
@'localhost';***

Η πλήρης σύνταξη της εντολής είναι:

***REVOKE DELETE, INSERT, UPDATE ON Thursday.* FROM 'scott'
@'localhost';
REVOKE priv_type [, priv_type] ...***

```
ON priv_level
FROM user [, user] ...
priv_level:
  *
  / *.* 
  / db_name.* 
  / db_name.tbl_name 
  / tbl_name 
  / db_name.routine_name;
```

Πέρα από την ανάκληση ορισμένων δικαιωμάτων μπορούμε να ανακαλέσουμε όλα τα δικαιώματα. Για την ανάκληση όλων των δικαιωμάτων του χρήστη σε κάθε επίπεδο η REVOKE έχει μια ειδική σύνταξη (σε αυτή την σύνταξη δεν έχει ON τμήμα) και πρέπει να εκτελέσουμε revoke all privileges από τον user ή τους users.

```
REVOKE ALL PRIVILEGES, GRANT OPTION
FROM user [, user] ...;
```

Παράδειγμα

Για την ανάκληση των δικαιωμάτων τύπου GRANT από ένα λογαριασμό χρειάζεται να κάνουμε ανάκληση (revoke) σε μια ξεχωριστή εντολή. Αν για παράδειγμα ο χρήστης scott έχει τη δυνατότητα να κάνει grant τα δικαιώματα του στη βάση thursday σε άλλους χρήστες, μπορούμε να ανακαλέσουμε τα δικαιώματα του με την εντολή:

```
REVOKE GRANT OPTION ON thursday.* FROM 'scott'@'localhost';
```

Για να ανακαλέσουμε όλα τα δικαιώματα που έχει ένας λογαριασμός με όνομα scott:

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'scott'@'localhost';
```

Η REVOKE είναι μια πολύ κρίσιμη εντολή και τα αποτελέσματα μιας λανθασμένης εκτέλεσης μπορεί να αποβούν οδυνηρά για την βάση. Έτσι η MySQL έχει την εντολή SHOW GRANTS, η οποία είναι χρήσιμη για να δει κάποιος τα δικαιώματα πριν και μετά την εκτέλεση κάθε REVOKE:

```
SHOW GRANTS FOR 'scott'@'myhost.example.com';
```

Μια πιθανή έξοδος σε αυτή την εντολή μπορεί να είναι η παρακάτω:

+-----+

```
| Grants for 'scott'@myhost.example.com |
+-----+
| GRANT FILE ON *.* TO 'scott'@'myhost.example.com' |
| GRANT SELECT ON `mydb`.* TO 'scott'@'myhost.example.com' |
| GRANT UPDATE ON `test`.`mytable` TO 'scott'@'myhost.example.com' |
+-----+
```

```
mysql> show grants
-> ;
+-----+
! Grants for albert@%
! GRANT SELECT, SHOW DATABASES, LOCK TABLES, EVENT ON *.* TO 'albert'@'%' IDENTIFIED BY PASSWORD '*C38EC5B6EA7BEA73216A344F7EECB9AA120622EB' ;
! GRANT SELECT, CREATE TEMPORARY TABLES, LOCK TABLES ON `sunday`.* TO 'albert'@'%' ;
+-----+
```

Εικόνα 106 Αποτέλεσμα show grants

Αυτό μας λέει ότι ο λογαριασμός 'scott' έχει global, επιπέδου βάσης και επιπέδου πίνακα δικαιώματα.

Μια καλή πρακτική για να αφαιρέσετε κάποια ή όλα τα δικαιώματα είναι να μετατραπούν οι GRANT εντολές της SHOW GRANTS, τα ονόματα δικαιωμάτων, επιπέδου και τα ονόματα χρηστών όπως εμφανίζονται, σε REVOKE.

Παράδειγμα

Για να καταργήσουμε τα δικαιώματα global FILE εκτελούμε:

REVOKE FILE ON *.* FROM 'scott'@'myhost.example.com';

Εκτελώντας τώρα την *SHOW GRANTS FOR 'scott'@'myhost.example.com'*

```
+-----+
| Grants for jen@myhost.example.com |
+-----+
| GRANT USAGE ON *.* TO 'scott'@'myhost.example.com' |
| GRANT SELECT ON `mydb`.* TO 'scott'@'myhost.example.com' |
| GRANT UPDATE ON `test`.`mytable` TO 'scott'@'myhost.example.com' |
+-----+
```

Σημειώνουμε ότι αν καταργήσουμε όλα τα δικαιώματα από ένα χρήστη δεν θα καταργηθεί και ο ίδιος ο χρήστης. Για να γίνει αυτό θα πρέπει να εκτελέσουμε την *DROP USER*.

13.12. Περιορισμός δικαιωμάτων πρόσβασης

Εξ ορισμού δεν υπάρχει όριο στον αριθμό των συνδέσεων ενός χρήστη στο διακομιστή ή στον αριθμό των ερωτημάτων που μπορεί να εκτελέσει. Σε ορισμένες περιπτώσεις όμως κάτι τέτοιο μπορεί να απαιτηθεί και αυτό γίνεται με την εντολή *GRANT*. Με αυτό τον τρόπο δεν επιτυγχάνεται μόνο περιορισμός στην πρόσβαση κάποιου χρήστη αλλά μπορούν να τεθούν όρια στην κατανάλωση πόρων του εξυπηρετητή από ένα λογαριασμό για:

- Τον αριθμό των συνδέσεων του χρήστη ανά ώρα,
- Τον αριθμό των ερωτημάτων στα δεδομένα από το χρήστη ανά ώρα,
- Τον αριθμό των ενημερώσεων στα δεδομένα από το χρήστη ανά ώρα,
- Τον αριθμό των ταυτόχρονων συνδέσεων του χρήστη στο διακομιστή.

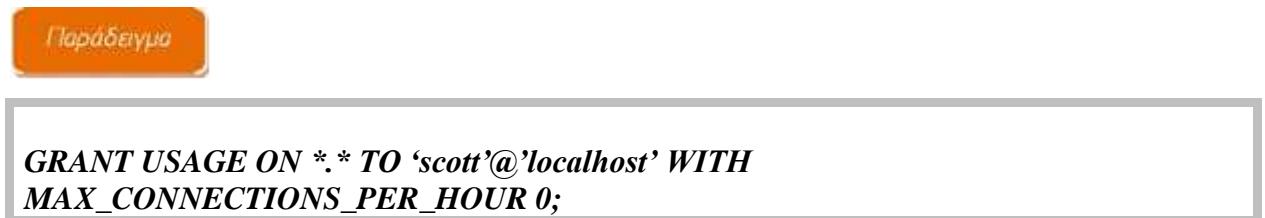
Κάθε ένα από αυτά τα όρια των πόρων καθορίζεται χρησιμοποιώντας τις επιλογές της *WITH*. Στο παράδειγμα που ακολουθεί δημιουργείται ένας λογαριασμός που μπορεί να χρησιμοποιεί τη βάση δεδομένων *test* αλλά μπορεί να συνδεθεί με το διακομιστή το πολύ 10 φορές ανά ώρα. Ο λογαριασμός μπορεί να εκτελέσει 50 ερωτήματα ανά ώρα, και 20 από αυτά τα ερωτήματα μπορούν να τροποποιήσουν τα δεδομένα:

Παράδειγμα

```
GRANT ALL ON test.* TO 'scott'@'localhost' IDENTIFIED BY  
'SomePass'  
WITH MAX_CONNECTIONS_PER_HOUR 10  
MAX_QUERIES_PER_HOUR 50  
MAX_UPDATES_PER_HOUR 20;
```

Η σειρά με την οποία τοποθετούνται οι επιλογές της *WITH* δεν έχει σημασία. Για να ακυρώσουμε κάθε έναν από αυτούς τους περιορισμούς στα εξ ορισμού απεριόριστα

αρκεί να βάλουμε την τιμή μηδέν. Παρόλαυτά αυτό δεν σημαίνει απεριόριστα αλλά λαμβάνει το μέγιστο αριθμό που δίνει το σύστημα.



Στο MySQL Workbench μπορούμε να περιορίσουμε τα δικαιώματα πολύ εύκολα μέσω της οθόνης της Εικόνα 107 Περιορισμός δικαιωμάτων.



Εικόνα 107 Περιορισμός δικαιωμάτων

13.13. Δικαιώματα για διαχείριση χρηστών

Οι λογαριασμοί που κάνουν διαχείριση των χρηστών απαιτείται να έχουν τα εξής δικαιώματα:

- ✓ Για την δημιουργία CREATE USER ή INSERT,
- ✓ Για την διαγραφή CREATE USER ή DELETE,
- ✓ Για την μετονομασία CREATE USER ή UPDATE,
- ✓ Για απόδοση δικαιωμάτων GRANT απαιτείται GRANT OPTION και τα δικαιώματα που θέλει να αποδώσει στους άλλους,
- ✓ Για άρση δικαιωμάτων REVOKE αν δεν έχει ALL PRIVILEGES απαιτεί GRANT OPTION και τα δικαιώματα που θέλει να ανακαλέσει από τους άλλους,

- ✓ Για ανάκληση όλων των δικαιωμάτων απαιτούνται δικαιώματα CREATE USER ή UPDATE,
- ✓ Για την αλλαγή κωδικού πρόσβασης του τα δικαιώματα SET PASSWORD και για αλλαγή κωδικού πρόσβασης άλλου λογαριασμού δικαιώματα CREATE USER ή UPDATE,
- ✓ Κάθε χρήστης μπορεί να χρησιμοποιήσει την SET PASSWORD για να τροποποιήσει το δικό του κωδικό πρόσβασης αλλά όχι των άλλων,
- ✓ Για να εκτελεστεί η SHOW GRANTS για άλλους χρήστες απαιτούνται δικαιώματα SELECT στη βάση, ενώ για να δει τα δικαιώματα του ίδιου λογαριασμού δεν χρειάζεται κάτι επιπλέον.

14. Συντήρηση – Υποστήριξη πινάκων

Μελετώντας το παρόν κεφάλαιο οι επιμορφωνόμενοι θα είναι σε θέση να:

- συντηρούν πίνακες της βάσης δεδομένων
- δημιουργούν αντίγραφα ασφαλείας της βάσης δεδομένων
- εξάγουν το σχήμα της βάσης δεδομένων
- αποκαθιστούν (restore) μια βάση δεδομένων
- εξάγουν δεδομένα
- εισάγουν δεδομένα
- ανακτούν μέτα-πληροφορία για όλα αντικείμενα του εξυπηρετητή.

14.1. Τύποι παρεχόμενων εντολών

Η MySQL παρέχει μια λίστα από χρήσιμες εντολές DCL που επιτρέπουν να εκτελούμε εργασίες συντήρησης στους πίνακες των βάσεων δεδομένων. Οι εργασίες αυτές είναι τόσο ελέγχου όσο και αποκατάστασης των προβλημάτων. Επίσης διαθέτει και εργασίες συντήρησης με σκοπό την ανάλυση και την βελτιστοποίηση (optimization).

Οι τύποι των εργασιών συντήρησης ενός πίνακα πιο συγκεκριμένα συνοψίζονται στα:

- ✓ Λειτουργία ελέγχου ενός πίνακα, εκτελεί έλεγχο ακεραιότητας για να βεβαιωθεί ότι η δομή του πίνακα και τα περιεχόμενα του δεν έχουν κάποιο πρόβλημα. Αυτή η λειτουργία μπορεί να εκτελεστεί τόσο σε πίνακες τύπου MyISAM όσο και σε πίνακες τύπου InnoDB.
- ✓ Λειτουργία επιδιόρθωσης ενός πίνακα, διορθώνονται όλα τα προβλήματα ακεραιότητας και επανέρχεται ο πίνακας σε μια γνωστή κατάσταση που μπορεί να χρησιμοποιηθεί. Αυτή η λειτουργία είναι εφαρμόσιμη μόνο σε πίνακες MyISAM.
- ✓ Λειτουργία ανάλυσης πίνακα ενημερώνει τα στατιστικά στοιχεία σχετικά με την κατανομή των τιμών των δεικτών του πίνακα, καθώς αυτή είναι η πληροφορία την οποία χρησιμοποιεί ο βελτιστοποιητής εκτέλεσης λειτουργιών της βάσης για να εκτελούνται με τον αποδοτικότερο τρόπο τα όποια ερωτήματα. Αυτή η

λειτουργία μπορεί να εκτελεστεί τόσο σε πίνακες βάσεων MyISAM όσο και σε πίνακες InnoDB.

- ✓ Λειτουργία βελτιστοποίησης, αναδιοργανώνει έναν πίνακα, ώστε το περιεχόμενό του να προσπελαύνεται αποτελεσματικότερα. Αυτή η λειτουργία μπορεί να εκτελεστεί τόσο σε πίνακες βάσεων MyISAM όσο και σε πίνακες InnoDB.

Από αυτές τις λειτουργίες η ανάλυση και η βελτιστοποίηση καλό είναι να εκτελούνται περιοδικά για να διατηρούν τους πίνακες στην καλύτερη απόδοση.

Όταν η MySQL αναλύει ένα MyISAM ή InnoDB πίνακα, ενημερώνει τα στατιστικά των δεικτών. Ο βελτιστοποιητής χρησιμοποιεί αυτά τα στατιστικά όταν επεξεργάζεται κάποιο ερώτημα για να λάβει τις βέλτιστες αποφάσεις για τον τρόπο που θα αναζητήσει τις εγγραφές στον πίνακα, καθώς και την σειρά που θα διαβάσει τους πίνακες σε ενδεχόμενα ερωτήματα με join. Ειδικά στην περίπτωση της βελτιστοποίησης σε πίνακες MyISAM εκτελείται ανασυγκρότηση του αρχείου αποθήκευσης από μη χρησιμοποιήσιμα στοιχεία κενά, και ενημερώνει τα στατιστικά των δεικτών. Έτσι ο πίνακας αποκρίνεται αποδοτικότερα. Οι εργασίες ανάλυσης και βελτιστοποίησης έχουν τα καλύτερα αποτελέσματα όταν εκτελούνται σε πίνακες των οποίων δεν επηρεάζονται τα στοιχεία εκείνη τη στιγμή.

Τα εργαλεία που παρέχει η MySQL για τις παραπάνω εργασίες είναι SQL εντολές όπως CHECK TABLE και REPAIR TABLE, τα διάφορα προγράμματα διαχείρισης όπως το mysqlcheck, myisamchk και τις δυνατότητες του διακομιστή βάσης για αυτόματη ανάκτηση (auto-recovery).

14.2. Εντολές SQL για την συντήρηση πινάκων

Η MySQL έχει μια σειρά από SQL εντολές για την εκτέλεση συντήρησης πινάκων, οι εντολές λαμβάνουν ως παραμέτρους ένα ή περισσότερα ονόματα πινάκων και κάποιες παραμέτρους εκτέλεσης αυτών, οι οποίες παραμετροποιούν την εκτέλεση. Το όνομα ενός πίνακα μπορεί να αναφερθεί ως *table_name* το οποίο αναφέρεται στην βάση που είναι σε χρήση εκείνη τη στιγμή, ή *db_name.table_name* το οποίο αναφέρεται σε ένα πίνακα σε συγκεκριμένη βάση.

14.2.1. Εντολή ελέγχου πινάκων (CHECK TABLE)

Η εντολή *CHECK TABLE* εκτελεί έναν έλεγχο ακεραιότητας στον πίνακα και τα περιεχόμενα του, ενώ ενημερώνει και τα στατιστικά στοιχεία για τον πίνακα. Η εντολή μπορεί να εκτελεστεί για πίνακες τύπου MyISAM και InnoDB. Η εντολή μπορεί να εκτελεστεί και για όψεις (views), στην περίπτωση αυτή επαληθεύει τον ορισμό της όψης (view).

Παράδειγμα

Αν η βάση *thursday* είναι η τρέχουσα, οι εντολές για έλεγχο του πίνακα *employees*, σύμφωνα με τα παραπάνω είναι:

```
CHECK TABLE employees;  
ἢ  
CHECK TABLE thursday.employees;
```

Η εκτέλεση της παραπάνω εντολής έχει ως πιθανό αποτέλεσμα στην κονσόλα το παρακάτω:

```
+-----+-----+-----+-----+  
| Table | Op  | Msg_type | Msg_text |  
+-----+-----+-----+-----+  
| thursday.employees | check | status | OK |  
+-----+-----+-----+-----+
```

Όπου το *Op* είναι η εντολή που εκτελέστηκε, *Msg_type* είναι η απάντηση της επιτυχίας ή αποτυχίας ελέγχου *success* or *failure* και *Msg_text* επιπλέον πληροφορία.

14.2.2. Επιδιόρθωση πινάκα (REPAIR TABLE)

Η εντολή *REPAIR TABLE* επισκευάζει και διορθώνει τα όποια προβλήματα σε ένα πίνακα που έχει καταστραφεί. Η λειτουργία μπορεί να εφαρμοστεί μόνο σε πίνακες MyISAM. Υπάρχει δυνατότητα να ρυθμιστεί ώστε ο εξυπηρετητής MySQL να επισκευάζει αυτόματα τους πίνακες. Με αυτό τον τρόπο ο εξυπηρετητής MySQL ελέγχει κάθε MyISAM πίνακα αυτόματα και αν χρήζει επισκευής εκτελεί την λειτουργία επισκευής. Για να ενεργοποιήσουμε αυτή την δυνατότητα πρέπει να

εκκινήσουμε τον εξυπηρετητή με την επιλογή --myisam-recover. Η επιλογή recover μπορεί να πάρει διάφορες τιμές:

- ✓ DEFAULT για προκαθορισμένο έλεγχο,
- ✓ BACKUP εκτελεί και backup σε κάθε πίνακα που θα τροποποιηθεί,
- ✓ FORCE προκαλεί ανάκτηση ακόμα και αν συμβεί απώλεια δεδομένων κάποιας εγγραφής,
- ✓ QUICK εκτελεί γρήγορη ανάκτηση, παρακάμπτοντας τους πίνακες που δεν χρειάζονται διόρθωση.

Παράδειγμα

Για την επισκευή του πίνακα polites χρειάζεται να εκτελεστεί η παρακάτω εντολή:

```
repair table polites;
```

Παράδειγμα

Για να ρυθμιστεί ώστε ο εξυπηρετητής MySQL να επισκευάζει αυτόματα τους πίνακες πρέπει να εκκινήσουμε τον εξυπηρετητή με την επιλογή --myisam-recover όπου εκτελεί και backup σε κάθε πίνακα που θα τροποποιηθεί:

```
mysqld --myisam-recover=FORCE,BACKUP;
```

14.2.3. Ανάλυση πίνακα (ANALYZE TABLE)

Η εντολή *ANALYZE TABLE* ενημερώνει τον πίνακα με πληροφορίες σχετικά με την κατανομή των τιμών κλειδιών του πίνακα. Οι πληροφορίες χρησιμοποιούνται για χρήση της δημιουργίας βέλτιστης εκτέλεσης εντολών στον πίνακα. Αυτή η λειτουργία μπορεί να εκτελεστεί τόσο σε πίνακες βάσεων τύπου MyISAM όσο και σε InnoDB.

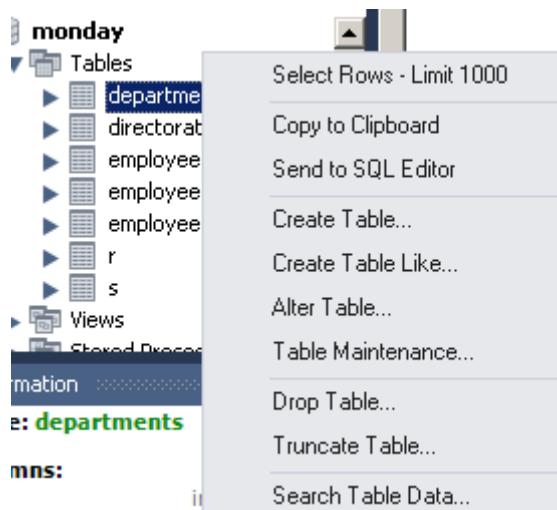
Στον MySQL Client εκτελείται:

```
ANALYZE TABLE table_name;
```



Εικόνα 108

Μέσα από το MySQL Workbench γίνονται πιο εύκολα μέσα από την επιλογή του μενού Table Maintenance



Εικόνα 109 Table Maintenance

Name	Engine
departments	InnoDB
directorates	InnoDB
employees	InnoDB
employees_sub	InnoDB
employees_tbl	InnoDB
r	InnoDB
s	InnoDB

Table Maintenance Operations

Select tables and click the operation you want to perform.
NOTE: Some commands may require locking tables until completion, which may take a long time for large tables.

Analyze Table

Analyzes and stores the key distribution for a table.
During the analysis, the table is locked with a read lock for InnoDB and MyISAM.

Don't write to BINLOG (local)

Optimize Table

Reorganizes the physical storage of table data and associated index data, to reduce storage space and improve I/O efficiency when accessing the table.

Optimize FULLTEXT only Number of words to optimize per run: 2000

Don't write to BINLOG (local)

Check Table

CHECK TABLE checks a table or tables for errors.
For MyISAM tables, the key statistics are updated as well.

Quick Fast Changed

Εικόνα 110 Table Maintenance

14.2.4. Βελτιστοποίηση πίνακα (Optimize table)

Η εντολή *OPTIMIZE TABLE* καθαρίζει ένα τύπου MyISAM πίνακα ανασυγκροτώντας τον. Αυτό περιλαμβάνει αφαίρεση αχρησιμοποίητου χώρου που προκύπτει από διαγραφές και ενημερώσεις ή από στοιχεία τα οποία δεν έχουν αποθηκευτεί διαδοχικά. Επίσης ταξινομεί τους δείκτες που είναι εκτός σειράς και ενημερώνει τα στατιστικά του πίνακα.

Στον MySQL Client εκτελείται:

```
OPTIMIZE TABLE table_name;
```

Σε command line με την mysqlcheck (θα το δούμε παρακάτω)

```
MYSQLCHECK -u root -p --optimize --all-databases
MYSQLCHECK -u root -p --optimize database_name
```

Η εντολή *OPTIMIZE TABLE* λειτουργεί και για πίνακες τύπου InnoDB, αλλά ουσιαστικά είναι σαν να εκτελείται *ALTER TABLE* μόνο που ελευθερώνει τον αχρησιμοποίητο χώρο από διαγραμμένες εγγραφές και ξαναχτίζει τον πίνακα και τα indexes του ενώ ενημερώνει τους πίνακες στατιστικών.

```
ALTER TABLE table_name ENGINE='InnoDB';
```



Εικόνα 111

14.3. Εργαλεία πελάτη για την συντήρηση πινάκων

Οι εντολές που αναφέρθηκαν στις προηγούμενες ενότητες μπορούν να εκτελεστούν από το εργαλείο πελάτη mysql ή άλλες εφαρμογές διαχείρισης της βάσης MySQL. Έτσι μπορούν να κατασκευαστούν εφαρμογές οι οποίες θα εκτελούν διαχειριστικές εργασίες και χρησιμοποιούν αυτές τις εντολές. Ο εξυπηρετητής MySQL παρέχει και αυτός τις ίδιες διαχειριστικές δυνατότητες ελέγχου και αποκατάστασης πινάκων. Επίσης άλλα εργαλεία όπως το mysqlcheck μπορεί να ελέγχει, επιδιορθώσει, αναλύσει και βελτιστοποιήσει πίνακες. Το εργαλείο myisamchk εκτελεί τις ίδιες εργασίες συντήρησης για πίνακες MyISAM. Παρόλαντά έχει μια διαφορετική προσέγγιση από το βασικό διαχειριστικό εργαλείο MySQL και το εργαλείο mysqlcheck. Η βασική διαφορά παρουσιάζεται στον τρόπο λειτουργίας όπου αντί να αποστέλλει εντολές στον εξυπηρετητή MySQL, η myisamchk διαβάζει και τροποποιεί άμεσα τα αρχεία του πίνακα. Για αυτό το λόγο είναι καλή πρακτική όταν χρησιμοποιούμε το εργαλείο myisamchk να είμαστε σίγουροι ότι ο εξυπηρετητής δεν προσπελαύνει τους πίνακες την ίδια στιγμή.

14.3.1. Το εργαλείο mysqlcheck

Το εργαλείο *mysqlcheck* ελέγχει, επιδιορθώνει, αναλύει και βελτιστοποιεί τους πίνακες. Μπορεί να εκτελέσει όλες αυτές τις ενέργειες σε πίνακες MyISAM και κάποιες από αυτές σε πίνακες InnoDB. Παρέχει ένα περιβάλλον γραμμής εντολών για να εκτελεστούν οι SQL εντολές όπως *CHECK TABLE* and *REPAIR TABLE*. Το εργαλείο *mysqlcheck* έχει ιδιότητες τέτοιες που κάποιες φορές το κάνουν πιο εύχρηστο από την απευθείας εκτέλεση SQL εντολών. Έτσι για παράδειγμα εάν οριστεί η βάση δεδομένων τότε από μόνο του καταλαβαίνει ποιους πίνακες περιλαμβάνει και εκτελεί τις εντολές για να τους επεξεργαστεί όλους, έτσι δεν απαιτείται να ονοματιστεί κάθε πίνακας ξεχωριστά. Επίσης επειδή το εργαλείο *mysqlcheck* ως εργαλείο γραμμής εντολών είναι πιο εύκολο να χρησιμοποιηθεί σε εργασίες οι οποίες εκτελούν προγραμματισμένες συντηρήσεις. Το εργαλείο *mysqlcheck* έχει τρείς καταστάσεις λειτουργίας, οι οποίες ρυθμίζονται ανάλογα με τις παραμέτρους που χρησιμοποιούνται:

- Εξ' ορισμού, *mysqlcheck* ερμηνεύει το πρώτο όρισμα που δεν είναι παράμετρος ως όνομα βάσης δεδομένων και ελέγχει όλους τους πίνακες της βάσης με αυτό

το όνομα. Εάν κάποια άλλη παράμετρος ακολουθεί το όνομα της βάσης δεδομένων το εργαλείο το ερμηνεύει σαν όνομα πίνακα και ελέγχει μόνο αυτούς τους πίνακες.

Παράδειγμα

Το παράδειγμα ελέγχει όλους τους πίνακες της βάσης δεδομένων *world*, ενώ το δεύτερο μόνο τους πίνακες *City* και *Country*:

```
mysqlcheck world  
mysqlcheck world City Country
```

- Με το όρισμα *--databases* (ή *-B*) το εργαλείο *mysqlcheck* ερμηνεύει όλα τα ορίσματα σαν ονόματα βάσεων και ελέγχει όλους τους πίνακες σε κάθε μια από τις βάσεις.

Παράδειγμα

Το ακόλουθο παράδειγμα ελέγχει τους πίνακες των βάσεων *world* και *test*:

```
mysqlcheck --databases world test
```

- Με το όρισμα *--all-databases* (ή *-A*), το εργαλείο *mysqlcheck* ελέγχει όλους τους πίνακες όλων των βάσεων:

Παράδειγμα

```
mysqlcheck --all-databases
```

Επίσης το εργαλείο *mysqlcheck* υποστηρίζει και ορίσματα τα οποία προσδιορίζουν ποια λειτουργία πρέπει να εκτελεστεί στους πίνακες. Τα ορίσματα *--check*, *--repair*, *--analyze* και *--optimize* εκτελούν αντίστοιχα έλεγχο, επιδιόρθωση, ανάλυση, και βελτιστοποίηση. Εξ' ορισμού αν δεν δίνεται καμία παράμετρος εκτελείται έλεγχος. Μια καλή πρακτική

χρήσης είναι να εκτελείται η *mysqlcheck* χωρίς παραμέτρους, εάν προκύπτει κάποιο σφάλμα τότε να εκτελείται ξανά η *mysqlcheck* πρώτα με *--repair --quick*, για γρήγορη επιδιόρθωση. Εάν αυτά αποτύχουν τότε συστήνεται να εκτελεστεί *mysqlcheck* μόνο με *--repair* για κανονική επιδιόρθωση και μετά αν είναι αναγκαίο με *--repair* και *--force*.

Παράδειγμα

Όλες οι επιλογές

mysqlcheck --check;

mysqlcheck --repair;

mysqlcheck --analyze;

mysqlcheck --optimize;

Σειρά εντολών καλής πρακτικής

mysqlcheck

mysqlcheck --repair --quick;

mysqlcheck --repair;

mysqlcheck --repair και --force;

14.3.2. Το εργαλείο myisamchk

Το εργαλείο *myisamchk* εκτελεί συντήρηση πινάκων τύπου *MyISAM*. Ουσιαστικά είναι παρόμοιο με το *mysqlcheck*, αλλά τα δύο προγράμματα διαφέρουν σε συγκεκριμένα σημεία:

- Η βασικότερη διαφορά είναι ότι και τα δύο προγράμματα μπορούν να ελέγξουν, επιδιορθώσουν και να αναλύσουν πίνακες τύπου *MyISAM*. Το *mysqlcheck* μπορεί να βελτιστοποιεί πίνακες τύπου *MyISAM*, όπως και να ελέγχει πίνακες τύπου *InnoDB*. Υπάρχουν μια σειρά από λειτουργίες που η *myisamchk* μπορεί να εκτελεί και η *mysqlcheck* δεν μπορεί, όπως η απενεργοποίηση και ενεργοποίηση δεικτών.
- Τα δύο εργαλεία διαφέρουν σημαντικά στον τρόπο λειτουργίας τους. Το *mysqlcheck* είναι ένα πρόγραμμα πελάτη το οποίο επικοινωνεί με το MySQL

εξυπηρετητή μέσα από το δίκτυο. Αυτό σημαίνει ότι το *mysqlcheck* απαιτεί να είναι ενεργός ο εξυπηρετητής, από την άλλη σημαίνει ότι με το *mysqlcheck* μπορούμε να συνδεθούμε και σε απομακρυσμένους εξυπηρετητές. Αντίθετα το *myisamchk* δεν είναι ένα πρόγραμμα τύπου πελάτη. Είναι ένα εργαλείο το οποίο έχει επίδραση κατευθείαν στα αρχεία τα οποία αποθηκεύονται πίνακες MyISAM. Αυτό σημαίνει ότι το *myisamchk* πρέπει να εκτελεστεί στον εξυπηρετητή που βρίσκονται αυτά τα αρχεία. Επίσης θα πρέπει να υπάρχουν τα κατάλληλα δικαιώματα στα αρχεία αυτά.

- Τα δύο προγράμματα διαφέρουν στη σχέση τους με τον εξυπηρετητή όταν εκτελούνται. Με το *mysqlcheck*, δεν υπάρχει κανένα πρόβλημα διάδρασης με τον εξυπηρετητή γιατί το *mysqlcheck* ζητά από τον ίδιο τον εξυπηρετητή να εκτελέσει τον έλεγχο και την επιδιόρθωση των πινάκων. Με το *myisamchk* πρέπει να βεβαιωθεί ο χρήστης ότι ο εξυπηρετητής δεν έχει ανοικτούς τους πίνακες και δεν τους επεξεργάζεται ταυτόχρονα. Εάν τα αρχεία του πίνακα χρησιμοποιούνται από *myisamchk* και το διακομιστή ταυτόχρονα είναι δυνατόν να πάρει λανθασμένα αποτελέσματα ή ακόμα και να προκληθεί ζημιά στον πίνακα. Ο πιο σίγουρος τρόπος για να αποφευχθεί η σύγκρουση κατά την εκτέλεση του *myisamchk* είναι να σταματήσει ο διακομιστής MySQL. Επίσης μπορεί να αφεθεί να τερματίσει η λειτουργία του διακομιστή και να κλειδώσει τους πίνακες κατά τον έλεγχο ή την επισκευή τους με το *myisamchk*.
- Επειδή θα πρέπει να αποφευχθεί η ταυτόχρονη χρήση πινάκων, η διαδικασία για τη χρήση *myisamchk* διαφέρει από εκείνη για τη χρήση *mysqlcheck*. Έτσι συστήνεται να εκτελείται η διαδικασία συντήρησης με *myisamchk* ως εξής:

1. Αρχικά βεβαιωνόμαστε ότι ο εξυπηρετητής δεν έχει πρόσβαση στους πίνακες όσο δουλεύουμε με αυτούς. Ένας καλός τρόπος για να βεβαιωθούμε για αυτό είναι να σταματήσουμε το εξυπηρετητή.
2. Από την γραμμή εντολών πρέπει να πάμε στον φάκελο που βρίσκονται τα αρχεία της βάσης δεδομένων. Αυτός είναι ο υποφάκελος που θα έχει το ίδιο όνομα με την βάση και θα περιέχει τα αρχεία που πινάκων που χρειάζεται να ελεγχθούν.
3. Στη συνέχεια καλείται η *myisamchk* με τις επιλογές που δείχνουν τι θέλουμε να εκτελέσουμε, ακολουθούμενη από τις παραμέτρους που δηλώνουν τους πίνακες στους οποίους θα εκτελεστεί η *myisamchk*. Κάθε παράμετρος μπορεί να είναι το όνομα του πίνακα ή το όνομα του αρχείου που αποθηκεύει τον πίνακα. Τα αρχεία αυτά έχουν

όνομα όπως του πίνακα με κατάληξη .MYI. Έτσι μπορούμε να αναφερθούμε είτε στον πίνακα σαν *όνομα_πίνακα* ή στο αρχείο σαν *όνομα_πίνακα.MYI*.

4. Επανεκκίνηση του εξυπηρετητή. Η εξ ορισμού λειτουργία του εργαλείου *myisamchk* είναι ο έλεγχος πινάκων. Εάν αυτό είναι που επιθυμούμε να κάνουμε τότε δεν απαιτούνται επιπλέον παράμετροι και χρειάζονται μόνο το όνομα του πίνακα ή των πινάκων. Στο επόμενο παράδειγμα δείχνουμε ακριβώς αυτό:

Παράδειγμα

```
myisamchk City;  
η  
myisamchk City.MYI;
```

Για να επισκευάσουμε ένα πίνακα χρησιμοποιούμε την επιλογή --recover:

Παράδειγμα

```
myisamchk --recover City;
```

Εάν κατά την διάρκεια επισκευής με την παράμετρο --recover εμφανιστεί κάποιο πρόβλημα που δεν μπορεί να επισκευαστεί τότε μπορεί να δοκιμαστεί το -safe - recover.

14.3.3. Επιλογές των mysqlcheck και myisamchk

Τα εργαλεία *mysqlcheck* και *myisamchk* έχουν και τα δύο αρκετές παραμέτρους κάποιες από τις οποίες αναφέραμε παραπάνω. Στη συνέχεια παραθέτουμε συνολικά μερικές από τις πιο χρησιμοποιούμενες επιλογές, οι περισσότερες εκ των οποίων εκτελούνται και από τα δύο εργαλεία.

- --analyze or -a, Αναλύει την κατανομή των βασικών τιμών στον πίνακα. Αυτό μπορεί να βελτιώσει τις επιδόσεις των ερωτημάτων,

- `--auto-repair` (μόνο στη `mysqlcheck`), επισκευάζει πίνακες αυτόματα μόλις βρει κάποιο πρόβλημα,
- `--check or -c`, ελέγχει τον πίνακα, αυτή είναι και η προεπιλεγμένη ενέργεια όταν καμία άλλη παράμετρος δεν έχει καθοριστεί,
- `--check-only-changed or -C`, ελέγχει μόνο τους πίνακες που έχουν τροποποιηθεί πριν από τον τελευταίο έλεγχο τους ή δεν έχουν κλείσει σωστά,
- `--fast or -F`, ελέγχει μόνο τους πίνακες που δεν έχουν κλείσει σωστά,
- `--extended` (for `mysqlcheck`), `--extend-check` (for `myisamchk`), or `-e` (for both programs), εκτελεί μια εκτεταμένη λειτουργία ελέγχου πίνακα. Στην περίπτωση του `mysqlcheck`, όταν δοθεί αυτή η επιλογή και συνδυαστεί με την επιλογή επισκευής, πραγματοποιείται μια πιο ενδελεχής επισκευή. Έτσι η `mysqlcheck --repair--extended` πραγματοποιεί βαθύτερη ανάλυση από την `mysqlcheck --repair`,
- `--medium-check or -m`, εκτελεί έναν μέτριο έλεγχο πίνακα,
- `--quick or -q`, και στα δύο εργαλεία η `--quicck` χωρίς την επιλογή `repair` ελέγχει μόνο τους δείκτες και όχι το αρχείο,
- `--repair` (`mysqlcheck`), `--recover` (`myisamchk`), ή `-r` (και για τα δύο), εκτελούν μια επιδιόρθωση πίνακα.

14.4. Διαφορές σε πίνακες τύπου MyISAM και InnoDB

14.4.1. Επιδιόρθωση πινάκων τύπου InnoDB

Στην περίπτωση των πινάκων τύπου InnoDB μπορεί να πραγματοποιηθεί έλεγχος μέσω του `CHECK TABLE` ή με άλλο τρόπο, όμως δεν μπορεί να γίνει επισκευή μιας και η `REPAIR TABLE` είναι μόνο για MyISAM. Στην περίπτωση αυτή πρέπει να ανακτήσουμε τον πίνακα σε μια άρτια κατάσταση μέσω διαδικασίας dumping (εξηγείται παρακάτω) ακολουθώντας τα εξής βήματα:

```
mysqldump db_name table_name > dump_file;
mysql db_name < dump_file;
```

Σε περίπτωση δυσλειτουργίας του εξυπηρετητή MySQL ή του πελάτη μπορεί να δημιουργηθεί τέτοιο πρόβλημα ώστε να απαιτείται ορισμένοι πίνακες τύπου InnoDB να

χρειάζονται επισκευές. Κανονικά, αρκεί να γίνει επανεκκίνηση του εξυπηρετητή, επειδή η μηχανή αποθήκευσης InnoDB εκτελεί αυτόματη ανάκτηση ως μέρος της εκκίνησης. Σε σπάνιες περιπτώσεις όμως, ο εξυπηρετητής δεν μπορεί να ξεκινήσει λόγω της αποτυχίας αυτόματης ανάκτησης του InnoDB. Στην περίπτωση αυτή χρησιμοποιούμε την ακόλουθη διαδικασία:

- Επανεκκίνηση του εξυπηρετητή με την επιλογή - `innodb_force_recovery` με επιλογή με τιμή στην κλίμακα 1-6. Αυτές οι τιμές υποδεικνύουν την αύξηση των επιπέδων της προσοχής στην αποφυγή μιας σύγκρουσης, καθώς και την αύξηση των επιπέδων ανοχής για πιθανή ασυνέπεια στους ανακτημένους πίνακες. Μια καλή τιμή για να ξεκινήσετε είναι 4.

```
mysqld --defaults-file="C:\Program Files\MySQL\MySQL Server 5.6\my.ini" –standalone –console –innodb_force_recovery=1
```

- Όταν γίνει έναρξη του εξυπηρετητή με `--innodb_force_recovery` με μια μη μηδενική τιμή, διαχειρίζεται το tablespace μόνο για ανάγνωση και δεν μπορούν να εκτελεστούν εντολές εισαγωγής ή διαγραφής. Έτσι πρέπει να εκτελεστεί `dump` στους πίνακες InnoDB με χρήση `mysqldump` και μετά να διαγραφούν με `drop`. Στην συνέχεια γίνεται επανεκκίνηση του εξυπηρετητή χωρίς `--innodb_force_recovery` και ανακτώνται οι πίνακες. Πρέπει να δημιουργηθούν νέοι πίνακες και να γίνει εισαγωγή των δεδομένων των κατεστραμμένων πινάκων.
- Εάν αποτύχουν όλα τα παραπάνω τότε οι πίνακες τύπου InnoDB ανακτώνται μόνο από προηγούμενο αντίγραφο ασφαλείας.

Table	Op	Msg_type	Msg_text
monday.departments	repair	note	The storage engine for the table doesn't support repair

Εικόνα 112 Repair table

14.4.2. Επιδιόρθωση πινάκων τύπου MyISAM

Στη περίπτωση των πινάκων τύπου *MyISAM* μπορεί να πραγματοποιηθεί έλεγχος μέσω του REPAIR TABLE. Για την διαδικασία αυτή υπάρχουν δύο τρόποι με τα εξής βήματα:

- **Τρόπος 1:** SQL εντολή Repair σε κάθε corrupted πίνακα με χρήση του MySQL Client:

```
repair table table_name;
```

- **Τρόπος 2:** εκτελούμε command line την εντολή mysqlcheck που κάνει repair την database μιας από corrupted tables.

```
mysqlcheck -u root -p --auto-repair database_name;
```

- Εφαρμογή της εντολής σε όλες τις database του Server

```
mysqlcheck -u root -p --auto-repair --all-databases;
```

14.4.3. Καταστροφή στην Βάση Δεδομένων

Η περίπτωση της καταστροφής βάσης (database corruption) συμβαίνει συνήθως ως αποτέλεσμα αστοχίας υλικού (hardware failure ή disk failure), ή όταν ο δίσκος γεμίσει. Η MySQL μας δίνει τρόπους για να ελέγξουμε αν η βάση μας έχει καταστραφεί.

- **Τρόπος 1:** ελέγχουμε κάθε πίνακα της database με SQL εντολή αν είναι corrupted. Απαιτείται η χρήση MySQL Client

```
check table table_name;
```

- **Τρόπος 2:** εκτελούμε command line την εντολή mysqlcheck που ελέγχει την database μας από corrupted tables.

```
mysqlcheck -u root -p --check database_name;
```

ή

Εφαρμόζουμε την εντολή σε όλες τις βάσεις του εξυπηρετητή

```
mysqlcheck -u root -p --check --all-databases;
```

14.5. Αντίγραφα ασφαλείας

Σε μια MySQL βάση δεδομένων, όπως και σε κάθε άλλη, είναι καλό να διατηρούμε αντίγραφα ασφαλείας της βάσης δεδομένων για να αποφευχθεί το ενδεχόμενο απώλειας δεδομένων μετά από μια απρόβλεπτη κατάσταση ή από αστοχία υλικού, φαινόμενο το οποίο μπορεί να οδηγήσει σε απώλεια ή καταστροφή δεδομένων. Τα αντίγραφα ασφαλείας είναι επίσης χρήσιμα στην περίπτωση που γίνει κατά λάθος διαγραφή πίνακα ή στοιχείων. Επίσης μια ακόμα χρησιμότητα των αντιγράφων ασφαλείας είναι στην περίπτωση μετάπτωσης της όποιας υπηρεσίας σε άλλο εξυπηρετητή. Στην περίπτωση της MySQL τα αντίγραφα ασφαλείας μπορεί να γίνουν είτε με αντιγραφή των αρχείων της βάσης δεδομένων άμεσα, ή με τη χρήση προγραμμάτων που έχουν σχεδιαστεί για το σκοπό αυτό. Τέτοια προγράμματα είναι τα mysqldump, mysqlhotcopy, MySQL Administrator, και το InnoDB Hot Backup.

Για να ολοκληρωθεί πλήρως η ανάκτηση μιας βάσης δεδομένων, μετά από απώλεια ή ζημιά, απαιτείται το αντίγραφο ασφαλείας για να την επαναφέρετε στην κατάστασή που τη λάβατε στο αντίγραφο της και στη συνέχεια να εκτελεστούν όλες οι εντολές που

περιέχονται στο αρχείο καταγραφής για να γίνουν ξανά οι αλλαγές των δεδομένων αφού δημιουργήθηκε το αντίγραφο ασφαλείας.

Ορισμένες βασικές αρχές και καλές πρακτικές για τα αντίγραφά ασφαλείας είναι:

- Η τακτική λήψη αντιγράφων ασφαλείας,
- Η ενεργοποίηση του αρχείου καταγραφής ώστε να υπάρχουν οι ενέργειες αλλαγών που πραγματοποιήθηκαν μετά την λήψη του αντιγράφου ασφαλείας,
- Η αρχικοποίηση των αρχείων καταγραφής (log files) μετά την λήψη αντιγράφου, έτσι ώστε ο εξυπηρετητής να ξεκινάει ένα νέο αρχείο κάθε φορά,
- Η αποθήκευση του φακέλου δεδομένων και των αντιγράφων ασφαλείας σε διαφορετικά φυσικά μέσα ώστε στην περίπτωση καταστροφής να μην καταστραφούν και τα δύο,
- Η ενσωμάτωση της λήψης αντιγράφων ασφαλείας τόσο της βάσης όσο και του αρχείου καταγραφής στο λειτουργικό σύστημα ώστε να είναι εύκολη και η πιθανή ανάκτηση τους αν είναι αναγκαίο.

Υπάρχουν δύο δυνατότητες λήψης αντιγράφων ασφαλείας:

- **Δυαδικά αντίγραφα**, είναι αντίγραφα των αρχείων στα οποία αποθηκεύονται τα περιεχόμενα της βάσης. Αυτά τα αρχεία διατηρούν τη βάση όπως η MySQL την αποθηκεύει στο δίσκο. Η ανάκτηση γίνεται με αντιγραφή των αρχείων στη φυσική τους θέση.
- **Αντίγραφο ασφαλείας κειμένου**, είναι ένα αντίγραφο από dump των περιεχομένων της βάσης σε αρχείο κειμένου. Η ανάκτηση περιλαμβάνει το φόρτωμα αυτών των αρχείων με επεξεργασία από τον εξυπηρετητή. Έτσι χρησιμοποιούνται τεχνικές SELECT ... INTO OUTFILE SQL statement, mysqldump, και MySQL Administrator.

Οι δύο μορφές δημιουργίας αντιγράφων ασφαλείας έχουν διαφορετικά πλεονεκτήματα και μειονεκτήματα. Η βασική διαφορά έγκειται στην επιλογή της ταχύτερης μεθόδου ή αυτής με τη μεγαλύτερη φορητότητα. Είναι ταχύτερο να πραγματοποιηθεί ένα δυαδικό αντίγραφο ασφαλείας, διότι αφορά μόνο λειτουργίες αντιγραφής αρχείων χωρίς να απαιτείται γνώση για την εσωτερική τους δομή. Ωστόσο, εάν το αντίγραφο ασφαλείας πρόκειται να χρησιμοποιηθεί για τη μεταφορά δεδομένων σε ένα άλλο μηχάνημα που

χρησιμοποιεί μια διαφορετική αρχιτεκτονική, τα αρχεία πρέπει να χαρακτηρίζονται από φορητότητα. Με την έννοια της φορητότητας εννοούμε ότι τα αρχεία είναι ανεξάρτητα από το μηχάνημα και ότι μπορούν να αντιγραφούν απευθείας από έναν εξυπηρετητή MySQL σε έναν άλλο, σε μια διαφορετική μηχανή και αυτή να είναι σε θέση και να έχει πρόσβαση στο περιεχόμενό τους χωρίς προβλήματα. Με τα δυαδικά αντίγραφα ασφαλείας χρειάζεται κατά την διάρκεια λήψης αντιγράφου ασφαλείας να μην τροποποιείται κάποιο αρχείο.

Η διαδικασία λήψης αντιγράφου ασφαλείας με κείμενο είναι βραδύτερη επειδή ο εξυπηρετητής πρέπει να διαβάσει τους πίνακες και στη συνέχεια είτε να γράψει το περιεχόμενο σε αρχεία στο δίσκο ή να στείλει τα περιεχόμενα σε ένα πρόγραμμα-πελάτη που προσπελαύνει τους πίνακες. Ένα τέτοιο παράδειγμα είναι το εργαλείο πελάτης mysqldump, με το οποίο λαμβάνεται ένα πίνακα περιεχομένων από τον εξυπηρετητή και στη συνέχεια γράφεται σε αρχείο ως δηλώσεις INSERT που μπορεί να φορτωθούν σε οποιοδήποτε πίνακα. Τα αντίγραφα ασφαλείας τύπου κειμένου είναι φορητά που σημαίνει ότι μπορούν να χρησιμοποιηθούν ανεξάρτητα της πλατφόρμας. Τέλος τα αντίγραφα ασφαλείας δυαδικού τύπου εξαρτώνται από τη μηχανή αποθήκευσης και χρησιμοποιούνται σε τοπικούς εξυπηρετητές. Σε αντίθεση τα αντίγραφα ασφαλείας τύπου κειμένου δεν εξαρτώνται από τη μηχανή αποθήκευσης και λαμβάνονται σε τοπικούς είτε απομακρυσμένους εξυπηρετητές.

14.6. Δημιουργία Δυαδικών αντιγράφων ασφαλείας για MyISAM

Για να δημιουργηθεί ένα δυαδικό αντίγραφο ασφαλείας πίνακα τύπου MyISAM αρκεί η αντιγραφή των αρχείων με κατάληξη .frm, .MYD, and .MYI τα οποία χρειάζονται για την αναπαράσταση κάθε πίνακα. Για να γίνει αυτό οι πίνακες δεν πρέπει να χρησιμοποιούνται από άλλο πρόγραμμα κατά την αντιγραφή. Έτσι συνήθως διακόπτεται η λειτουργία ακόμα και του εξυπηρετητή και πραγματοποιείται η αντιγραφή. Μια άλλη επιλογή είναι να κλειδωθούν οι πίνακες που θα αντιγραφούν. Για παράδειγμα για να αντιγραφεί ο πίνακας Country κλειδώνεται και εκτελούνται όλες οι εκκρεμείς εντολές:

```
USE world;  
LOCK TABLES Country READ;  
FLUSH TABLES Country;
```

Στην συνέχεια μετά την αντιγραφή πρέπει να ξεκλειδώσουμε τον πίνακα:

UNLOCK TABLES;

Αυτό λειτουργεί μόνο στα συστήματα βασισμένα σε Unix. Στα Windows πρέπει να σταματήσει ο εξυπηρετητής. Ένας άλλος τρόπος είναι να γίνει χρήση του *mysqlhotcopy*, το οποίο αναλαμβάνει τόσο το κλείδωμα όσο και την εκτέλεση των εντολών που εκκρεμούν αλλά ξεφεύγει των ορίων του παρόντος υλικού. Για να ανακτηθεί ο πίνακας από ένα αντίγραφο ασφαλείας δυαδικού τύπου αρκεί να αντιγραφούν τα αρχεία αφού έχει σταματήσει ο εξυπηρετητής.

14.7. Δημιουργία Δυαδικών αντιγράφων ασφαλείας για InnoDB

Το δυαδικό αντίγραφο ασφαλείας στην περίπτωση του InnoDB είναι ουσιαστικά η αντιγραφή όλων των αρχείων που χρησιμοποιεί η τύπου InnoDB βάση. Τα βήματα που ακολουθούνται είναι:

1. Σταμάτημα του εξυπηρετητή, χωρίς να έχουν εμφανιστεί λάθη,
2. Δημιουργία αντιγράφων ασφαλείας των:
 - a. .frm αρχείων για κάθε πίνακα InnoDB.
 - b. Των αρχείων του χώρου πινάκων (tablespace), συμπεριλαμβανομένων και των αρχείων του κοινού χώρου πινάκων καθώς και των αρχείων τύπου .ibd,
 - c. Των αρχείων καταγραφής συμβάντων,
 - d. Κάθε επιλογή ρύθμισης για επαναφορά του αντιγράφου από το μηδέν.
3. Επανεκκίνηση του εξυπηρετητή

Υπάρχει και η δυνατότητα μέσω InnoDB Hot Backup αλλά ξεφεύγει των ορίων του παρόντος υλικού.

Για την ανάκτηση ενός χώρου πινάκων τύπου InnoDB μέσω δυαδικού αντιγράφου ασφαλείας πρέπει να σταματήσει ο εξυπηρετητής, να αντικατασταθούν όλα τα στοιχεία όπως αυτά έχουν ληφθεί σε αντίγραφα και να επανεκκινήσει ο εξυπηρετητής.

14.8. Παράγοντες δυαδικής φορητότητας

Η δυαδική φορητότητα είναι χρήσιμη αν χρειάζεται να ληφθεί δυαδικό αντίγραφο

ασφαλείας που δημιουργήθηκε σε ένα μηχάνημα και να χρησιμοποιηθεί σε ένα άλλο μηχάνημα που έχει μια διαφορετική αρχιτεκτονική. Για παράδειγμα, χρησιμοποιώντας ένα δυαδικό αντίγραφο ασφαλείας ένας τρόπος είναι να αντιγραφούν οι βάσεις δεδομένων από τον ένα εξυπηρετητή MySQL στον άλλο. Στην περίπτωση των τύπου MyISAM βάσεων, η δυαδική φορητότητα είναι ότι μπορούν να αντιγραφούν κατευθείαν τα αρχεία για ένα πίνακα MyISAM από ένα εξυπηρετητή MySQL σε ένα άλλο μηχάνημα και ο εξυπηρετητής θα έχει πρόσβαση στα αρχεία. Για την περίπτωση InnoDB, η δυαδική φορητότητα σημαίνει ότι μπορούν να αντιγραφούν τα αρχεία πινάκων από ένα εξυπηρετητή MySQL σε ένα άλλο μηχάνημα και σε αυτό ο εξυπηρετητής MySQL να είναι σε θέση να έχει πρόσβαση στο tablespace. Εξ' ορισμού στην περίπτωση των InnoDB πινάκων γίνεται αποθήκευση όλων σε ένα tablespace. Έτσι φορητότητα του tablespace είναι συνάρτηση του κατά πόσον όλοι οι επιμέρους πίνακες InnoDB είναι φορητοί. Οι πίνακες τύπου MyISAM και InnoDB είναι δυαδικά φορητοί αν πληρούνται οι όροι:

- Τα δύο μηχανήματα χρησιμοποιούν αριθμητική ακεραίων συμπληρώματος δύο,
- Τα δύο μηχανήματα χρησιμοποιούν IEEE μορφή κινητής υποδιαστολής, ή οι πίνακες περιέχουν μη κινητής υποδιαστολής στήλες.

Μια τρίτη συνθήκη είναι ότι για την InnoDB δυαδική φορητότητα πρέπει να χρησιμοποιούνται ονόματα με μικρά γράμματα για τις βάσεις και τους πίνακες. Γενικά αποτελεί καλή πρακτική να χρησιμοποιούνται ονόματα συνεχόμενα και με μικρά λατινικά γράμματα που όμως έχουν κάποια έννοια σαν λέξεις για μας.

Μια εναλλακτική λύση για να κάνει ένα δυαδικό αντίγραφο ασφαλείας φορητό είναι να γίνει ο πίνακας σε μορφή περιεχομένων κειμένου (για παράδειγμα, με mysqldump). Αυτή η τεχνική μπορεί να είναι χρήσιμη για την αντιγραφή επιμέρους πινάκων InnoDB από ένα διακομιστή στον άλλο ή εάν δεν πληρούνται οι όροι φορητότητας.

14.9. Αντίγραφα ασφαλείας InnoDB σε λειτουργία

Το πρόγραμμα αντιγράφων ασφαλείας InnoDB σε λειτουργία (Hot Backup program - ibbackup) είναι ένα εμπορικό προϊόν διαθέσιμο μόνο για βάσεις δεδομένων τύπου InnoDB. Μπορεί να λάβει αντίγραφα ασφαλείας ενώ λειτουργεί ο εξυπηρετητής χωρίς να επηρεάσει την κανονική λειτουργία του και της βάσης. Είναι διαθέσιμο σε εκδόσεις

Unix και Windows.

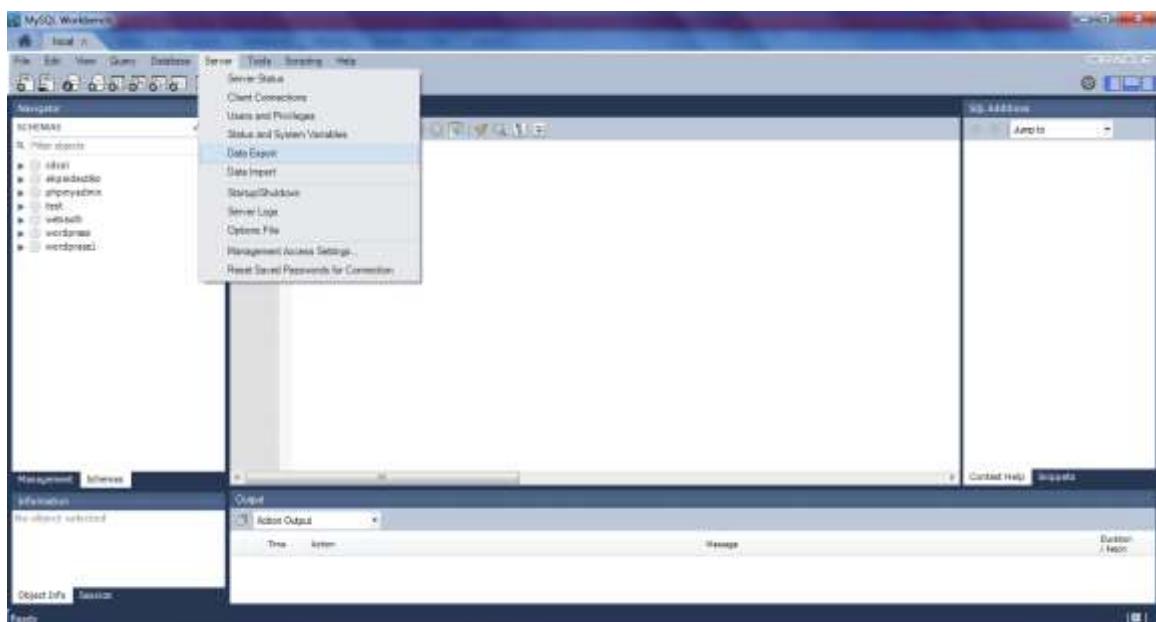
14.10. Ανάκτηση βάσης δεδομένων (Restore Database)

Γενικά, όπως αναφέραμε και προηγούμενα για να επανέλθει μια βάση δεδομένων σε κατάσταση πριν κάποια καταστροφή απαιτούνται τα αρχεία αντιγράφων ασφαλείας (**backup files**) και τα αρχεία καταγραφή (**binary logs**). Έτσι πρέπει να εκτελεστούν τα εξής βήματα:

- Ανάκτηση (restore) των πινάκων στην κατάσταση που ήταν έως την στιγμή του τελευταίου αντιγράφου ασφαλείας (**backup**) από τα αρχεία αντιγράφων ασφαλείας (**backup files**),
- Ανάκτηση από τα αρχεία καταγραφής (binary logs) των εντολών που πραγματοποιήθηκαν και τροποποίησαν τη βάση, μεταξύ της στιγμής αντιγράφου ασφαλείας και την στιγμή της ανάκτησης.

14.11. Εξαγωγή Δεδομένων

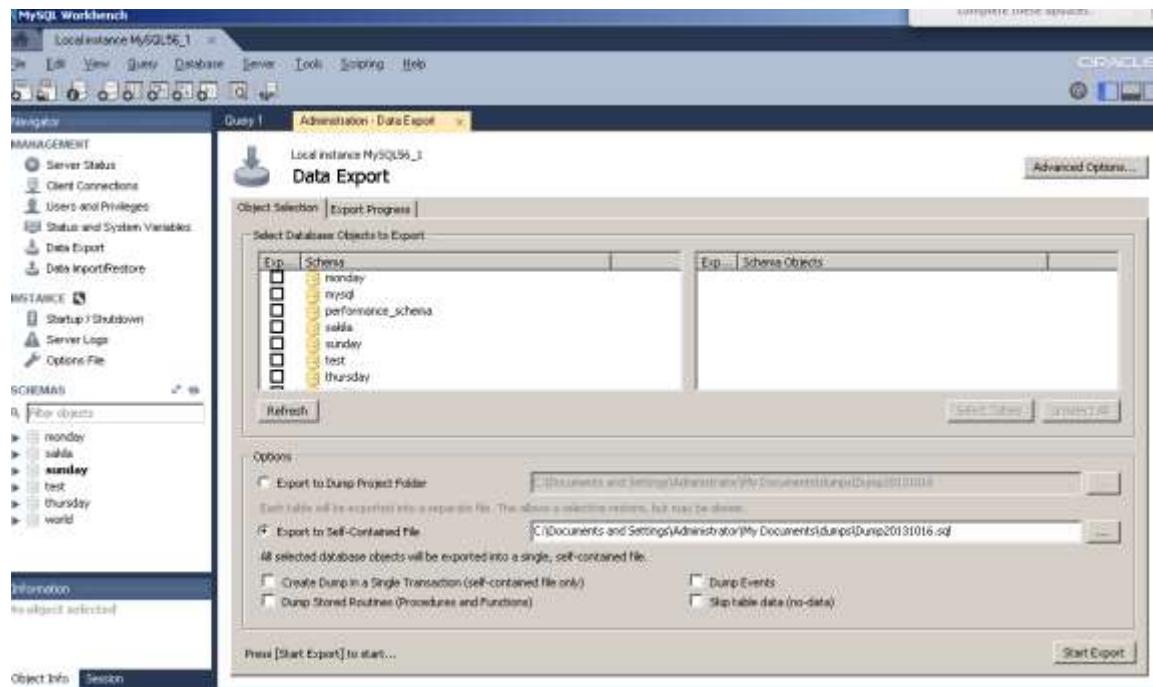
Η εξαγωγή δεδομένων μιας βάσης MySQL μπορεί να πραγματοποιηθεί μέσω του MySQL Workbench από την επιλογή Server -> Data Export Εικόνα 113 Εξαγωγή Δεδομένων.



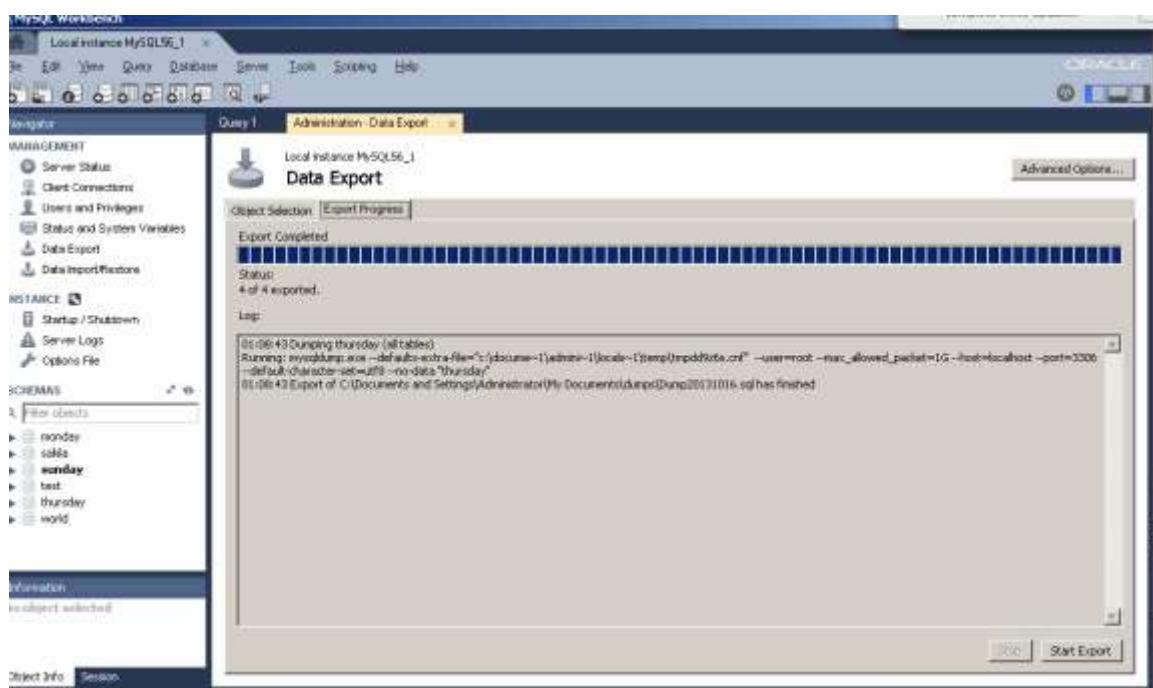
Εικόνα 113 Εξαγωγή Δεδομένων

Στη συνέχεια θα έλθουμε στην οθόνη της Εικόνα 113 Εξαγωγή Δεδομένων που

επιλέγουμε export to Self contained file και πατάμε Start Export και έχουμε μια οθόνη της μορφής της Εικόνα 115 Εκτέλεση εξαγωγής δεδομένων. Παρόμοια είναι η αντιμετώπιση αν χρησιμοποιήσουμε την επιλογή Export to Dump Project Folder μόνο που σε αυτή την περίπτωση θα δημιουργηθεί ένας φάκελος και κάθε πίνακας θα αποθηκευτεί σε αυτόν με την μορφή ξεχωριστού αρχείου αντί ενιαίου.



Εικόνα 114 Εξαγωγή δεδομένων



Εικόνα 115 Εκτέλεση εξαγωγής δεδομένων

Δραστηριότητα

Χρησιμοποιώντας το MySql Workbench δημιουργήστε δύο αντίγραφα ασφαλείας για την βάση δεδομένων Thursday

- ένα Dump με την μορφή αρχείου (file name:thursday_wb.sql)
- ένα Dump σε folder (folder name : thursday_wb_dump20131024)

Εξαγωγή δεδομένων όμως μπορεί να επιτευχθεί και μέσω της εντολής SELECT ... INTO OUTFILE

Η πλήρης σύνταξη της εντολής είναι:

```
SELECT [fields] INTO OUTFILE 'file_name' FROM tbl_name;
```

Το όνομα του αρχείου είναι η θέση στο δίσκο που θα αποθηκευτούν τα δεδομένα. Στην περίπτωση του SELECT ... INTO OUTFILE τα αποτελέσματα δεν βγαίνουν ποτέ από τον εξυπηρετητή και το αρχείο γράφεται πάντα στον εξυπηρετητή. Για λόγους ασφαλείας και αποφυγής σφαλμάτων η εντολή απαιτεί να μην υπάρχει αρχείο με ίδιο όνομα. Για την ανάκτηση του αρχείου θα πρέπει κάποιος να έχει δικαιώματα (τουλάχιστον FILE) στον εξυπηρετητή, κύριος όμως του αρχείου είναι ο διαχειριστής της βάσης. Το εξαγόμενο αρχείο γράφει μια γραμμή για κάθε εγγραφή που επιστρέφει η εντολή select, κάθε στήλη χωρίζεται με tab και οι γραμμές χωρίζονται με newline.

Παράδειγμα

```
SELECT * INTO OUTFILE 'C:\load_data.txt' FROM employees_tbl;
```

Δραστηριότητα

Δημιουργήστε ένα csv file με όνομα, επώνυμο, ηλικία, μισθό που θα προέρχεται από τον πίνακα employees_tbl στην βάση δεδομένων Thursday.

14.12. Εξαγωγή δεδομένων από γραμμή εντολών

Το πρόγραμμα πελάτη mysqldump εξάγει τα δεδομένα ενός πίνακα σε αρχείο. Μπορεί να εξάγει τους πίνακες σε αρχεία tab-delimited ή να παράγει αρχεία τα οποία περιέχουν εντολές SQL τύπου create table και insert.

Η βασική δομή εντολής είναι:

```
mysqldump - -tab=dir_name options db_name tbl_name;
```

`db_name` το όνομα της βάσης δεδομένων που περιέχει τον πίνακα του οποίου πρόκειται να εξαχθούν τα δεδομένα.

`tbl_name` το όνομα του πίνακα που θα εξαχθεί. Για να εξαχθούν πολλαπλοί πίνακες απλά γράφουμε όλους τους πίνακες, αν δεν βάλουμε όνομα πίνακα τότε γίνεται εξαγωγή όλων των πινάκων της βάσης.

Το τμήμα των επιλογών της εντολής `mysqldump` μπορεί να περιλαμβάνει παραμέτρους σύνδεσης όπως `--host` ή `--user`. Πρέπει να παρασχεθούν αυτές οι πληροφορίες αν οι εξ ορισμού δεν είναι κατάλληλες. Επίσης υπάρχουν επιλογές για την χρήση της ίδιας της εντολής οι οποίες μπορούν να εμφανιστούν αν εκτελέσουμε την εντολή με την παράμετρο `-help`.

Παράδειγμα

Θέλουμε να εξάγουμε με χρήση της εντολής γραμμής `mysqldump` τα δεδομένα από ένα πίνακα με όνομα `City` της βάσης `world` χρησιμοποιώντας το φάκελο `/tmp`

```
mysqldump - -tab=/tmp world City;
```

Η έξοδος αποτελείται από ένα αρχείο `City.sql` που περιέχει την εντολή `create table` και ένα `City.txt` το οποίο περιέχει τα δεδομένα του πίνακα. Για να ανακτηθούν τα δεδομένα από αυτά τα αρχεία θα πρέπει πρώτα να πάμε στο φάκελο που έχει τα αρχεία μετά να γίνει χρήση της εντολής `mysql` για να χρησιμοποιηθεί το `.sql` και μετά να γίνει χρήση της εντολής `mysqldump`.

```
cd /tmp  
mysql world < City.sql;  
mysqldump world City.txt;
```

Εάν ένα αρχείο εξόδου δεδομένων έχει την προκαθορισμένη δομή εξαγωγής όπως περιγράφηκε παραπάνω, περιέχει γραμμές που τελειώνουν με νέα γραμμή και κάθε γραμμή έχει τιμές που χωρίζονται με `tab`, δεν απαιτείται κάποια επιπλέον παράμετρος για να εξαχθεί. Στην περίπτωση όμως αρχείων με άλλη δομή πρέπει να γίνει χρήση των παραμέτρων

- `--lines-terminated-by=string`
- `--fields-terminated-by=string`

- --fields-enclosed-by=char or --fields-optionally-enclosed-by=char
- --fields-escaped-by=char

Με αυτές τις παραμέτρους παρέχεται μεγάλη ευχέρεια τόσο για να εξαχθούν τα δεδομένα μιας βάσης σε αρχεία με κάθε δομή, όσο και να εισαχθούν δεδομένα από αρχεία κάθε δομής, όπως θα δούμε στη συνέχεια.

Παράδειγμα

Η ακόλουθη εντολή εξάγει ένα αρχείο στο οποίο οι γραμμές τελειώνουν με carriage return/linefeed:

```
mysqldump --lines-terminated-by="\\r\\n" world City.txt;
```

Οι επιλογές --all --databases και --databases χρησιμοποιούνται για την εξαγωγή πολλάπλων βάσεων δεδομένων και δεν μπορούν να χρησιμοποιηθούν σε συνδυασμό με την -- tab

14.13. Εισαγωγή Δεδομένων

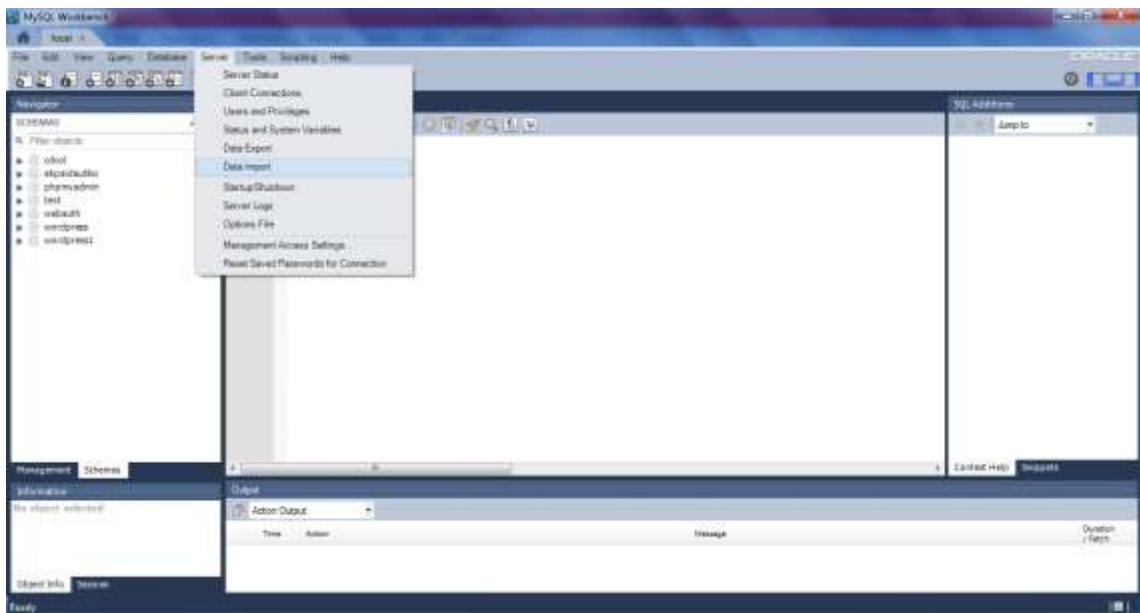
Για την εισαγωγή δεδομένων έχουμε τρεις δυνατότητες:

Πρώτη δυνατότητα με γραμμή εντολών, ανάλογα με αυτό που δείξαμε και προηγούμενα, και με χρήση της εντολής

```
mysql -u [uname] -p < file_script.sql;
```

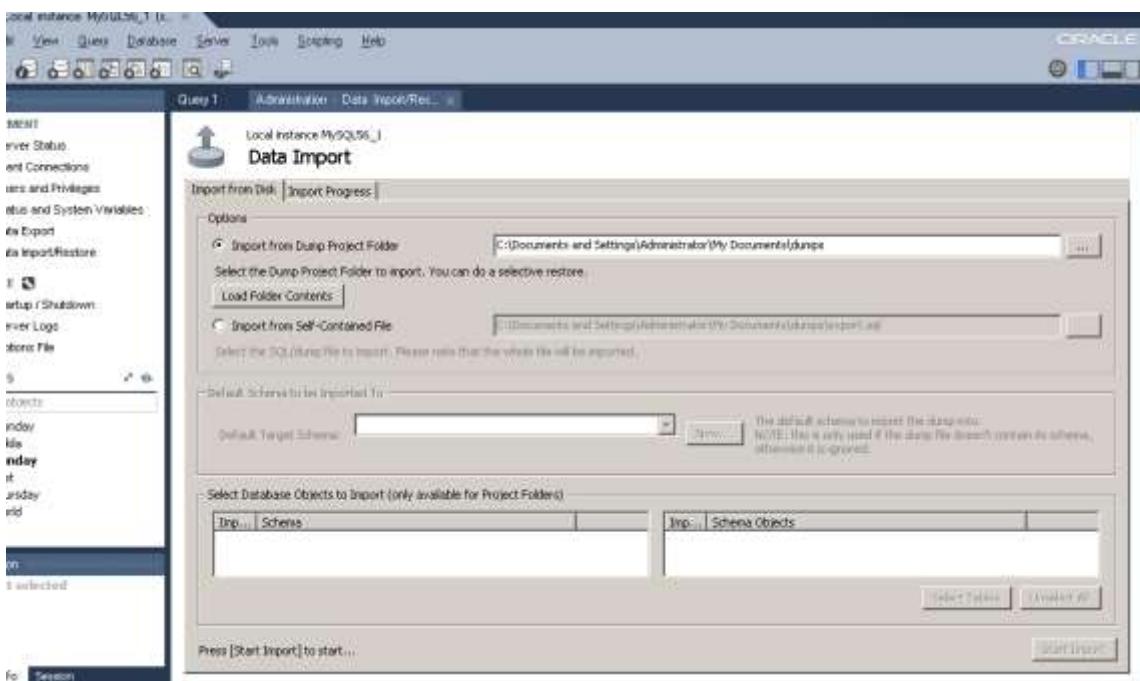
όπου το αρχείο file_script.sql περιέχει τα δεδομένα που θέλουμε να εισάγουμε σε μορφή αρχείου .sql

Δεύτερη είναι με χρήση της Load Data Infile που θα αναλύσουμε πιο κάτω. Η Τρίτη είναι η πιο εύκολη μέθοδος με χρήση της επιλογής Data Import του Workbench (Εικόνα 116 Εισαγωγή δεδομένων)

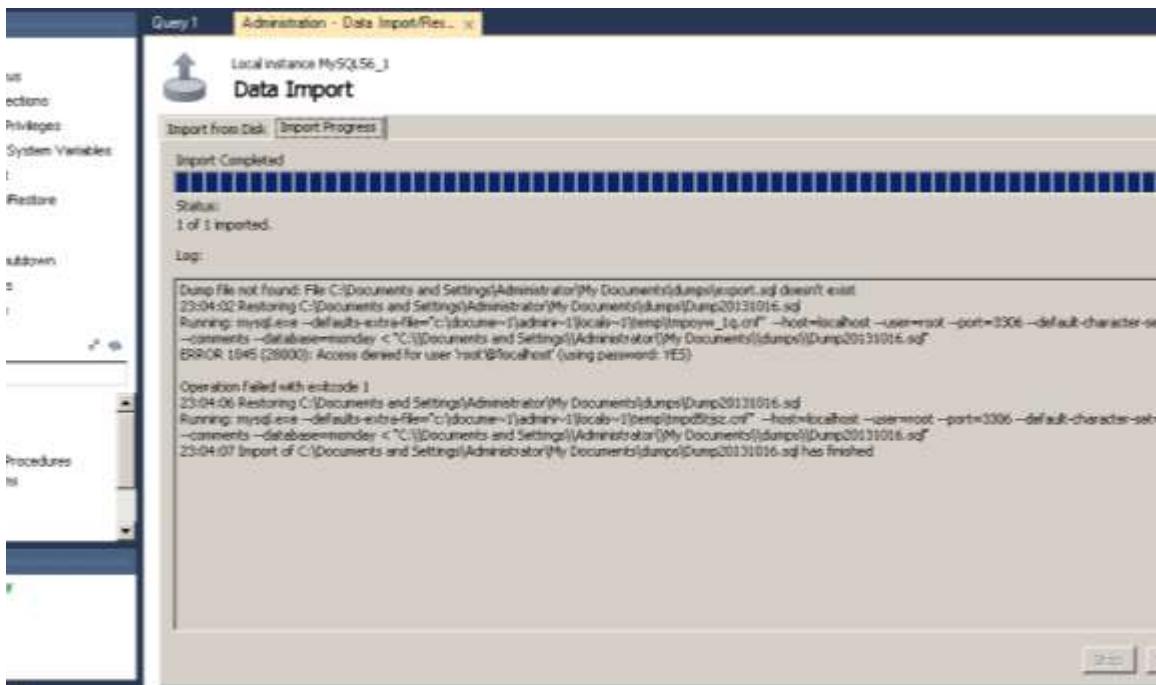


Εικόνα 116 Εισαγωγή δεδομένων

Στη συνέχεια θα έλθουμε στην οθόνη της Εικόνα 117 Εισαγωγή δεδομένων που επιλέγουμε import from Self contained file και πατάμε Start Import και έχουμε μια οθόνη της μορφής της Εικόνα 118 Εκτέλεση εισαγωγής δεδομένων Παρόμοια είναι η αντιμετώπιση αν χρησιμοποιήσουμε την επιλογή Import from Dump Project Folder μόνο που σε αυτή την περίπτωση θα χρησιμοποιήσουμε ένα φάκελο και κάθε πίνακας θα έχει αποθηκευτεί σε αυτόν με την μορφή ξεχωριστού αρχείου αντί ενιαίου.



Εικόνα 117 Εισαγωγή δεδομένων



Εικόνα 118 Εκτέλεση εισαγωγής δεδομένων

14.14. Εισαγωγή δεδομένων (import data) από αρχείο

Η εντολή LOAD DATA INFILE παρέχει έναν εναλλακτικό τρόπο για εισαγωγή νέων δεδομένων σε ένα πίνακα, η διαφορά είναι ότι η INSERT τις τιμές που εισάγει τις έχει στην εντολή ενώ η LOAD DATA INFILE τις διαβάζει από αρχείο. Η πλήρης σύνταξη της εντολής είναι:

```

LOAD DATA [LOCAL] INFILE 'file_name'
INTO TABLE tbl_name
[FIELDS [TERMINATED BY 'string']]
[LINES
[STARTING BY 'string']
[TERMINATED BY 'string']]
]
[IGNORE number LINES]
[(col_name_or_user_var,...)]
[SET col_name = expr,...];

```

Τα ελάχιστα αναγκαία πεδία είναι

```

LOAD DATA [LOCAL] INFILE 'file_name'
INTO TABLE tbl_name;

```

Το αρχείο είναι ένα string σε quotes. Στα windows τα διαχωριστικά στα pathnames

πρέπει να γραφούν είτε ως '/' ή '\\'. Για να φορτωθεί ένα αρχείο με το όνομα π.χ. C:\mydata\data.txt, μπορούν να εκτελεστούν ένα από τα δύο ακόλουθα:

LOAD DATA INFILE 'C:/mydata/data.txt' into table t;

ή

LOAD DATA INFILE 'C:\\mydata\\data.txt' into table t;

Η MySQL υποθέτει ότι το αρχείο βρίσκεται στον εξυπηρετητή και έχει την απαιτούμενη μορφή (οι στήλες χωρίζονται από tab και κάθε γραμμή τελειώνει σε newline) και ότι κάθε γραμμή περιέχει δεδομένα. Η εντολή έχει όλες τις άλλες μη υποχρεωτικές παραμέτρους για να διαχειριστεί τις εξαιρέσεις αυτών των περιπτώσεων. Για παράδειγμα η παράμετρος LOCAL χρησιμοποιείται για να διαβάσουμε αρχείο που βρίσκεται στον χρήστη και όχι στον εξυπηρετητή.

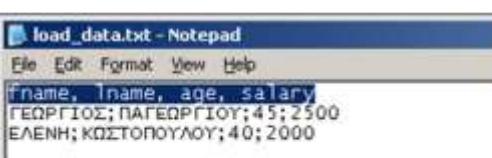
Παράδειγμα

```
LOAD DATA LOCAL INFILE 'C:\\load_data.txt' into
table employees_tbl FIELDS TERMINATED BY ','

LINES TERMINATED BY '\\n' ignore 1 lines
(@fname,@lname,@age,@salary) set age=@age,
fname=@fname, lname=@lname, salary=@salary;
```

Στο τέλος της εκτέλεσης της εντολής εμφανίζεται μήνυμα όπως αυτό της Εικόνας 119 Load data όπου μας δίνει τις εγγραφές που διαβάστηκαν από το αρχείο, οι εγγραφές του πίνακα που διαγράφηκαν (ανανεώθηκαν) από τις εγγραφές του αρχείου που είχαν το ίδιο πρωτεύον κλειδί, τις εγγραφές που αγνοήθηκαν γιατί είχαν κλειδί που ήδη υπήρχε στον πίνακα, και τέλος τα προβλήματα που βρέθηκαν στο αρχείο εισόδου, όπως ανύπαρκτα δεδομένα.

```
MySQL 5.6 Command Line Client - Unicode
@salary) set age=@age, fname=@fname, lname=@lname, salary=@salary;
Query OK, 2 rows affected, 1 warning (0.00 sec)
Records: 2  Deleted: 0  Skipped: 0  Warnings: 1
mysql>
```



	id	fname	lname	age	salary
	37	ΓΕΩΡΓΙΟΣ	ΠΑΓΕΩΡΓΙΟΥ	45	2500
	38	ΕΛΕΝΗ	ΚΩΣΤΟΠΟΥΛΟΥ	40	2000

Για να εκτελέσουμε την εντολή αυτή αρκεί να έχουμε δικαιώματα insert και delete, ενώ χρειάζονται πλήρη δικαιώματα ως προς τον διακομιστή που βρίσκεται το αρχείο. Τέλος η εντολή είναι αποτελεσματικότερη από το insert είτε αυτή τρέχει για αρχείο στον εξυπηρετητή είτε σε κάποιο πελάτη γιατί είναι μικρότερος ο φόρτος για την φόρτωση και την ανάλυση των δεδομένων, η φόρτωση γίνεται με απλή εκτέλεση.

14.15. Εισαγωγή δεδομένων από γραμμή εντολών

Το πρόγραμμα πελάτη msqlexport φορτώνει δεδομένα σε πίνακες. Παρέχει μια διεπαφή γραμμής εντολών στο LOAD DATA INFILE. Κάθε εντολή msqlexport ελέγχει τις παραμέτρους που δίνονται σε αυτή στην γραμμή εντολών, συνδέεται στον εξυπηρετητή και για κάθε αρχείο που υπάρχει στην εντολή εκτελεί μια LOAD DATA INFILE οποία φορτώνει τα στοιχεία στον κατάλληλο πίνακα. Η εντολή msqlexport είναι πολύ χρηστική και τρέχει και από πρόγραμμα πελάτη, όχι μόνο από εξυπηρετητή.

Η βασική δομή εντολής είναι

```
mysqlexport options db_name input_file;
```

Η msqlexport έχει διάφορες παραμέτρους.

db_name όνομα της βάσης δεδομένων που περιέχει τον πίνακα που πρόκειται να φορτωθεί, **input_file** το όνομα του αρχείου που περιέχει τα δεδομένα που πρόκειται να φορτωθεί. Μπορούν να τοποθετηθούν διάφορα ονόματα αρχείων εισόδου μετά τη δήλωση της βάσης δεδομένων. Η msqlexport χρησιμοποιεί ένα όνομα αρχείου για να καθορίσει το όνομα του αντίστοιχου πίνακα στο οποίο θα πρέπει να φορτωθούν τα περιεχόμενα του αρχείου. Το πρόγραμμα το κάνει αυτό με την απομάκρυνση κάθε επέκτασης αρχείου και της τελείας, χρησιμοποιώντας το αποτέλεσμα ως όνομα πίνακα.

Παράδειγμα

Θέλουμε να εισάγουμε με χρήση της εντολής γραμμής msqlexport τα δεδομένα από ένα αρχείο με όνομα City.txt ή City.dat ως στοιχεία εισόδου για να φορτωθούν σε ένα πίνακα με το όνομα City. Αφού προσδιορίστει το όνομα του πίνακα με βάση το όνομα του αρχείου, η msqlexport εκτελεί μια εντολή LOAD DATA INFILE για να φορτώσει

το αρχείο σε ένα πίνακα.

Κάθε πίνακας για να φορτωθεί με στοιχεία με χρήση της mysqlimport πρέπει να προϋπάρχει και κάθε αρχείο πρέπει να περιέχει μόνο τιμές δεδομένων. Η mysqlimport δεν χρησιμοποιείται για να διαχειρίζεται αρχεία που έχουν εξαχθεί από διαδικασία dump και αποτελούνται από SQL ερωτήματα.

Το τμήμα των επιλογών της εντολής mysqlimport μπορεί να περιλαμβάνει παραμέτρους σύνδεσης όπως --host ή --user. Πρέπει να παρασχεθούν αυτές οι πληροφορίες αν οι εξ ορισμού δεν είναι κατάλληλες. Επίσης υπάρχουν επιλογές για την χρήση της ίδιας της εντολής οι οποίες μπορούν να εμφανιστούν αν εκτελέσουμε την εντολή με την παράμετρο –help.

Εάν ένα αρχείο εισόδου δεδομένων έχει την προκαθορισμένη δομή εξαγωγής, δηλαδή περιέχει γραμμές που τελειώνουν με νέα γραμμή και κάθε γραμμή έχει τιμές που χωρίζονται με tab, δεν απαιτείται κάποια επιπλέον παράμετρος για να εισαχθεί. Στην περίπτωση όμως αρχείων με άλλη δομή πρέπει να γίνει χρήση των παραμέτρων

- --lines-terminated-by=string
- --fields-terminated-by=string
- --fields-enclosed-by=char or --fields-optionally-enclosed-by=char
- --fields-escaped-by=char

Με αυτές τις παραμέτρους παρέχεται μεγάλη ευχέρεια να εισαχθούν δεδομένα από αρχεία κάθε δομής.

Παράδειγμα

Η ακόλουθη εντολή φορτώνει ένα αρχείο στο οποίο οι γραμμές τελειώνουν με carriage return/linefeed:

```
mysqlimport --lines-terminated-by="\\r\\n" world City.txt;
```

Άλλες επιλογές είναι οι ακόλουθες:

- --ignore or --replace
- --local

14.16. Ανάκτηση μεταδεδομένων του εξυπηρετητή MySQL

Οι βάσεις δεδομένων περιέχουν δεδομένα, αλλά οι πληροφορίες σχετικά με τον τρόπο

που είναι δομημένες οι βάσεις είναι τα μεταδεδομένα. Η MySQL παρέχει πρόσβαση σε μεταδεδομένα για τη βάση δεδομένων, τους πίνακες, και τα άλλα αντικείμενα.

Σε αυτό το σημείο θα δούμε:

- Μεθόδους πρόσβασης στα Μεταδεδομένα
- Χρήση της βάσης information_schema για πρόσβαση στα μεταδεδομένα
- Χρήση της SHOW και DESCRIBE για πρόσβαση στα μεταδεδομένα
- Χρήση της mysqlshow για πρόσβαση στα μεταδεδομένα

14.16.1. Μέθοδος Information_SCHEMA

Η μέθοδος INFORMATION_SCHEMA είναι ένα κεντρικό σύστημα αποθήκευσης των μεταδεδομένων. Εμφανίζεται από την έκδοση 5 και μετά σαν μια εξέλιξη των εντολών SHOW (είναι ο εναλλακτικός τρόπος που θα αναφερθεί παρακάτω). Η μέθοδος INFORMATION_SCHEMA είναι προσαρμοσμένη περισσότερο στην SQL παρά σε μια ειδική υλοποίηση της MySQL, όπως η SHOW.

Η INFORMATION_SCHEMA είναι μια εικονική βάση δεδομένων η οποία δεν είναι αποθηκευμένη στον δίσκο αλλά περιέχει πίνακες όπως κάθε άλλη βάση και τα δεδομένα αυτών μπορούν να ανακτηθούν με χρήση εντολών SELECT.

Παράδειγμα

Για να λάβουμε μια λίστα από όλους τους πίνακες που περιέχονται στο σχήμα INFORMATION_SCHEMA αρκεί να εκτελεστεί το παρακάτω:

```
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES  
WHERE TABLE_SCHEMA = 'INFORMATION_SCHEMA'  
ORDER BY TABLE_NAME;
```

Το αποτέλεσμα της παραπάνω εντολής είναι στην Εικόνα 120 Αποτελέσματα

```

MariaDB [test] > SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
-> WHERE TABLE_SCHEMA = 'INFORMATION_SCHEMA'
-> ORDER BY TABLE_NAME;
+-----+
| TABLE_NAME |
+-----+
| CHARACTER_SETS
| COLLATIONS
| COLUMN_CHARACTER_SET_APPLICABILITY
| COLUMNS
| COLUMN_PRIVILEGES
| ENGINES
| EVENTS
| FILES
| GLOBAL_STATUS
| GLOBAL_VARIABLES
| INNODB_BUFFER_PAGE
| INNODB_BUFFER_PAGE_LRU
| INNODB_BUFFER_POOL_STATE
| INNODB_CNF
| INNODB_COPROM
| INNODB_COPROM_ASSET
| INNODB_CNF_RESET
| INNODB_DURABILITY
| INNODB_LOCK_UNITS
| INNODB_TEN
| KEY_COLUMN_USAGE
| PARTITIONS
| PARTITIONS
| PLUGINS
| REFERENTIAL_CONSTRAINTS
| ROUTINES
| SCHEMATA
| SCHEMA_PRIVILEGES
| SESSION_STATUS
| STATUS_VARIABLES
| STATISTICS
| TABLES
| TABLESPACES
| TABLE_CONSTRAINTS
| TABLE_PRIVILEGES
| TRIGGERS
| USER_PRIVILEGES
| VIEWS
+-----+
0 rows in set (0.00 sec)

```

Εικόνα 120 Αποτελέσματα

Για να δούμε τα ονόματα των στηλών του πίνακα CHARACTER_SETS του σχήματος INFORMATION_SCHEMA εκτελούμε το παρακάτω

```

SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = 'INFORMATION_SCHEMA'
AND TABLE_NAME = 'CHARACTER_SETS';

```

Το αποτέλεσμα της παραπάνω εντολής είναι στην παρακάτω εικόνα.

```

MariaDB [(none)] > SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS
-> WHERE TABLE_SCHEMA = 'INFORMATION_SCHEMA'
-> AND TABLE_NAME = 'CHARACTER_SETS';
+-----+
| COLUMN_NAME |
+-----+
| CHARACTER_SET_NAME
| DEFAULT_COLLATE_NAME
| DESCRIPTION
| MAXLEN
+-----+
4 rows in set (0.00 sec)

```

Εικόνα 121 Αποτελέσματα

Οι μέχρι τώρα εντολές δίνουν μεταδεδομένα για το σχήμα μεταδεδομένων της βάσης.
Μπορούμε να ανακτήσουμε μεταδεδομένα και για τα άλλα σχήματα της βάσης

Παράδειγμα

Για παράδειγμα για να δούμε τα μεταδεδομένα ενός σχήματος με όνομα inedb

```
SELECT TABLE_NAME, TABLE_ROWS, TABLE_COLLATION
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'inepdb';
```

Το αποτέλεσμα φαίνεται στην Εικόνα 122 Αποτελέσματα SCHEMA

TABLE_NAME	TABLE_ROWS	TABLE_COLLATION
departments	5	utf8_general_ci
directorate	5	utf8_general_ci
employees	5	utf8_general_ci
employees_1	5	utf8_general_ci
insured_details	4	utf8_general_ci
insured_tha	6	utf8_general_ci

Εικόνα 122 Αποτελέσματα SCHEMA

14.16.2. Ανάκτηση Metadata με χρήση εντολών Show

Με την χρήση της εντολής show μπορούμε να δούμε διάφορες πληροφορίες σχετικά με τα μεταδεδομένα της βάσης. Έτσι:

- Για να δούμε τις βάσεις/σχήματα έχουμε την εντολή
show databases;
- Για να δούμε όλους τους πίνακες μιας βάσης/σχήματα
show tables;
- Για να δούμε όλες τις όψεις μιας βάσης/σχήματα
show views;
- Αν δεν έχουμε ορίσει προηγουμένως για ποια βάσης/σχήμα εκτελούμε την εντολή ανάκτησης μεταδεδομένων τότε το καθορίζουμε στην εντολή
show tables [FROM database_name];
- Για να δούμε τις στήλες ενός πίνακα:
show columns from table_name; ή
show fields from table_name; ή
describe table_name;
- Για να δούμε τα δυαδικά αρχεία καταγραφής (binary logs)
SHOW BINARY LOGS;
- Για να δούμε τι περιέχει ένα binary log π.χ. filename-bin.00001 log
SHOW BINLOG EVENTS IN 'filename-bin.00001';

- Για να δούμε όλα τα διαθέσιμα character sets
SHOW CHARACTER SET;
- Για να δούμε την λίστα με τα collations που υποστηρίζονται από τον server
SHOW COLLATION;
- Για να δούμε τα warnings, errors
SHOW WARNINGS;
SHOW ERRORS
- Για να δούμε τα variables του server
SHOW VARIABLES;
SHOW VARIABLES LIKE '%log%';
- Για να δούμε τα δικαιώματα ενός χρήστη (privileges)
SHOW GRANTS [FOR user]
SHOW GRANTS FOR 'albert'@'%';
- Για να δούμε την λίστα με όλα τα δικαιώματα
SHOW PRIVILEGES;
- Για να δούμε το status του server
SHOW STATUS;
- Για να δούμε την κατάσταση των πινάκων μιας βάσης δεδομένων
SHOW TABLE STATUS FROM database_name
- Για να δούμε τους χρήστες
USE MYSQL;
SELECT USER, HOST FROM MYSQL.USER;

14.16.3. Διαφορά μεταξύ INFORMATION_SCHEMA Και SHOW

Για τις περισσότερες περιπτώσεις η εντολή INFORMATION_SCHEMA και η SHOW έχουν το ίδιο αποτέλεσμα στο ποια μεταδεδομένα θα επιστρέψουν. Ορισμένες φορές όμως κάποια από τις δύο είναι προτιμότερη. Έτσι τα πλεονεκτήματα της INFORMATION_SCHEMA έναντι της SHOW είναι:

1. INFORMATION_SCHEMA είναι βασισμένη σε standard SQL, ενώ η SHOW όχι,
2. Με την INFORMATION_SCHEMA χρειάζεται μόνο χρήση της εντολής SELECT για να πάρεις τα μεταδεδομένα άσχετα από το τι ζητάς. Στη SHOW

έχεις διαφορετική εντολή και σύνταξη κάθε φορά ανάλογα με τα μεταδεδομένα που αναζητούνται,

3. Με την INFORMATION_SCHEMA υπάρχει πλήρης ελευθερία στο τι θα ανακτηθεί και πολλές επιλογές περιορισμού των ανακτόμενων μεταδεδομένων. Στον αντίποδα η SHOW δεν είναι τόσο ευέλικτη αν και υποστηρίζει την παράμετρο LIKE και μετά την έκδοση MySQL 5 έχει προστεθεί και η παράμετρος WHERE. Σε κάθε περίπτωση δεν υποστηρίζεται η ORDER και ο περιορισμός των εμφανιζόμενων στηλών,
4. Με την INFORMATION_SCHEMA μπορεί να γίνει χρήση όλων των γνωστών τρόπων μέσω joins, unions, και subqueries,
5. Με τη χρήση CREATE TABLE ... SELECT or INSERT ... SELECT, τα δεδομένα της INFORMATION_SCHEMA μπορούν να ανακτηθούν και να αποθηκευτούν σε έναν άλλο πίνακα σε αντίθεση με τη SHOW που είναι μόνο για εμφάνιση.

14.16.4. Τα πλεονεκτήματα της SHOW έναντι της INFORMATION_SCHEMA είναι:

1. Η SHOW είναι διαθέσιμη σε όλες τις εκδόσεις προ της MySQL 5,
2. Η SHOW είναι συνήθως πιο σύντομη, για παράδειγμα οι δύο εντολές παρακάτω δίνουν το ίδιο αποτέλεσμα,
3. Οι μεταβλητές της SHOW δείχνουν μεταβλητές συστήματος.

```
SHOW TABLES FROM inepdb;
```

```
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES  
WHERE TABLE_SCHEMA = 'inepdb';
```

15. Ευρετήριο Όρων

- ACID, 113
alias, 134, 135
ALL, 189
ANALYZE TABLE, 240
AUTO_INCREMENT, 28, 102
AUTOCOMMIT, 113
backup, 251
command-line, 49
COMMIT, 106, 112
constraint, 29
CREATE VIEW, 204
CROSS JOIN, 121
Data Control Language, 40
Data Definition Language, 40
Data Manipulation Language, 40
datetime, 67
decimal, 66
DELETE, 100
DISTINCT, 80
EXISTS, 192
foreign key, 30
Forward Engineering, 38
full outer join, 146
Functions, 45
GRANT, 222, 227, 234
group by, 153
HAVING, 154, 182
Hot Backup, 255
IN, 191
InnoDB, 44, 59
INSERT, 85
integer, 66
IS-A, 21
JOIN, 196
left join, 142
LOWER, 158
mysqldump, 258
mysqlimport, 264
MyISAM, 44, 59
MySQL Community Edition, 43
MySQL Connectors, 56
MySQL Enterprise Monitor, 53
MySQL Workbench, 55
mysql.exe, 49
nonequijoin, 140
nonrepeatable read, 115
NULL, 17, 86, 173
OPTIMIZE TABLE, 242
OUTER JOIN, 140
Partitioning, 45
Phantom read, 116
primary key, 73
referential integrity constraint, 29
REPAIR TABLE, 250
REPLACE, 93
Replication, 44
restore, 256
reverse engineering, 40
right join, 144
ROLLBACK, 106
SAVEPOINT, 108
SELECT, 78
self join, 136
Serializable, 117
storage engines, 44
Stored Procedures, 45
TRANSACTION, 105
Triggers, 45
TRUNCATE, 102
UPPER, 158
varchar, 66
Views, 45
WHERE, 80, 182
WorkBench, 26
Αθροισμα, 150
Αναγνωριστικά, 63
αναδρομική συσχέτιση, 30
αναφορική ακεραιότητα, 73
Ανθεκτικότητα, 114
αντίγραφο ασφαλείας, 251
αντικατάσταση, 158
αντιστροφή, 159
Απομόνωση, 114
Ατομικότητα, 113
βαθμός σχέσης, 25
Βάση Δεδομένων, 10
γενίκευση, 22
γλώσσα ελέγχου δεδομένων, 40
γλώσσα ορισμού δεδομένων, 40
γλώσσα χειρισμού δεδομένων, 40, 41
γνώρισμα, 17, 25
γραμμή εντολών, 49
Δημιουργία Βάσης Δεδομένων, 64
δημιουργία πίνακα, 68
διαγραφή, 100
διαγραφή εγγραφών, 102
Διάδοση Διαγραφής, 102
διαφορά, 173, 177
διαχείριση λογαριασμών, 210
δικαιώματα, 224
EER Diagram, 26
εισαγωγή δεδομένων, 85, 260, 262
εκδόσεις MySQL, 46
Ελάχιστη τιμή, 151
εννοιολογικός σχεδιασμός, 12
ένωση, 174
εξογκωγή δεδομένων, 256
εξειδίκευση, 22
εξωτερική συνένωση, 140, 142, 144, 146
επανάληψη, 159
επίπεδα απομόνωσης, 114
εσωτερική συνένωση, 125, 130
ευρετήριο, 75
ημερομηνία, 165

- ιεραρχία IS-A, 21
 κανονικές μορφές, 60
 κανονικοποίηση, 59
 Καρτεσιανό γινόμενο, 119
 κατάλογος δεδομένων, 59
 Καταμέτρηση, 151
 κλειδί, 29
 κλειδώμα σε επίπεδο εγγραφής, 118
 κωδικός πρόσβασης, 211
 λογικοί τελεστές, 82
 λόγος πληθικότητας, 19
 μαθηματικές συναρτήσεις, 162
 Μέγιστη τιμή, 151
 Μέσος Όρος, 150
 μοντέλο δεδομένων, 12, 24
 μοντέλο οντοτήτων συσχετίσεων, 14
 μοντέλο Οντοτήτων-Συσχετίσεων, 13
 μοντέλο πελάτη - εξυπηρετητή, 48
 ξένο κλειδί, 30, 32, 75
 ολική συμμετοχή, 23
 ομαδοποίηση, 153
 οντότητα, 17
 όψη, 199, 203, 204, 207
 περιορισμοί, 29
 περιορισμός ακεραιότητας οντοτήτων, 29
 περιορισμός αναφορικής ακεραιότητας, 29
 περιορισμός κάλυψης, 23
 περιορισμός συμμετοχής, 23
 πίνακας, 25
 πλειάδα, 25
 πληθικότητα, 20
 πολύπλοκα ερωτήματα, 132
 πρωτεύον κλειδί, 18, 29, 73
 στιγμιότυπο, 25
 συναθροιστικές συναρτήσεις, 150
 συναλλαγή, 104
 συναρτήσεις, 171
 συνάρτηση, 157
 συνένωση, 122, 136
 συνένωση αλφαριθμητικών, 157
 Συνέπεια, 113
 σύνθετο πρωτεύον κλειδί, 28
 σύνολο, 172
 Σύστημα Διαχείρισης Βάσεων Δεδομένων, 11
 συσχέτιση, 18
 σχεδιαστικό εργαλείο σχεσιακών μοντέλων, 26
 σχέση, 25
 Σχεσιακό μοντέλο στο Workbench, 36
 σχήμα σχέσης, 25
 τελεστές σύγκρισης, 82
 τετραγωνική ρίζα, 164
 τομή, 176
 τύποι δεδομένων, 66
 υπερ-κλειδί, 29
 υποερωτήματα, 180
 φυσική συνένωση, 122
 ψευδώνυμο, 134, 135
 ώρα, 165

16. Γλωσσάρι

Διάγραμμα Οντοτήτων Συσχετίσεων (ΔΟΣ) (Entity-Relationship Diagram)

Σχηματικό ή Διάγραμμα το οποίο απεικονίζει το Μοντέλο Οντοτήτων Συσχετίσεων μίας Βάσης Δεδομένων. Εφόσον πρόκειται για το Εκτεταμένο Μοντέλο Οντοτήτων Συσχετίσεων ονομαζεται διάγραμμα ΕΟΣ (EER diagram).

Εγγραφή (record)

Απλή δομή δεδομένων, που αποτελείται από δύο ή περισσότερα πεδία, που αποθηκεύονται σε διαδοχικές θέσεις μνήμης. Η εγγραφή συχνά αναπαρίσταται με μια γραμμή σε έναν πίνακα ΒΔ.

Ευρετήριο (index)

Πρόκειται για μία δομή δεδομένων που δίνει τη δυνατότητα για γρήγορη αναζήτηση στις γραμμές ενός πίνακα, συνήθως χρησιμοποιώντας μια δενδρική δομή δεδομένων (B-tree) που αναπαριστά όλες τις τιμές μιας στήλης ή ενός συνόλου στηλών. Τα ευρετήρια είναι κρίσιμα για την απόδοση των ερωτημάτων στις βάσεις δεδομένων. Απαιτείται για το λόγο αυτό προσεκτικός σχεδιασμός τους ώστε να διευκολύνουν και να επιταχύνουν την ανάκτηση αποτελεσμάτων.

Κανονικοποίηση

Πρόκειται για μία στρατηγική σχεδίασης Βάσεων Δεδομένων όπου τα δεδομένα διαμοιράζονται σε πολλαπλούς πίνακες. Στόχος της στρατηγικής είναι να μειώνονται ή και να εξαλείγονται οι περιπτώσεις εμφάνισης διπλών τιμών με τη χρήση παραπομπής σε μία γραμμή που αντιπροσωπεύεται από ένα ID.

Μοντέλο Οντοτήτων Συσχετίσεων (Entity Relationship Model)

Εννοιολογική μέθοδος αναπαράστασης των δεδομένων μίας Βάση Δεδομένων με σκοπό την αφαιρετική απεικόνιση – περιγραφή.

Μοντέλο πελάτη – εξυπηρετητή (Client – server model)

Αρχιτεκτονική εφαρμογών ή/και δικτύων, στην οποία πολλοί χρήστες (πελάτες - clients) κάθε ένας από τους οποίους χρησιμοποιεί συνήθως ένα PC, συνδέονται για να αποκτήσουν πρόσβαση σε δεδομένα ή

υπηρεσία με ένα ή περισσότερα ισχυρά υπολογιστικά μηχανήματα (εξυπηρετητές - servers). Είναι δυνατό ο πελάτης να βρίσκεται στο ίδιο υπολογιστικό συστημα εξυπηρετητή που φιλοξενεί την υπηρεσία και τα δεδομένα (host). Παράδειγμα εφαρμογών που εφαρμόζουν τη συγκεκριμένη αρχιτεκτονική είναι το email, το WWW (ως υπηρεσία), η δικτυακή εκτύπωση κ.α.

Ξένο κλειδί.

Πρόκειται για ένα είδος συσχέτισης δηλαδή δεικτών μεταξύ γραμμών δεδομένων πίνακα. Η συσχέτιση στην περίπτωση της MySQL δηλώνεται σε μία στήλη και στους δύο πίνακες. Η χρήση της συσχέτισης με ξένο κλειδί επιτρέπει γρήγορη αναζήτηση συσχετιζόμενης πληροφορίας αλλά και την εφαρμογή αναφορικής ακεραιότητας αποτρέποντας τους δείκτους να γίνουν άκυροι καθώς εισάγονται, ενημερώνονται και διαγράφονται δεδομένα.

Περιορισμός.

Πρόκειται για αυτοματοποιημένο έλεγχο που μπορεί να σταματήσει τροποποιήσεις στη Βάση Δεδομένων που θα οδηγούσαν σε ασυνέπειες δεδομένων.

Πλειάδα (tuple)

Τεχνικός όρος που αφορά σε ένα σύνολο διατεταγμένων δεδομένων. Χρησιμοποιείται στη θεωρία των Βάσεων Δεδομένων.

Πληθικότητα (cardinality)

Ο αριθμός των διαφορετικών τιμών που μπορεί να υπάρξουν σε μία στήλη πίνακα.

Πρωτεύον Κλειδί

Ένα σύνολο γνωρισμάτων – στηλών που μπορεί να προσδιορίζει μοναδικά κάθε γραμμή πίνακα. Πρέπει να περιλαμβάνει μοναδικές τιμές και να μην περιέχει τιμές NULL.

Συναλλαγή Βάσης Δεδομένων (Database Transaction)

Αποτελεί ένα ενιαίο σύνολο εντολών που εκτελούνται σε μία Βάση Δεδομένων, αξιόπιστα και ανεξάρτητα από άλλες συναλλαγές. Καλείται επίσης και ως δοσοληψία.

Σχεσιακός

Μία σημαντική άποψη για τις σύγχρονες βάσεις δεδομένων. Ο εξυπηρετητής βάσεων δεδομένων κωδικοποιεί και ενεργοποιεί συσχετίσεις όπως 1-1, 1-N, M-N και διατηρεί συνθήκες μοναδικότητας. Με χρήση των συσχετίσεων είναι δυνατό να διατηρείται η ακεραιότητα και η συνέπεια των δεδομένων στις ΒΔ.

Application programming interface (API)

Ένα σύνολο διαδικασιών και συναρτήσεων. Σε κάθε API παρέχονται προδιαγραφές για τις διαδικασίες και συναρτήσεις και σταθερή ονοματολογία ώστε να διευκολύνεται η χρήση τους από τους προγραμματιστές.

Commit

Πρόκειται για μια SQL δήλωση η οποία ολοκληρώνει μία συναλλαγή και κάνοντας μόνιμες τις τροποποιήσεις που τυχόν προκαλεί η συναλλαγή αυτή. Η αντίθετη διαδικασία είναι το rollback κατά το οποίο επαναφέρονται στην αρχική κατάσταση τυχόν αλλαγές που έχουν προκληθεί από μία συναλλαγή. Στη MySQL έχει προεπιλεγεί η ρύθμιση της αυτόματης εφαρμογής του commit (autocommit) ώστε να πραγματοποιούνται άμεσα οι αλλαγές που προκαλούν οι δηλώσεις SQL.

NULL

Μία ειδική τιμή στην SQL που υποδεικνύει την απουσία δεδομένων. Οποιαδήποτε αριθμητική λειτουργία ή συνθήκη ισότητα που εμπεριέχει τιμή NULL επιστρέφει με τη σειρά της τιμή NULL. Είναι παρόμοια με την έννοια Not-A-Number (NaN) των αριθμών κινητής υποδιαστολής κατά την IEEE.

schema

Εννοιολογικά, ένα σχήμα είναι ένα σύνολο συσχετιζόμενων αντικειμένων βάσης δεδομένων όπως πίνακες, στήλες, τύποι δεδομένων στηλών, ευρετήρια, ξένα κλειδιά κ.α. Αυτά τα αντικείμενα διασυνδέονται με SQL συντακτικό καθώς οι στήλες δημιουργούν πίνακες, τα ξένα κλειδιά αναφέρονται σε πίνακες και στήλες κλπ. Ιδανικά, τα αντικείμενα αυτά θα συνδέονται λογικά και θα αποτελούν όλα μαζί ως μία ενιαία εφαρμογή. Στη MySQL ένα σχήμα (schema) είναι συνώνυμο με μία Βάση Δεδομένων. Είναι δυνατό να αντικαθιστά κανείς τη λέξη DATABASE με τη λέξη SCHEMA στις δηλώσεις SQL στη MySQL. Σε άλλα ΣΔΒΔ το σχήμα και η ΒΔ δεν είναι συνώνυμα. Για παράδειγμα στο ΣΔΒΔ της Oracle ένα σχήμα αναπαριστά μόνο ένα μέρος της βάσης δεδομένων, όπως πίνακες και αντικείμενα που ανήκουν σε ένα συγκεκριμένο χρήστη.

17. Βιβλιογραφία

- [1] Συστήματα Βάσεων Δεδομένων (4η Εκδοση), Silberschatz, Korth, Sudarshan
- [2] Θεμελιωδεις Αρχές Συστημάτων Βάσεων Δεδομένων, R. Elmasri – S.B. Navathe (μετάφραση Μιχάλης Χατζόπουλος)
- [3] Συστήματα Βάσεων Δεδομένων: θεωρία και πρακτική εφαρμογή, I. Μανωλόπουλος, A.N. Παπαδόπουλος
- [4] MySQL 5.0 Certification Study Guide, Paul Dubois, Stefan Hinz, Carsten Pedersen, 24/08/2005
- [5] Beginning MySQL, Robert Sheldon and Geoff Moes
- [6] Beginning MySQL, Paul Wilton and John W. Colby
- [7] MySQL 5.5: Storage Engine Performance Benchmark for MyISAM and InnoDB
- [8] MySQL 5.7 Reference manual (<http://dev.mysql.com/doc/refman/5.7/en/>)

18. ΠΑΡΑΡΤΗΜΑ – ΛΙΣΤΑ ΕΙΚΟΝΩΝ

ΕΙΚΟΝΑ 1 Οι φασεις σχεδιασμού βασεων δεδομενων	13
ΕΙΚΟΝΑ 2 Αναπαρασταση ενος ΔΟΣ	16
ΕΙΚΟΝΑ 3 ΣΥΜΒΟΛΙΣΜΟΙ ΔΟΣ (ΣΥΝΟΠΤΙΚΑ)	16
ΕΙΚΟΝΑ 4 Αναπαρασταση οντοτητων σε ενα ΔΟΣ	18
ΕΙΚΟΝΑ 5 Αναπαρασταση οντοτητων και συσχετισεων ΔΟΣ	19
ΕΙΚΟΝΑ 6 ΣΧΗΜΑΤΙΚΗ ΑΠΕΙΚΟΝΙΣΗ ΣΥΣΧΕΤΙΣΗΣ	19
ΕΙΚΟΝΑ 7 ΠΑΡΑΔΕΙΓΜΑ IS-A	21
ΕΙΚΟΝΑ 8 ΠΑΡΑΔΕΙΓΜΑ ΕΙΔΙΚΕΥΣΗΣ	22
ΕΙΚΟΝΑ 9 ΠΑΡΑΔΕΙΓΜΑ ΓΕΝΙΚΕΥΣΗΣ	23
ΕΙΚΟΝΑ 10 ΠΕΡΙΒΑΛΛΟΝ ΣΧΕΔΙΑΣΗΣ ΣΤΟ MySQL WorkBENCH.....	26
ΕΙΚΟΝΑ 11 ΠΕΡΙΒΑΛΛΟΝ ΣΧΕΔΙΑΣΗΣ ΠΙΝΑΚΩΝ COLUMNS TAB ΣΤΟ MySQL WorkBENCH.....	28
ΕΙΚΟΝΑ 12 ΠΕΡΙΟΡΙΣΜΟΙ ΑΝΑΦΟΡΙΚΗΣ ΑΚΕΡΑΙΟΤΗΤΑΣ ΓΙΑ ΤΟ ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ ΤΗΣ ΒΔ ΥΠΟΥΡΓΕΙΟ	31
ΕΙΚΟΝΑ 13 ΠΕΡΙΟΡΙΣΜΟΙ ΑΝΑΦΟΡΙΚΗΣ ΑΚΕΡΑΙΟΤΗΤΑΣ ΓΙΑ ΤΟΝ ΠΡΟΪΣΤΑΜΕΝΟ ΣΤΗ ΣΧΕΣΗ ΥΠΑΛΛΗΛΟΣ	32
ΕΙΚΟΝΑ 14 ΠΕΡΙΟΡΙΣΜΟΙ ΑΝΑΦΟΡΙΚΗΣ ΑΚΕΡΑΙΟΤΗΤΑΣ ΓΙΑ ΤΗ ΒΔ ΥΠΟΥΡΓΕΙΟ	32
ΕΙΚΟΝΑ 15 ΠΕΡΙΒΑΛΛΟΝ ΣΧΕΔΙΑΣΗΣ ΠΙΝΑΚΩΝ FOREIGN KEYS TAB ΣΤΟ MySQL WorkBENCH	33
ΕΙΚΟΝΑ 16 ΠΕΡΙΒΑΛΛΟΝ ΣΧΕΔΙΑΣΗΣ ΣΧΕΣΙΑΚΟΥ ΜΟΝΤΕΛΟΥ ΣΤΗ MySQL.....	37
ΕΙΚΟΝΑ 17 ΠΕΡΙΒΑΛΛΟΝ ΣΧΕΔΙΑΣΗΣ ΔΙΑΓΡΑΜΜΑΤΟΣ ΣΧΕΣΙΑΚΟΥ ΜΟΝΤΕΛΟΥ ΣΤΗ MySQL.....	37
ΕΙΚΟΝΑ 18 ΣΧΕΔΙΑΣΗ ΕΝΟΣ ΠΙΝΑΚΑ ΣΤΟ ΣΧΕΣΙΑΚΟ ΔΙΑΓΡΑΜΜΑ	38
ΕΙΚΟΝΑ 19 ΣΧΕΔΙΑΣΗ ΤΩΝ ΠΙΝΑΚΩΝ ΚΑΙ ΤΩΝ ΞΕΝΩΝ ΚΛΕΙΔΙΩΝ ΣΤΟ ΣΧΕΣΙΑΚΟ ΔΙΑΓΡΑΜΜΑ	38
ΕΙΚΟΝΑ 20 ΣΧΕΔΙΑΣΗ ΤΩΝ ΠΙΝΑΚΩΝ ΣΥΝΟΛΙΚΑ ΣΤΟ ΣΧΕΣΙΑΚΟ ΔΙΑΓΡΑΜΜΑ ΜΕ ΔΥΟ ΔΙΑΦΟΡΕΤΙΚΕΣ ΕΠΙΛΟΓΕΣ ΑΠΕΙΚΟΝΙΣΗΣ ΤΩΝ ΣΥΣΧΕΤΙΣΕΩΝ.....	38
ΕΙΚΟΝΑ 21 ΣΧΕΔΙΑΣΗ ΤΩΝ ΠΙΝΑΚΩΝ ΣΥΝΟΛΙΚΑ ΣΤΟ ΣΧΕΣΙΑΚΟ ΔΙΑΓΡΑΜΜΑ ΜΕ ΑΠΛΗ ΠΑΡΟΥΣΙΑΣΗ.....	39
ΕΙΚΟΝΑ 22 Οδηγος FORWARD ENGINEERING ΑΡΧΙΚΗ ΕΠΙΛΟΓΗ ΕΞΥΠΗΡΕΤΗΤΗ MySQL ΚΑΙ 2Ο ΒΗΜΑ ΕΠΙΛΟΓΩΝ	39
ΕΙΚΟΝΑ 23 Οδηγος FORWARD ENGINEERING 3Ο ΒΗΜΑ ΕΠΙΛΟΓΩΝ ΑΝΤΙΚΕΙΜΕΝΩΝ ΓΙΑ ΔΗΜΙΟΥΡΓΙΑ	40
ΕΙΚΟΝΑ 24 Οδηγος FORWARD ENGINEERING 5Ο ΟΛΟΚΛΗΡΩΣΗ ΕΚΤΕΛΕΣΗΣ ΚΑΙ	40
ΕΙΚΟΝΑ 25 Το MySQL COMMAND PROMPT – ΓΡΑΜΜΗ ΕΝΤΟΛΩΝ ΠΕΛΑΤΗ MySQL	50
ΕΙΚΟΝΑ 26 Το MySQL DASHBOARD ..	54
ΕΙΚΟΝΑ 27 Το QUERY ANALYZER ..	55
ΕΙΚΟΝΑ 28 Το MySQL WorkBENCH – ΚΕΝΤΡΙΚΗ ΟΘΟΝΗ.....	56
ΕΙΚΟΝΑ 29 Το MySQL WorkBENCH – SQL EDITOR – ΕΠΕΞΕΡΓΑΣΤΗΣ ΕΝΤΟΛΩΝ SQL	56
ΕΙΚΟΝΑ 30 ΑΠΟΤΕΛΕΣΜΑΤΑ.....	83
ΕΙΚΟΝΑ 31 ΠΙΝΑΚΑΣ DEPARTMENTS	87
ΕΙΚΟΝΑ 32 Προσθήκη Πλειαδας-Εγγραφής	87
ΕΙΚΟΝΑ 33 ΠΙΝΑΚΑΣ DEPARTMENTS ΜΕΤΑ ΤΗΝ ΠΡΟΣΘΗΚΗ	87
ΕΙΚΟΝΑ 34 Ο πινακας EMPLOYEES ΠΕΡΙΕΧΕΙ ΤΟΥΣ ΥΠΑΛΛΗΛΟΥΣ ΤΟΥ ΦΟΡΕΑ.....	98
ΕΙΚΟΝΑ 35 ΠΙΝΑΚΑΣ DEPARTMENTS	100
ΕΙΚΟΝΑ 36 Ορισμος SAVEPOINTS ΣΕ ΜΙΑ ΣΥΝΑΛΛΑΓΗ	109
ΕΙΚΟΝΑ 37 ΠΙΝΑΚΑΣ EMPLOYEES ΜΕΤΑ ΤΗΝ ΕΚΤΕΛΕΣΗ ΤΗΣ ΣΥΝΑΛΛΑΓΗΣ	111
ΕΙΚΟΝΑ 38 ΠΙΝΑΚΑΣ EMPLOYEES (ΚΑΡΤΕΣΙΑΝΟ ΓΙΝΟΜΕΝΟ)	120
ΕΙΚΟΝΑ 39 ΠΙΝΑΚΑΣ DEPARTMENTS (ΚΑΡΤΕΣΙΑΝΟ ΓΙΝΟΜΕΝΟ)	120
ΕΙΚΟΝΑ 40 Το ΚΑΡΤΕΣΙΑΝΟ ΓΙΝΟΜΕΝΟ ΤΩΝ ΠΙΝΑΚΩΝ EMPLOYEES ΚΑΙ DEPARTMENTS	120
ΕΙΚΟΝΑ 41 ΦΥΣΙΚΗ ΣΥΝΕΝΩΣΗ - ΠΙΝΑΚΕΣ DEPARTMENTS, ADDRESS	124
ΕΙΚΟΝΑ 42 Το αποτελεσμα της φυσικης συνενωσης στους πινakes DEPARTMENTS και ADDRESS	125
ΕΙΚΟΝΑ 43 ΕΣΩΤΕΡΙΚΗ ΣΥΝΕΝΩΣΗ - INNER JOIN	125
ΕΙΚΟΝΑ 44 ΠΑΡΑΔΕΙΓΜΑ ΕΣΩΤΕΡΙΚΗΣ ΣΥΝΕΝΩΣΗΣ	127
ΕΙΚΟΝΑ 45 Το αποτελεσμα της εσωτερικης συνενωσης	128
ΕΙΚΟΝΑ 46 Εσωτερικη συνενωση με την χρηση του τελεση USING	129
ΕΙΚΟΝΑ 47 Το αποτελεσμα της εσωτερικης συνενωσης με την χρηση του USING	129
ΕΙΚΟΝΑ 48 Εσωτερικη συνενωση και επιπλεον συνοψη	131
ΕΙΚΟΝΑ 49 Αποτελεσμα Εσωτερικης συνενωσης με επιπλεον συνοψη	132
ΕΙΚΟΝΑ 50 ΠΑΡΑΔΕΙΓΜΑ ΣΥΝΕΝΩΣΗΣ ΣΕ ΠΕΡΙΣΣΟΤΕΡΟΥΣ ΑΠΟ ΔΥΟ ΠΙΝΑΚΕΣ	133
ΕΙΚΟΝΑ 51 Το αποτελεσμα συνενωσης σε τρεις πινakes	134
ΕΙΚΟΝΑ 52 Μετονομασια Γνωρισματων-Στηλων πινaka	136
ΕΙΚΟΝΑ 53 Σχημα Πινaka	138

ΕΙΚΟΝΑ 54 SELF-JOIN ΔΥΟ ΑΝΤΙΓΡΑΦΑ ΤΟΥ ΠΙΝΑΚΑ EMPLOYEES.....	139
ΕΙΚΟΝΑ 55 ΑΠΟΤΕΛΕΣΜΑ SELF-JOIN ΣΥΝΕΝΩΣΗΣ ΣΤΟΝ ΠΙΝΑΚΑ EMPLOYEES	139
ΕΙΚΟΝΑ 56 ΕΞΩΤΕΡΙΚΗ ΣΥΝΕΝΩΣΗ Η OUTER JOIN.....	141
ΕΙΚΟΝΑ 57 ΑΡΙΣΤΕΡΗ ΕΞΩΤΕΡΙΚΗ ΣΥΝΕΝΩΣΗ (LEFT OUTER JOIN).....	142
ΕΙΚΟΝΑ 58 ΠΑΡΑΔΕΙΓΜΑ ΑΡΙΣΤΕΡΗΣ ΕΞΩΤΕΡΙΚΗΣ ΣΥΝΕΝΩΣΗΣ	143
ΕΙΚΟΝΑ 59 ΑΠΟΤΕΛΕΣΜΑ ΠΑΡΑΔΕΙΓΜΑΤΟΣ ΑΡΙΣΤΕΡΗΣ ΕΞΩΤΕΡΙΚΗΣ ΣΥΝΕΝΩΣΗΣ	144
ΕΙΚΟΝΑ 60 ΔΕΞΙΑ ΕΞΩΤΕΡΙΚΗ ΣΥΝΕΝΩΣΗ (RIGHT OUTER JOIN)	144
ΕΙΚΟΝΑ 61 ΠΑΡΑΔΕΙΓΜΑ ΔΕΞΙΑΣ ΕΞΩΤΕΡΙΚΗΣ ΣΥΝΕΝΩΣΗΣ.....	145
ΕΙΚΟΝΑ 62 ΑΠΟΤΕΛΕΣΜΑ ΠΑΡΑΔΕΙΓΜΑΤΟΣ ΔΕΞΙΑΣ ΕΞΩΤΕΡΙΚΗΣ ΣΥΝΕΝΩΣΗΣ	146
ΕΙΚΟΝΑ 63 ΠΛΗΡΗ ΕΞΩΤΕΡΙΚΗ ΣΥΝΕΝΩΣΗ (FULL OUTER JOIN)	147
ΕΙΚΟΝΑ 64 ΠΑΡΑΔΕΙΓΜΑ ΠΛΗΡΗΣ ΕΞΩΤΕΡΙΚΗΣ ΣΥΝΕΝΩΣΗΣ.....	148
ΕΙΚΟΝΑ 65 ΑΠΟΤΕΛΕΣΜΑ ΠΑΡΑΔΕΙΓΜΑΤΟΣ ΠΛΗΡΟΥΣ ΕΞΩΤΕΡΙΚΗΣ ΣΥΝΕΝΩΣΗΣ	149
ΕΙΚΟΝΑ 66 ΠΑΡΑΔΕΙΓΜΑ ΣΥΝΑΘΡΟΙΣΤΙΚΗΣ ΣΥΝΑΡΤΗΣΗΣ MAX - Μεγιστος Μισθος του Φορέα	151
ΕΙΚΟΝΑ 67 ΑΠΟΤΕΛΕΣΜΑ ΣΥΝΑΘΡΟΙΣΤΙΚΩΝ ΣΥΝΑΡΤΗΣΕΩΝ ΜΕ ΧΡΗΣΗ ΤΟΥ ΤΕΛΕΣΤΗ WHERE	152
ΕΙΚΟΝΑ 68 ΑΠΟΤΕΛΕΣΜΑ COUNT, ΕΠΙΣΤΡΕΦΕΙ ΤΟΝ ΑΡΙΘΜΟ ΤΩΝ ΥΠΑΛΛΗΛΩΝ ΤΟΥ ΤΜΗΜΑΤΟΣ ΠΛΗΡΟΦΟΡΙΚΗΣ	152
ΕΙΚΟΝΑ 69 Ο ΠΙΝΑΚΑΣ EMPLOYEES ΧΩΡΙΣΤΗΚΕ ΣΕ ΤΡΕΙΣ ΟΜΑΔΕΣ ΒΑΣΕΙ ΤΟΥ ΤΜΗΜΑΤΟΣ ΠΟΥ ΕΧΟΥΝ ΤΟΠΟΘΕΤΗΘΕΙ ΟΙ ΥΠΑΛΛΗΛΟΙ	154
ΕΙΚΟΝΑ 70 ΟΙ ΕΡΓΑΖΟΜΕΝΟΙ ΜΕ ΜΙΣΘΟ 1000, 1500, 2000 ΕΥΡΩ.....	172
ΕΙΚΟΝΑ 71.....	173
ΕΙΚΟΝΑ 72 Η ΠΡΑΞΗ ΤΗΣ ΕΝΩΣΗΣ ΔΥΟ ΣΥΝΟΛΩΝ.....	174
ΕΙΚΟΝΑ 73 Η ΠΡΑΞΗ ΤΗΣ ΕΝΩΣΗΣ.....	175
ΕΙΚΟΝΑ 74 Η ΠΡΑΞΗ ΤΗΣ ΤΟΜΗΣ ΔΥΟ ΣΥΝΟΛΩΝ.....	176
ΕΙΚΟΝΑ 75 Η ΠΡΑΞΗ ΤΗΣ ΤΟΜΗΣ.....	177
ΕΙΚΟΝΑ 76 Η ΠΡΑΞΗ ΤΗΣ ΔΙΑΦΟΡΑΣ ΔΥΟ ΣΥΝΟΛΩΝ.....	178
ΕΙΚΟΝΑ 77 Η ΠΡΑΞΗ ΤΗΣ ΔΙΑΦΟΡΑΣ.....	178
ΕΙΚΟΝΑ 78 ΤΙ ΕΙΝΑΙ ΤΟ ΥΠΟΕΡΩΤΗΜΑ	180
ΕΙΚΟΝΑ 79.....	181
ΕΙΚΟΝΑ 80.....	183
ΕΙΚΟΝΑ 81.....	184
ΕΙΚΟΝΑ 82 ΑΠΟΤΕΛΕΣΜΑ ΑΝΑΖΗΤΗΣΗΣ ΣΤΟ VIEW	202
ΕΙΚΟΝΑ 83.....	202
ΕΙΚΟΝΑ 84.....	207
ΕΙΚΟΝΑ 85.....	207
ΕΙΚΟΝΑ 86 ΔΗΜΙΟΥΡΓΙΑ ΧΡΗΣΤΗ ROOT	210
ΕΙΚΟΝΑ 87 ΑΔΥΝΑΜΟΣ ΚΩΔΙΚΟΣ ΠΡΟΣΒΑΣΗΣ.....	210
ΕΙΚΟΝΑ 88 ΔΗΜΙΟΥΡΓΙΑ ΧΡΗΣΤΗ.....	211
ΕΙΚΟΝΑ 89 ΔΗΜΙΟΥΡΓΙΑ ΧΡΗΣΤΗ.....	211
ΕΙΚΟΝΑ 90 ΠΙΘΑΝΟΙ ΡΟΛΟΙ ΧΡΗΣΤΗ.....	211
ΕΙΚΟΝΑ 91 ΔΗΜΙΟΥΡΓΙΑ ΧΡΗΣΤΗ - ΔΙΑΦΟΡΟΠΟΙΗΣΗ ΑΠΟ ΆΛΛΕΣ ΒΑΣΕΙΣ	212
ΕΙΚΟΝΑ 92 USERS AND PRIVILIGES.....	215
ΕΙΚΟΝΑ 93 ΟΘΟΝΗ USERS AND PRIVILEGES	215
ΕΙΚΟΝΑ 94 ΔΗΜΙΟΥΡΓΙΑ ΝΕΟΥ ΧΡΗΣΤΗ	216
ΕΙΚΟΝΑ 95 ΠΡΟΕΙΔΟΠΟΙΗΣΗ ΓΙΑ PASSWORD.....	216
ΕΙΚΟΝΑ 96 ΔΙΚΑΙΩΜΑΤΑ ΤΟΥ ΧΡΗΣΤΗ.....	217
ΕΙΚΟΝΑ 97 ΔΙΚΑΙΩΜΑΤΑ ΧΡΗΣΤΗ ΣΕ ΟΛΑ ΤΑ ΣΧΗΜΑΤΑ.....	217
ΕΙΚΟΝΑ 98 ΔΙΚΑΙΩΜΑΤΑ ΧΡΗΣΤΗ ΣΕ ΣΥΓΚΕΚΡΙΜΕΝΟ ΣΧΗΜΑ	218
ΕΙΚΟΝΑ 99 ΕΜΦΑΝΙΣΗ ΔΙΚΑΙΩΜΑΤΩΝ ΧΡΗΣΤΗ	218
ΕΙΚΟΝΑ 100 ΆΛΛΑΓΗ ΟΝΟΜΑΤΟΣ ΧΡΗΣΤΗ	219
ΕΙΚΟΝΑ 101 ΔΙΑΓΡΑΦΗ ΧΡΗΣΤΗ	220
ΕΙΚΟΝΑ 102 ΕΠΙΒΕΒΑΙΩΣΗ ΔΙΑΓΡΑΦΗΣ ΧΡΗΣΤΗ	221
ΕΙΚΟΝΑ 103 ΆΛΛΑΓΗ ΚΩΔΙΚΟΥ ΠΡΟΣΒΑΣΗΣ	223
ΕΙΚΟΝΑ 104 ΑΠΟΔΟΣΗ ΔΙΚΑΙΩΜΑΤΩΝ ΣΤΟ ΧΡΗΣΤΗ.....	229
ΕΙΚΟΝΑ 105 ΑΠΟΔΟΣΗ ΔΙΚΑΙΩΜΑΤΩΝ ΣΤΟ ΧΡΗΣΤΗ.....	230
ΕΙΚΟΝΑ 106 ΑΠΟΤΕΛΕΣΜΑ SHOW GRANTS.....	232
ΕΙΚΟΝΑ 107 ΠΕΡΙΟΡΙΣΜΟΣ ΔΙΚΑΙΩΜΑΤΩΝ	234
ΕΙΚΟΝΑ 108.....	240
ΕΙΚΟΝΑ 109 TABLE MAINTENANCE.....	240

ΕΙΚΟΝΑ 110 TABLE MAINTENANCE.....	240
ΕΙΚΟΝΑ 111.....	241
ΕΙΚΟΝΑ 112 REPAIR TABLE.....	249
ΕΙΚΟΝΑ 113 ΕΞΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ	255
ΕΙΚΟΝΑ 114 ΕΞΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ	256
ΕΙΚΟΝΑ 115 ΕΚΤΕΛΕΣΗ ΕΞΑΓΩΓΗΣ ΔΕΔΟΜΕΝΩΝ	256
ΕΙΚΟΝΑ 116 ΕΙΣΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ	260
ΕΙΚΟΝΑ 117 ΕΙΣΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ	260
ΕΙΚΟΝΑ 118 ΕΚΤΕΛΕΣΗ ΕΙΣΑΓΩΓΗΣ ΔΕΔΟΜΕΝΩΝ.....	261
ΕΙΚΟΝΑ 119.....	262
ΕΙΚΟΝΑ 120 ΑΠΟΤΕΛΕΣΜΑΤΑ.....	266
ΕΙΚΟΝΑ 121 ΑΠΟΤΕΛΕΣΜΑΤΑ.....	266
ΕΙΚΟΝΑ 122 ΑΠΟΤΕΛΕΣΜΑΤΑ SCHEMA	267
ΕΙΚΟΝΑ 123 ΣΧΕΣΙΑΚΟ ΔΙΑΓΡΑΜΜΑ ΤΗΣ ΒΔ PAYROLL.....	305
ΕΙΚΟΝΑ 124 ΣΧΕΣΙΑΚΟ ΔΙΑΓΡΑΜΜΑ ΤΗΣ ΒΔ PAYROLL ΜΕ ΕΝΔΕΙΞΗ ΤΩΝ PRIMARY & FOREIGN KEYS	306

19. ΠΑΡΑΡΤΗΜΑ - ΕΙΚΟΝΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ MySQL 5.6



Downloads



All Databases – MySQL



MySQL

A screenshot of a Microsoft Internet Explorer browser window showing the MySQL Downloads page at 'http://www.mysql.com/downloads/'. The main content area is titled 'MySQL Downloads' and features a section for 'MySQL Enterprise Edition (commercial)'. It lists various components: MySQL Database, MySQL Storage Engines (InnoDB, MyISAM, etc.), MySQL Connectors (JDBC, ODBC, .Net, etc.), MySQL Workbench, MySQL Replicator, MySQL Partitioning, MySQL Enterprise Backup, MySQL Enterprise Monitor, MySQL Enterprise HA, MySQL Enterprise Scalability, MySQL Enterprise Security, and MySQL Enterprise Audit. On the left, there is a sidebar with contact information for various countries and a 'Contact Us Below' button.

4

MySQL Community Edition

The screenshot shows the MySQL Downloads page in a web browser. The main content area is titled "MySQL Community Edition (GPL)". It includes a brief description: "MySQL Community Edition (GPL) is the world's most popular open-source database. MySQL Community Edition is the MySQL distribution that includes the MySQL Server, MySQL Client, MySQL Utilities, MySQL Connectors, and MySQL Workbench." Below this, there are download links for "MySQL Server 5.6.27" and "MySQL 5.6.27". A "Download" button is visible at the bottom of the section.

5

MySQL Community Server

The screenshot shows the MySQL Downloads page in a web browser. The main content area is titled "MySQL Enterprise Edition". It includes a brief description: "MySQL Enterprise Edition includes the most comprehensive set of advanced features and management tools for MySQL." Below this, there are download links for "MySQL Enterprise Edition 5.6.27" and "MySQL Enterprise Edition 5.6.27". A "Download" button is visible at the bottom of the section.

6

MySQL Community Server

The screenshot shows the MySQL Download MySQL Community Server page. The URL in the address bar is <http://dev.mysql.com/downloads/mysql/>. The page title is "MySQL - Download MySQL Community Server". The main content area is titled "Download MySQL Community Server". It features a sidebar with links to various MySQL products like MySQL Enterprise Edition, MySQL Cluster, MySQL Community Server, etc. A "Contact Sales" section provides phone numbers for USA, Canada, Germany, France, and Australia. The main content area includes sections for "MySQL Documentation" and "Archived Versions". A sidebar on the right contains a note about MySQL open-source software being licensed under the GPL license and offers to purchase commercial support.

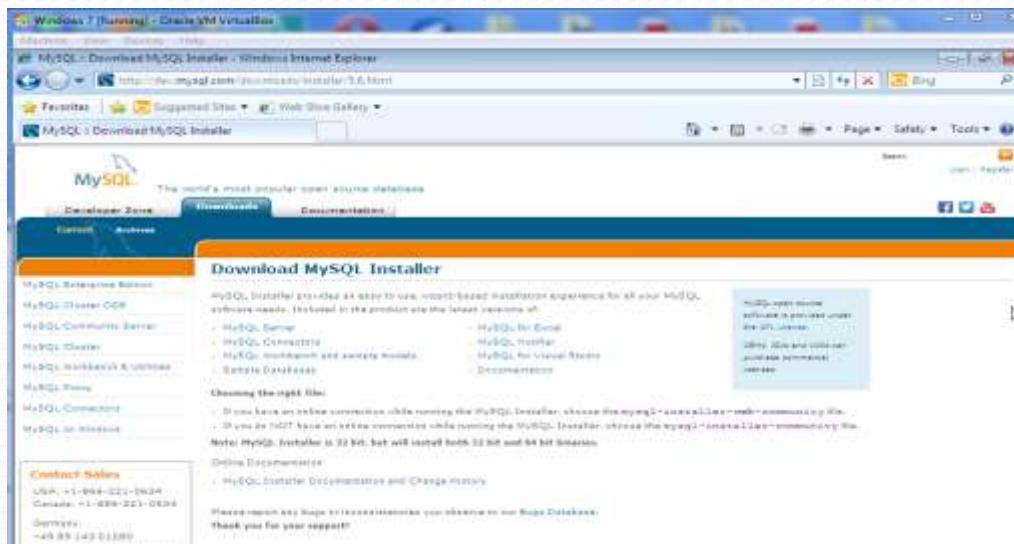
7

MySQL Installer msi

The screenshot shows the same MySQL Download MySQL Community Server page, but the main content area is now focused on the "MySQL Installer 5.6 for Windows". It features a large image of the Windows logo and the text "All MySQL Products. For All Windows Platforms. In One Package.". Below this, there are download links for "Windows (x86, 64-bit), MySQL Installer MSI" and "Windows (x86, 64-bit), ZIP Archive". There are also other download links for "Windows (x86, 64-bit), ZIP Archive" and "Windows (x86, 32-bit), ZIP Archive". A note at the bottom suggests using MD5 checksums and GnuPG signatures to verify package integrity.

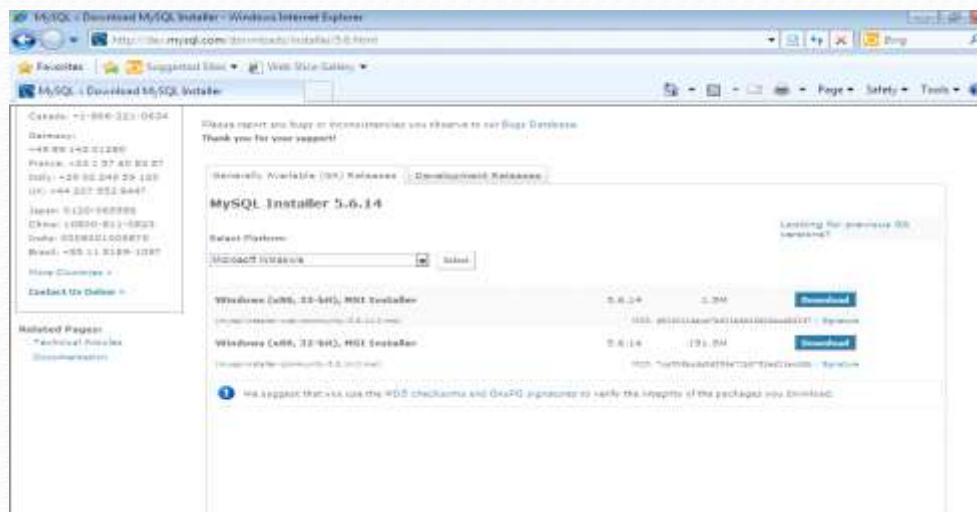
8

MySQL Installer



9

MySQL Installer Community 5.6.14



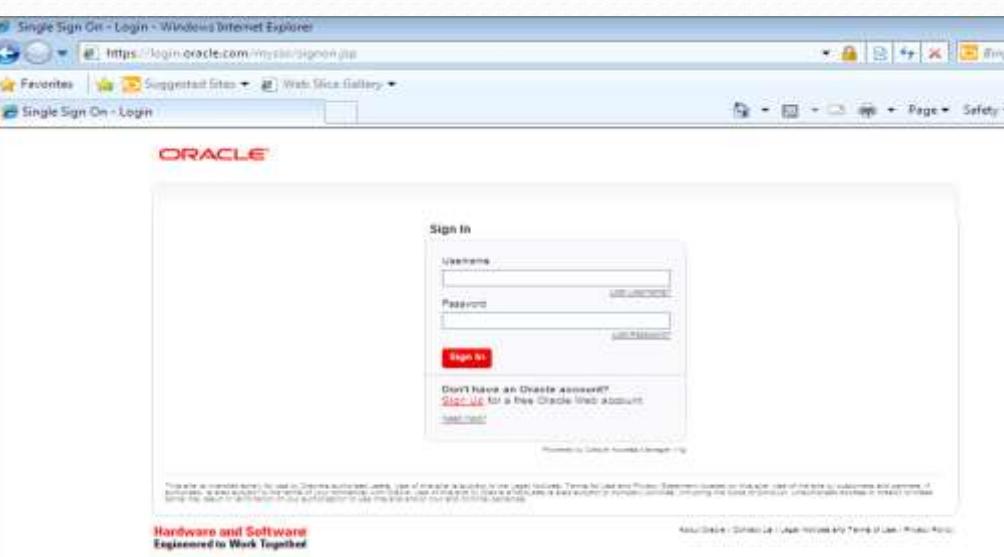
10

Login



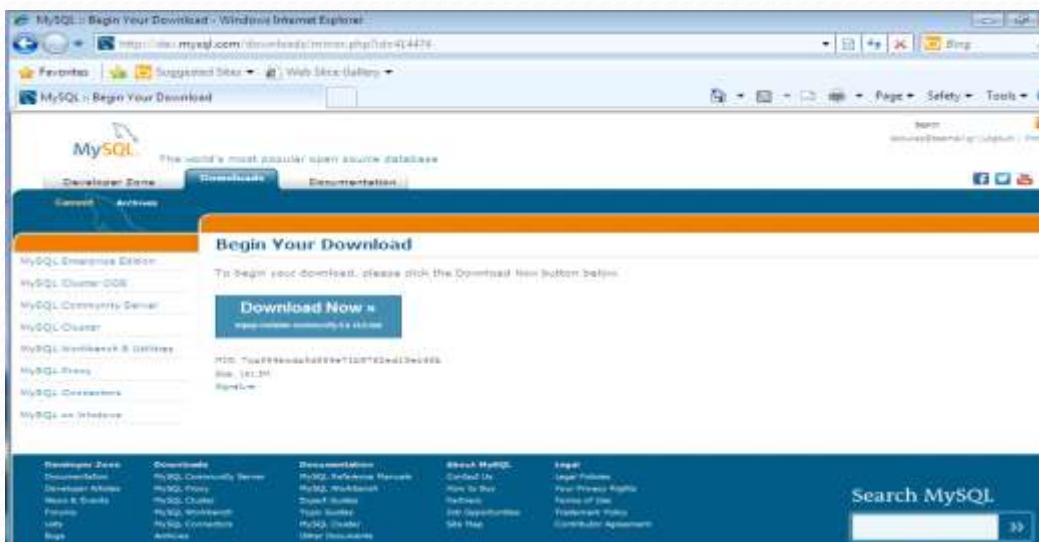
11

Oracle SSO



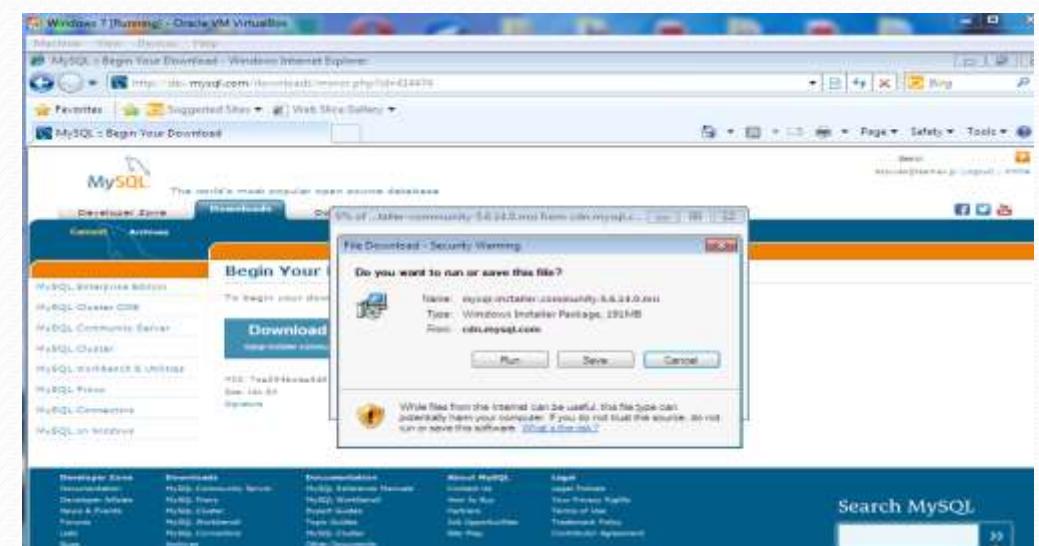
12

Download



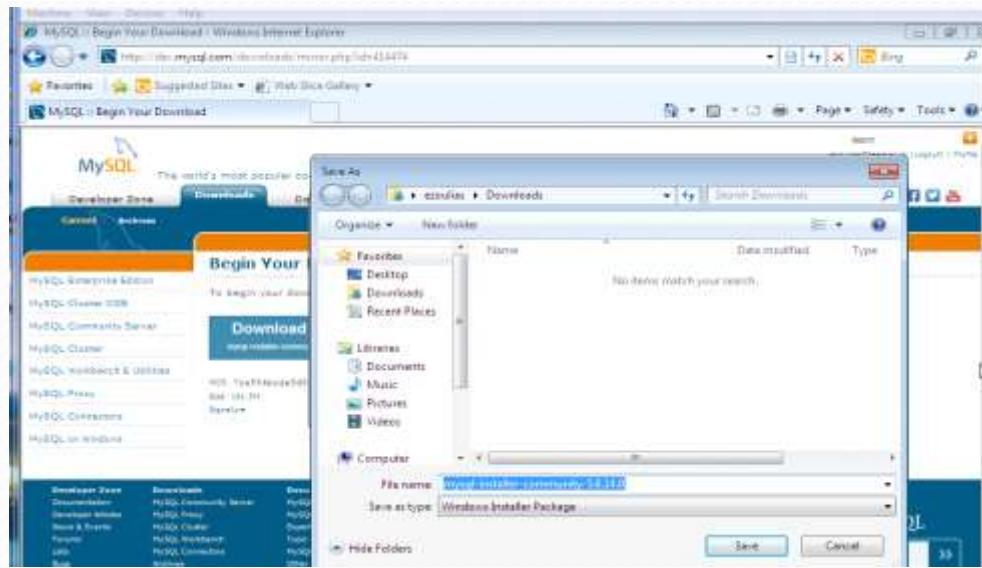
13

Save



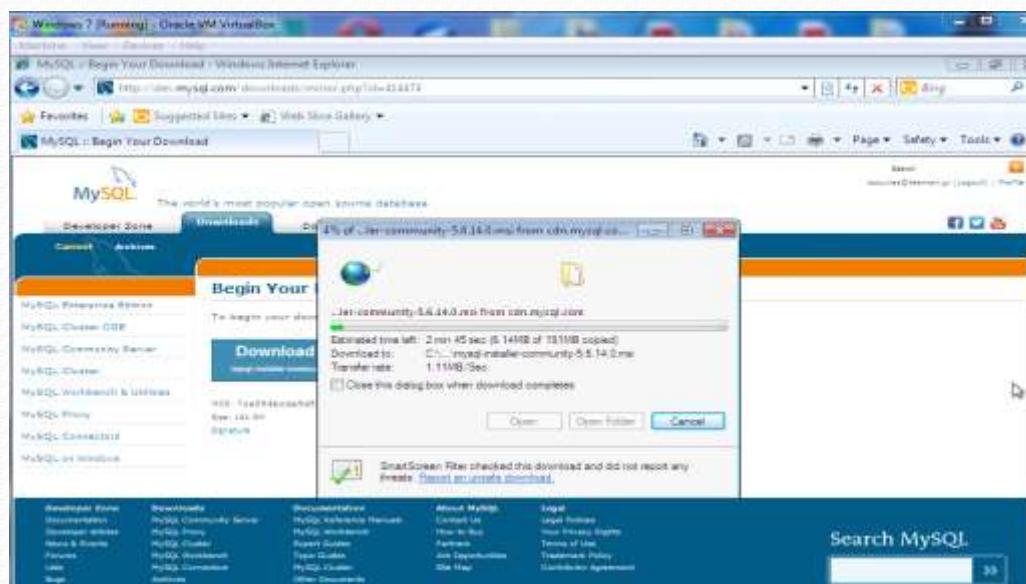
14

Save



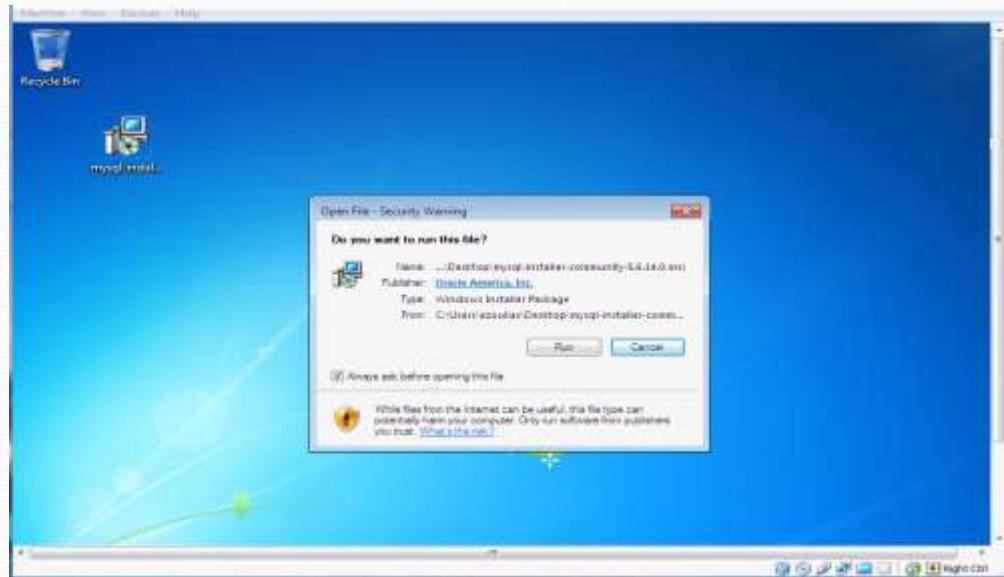
15

Download



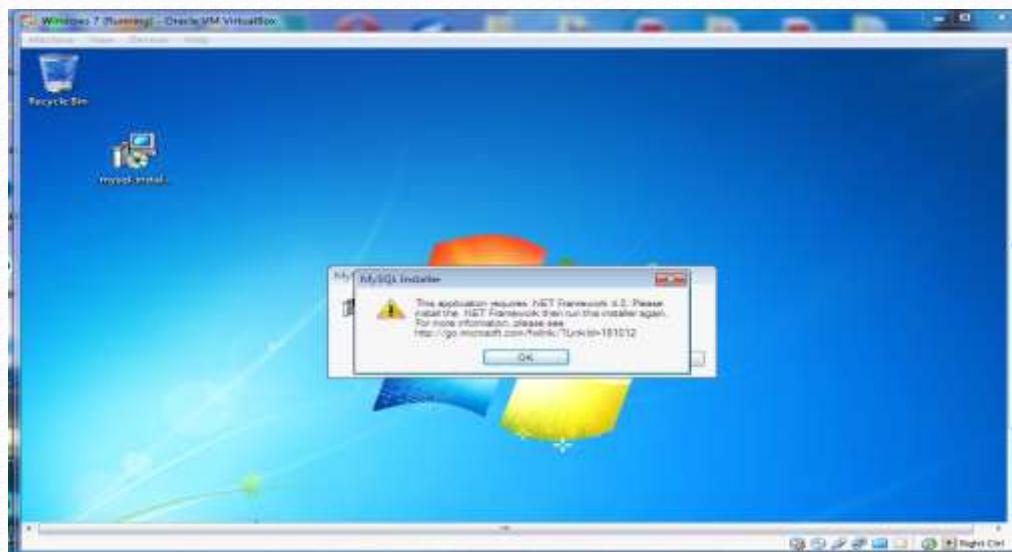
16

Installation



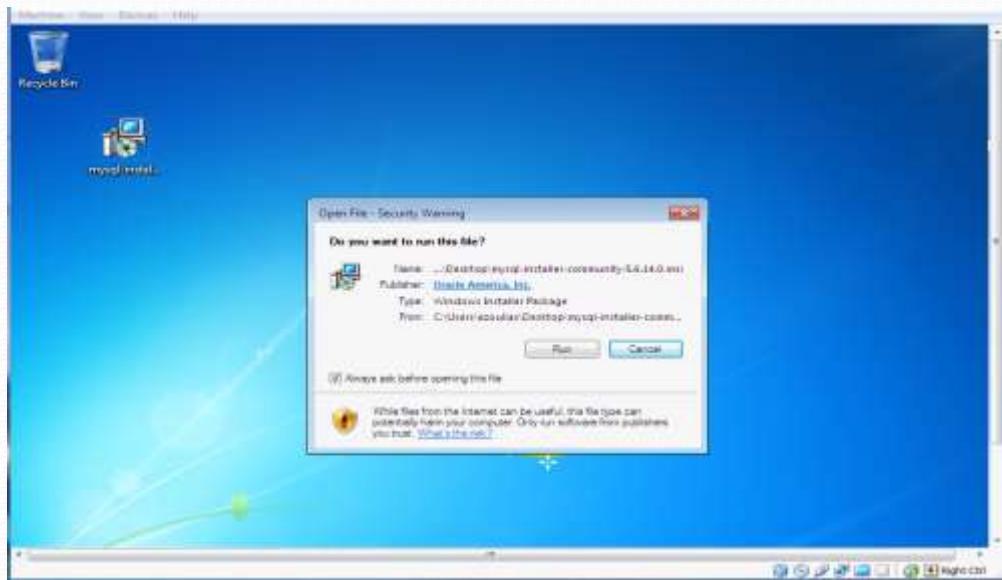
17

Ανάγκη εγκατάστασης .NET 4.0 και VC++



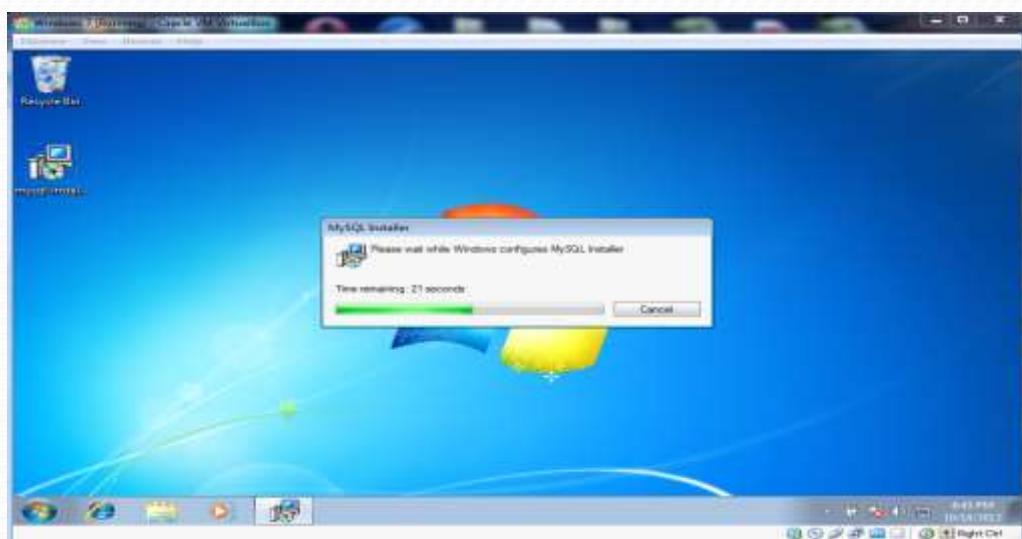
18

Installation



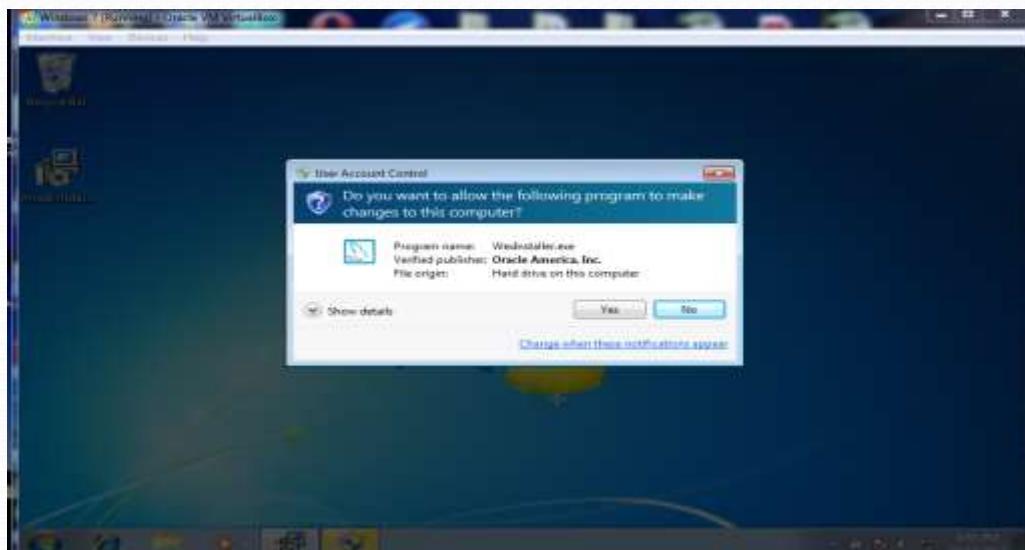
19

Installation

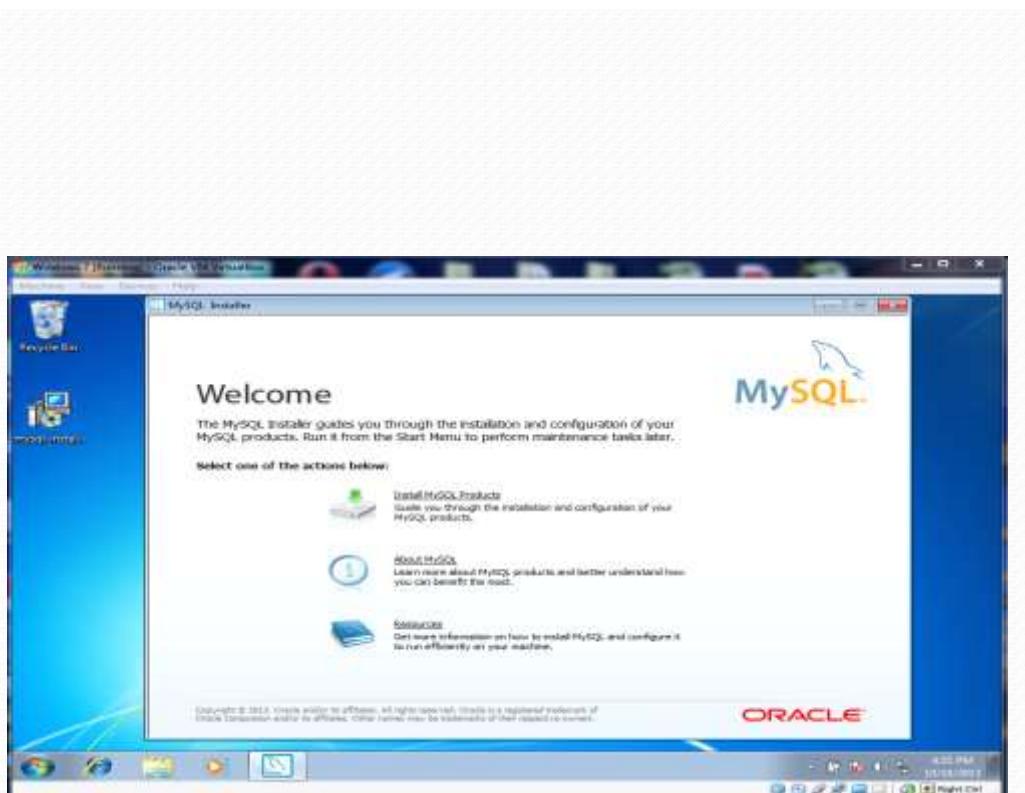


20

Installation - Yes

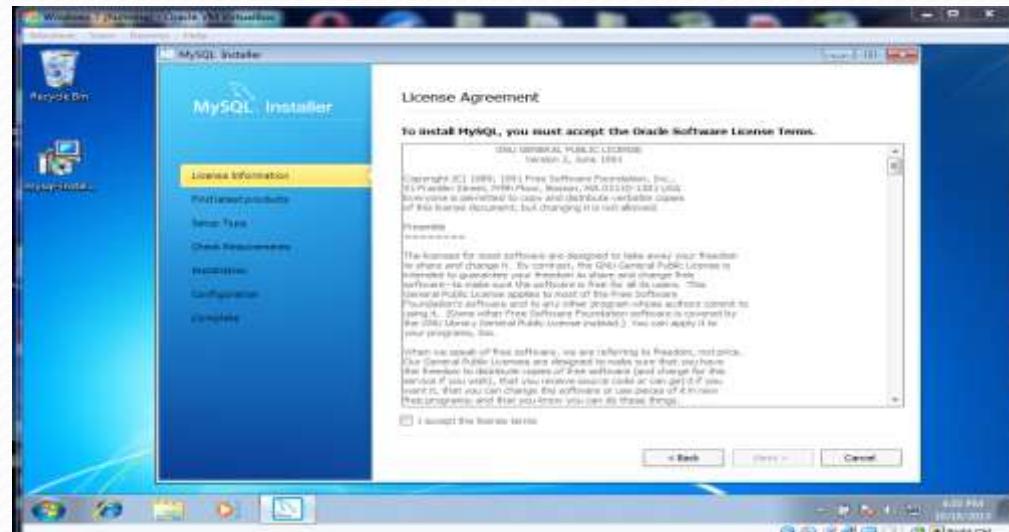


21



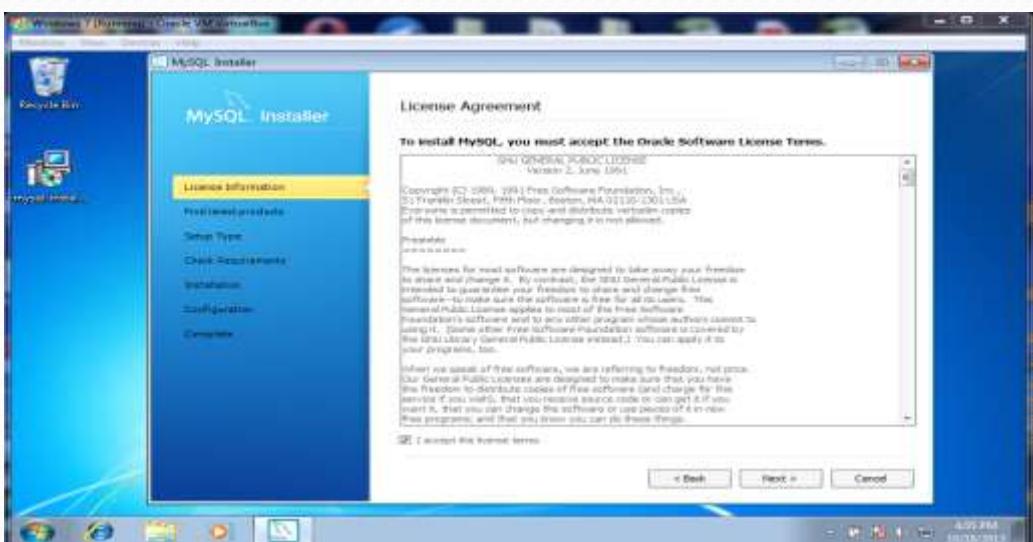
22

I agree



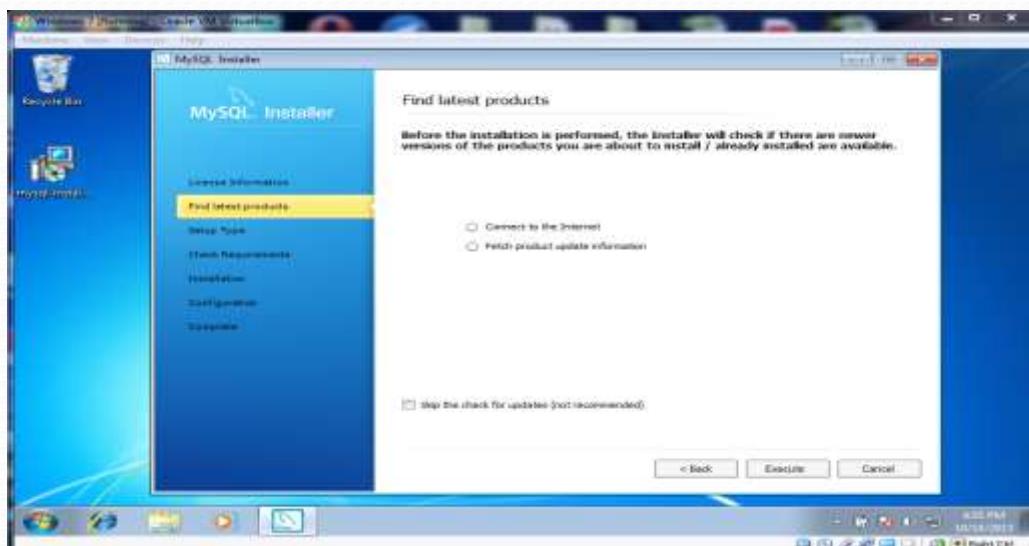
23

Next



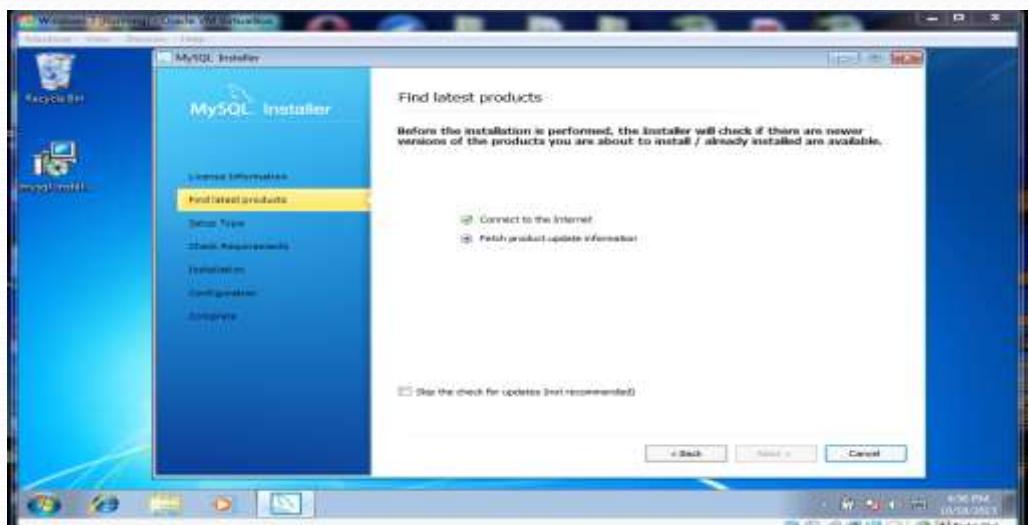
24

Execute



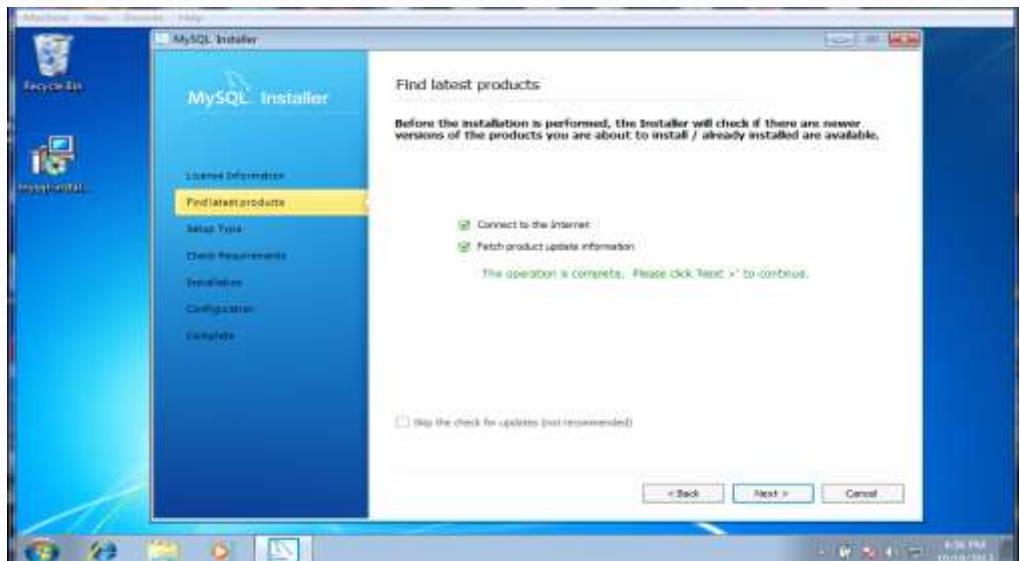
25

Wait



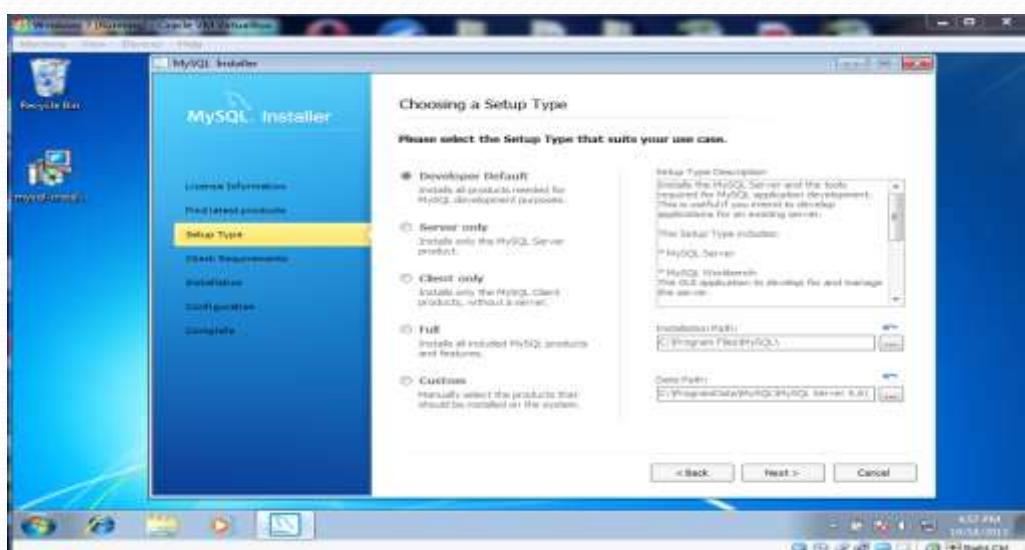
26

Next



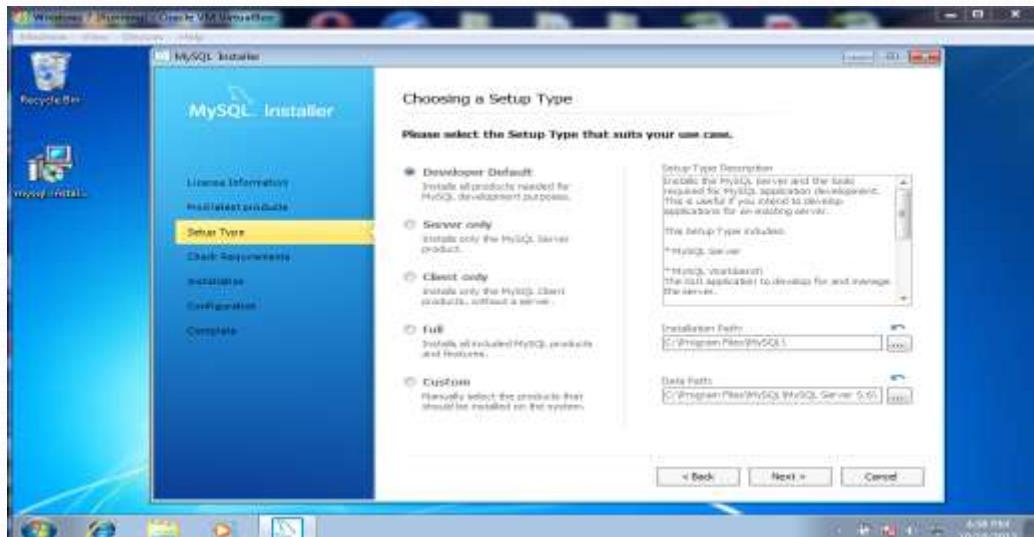
27

Αλλαγή Data path



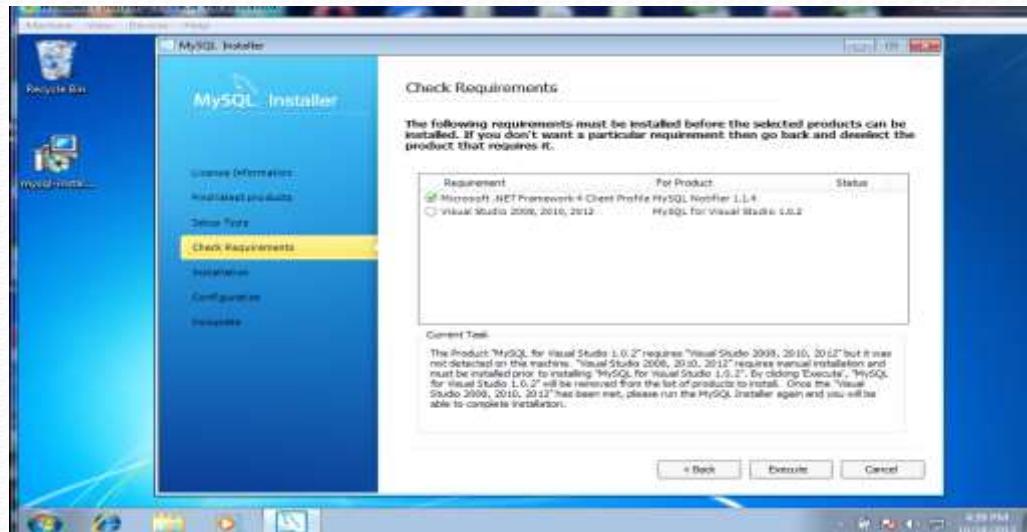
28

Αλλαγή Data path



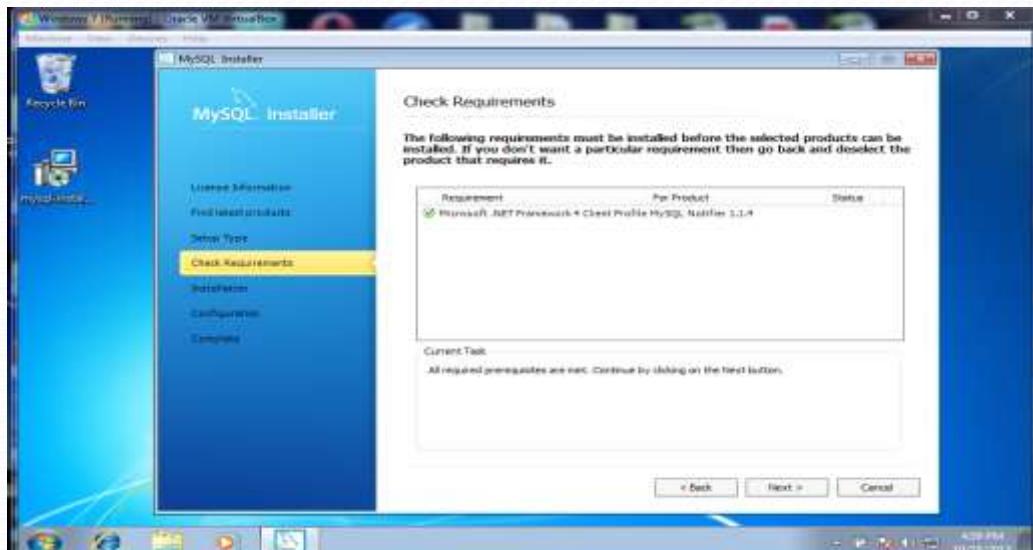
29

Execute



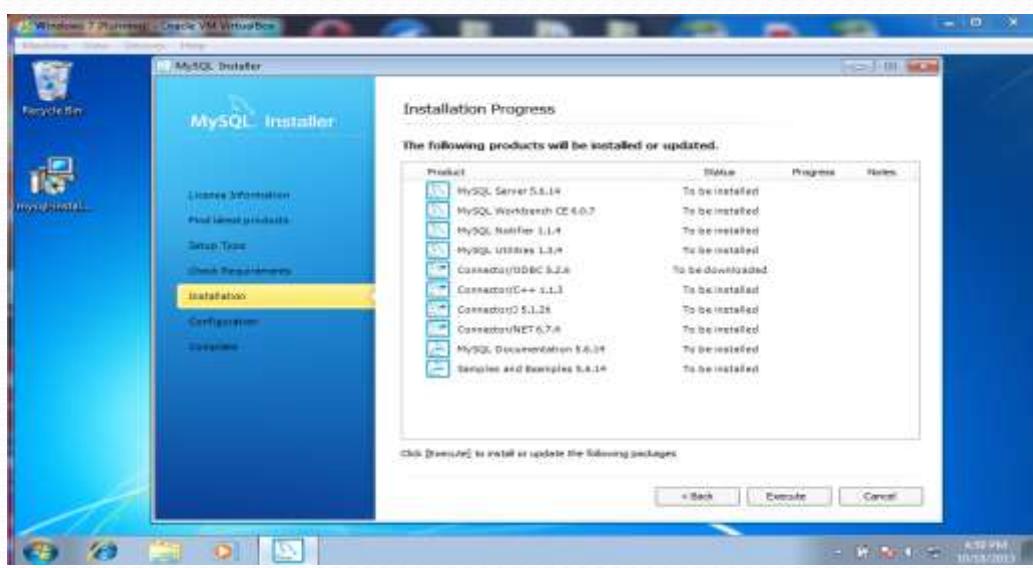
30

Next



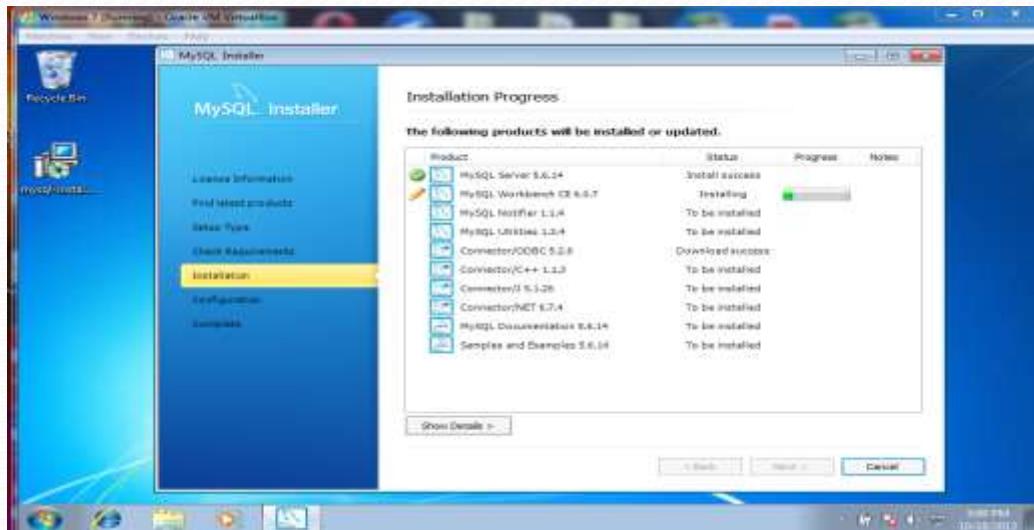
31

Execute



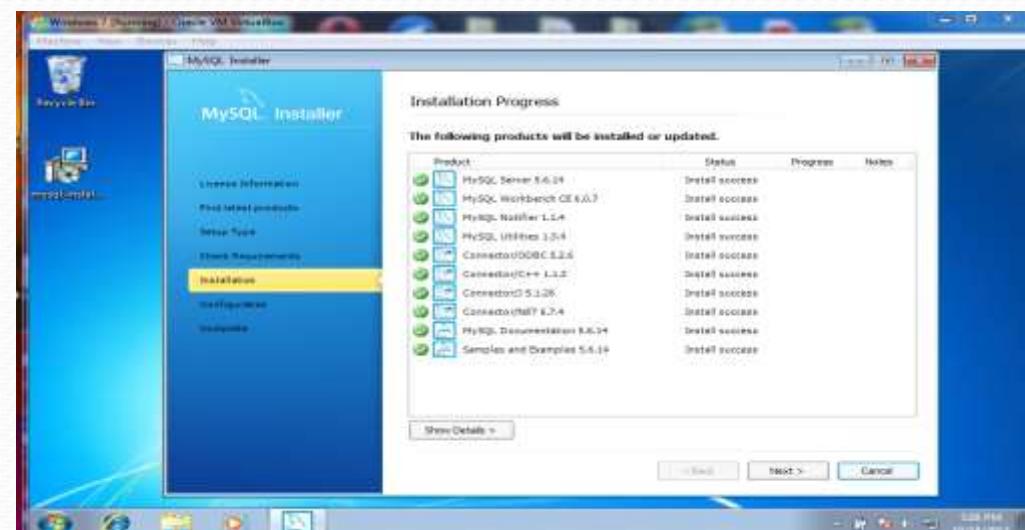
32

Wait....



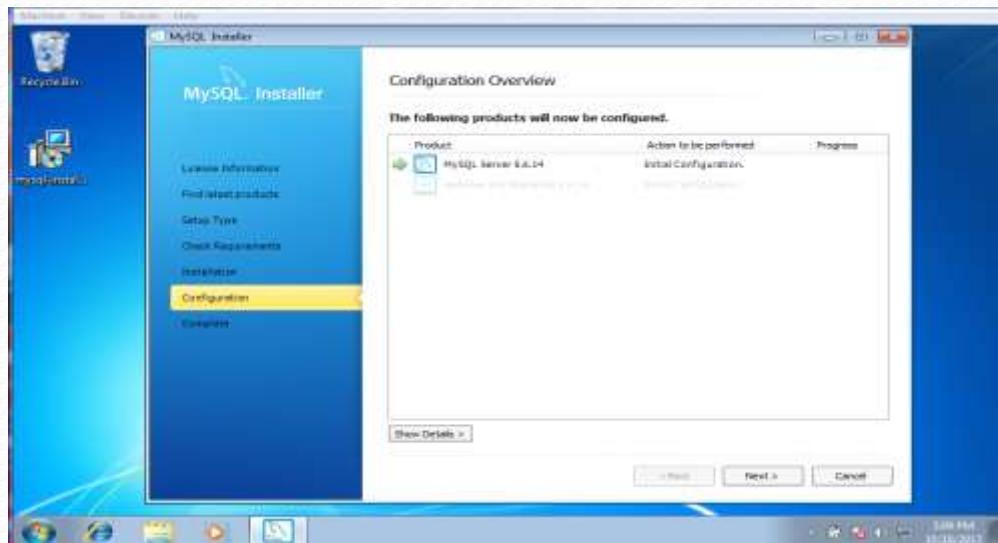
33

Finished - Successfully



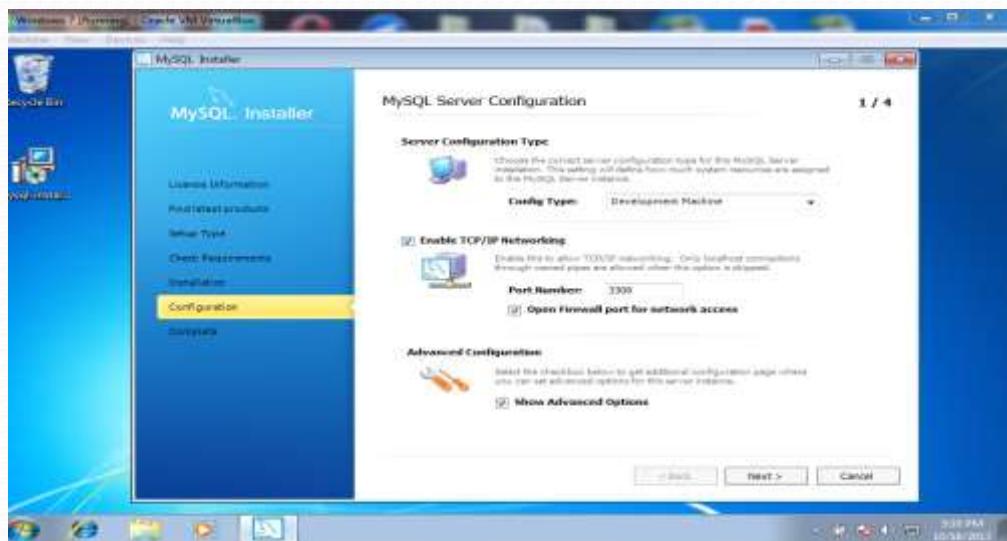
34

Configuration



35

MySQL Server Configuration



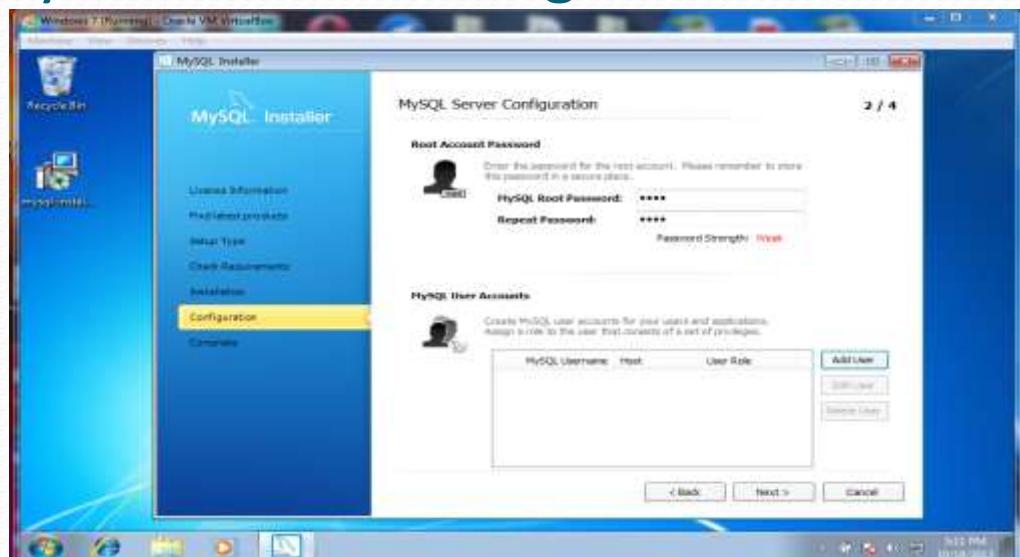
37

MySQL Server Configuration - Users



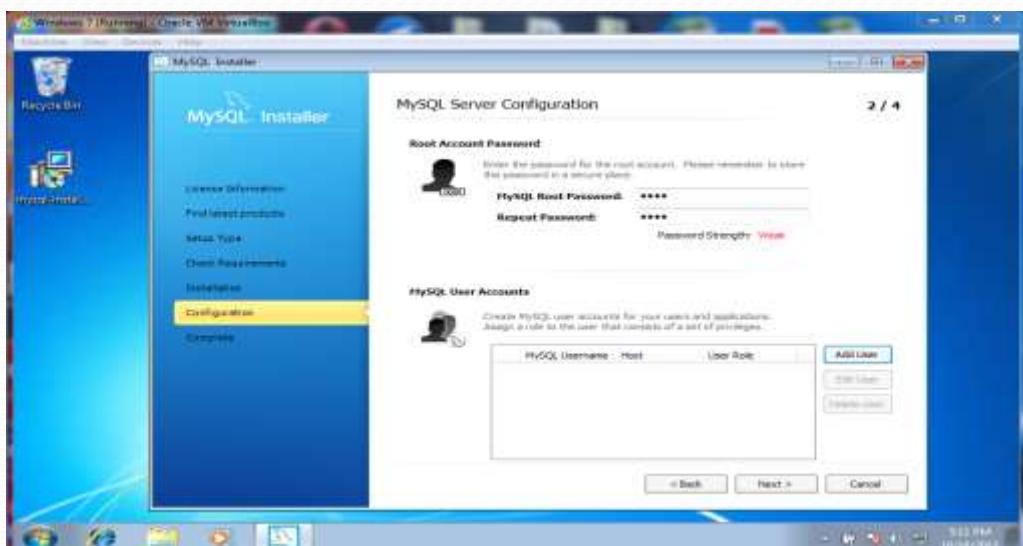
38

MySQL Server Configuration - Users



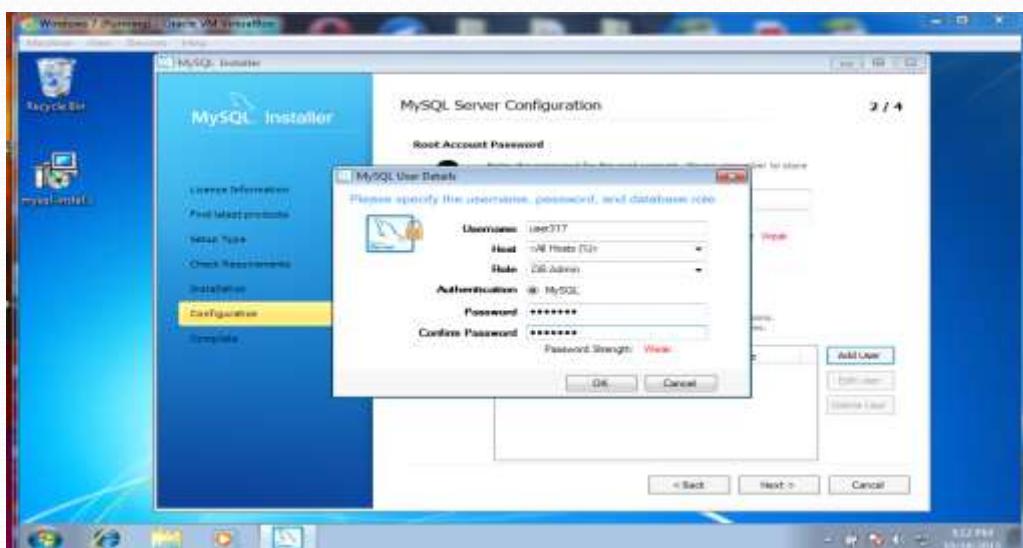
39

MySQL Server Configuration - Users



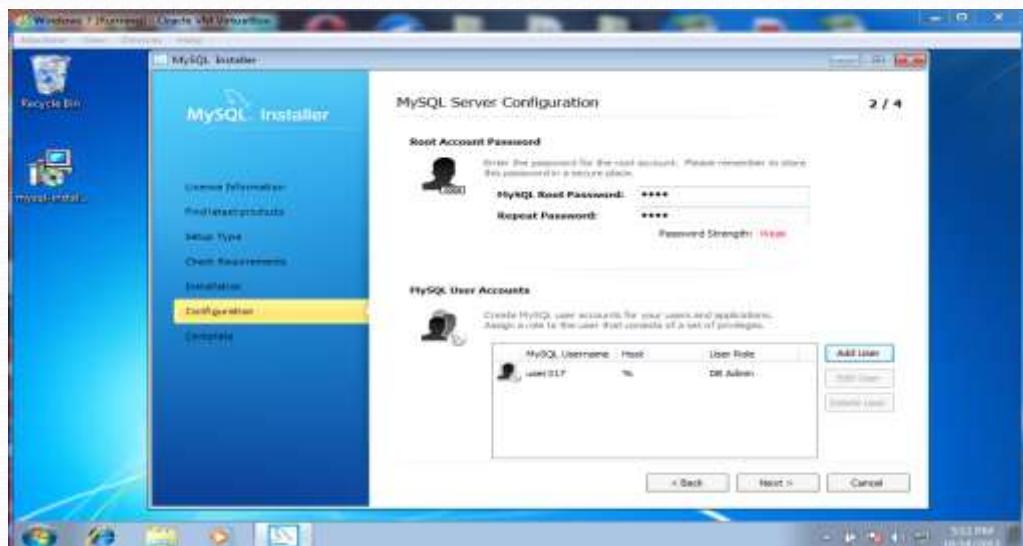
40

MySQL Server Configuration - Users



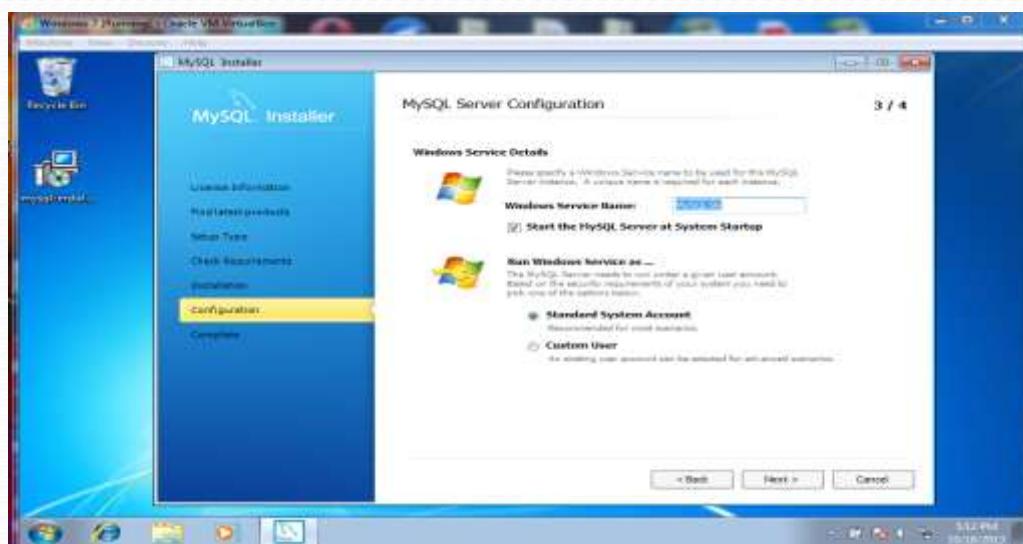
41

MySQL Server Configuration - Users



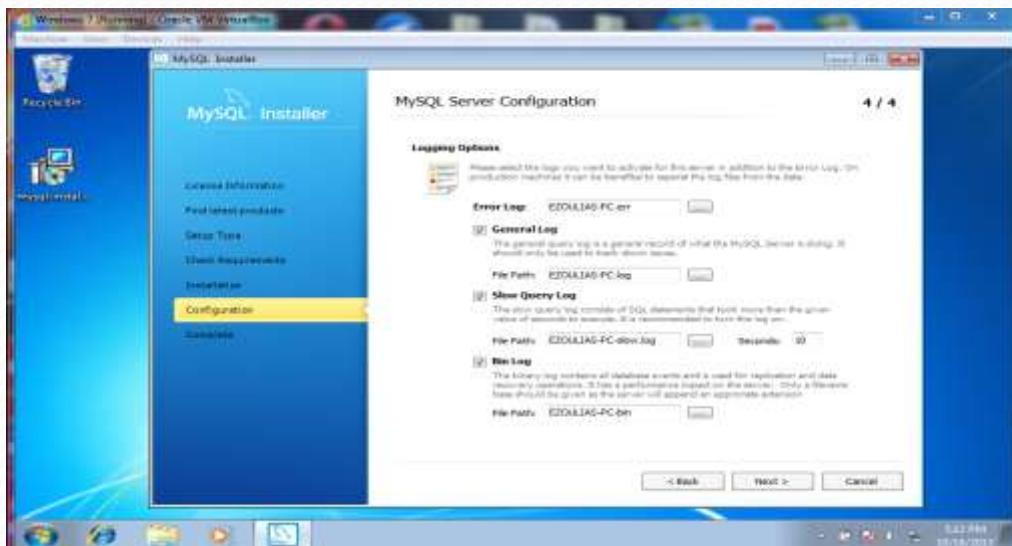
42

MySQL Server Configuration



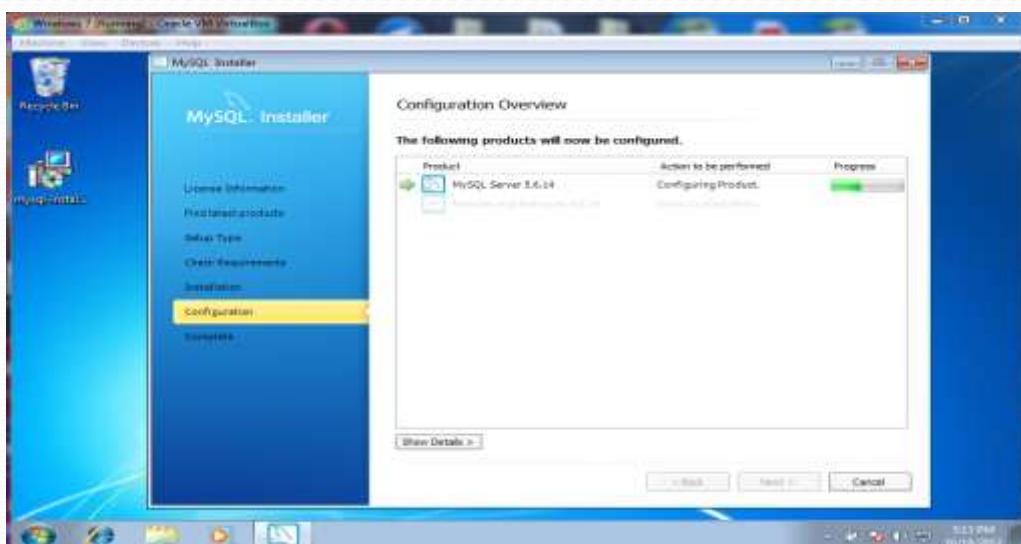
43

MySQL Server Configuration - Logs



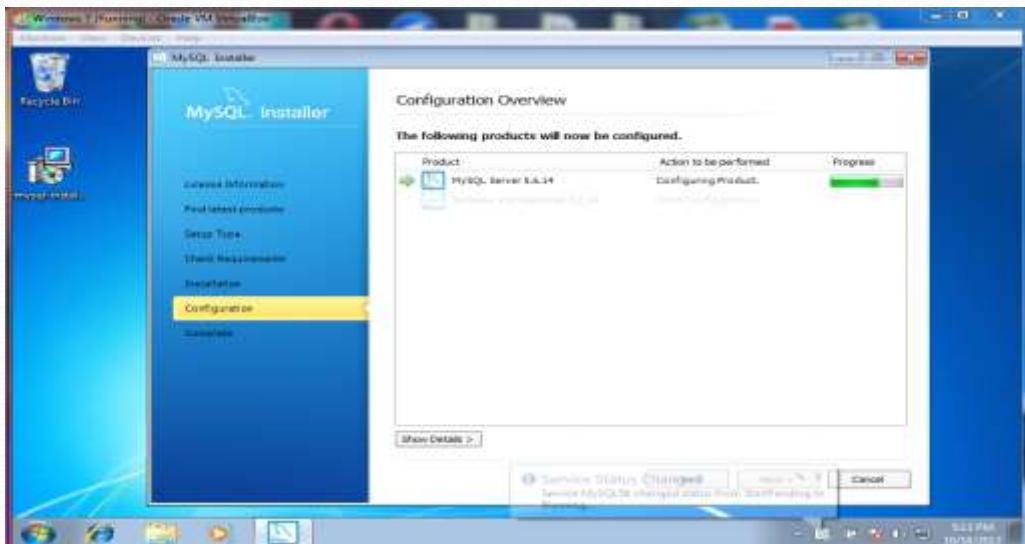
44

MySQL Server Configuration



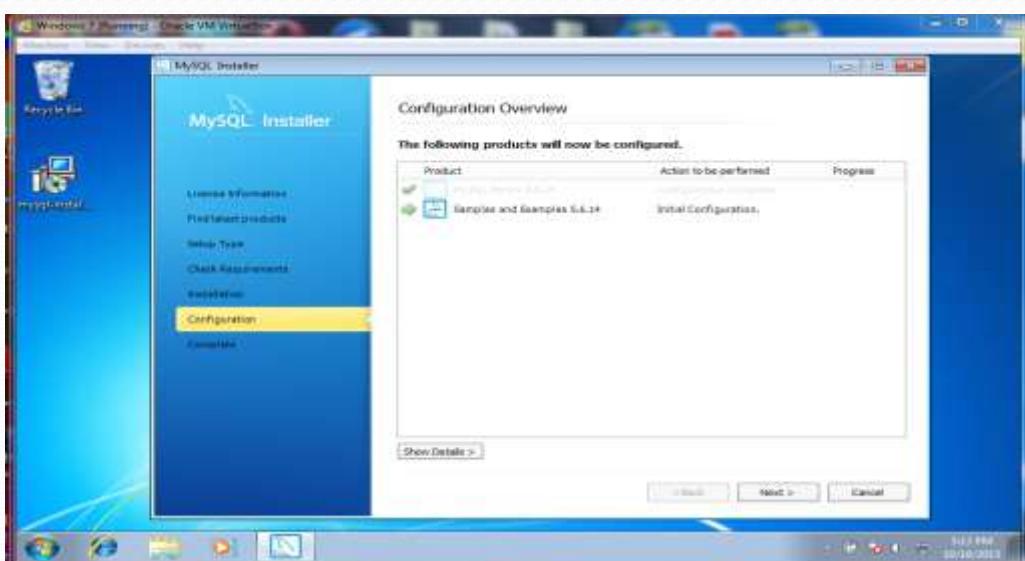
45

MySQL Server Configuration



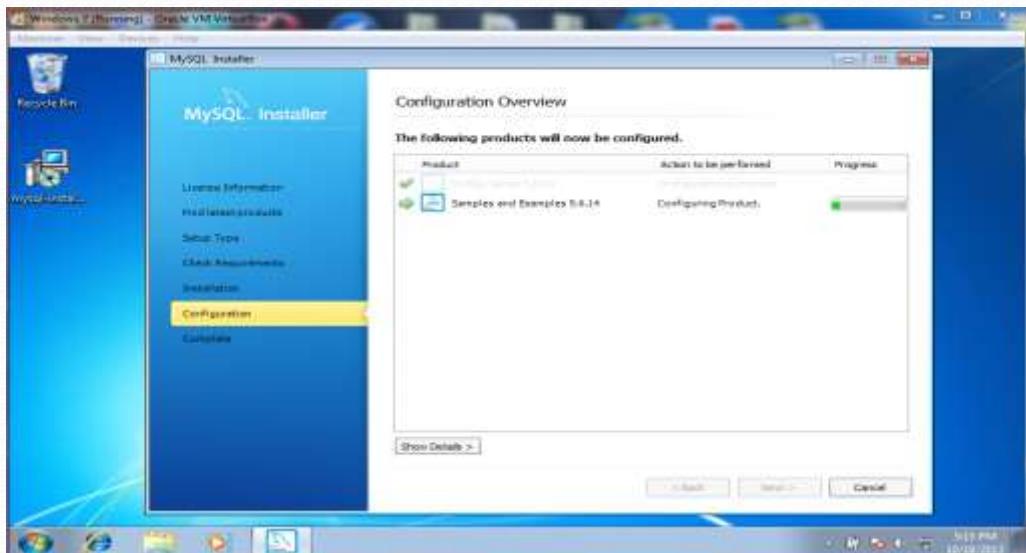
46

MySQL Server Configuration

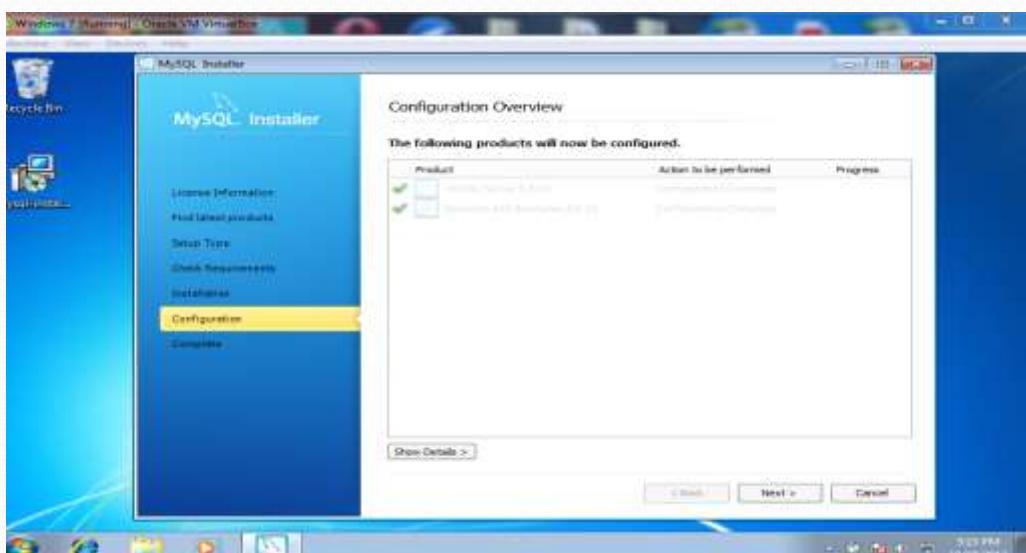


47

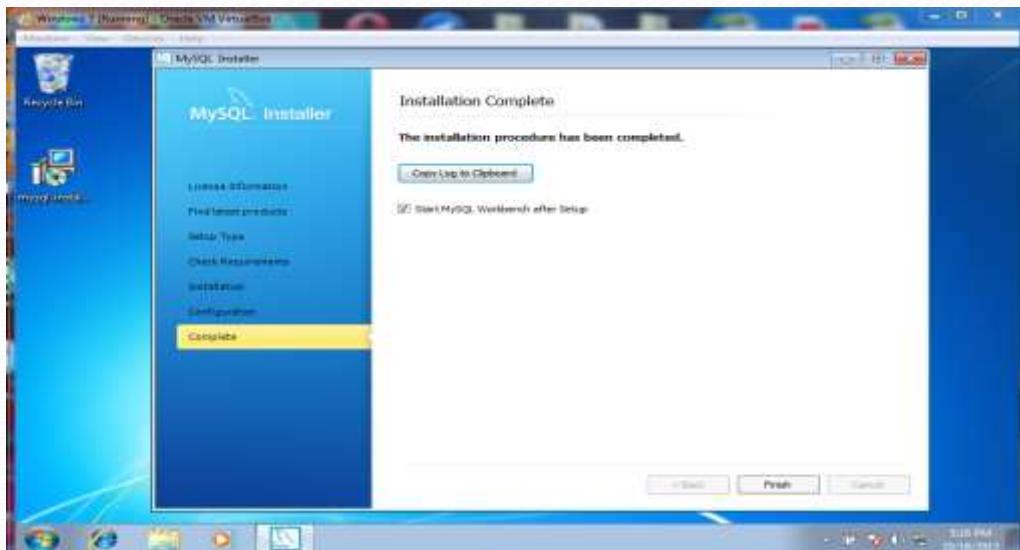
MySQL Server Configuration



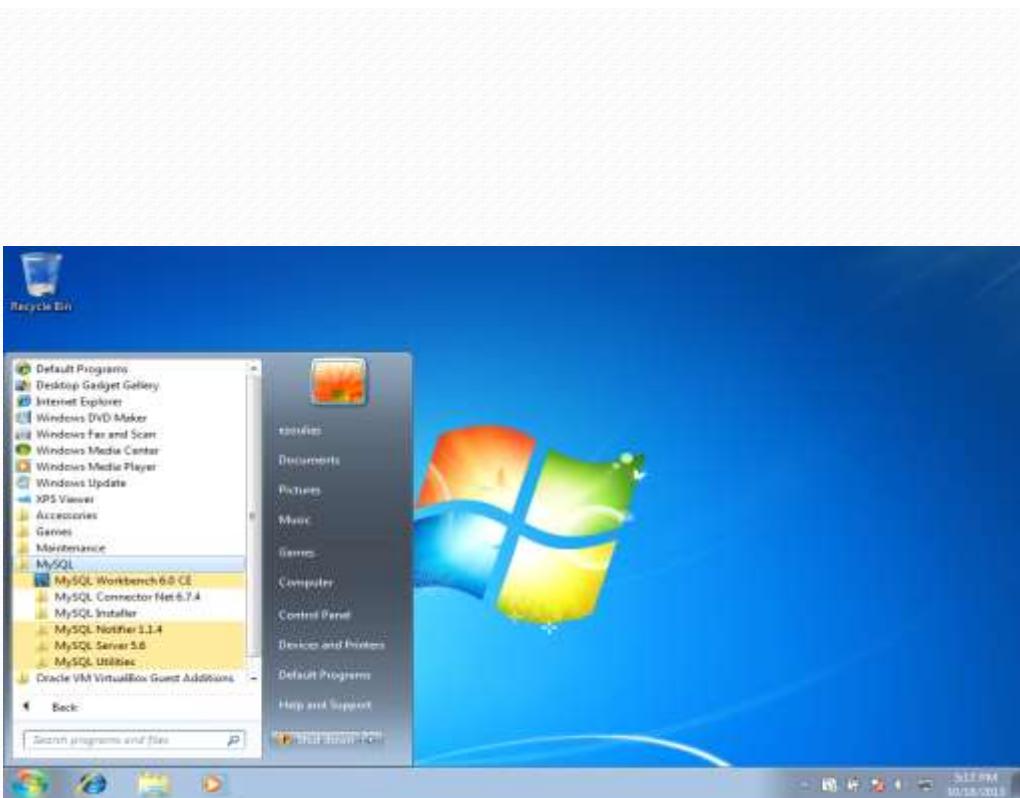
48



49

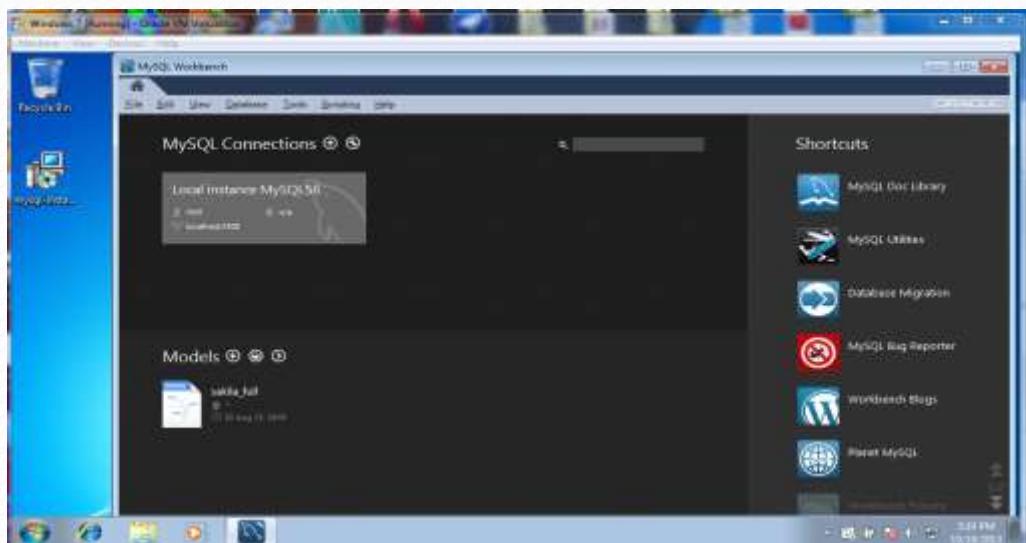


50



51

MySQL Workbench



52

20. ΜΕΛΕΤΗ ΠΕΡΙΠΤΩΣΗΣ

Σκοπός και στόχοι ενότητας

Σκοπός αυτής της ενότητας είναι η εμπέδωση γνώσεων και δεξιοτήτων που αποκτήθηκαν κατά τη διάρκεια του εκπαιδευτικού προγράμματος μέσα από ένα αναλυτικό παράδειγμα σχετικό με το χώρο της Δημόσιας Διοίκησης (Π.Σ. ΜΙΣΘΟΔΟΣΙΑΣ).

Πιο συγκεκριμένα, μετά το τέλος αυτής της ενότητας, οι επιμορφωμένοι θα έχουν ολοκληρώσει μία σειρά επαναληπτικών ασκήσεων σχετικά με:

- το σχεσιακό μοντέλο,
- τη δημιουργία βάσης δεδομένων,
- την εισαγωγή σχεσιακού μοντέλου και εισαγωγή δεδομένων,
- τη τροποποίηση στους πίνακες του σχεσιακού μοντέλου – αλλαγή σχεδίασης των πινάκων,
- τη δημιουργία ερωτημάτων διαφόρων κατηγοριών σχετικά με:
 - ανάκτηση πληροφοριών,
 - ανάκτηση σύνθετων πληροφοριών,
 - εισαγωγή και ενημέρωση πληροφοριών,
 - διαγραφή πληροφοριών,
- Τη δημιουργία χρήστη και απόδοση δικαιωμάτων.

Εισαγωγή

Στην υπηρεσία σας έχει υλοποιηθεί (με εσωτερική ανάπτυξη) μηχανογραφική εφαρμογή για τη μισθοδοσία των υπαλλήλων, βασιζόμενη στο σχεσιακό μοντέλο και στη χρήση του ΣΔΒΔ της MySQL.

Για τη δημιουργία της βάσης δεδομένων έχουν ληφθεί υπόψη οι ακόλουθες προδιαγραφές (να αναφερθεί ότι η συγκεκριμένη μελέτη περίπτωσης έχει μόνο εκπαιδευτικούς σκοπούς καλύπτοντας ένα μικρό μέρος των λειτουργιών ενός συστήματος μισθοδοσίας σε ένα δημόσιο φορέα):

1. Ένας υπάλληλος ανήκει σε μία μόνο κατηγορία υπαλλήλων (π.χ. ΠΕ, ΤΕ , ΔΕ, ΥΕ) ενώ σε μία κατηγορία ανήκουν ένας ή περισσότεροι υπάλληλοι.

2. Ένας υπάλληλος ανήκει σε ένα μόνο κλάδο υπαλλήλων (π.χ. Διοικητικού, Πληροφορικής κλπ) ενώ σε ένα κλάδο ανήκουν ένας ή περισσότεροι υπάλληλοι.
3. Ένας υπάλληλος ανήκει σε ένα μόνο ασφαλιστικό φορέα (π.χ. Δημόσιο, ΕΤΑΑΤΣΜΕΔΕ κλπ) ενώ σε έναν ασφαλιστικό φορέα ανήκουν ένας ή περισσότεροι υπάλληλοι.
4. Ένας υπάλληλος ανήκει σε ένα μόνο μισθολογικό κλιμάκιο συγκεκριμένης κατηγορίας και βαθμού του (π.χ. MK1 ΠΕ βαθμού B) ενώ σε ένα μισθολογικό κλιμάκιο συγκεκριμένης κατηγορίας και βαθμού ανήκουν ένας ή περισσότεροι υπάλληλοι.
5. Ένας υπάλληλος λαμβάνει ένα η περισσότερα επιδόματα ενώ ένα επίδομα δίνεται σε ένα ή περισσότερους υπαλλήλους

Σχεσιακό Μοντέλο

Η βάση δεδομένων **payroll** που σας έχει δοθεί περιλαμβάνει έτοιμους τους απαραίτητους πίνακες και τις συσχετίσεις των παραπάνω οντοτήτων σύμφωνα με το σχεσιακό μοντέλο (βλ. Εικόνα 123).

Πιο συγκεκριμένα περιλαμβάνονται οι ακόλουθοι πίνακες:

categories (κατηγορίες εκπαίδευσης)

categories
id_category INT(11)
dscr_category VARCHAR(50)

branches (κλάδοι)

branches
id_branch INT(11)
dscr_branch VARCHAR(50)

insurances (ασφαλιστικά ταμεία)

insurances
id_insurance INT(11)
dscr_insurance VARCHAR(30)
pension_reservation_percentage INT(11)
health_reservation_percentage INT(11)

salary_levels (μισθολογικά κλιμάκια)

salary_levels
id_salary_level INT(11)
salary_level VARCHAR(1)
basic_salary FLOAT
id_grade INT(11) (FK)

benefits (επιδόματα)

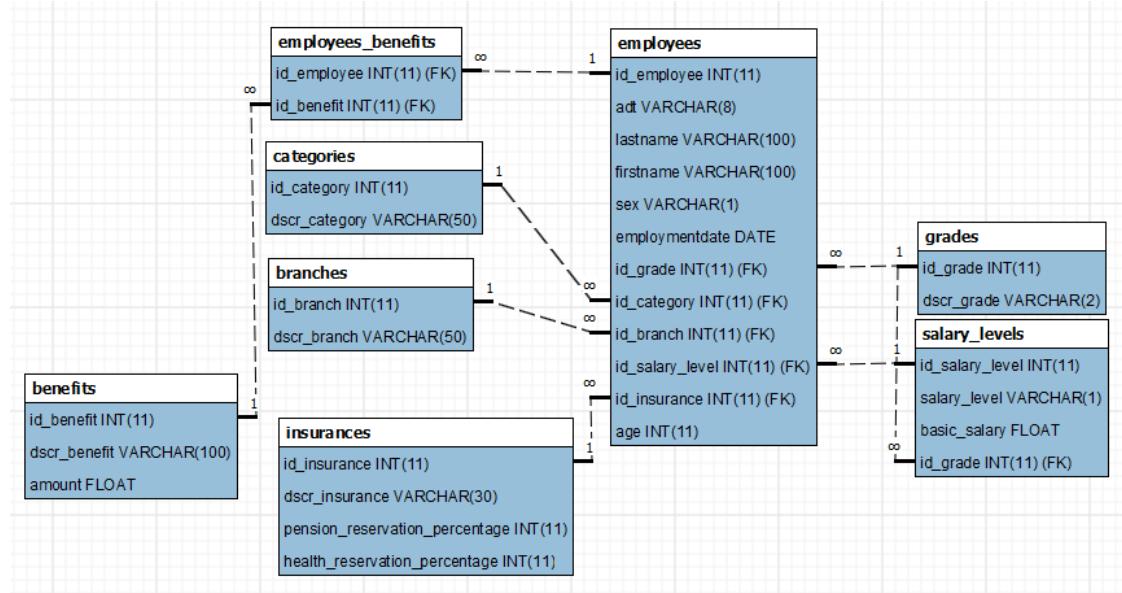
benefits
id_benefit INT(11)
dscr_benefit VARCHAR(100)
amount FLOAT

employees (υπάλληλοι)

employees
id_employee INT(11)
adt VARCHAR(8)
lastname VARCHAR(100)
firstname VARCHAR(100)
sex VARCHAR(1)
employmentdate DATE
id_grade INT(11) (FK)
id_category INT(11) (FK)
id_branch INT(11) (FK)
id_salary_level INT(11) (FK)
id_insurance INT(11) (FK)
age INT(11)

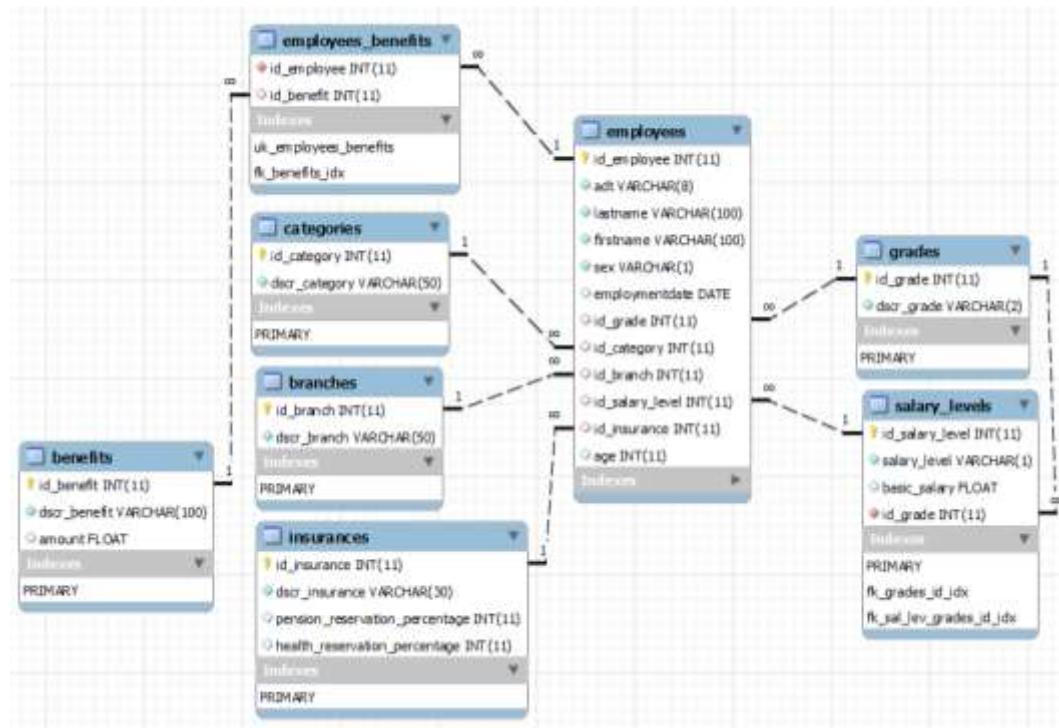
employees_benefits (επιδόματα υπαλλήλων)

employees_benefits
id_employee INT(11) (FK)
id_benefit INT(11) (FK)



Εικόνα 123 Σχεσιακό διάγραμμα της ΒΔ payroll

Στη συνέχεια, ακολουθεί η περιγραφή των πεδίων των παραπάνω πινάκων που περιλαμβάνει: το όνομα, τον τύπο, το μέγεθος και τα χαρακτηριστικά ακεραιότητας τους (π.χ. πρωτεύοντος κλειδιού, ξένου κλειδιού)



Εικόνα 124 Σχεσιακό διάγραμμα της ΒΔ payroll με ένδειξη των primary & foreign keys.

Δεδομένα Πινάκων

Σημειώνεται ότι το SQL script περιλαμβάνεται η δημιουργία του σχήματος της ΒΔ μαζί με ενδεικτικά δεδομένα για τη ΒΔ payroll.

Ασκήσεις

Άσκηση 1η (Πίνακες – Αλλαγή σχεδίασης)

- 1) Να δημιουργηθεί βάση δεδομένων **payroll** τύπου **InnoDB** με **collation utf8_unicode_ci**
 - 2) Να δημιουργήσετε χρήστη **payroll_user** με κωδικό πρόσβασης **12345** και δικαιώματα σύνδεσης από το τοπικό μηχάνημα. Να αποδοθούν πλήρη δικαιώματα πρόσβασης μόνο για το σχήμα **payroll**.
 - 3) Να εκτελέσετε το script (**payroll_srcipt.sql**) που περιλαμβάνει την δημιουργία του σχεσιακού μοντέλου και την εισαγωγή δεδομένων στην βάση **payroll**.
 - 4) Να δημιουργηθεί με χρήση του Workbench το EER διάγραμμα του σχήματος της βάσης **payroll**.
 - 5) Να δημιουργηθεί στο σχήμα **payroll**:
 - a) πίνακας **languages** με γνωρίσματα:

- i) id_language τύπου int που θα αποτελεί και το πρωτεύον κλειδί του πίνακα
 - ii) dscr_language τύπου αλφαριθμητικού 100 χαρακτήρων
- 6) Να τροποποιήσετε τον πίνακα employees εισάγοντας γνώρισμα id_native_language που αναπαριστά τον κωδικό της μητρικής γλώσσας του υπαλλήλου. Να δημιουργηθεί ξένο κλειδί με όνομα fk_languages_id που αναφέρεται στον πίνακα languages.

Άσκηση 2η (Ερωτήματα)

- 1) Να δημιουργηθεί ερώτημα που να εμφανίζει τους υπαλλήλους (firstname, lastname, adt, age) ηλικίας μεγαλύτερης από 35 χρονών ταξινομώντας τους αλφαριθμητικά με επώνυμο και όνομα.
- 2) Να δημιουργηθεί ερώτημα που να εμφανίζει τους άντρες υπαλλήλους (firstname, lastname, adt, age) (θεωρήστε την παραδοχή του μοντέλου ότι sex = 0: άντρας)
- 3) Να δημιουργηθεί ερώτημα το οποίο εμφανίζει τους υπαλλήλους με όνομα που αρχίζει από Π.
- 4) Το τμήμα προσωπικού θέλει να αποστέλλει ευχετήριο δώρο σε κάθε εργαζόμενο. Για να οργανώσει αυτή τη διαδικασία θέλει να ξέρει ποια είναι τα ονόματα που υπάρχουν μεταξύ των υπαλλήλων. Να δημιουργηθεί ερώτημα για το Τμήμα Προσωπικού ώστε να εξυπηρετηθεί η ανάγκη.
- 5) Για να γίνει πρόβλεψη της προμήθειας δώρων για την ανάγκη της ερώτησης 3 το τμήμα προσωπικού θέλει να ξέρει και ποιος είναι ο αριθμός των υπαλλήλων σε κάθε όνομα. Να δημιουργηθεί ερώτημα για το Τμήμα Προσωπικού ώστε να εξυπηρετηθεί η ανάγκη.
- 6) Να δημιουργηθεί ερώτημα το οποίο να εμφανίζει τους υπαλλήλους (firstname, lastname, age, dscr_category) κατηγορίας ΠΕ και ηλικίας μεγαλύτερης του 40.
- 7) Να δημιουργηθεί ερώτημα το οποίο εμφανίζει το πλήθος των υπαλλήλων ανά κλάδο.
- 8) Να δημιουργηθεί ερώτημα το οποίο να εμφανίζει τους κλάδους που έχουν πλήθος υπαλλήλων μεγαλύτερο του 5. Να εμφανίζεται ο κωδικός του κλάδου, ο κλάδος και το πλήθος των υπαλλήλων στον κλάδο.
- 9) Να δημιουργηθεί ερώτημα με χρήση ομαδοποίησης το οποίο να εμφανίζει τον κωδικό του υπαλλήλου και το συνολικό ποσό αποδοχών από επιδόματα που λαμβάνει και για όλους τους υπαλλήλους.

- 10) Να δημιουργηθεί ερώτημα το οποίο να εμφανίζει τον κωδικό του υπαλλήλου ή τους κωδικούς των υπαλλήλων με τον μεγαλύτερο άθροισμα αποδοχών προερχόμενο από επιδόματα.
- 11) Να δημιουργηθεί ερώτημα το οποίο εμφανίζει ανά κωδικό κλάδου τους υπαλλήλους (firstname, lastname, adt, age) που έχουν το μέγιστο μισθό στον κλάδο και αν αυτός είναι πάνω από 1500.
- 12) Να δημιουργηθεί ερώτημα το οποίο εμφανίζει τους υπαλλήλους (firstname, lastname, adt, age) που έχουν βαθμό Α,Β,Γ.
- 13) Να δημιουργηθεί ερώτημα το οποίο εμφανίζει τους υπαλλήλους (firstname, lastname, dscr_branch) που δεν ανήκουν στους κλάδους ‘ΔΙΟΙΚΗΤΙΚΟΥ’ και ‘ΟΙΚΟΝΟΜΙΚΟΥ’.
- 14) Να δημιουργηθεί ερώτημα το οποίο εμφανίζει τους υπαλλήλους (firstname, lastname, adt, age) που έχουν ηλικία μεγαλύτερη από την ηλικία του ΠΑΠΑΔΑΚΗ ΑΝΔΡΕΑ.
- 15) Να δημιουργηθεί ερώτημα το οποίο εμφανίζει τους υπαλλήλους (firstname, lastname, adt, age) που έχουν ηλικία μεγαλύτερη από όλες τις ηλικίες των υπαλλήλων του κλάδου ‘ΔΙΟΙΚΗΤΙΚΟΥ’.
- 16) Γράψτε ερώτημα που επιστρέφει το τμήμα με το μεγαλύτερο άθροισμα μισθών.
- 17) Γράψτε κατάλληλη SQL εντολή για να εισάγεται έναν νέο υπάλληλο με βάση τα ακόλουθα στοιχεία:
- ΑΔΤ: X132456
 - Όνοματεπώνυμο: Ιωάννης Νέστος
 - Ημερομηνία Πρόσληψης: 1/3/2014
 - Βαθμού Ε
 - Κλιμακίου (salary_level) 1
 - Κλάδου ΜΗΧΑΝΙΚΩΝ
 - Ταμείου ασφάλισης ΕΤΑΑ
 - Ηλικία 29
- 18) Γράψτε κατάλληλη SQL εντολή για να ενημερώσετε το κλιμάκιο (salary_level) 2 του Βαθμού Δ, ώστε να είναι επιπέδου 3 θεωρώντας ότι έχουν συμπληρωθεί οι απαραίτητες προϋποθέσεις.
- 19) Δημιουργείστε ένα view για το ερώτημα 13 με όνομα `oldest_administration`.
- 20) Γράψτε κατάλληλη SQL εντολή και διαγράψτε τον εργαζόμενο με όνομα ΝΙΚΟΛΑΟΥ ΣΤΥΛΙΑΝΗ.

- 21) Δημιουργείστε ένα store procedure το οποίο να εκτελεί το ερώτημα 12 για οποιοδήποτε ζεύγος επώνυμο και όνομα (lastname, firstname) του δώσουμε κατά την κλήση του.
- 22) Εξάγετε ένα αντίγραφο ασφαλείας της τελικής ΒΔ στην τελική της κατάσταση με όνομα αρχείου ekddDB.sql, περιλαμβάνοντας και τις procedures και τα δεδομένα.

ΑΠΑΝΤΗΣΕΙΣ

Άσκηση 1η (Απαντήσεις)

- 1) Μέσα από το Workbench δημιουργία βάσης με τις συγκεκριμένες επιλογές, όνομα **payroll** τύπου **InnoDB** με **collation utf8_unicode_ci**, να δοθεί έμφαση και στο αυτόματα παραγόμενο query.
- 2) Μέσα από το Workbench με τις ανάλογες τιμές στις επιλογές, να δοθεί έμφαση και στο αυτόματα παραγόμενο query.

ή

Με τις εντολές

1. CREATE USER 'payroll_user'@'localhost' IDENTIFIED BY '12345'
 2. GRANT ALL ON payroll.* TO 'payroll_user' @'localhost';
- 3) Εκτέλεση του payroll_srcipt.sql μέσα από το Workbench.
 - 4) Χρήση του Workbench.
 - 5) Να δημιουργηθεί στο σχήμα payroll:
 - a) πίνακας **languages** :
CREATE TABLE `languages` (
`id_language` int(11) NOT NULL AUTO_INCREMENT,
`dscr_language` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
PRIMARY KEY (`id_language`)
)

- 6) ALTER TABLE `employees`
ADD COLUMN `id_native_language` INT NULL ,
ADD CONSTRAINT `fk_languages_id`
FOREIGN KEY (`id_native_language`)
REFERENCES `payroll`.`languages`(`id_language`)
ON DELETE NO ACTION
ON UPDATE NO ACTION;

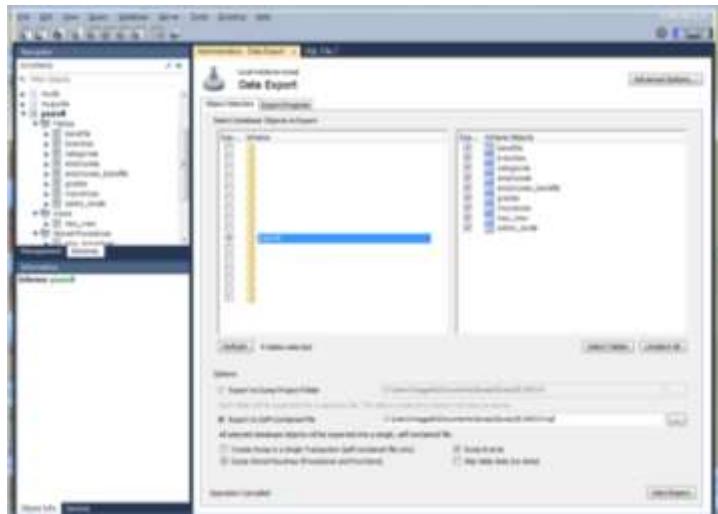
Άσκηση 2η (Απαντήσεις)

- 1) select firstname, lastname, adt, age from employees where age > 35 order by lastname, firstname
- 2) select firstname, lastname, adt, age from employees where sex='0'
- 3) select * from employees where lastname like 'Π%'
- 4) select distinct firstname from employees

- 5) select firstname, count(*) from employees
group by firstname
- 6) select e.firstname, e.lastname, e.age,c.dscr_category from employees e, categories c where e.id_category=c.id_category and e.age > 40 and c.dscr_category='ΠΕ'
- 7) select id_branch, count(*) from employees
group by id_branch
- 8) select b.id_branch,b.dscr_branch, count(*) from employees e, branches b
where e.id_branch=b.id_branch
group by e.id_branch
having count(*)>5
- 9) select e.id_employee, sum(amount)
from employees_benefits e, benefits b
where e.id_benefit=b.id_benefit
group by e.id_employee
- 10) select e.id_employee, sum(amount)
from employees_benefits e, benefits b
where e.id_benefit=b.id_benefit
group by e.id_employee
- 11) SELECT e.id_branch, e.lastname, e.firstname, s.basic_salary FROM employees e,
salary_levels s
where e.id_salary_level = s.id_salary_level
group by e.id_branch
having max(s.basic_salary) > 1500
- 12) SELECT e.id_branch, e.lastname, e.firstname, g.dscr_grade FROM employees e, grades g
where e.id_grade = g.id_grade
and g.dscr_grade in ('A', 'B', 'T')
- 13) SELECT e.lastname, e.firstname, b.dscr_branch FROM employees e, branches b
where b.id_branch = e.id_branch
and b.dscr_branch not in ('ΔΙΟΙΚΗΤΙΚΟΥ', 'ΟΙΚΟΝΟΜΙΚΟΥ')
- 14) SELECT lastname, firstname, age FROM employees e
where age > (select age from employees where lastname='ΠΑΠΑΔΑΚΗΣ' and
firstname='ΑΝΔΡΕΑΣ')

15) SELECT * FROM employees e where e.age > ALL(select age from employees e INNER JOIN branches b on e.id_branch=b.id_branch WHERE dscr_branch='ΔΙΟΙΚΗΤΙΚΟΥ')
 16) select e.id_employee, sum(amount)
 from employees_benefits e, benefits b
 where e.id_benefit=b.id_benefit
 group by e.id_employee
 having sum(amount)>=ALL (select sum(amount)
 from employees_benefits e, benefits b
 where e.id_benefit=b.id_benefit
 group by e.id_employee)
 17) INSERT INTO `payroll`.`employees`
 (`adt`, `lastname`, `firstname`, `sex`, `employmentdate`, `id_grade`, `id_category`,
 `id_branch`, `id_salary_level`, `id_insurance`, `age`)
 VALUES
 ('X132456', 'Νέστος', 'Ιωάννης', 1, '2014-03-01', 5, 1, 4, 13, 3, 29);
 18) UPDATE `payroll`.`employees`
 SET `id_salary_level` = 12
 WHERE id_salary_level =11 ;
 19) CREATE VIEW `oldest_administration` AS
 13)SELECT * FROM employees e where
 e.age > ALL(select age from employees e INNER JOIN branches b on
 e.id_branch=b.id_branch WHERE dscr_branch='ΔΙΟΙΚΗΤΙΚΟΥ')
 20) DELETE from employees where firstname='ΣΤΥΛΙΑΝΗ' and lastname='ΝΙΚΟΛΑΟΥ'
 21) DELIMITER \$\$
 CREATE DEFINER=`root`@`localhost` PROCEDURE `new_procedure`(IN param0
 varchar(100), IN param1 varchar(100))
 BEGIN
 SELECT lastname, firstname, age FROM employees e
 where age > (select age from employees where lastname=param0
 and firstname=param0);
 END
 Στη συνέχεια κλήση της με:
 CALL `payroll`.`new_procedure`("","");

22) Χρησιμοποιείστε τη λειτουργία **data export** από το μενού **server** και επιλέξτε την ΒΔ payroll. Δηλώστε το φάκελο αποθήκευσης στην επιλογή export to self contained file καθώς και το όνομα του sql αρχείου. Στη συνέχεια επιλέξτε start export.



Σχεσιακό διάγραμμα της ΒΔ payroll με ένδειξη των primary & foreign keys.