Coding Factory

Συστήματα Βάσεων Δεδομένων

Περιεχόμενα

1.1 Έννοιες των συστημάτων βάσεων δεδομένων

Βάση Δεδομένων (Database) είναι μία οργανωμένη συλλογή από δεδομένα. Πιο συγκεκριμένα, Βάση Δεδομένων είναι μία συλλογή από εγγραφές, που αποθηκεύονται σε ένα σύστημα υπολογιστή με συστηματικό τρόπο και επιτρέπουν τη διενέργεια πράξεων όπως εισαγωγή, διαγραφή, ενημέρωση και αναζήτηση των εγγραφών, από ένα ειδικό πρόγραμμα που ονομάζεται Σύστημα Διαχείρισης Βάσεων Δεδομένων (DataBase Management System – DBMS) .

Για να μπορεί να υπάρχει τυπική και αποτελεσματική εισαγωγή, τροποποίηση, διαγραφή και ανάκτηση εγγραφών, κάθε εγγραφή δομείται από ένα σύνολο από γνωρίσματα ή χαρακτηριστικές ή ιδιότητες (attributes) ή πεδία (fields).

Η οργάνωση, η περιγραφή και ορισμός των εγγραφών μιας Βάσης Δεδομένων είναι γνωστά ως Σχήμα της Βάσης (Database Schema). Το Σχήμα περιγράφει τα αντικείμενα μιας ΒΔ καθώς και τις σχέσεις μεταξύ τους. Υπάρχουν αρκετοί διαφορετικοί τρόποι περιγραφής του σχήματος μιας ΒΔ, που είναι γνωστοί ως Μοντέλα ΒΔ ή Μοντέλα Δεδομένων.

Το πιο κοινό σε χρήση μοντέλο σήμερα είναι το Σχεσιακό Μοντέλο (Relational Model). Άλλα μοντέλα είναι το Δικτυωτό Μοντέλο (Network Model), το Ιεραρχικό Μοντέλο (Hierarchical Model), το Διαστασιοποιημένο Μοντέλο (Dimensional Model), το Αντικειμενοστραφές-Σχεσιακό Μοντέλο (Object-Relational Model) και το Αντικειμενοστραφές Μοντέλο (Object-Oriented Model).

Πολλοί επιστήμονες στον χώρο των ΒΔ θεωρούν ότι μια συλλογή από δεδομένα αποτελεί ΒΔ, όταν περιλαμβάνει τα ακόλουθα χαρακτηριστικά:

- Τα δεδομένα (και οι σχέσεις μεταξύ τους) τυγχάνουν διαχείρισης που εγγυάται την ακεραιότητά τους (integrity)
- Επιτρέπει ταυτόχρονη πρόσβαση από πολλούς χρήστες (multi-user)
- Ορίζει ένα Σχήμα Βάσης Δεδομένων
- Υποστηρίζει μια γλώσσα για την δημιουργία και τη διαχείριση των δεδομένων
- Απαντά σε ερωτήσεις (queries) διενεργώντας αναζητήσεις

Για παράδειγμα, το τελευταίο χαρακτηριστικό είναι πολύ σημαντικό να επιτελείται από το Σύστημα Διαχείρισης της ΒΔ, διαφορετικά θα υπήρχε ένας απλός εξυπηρετητής αρχείων.

Από θεωρητική άποψη, οι Βάσεις Δεδομένων επιτυγχάνουν έναν αξιοσημείωτο τεχνολογικά ιστορικό διαχωρισμό. Τα παλαιότερα (πριν την δεκαετία του ’80) προγράμματα ή συστήματα διαχείρισης αρχείων ήταν αποκλειστικά συσχετισμένα με ένα σύνολο από ασύνδετα πολλές φορές επίπεδα (flat) αρχεία τα οποία τα διαχειρίζονταν τα ίδια τα προγράμματα. Κάτι τέτοιο σημαίνει πως ο ορισμός, το νόημα και ακολούθως η διαχείριση των αρχείων ήταν πλήρως εξαρτημένη από τα αντίστοιχα προγράμματα, που γνώριζαν τη δομή των αρχείων τους, αφού τα ίδια την είχαν ορίσει.

Ένας τέτοιος μηχανισμός είχε δύο σημαντικά μειονεκτήματα: α) ότι αν άλλαζε κάτι στη δομή των αρχείων θα έπρεπε να μεταγλωττιστούν ξανά όλα τα προγράμματα και β) ότι τα αρχεία μπορούσαν να χρησιμοποιηθούν μόνο από τα προγράμματα που τα είχαν ορίσει.

Ιδιαίτερα το δεύτερο χαρακτηριστικό υπογραμμίζει και κάτι ακόμα, ότι το νόημα των δεδομένων που περιείχαν τα αρχεία, δεν οριζόταν ανεξάρτητα, αλλά από τις εφαρμογές που όριζαν τα δεδομένα των αρχείων. Αν για παράδειγμα μια εφαρμογή, ανακτούσε από ένα αρχείο έναν αριθμό, μόνο η ίδια γνώριζε ότι ο αριθμός αυτός όριζε για παράδειγμα την ηλικία ενός ατόμου (και όχι ένα ποσό ή κάτι άλλο).

Θα ήταν όμως πολύ πιο αποτελεσματικό και αποδοτικό αν ένα σύνολο αρχείων μπορούσαν να το διαχειρίζονται πολλές διαφορετικές εφαρμογές δίχως να πρέπει οι ίδιες να το έχουν ορίζει, αλλά εντούτοις να γνωρίζουν το νόημα των δεδομένων του. Αυτό το χαρακτηριστικό επιτυγχάνεται με τα Συστήματα ΒΔ, τα οποία ενσωματώνουν τόσο τα

δεδομένα, όσο και το νόημα που τα συνοδεύει. Με την κατάλληλη τεκμηρίωση στη ΒΔ, κάθε εφαρμογή που θέλει να την χρησιμοποιήσει, μπορεί να το κάνει, δίχως προηγουμένως να την έχει η ίδια ορίσει.

Από ιστορική σκοπιά, τα πρώτα συστήματα ΒΔ αναπτύχθηκαν την δεκαετία του ’70. Πρώτα, η επιτροπή CODASYL ανέπτυξε το δικτυωτό μοντέλο, ενώ στη συνέχεια η εταιρεία North American Rockwell ανέπτυξε το ιεραρχικό μοντέλο. Το σχεσιακό μοντέλο προτάθηκε το 1970, από τον E. F. Codd. Η βασική κριτική του Codd στα υπάρχοντα μοντέλα ήταν ότι δυσχέραιναν την περιγραφή της αφηρημένης δομής των δεδομένων με περιγραφές του τρόπου φυσικής πρόσβασης σε αυτά.

Για αρκετά έτη το σχεσιακό μοντέλο παρέμεινε μόνο για ακαδημαϊκό ενδιαφέρον, μέχρι το 1976, που αναπτύχθηκαν τα πρώτα ερευνητικά σχεσιακά συστήματα Ingress και System-R, ενώ τα πρώτα εμπορικά συστήματα Oracle και DB2 παρουσιάστηκαν περίπου το 1980.

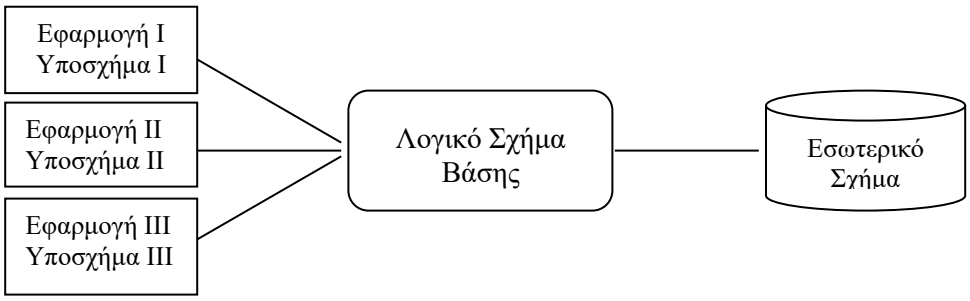
Κατά τη διάρκεια της ένατης δεκαετίας (1980-1990) εμφανίστηκαν κατανεμημένα συστήματα ΒΔ, δίχως μεγάλη επιρροή στην αγορά.

Την δεκαετία του 1990, η κατεύθυνση στο χώρο των ΒΔ στράφηκε προς την αντικειμενοστραφή τεχνολογία. Ο λόγος ήταν πως υπήρχαν εφαρμογές με σύνθετα δεδομένα, όπως χωροχρονικά και πολυμεσικά δεδομένα, τα οποία δεν μπορούσαν να τα διαχειριστούν εύκολα τα σχεσιακά συστήματα. Κάποιες από αυτές τις ιδέες υιοθετήθηκαν από κατασκευαστές συστημάτων ΒΔ, οι οποίοι ενσωμάτωσαν κάποια χαρακτηριστικά στα συστήματά τους δημιουργώντας ένα υβρίδιο σχεσιακού και αντικειμενοστραφούς μοντέλου, το σχεσιακό-αντικειμενοστραφές μοντέλο (object-relational model).

Την τρέχουσα δεκαετία, ενδιαφέρον παρουσιάζει η μεταφορά των συστημάτων ΒΔ στο χώρο του Internet με την υιοθέτηση των Βάσεων Δεδομένων XML. Οι XML ΒΔ προσπαθούν να άρουν τον παραδοσιακό διαχωρισμό ανάμεσα σε έγγραφα και δεδομένα, επιτρέποντας σε όλο το πληροφοριακό υλικό ενός οργανισμού να είναι αποθηκευμένο σε ένα μέρος.

1.2 Αρχιτεκτονική των συστημάτων βάσεων δεδομένων

Στο επίπεδο της Αρχιτεκτονικής προσέγγισης, κάθε σύστημα Διαχείρισης ΒΔ, αποτελείται από τρία μέρη: το λογικό, το εσωτερικό και το εξωτερικό επίπεδο.



Σχήμα 1.1: Αρχιτεκτονική ενός Συστήματος Βάσης Δεδομένων

Το Λογικό Σχήμα αποτελείται από το σύνολο των ορισμών για όλους τους πίνακες, τα πεδία, τις συσχετίσεις και την ακεραιότητα των εγγραφών. Αποτελεί την συνολική εικόνα μιας ΒΔ και μπορεί να οριστεί με τη χρήση της γλώσσας DDL (Data Definition Language), που αποτελεί υποσύνολο της γλώσσας SQL.

Το Λογικό Σχήμα είναι τελείως ανεξάρτητο από το φυσικό επίπεδο και τη φυσική οργάνωση των δεδομένων. Ακόμα και στο επίπεδο της ορολογίας, ενώ στο λογικό επίπεδο (Σχεδιασμός της Βάσης) αναφερόμαστε σε σχέσεις (relations) ή πίνακες (tables), ιδιότητες (attributes) και πλειάδες (tuples), στο φυσικό επίπεδο αναφερόμαστε σε αρχεία (files), πεδία (fields) και εγγραφές (records) αντίστοιχα.

Το Λογικό Υποσχήμα αποτελεί ένα υποσύνολο της Βάσης και αντιστοιχεί σε ένα σύνολο αντικειμένων της ΒΔ, που χρησιμοποιεί ένας χρήστης ή μία εφαρμογή. Για παράδειγμα μία επιχείρηση που αποτελείται από τρία τμήματα, το οικονομικό τμήμα, το τμήμα της διαφήμισης και το τμήμα της αποθήκης μπορεί να χρησιμοποιεί ένα Λογικό

Σχήμα (Global Schema) και τρία Λογικά Υποσχήματα, ένα για κάθε επιχειρησιακό τμήμα. Με τον τρόπο αυτό κάθε τμήμα χρησιμοποιεί το κομμάτι της ΒΔ, που το αφορά μειώνοντας την πολυπλοκότητα της Βάσης, τις μη αποδοτικές αλληλοσυσχετίσεις και αυξάνοντας την χρήση και την ειδίκευση στο πληροφοριακό μοντέλο που το αφορά. Αν μάλιστα, η ΒΔ ήταν και φυσικά κατανεμημένη, τότε θα μπορούσε να μειωθεί και ο φόρτος της, αυξάνοντας παράλληλα τις επιδόσεις.

Τα Λογικά Υποσχήματα είναι γνωστά και ως Όψεις (Views) και η γλώσσα SQL δίνει τη δυνατότητα ορισμού όψεων.

Το εσωτερικό σχήμα αφορά την φυσική οργάνωση των δεδομένων και δίνει έμφαση σε μηχανισμούς ορισμού δομών δεδομένων (B-δένδρα, B+ δένδρα, κλπ.), σε μηχανισμούς προσπέλασης των φυσικών αρχείων (κλείδωμα εγγραφών, πολλαπλή προσπέλαση, κλπ.), μηχανισμούς ασφάλειας και γενικά σε μηχανισμούς αποδοτικής πρόσβασης δεδομένων από αποθηκευτικούς χώρους.

Ο πιο γνωστός μηχανισμός οργάνωσης των εγγραφών στο φυσικό επίπεδο είναι οι δομή των B-δένδρων, μιας δομής που επιτρέπει αναζήτηση σε χρόνο ανάλογο του ύψους του B-δένδρου, $\log_{bf}n$, (και όχι του πλήθους των εγγραφών του), όπου bf (branching factor) είναι ο βαθμός του B-δένδρου (δηλ. το μέγιστο πλήθος των απογόνων κάθε κόμβου), ενώ συνήθως η συσχέτιση των λογικών εγγραφών με τις αντίστοιχες εγγραφές στο δίσκο γίνεται με το μηχανισμό του hashing, μιας μαθηματικής συνάρτησης δηλαδή, που συσχετίζει με μοναδικό τρόπο λογικές εγγραφές με διευθύνσεις μνήμης ή δίσκου.

Για παράδειγμα, αν σε ένα αρχείο με ευρετήριο οργανωμένο σε μορφή B-δένδρου βαθμού δέκα (10), υπάρχουν 10^6 εγγραφές, τότε μία τυχαία εγγραφή μπορεί να ανευρεθεί στην χειρότερη περίπτωση σε έξι (6) αναζητήσεις και στη συνέχεια σε ένα μόνο βήμα (hash table) να οδηγηθεί στη φυσική εγγραφή στο δίσκο.

1.3 Δομές και Μοντέλα Δεδομένων

Ένα μοντέλο δεδομένων είναι ένας οργανωμένος τρόπος ορισμού των δομών δεδομένων μιας Βάσης καθώς και των συσχετίσεων μεταξύ των τύπων των δομών, δηλαδή

του Λογικού Σχήματος της Βάσης, ώστε να μπορεί αυτό με τυπικό τρόπο να μετατραπεί στο αντίστοιχο εσωτερικό σχήμα.

Κατά τη διαδικασία ανάλυσης του συστήματος μιας εμπορικής ή άλλης εφαρμογής (ανάλυση απαιτήσεων) καταγράφονται τα αρχεία και τα δεδομένα της επιχείρησης και σχηματίζεται, σε ένα πρώτο βήμα, το εννοιολογικό μοντέλο δεδομένων του οργανισμού, το οποίο στη συνέχεια και μετά από μία διαδικασία τυποποίησης (κανονικοποίηση, κλπ) μπορεί να εισαχθεί στο Σύστημα Διαχείρισης ΒΔ με τη χρήση μιας κατάλληλης τυπικής γλώσσας ορισμού δεδομένων, όπως η DDL (SQL).

Για τη σχεδίαση του Λογικού Μοντέλου Δεδομένων, υπάρχουν διάφορες προσεγγίσεις που διαφέρουν ως προς την μεθοδολογία και τους τύπους δομών και συσχετίσεων που χρησιμοποιούνται. Οι πιο βασικές προσεγγίσεις αναλύονται στη συνέχεια.

1.3.1 Το Ιεραρχικό Μοντέλο

Το ιεραρχικό μοντέλο οργανώνει τα δεδομένα χρησιμοποιώντας δύο βασικές δομές, τις εγγραφές (records) και τα σύνολα (sets). Οι εγγραφές περιέχουν πεδία (fields), ενώ τα σύνολα ορίζουν τις συσχετίσεις ένα-προς-ένα ή ένα-προς-πολλά. Η συσχέτιση πολλά-προς-ένα δεν είναι επιτρεπτή στο ιεραρχικό μοντέλο. Γενικά, το ιεραρχικό μοντέλο ακολουθεί την δενδρική δομή οργάνωσης των πληροφοριών, όπου για κάθε κόμβο υπάρχει ένας γονέας (ιδιοκτήτης) και πολλοί απόγονοι. Ένας κόμβος σε μια ιεραρχία μπορεί να θεωρηθεί ως ένας τύπος εγγραφής, ο οποίος μπορεί να είναι ιδιοκτήτης ενός ή περισσοτέρων διαφορετικών τύπων εγγραφών με τις οποίες συνδέεται με τη χρήση συσχετίσεων.

Στο ιεραρχικό μοντέλο μπορεί να υπάρχει σημαντικός βαθμός πλεονασμού δεδομένων από τη στιγμή που δεν υπάρχει κάποια τυπική διαδικασία που να εγγυάται το αντίθετο. Επίσης, η συνέπεια της Βάσης, δηλαδή η διάδοση αλλαγών σε όλο το μήκος της ιεραρχίας και στις εγγραφές που επηρεάζονται, ανήκει στον προγραμματιστή.

1.3.2 Το Δικτυωτό Μοντέλο

Το Δικτυωτό μοντέλο διαφέρει από το Ιεραρχικό στο ότι μπορεί να συμπεριλάβει και σχέσεις πολλά-προς-ένα, δηλαδή ένας απόγονος να έχει πάνω από ένα γονέα-ιδιοκτήτη. Αυτό το χαρακτηριστικό βοηθάει στην μείωση του πλεονασμού γιατί μπορεί ένας κόμβος να αναφερθεί άμεσα σε οποιονδήποτε άλλο κόμβο του συνόλου των κόμβων. Επειδή όμως το παραπάνω μοντέλο είναι πολύ περίπλοκο και δεν μπορεί να διαχειριστεί εύκολα, η επιτροπή CODASYL ορίζει στο Δικτυωτό Μοντέλο, ότι μόνο ένας κόμβος μπορεί να οριστεί ως ιδιοκτήτης και οι κόμβοι με τους οποίους συνδέεται ο ιδιοκτήτες να ορισθούν ως μέλη (members) ενός συνόλου. Στην περίπτωση που κάποιο μέλος πρέπει να συνδεθεί και με άλλο ιδιοκτήτη, τότε δημιουργείται άλλο σύνολο με σημαντικό όμως βαθμό επικάλυψης, ‘άρα και πλεονασμού.

Οι λειτουργίες του Δικτυωτού μοντέλου είναι λειτουργίες “πλοήγησης”: η πλοήγηση ανάμεσα στις εγγραφές γίνεται ξεκινώντας από μία “τρέχουσα θέση” και ακολουθώντας τις συσχετίσεις από εγγραφή σε εγγραφή. Οι εγγραφές μπορούν επίσης να εντοπίζονται με τη χρήση κλειδιών, δηλαδή μοναδικών τιμών σε ένα συγκεκριμένο πεδίο δεδομένων.

Αν και δεν είναι αναγκαίο χαρακτηριστικό του μοντέλου, οι δικτυωτές ΒΔ συνήθως υλοποιούν τις συσχετίσεις συνόλων με χρήση δεικτών (pointers) που “δείχνουν” άμεσα στη διεύθυνση της εγγραφής στο δίσκο. Το χαρακτηριστικό αυτό παρέχει πολύ καλές επιδόσεις αναζήτησης σε βάρος όμως των λειτουργιών φόρτωσης και αναδιοργάνωσης της Βάσης.

1.3.3 Το Σχεσιακό Μοντέλο

Το Σχεσιακό Μοντέλο παρουσιάστηκε αρχικά από τον E. F. Codd το 1970. Πρόκειται για ένα μαθηματικό μοντέλο που βασίζεται στη σχεσιακή άλγεβρα (relational algebra).

Στη σχεσιακή ορολογία “σχέση” (relation) είναι ένα σύνολο από “ιδιότητες” (attributes) που σχηματίζουν μία λογική οντότητα. “Πλειάδα” είναι μία ακολουθία από “ιδιότητες”, που παίρνουν τιμές από προκαθορισμένα σύνολα τιμών ή πεδία ορισμού. “Πεδίο ορισμού” είναι, όπως στα μαθηματικά, ένα προκαθορισμένο σύνολο τιμών ή τύπος δεδομένων, στο οποίο ανήκει η κάθε “ιδιότητα”.

Τα εμπορικά προϊόντα που αποτελούν υλοποιήσεις του σχεσιακού μοντέλου, στην πραγματικότητα υλοποιούν μία προσέγγιση του μοντέλου, που ορίστηκε από τον Codd. Οι δομές δεδομένων σε αυτά τα προϊόντα ονομάζονται "πίνακες» αντί για "σχέσεις" με βασική διαφορά ότι οι πίνακες μπορούν να περιέχουν διπλές γραμμές (εγγραφές) και να συμπεριφέρνονται ως "ταξινομημένοι" (ordered). Επίσης, στην ορολογία των εμπορικών συστημάτων μία πλειάδα αναφέρεται ως "εγγραφή" η οποία μπορεί να περιέχει ένα σύνολο από "πεδία" (fields), όπως ονομάζονται οι "ιδιότητες" του μοντέλου του Codd. Η ίδια κριτική ασκείται από τον Codd και για την γλώσσα SQL, που αποτελεί τη βασική γλώσσα προγραμματισμού σε σχεσιακές ΒΔ, γύρω από το κατά πόσο μπορεί να ονομάζεται σχεσιακή. Αλλά το γεγονός είναι πως τα εμπορικά προϊόντα έχουν καθιερώσει μία de facto ορολογία, που συνήθως ακολουθείται στην κοινή χρήση των όρων.

Μία Σχεσιακή ΒΔ μπορεί να περιέχει πολλούς πίνακες, που κάθε ένας αποτελεί –με τεχνικούς όρους- ένα δυσδιάστατο πίνακα στοιχείων, όπου όλα τα μέλη μιας στήλης έχουν τιμές που ανήκουν στον ίδιο τύπο δεδομένων και όλα τα μέλη μιας γραμμής σχετίζονται το ένα με το άλλο σχηματίζοντας μία εγγραφή. Οι στήλες ενός πίνακα μπορεί να έχουν διάφορους τύπους δεδομένων, όπως χαρακτήρες, ακέραιοι, ημερομηνία, κλπ.

Οι σχέσεις ανάμεσα στους πίνακες δεν ορίζονται άμεσα με χρήση δεικτών ή άλλων άμεσων μηχανισμών διευθυνσιοδότησης αλλά με τη χρήση κλειδιών. Μοναδικό Κλειδί (Unique Key) είναι μία συλλογή από ένα ή περισσότερα πεδία ενός πίνακα που μπορούν να προσδιορίσουν κάθε εγγραφή του πίνακα με μοναδικό τρόπο. Πρωτεύον Κλειδί (Primary Key) είναι ένα από τα Μοναδικά Κλειδιά το οποίο προτιμάται για τον μοναδικό προσδιορισμό των εγγραφών. Αν το πρωτεύον κλειδί περιλαμβάνει ένα μόνο πεδίο του πίνακα τότε ονομάζεται απλό, ενώ αν περιλαμβάνει περισσότερα από ένα πεδία ονομάζεται σύνθετο (complex). Δευτερεύον κλειδί, είναι ένα πεδίο του πίνακα το οποίο χρησιμοποιείται για τον εντοπισμό συνόλων ομοειδών εγγραφών.

Για παράδειγμα, σε ένα πίνακα που περιλαμβάνει τους φορολογούμενους πολίτες μιας χώρας, ως πρωτεύον κλειδί θα μπορούσε να είναι ο Αριθμός Φορολογικού τους Μητρώου και ως δευτερεύον το επίθετό τους. Γενικά, τον ρόλο δευτερευόντων κλειδιών τον αναλαμβάνουν πεδία τα οποία χρησιμοποιούνται συχνά σε αναζητήσεις, γιατί ορίζοντας ένα πεδίο ως κλειδί (είτε πρωτεύον είτε δευτερεύον) το σύστημα Διαχείρισης της ΒΔ δημιουργεί ένα ευρετήριο (index) που επιταχύνει τη λειτουργία της αναζήτησης.

Ένα κλειδί που έχει ένα εξωτερικό και αντικειμενικό νόημα (όπως π.χ. ο αριθμός ταυτότητας, μητρώο, κλπ) στον πραγματικό κόσμο, μπορεί να χρησιμοποιηθεί και ως "φυσικό" πρωτεύον κλειδί σε ένα πίνακα, διαφορετικά μπορεί να χρησιμοποιηθούν ως πρωτεύοντα κλειδιά τεχνητά πεδία (για παράδειγμα "κωδικός") τα οποία μπορεί να λαμβάνουν τυχαίες, αλλά διαφορετικές τιμές.

Οι χρήστες αιτούνται δεδομένων από μια σχεσιακή βάση δεδομένων, αποστέλλοντας μια ερώτηση (query) προς τη βάση σε όρους μιας ειδικής γλώσσας, που ονομάζεται SQL (Structured Query Language).

Σε απάντηση της ερώτησης, η Βάση Δεδομένων αποστέλλει ένα σύνολο αποτελεσμάτων (result set), που είναι μία ακολουθία γραμμών (εγγραφών). Η πιο απλή ερώτηση είναι να επιστραφούν όλες οι εγγραφές ενός πίνακα, αλλά πιο συχνά τα περιεχόμενα ενός πίνακα φιλτράρονται με βάση συγκεκριμένα κριτήρια.

Συχνά επίσης, οι ερωτήσεις προς τη ΒΔ αφορούν δεδομένα από περισσότερους από έναν πίνακες. Σε αυτή την περίπτωση τα δεδομένα από τους πίνακες συγκεντρώνονται σε ένα πίνακα με τον συνδυασμό όλων των δυνατών συνδυασμών από τις γραμμές των πινάκων (καρτεσιανό γινόμενο) και φιλτράρονται ώστε να εξέλθουν μόνο τα στοιχεία της ερώτησης.

1.4 Σχεδιασμός Βάσεων δεδομένων

Σχεδιασμός μιας Βάσης Δεδομένων (Database Design) είναι μια συστηματική διαδικασία καταγραφής και παρουσίασης με τυπικές μεθόδους (διαγράμματα) των δεδομένων μιας εφαρμογής.

Ο σχεδιασμός μιας ΒΔ είναι μια σημαντική διαδικασία (ίσως η σημαντικότερη διαδικασία κατά την ανάπτυξη μιας εφαρμογής) και έχει τους παρακάτω στόχους:

– Να απεικονίσει τυπικά με διαγράμματα οντοτήτων-σχέσεων (Entity-Relationship models – ER Models) την οργάνωση και δομή της βάσης

- Να εξαλείψει τυχόν ανωμαλίες που θα μπορούσαν να προκύψουν σε μα βάση κατά τη φάση εισαγωγής, ενημέρωσης, διαγραφής και αναζήτησης στοιχείων και να εξασφαλίζει την ακεραιότητα και την συνέπεια της βάσης
- Να μειώσει την πολυπλοκότητα της βάσης

Οι δύο βασικές προσεγγίσεις για το σχεδιασμό μιας βάσης είναι:

- Η από-πάνω-προς-τα-κάτω (άνωθεν – top-down)
- Από-κάτω-προς-τα-πάνω (κάτωθεν – bottom-up)

Η πρώτη προσέγγιση ξεκινά με την καταγραφή του εννοιολογικού σχήματος ενός οργανισμού (αρχεία, διαδικασίες, κλπ) και στη συνέχεια με μία διαδικασία αποσύνθεσης και κανονικοποίησης σταδιακά προχωρά προς τον ορισμό των οντοτήτων και των σχέσεων μεταξύ τους, τον ορισμό των ιδιοτήτων των σχέσεων και των τύπων των ιδιοτήτων καθώς και άλλων χαρακτηριστικών τους και τελικά καταλήγει στον σχεδιασμό του λογικού σχήματος της βάσης (ER Model).

Η δεύτερη προσέγγιση ξεκινά με την καταγραφή μεμονωμένων στοιχείων και δεδομένων ενός οργανισμού και στη συνέχεια με διαδικασίες κανονικής σύνθεσης προχωρά προς τον ορισμού του λογικού σχήματος της βάσης.

Συνήθως ο σχεδιασμός μιας βάσης ακολουθεί την πρώτη προσέγγιση που είναι πιο φυσική, ενώ ο δεύτερος τρόπος ακολουθείται συνήθως κατά τη φάση της υλοποίησης της βάσης.

1.4.1 Κανονικοποίηση και Συναρτησιακές Εξαρτήσεις

Κατά τη διαδικασία αποσύνθεσης του εννοιολογικού σχήματος και προσπάθειας ορισμού του λογικού σχήματος θα πρέπει να ακολουθούνται διάφοροι κανόνες που εξετάζουν και βελτιστοποιούν τους τρόπους συσχέτισης των στοιχείων ή ιδιοτήτων της βάσης.

Η διαδικασία αυτή της αποσύνθεσης των εννοιολογικών οντοτήτων μιας βάσης με συγκεκριμένους κανόνες συσχετίσεων των στοιχείων έχει ως σκοπό την εξασφάλιση της ακεραιότητας της βάσης καθώς και την μείωση πλεονασμού της βάσης και ονομάζεται κανονικοποίηση.

Οι τρεις βασικοί τύποι συσχετίσεων που εξετάζονται κατά τη διαδικασία της κανονικοποίησης είναι οι παρακάτω:

Συναρτησιακή εξάρτηση. Λέμε ότι ένα στοιχείο ή ιδιότητα ή γνώρισμα Φ2 είναι συναρτησιακά εξαρτημένο από το Φ1 (Φ1 —> Φ2) όταν κάθε τιμή του Φ2 εξαρτάται αποκλειστικά από το Φ1. Αν το Φ1 είναι σύνθετο γνώρισμα τότε το Φ2 είναι συναρτησιακά εξαρτημένο από το Φ1 όταν κάθε τιμή του Φ2 εξαρτάται από το συνδυασμό των τιμών του Φ1.

Μεταβατική Εξάρτηση. Λέμε ότι ένα γνώρισμα Φ3 είναι μεταβατικά εξαρτημένο από ένα γνώρισμα Φ1, όταν υπάρχει ένα γνώρισμα Φ2 που είναι συναρτησιακά εξαρτημένο από το Φ1, ενώ το Φ3 είναι συναρτησακά εξαρτημένο από το Φ2 (Φ1—> Φ2 —>Φ3).

Εξάρτηση Πολλαπλών Τιμών. Λέμε ότι ένα γνώρισμα Φ1 προσδιορίζει ένα ή περισσότερα γνωρίσματα Φ2 με πολλαπλές τιμές όταν για μία τιμή του Φ1 αντιστοιχούν πολλές τιμές του Φ2. Για παράδειγμα, όταν ένας κωδικός γιατρού (ένας γιατρός) σχετίζεται με πολλούς κωδικούς ασθενών (πολλούς ασθενείς) τότε υπάρχει εξάρτηση πολλαπλών τιμών.

1.4.2 Γνωρίσματα Κλειδιά

Υποψήφιο κλειδί είναι ένα γνώρισμα (απλό) ή ένα σύνολο γνωρισμάτων (σύνθετο υποψήφιο κλειδί) που μπορεί να αναγνωρίσει με μοναδικό τρόπο μια εγγραφή. Στην περίπτωση που το υποψήφιο κλειδί είναι σύνθετο θα πρέπει να αποτελείται από τον ελάχιστο αριθμό ιδιοτήτων που μπορούν να αναγνωρίσουν με μοναδικό τρόπο κάθε εγγραφή και να μην περιέχει πλεονάζοντα γνωρίσματα.

Πρωτεύον κλειδί (Primary Key) είναι το υποψήφιο κλειδί που τελικά επιλέγεται για να χρησιμοποιηθεί ως μοναδικό αναγνωριστικό των πλειάδων (εγγραφών) μιας σχέσης

(πίνακα). Οι τιμές ενός πρωτεύοντος κλειδιού δεν πρέπει να επαναλαμβάνονται. Επίσης, ο ορισμός ενός πρωτεύοντος κλειδιού δημιουργεί αυτόματα ένα ευρετήριο με βάση το πρωτεύον κλειδί (primary index) ώστε οι αναζητήσεις με βάση το πρωτεύον κλειδί να επιταχύνονται σημαντικά.

Δευτερεύον κλειδί (Secondary Key) είναι ένα είναι ένα γνώρισμα ή ένα σύνολο γνωρισμάτων που χρησιμοποιείται όταν σε μια βάση γίνονται αναζητήσεις με γνωρίσματα που δεν είναι πρωτεύοντα κλειδιά. Αν για παράδειγμα σε ένα πίνακα πελατών, το πρωτεύον κλειδί είναι ο κωδικός πελάτη, αλλά οι αναζητήσεις γίνονται συνήθως με το επώνυμο του πελάτη, τότε θα πρέπει να οριστεί το επώνυμο ως δευτερεύον κλειδί, ώστε να επιταχυνθεί η αναζήτηση. Σε τεχνικό επίπεδο ο ορισμός ενός δευτερεύοντος κλειδιού γίνεται με τον ορισμό ενός ευρετηρίου -- CREATE INDEX myIndex ON PELATES (EPWNYMO) -- για το συγκεκριμένο πεδίο (γνώρισμα), ώστε κατά τη διαδικασία αναζήτησης να χρησιμοποιείται το ευρετήριο και να επιταχύνεται η αναζήτηση.

Ξένο κλειδί (Foreign Key) είναι ένα ή περισσότερα πεδία (απλό ή σύνθετο) ενός πίνακα το οποία είναι πρωτεύον κλειδί σε ένα άλλο πίνακα. Οι συνδέσεις ανάμεσα στους πίνακες μιας βάσης υλοποιείται μέσω συσχετίσεων στο πρωτεύον κλειδί ενός πίνακα και του ξένου κλειδιού του συνδεδεμένου πίνακα. Τα ξένα κλειδιά μπορούν να έχουν πολλές εγγραφές με την ίδια τιμή ξένου κλειδιού, που όλες σχετίζονται με μία εγγραφή του συσχετιζόμενου πρωτεύοντος κλειδιού. Επίσης, είναι σημαντικό να διατηρείται η ακεραιότητα της βάσης με ελέγχους των τιμών των ξένων κλειδιών.

Για παράδειγμα, δεν μπορεί ένα ξένο κλειδί να περιέχει μια τιμή για την οποία δεν υπάρχει αντίστοιχο πρωτεύον κλειδί. Επίσης, θα πρέπει να εξασφαλίζεται σε περιπτώσεις αλλαγής της τιμής ενός πρωτεύοντος κλειδιού να αλλάζουν ταυτόχρονα και οι αντίστοιχες τιμές του ξένου κλειδιού, ή σε περιπτώσεις διαγραφής ενός πρωτεύοντος κλειδιού να διαγράφονται ίσως και οι αντίστοιχες εγγραφές με το ξένο κλειδί, ώστε η βάση να είναι συνεπής.

1.4.3 Κανονικές Μορφές (Normal Forms)

Η διαδικασία της κανονικοποίησης περιλαμβάνει πέντε κανονικές μορφές (Normal Forms – NF), δηλαδή πέντε βήματα που μπορούν να ακολουθηθούν, ώστε η βάση να θεωρείται κανονικοποιημένη.

1.4.3.1 Πρώτη κανονική Μορφή (1NF)

Μία σχέση (πίνακας) βρίσκεται στην πρώτη κανονική μορφή, όταν κάθε γνώρισμα είναι απλό ή ατομικό και όχι σύνθετο. Για παράδειγμα, μια σχέση που περιέχει ένα γνώρισμα ΔΙΕΥΘΥΝΣΗ δεν είναι στην πρώτη κανονική μορφή, γιατί το πεδίο ΔΙΕΥΘΥΝΣΗ δεν είναι ατομικό πεδίο, αλλά σύνθετο (περιλαμβάνει την ΟΔΟ, τον ΑΡΙΘΜΟ, κλπ).

1.4.3.2 Δεύτερη Κανονική Μορφή (2NF)

Μία σχέση είναι στη δεύτερη κανονική μορφή όταν είναι στην πρώτη κανονική μορφή και κάθε μη πρωτεύον γνώρισμα είναι συναρτησιακά εξαρτημένο από ολόκληρο το υποψήφιο κλειδί (whole key). Με άλλα λόγια η δεύτερη κανονική μορφή παραβιάζεται όταν υπάρχει έστω ένα μη πρωτεύον γνώρισμα που εξαρτάται συναρτησιακά από κάποιο υποσύνολο του υποψήφιου κλειδιού.

1.4.3.3 Τρίτη κανονική Μορφή (3NF)

Μια σχέση είναι στην τρίτη κανονική μορφή όταν είναι στην δεύτερη κανονική μορφή και κάθε μη πρωτεύον γνώρισμα είναι συναρτησιακά εξαρτημένο μόνο από το υποψήφιο κλειδί, δηλαδή δεν υπάρχει μεταβατική εξάρτηση από μη πρωτεύον γνώρισμα. Με άλλα λόγια η τρίτη κανονική μορφή παραβιάζεται όταν ένα μη πρωτεύον γνώρισμα εξαρτάται συναρτησιακά από κάποιο άλλο μη πρωτεύον γνώρισμα

Σε περιπτώσεις που υπάρχουν περισσότερα από ένα σύνθετα υποψήφια κλειδιά, τότε μια σχέση είναι σε ισχυρή τρίτη κανονική μορφή ή αλλιώς Boyce-Codd Normal Form όταν είναι στην τρίτη κανονική μορφή και επιπλέον δεν υπάρχει συναρτησιακή εξάρτηση ανάμεσα στα υποψήφια κλειδιά. Αν για παράδειγμα μια σχέση περιέχει δύο σύνθετα

υποψήφια κλειδιά (Α, Β) και (Β, Γ) και βρίσκεται στην τρίτη κανονική μορφή αλλά ισχύει Β—>Γ τότε δεν βρίσκεται στην Boyce-Codd μορφή γιατί υπάρχει συναρτησιακή εξάρτηση ανάμεσα σε μέρη των δύο υποψήφιων κλειδιών.

1.4.3.4 Τέταρτη Κανονική Μορφή (4NF)

Μια σχέση είναι στην τέταρτη κανονική μορφή αν είναι στην τρίτη κανονική μορφή και ταυτόχρονα δεν περιέχει εξαρτήσεις πολλαπλών τιμών.

1.4.3.5 Πέμπτη κανονική Μορφή (5NF)

Μία σχέση βρίσκεται στην πέμπτη κανονική μορφή – γνωστή ως συνδετική εξάρτηση (projection-join normal-form PJ/NF), όταν βρίσκεται στην τέταρτη κανονική μορφή και μπορεί να αποσυντεθεί περαιτέρω, δηλαδή να κατασκευαστεί από μικρότερες σχέσεις, που συνολικά να παρέχουν μικρότερο πλεονασμό στη βάση.

Με άλλα λόγια μία σχέση είναι στην πέμπτη κανονική μορφή αν και μόνο αν κάθε συνδετική εξάρτηση υπονοείται μόνο από το πρωτεύον κλειδί της σχέσης. Αν μια σχέση μπορεί να ανακατασκευαστεί από μικρότερες σχέσεις που όλες έχουν το ίδιο πρωτεύον κλειδί, τότε η σχέση είναι στην πέμπτη κανονική μορφή. Επίσης, η πέμπτη κανονική μορφή δεν διαφέρει από την τέταρτη κανονική μορφή εκτός εάν υπάρχει κάποιος περιορισμός συνδετικής εξάρτησης.

1.5 Παράδειγμα Κανονικοποίησης

Έστω ένα Πανεπιστήμιο που αποτελείται από διάφορα επιστημονικά τμήματα και βρίσκεται κατανεμημένο σε πολλές πόλεις της Ελλάδος και διαθέτει σε κάθε πόλη ένα αριθμό αιθουσών, όπου καθηγητές διδάσκουν μαθήματα σε φοιτητές διαφόρων επιστημονικών τμημάτων. Κάθε καθηγητής κάνει συγκεκριμένα μαθήματα σε συγκεκριμένα

τμήματα, συγκεκριμένους φοιτητές και συγκεκριμένες αίθουσες, ένας φοιτητής παρακολουθεί συγκεκριμένα μαθήματα σε συγκεκριμένα αίθουσες και σε μια αίθουσα γίνονται συγκεκριμένα μαθήματα.

Μία αρχική ανάλυση του παραπάνω παραδείγματος δίνει τις παρακάτω σχέσεις:

ΚΑΘΗΓΗΤΕΣ(κωδικός, επώνυμο, όνομα, διεύθυνση, τίτλος, κ_μαθήματος, κ_αίθουσας, κ_φοιτητή, κ_τμήματος)

ΦΟΙΤΗΤΕΣ (κωδικός, όνομα, επώνυμο, διεύθυνση, ηλικία, κ_μαθήματος, κ_αίθουσας, κ_καθηγητή, κ_τμήματος)

ΤΜΗΜΑΤΑ (κωδικός, ονομασία, πόλη, κ_μαθήματος, κ_αίθουσας, κ_καθηγητή, κ_φοιτητή)

ΑΙΘΟΥΣΕΣ (κωδικός, ονομασία, πλήθος_θέσεων, κ_τμήματος, κ_μαθήματος, κ_καθηγητή, κ_φοιτητή)

ΜΑΘΗΜΑΤΑ (κωδικός, ονομασία, κ_τμήματος, κ_αίθουσας, κ_καθηγητή, κ_φοιτητή)

Οι σχέσεις ΚΑΘΗΓΗΤΕΣ και ΦΟΙΤΗΤΕΣ δεν είναι στην πρώτη κανονική μορφή γιατί περιέχουν το πεδίο διεύθυνση που είναι σύνθετο.

Αποσυνθέτουμε το πεδίο διεύθυνση στα πεδία (οδός, αριθμός, ΤΚ) και έχουμε:

ΚΑΘΗΓΗΤΕΣ(κωδικός, επώνυμο, όνομα, οδός, αριθμός, ΤΚ, τίτλος, τμήμα, αίθουσα, μάθημα, φοιτητής)

ΦΟΙΤΗΤΕΣ (κωδικός, όνομα, επώνυμο, οδός, αριθμός, ΤΚ, ηλικία, τμήμα, αίθουσα, μάθημα, καθηγητής)

Όλες οι σχέσεις τώρα είναι στην πρώτη κανονική μορφή. Επίσης είναι στην δεύτερη κανονική μορφή και στην τρίτη κανονική μορφή γιατί κάθε μη-πρωτεύον γνώρισμα είναι συναρτησιακά εξαρτημένο από το πρωτεύον κλειδί και μόνο από το πρωτεύον κλειδί.

Όμως καμία από τις παραπάνω σχέσεις δεν είναι στην τέταρτη κανονική μορφή γιατί περιέχουν συναρτήσεις πολλαπλών τιμών και συγκεκριμένα:

ΚΑΘΗΓΗΤΕΣ: κωδικός -->> κ_μαθήματος, κωδικός -->> κ_αίθουσας, κωδικός -->> κ_φοιτητή, κωδικός -->> κ_τμήματος

ΦΟΙΤΗΤΕΣ: κωδικός -->> κ_μαθήματος, κωδικός -->>κ_αίθουσας, κωδικός -->>κ_καθηγητή, κωδικός -->>κ_τμήματος

ΤΜΗΜΑΤΑ: κωδικός -->> κ_μαθήματος, κωδικός -->>κ_αίθουσας, κωδικός -->>κ_καθηγητή, κωδικός -->> κ_φοιτητή

ΑΙΘΟΥΣΕΣ: κωδικός -->> κ_μαθήματος, κωδικός -->> κ_τμήματος, κωδικός -->>κ_καθηγητή, κωδικός -->> κ_φοιτητή

ΜΑΘΗΜΑΤΑ: κωδικός -->> κ_αίθουσας, κωδικός -->> κ_τμήματος, κωδικός -->>κ_καθηγητή, κωδικός -->> κ_φοιτητή

Επομένως αποσυνθέτουμε όλες τις σχέσεις δημιουργώντας μία ενδιάμεση σχέση ΔΙΔΑΣΚΑΛΙΕΣ (κ_τμήματος, κ_μαθήματος, κ_καθηγητή, κ_αίθουσας, κ_φοιτητή), που έχει σύνθετο πρωτεύον κλειδί που αποτελείται από πεδία ξένα κλειδιά των βασικών πινάκων δηλαδή ουσιαστικά αποτελείται από όλα τα πρωτεύοντα κλειδιά των βασικών σχέσεων.

Έτσι έχουμε τους παρακάτω πίνακες, που είναι στην τέταρτη κανονική μορφή:

ΚΑΘΗΓΗΤΕΣ(κωδικός, επώνυμο, όνομα, οδός, αριθμός, ΤΚ, τίτλος)

ΦΟΙΤΗΤΕΣ (κωδικός, όνομα, επώνυμο, οδός, αριθμός, ΤΚ, ηλικία)

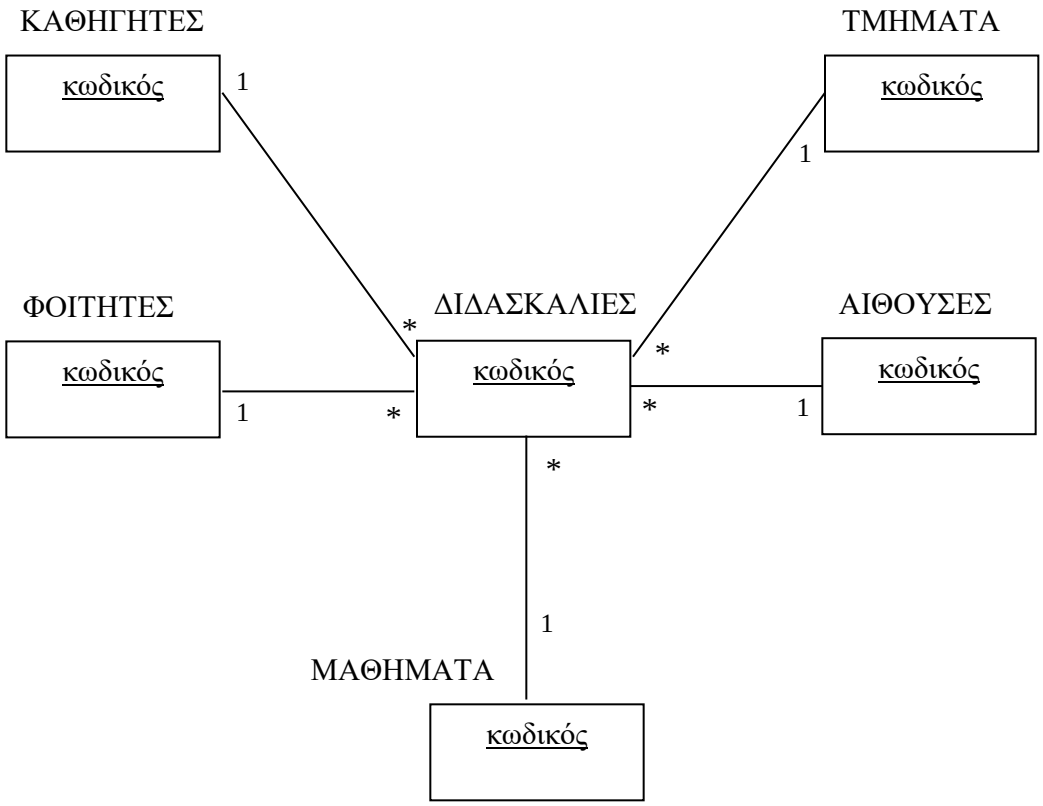
ΤΜΗΜΑΤΑ (κωδικός, ονομασία, πόλη)

ΑΙΘΟΥΣΕΣ (κωδικός, ονομασία, πλήθος_θέσεων)

ΜΑΘΗΜΑΤΑ (κωδικός, ονομασία)

ΔΙΔΑΣΚΑΛΙΕΣ (κ_τμήματος, κ_μαθήματος, κ_καθηγητή, κ_αίθουσας, κ_φοιτητή)

Διαγραμματικά, οι παραπάνω σχέσεις μπορούν να απεικονιστούν σε ένα διάγραμμα οντοτήτων-σχέσεων (ER Model - βλ. παρακάτω σχήμα), όπου το σύμβολο 1...* δηλώνει μία σχέση 1:N (ένα-προς-πολλά).



Σχήμα 1.2: Διάγραμμα Οντοτήτων-Σχέσεων (E-R Model)

Σε μία έγκυρη ΒΔ επιτρέπονται μόνο σχέσεις ένα-προς-πολλά γιατί διαφορετικά αν υπήρχαν σχέσεις 1:1 αυτό θα σήμαινε συναρτησιακή εξάρτηση και θα έπρεπε να συμπεριληφθούν σε μία σχέση, ενώ αν υπήρχαν σχέσεις Ν:Μ τότε αυτό σημαίνει πολλαπλή συναρτησιακή εξάρτηση και θα πρέπει να δημιουργηθεί ένας ενδιάμεσος πίνακας που να περιλαμβάνει ως πρωτεύον κλειδί ένα σύνθετο κλειδί που να περιλαμβάνει τα πρωτεύοντα κλειδιά των δύο ή περισσοτέρων σχέσεων και επομένως οι αρχικές σχέσεις να έχουν στη συνέχεια σχέση ένα-προς πολλά με τον ενδιάμεσο πίνακα.

Στο αρχικό παράδειγμα είναι προφανές ότι όλες οι σχέσεις:

ΚΑΘΗΓΗΤΕΣ(κωδικός, επώνυμο, όνομα, οδός, αριθμός, ΤΚ, τίτλος)

ΦΟΙΤΗΤΕΣ (κωδικός, όνομα, επώνυμο, οδός, αριθμός, ΤΚ, ηλικία)

ΤΜΗΜΑΤΑ (κωδικός, ονομασία, πόλη)

ΑΙΘΟΥΣΕΣ (κωδικός, ονομασία, πλήθος_θέσεων)

ΜΑΘΗΜΑΤΑ (κωδικός, ονομασία)

είχαν σχέση πολλά-προς-πολλά και πρακτικά κάποιος έμπειρος αναλυτής θα μπορούσε από την αρχή να οδηγηθεί σε αυτές τις σχέσεις και στη συνέχεια να δημιουργήσει τον ενδιάμεσο πίνακα ΔΙΔΑΣΚΑΛΙΕΣ (κ_τμήματος, κ_μαθήματος, κ_καθηγητή, κ_αίθουσας, κ_φοιτητή), που αποτελείται από τα πρωτεύοντα κλειδιά των βασικών πινάκων.

Αν όμως θεωρήσουμε τον κανόνα ότι για παράδειγμα ένας καθηγητής κάνει πολλά συγκεκριμένα μαθήματα σε συγκεκριμένους φοιτητές και συγκεκριμένες αίθουσες, ένας φοιτητής παρακολουθεί συγκεκριμένα μαθήματα σε συγκεκριμένες αίθουσες και σε μια αίθουσα γίνονται συγκεκριμένα μαθήματα, τότε υπάρχει συνδετική εξάρτηση ανάμεσα σε καθηγητές-μαθήματα, καθηγητές-φοιτητές, καθηγητές-αίθουσες, φοιτητές-μαθήματα και φοιτητές-αίθουσες και αίθουσες-μαθήματα καθώς και τμήματα-καθηγητές, τμήματα-φοιτητές, τμήματα-μαθήματα και τμήματα-αίθουσες.

Επομένως η σχέση ΔΙΔΑΣΚΑΛΙΕΣ (κ_τμήματος, κ_μαθήματος, κ_καθηγητή, κ_αίθουσας, κ_φοιτητή) μπορεί να κατασκευαστεί από τις σχέσεις:

(κ_καθηγητή, κ_μαθήματος), (κ_καθηγητή, κ_φοιτητή), (κ_καθηγητή, κ_αίθουσας), (κ_μαθήματος, κ_φοιτητή), (κ_φοιτητή, κ_αίθουσας), (κ_μαθήματος, κ_αίθουσας) (κ_τμήματος, κ_καθηγητή), (κ_τμήματος, κ_φοιτητή), (κ_τμήματος, κ_μαθήματος), (κ_τμήματος, κ_αίθουσας)

που είναι όλες στην πέμπτη κανονική μορφή. Επιπλέον, όλες οι άλλες σχέσεις είναι στην πέμπτη κανονική μορφή γιατί δεν μπορούν να αποσυντεθούν περαιτέρω σε μικρότερες σχέσεις, που να έχουν διαφορετικό πρωτεύον κλειδί από τις αρχικές.

1.6 Η γλώσσα SQL (Structured Query Language)

Η γλώσσα SQL είναι μία γλώσσα διαχείρισης βάσεων δεδομένων που υλοποιείται σήμερα από κάθε σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (Relational Database management System – RDBMS) και έχει πιστοποιηθεί από τους οργανισμούς ANSI και ISO ως πρότυπη γλώσσα για σχεσιακές βάσεις δεδομένων.

Η γλώσσα SQL αποτελείται από δύο μέρη:

- **DDL** (Data Definition Language) – Γλώσσα ορισμού δεδομένων που παρέχει ένα σύνολο εντολών για την δημιουργία, τροποποίηση και διαγραφή πινάκων, γνωρισμάτων, όψεων, ευρετηρίων, κλπ δομών μιας βάσης δεδομένων

- **DML** (Data Manipulation Language) – Γλώσσα διαχείρισης Δεδομένων που παρέχει εντολές για την εισαγωγή, τροποποίηση, διαγραφή και αναζήτηση (επιλογή) δεδομένων από τη βάση

Όπως κάθε γλώσσα προγραμματισμού, έτσι και η SQL καθιερώνει ορισμένου τύπους δεδομένων για τον ορισμό των γνωρισμάτων και μεταβλητών του κάθε τύπου.

Οι βασικοί τύποι δεδομένων που καθιερώνονται στην ANSI SQL είναι οι παρακάτω:

integer	ακέραιος αριθμός
numeric(p,d)	πραγματικός με p ψηφία, από τα οποία το πολύ d είναι δεκαδικά
real	πραγματικός αριθμός
double	πραγματικός αριθμός διπλάσιος του πραγματικού (real)
float(n)	αριθμός κινητής υποδιαστολής με ακρίβεια n ψηφίων
date	Ημερομηνία yyyy/mm/dd
time	Χρόνος hh:mm:ss
char(n)	συμβολοσειρά μήκους n, σταθερό μήκος n
varchar(n)	συμβολοσειρά μήκους n, μεταβλητό μήκος n. Αν έχουμε μικρότερου μήκους συμβολοσειρά, οι θέσεις που δεν χρησιμοποιούνται, απελευθερώνονται.

Στις διάφορες εμπορικές υλοποιήσεις καθιερώνονται και άλλοι τύποι δεδομένων εκτός από τους παραπάνω, όπως ο τύπος BLOB (Binary Large Object) για αποθήκευση ψηφιακών δεδομένων μεγάλου μεγέθους (π.χ. φωτογραφίες) κ.α..

Στη συνέχεια θα χρησιμοποιήσουμε παραδειγματικά τις σχέσεις, του προηγούμενου σχήματος, όπου για απλοποίηση χρησιμοποιούμε τον πίνακα Διδασκαλίες, έστω και αν είναι στην τέταρτη κανονική μορφή, ως τον ενδιάμεσο πίνακα των βασικών πινάκων.

ΚΑΘΗΓΗΤΕΣ(κωδικός, επώνυμο, όνομα, οδός, αριθμός, ΤΚ, τίτλος)

ΦΟΙΤΗΤΕΣ (κωδικός, όνομα, επώνυμο, οδός, αριθμός, ΤΚ, ηλικία)

ΤΜΗΜΑΤΑ (κωδικός, ονομασία, πόλη)

ΑΙΘΟΥΣΕΣ (κωδικός, ονομασία, πλήθος_θέσεων)

ΜΑΘΗΜΑΤΑ (κωδικός, ονομασία)

ΔΙΔΑΣΚΑΛΙΕΣ (κ_τμήματος, κ_μαθήματος, κ_καθηγητή, κ_αίθουσας, κ_φοιτητή)

1.6.1 DDL (Data Definition Language)

Η γλώσσα DDL της SQL χρησιμοποιείται για την δημιουργία πινάκων, ευρετηρίων, όψεων, περιορισμών ορθότητας, ορισμού τύπου δεδομένων, δομών αποθήκευσης, ασφάλειας και ελέγχου πρόσβασης χρηστών.

1.6.1.1 Δημιουργία Πίνακα

Η **δημιουργία πίνακα** υλοποιείται με την εντολή `CREATE TABLE` όπως παρακάτω:

```
CREATE TABLE <όνομα_πίνακα>
(
      <F1> <T1> <C1>
      <F2> <T2> <C2>
      .
      .
      .
      <Fn> <Tn> <Cn>
);
```

όπου Fi (i=1,...,n) είναι τα γνωρίσματα του πίνακα, Ti (i=1,...,n) τα πεδία ορισμού και Ci προαιρετικούς περιορισμούς.

Για παράδειγμα η δημιουργία του πίνακα ΦΟΙΤΗΤΕΣ μπορεί να γίνει με την παρακάτω εντολή:

```
CREATE TABLE Φοιτητές
(
      Κωδικός integer NOT NULL,
      όνομα char(50) NOT NULL,
      επώνυμο char(50) NOT NULL,
```

```
      οδός char(40),
      αριθμός char(3),
      τκ char(5),
      ηλικία integer
      CONSTRAINT  pr_con PRIMARY KEY (κωδικός),
      CHECK (ηλικία > 0)
);
```

Η δημιουργία του πίνακα Καθηγητές γίνεται όπως παρακάτω:

```
CREATE TABLE Καθηγητές
(
      Κωδικός integer NOT NULL,
      όνομα char(50) NOT NULL,
      επώνυμο char(50) NOT NULL,
      οδός char(40),
      αριθμός char(3),
      τκ char(5),
      τίτλος char(10),
      CONSTRAINT  pr_con2 PRIMARY KEY (κωδικός),
      CHECK (τίτλος) IN ("Mr", "Mrs", "Dr")
);
```

Οι εντολές CONSTRAINT <όνομα_περιορισμού> PRIMARY KEY (πρωτεύον-κλειδί), δημιουργούν ένα περιορισμό πρωτεύοντος κλειδιού για το πεδίο πρωτεύον-κλειδί. Οι εντολές CHECK(α) αφορούν περιορισμούς ορθότητας. Η εντολή CHECK ελέγχει αν η συνθήκη α είναι αληθής. Ο περιορισμός NOT NULL ελέγχει αν το αντίστοιχο πεδίο έχει τιμή ή είναι αόριστο (NULL) και δεν επιτρέπει στο συγκεκριμένο πεδίο να μην έχει κάποια τιμή (υποχρεωτικό πεδίο).

1.6.1.2 Διαγραφή Πίνακα

Η **διαγραφή πίνακα** γίνεται με την εντολή DROP TABLE, όπως παρακάτω:

```
DROP TABLE <όνομα_πίνακα>
```

Η εντολή DROP TABLE διαγράφει και τα περιεχόμενα του πίνακα

1.6.1.3 Διαγραφή Βάσης Δεδομένων

Η **διαγραφή μιας ολόκληρης Βάσης Δεδομένων**, δηλαδή όλων των πινάκων της Βάσης γίνεται με την εντολή DROP DATABASE, όπως παρακάτω:

```
DROP DATABASE <όνομα_βάσης_δεδομένων>
```

1.6.1.4 Αλλαγή Στοιχείων Πινάκων

Η **αλλαγή των στοιχείων ενός πίνακα** γίνεται με την εντολή ALTER TABLE, όπως παρακάτω:

```
ALTER TABLE  <όνομα_πίνακα>
      ADD (όνομα_πεδίου  τύπος_δεδομένων);
       --  προσθέτει ένα πεδίο

ALTER TABLE  <όνομα_πίνακα>
      DROP (όνομα_ πεδίου );
--  διαγράφει ένα πεδίο
```

```
ALTER TABLE  <όνομα_πίνακα>
     MODIFY (όνομα_ πεδίου  τύπος_δεδομένων);
-- τροποποιεί ένα πεδίο

ALTER TABLE  <όνομα_πίνακα>
     ADD CONSTRAINT <όνομα_περιορισμού>
-- προσθέτει ένα περιορισμό
```

Η τελευταία μορφή της εντολής ALTER TABLE χρησιμοποιείται για να προστεθούν περιορισμοί ορθότητας σε ένα πίνακα, που αρχικά δεν μπορούν να δημιουργηθούν γιατί δεν έχει ακόμα δημιουργηθεί ο συνδεδεμένος πίνακας.

Για παράδειγμα μπορούμε να προσθέσουμε ένα περιορισμό ξένου κλειδιού στον πίνακα Διδασκαλίες προς τον πίνακα Φοιτητές, ως εξής:

```
ALTER TABLE Διδασκαλίες
     ADD CONSTRAINT FK_Constraint
     FOREIGN KEY (κ_φοιτητή)
     REFERENCES ΦΟΙΤΗΤΕΣ(κωδικός);
```

1.6.1.5 Δημιουργία ευρετηρίου

CREATE INDEX <όνομα_ευρετηρίου> ON P(a, b),

που δημιουργεί ένα ευρετήριο που αποτελείται από τα πεδία α, β στον πίνακα P.

Παράδειγμα:

```
CREATE INDEX myIndx ON ΦΟΙΤΗΤΕΣ(επώνυμο);
```

1.6.2 DML (Data Manipulation Language)

Η κύρια μορφή εντολής (ερώτησης -- query) της SQL για την αναζήτηση ή επιλογή στοιχείων είναι η εντολή **select,** που χρησιμοποιείται όπως παρακάτω:

```
SELECT ATR1, ATR2,..., ATRn
         FROM R1,..., Rm
         WHERE C
```

όπου ATRi είναι οι ιδιότητες ή γνωρίσματα των σχέσεων, Rj είναι οι σχέσεις και C τα κριτήρια επιλογής.

1.6.2.1 Παραδείγματα χρήσης ερωτήσεων επιλογής (queries)

1. Εμφάνισε (όλα τα γνωρίσματα) για όλες τις αίθουσες

```
SELECT *
   FROM ΑΙΘΟΥΣΕΣ;
```

Με το σύμβολο * εννοούνται όλα τα πεδία ATR1, ATR2,..., ATRn, ενώ στο παραπάνω παράδειγμα εννοούνται όλα τα πεδία: κωδικός, ονομασία, πλήθος_θέσεων του πίνακα ΑΙΘΟΥΣΕΣ.

2. Βρες τα στοιχεία των τμημάτων που βρίσκονται στην Καβάλα

```
SELECT *
     FROM ΤΜΗΜΑΤΑ
     WHERE πόλη = "Καβάλα";
```

3. Βρες τα μαθήματα που έχουν κωδικό > 10

```
SELECT *
       FROM ΜΑΘΗΜΑΤΑ
       WHERE κωδικός > 10;
```

4. Βρες τις διευθύνσεις όλων των Φοιτητών

```
SELECT οδός, αριθμός, ΤΚ
     FROM ΦΟΙΤΗΤΕΣ;
```

5. Βρες τις ηλικίες των φοιτητών

```
SELECT ηλικία
     FROM ΦΟΙΤΗΤΕΣ;
```

Αν δεν θέλουμε επαναλαμβανόμενες τιμές, π.χ. πολλές φορές την ίδια ηλικία, τότε μπορούμε να χρησιμοποιήσουμε την δεσμευμένη λέξη `DISTINCT`:

```
SELECT DISTINCT ηλικία
     FROM ΦΟΙΤΗΤΕΣ;
```

Με την εντολή `DISTINCT` διαγράφονται οι πολλαπλές τιμές, ενώ με την εντολή `ALL` συμπεριλαμβάνονται όλες οι τιμές. Οι `DISTINCT` και `ALL` είναι συμπληρωματικές πράξεις.

Η γλώσσα SQL παρέχει τελεστές σύγκρισης όπως <, >, =, BETWEEN x AND y, και LIKE.

Επίσης, παρέχονται χαρακτήρες μπαλαντέρ όπως ο χαρακτήρας _ (underscore) που σημαίνει ένας οποιοσδήποτε χαρακτήρας και ο χαρακτήρας % που σημαίνει πολλοί οποιοιδήποτε χαρακτήρες.

6. Αν θέλαμε να εμφανίσουμε όλους τους φοιτητές που το επώνυμό τους αρχίζει από ΠΑΠΑ, τότε θα έχουμε:

```
SELECT Φ.όνομα
      FROM ΦΟΙΤΗΤΕΣ Φ
      WHERE Φ.επώνυμο LIKE "ΠΑΠΑ%";
```

7. Βρες όλους τους φοιτητές που έχουν ηλικία ανάμεσα σε 20 και 25

```
SELECT Φ.επώνυμο, Φ.όνομα
      FROM ΦΟΙΤΗΤΕΣ Φ
      WHERE Φ.ηλικία BETWEEN 20 AND 25;
```

Επίσης, η SQL επιτρέπει την χρήση ψευδωνύμων (alias) όπως το ψευδώνυμο Φ που αναφέρεται στον πίνακα ΦΟΙΤΗΤΕΣ και αφού ορίζεται στην γραμμή του FROM όπως παραπάνω, στη συνέχεια μπορεί να χρησιμοποιείται αντί του ονόματος του πίνακα.

8. Βρες το ονοματεπώνυμο των φοιτητών που σπουδάζουν σε τμήματα στην Λαμία

```
SELECT επώνυμο
      FROM ΦΟΙΤΗΤΕΣ, ΤΜΗΜΑΤΑ, ΔΙΔΑΣΚΑΛΙΕΣ
      WHERE ΤΜΗΜΑΤΑ.κωδικός = ΔΙΔΑΣΚΑΛΙΕΣ.κ_τμήματος
      AND ΔΙΔΑΣΚΑΛΙΕΣ.κ_φοιτητή = ΦΟΙΤΗΤΕΣ.κωδικός
      AND ΤΜΗΜΑΤΑ.πόλη LIKE "Λαμία";
```

Στην παραπάνω ερώτηση επειδή αναζητούνται δεδομένα που σχετίζονται και με τους δύο πίνακες: ΦΟΙΤΗΤΕΣ και ΤΜΗΜΑΤΑ και για να μπορέσει να υλοποιηθεί η σύνδεση (join) θα πρέπει αναγκαστικά να συμπεριλαμβάνεται στην ερώτηση (query) και ο ενδιάμεσος πίνακας ΔΙΔΑΣΚΑΛΙΕΣ, που υλοποιεί τη σύνδεση ανάμεσα στους δύο βασικούς πίνακες.

Με τη φράση:

```
FROM ΦΟΙΤΗΤΕΣ, ΤΜΗΜΑΤΑ, ΔΙΔΑΣΚΑΛΙΕΣ
```

σχηματίζεται αρχικά το καρτεσιανό γινόμενο των τριών πινάκων (ΤΜΗΜΑΤΑ, ΦΟΙΤΗΤΕΣ, ΔΙΔΑΣΚΑΛΙΕΣ)

και στη συνέχεια με τη φράση:

```
WHERE ΤΜΗΜΑΤΑ.κωδικός = ΔΙΔΑΣΚΑΛΙΕΣ.κ_τμήματος
AND ΦΟΙΤΗΤΕΣ.κωδικός = ΔΙΔΑΣΚΑΛΙΕΣ.κ_φοιτητή
```

επιλέγονται οι πλειάδες που έχουν ίδιους κωδικούς φοιτητών και ίδιους κωδικούς τμημάτων.

Τελικά με τη φράση: AND ΤΜΗΜΑΤΑ.πόλη LIKE "Λαμία", επιλέγονται οι πλειάδες που έχουν ως πόλη την Λαμία και εμφανίζεται το επώνυμο του κάθε φοιτητή, όπως ζητείται.

Εκτός από την παραπάνω μορφή σύνδεσης (join) που επιλέγει μόνο τα στοιχεία των δύο πινάκων που έχουν την ίδια τιμή στα κοινά πεδία, υπάρχουν και δύο άλλες μορφές σύνδεσης: αριστερή σύνδεση (left join) και δεξιά σύνδεση (right join) που επιλέγουν όλα τα στοιχεία του αριστερού ή δεξιού πίνακα και μόνο εκείνα τα στοιχεία του δεξιά ή αριστερά πίνακα αντίστοιχα που έχουν την ίδια τιμή στο πεδίο επιλογής. Αυτό σημαίνει πως κάποια από όλα τα πεδία που επιλέγονται μπορεί να μην έχουν αντίστοιχο ίδιο πεδίο στον άλλο πίνακα και επομένως τα πεδία που επιλέγονται να είναι null.

Αν για παράδειγμα συσχετίζαμε με κανονική σύνδεση τον πίνακα ΠΕΛΑΤΩΝ και ΠΡΟΪΟΝΤΩΝ, τότε στην έξοδο του ερωτήματος θα εμφανιζόταν μόνο οι πελάτες και τα προϊόντα που θα είχαν την ίδια τιμή στα πεδία κωδικός_πελάτη και κωδικός_προϊόντος.

Αν όμως σχετίζαμε τους δύο πίνακες με αριστερή σύνδεση τότε θα εμφανίζονταν όλοι οι πελάτες του πίνακα ΠΕΛΑΤΕΣ και αν κάποιοι πελάτες δεν είχαν αγοράσει προϊόντα θα εμφανιζόταν στην έξοδο του ερωτήματος αλλά με την τιμή null στην θέση του προϊόντος.

Με δεξιά σύνδεση θα εμφανίζονταν όλα τα προϊόντα του πίνακα ΠΡΟΪΟΝΤΑ και μόνο εκείνοι οι πελάτες που τα είχαν αγοράσει. Στις περιπτώσεις προϊόντων που δεν είχαν αγοραστεί από κανένα πελάτη θα εμφανιζόταν null στη θέση του πελάτη.

Οι περιπτώσεις αριστερής και δεξιάς σύνδεσης μπορούν να χρησιμοποιούνται για λόγους τεκμηρίωσης του συνόλου των εγγραφών ενός πίνακα.

Από άποψη επίδοσης μπορούμε να παρατηρήσουμε ότι ο υπολογισμός του καρτεσιανού γινομένου μεταξύ δύο ή περισσοτέρων πινάκων μπορεί να απαιτεί υψηλό κόστος μνήμης και αναζήτησης σε περιπτώσεις που κάποιος ή κάποιοι από τους συμμετέχοντες πίνακες είναι μεγάλος/οι.

Σε πολλά συστήματα ΒΔ ακολουθούνται στρατηγικές βελτιστοποίησης και επιτάχυνσης της εκτέλεσης των ερωτήσεων κυρίως με μεθόδους ελέγχου του κόστους εκτέλεσης του γράφου εκτέλεσης που δημιουργείται για κάθε ερώτημα. Επίσης, σε πολλά συστήματα ΒΔ προτείνονται τρόποι επιτάχυνσης της εκτέλεσης με την δημιουργία ευρετηρίων ή εναλλακτικών διαδρομών και τρόπων εκτέλεσης των ερωτημάτων.

Τέλος, σε συστήματα που χρησιμοποιούν μεθόδους σημασιολογικής βελτιστοποίησης (semantic query optimization), χρησιμοποιούνται οι περιορισμοί (constraints) που έχουν οριστεί στο σχήμα της ΒΔ ώστε να δημιουργηθεί ένα νέο βελτιστοποιημένο ερώτημα που να εκτελείται γρηγορότερα. Αν για παράδειγμα στο πεδίο ΜΙΣΘΟΣ ενός πίνακα υπάρχει ο περιορισμός ”< 10000”, τότε σε ένα SELECT των υπαλλήλων με μισθό ”> 10000” ο σημασιολογικός βελτιστοποιητής αναγνωρίζει τον περιορισμό και δεν εκτελεί καθόλου το ερώτημα (ή επιστρέφει δίχως να το εκτελέσει).

9. Οι ονομασίες των αιθουσών που βρίσκονται στη Χίο

```
SELECT ονομασία
      FROM ΑΙΘΟΥΣΕΣ, ΤΜΗΜΑΤΑ, ΔΙΔΑΣΚΑΛΙΕΣ
      WHERE ΤΜΗΜΑΤΑ.κωδικός = ΔΙΔΑΣΚΑΛΙΕΣ.κ_τμήματος
      AND ΔΙΔΑΣΚΑΛΙΕΣ.κ_αίθουσας = ΑΙΘΟΥΣΕΣ.κωδικός
      WHERE ΤΜΗΜΑΤΑ.πόλη LIKE "Χίος";
```

10. Οι κωδικοί των φοιτητών που έχουν ηλικία > 25

```
SELECT κωδικός
      FROM ΦΟΙΤΗΤΕΣ
      WHERE ηλικία > 25;
```

11. Ονοματεπώνυμα των Καθηγητών που έχουν μάθημα σε αίθουσες με ονομασία "Αμφιθέατρο Α"

```
SELECT Ε.επώνυμο, Ε. όνομα
     FROM Καθηγητές Κ, ΔΙΔΑΣΚΑΛΙΕΣ Δ, Αίθουσες Α
     WHERE Κ.κωδικός = Δ. κ_καθηγητή
     AND Δ. κ_αίθουσας = Α.κωδικός
     AND Α.ονομασία = "Αμφιθέατρο Α";
```

12. Κωδικοί φοιτητών που έχουν μάθημα στην αίθουσα με ονομασία "Αμφιθέατρο Α"
και το μάθημα έχει ονομασία "ΦΥΣΙΚΗ 1"

```
SELECT Φ.επώνυμο, Φ.όνομα
     FROM Φοιτητές Φ, ΔΙΔΑΣΚΑΛΙΕΣ Δ, Αίθουσες Α, Μαθήματα Μ
     WHERE Φ.κωδικός = Δ.κ_φοιτητή
     AND Δ.κ_αίθουσας = Α.κωδικός
```

```
      AND Δ. κ_μαθήματος = Μ.κωδικός
      AND Α.ονομασία LIKE "Αμφιθέατρο Α";
      AND Μ.ονομασία LIKE "ΦΥΣΙΚΗ 1";
```

Μία άλλη κατηγορία πράξεων στην SQL είναι οι πράξεις που αφορούν σύνολα (συνολοθεωρητικές πράξεις), όπως: Ένωση (UNION), Τομή (INTERSECT) και Αφαίρεση (EXCEPT).

1. Ονοματεπώνυμα φοιτητών που έχουν ηλικία 20 ή 30

```
SELECT Φ.επώνυμο, Φ.όνομα
      FROM Φοιτητές Φ
      WHERE Φ.ηλικία = 20
UNION
SELECT Φ.επώνυμο, Φ.όνομα
      FROM Φοιτητές Φ
      WHERE Φ.ηλικία = 30;
```

2. Ονόματα φοιτητών που έχουν είτε μάθημα "ΦΥΣΙΚΗ 1" ή "ΧΗΜΕΙΑ 1"

```
SELECT Φ.όνομα
      FROM Φοιτητές Φ, Μαθήματα Μ, Διδασκαλίες Δ
      WHERE Μ.κωδικός = Δ.κ_μαθήματος
      AND Φ.κωδικός = Δ.κ_φοιτητή
      AND Μ.ονομασία LIKE "ΦΥΣΙΚΗ 1"
UNION
SELECT Φ.ονομα
      FROM Φοιτητές Φ, Μαθήματα Μ, Διδασκαλίες Δ
      WHERE Μ.κωδικός = Δ.κ_μαθήματος
      AND Φ.κωδικός = Δ.κ_φοιτητή
      AND Μ.ονομασία LIKE "ΧΗΜΕΙΑ 1";
```

3. Φοιτητές που έχουν μάθημα "ΦΥΣΙΚΗ 1" και "ΧΗΜΕΙΑ 1"

```
SELECT Φ.ονομα
     FROM Φοιτητές Φ, Μαθήματα Μ, Διδασκαλίες Δ
     WHERE Μ.κωδικός = Δ.κ_μαθήματος
     AND Φ.κωδικός = Δ.κ_φοιτητή
     AND Μ.ονομασία LIKE "ΦΥΣΙΚΗ 1"
INTERSECT
SELECT Φ.όνομα
     FROM Φοιτητές Φ, Μαθήματα Μ, Διδασκαλίες Δ
     WHERE Μ.κωδικός = Δ.κ_μαθήματος
     AND Φ.κωδικός = Δ.κ_φοιτητή
     AND Μ.ονομασία LIKE "ΧΗΜΕΙΑ 1";
```

4. Φοιτητές που έχουν μάθημα "ΦΥΣΙΚΗ 1" αλλά όχι "ΧΗΜΕΙΑ 1"

```
SELECT Φ.ονομα
     FROM Φοιτητές Φ, Μαθήματα Μ, Διδασκαλίες Δ
     WHERE Μ.κωδικός = Δ.κ_μαθήματος
     AND Φ.κωδικός = Δ.κ_φοιτητή
     AND Μ.ονομασία LIKE "ΦΥΣΙΚΗ 1"
EXCEPT
SELECT Φ.όνομα
     FROM Φοιτητές Φ, Μαθήματα Μ, Διδασκαλίες Δ
     WHERE Μ.κωδικός = Δ.κ_μαθήματος
     AND Φ.κωδικός = Δ.κ_φοιτητή
     AND Μ.ονομασία LIKE "ΧΗΜΕΙΑ 1";
```

Παρατηρούμε από την σκοπιά της επίδοσης ότι στις περιπτώσεις συνολοθεωρητικών πράξεων ελαχιστοποιούνται τα καρτεσιανά γινόμενα ή δεν χρειάζονται καθόλου (όπως στα

συγκεκριμένα παραδείγματα) και επομένως ελαχιστοποιείται και το υπολογιστικό κόστος (σε χώρο και χρόνο).

Σε πολλές εμπορικά συστήματα Βάσεων Δεδομένων δεν παρέχονται όλες οι πράξεις συνόλων, οπότε οι προγραμματιστές θα πρέπει να τις υλοποιούν με έμμεσους τρόπους.

Μια άλλη κατηγορία πράξεων στην SQL είναι οι αθροιστικές συναρτήσεις (AGGREGATE FUNCTIONS), όπου εφαρμόζεται μια μαθηματική συνάρτηση σε όλες τις τιμές μιας στήλης.

Τέτοιες συναρτήσεις είναι:

```
AVG: μέσος όρος
MIN: ελάχιστο
MAX: μέγιστο
SUM: άθροισμα
COUNT: καταμέτρηση
```

Παραδείγματα:

1. Μέσος όρος ηλικιών των φοιτητών

```
SELECT AVG (ηλικία)
FROM Φοιτητές;
```

2. Σύνολο θέσεων όλων των αιθουσών

```
SELECT SUM (πλήθος_θέσεων)
FROM Αίθουσες;
```

3. Πλήθος φοιτητών με ηλικία 30

```
SELECT COUNT (*)
       FROM Φοιτητές
       WHERE ηλικία = 30;
```

4. Πόσες διαφορετικές ηλικίες έχουν οι Φοιτητές

```
SELECT COUNT (DISTINCT ηλικία)
     FROM Φοιτητές;
```

Πολλές φορές θέλουμε να εμφανίζουμε ομαδοποιημένα αποτελέσματα ως προς κάποια γνωρίσματα. Ιδιαίτερα σε εφαρμογές που αφορούν κατηγορίες δεδομένων, όπως άντρες-γυναίκες, πόλεις, περιφέρειες, νομοί μιας χώρας, μορφωτικά επίπεδα, οικογενειακές καταστάσεις, κλπ, είναι χρήσιμο να εμφανίζουμε τα αποτελέσματα ομαδοποιημένα.

Η φράση GROUP BY ομαδοποιεί τα αποτελέσματα μιας ερώτησης ως προς το πεδίο (ή τα πεδία) που θέλουμε. Όταν ομαδοποιούμε με την GROUP BY, στο SELECT δεν επιτρέπεται να έχουμε άλλα πεδία εκτός από αυτό που εμφανίζεται στο GROUP BY καθώς και συγκεντρωτικές συναρτήσεις.

Παραδείγματα:

1. Μέσος όρος θέσεων αιθουσών ανά πόλη

```
SELECT Τ.πόλη, AVG (Α.πλήθος_θέσεων)
     FROM Αίθουσες Α, Τμήματα Τ, Διδασκαλίες Δ
     WHERE Α.κωδικός = Δ.κ_αίθουσας
     AND Τ.κωδικός = Δ.κ_τμήματος
     GROUP BY Τ.πόλη;
```

2. Πλήθος φοιτητών ανά ηλικία

```
SELECT ηλικία, COUNT (κωδικός)
      FROM Φοιτητές
      GROUP BY ηλικία;
```

Η ομαδοποίηση μπορεί να γίνει ως προς περισσότερα του ενός πεδία.

3. Πλήθος φοιτητών ανά πόλη και ηλικία

```
SELECT Τ.πόλη, Φ.ηλικία, COUNT (κωδικός)
      FROM Φοιτητές Φ, Τμήματα Τ, Διδασκαλίες Δ
      WHERE Φ.κωδικός = Δ.κ_φοιτητή
      AND Τ.κωδικός = Δ.κ_τμήματος
      GROUP BY Τ.πόλη, Φ.ηλικία;
```

4. Αίθουσες που έχουν μέσο πλήθος θέσεων κάτω από 50

```
SELECT ονομασία, AVG (πλήθος_θέσεων)
       FROM Αίθουσες
       GROUP BY ονομασία
       HAVING AVG(πλήθος_θέσεων) < 50;
```

Θα πρέπει να παρατηρήσουμε ότι ενώ η WHERE ψάχνει στην κάθε πλειάδα, η HAVING ψάχνει στην ομάδα.

Σε πολλές περιπτώσεις δεν γνωρίζουμε την τιμή ενός πεδίου σε μια ή περισσότερες πλειάδες. Η έλλειψη τιμής αναγνωρίζεται από την βάση δεδομένων σαν NULL. Η τιμή NULL δεν είναι 0, αλλά μη-τιμή.

Για παράδειγμα στην παρακάτω ερώτηση:

```
SELECT SUM(πλήθος_θέσεων)
FROM Αίθουσα;
```

η τιμή NULL αγνοείται στο άθροισμα (εδώ θεωρείται 0).

Η SQL επιτρέπει φωλιασμένες ερωτήσεις (nested queries) της μορφής:

```
SELECT  ...
       FROM...
       WHERE
           (SELECT...
                 FROM ...
                 WHERE ...  );
```

Παραδείγματα:

1. Ονόματα και επώνυμα φοιτητών που διδάσκονται ΦΥΣΙΚΗ στην Λαμία

```
SELECT DISTINCT Φ.επώνυμο, Φ.όνομα
     FROM Φοιτητές Φ, Τμήματα Α, Διδασκαλίες Δ
     WHERE Φ.κωδικός = Δ.κωδικός_φοιτητή
          AND Δ.κ_τμήματος. = Τ.κωδικός
          AND Τ.πόλη = "Λαμία"
          AND Φ.κωδικός IN
(SELECT Φ.κωδικός
     FROM Φοιτητές Φ, Μαθήματα Μ, Διδασκαλίες Δ
     WHERE Φ.κωδικός = Δ.κωδικός_φοιτητή
          AND Δ.κ_μαθήματος. = Μ.κωδικός
             AND Μ.ονομασία. = "ΦΥΣΙΚΉ");
```

Η δεσμευμένη λέξη IN ελέγχει αν μια τιμή ανήκει σε ένα σύνολο.

2. Ονόματα Φοιτητών που έχουν και Φυσική και Χημεία

```
SELECT Φ.επώνυμο, Φ.όνομα
FROM Φοιτητές Φ, Μαθήματα Μ, Διδασκαλίες Δ
WHERE Φ.κωδικός = Δ.κωδικός_φοιτητή
AND Δ.κ_μαθήματος. = Μ.κωδικός
AND Μ.ονομασία. = "ΦΥΣΙΚΉ"
AND Φ.κωδικός IN
               (SELECT Φ.κωδικός
                FROM Φοιτητές Φ, Μαθήματα Μ, Διδασκαλίες Δ
                WHERE Φ.κωδικός = Δ.κ_φοιτητή
                AND Δ.κ_μαθήματος. = Μ.κωδικός
                AND Μ.ονομασία. = "ΧΗΜΕΙΑ");
```

3. Ονομασία αιθουσών που έχουν πλήθος θέσεων μεγαλύτερο από κάποια από τις αίθουσες της Καβάλας

```
SELECT ονομασία
       FROM Αίθουσες
       WHERE πλήθος_θέσεων > SOME
                    (SELECT πλήθος_θέσεων
                    FROM Τμήματα Τ, Αίθουσες Α, Διδασκαλίες Δ
                    WHERE Τ.κωδικός = Δ.κ_τμήματος
                    AND Δ.κ_αίθουσας. = Α.κωδικός
                    AND Τ.πόλη = "Καβάλα");
```

Η φράση SOME συγκρίνει την τιμή αριστερά με την μικρότερη τιμή δεξιά στην ερώτηση. Στο παραπάνω παράδειγμα αναζητά τιμές πλήθους θέσεων μεγαλύτερες από τουλάχιστον το ελάχιστο πλήθος θέσεων του δεξιού υποσυνόλου που περιέχει όλα τα πλήθη θέσεων των αιθουσών της Καβάλας.

Αν θέλαμε να βρούμε τις αίθουσες με πλήθος θέσεων μεγαλύτερο από όλα τα πλήθη θέσεων όλων των αιθουσών της Καβάλας, τότε θα πρέπει να χρησιμοποιήσουμε τη φράση ALL που συγκρίνει με τη μέγιστη τιμή του δεξιού υποσυνόλου.

```
SELECT ονομασία
       FROM Αίθουσες
       WHERE πλήθος_θέσεων > ALL
                    (SELECT πλήθος_θέσεων
                    FROM Τμήματα Τ, Αίθουσες Α, Διδασκαλίες Δ
                    WHERE Τ.κωδικός = Δ.κ_τμήματος
                    AND Δ.κ_αίθουσας. = Α.κωδικός
                    AND Τ.πόλη = "Καβάλα");
```

Μπορούμε να δημιουργούμε και ερωτήσεις όπου να συμπεριλαμβάνουμε όλα τα παραπάνω χαρακτηριστικά: ομαδοποίηση, συγκεντρωτικές συναρτήσεις, φωλιασμένα SELECT και τελεστές συνόλων.

Παραδείγματα:

1. Πόλεις και μέσος όρος θέσεων ανά πόλη

```
SELECT Τ.πόλη,  AVG (Α.πλήθος_θέσεων)
     FROM Τμήματα Τ, Αίθουσες Α, Διδασκαλίες Δ
     WHERE Τ.κωδικός = Δ.κ_τμήματος
```

```
      AND Δ.κ_αίθουσας. = Α.κωδικός
      GROUP BY Τ.πόλη;
```

2. Πόλεις που έχουν πλήθος θέσεων μεγαλύτερο από το μέσο όρος θέσεων

```
SELECT Τ.πόλη
      FROM Τμήματα Τ, Αίθουσες Α, Διδασκαλίες Δ
      WHERE Τ.κωδικός = Δ.κ_τμήματος
      AND Δ.κ_αίθουσας. = Α.κωδικός
      GROUP BY Τ.πόλη
      HAVING AVG(Α.πλήθος_θέσεων) >= ALL
                                    (SELECT AVG(πλήθος_θέσεων)
                                    FROM Αίθουσες);
```

3 Πόλεις που έχουν μέσο όρο θέσεων μεγαλύτερο από 500

```
SELECT Τ.πόλη
       FROM Τμήματα Τ, Αίθουσες Α, Διδασκαλίες Δ
       WHERE Τ.κωδικός = Δ.κ_τμήματος
       AND Δ.κ_αίθουσας. = Α.κωδικός
       GROUP BY Τ.πόλη
       HAVING AVG(Α.πλήθος_θέσεων) >= 500;
```

4. Καθηγητές που έχουν έστω μία διδασκαλία μαθημάτων

```
SELECT Κ.επώνυμο, Κ.όνομα
FROM Καθηγητές Κ
WHERE EXISTS
               (SELECT Δ.*
                FROM Διδασκαλίες Δ
                WHERE Κ.κωδικός = Δ.κ_καθηγητή);
```

Το παραπάνω ερώτημα χρησιμοποιεί τη φράση EXISTS που συντάσσεται ως:

```
SELECT πεδία
      FROM πίνακες
      WHERE EXISTS (υποερώτημα);
```

και ελέγχει αν το σύνολο του υποερωτήματος είναι άδειο. Αν το σύνολο που εμπεριέχεται στις παρενθέσεις δεν είναι άδειο, τότε η φράση EXISTS επιστρέφει ως true. Αν επιστρέφεται κενό σύνολο, τότε EXISTS θα πάρει την τιμή false.

Στο παραπάνω παράδειγμα η φράση EXISTS υλοποιεί ουσιαστικά την συνολοθεωρητική πράξη της τομής ανάμεσα στα σύνολα των καθηγητών και των διδασκαλιών με βάση το πεδίο κωδικός καθηγητή. Η τομή υλοποιείται τεχνικά με την πράξη της εξωτερικής σύνδεσης (external join) ανάμεσα σε δύο πίνακες (βλ. `WHERE Κ.κωδικός = Δ.κ_καθηγητή` μέσα στο δεξί υποσύνολο).

5. Τμήματα που έχουν παραρτήματα στην Καβάλα ή στην Δράμα

```
SELECT T1.ονομασία
FROM Τμήματα T1
WHERE EXISTS
                (SELECT *
                 FROM Τμήματα T2
                 WHERE T2.πόλη = "Δράμα" OR T2.πόλη = "Καβάλα"
                 AND T1.ονομασία = T2.ονομασία);
```

Στο παραπάνω παράδειγμα οι δύο πίνακες που συνδέονται εξωτερικά με την φράση EXISTS είναι μόνο ο πίνακας Τμήματα που χρησιμοποιείται με δύο διαφορετικά ψευδώνυμα (T1, T2).

5. Ονόματα καθηγητών που έχουν διδασκαλία σε όλα τα τμήματα

```
SELECT K.επώνυμο
      FROM Καθηγητές K
      WHERE NOT EXISTS
            ((SELECT T1.κωδικός
                  FROM Τμήματα T1)
            EXCEPT
            (SELECT DISTINCT Δ.κ_τμήματος
                  FROM Διδασκαλίες Δ
                  WHERE K.κωδικός = Δ.κ_καθηγητή));
```

Το πρώτο SELECT μετά τη φράση NOT EXISTS επιστρέφει όλους τους κωδικούς τμημάτων, ενώ το δεύτερο SELECT επιστρέφει τους κωδικούς τμημάτων που κάνει μάθημα ο κάθε καθηγητής έτσι όπως προέρχεται από το πρώτο αριστερά εξωτερικό SELECT.

Η φράση EXCEPT αφαιρεί από τους κωδικούς όλων τμημάτων (1ο SELECT μετά το NOT EXISTS) τα τμήματα που κάνει μάθημα ο κάθε καθηγητής (τελευταίο κάτω-δεξιά SELECT) και αν το αποτέλεσμα της αφαίρεσης είναι το κενό σύνολο (δηλ. το NOT EXISTS επιστρέψει true) τότε αυτό σημαίνει ότι ο καθηγητής κάνει μάθημα σε όλα τα τμήματα, αφού για να είναι κανό το αποτέλεσμα της αφαίρεσης σημαίνει ότι τα δύο σύνολα, το αρχικό δηλαδή όλων των κωδικών τμημάτων και το σύνολο των μαθημάτων κάθε καθηγητή είναι ίδια.

1.6.3 ΟΨΕΙΣ – (VIEWS)

Όψη είναι το υποσύνολο μιας βάσης, που βασίζεται στο αποτέλεσμα ενός ερωτήματος (SELECT) και μπορεί να χρησιμοποιηθεί ως ανεξάρτητος πίνακας, δηλαδή να λάβει μέρος στη φράση FROM, ως κανονικός πίνακας. Όψεις χρησιμοποιούνται για λόγους ασφαλείας, όταν δεν θέλουμε ένας χρήστης ή μια ομάδα χρηστών να έχει πρόσβαση σε όλη τη βάση,

αλλά μόνο σε δεδομένα που τους αφορούν αλλά και για λόγους λειτουργικούς, όταν θέλουμε να αυξήσουμε την αποδοτικότητα μιας βάσης και την κατανέμουμε σε περισσότερους από έναν εξυπηρετητές.

1.6.3.1 Δημιουργία όψης

Η δημιουργία μιας όψης και επομένως ενός ιδεατού πίνακα γίνεται με τη φράση:

```
CREATE VIEW <όνομα_όψης> AS <ερώτηση_SELECT>
```

Τις όψεις τις χρησιμοποιούμε συνήθως για επιλογή, αναζήτηση και επεξεργασία δεδομένων, ενώ πράξεις εισαγωγής, ενημέρωσης και διαγραφής μπορεί να έχουν ως αποτέλεσμα προβλήματα συνέπειας της βάσης και για το λόγο αυτό, σε πολλά εμπορικά συστήματα δεν επιτρέπονται ή επιτρέπονται κάτω από κάποιες συνθήκες.

Παράδειγμα: Να βρεθούν τα ονόματα των πελατών ανά υποκατάστημα (θέλουμε μόνο τα στοιχεία branch_name και customer_name)

```
CREATE VIEW Τμήματα_Καθηγητές AS
      (SELECT Τ.ονομασία, Κ.επώνυμο, Κ.όνομα
      FROM Καθηγητές Κ, Τμήματα Τ, Διδασκαλίες Δ
      WHERE Κ.κωδικός = Δ.κ_καθηγητή
      AND Δ.κ_τμήματος. = Τ.κωδικός);
```

Στη συνέχεια μπορούμε να χρησιμοποιήσουμε την όψη ως πίνακα σε μια εντολή SELECT:

```
SELECT επώνυμο, όνομα
       FROM Τμήματα_Καθηγητές
       WHERE ονομασία = "Πληροφορική";
```

1.6.4 Διαχείριση εισαγωγών, διαγραφών και τροποποιήσεων

Με την γλώσσα DML της SQL μπορούμε εκτός από αναζητήσεις να εισάγουμε, τροποποιήσουμε και διαγράψουμε δεδομένα.

1.6.4.1 Εισαγωγή

Η εισαγωγή μιας εγγραφής σε ένα πίνακα γίνεται με την εντολή INSERT INTO όπως παρακάτω:

```
INSERT INTO ΤΜΗΜΑΤΑ
VALUES (1, "Τμήμα Οικονομικής Επιστήμης");
```

η οποία εισάγει μια νέα εγγραφή στον πίνακα Τμήματα. Η μορφή αυτή της INSERT χρησιμοποιείται όταν εισάγουμε όλα τα πεδία μιας εγγραφής με τη σωστή σειρά.

Σε περιπτώσεις που δεν θέλουμε να συμπληρώσουμε όλα τα πεδία μιας εγγραφής, αλλά μόνο κάποια από αυτά τότε θα πρέπει να δηλώσουμε μετά το όνομα του πίνακα και τη λίστα των γνωρισμάτων στα οποία θα αντιστοιχηθούν οι τιμές που ακολουθούν την φράση VALUES:

```
INSERT INTO ΦΟΙΤΗΤΕΣ (κωδικός, επώνυμο, ηλικία)
VALUES (122, "Παπαδόπουλος", 18);
```

Γενικά, η λίστα γνωρισμάτων είναι προαιρετική. Αν δεν την δίνουμε, τότε θα πρέπει να προσέχουμε να δίνουμε τις τιμές στη σωστή σειρά.

Όταν σχεδιάζεται μια εφαρμογή θα πρέπει να κάποια πεδία που για την εφαρμογή θεωρούνται υποχρεωτικό να εισαχθούν σε μια εγγραφή, να χαρακτηρίζονται ως υποχρεωτικά, ώστε το σύστημα ΒΔ να ελέγχει αυτόματα αν συμπεριλαμβάνονται σε μια

εντολή INSERT. Αν δεν συμβαίνει κάτι τέτοιο και ο προγραμματιστής δεν έχει χαρακτηρίσει κάποια πεδία ως υποχρεωτικά, τότε ένας δεύτερος τρόπος να ελέγχεται η ορθότητα των εισαγωγών είναι μέσα από συναρτήσεις ή διαδικασίες γνωστές ως σκανδάλες (triggers) που εκτελούνται αυτόματα όταν γίνεται κάποια εισαγωγή, διαγραφή ή ενημέρωση στη βάση.

Συνήθως οι έλεγχοι για εισαγωγές πεδίων γίνονται αυτόματα από τη ΒΔ με τον πρώτο τρόπο του χαρακτηρισμού κάποιων πεδίων ως υποχρεωτικών, ενώ ο δεύτερος τρόπος με τις σκανδάλες χρησιμοποιείται συνήθως για τον έλεγχο τον έλεγχο και τη διαχείριση της εφαρμογής.

Για παράδειγμα, έλεγχοι ορθών εισαγωγών στη βάση (εισαγωγή τιμής στο πεδίο ηλικία και όχι χαρακτήρων ή υποχρεωτική εισαγωγή ενός επωνύμου καθηγητή ή φοιτητή) μπορούν αυτόματα να γίνονται από τη βάση, ενώ διαδικασίες όπως αυτόματα ενημέρωση πεδίων άλλων πινάκων που εξαρτώνται από αλλαγές σε κάποιο άλλο πίνακα θα πρέπει να γίνονται με σκανδάλες.

Για παράδειγμα, αν αλλάξει ένα πεδίο Ποσό_Εσόδων σε ένα πίνακα ΕΣΟΔΑ, τότε θα πρέπει να δημιουργηθεί μια σκανδάλη (CREATE TRIGGER ON UPDATE ….) ώστε αυτόματα να αλλάξει το πεδίο Σύνολο ενός άλλου πίνακα π.χ. ΠΡΟΫΠΟΛΟΓΙΣΜΟΣ.

Η φράση INSERT μπορεί να χρησιμοποιηθεί και για πολλαπλές εισαγωγές, όταν βασίζεται σε ένα SELECT:

Έστω ότι έχουμε ένα πίνακα ΚΑΘΗΓΗΤΕΣ_ΦΥΣΙΚΗΣ (κωδικός, επώνυμο, όνομα, διεύθυνση, τίτλος) και θέλουμε να εισάγουμε όλους τους καθηγητές με τίτλο "Dr" στον βασικό πίνακα Καθηγητές.

```
INSERT INTO Καθηγητές
        SELECT κωδικός, επώνυμο, όνομα, διεύθυνση, τίτλος
        FROM ΚΑΘΗΓΗΤΕΣ_ΦΥΣΙΚΗΣ
        WHERE τίτλος = "Dr";
```

Με αυτόν τον τρόπο δεν χρειάζεται να κάνουμε ξεχωριστό INSERT για κάθε καθηγητή.

1.6.4.2 Ενημέρωση

Η ενημέρωση ουσιαστικά διαγράφει και επανεισάγει μια νέα τιμή. Έστω ότι κάθε χρόνο θα πρέπει να ανανεώνεται το πεδίο ηλικία των φοιτητών:

```
UPDATE Φοιτητές
      SET ηλικία = ηλικία + 1;
```

Έστω ότι το τμήμα Πληροφορικής άλλαξε ονομασία και έγινε Τμήμα Εφαρμοσμένης Πληροφορικής, τότε:

```
UPDATE Τμήματα
      SET ονομασία = "Τμήμα Εφαρμοσμένης Πληροφορικής"
      WHERE ονομασία == "Τμήμα Πληροφορικής";
```

Έστω ότι διπλασιάζουμε τις θέσεις των Αιθουσών, που έχουν πλήθος θέσεων μικρότερο από το μέσο όρο των θέσεων όλων των Αιθουσών, ενώ τις Αίθουσες που έχουν πλήθος θέσεων μεγαλύτερο από το μέσο όρο απλά προσθέτουμε 10 θέσεις

```
UPDATE Αίθουσες
      SET πλήθος_θέσεων = πλήθος_θέσεων * 2
      WHERE πλήθος_θέσεων <
            (SELECT AVG(πλήθος_θέσεων)
            FROM Αίθουσες);

UPDATE Αίθουσες
      SET πλήθος_θέσεων = πλήθος_θέσεων + 10
      WHERE πλήθος_θέσεων >
            (SELECT AVG(πλήθος_θέσεων);
```

Το δεύτερο UPDATE θα πρέπει να εκτελεστεί πρώτο, διαφορετικά μπορεί κάποια αίθουσα να διπλασιάσει αρχικά τις θέσεις της και στη συνέχεια, λόγω του πολλαπλασιασμού να έχει πλήθος θέσεων μεγαλύτερο από το μέσο όρο και να προστεθούν και άλλες 10 θέσεις.

1.6.4.3 Διαγραφή

Η διαγραφή εγγραφών από ένα πίνακα γίνεται με τη φράση DELETE FROM, ως εξής:

```
DELETE FROM <T>
       WHERE C;
```

Όπου Τ είναι ένας πίνακας και C μία συνθήκη WHERE.

Διαγραφή του τμήματος με όνομα Στατιστική:

```
DELETE FROM Φοιτητές
     WHERE ηλικία > 40;
```

Θα πρέπει πρώτα να διαγραφούν και όλες οι εγγραφές του πίνακα Διδασκαλίες που αφορούν τους παραπάνω φοιτητές:

```
DELETE FROM Διδασκαλίες
     WHERE κ_φοιτητή IN
          (SELECT κωδικός
          FROM Φοιτητές
          WHERE ηλικία > 40);
```

Η σειρά διαγραφής είναι σημαντική για την διατήρηση της ακεραιότητας μιας Βάσης. Αν για παράδειγμα διαγράφαμε πρώτα τους φοιτητές, δεν θα είχαμε τους κωδικούς τους, ώστε στη συνέχεια να τους διαγράφαμε από τον πίνακα Διδασκαλίες.

Διαγραφή όλων των φοιτητών με ηλικία μεγαλύτερη από το μέσο όρο ηλικίας όλων των φοιτητών.

```
DELETE FROM Φοιτητές
      WHERE ηλικία >
            (SELECT AVG(ηλικία)
            FROM Φοιτητές);
```

Γενικά, σε περιπτώσεις εισαγωγών, αλλαγών και διαγραφών στοιχείων πινάκων χρειάζεται να ενημερώνουμε και τους πίνακες με τους οποίους συνδέονται οι βασικοί πίνακες, ώστε η ΒΔ να διατηρεί την ακεραιότητα αναφορών (Referential Integrity).

Για παράδειγμα, αν ένας καθηγητής διαγραφεί, τότε θα πρέπει να ενημερωθεί ο πίνακας Διδασκαλίες, ώστε να διαγραφούν και από εκεί όλες οι εγγραφές που περιέχουν ως κωδικό καθηγητή, τον καθηγητή που διαγράφηκε από τον βασικό πίνακα, όπως έγινε στο προηγούμενο παράδειγμα με τους φοιτητές που έχουν ηλικία μεγαλύτερη από 40 έτη.

1.7 Ασφάλεια Βάσεων Δεδομένων

Ασφάλεια ενός συνόλου δεδομένων εννοείται η εμπιστευτικότητα (confidentiality), μυστικότητα (secrecy), ακεραιότητα (integrity) και διαθεσιμότητα (availability) των δεδομένων καθώς και η αποφυγή της αλλαγής ή τροποποίησης (modification), διαγραφής (deletion), καταστροφής (interruption) ή πλαστογράφησης (fabrication) των στοιχείων.

Υπεύθυνος για την ασφάλεια των δεδομένων μιας ΒΔ είναι ο διαχειριστής της Βάσης (Database Administrator).

Πιο συγκεκριμένα, οι απαιτήσεις ασφάλειας στα συστήματα ΒΔ είναι οι παρακάτω:

- Φυσική ακεραιότητα των δεδομένων, ώστε τα δεδομένα να ανθίστανται σε φυσικές καταστροφές ή σε κάθε περίπτωση καταστροφής στοιχείων η ΒΔ να μπορεί να ανακτηθεί (recover). Η ΒΔ μπορεί να καταστραφεί είτε συνολικά είτε κάποια μεμονωμένα στοιχεία της ΒΔ να καταστραφούν. Η τυπική διαδικασία ασφάλειας που ακολουθείται για την προστασία των δεδομένων από φυσικές και άλλες καταστροφές είναι η τήρηση αντιγράφων ασφαλείας (back up).

- Λογική ακεραιότητα των δεδομένων, ώστε η δομή της ΒΔ να παραμένει ακέραια και να μην υπόκειται σε αλλαγές όταν απαιτείται ή να διαχέει τις αλλαγές όταν χρειάζεται.

- Ακεραιότητα των δεδομένων, ώστε τα δεδομένα να είναι ασφαλή και να διαθέτουν όλες τις ιδιότητες που αναφέρθηκαν στην εισαγωγή.

- Δυνατότητα ελέγχου (auditability), ώστε οποιεσδήποτε αλλαγές σε μια ΒΔ να καταγράφεται πότε έγιναν και από ποιους έγιναν.

- Έλεγχος πρόσβασης, ώστε να δίνονται συγκεκριμένα δικαιώματα πρόσβασης σε συγκεκριμένους χρήστες ή ομάδες χρηστών (ρόλοι – roles) για συγκεκριμένους πόρους ή δεδομένα ή τρόπους προσπέλασης. Τα δικαιώματα πρόσβασης μπορεί να είναι προσωπικά για κάθε χρήστη ή ομαδικά σε ομάδες χρηστών. Οι ρόλοι (roles) σε μια ΒΔ είναι ονομασμένες ομάδες (named groups) από συγκεκριμένα δικαιώματα (privileges) που εκχωρούνται (grand) σε άλλους χρήστες ή ομάδες. Οι ρόλοι χρησιμοποιούνται για πιο εύκολη διαχείριση της ασφάλειας σε μια ΒΔ γιατί αν χρειαστεί να αλλάξει ένα δικαίωμα για πολλούς χρήστες, αρκεί να αλλάξει το συγκεκριμένο δικαίωμα για τον αντίστοιχο ρόλο. Επίσης, μπορεί να γίνει επιλεκτική εκχώρηση ρόλων σε ένα χρήστη γεγονός που επιτρέπει τον έλεγχο των δικαιωμάτων κάθε χρήστη σε κάθε κατάσταση της ΒΔ.

- Αυθεντικότητα του χρήση (User authentication), ώστε να εγγυάται η ταυτοποίηση, εξουσιοδότηση και αναγνώριση του χρήστη και για λογιστικούς λόγους (χρήση των πόρων) και για λόγους ασφάλειας (auditing) και για λόγους δικαιωμάτων πρόσβασης.

- Διαθεσιμότητα (availability), που είναι η δυνατότητα πρόσβασης από κάθε εξουσιοδοτημένο χρήστη στα δεδομένα και τους πόρους που είναι εξουσιοδοτημένος.

Ένα σημαντικό πρόβλημα σε μια ΒΔ είναι η εμφάνιση λαθών στο μέσο μιας σειράς αλλαγών (συναλλαγή – transaction), όπου μπορεί κάποια στοιχεία να έχουν νέες τιμές και κάποια παλιές.

Η λύση στο πρόβλημα είναι η ενημέρωση σε δύο φάσεις (two-phase update), όπου στην πρώτη φάση (intent phase) όλα τα στοιχεία προς αλλαγή ανακτώνται και γίνονται οι απαραίτητες αλλαγές που δεν επηρεάζουν όμως τη ΒΔ. Στην δεύτερη φάση (commit phase) της αποστολής προς τη ΒΔ των αλλαγών, οι αλλαγές γίνονται στη Βάση με μόνιμο τρόπο. Σε κάθε βήμα των αλλαγών, στη δεύτερη φάση, αν το σύστημα αποτύχει, τότε ακολουθείται η αντίστροφη διαδικασία (roll back) αποκατάστασης της αρχικής κατάστασης της ΒΔ πριν την διενέργεια των αλλαγών, αλλιώς αν δεν συμβεί λάθος τότε οι αλλαγές είναι μόνιμες.