Η Θεωρία των Βάσεων Δεδομένων

Εισαγωγή στις Βάσεις Δεδομένων

Η αλματώδης ανάπτυξη των Επιστήμων της πληροφοριας και των Τηλεπικοινωνιών (ΤΠΕ) τα τελευταία χρόνια έχει καταστήσει στις ανεπτυγμένες κοινωνίες την πληροφορία ως ένα από τα βασικότερα και πολυτιμότερα αγαθά. Παραδοσιακές Εφαρμογές (Ενημέρωση τραπεζικού λογαριασμού, Κράτηση ξενοδοχείου ή αεροπορικού ταξιδιού, Αναζήτηση βιβλιογραφικών στοιχείων σε κατάλογο βιβλιοθήκης) ή Πολυμεσικά Συστήματα Β.Δ. ή Γεωγραφικά Πληροφοριακά Συστήματα (G.I.S.) είμαι μερικά παραδείγματα.

–

Είναι κοινός τόπος σήμερα η εκτίμηση ότι το αγαθό της πληροφορίας είναι επιθυμητό απ' όλους τους εργαζόμενους αλλά και τους εκπαιδευόμενους. Η εξαγωγή και ανάλυση χρήσιμων πληροφοριών για λήψη αποφάσεων αυξάνει την παραγωγικοτητα στην εργασία τους.

Τα συστήματα βάσεων δεδομένων τα χρησιμοποιούμε για να μπορούμε να αποθηκεύσουμε, να επεξεργαστούμε αλλά και να εκμεταλλευτούμε αποδοτικά αυτόν τον τεράστιο όγκο των πληροφοριών που αυξάνονται με αλματώδεις ρυθμούς καθημερινά.

Τα Δεδομένα και οι Πληροφορίες

Με τον όρο πληροφορία αναφερόμαστε συνήθως σε ειδήσεις, γεγονότα και έννοιες που αποκτάμε από την καθημερινή μας επικοινωνία και τα θεωρούμε ως αποκτηθείσα γνώση, ενώ τα δεδομένα μπορούν να είναι μη κατάλληλα επεξεργασμένα και μη ταξινομημένα σύνολα πληροφοριών. Ένας αυστηρός ορισμός για το τι είναι δεδομένα και τι είναι πληροφορία, σύμφωνα με την επιτροπή ANSI των ΗΠΑ, είναι ο εξής :

- **Δεδομένα (data)** είναι μια παράσταση, όπως γράμματα, αριθμοί, σύμβολα κ.ά. στα οποία μπορούμε να δώσουμε κάποια σημασία (έννοια).
- **Πληροφορία (information)** είναι η σημασία που δίνουμε σ' ένα σύνολο από δεδομένα, τα οποία μπορούμε να επεξεργαστούμε βάσει προκαθορισμένων κανόνων και να βγάλουμε έτσι κάποια χρήσιμα συμπεράσματα. Με τις πληροφορίες περιορίζεται η αβεβαιότητα που έχουμε για διάφορα πράγματα και βοηθιόμαστε έτσι στο να λάβουμε σωστές αποφάσεις.

Τα δεδομένα μπορούν να θεωρηθούν ως τρόποι αναπαράστασης εννοιών και γεγονότων τεράστιου όγκου δεδομένων όπως απαιτούν οι κοινωνικές συνθήκες σήμερα, δεν λύνει τελείως το πρόβλημα της σωστής οργάνωσης και ταξινόμησης των δεδομένων. Τα δεδομένα θα πρέπει να οργανωθούν με τέτοιο τρόπο έτσι ώστε να μπορούμε να τα εντοπίζουμε και να τα αξιοποιούμε εύκολα και γρήγορα και τη στιγμή που τα χρειαζόμαστε.

Σημείωση: για τις βάσεις δεδομένων οι όροι Δεδομένα και Πληροφορίες θα πρέπει να θεωρούνται συνώνυμοι γιατί και τα δύο ειναι αποθηκευμένες τιμές σε μία οργανωμένη δομή

Ένα κλασικό παράδειγμα μη σωστής οργάνωσης δεδομένων θα ήταν για παράδειγμα ο τηλεφωνικός κατάλογος μιας πόλης, όπου οι συνδρομητές δεν θα ήταν καταχωρημένοι αλφαβητικά σύμφωνα με το επώνυμο και το όνομά τους, αλλά εντελώς τυχαία. Ένας τέτοιος

τηλεφωνικός κατάλογος θα περιείχε μια τεράστια ποσότητα δεδομένων αλλά θα ήταν ουσιαστικά άχρηστος.

Ένα άλλο παράδειγμα μη σωστής οργάνωσης δεδομένων θα ήταν μια πολύ μεγάλη βιβλιοθήκη με χιλιάδες τόμους βιβλίων και χωρίς να διαθέτει κάποιο υποτυπώδες σύστημα οργάνωσης και ταξινόμησης των βιβλίων. Ούτε οι υπάλληλοι της βιβλιοθήκης θα μπορούσαν να κάνουν τη δουλειά τους αλλά ούτε και οι επισκέπτες θα μπορούσαν να αξιοποιήσουν την πληθώρα των πληροφοριών που περιέχονται στα βιβλία. Εκτός λοιπόν από τη μόνιμη αποθήκευση των δεδομένων, χρειαζόμαστε και κάποιους τρόπους ευέλικτης και αποδοτικής οργάνωσής τους.

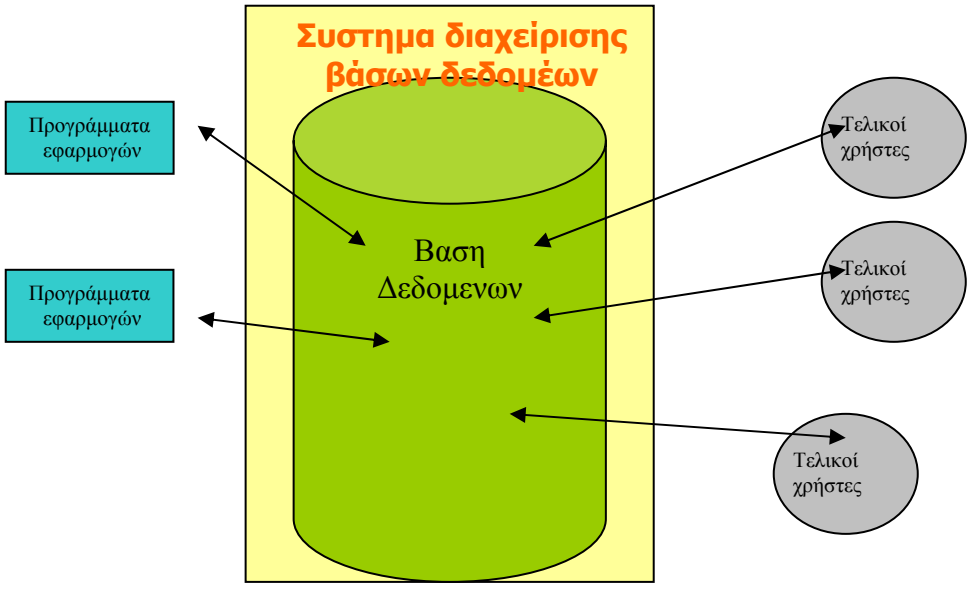
Χαρακτηριστικά παραδείγματα δεδομένων που απαιτούν σωστή και αποδοτική οργάνωση είναι τα εξής :

- Τα στοιχεία υπαλλήλων, πελατών, προμηθευτών και παραγγελιών μιας εμπορικής επιχείρησης.
- Τα στοιχεία υλικών μιας αποθήκης.
- Τα στοιχεία ταινιών, πελατών και δανεισμών μιας βιντεολέσχης.
- Τα στοιχεία υπαλλήλων, γιατρών, ασθενών αλλά και υλικών ενός νοσοκομείου.
- Τα στοιχεία βιβλίων, χρηστών (δανειστών) και δανεισμών μιας βιβλιοθήκης.

Συστατικά Βάσης Δεδομένων

Μια βάση Δεδομένων απαρτίζεται από τέσσερα βοσικά δομικάς τοιχεία

- Τα αποθηκευμένα στοιχεία (δεδομένα ή πληροφορίες)
- Το υλικό (hardware)
- Το λογισμικό (software)
- Τους χρήστες



Ειδικότερα ανόλογα με το είδος του Υλικού υπάρχουν «μεγάλα συστήματα» και «μικρά συστήματα» , ενώ σε σχέση με τον πόσοι χρήστες έχουν πρόσβαση στη βάση δεδομένω την ίδια χρονική στιγμή σε¨:
Σύστημα ενός χρήστη(single user system) και
Σύστημα πολλων χρηστών (multiuser system)

Η Οργάνωση Αρχείων

Ο πιο γνωστός τρόπος οργάνωσης δεδομένων με τη χρήση ηλεκτρονικών υπολογιστών είναι σε αρχεία εγγραφών. Για να κατανοήσουμε καλύτερα ορισμένες έννοιες, θα εξετάσουμε την περίπτωση ενός αρχείου πελατών και παραγγελιών μιας εμπορικής επιχείρησης. Για να οργανώσουμε σωστά το αρχείο μας, θα πρέπει να δημιουργήσουμε καρτέλες για τους πελάτες, αλλά και για τις παραγγελίες τους αργότερα, που θα πρέπει να περιέχουν τα εξής στοιχεία ανά πελάτη :

- Κωδικός
- Επώνυμο
- Όνομα
- Διεύθυνση
- ΤΚ
- Πόλη
- Τηλέφωνο
- ΑΦΜ
- ΔΟΥ

Η αντιστοίχηση του παλιού τρόπου οργάνωσης με τις καρτέλες σε σχέση με τον σύγχρονο ηλεκτρονικό τρόπο οργάνωσης, έχει ως εξής :

- Συρτάρι – Αρχείο Δεδομένων
- Καρτέλα πελάτη – Εγγραφή του αρχείου δεδομένων
- Στοιχείο της καρτέλας – Πεδίο της εγγραφής

Ένα αρχείο (file) θα μπορούμε να το χαρακτηρίσουμε σαν ένα σύνολο που αποτελείται από οργανωμένα ομοειδή στοιχεία. Τα στοιχεία ενός αρχείου μπορούμε να τα οργανώσουμε σε λογικές ενότητες και το σύνολο των στοιχείων που περιέχει μια λογική ενότητα καλείται εγγραφή (record). Το κάθε στοιχείο της εγγραφής καλείται πεδίο (field). Το πεδίο αποτελεί και τη μικρότερη δυνατή υποδιαίρεση των στοιχείων ενός αρχείου. Ένα πεδίο χαρακτηρίζεται από τον μέγιστο αριθμό των χαρακτήρων (bytes) που απαιτούνται για την καταχώρησή του στη μνήμη του υπολογιστή και που αποκαλείται μήκος του πεδίου (field length).

Σε μια οργάνωση αρχείου όπως είναι οι πελάτες μιας εμπορικής επιχείρησης που είδαμε νωρίτερα, τα αντίστοιχα πεδία όλων των εγγραφών καταλαμβάνουν τον ίδιο αριθμό σε bytes που είναι αυτός που έχουμε ορίσει κατά τη δημιουργία του αρχείου. Για παράδειγμα, αν ορίσαμε ότι το πεδίο Επώνυμο θα έχει μήκος 15 χαρακτήρες, τότε το πεδίο της εγγραφής του πελάτη με επώνυμο Παπαδόπουλος, αλλά και το πεδίο της εγγραφής του πελάτη με επώνυμο Βες θα καταλαμβάνουν από 15 bytes στη μνήμη του υπολογιστή, ενώ αν ένας πελάτης ονομάζεται Παπαχριστοδουλόπουλος, τότε θα γίνει αποκοπή του επωνύμου του και θα καταχωρηθούν στη μνήμη του υπολογιστή μόνο τα 15 πρώτα γράμματα, δηλ. τα Παπαχριστοδουλό.

Ένα πεδίο χαρακτηρίζεται ακόμη και από το είδος των δεδομένων που μπορεί να περιέχει, όπως :

- Αλφαριθμητικό (alphanumeric), μπορεί να περιέχει γράμματα, ψηφία ή και ειδικούς χαρακτήρες.
- Αριθμητικό (numeric), μπορεί να περιέχει μόνο αριθμούς.
- Αλφαβητικό (alphabetic), μπορεί να περιέχει μόνο γράμματα (αλφαβητικούς χαρακτήρες).
- Ημερομηνίας (date), μπορεί να περιέχει μόνο ημερομηνίες.

- Δυαδικό (binary), μπορεί να περιέχει ειδικού τύπου δεδομένα, όπως εικόνες, ήχους κ.ά.
- Λογικό (logical), μπορεί να περιέχει μόνο μία από δύο τιμές, οι οποίες αντιστοιχούν σε δύο διακριτές καταστάσεις και μπορούν να χαρακτηρισθούν σαν 0 και 1 ή σαν αληθές (true) και ψευδές (false).
- Σημειώσεων (memo), μπορεί να περιέχει κείμενο με μεταβλητό μήκος, το οποίο μπορεί να είναι και αρκετά μεγάλο και συνήθως αποθηκεύεται σαν ξεχωριστό αρχείο από το κύριο αρχείο.

Όσον αφορά τις εγγραφές, χρήσιμοι ορισμοί είναι οι εξής :

- Μήκος εγγραφής (record length) καλείται το άθροισμα που προκύπτει από τα μήκη των πεδίων που την αποτελούν.
- Δομή εγγραφής (record layout) ή γραμμογράφηση καλείται ο τρόπος με τον οποίο οργανώνουμε τα πεδία μιας εγγραφής.
- Διάβασμα (read) από αρχείο σημαίνει τη μεταφορά των δεδομένων του αρχείου, που γίνεται συνήθως ανά μία εγγραφή, από το μέσο αποθήκευσης (σκληρό δίσκο ή δισκέτα) στην κεντρική μνήμη του υπολογιστή για επεξεργασία.
- Γράψιμο (write) σε αρχείο σημαίνει μεταφορά των δεδομένων του αρχείου, που γίνεται συνήθως ανά μία εγγραφή, από την κεντρική μνήμη του υπολογιστή στο μέσο αποθήκευσης (σκληρό δίσκο ή δισκέτα).

Προβλήματα της Οργάνωσης Αρχείων

Στα αρχικά στάδια της οργάνωσης αρχείων, ήταν πολύ συνηθισμένη πρακτική η δημιουργία ξεχωριστών εφαρμογών (προγραμμάτων) και ξεχωριστών αρχείων, όπως για παράδειγμα η δημιουργία ενός αρχείου πελατών και ενός άλλου ανεξάρτητου αρχείου για τις παραγγελίες των πελατών. Τα προβλήματα που προέκυψαν από την πρακτική αυτή είναι τα εξής :

- **Πλεονασμός των δεδομένων (data redundancy)**. Υπάρχει η περίπτωση να έχουμε επανάληψη των ίδιων δεδομένων σε αρχεία διαφορετικών εφαρμογών. Για παράδειγμα, αν έχουμε ένα αρχείο πελατών και ένα αρχείο παραγγελιών αυτών των πελατών, είναι σχεδόν σίγουρο ότι θα υπάρχουν κάποια στοιχεία των πελατών που θα υπάρχουν και στα δύο αρχεία.
- **Ασυνέπεια των δεδομένων (data inconsistency)**. Αυτό μπορεί να συμβεί όταν υπάρχουν τα ίδια στοιχεία των πελατών (πλεονασμός) και στο αρχείο πελατών και στο αρχείο παραγγελιών και χρειασθεί να γίνει κάποια αλλαγή στη διεύθυνση ή στα τηλέφωνα κάποιου πελάτη, οπότε είναι πολύ πιθανό να γίνει η διόρθωση μόνο στο ένα αρχείο και όχι και στο άλλο.
- **Αδυναμία μερισμού δεδομένων (data sharing)**. Μερισμός δεδομένων σημαίνει δυνατότητα για κοινή χρήση των στοιχείων κάποιων αρχείων. Για παράδειγμα, ο μερισμός δεδομένων θα ήταν χρήσιμος αν με την παραγγελία ενός πελάτη μπορούμε να έχουμε πρόσβαση την ίδια στιγμή στο αρχείο πελατών για να δούμε το υπόλοιπο του πελάτη και μετά στο αρχείο της αποθήκης για να δούμε αν είναι διαθέσιμα τα προϊόντα που παρήγγειλε ο συγκεκριμένος πελάτης. Η αδυναμία μερισμού δεδομένων δημιουργεί καθυστέρηση στη λήψη αποφάσεων και στην εξυπηρέτηση των χρηστών.
- **Αδυναμία προτυποποίησης**. Έχει να κάνει με την ανομοιομορφία και με την διαφορετική αναπαράσταση και οργάνωση των δεδομένων στα αρχεία των εφαρμογών. Η αδυναμία αυτή δημιουργεί προβλήματα προσαρμογής των χρηστών

καθώς και προβλήματα στην ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων.

Οι Βάσεις Δεδομένων και τα ΣΔΒΔ (DBMS)

Για να δοθεί μια λύση σ' όλα τα παραπάνω προβλήματα, και με βάση το γεγονός ότι η χρήση των ηλεκτρονικών υπολογιστών και συνεπώς η ηλεκτρονική καταχώρηση και επεξεργασία δεδομένων αυξήθηκε κατακόρυφα ήδη από τη δεκαετία του '70 στις μεγάλες επιχειρήσεις και άρα είχαμε πάρα πολλές εφαρμογές να επεξεργάζονται δεδομένα σε πάρα πολλά αρχεία ταυτόχρονα, προτάθηκε η συνένωση όλων των αρχείων μιας εφαρμογής. Εκτός, όμως, από τη συνένωση των αρχείων, απαιτείτο και μια σωστή οργάνωσή τους. Δημιουργήθηκαν έτσι οι Τράπεζες Πληροφοριών ή Βάσεις Δεδομένων (Data Bases).

Μια **Βάση Δεδομένων (ΒΔ)** είναι ένα σύνολο αρχείων με υψηλό βαθμό οργάνωσης τα οποία είναι συνδεδεμένα μεταξύ τους με λογικές σχέσεις, έτσι ώστε να μπορούν να χρησιμοποιούνται από πολλές εφαρμογές και από πολλούς χρήστες ταυτόχρονα. Υπάρχει ένα ειδικό λογισμικό το οποίο μεσολαβεί ανάμεσα στις αρχεία δεδομένων και τις εφαρμογές που χρησιμοποιούν οι χρήστες και αποκαλείται **Σύστημα Διαχείρισης Βάσης Δεδομένων (ΣΔΒΔ) ή DBMS (Data Base Management System)**. Το ΣΔΒΔ είναι στην ουσία ένα σύνολο από προγράμματα και υπορουτίνες που έχουν να κάνουν με τον χειρισμό της βάσης δεδομένων, όσον αφορά τη δημιουργία, τροποποίηση, διαγραφή στοιχείων, με ελέγχους ασφαλείας κ.ά.

Οι χρήστες των εφαρμογών αντλούν τα στοιχεία που τους ενδιαφέρουν από τη βάση δεδομένων χωρίς να είναι σε θέση να γνωρίζουν με ποιο τρόπο είναι οργανωμένα τα δεδομένα σ' αυτήν. Το ΣΔΒΔ παίζει τον ρόλο του μεσάζοντα ανάμεσα στον χρήστη και τη βάση δεδομένων και μόνο μέσω του ΣΔΒΔ μπορεί ο χρήστης να αντλήσει πληροφορίες από τη βάση δεδομένων. Ένα ΣΔΒΔ μπορεί να είναι εγκατεστημένο σ' έναν μόνο υπολογιστή ή και σ' ένα δίκτυο υπολογιστών και μπορεί να χρησιμοποιείται από έναν χρήστη ή και από πολλούς χρήστες.

Ένα Σύστημα Βάσης Δεδομένων (ΣΒΔ) ή DBS (Data Base System) αποτελείται από το υλικό, το λογισμικό, τη βάση δεδομένων και τους χρήστες. Είναι δηλαδή ένα σύστημα με το οποίο μπορούμε να αποθηκεύσουμε και να αξιοποιήσουμε δεδομένα με τη βοήθεια ηλεκτρονικού υπολογιστή. Αναλυτικά :

- Το **υλικό (hardware)** αποτελείται όπως είναι γνωστό από τους ηλεκτρονικούς υπολογιστές, τα περιφερειακά, τους σκληρούς δίσκους, τις μαγνητικές ταινίες κ.ά., όπου είναι αποθηκευμένα τα αρχεία της βάσης δεδομένων αλλά και τα προγράμματα που χρησιμοποιούνται για την επεξεργασία τους.
- Το **λογισμικό (software**) είναι τα προγράμματα που χρησιμοποιούνται για την επεξεργασία των δεδομένων (στοιχείων) της βάσης δεδομένων.
- Η **βάση δεδομένων (data base)** αποτελείται από το σύνολο των αρχείων όπου είναι αποθηκευμένα τα δεδομένα του συστήματος. Τα στοιχεία αυτά μπορεί να βρίσκονται αποθηκευμένα σ' έναν φυσικό υπολογιστή αλλά και σε περισσότερους. Όμως, στον χρήστη δίνεται η εντύπωση ότι βρίσκονται συγκεντρωμένα στον ίδιο υπολογιστή. Τα δεδομένα των αρχείων αυτών είναι ενοποιημένα (data integration), δηλ. δεν υπάρχει πλεονασμός (άσκοπη επανάληψη) δεδομένων και μερισμένα (data sharing), δηλ. υπάρχει δυνατότητα ταυτόχρονης προσπέλασης των δεδομένων από πολλούς χρήστες. Ο κάθε χρήστης έχει διαφορετικά δικαιώματα και βλέπει διαφορετικό κομμάτι της βάσης δεδομένων, ανάλογα με τον σκοπό για τον οποίο συνδέεται.
- Οι **χρήστες (users)** μιας βάσης δεδομένων χωρίζονται στις εξής κατηγορίες :

- Τελικοί χρήστες (end users). Χρησιμοποιούν κάποια εφαρμογή για να παίρνουν στοιχεία από μια βάση δεδομένων, έχουν τις λιγότερες δυνατότητες επέμβασης στα στοιχεία της βάσης δεδομένων, χρησιμοποιούν ειδικούς κωδικούς πρόσβασης και το σύστημα τούς επιτρέπει ανάλογα πρόσβαση σε συγκεκριμένο κομμάτι της βάσης δεδομένων.
- Προγραμματιστές εφαρμογών (application programmers). Αναπτύσσουν τις εφαρμογές του ΣΒΔ σε κάποια από τις γνωστές γλώσσες προγραμματισμού.
- Διαχειριστής δεδομένων (data administrator – DA). Έχει τη διοικητική αρμοδιότητα και ευθύνη για την οργάνωση της βάσης δεδομένων και την απόδοση δικαιωμάτων πρόσβασης στους χρήστες.
- Διαχειριστής βάσης δεδομένων (database administrator – DBA). Λαμβάνει οδηγίες από τον διαχειριστή δεδομένων και είναι αυτός που διαθέτει τις τεχνικές γνώσεις και αρμοδιότητες για τη σωστή και αποδοτική λειτουργία του ΣΔΒΔ.

Η Αρχιτεκτονική των ΣΔΒΔ

Όπως είδαμε νωρίτερα, ένα ΣΔΒΔ (Σύστημα Διαχείρισης Βάσης Δεδομένων) έχει σαν αποστολή τη διαχείριση των δεδομένων των αρχείων της βάσης, δηλ. την προσθήκη, διαγραφή, τροποποίηση εγγραφών, την αναζήτηση μέσα στις εγγραφές κ.ά.). Το ΣΔΒΔ δέχεται αιτήσεις από τους χρήστες των εφαρμογών και επικοινωνεί με τα αρχεία της βάσης δεδομένων για να τις διεκπεραιώσει.

Αυτή η κοινή διεπαφή (interface) των εφαρμογών με τα αρχεία αποκαλείται **λογική διεπαφή**. Οι εφαρμογές που δημιουργούμε δεν απασχολούνται με τον τρόπο που είναι αποθηκευμένα τα δεδομένα, πόσο χώρο καταλαμβάνουν κοκ και αυτή η ιδιότητα είναι γνωστή ως **ανεξαρτησία δεδομένων**.

Αυτό σημαίνει πρακτικά ότι οποιαδήποτε αλλαγή στον τρόπο οργάνωσης των αρχείων της βάσης δεδομένων δεν θα συνεπάγεται και αλλαγή στις εφαρμογές· ένα πρόβλημα που ταλαιπωρούσε πολύ τους προγραμματιστές παλαιοτέρων εποχών. Ακόμη, η προσθήκη, η κατάργηση ή και η τροποποίηση κάποιων εφαρμογών δεν θα έχει καμία επίπτωση στον τρόπο οργάνωσης των αρχείων της βάσης δεδομένων. Στα ΣΔΒΔ έχει επικρατήσει η λεγόμενη αρχιτεκτονική των τριών επιπέδων (βαθμίδων), όπου τα τρία επίπεδα είναι τα εξής :

- **Εσωτερικό επίπεδο (internal level)**, έχει να κάνει με την αποθήκευση των αρχείων στον σκληρό δίσκο, δηλ. την πραγματική ή φυσική κατάστασή τους.
- **Εξωτερικό επίπεδο (external level)**, έχει να κάνει με τους χρήστες είτε αυτοί είναι απλοί χειριστές, είτε προγραμματιστές ή και οι διαχειριστές της βάσης δεδομένων.
- **Εννοιολογικό επίπεδο (conceptual level)**, είναι ένα ενδιάμεσο επίπεδο που διασυνδέει τα δύο άλλα επίπεδα και έχει να κάνει με τη λογική σχεδίαση των

Οι Οντότητες (Entities)

Με τον όρο **οντότητα (entity)** εννοούμε ένα αντικείμενο, ένα πρόσωπο, μια κατάσταση και γενικά ο,τιδήποτε μπορεί να προσδιορισθεί σαν ανεξάρτητη ύπαρξη (αυτόνομη μονάδα του φυσικού κόσμου). Για παράδειγμα, σε μια βάση δεδομένων μιας εμπορικής εταιρείας, οντότητες μπορεί να είναι οι εργαζόμενοι, οι πελάτες, οι προμηθευτές, οι παραγγελίες, τα είδη της αποθήκης (προϊόντα) κ.ά.

Το **Μοντέλο Οντοτήτων Συσχετίσεων (Entity Relationship Model, ER Model)** είναι μια διαγραμματική αναπαράσταση της δομής μιας βάσης δεδομένων και χρησιμοποιείται κατά τη φάση του λογικού σχεδιασμού της βάσης. Δηλαδή, δεν ασχολείται

με τον τρόπο που αποθηκεύονται τα δεδομένα της βάσης, αλλά με την ταυτοποίηση των δεδομένων και με τον τρόπο με τον οποίο αυτά συσχετίζονται μεταξύ τους.

Θα δούμε ένα παράδειγμα μιας εταιρείας, η οποία περιέχει δεδομένα που αφορούν τους υπαλλήλους της (employees), τα τμήματά της (departments) και τα έργα (projects) που έχουν αναλάβει αυτά τα τμήματα. Ένα τμήμα της εταιρείας μπορεί να εποπτεύει ένα ή περισσότερα έργα (projects) και ένας υπάλληλος ανήκει σ' ένα μόνο τμήμα της εταιρείας αλλά μπορεί να απασχολείται ταυτόχρονα σε πολλά έργα, τα οποία δεν είναι υποχρεωτικό να παρακολουθούνται από το ίδιο τμήμα.

Οι Ιδιότητες (Attributes)

Με τον όρο **ιδιότητα** ή **χαρακτηριστικό** ή και **πεδίο (attribute)** μιας οντότητας, αναφερόμαστε σ' ένα από τα συστατικά της στοιχεία που την περιγράφουν και την κάνουν να ξεχωρίζει από τα άλλα στοιχεία της ίδιας οντότητας. Για παράδειγμα, η οντότητα ΠΕΛΑΤΗΣ μπορεί να έχει ως ιδιότητες (χαρακτηριστικά) τον κωδικό, το επώνυμο, το όνομα, τη διεύθυνση, το τηλέφωνο, το ΑΦΜ κ.ά., με τη βοήθεια των οποίων μπορούμε να ξεχωρίσουμε τους πελάτες μεταξύ τους.

Επίσης, η οντότητα ΠΑΡΑΓΓΕΛΙΑ μπορεί να έχει ως ιδιότητες (χαρακτηριστικά) τον κωδικό, τον αριθμό παραστατικού, την ημερομηνία, τον κωδικό πελάτη, το προϊόν κ.ά., με τη βοήθεια των οποίων μπορούμε να ξεχωρίσουμε τις παραγγελίες μεταξύ τους Στο παράδειγμα της εταιρείας, μπορούμε να ορίσουμε έναν τύπο οντότητας για τους υπαλλήλους της εταιρείας (EMPLOYEE), έναν τύπο οντότητας για τα τμήματα που έχει η εταιρεία (DEPARTMENT) και έναν τύπο οντότητας για τα έργα που έχει αναλάβει η εταιρεία (PROJECT). Καθένας από τους παραπάνω τύπους οντοτήτων περιγράφεται από ένα όνομα και από το σύνολο των πεδίων που περιέχει. Οι πληροφορίες αυτές αποτελούν το **σχήμα (schema)** της οντότητας.

Τα Στιγμιότυπα (Snapshots)

Το κάθε διαφορετικό (αυτόνομο) στοιχείο μιας οντότητας αποκαλείται στιγμιότυπο (snapshot) ή και εμφάνιση της οντότητας. Για παράδειγμα, στην οντότητα ΠΕΛΑΤΗΣ, άλλο στιγμιότυπο είναι ο πελάτης με επώνυμο Παπαδόπουλος και άλλο στιγμιότυπο είναι ο πελάτης με επώνυμο Σουμπάσης.

Το Πρωτεύον Κλειδί (Primary Key)

Πρωτεύον κλειδί ή πεδίο κλειδί (primary key) μιας οντότητας καλείται εκείνη η ιδιότητα (ή ο συνδυασμός ιδιοτήτων) που έχει μοναδική τιμή για όλα τα στιγμιότυπα (εμφανίσεις) της οντότητας. Για παράδειγμα, στην οντότητα ΠΕΛΑΤΗΣ πρωτεύον κλειδί είναι ο κωδικός πελάτη, στην οντότητα ΠΑΡΑΓΓΕΛΙΑ πρωτεύον κλειδί μπορεί να είναι ο κωδικός παραγγελίας ή ο αριθμός παραστατικού κοκ.

Υπάρχουν περιπτώσεις όπου το πεδίο κλειδί ενός τύπου οντότητας μπορεί να μην είναι απλό αλλά σύνθετο, να αποτελείται δηλαδή από πολλά απλά πεδία και τότε η συνθήκη της μοναδικότητας για την τιμή του κλειδιού δεν εφαρμόζεται σε κάθε πεδίο του σύνθετου κλειδιού αλλά στο σύνολο του συνδυασμού αυτών των πεδίων.

Οι Συσχετίσεις (Relationships)

Με τον όρο **συσχέτιση (relationship)** αναφερόμαστε στον τρόπο σύνδεσης (επικοινωνίας) δύο ξεχωριστών οντοτήτων, ώστε να μπορούμε να αντλούμε στοιχεία (πληροφορίες) από τον συνδυασμό τους.

Για παράδειγμα, η οντότητα ΓΙΑΤΡΟΣ συσχετίζεται με την οντότητα ΑΣΘΕΝΗΣ αλλά και με την οντότητα ΚΛΙΝΙΚΗ στη βάση δεδομένων ενός νοσοκομείου. Μπορούμε

να δεχθούμε ότι ένας γιατρός παρακολουθεί (συσχετίζεται με) πολλούς ασθενείς, αλλά ένας ασθενής παρακολουθείται από (συσχετίζεται με) έναν μόνο γιατρό και επίσης ένας γιατρός συσχετίζεται με (ανήκει σε) μία μόνο κλινική, αλλά μια κλινική συσχετίζεται με (απασχολεί) πολλούς γιατρούς.

Στο παράδειγμα της εταιρείας, η οντότητα EMPLOYEE συσχετίζεται με την οντότητα DEPARTMENT και η οντότητα DEPARTMENT συσχετίζεται με την οντότητα PROJECTS. Ένας υπάλληλος ανήκει σ' ένα μόνο τμήμα και ένα τμήμα μπορεί να έχει πολλούς υπαλλήλους. Επίσης, ένα τμήμα εποπτεύει πολλά έργα αλλά ένα έργο εποπτεύεται από ένα μόνο τμήμα.

Το Ιεραρχικό Μοντέλο Βάσεων Δεδομένων

Υπάρχουν τρία βασικά μοντέλα που έχουν επικρατήσει στις βάσεις δεδομένων, το ιεραρχικό, το δικτυωτό και το σχεσιακό, και τα οποία αναπτύχθηκαν με βάση αντίστοιχες δομές. Το ιεραρχικό μοντέλο (hierarchical) έχει μια ιεραρχική δομή που θυμίζει δένδρο. Οι οντότητες μοιάζουν με απολήξεις από κλαδιά δένδρων και τοποθετούνται σε επίπεδα ιεραρχίας. Τα κλαδιά παριστάνουν τις συσχετίσεις ανάμεσα στις οντότητες.

Από μια οντότητα που βρίσκεται σ' ένα ανώτερο επίπεδο εκκινούν πολλά κλαδιά, καθένα από τα οποία καταλήγει σε μια οντότητα που βρίσκεται σ' ένα χαμηλότερο επίπεδο. Αλλά, σε κάθε οντότητα που βρίσκεται σ' ένα χαμηλότερο επίπεδο αντιστοιχεί μία και μόνο μία οντότητα που βρίσκεται σ' ένα ανώτερο επίπεδο. Το μοντέλο αυτό ήταν το πρώτο που εμφανίσθηκε αλλά σήμερα θεωρείται δύσχρηστο και ξεπερασμένο.

Το Δικτυωτό Μοντέλο Βάσεων Δεδομένων

Και στο δικτυωτό (network) μοντέλο, τα στοιχεία τοποθετούνται σ' ένα επίπεδο ιεραρχίας, αλλά κάθε στοιχείο μπορεί να συσχετισθεί με πολλά στοιχεία είτε σ' ένα κατώτερο ή σ' ένα ανώτερο επίπεδο.

Το Σχεσιακό Μοντέλο Βάσεων Δεδομένων

Το σχεσιακό (relational) μοντέλο έχει επικρατήσει σήμερα στην αναπαράσταση των δεδομένων καθώς διαθέτει σημαντικά πλεονεκτήματα ως προς τα άλλα δύο και οι βάσεις δεδομένων που σχεδιάζονται σύμφωνα μ' αυτό αποκαλούνται σχεσιακές (relational databases). Με τις σχεσιακές βάσεις δεδομένων διαθέτουμε έναν σαφή, απλό και εύκολα κατανοητό τρόπο για να μπορέσουμε να αναπαραστήσουμε και να διαχειριστούμε τα δεδομένα μας. Υστερούν μόνο σε ταχύτητα υπολογισμών και σε χώρο αποθήκευσης, αλλά μόνο όταν έχουμε να κάνουμε με πολύ μεγάλες βάσεις δεδομένων.

Στο μοντέλο αυτό οι βάσεις δεδομένων περιγράφονται με αυστηρές μαθηματικές έννοιες και ο χρήστης βλέπει τις οντότητες και τις συσχετίσεις με τη μορφή πινάκων (tables) και σχέσεων (relations) αντίστοιχα.

Ένας **πίνακας (table)** αποτελείται από γραμμές (rows) και στήλες (columns), όπου τοποθετούμε τα στοιχεία σε οριζόντια και κάθετη μορφή. Η κάθε στήλη του πίνακα χαρακτηρίζει κάποια ιδιότητα της οντότητας και αποκαλείται **χαρακτηριστικό (attribute)** ή **πεδίο (field)**, ενώ η κάθε γραμμή του πίνακα περιέχει όλες τις πληροφορίες (στήλες) που αφορούν ένα στοιχείο της οντότητας και αποκαλείται **πλειάδα (tuple)** ή **εγγραφή (record)**.

Κάθε πεδίο του πίνακα μπορεί να πάρει ορισμένες μόνο τιμές, οι οποίες μπορεί να καθορίζονται από τον τύπο δεδομένων της ιδιότητας, όπως ονόματα ή αριθμοί για παράδειγμα, ή και από αυτό που εκφράζει, όπως το ότι δεν μπορούμε να έχουμε αρνητικό βάρος ή αρνητικό ΑΦΜ, για παράδειγμα. Το σύνολο των αποδεκτών τιμών μιας οντότητας αποκαλείται **πεδίο ορισμού (domain**).

Για να μπορέσουμε να κατανοήσουμε τις σχεσιακές βάσεις δεδομένων, ένα πολύ χαρακτηριστικό παράδειγμα αποτελεί ένας πίνακας πελατών και ένας πίνακας παραγγελιών μιας εμπορικής εταιρείας.

Τα πεδία που μπορούμε να ορίσουμε στους πίνακες αυτούς είναι τα εξής :

ΠΙΝΑΚΑΣ (ΟΝΤΟΤΗΤΑ) ΠΕΛΑΤΕΣ

(ΚωδικόςΠελάτη, Επώνυμο, Όνομα, Διεύθυνση, ΤΚ, Πόλη, ΑΦΜ, Υπόλοιπο)

ΠΙΝΑΚΑΣ (ΟΝΤΟΤΗΤΑ) ΠΑΡΑΓΓΕΛΙΕΣ

(ΚωδικόςΠελάτη, ΚωδικόςΠαραγγελίας, Ημερομηνία, Είδος, Ποσότητα, ΤιμήΜονάδας)

Βλέπουμε ότι οι δύο πίνακες έχουν ένα κοινό πεδίο (στήλη), τον ΚωδικόςΠελάτη, και αυτό είναι απαραίτητο στις σχεσιακές βάσεις δεδομένων για να μπορέσουμε να κάνουμε τη δουλειά μας και να συνδυάσουμε πληροφορίες και από τους δύο πίνακες.

Όπως είναι εύκολα κατανοητό, η βασικότερη εργασία που έχουμε να κάνουμε κατά τον σχεδιασμό μιας σχεσιακής βάσης δεδομένων είναι να ορίσουμε τους πίνακες που θα χρησιμοποιήσουμε καθώς και τα πεδία που θα περιέχει ο καθένας απ' αυτούς. Η διαδικασία αυτή αποκαλείται κατασκευή του σχήματος (schema) μιας βάσης δεδομένων.

Οι κανόνες που πρέπει να ακολουθούμε πιστά κατά τον σχεδιασμό μιας σχεσιακής βάσης δεδομένων είναι οι εξής :

- Η κάθε οντότητα πρέπει να παριστάνεται ως ένας ξεχωριστός πίνακας.
- Η κάθε στήλη του πίνακα αντιστοιχεί σε μια ιδιότητα της οντότητας.
- Η κάθε γραμμή του πίνακα αντιστοιχεί σε μια εμφάνιση της οντότητας.
- Η κάθε γραμμή πρέπει να είναι μοναδική, δηλ. αποκλείεται να υπάρχουν δύο ή και περισσότερες γραμμές που να περιέχουν τα ίδια ακριβώς στοιχεία.
- Η σειρά εμφάνισης των γραμμών δεν έχει καμία σημασία.
- Η κάθε στήλη έχει μια δική της μοναδική ονομασία.
- Οι τιμές που ανήκουν στην ίδια στήλη πρέπει να είναι του ιδίου τύπου, δηλ. ή όλες αριθμοί ή όλες αλφαριθμητικές κοκ.
- Η στήλη που αποτελεί το πρωτεύον κλειδί (primary key) μιας οντότητας, δεν πρέπει να είναι ποτέ κενή (null).
- Αποκλείεται να υπάρχουν δύο ή και περισσότερες γραμμές που να περιέχουν την ίδια τιμή στο πρωτεύον κλειδί.
- Το πρωτεύον κλειδί μιας οντότητας αποκαλείται ξένο κλειδί (foreign key) σε μια άλλη οντότητα, με την οποία υπάρχει συσχετισμός.
- Μπορεί να υπάρχουν πολλές γραμμές που να έχουν την ίδια τιμή στο ξένο κλειδί.

Τα Σχεσιακά ΣΔΒΔ (RDBMS)

Τα Σχεσιακά Συστήματα Διαχείρισης Βάσεων Δεδομένων (ΣΣΔΒΔ) ή RBMS (Relational DataBase Management Systems) αναπτύχθηκαν με βάση το σχεσιακό μοντέλο και έχουν επικρατήσει πλήρως στον χώρο. Κατά τον σχεδιασμό και τη δημιουργία μιας σχεσιακής βάσης δεδομένων, οι πίνακες αποτελούν το μοναδικό δομικό και απαραίτητο στοιχείο για μπορέσουν να αναπαρασταθούν οι πληροφορίες που περιέχονται στη βάση δεδομένων.

Για να μπορέσουμε να προσθέσουμε, διαγράψουμε ή τροποποιήσουμε τα στοιχεία που περιέχονται σε μια βάση δεδομένων, χρησιμοποιούμε ειδικές γλώσσες προγραμματισμού που αποκαλούνται **γλώσσες ερωταπαντήσεων (query languages)**. Η γλώσσα που αποτελεί σήμερα ένα διεθνές πρότυπο για την επικοινωνία των χρηστών με τα Σχεσιακά ΣΔΒΔ είναι η **SQL (Structured Query Language) ή Δομημένη Γλώσσα Ερωτημάτων**.

Μπορεί να λειτουργήσει αυτόνομα αλλά και σε συνεργασία μ' άλλες γλώσσες προγραμματισμού.

Μια άλλη, φιλική προς τον χρήστη γλώσσα προγραμματισμού για να μπορούμε να υποβάλουμε ερωτήματα σε σχεσιακές βάσεις δεδομένων και να λαμβάνουμε απαντήσεις είναι η **QBE (Query By Example)**, η οποία χρησιμοποιεί φόρμες για τη γραφική απεικόνιση των ερωτημάτων μας.

Σήμερα, υπάρχουν εξελιγμένα εργαλεία διαχείρισης σε γραφικό και φιλικό προς τον χρήστη περιβάλλον για να κάνουμε τα εξής :

- Δημιουργία πινάκων
- Δημιουργία φορμών
- Δημιουργία ερωτημάτων
- Δημιουργία εκθέσεων (αναφορών)

Τα Σχεσιακά ΣΔΒΔ τα διακρίνουμε στα **μεγάλα**, τα οποία αφορούν κυρίως μεγάλους οργανισμούς και επιχειρήσεις, έχουν τεράστιο όγκο δεδομένων και πολλούς χρήστες ταυτόχρονα, και τέτοια συστήματα είναι τα Oracle, Ingres, Informix, SQL Server, MysqL, κ.ά. και τα **μικρά**, τα οποία αφορούν κυρίως απλούς χρήστες, όπως είναι η Microsoft Access, η Paradox, η FoxPro κ.ά.

Το Μοντέλο Οντοτήτων–Συσχετίσεων

Το μοντέλο που έχει επικρατήσει σήμερα για να παραστήσει τις έννοιες ή τη δομή μιας βάσης δεδομένων είναι το **Μοντέλο Οντοτήτων–Συσχετίσεων (ΟΣ)**. Οι βασικές (θεμελιώδεις) έννοιες του μοντέλου αυτού είναι οι εξής :

- Οντότητες
- Ιδιότητες ή Χαρακτηριστικά
- Συσχετίσεις

Για να αναπαραστήσουμε ένα Μοντέλο Οντοτήτων – Συσχετίσεων χρησιμοποιούμε ειδικά διαγράμματα, όπου τα ορθογώνια συμβολίζουν τις οντότητες, οι ρόμβοι τις συσχετίσεις και οι ελλείψεις τις ιδιότητες. Με ευθείες γραμμές συνδέουμε τις οντότητες που συσχετίζονται με κάποιο τρόπο μεταξύ τους. Όλα τα παραπάνω αποτελούν τη λογική δομή μιας βάσης δεδομένων, μια εργασία που είναι απαραίτητο να γίνει πριν από την καταχώριση και την επεξεργασία των στοιχείων (πληροφοριών) της βάσης δεδομένων.

Το μοντέλο οντοτήτων–συσχετίσεων αποτελεί μια γενική περιγραφή των γενικών στοιχείων που απαρτίζουν μια βάση δεδομένων και απεικονίζει την αντίληψη που έχουμε για τα δεδομένα (εννοιολογικό), χωρίς να υπεισέρχεται σε λεπτομέρειες υλοποίησης.

Οι Οντότητες

Με τον όρο **οντότητα (entity)** αναφερόμαστε σε κάθε αντικείμενο, έννοια, πρόσωπο ή κατάσταση που έχει μια ανεξάρτητη ύπαρξη. Είναι κάτι που ξεχωρίζει και μπορούμε να συγκεντρώσουμε πληροφορίες (στοιχεία) γι' αυτό. Η οντότητα είναι αντίστοιχη με την έννοια της εγγραφής που συναντάμε στα αρχεία και στους πίνακες αλλά και με την έννοια του αντικειμένου στις σύγχρονες αντικειμενοστραφείς γλώσσες προγραμματισμού.

Παραδείγματα οντοτήτων είναι τα εξής : ΠΕΛΑΤΗΣ, ΠΑΡΑΓΓΕΛΙΑ, ΠΡΟΜΗΘΕΥΤΗΣ, ΑΠΟΘΗΚΗ, ΜΑΘΗΤΗΣ, ΚΑΘΗΓΗΤΗΣ, ΑΘΛΗΤΗΣ, ΑΓΩΝΙΣΜΑ, ΧΩΡΑ, ΠΟΛΕΙΣ κ.ά.

Μια βάση δεδομένων μπορεί να περιέχει πολλές διαφορετικές οντότητες, οι οποίες απεικονίζονται με ορθογώνια παραλληλόγραμμα και συσχετίζονται μεταξύ τους ανά δύο.

Οι Ιδιότητες (Χαρακτηριστικά) των Οντοτήτων

Με τον όρο **ιδιότητες (properties**) ή **χαρακτηριστικά (attributes**) αναφερόμαστε στα συστατικά (δομικά) στοιχεία που προσδιορίζουν (αποτελούν) μια οντότητα. Η ιδιότητα είναι αντί`στοιχη με την έννοια του πεδίου που συναντάμε στα αρχεία και στους πίνακες αλλά και με την έννοια της μεταβλητής στις γλώσσες προγραμματισμού.

Για παράδειγμα, η οντότητα ΓΙΑΤΡΟΣ μπορεί να αποτελείται από τις ιδιότητες (χαρακτηριστικά) ΑριθμόςΜητρώου, Επώνυμο, Όνομα, Πατρώνυμο, Ειδικότητα, Βαθμός, ΈτοςΓέννησης, Διεύθυνση, ΑΦΜ, Τηλέφωνο, Κινητό κ.ά., ενώ η οντότητα ΑΘΛΗΤΗΣ μπορεί να αποτελείται από τις ιδιότητες (χαρακτηριστικά) ΚωδικόςΑθλητή, Επώνυμο, Όνομα, Πατρώνυμο, Αγώνισμα, Επίδοση, Σύλλογος, ΈτοςΓέννησης, Διεύθυνση, Τηλέφωνο, Κινητό κ.ά.

Απ' όλες τις ιδιότητες μιας οντότητας, υπάρχει μία μόνο ιδιότητα, και σπανιότερα ένας συνδυασμός δύο ή και περισσοτέρων ιδιοτήτων, η τιμή της οποίας είναι μοναδική και προσδιορίζει την κάθε εμφάνιση (στιγμιότυπο) της οντότητας και αποκαλείται **πρωτεύον κλειδί (primary key)**. Το πρωτεύον κλειδί εμφανίζεται στα διαγράμματα με υπογράμμιση ή με έντονη γραφή ή έχει ως πρόθεμα τον χαρακτήρα #.

Στο διάγραμμα οντοτήτων–συσχετίσεων οι ιδιότητες απεικονίζονται με σχήματα ελλειπτικής μορφής, τα οποία ενώνονται με ευθείες γραμμές με την οντότητα στην οποία ανήκουν.

Τα Κλειδιά

Όπως είδαμε και νωρίτερα, με τον όρο **κλειδί (key)** ή πιο σωστά **πρωτεύον κλειδί (primary key)** αναφερόμαστε σε μια ιδιότητα (πεδίο), ή σπανιότερα σ' ένα σύνολο ιδιοτήτων (πεδίων), η τιμή της οποίας είναι μοναδική σ' ολόκληρη την οντότητα (πίνακας). Στην πράξη, το πρωτεύον κλειδί έχει διαφορετική τιμή για κάθε εμφάνιση της οντότητας ή για κάθε γραμμή (εγγραφή) του πίνακα και ποτέ δεν μπορεί να έχει μηδενική (κενή) τιμή (null). Προσοχή, άλλο πράγμα είναι ο αριθμός 0 και άλλο πράγμα είναι η κενή τιμή (null), δηλ. η μη ύπαρξη τιμής.

Ο συνδυασμός δύο ή και περισσοτέρων ιδιοτήτων (πεδίων) για τη δημιουργία ενός πρωτεύοντος κλειδιού αποκαλείται **σύνθετο κλειδί**. Ένα παράδειγμα σύνθετου κλειδιού θα μπορούσε να είναι ο συνδυασμός των ιδιοτήτων Επώνυμο, Όνομα και Πατρώνυμο, εφόσον φυσικά είμαστε απολύτως βέβαιοι ότι δεν υπάρχουν δύο ή και περισσότερα άτομα με κοινές τιμές στις παραπάνω ιδιότητες.

Ξένο κλειδί αποκαλείται μια ιδιότητα (πεδίο) που είναι πρωτεύον κλειδί σε μια οντότητα (πίνακας) αλλά που υπάρχει και σε μια άλλη οντότητα (πίνακας) σαν απλή ιδιότητα. Τα ξένα κλειδιά είναι απαραίτητα για να μπορέσουμε να κάνουμε τις συσχετίσεις (συνδέσεις, επικοινωνίες) ανάμεσα στις οντότητες (πίνακες).

Για παράδειγμα, στην οντότητα ΣΥΛΛΟΓΟΣ, το πεδίο ΚωδικόςΣυλλόγου είναι πρωτεύον κλειδί, ενώ στην οντότητα ΑΘΛΗΤΗΣ, το πεδίο ΚωδικόςΣυλλόγου είναι ξένο κλειδί και είναι απαραίτητο για να μπορέσουμε να υλοποιήσουμε τη συσχέτιση ΑΝΗΚΕΙ, δηλ. να αντλήσουμε την πληροφορία ποιοι αθλητές ανήκουν σε ποιους συλλόγους. Προφανώς, στην οντότητα ΣΥΛΛΟΓΟΣ, το πεδίο Κωδικός Συλλόγου θα έχει μοναδικές τιμές, ενώ στην οντότητα ΑΘΛΗΤΗΣ, το πεδίο Κωδικός Συλλόγου θα έχει επαναλαμβανόμενες τιμές και αυτό γιατί πολλοί αθλητές θα ανήκουν στον ίδιο σύλλογο, αλλά ένας αθλητής ανήκει υποχρεωτικά σ' έναν και μόνο έναν σύλλογο.

Σ' ένα άλλο παράδειγμα, στην οντότητα ΓΙΑΤΡΟΣ, το πεδίο ΚωδικόςΓιατρού είναι πρωτεύον κλειδί, ενώ στην οντότητα ΑΣΘΕΝΗΣ, το πεδίο ΚωδικόςΓιατρού είναι ξένο κλειδί και είναι απαραίτητο για να μπορέσουμε να υλοποιήσουμε τη συσχέτιση ΠΑΡΑΚΟΛΟΥΘΕΙΤΑΙ, δηλ. να αντλήσουμε την πληροφορία ποιοι ασθενείς

παρακολουθούνται από ποιους γιατρούς. Προφανώς, στην οντότητα ΓΙΑΤΡΟΣ, το πεδίο ΚωδικόςΓιατρού θα έχει μοναδικές τιμές, ενώ στην οντότητα ΑΣΘΕΝΗΣ, το πεδίο ΚωδικόςΓιατρού θα έχει επαναλαμβανόμενες τιμές και αυτό γιατί πολλοί ασθενείς θα παρακολουθούνται από τον ίδιο γιατρό, αλλά ένας ασθενής παρακολουθείται μόνο από έναν γιατρό.

Αυτό αποτελεί βέβαια μια παραδοχή που κάνουμε για να μπορέσουμε να υλοποιήσουμε μια συσχέτιση σαν την παραπάνω σε μια βάση δεδομένων ενός Νοσοκομείου, αλλά μπορεί να θεωρήσει κάποιος ότι ένας ασθενής μπορεί να παρακολουθείται από πολλούς γιατρούς ταυτόχρονα, οπότε θα πρέπει να μεταβάλλουμε και τον τρόπο συσχέτισης των παραπάνω οντοτήτων.

Οι Συσχετίσεις Μεταξύ Οντοτήτων

Ο σωστός σχεδιασμός και προσδιορισμός των οντοτήτων και των ιδιοτήτων τους αποτελούν τα θεμελιώδη βήματα για τη σωστή σχεδίαση και υλοποίηση μιας βάσης δεδομένων. Μια συσχέτιση συνδέει δύο ή και περισσότερες οντότητες μεταξύ τους και παριστάνεται στο διάγραμμα οντοτήτων–συσχετίσεων μ' έναν ρόμβο.

Οι συσχετίσεις είναι απαραίτητες για να μπορέσουμε να αντλήσουμε πληροφορίες που αφορούν δύο ή και περισσότερες οντότητες, όπως για παράδειγμα ποιοι πελάτες έκαναν παραγγελίες κάποια συγκεκριμένη χρονική περίοδο (συσχέτιση ΠΑΡΑΓΓΕΛΝΕΙ) ή ποιοι αθλητές ανήκουν σε ποιους συλλόγους (συσχέτιση ΑΝΗΚΕΙ) ή ποιοι αθλητές έλαβαν μέρος σε αγωνίσματα μια συγκεκριμένη χρονιά (συσχέτιση ΣΥΜΜΕΤΕΧΕΙ) κοκ.

Όταν οι οντότητες που συμμετέχουν σε μια συσχέτιση είναι δύο, η συσχέτιση αποκαλείται **διμελής** ή **δυαδική**. Ο βαθμός μιας συσχέτισης είναι ίσος με το πλήθος των οντοτήτων που συμμετέχουν σ' αυτήν. Μια συσχέτιση μπορεί και η ίδια να έχει ιδιότητες που να περιγράφουν ορισμένα χαρακτηριστικά της, όπως για παράδειγμα η συσχέτιση ΠΑΡΑΓΓΕΛΙΑ ανάμεσα στις οντότητες ΠΕΛΑΤΗΣ και ΠΡΟΪΟΝ μπορεί να περιέχει τις ιδιότητες (πεδία) ΚωδικόςΠελάτη, ΚωδικόςΠροϊόντος, ΚωδικόςΠαραγγελίας, ΗμερομηνίαΠαραγγελίας, Ποσότητα κ.ά.

Στην περίπτωση αυτή το σωστό είναι να δημιουργήσουμε μια ακόμα οντότητα, την οντότητα ΠΑΡΑΓΓΕΛΙΑ, η οποία και θα περιέχει όλες τις παραπάνω ιδιότητες, και να μετονομάσουμε την προηγούμενη συσχέτιση από ΠΑΡΑΓΓΕΛΙΑ σε ΣΥΝΑΛΛΑΓΗ, που δεν θα περιέχει τώρα ιδιότητες. Έτσι, η παραπάνω συσχέτιση θα μετατραπεί από διμελή σε τριμελή.

Όταν σχεδιάζουμε μια βάση δεδομένων, θα πρέπει να εκχωρούμε ιδιότητες μόνο στις οντότητες και να έχουμε τις συσχετίσεις απλά και μόνο για να κατανοούμε τις λογικές συνδέσεις ανάμεσα στις οντότητες.

Οι Διμελείς Συσχετίσεις

Οι διμελείς συσχετίσεις μεταξύ οντοτήτων είναι αυτές που θα μας απασχολήσουν ιδιαίτερα και υπάρχουν τρία βασικά είδη συνδέσεων σ' αυτές, τα εξής :

- **Ένα-προς-ένα (1:1)**, όπου μια εμφάνιση της μιας οντότητας συνδέεται με μία και μόνο μία εμφάνιση της άλλης οντότητας. Για παράδειγμα, η οντότητα ΣΥΛΛΟΓΟΣ έχει έναν μόνο προπονητή, ενώ η οντότητα ΠΡΟΠΟΝΗΤΗΣ συνδέεται μ' έναν και μόνο έναν σύλλογο. Σ' ένα άλλο παράδειγμα, η οντότητα ΝΟΜΟΣ έχει μία μόνο πόλη σαν πρωτεύουσα, ενώ η οντότητα ΠΡΩΤΕΥΟΥΣΑ αντιστοιχεί σ' έναν και μόνο έναν νομό. Στην περίπτωση των διμελών συσχετίσεων του τύπου ένα-προς-ένα, μπορούμε να ενώσουμε τα στοιχεία και των δύο ιδιοτήτων και να δημιουργήσουμε μια μοναδική οντότητα (πίνακα).

- **Ένα-προς-πολλά (1:Μ)**, όπου μια εμφάνιση της μιας οντότητας συνδέεται με πολλές εμφανίσεις της άλλης οντότητας αλλά κάθε εμφάνιση της δεύτερης οντότητας συνδέεται με μία και μόνο μία εμφάνιση της πρώτης οντότητας. Για παράδειγμα, ένας ΠΕΛΑΤΗΣ κάνει πολλές παραγγελίες, αλλά μια ΠΑΡΑΓΓΕΛΙΑ αντιστοιχεί σ' έναν και μόνο έναν πελάτη. Σ' ένα άλλο παράδειγμα, ένας ΣΥΛΛΟΓΟΣ έχει πολλούς αθλητές, αλλά ένας ΑΘΛΗΤΗΣ ανήκει σ' έναν και μόνο έναν σύλλογο. Οι διμελείς συσχετίσεις του τύπου ένα-προς-ένα είναι οι πιο συχνά συναντώμενες και οι πιο βολικές στη διαχείριση.
- **Πολλά-προς-πολλά (Μ:Ν)**, όπου σε μια εμφάνιση της μιας οντότητας αντιστοιχούν πολλές εμφανίσεις της άλλης οντότητας και σε κάθε εμφάνιση της δεύτερης οντότητας αντιστοιχούν πολλές εμφανίσεις της πρώτης οντότητας. Για παράδειγμα, ένας ΑΘΛΗΤΗΣ συμμετέχει σε πολλούς αγώνες αλλά και σ' έναν ΑΓΩΝΑ λαμβάνουν μέρος πολλοί αθλητές. Σ' ένα άλλο παράδειγμα, ένας ΚΑΘΗΓΗΤΗΣ διδάσκει σε πολλούς μαθητές αλλά και ένας ΜΑΘΗΤΗΣ διδάσκεται από πολλούς καθηγητές. Για να μπορέσουμε να διαχειριστούμε μια διμελή σχέση του τύπου πολλά-προς-πολλά, θα πρέπει να δημιουργήσουμε έναν τρίτο πίνακα που θα περιέχει δύο μόνο ιδιότητες (πεδία), δηλ. τα πεδία κλειδιά των δύο οντοτήτων, οπότε ο συνδυασμός τους θα είναι και το πεδίο κλειδί (σύνθετο κλειδί) του νέου πίνακα.

Το Διάγραμμα Οντοτήτων Συσχετίσεων

Για να μπορέσουμε να διαμορφώσουμε το διάγραμμα οντοτήτων συσχετίσεων, θα πρέπει να ακολουθήσουμε τα εξής βήματα :

- Να ορίσουμε τις οντότητες (πίνακες) που θα ανήκουν στη βάση δεδομένων που θέλουμε να δημιουργήσουμε.
- Να ορίσουμε τις ιδιότητες (πεδία) και τα πρωτεύοντα κλειδιά της κάθε οντότητας (πίνακα).
- Να ορίσουμε τις συσχετίσεις ανάμεσα στις οντότητες.
- Δημιουργούμε το διάγραμμα οντοτήτων συσχετίσεων, όπου θα απεικονίσουμε τις οντότητες, τις ιδιότητές τους και τις συσχετίσεις τους.

Θα δούμε το διάγραμμα οντοτήτων συσχετίσεων για μια βάση δεδομένων με ομάδες (συλλόγους) ποδοσφαίρου, όπου θα έχουμε τις οντότητες ΑΘΛΗΤΗΣ, ΣΥΛΛΟΓΟΣ, ΠΡΟΠΟΝΗΤΗΣ και ΑΓΩΝΑΣ. Οι συσχετίσεις ανάμεσα στις οντότητες αυτές θα είναι οι εξής :

ΑΘΛΗΤΗΣ – ΣΥΛΛΟΓΟΣ : ένα-προς-πολλά (1:Μ)
ΣΥΛΛΟΓΟΣ – ΠΡΟΠΟΝΗΤΗΣ : ένα-προς-ένα (1:1)
ΑΘΛΗΤΗΣ – ΑΓΩΝΑΣ : πολλά-προς-πολλά (Μ:Ν)
11ΜΝΣΥΜΜΕΤΕΧΕΙ ΑΝΗΚΕΙ 1Μ

ΑΘΛΗΤΗΣ
Επώνυμο, Όνομα, Απολαβές, Διακρίσεις, Ημερομηνία Γέννησης, Κωδικός Αθλητή,

ΑΓΩΝΑΣ
Κωδικός Αγώνα, Ημερομηνία Τέλεσης, Αποτέλεσμα, Κωδικός Διαιτητή, Κωδικός Προπονητή, Κωδικός Συλλόγου1, Κωδικός Συλλόγου2 ,

ΠΡΟΠΟΝΗΤΗΣ
Επώνυμο, Όνομα, Απολαβές, Διακρίσεις, Κωδικός Προπονητή

ΣΥΛΛΟΓΟΣ
Επώνυμο, Έδρα, Γήπεδο, Κωδικός Συλλόγου

Pictures

Λογικός Σχεδιασμός μιας Βάσης Δεδομένων

Αφού έχουμε δημιουργήσει το διάγραμμα οντοτήτων συσχετίσεων και έχουμε επιλέξει το σχεσιακό μοντέλο δεδομένων για την υλοποίηση της βάσης δεδομένων, ακολουθούμε τη διαδικασία της κανονικοποίησης και είμαστε έτοιμοι για την καταχώριση των στοιχείων της βάσης δεδομένων. Ανάλογα τώρα με το είδος της διμελούς συσχέτισης, διακρίνουμε τις εξής περιπτώσεις ως προς τον λογικό σχεδιασμό που θα πρέπει να ακολουθήσουμε :

Αν η συσχέτιση των δύο πινάκων είναι ένα-προς-ένα, τότε μπορούμε είτε να συνενώσουμε τους δύο πίνακες, με τις αντίστοιχες εγγραφές φυσικά, ή να προσθέσουμε το ένα από τα δύο πεδία κλειδιά σαν ξένο κλειδί στον άλλον πίνακα ή τέλος να δημιουργήσουμε έναν καινούργιο πίνακα με μόνα πεδία τα πεδία κλειδιά των δύο πινάκων (σύνθετο κλειδί). Η προτιμότερη διαδικασία είναι η πρώτη, δηλ. η συνένωση των δύο πινάκων σ' έναν ενιαίο πίνακα.

Αν η συσχέτιση των δύο πινάκων είναι ένα-προς-πολλά, τότε μπορούμε είτε να προσθέσουμε το ένα από τα δύο πεδία κλειδιά σαν ξένο κλειδί στον άλλον πίνακα ή να δημιουργήσουμε έναν καινούργιο πίνακα με μόνα πεδία τα πεδία κλειδιά των δύο πινάκων (σύνθετο κλειδί). Η προτιμότερη διαδικασία είναι η πρώτη, δηλ. η προσθήκη του ξένου κλειδιού στην πλευρά 'πολλά' της σχέσης.

Αν η συσχέτιση των δύο πινάκων είναι πολλά-προς-πολλά, τότε το μόνο που μπορούμε και πρέπει να κάνουμε είναι να δημιουργήσουμε έναν καινούργιο πίνακα με μόνα πεδία τα πεδία κλειδιά των δύο πινάκων (σύνθετο κλειδί), όπου το κάθε πεδίο κλειδί από μόνο του γίνεται ξένο κλειδί. Οι δύο αρχικοί πίνακες δεν μεταβάλλονται.

Η Κανονικοποίηση

Τα προβλήματα που είναι πιθανό να παρουσιασθούν κατά τη διαδικασία της υλοποίησης του σχεδιασμού μιας βάσης δεδομένων είναι η περιττή (άσκοπη) επανάληψη πληροφοριών, που είναι γνωστή με τον όρο redundancy, καθώς και δυσκολίες στην ενημέρωση της βάσης δεδομένων. Τα παραπάνω προβλήματα είναι γνωστά ως **πλεονασμοί δεδομένων** και **ανωμαλίες ενημέρωσης** και για να αντιμετωπιστούν με επιτυχία, θα πρέπει να διασπάσουμε τις μεγάλες σχέσεις σε μικρότερες. Αυτό γίνεται με τη διαδικασία της κανονικοποίησης, έτσι ώστε η βάση δεδομένων να είναι έτοιμη για καταχώριση στοιχείων.

Η **κανονικοποίηση (normalization)** είναι μια τεχνική που ασχολείται με την ανάλυση των σχέσεων (συσχετίσεων) σε μια βάση δεδομένων, όπου κάνουμε μετατροπή των αρχικών μεγάλων σχέσεων σε μικρότερες.

Πλεονασμός Δεδομένων και Ανωμαλίες Ενημέρωσης

Με τον όρο πλεονασμός δεδομένων (data redundancy) εννοούμε την άσκοπη επανάληψη στοιχείων (πληροφοριών). Τα προβλήματα που προκύπτουν από τον πλεονασμό δεδομένων είναι γνωστά με τον όρο ανωμαλίες ενημέρωσης (update anomalies). Για να μπορέσουμε να κατανοήσουμε τις παγίδες του πλεονασμού δεδομένων, θα δούμε ένα παράδειγμα με τους πίνακες ΠΕΛΑΤΗΣ και ΠΑΡΑΓΓΕΛΙΑ. Η σχέση μεταξύ τους είναι ένα-προς-πολλά, δηλ. ένας πελάτης μπορεί να κάνει πολλές παραγγελίες, αλλά μια παραγγελία γίνεται μόνο από έναν πελάτη.

Αν αποφασίσουμε να δημιουργήσουμε έναν μόνον πίνακα, όπου σε κάθε γραμμή (εγγραφή) του θα υπάρχουν όλα τα στοιχεία της παραγγελίας και δίπλα όλα τα στοιχεία του πελάτη που έχει κάνει την παραγγελία, τότε θα έχουμε πέσει στην παγίδα του πλεονασμού δεδομένων και αυτό γιατί τα στοιχεία του κάθε πελάτη θα επαναλαμβάνονται για κάθε παραγγελία που έχει κάνει.

Ανωμαλία εισαγωγής έχουμε στην περίπτωση που θελήσουμε να καταχωρήσουμε μια καινούργια παραγγελία, οπότε θα πρέπει να καταχωρήσουμε εκ νέου κι όλα τα στοιχεία του

πελάτη που έκανε τη συγκεκριμένη παραγγελία, κάτι που είναι κουραστικό, χρονοβόρο και περιέχει τον κίνδυνο λαθών.

Ένα άλλο πρόβλημα με ανωμαλία εισαγωγής έχουμε στην περίπτωση που θελήσουμε να καταχωρήσουμε ένα καινούργιο πελάτη ο οποίος δεν έχει κάνει ακόμα καμία παραγγελία, οπότε η βάση δεδομένων δεν θα μας το επιτρέψει και αυτό γιατί δεν δέχεται κενή τιμή (null) για το πεδίο κλειδί Κωδικός Παραγγελίας.

Ανωμαλία διαγραφής έχουμε στην περίπτωση που θελήσουμε να διαγράψουμε μια παραγγελία που είναι η μοναδική ενός πελάτη, οπότε θα χάσουμε και όλες τις πληροφορίες του συγκεκριμένου πελάτη.

Ανωμαλία τροποποίησης έχουμε στην περίπτωση που θελήσουμε να αλλάξουμε κάποιο στοιχείο ενός πελάτη, όπως τη διεύθυνση ή το τηλέφωνό του, οπότε θα πρέπει να τροποποιήσουμε όλες τις εγγραφές του πίνακα όπου εμφανίζεται ο συγκεκριμένος πελάτης. Αν δεν κάνουμε την αλλαγή σ' όλες τις εγγραφές, τότε ο πελάτης θα εμφανίζεται να έχει δύο διευθύνσεις ή δύο τηλέφωνα κοκ. Μιλάμε τότε για μη συνεπή (inconsistent) βάση δεδομένων.

Ως γνωστόν, η λύση στο παραπάνω πρόβλημα είναι να δημιουργήσουμε έναν πίνακα με τα στοιχεία των πελατών και έναν ξεχωριστό πίνακα με τα στοιχεία των παραγγελιών, όπου θα υπάρχει και σαν πεδίο (ξένο κλειδί) ο ΚωδικόςΠελάτη.

Η Διαδικασία της Κανονικοποίησης

Η μέθοδος της κανονικοποίησης βοηθάει στον λογικό σχεδιασμό μιας βάσης δεδομένων και είναι συμπληρωματική του μοντέλου οντοτήτων συσχετίσεων. Το κέρδος για μας είναι ότι δεν υπάρχουν προβλήματα συνέπειας, πλεονασμού και εγκυρότητας των πληροφοριών της βάσης δεδομένων. Ακολουθώντας τη διαδικασία της κανονικοποίησης κάνουμε συνεχείς διασπάσεις των πινάκων σε πιο απλές και συμπαγείς μορφές, με στόχο πάντα να αποφύγουμε τον πλεονασμό (επανάληψη) των δεδομένων. Αφαιρούμε πεδία από τις αρχικές μεγάλες σχέσεις και τα τοποθετούμε σε νέες σχέσεις έτσι ώστε να μπορούμε να έχουμε τις ίδιες πληροφορίες και με τις νέες σχέσεις.

Μπορούμε να πούμε ότι κανονικοποίηση (normalization) είναι η διαδικασία μετατροπής των δεδομένων κάποιων σχέσεων (πινάκων) σε πιο απλές και πιο σαφείς σχέσεις, χωρίς πλεονασμούς (επαναλήψεις) των δεδομένων. Οι βασικές μορφές της κανονικοποίησης είναι τρεις, η πρώτη (1η NF), η δεύτερη (2η NF) και η τρίτη (3η NF). Θα ξεκινήσουμε τη μελέτη μας με μια βάση δεδομένων που θέλουμε να κατασκευάσουμε για τους αθλητές στο αγώνισμα των 100 μέτρων στίβου, οι οποίοι προέρχονται από διάφορους συλλόγους διαφόρων χωρών και όπου μας ενδιαφέρουν οι επιδόσεις τους σε διάφορους διεθνείς αγώνες.

Αρχικά, χωρίς καμία μελέτη, θα μπορούσε κάποιος να θεωρήσει ότι τα δεδομένα για τους αθλητές και τις επιδόσεις τους στους αγώνες θα ήταν κάπως έτσι :

Κωδικός_Αθλητή	**Επώνυμο_Όνομα**	**Κωδικός_Συλλόγου**	**Ονομασία_Συλλόγου**	**Κωδικός_Αγώνα**	**Αγώνας**	**Επίδοση**	**Κωδικός_Χώρας**	**Ονομασία_Χώρας**
100	Lewis Carl	200	America	01 02 03	LA 2004 MO 2003 RO 2002	10.08 10.04 10.07	300	USA

101	Wells John	201	Rangers	01 03 04	LA 2004 RO 2002 GE 2001	10.04 10.05 10.03	301	Britain

Παρατηρούμε ότι τα πεδία **Κωδικός_Αγώνα, Αγώνας** και **Επίδοση** έχουν περισσότερες από μία τιμές. Αυτό είναι αντίθετο με τις αρχές της σχεσιακής θεωρίας, γιατί κάθε σχέση του σχεσιακού μοντέλου θα πρέπει να έχει πεδία με μία και μοναδική τιμή σε κάθε σειρά (εγγραφή).

Δεν μπορούμε φυσικά να προσθέσουμε κι άλλα πεδία, δηλ. ένα πεδίο για κάθε αγώνα, γιατί στο σχεσιακό μοντέλο δεν μπορεί μια σχέση να έχει μεταβαλλόμενο αριθμό πεδίων και ούτε είμαστε σε θέση να γνωρίζουμε σε πόσους αγώνες πήρε μέρος ένας αθλητής.

Μια πρώτη λύση θα ήταν να μην έχουμε επαναλαμβανόμενες τιμές στην ίδια σειρά και να προσπαθήσουμε να δημιουργήσουμε έτσι έναν νέο πίνακα σε κάθε κελί του οποίου να περιέχεται μία μόνο τιμή, όπως φαίνεται στο παρακάτω σχήμα :

Κωδικός_Αθλητή	**Επώνυμο_Όνομα**	**Κωδικός_Συλλόγου**	**Ονομασία_Συλλόγου**	**Κωδικός_Αγώνα**	**Αγώνας**	**Επίδοση**	**Κωδικός_Χώρας**	**Ονομασία_Χώρας**
100	Lewis Carl	200	America	01	LA 2004	10.08	300	USA
100	Lewis Carl	200	America	02	MO 2003	10.04	300	USA
100	Lewis Carl	200	America	03	RO 2002	10.07	300	USA
101	Wells John	201	Rangers	01	LA 2004	10.04	301	Britain
101	Wells John	201	Rangers	03	RO 2002	10.05	301	Britain
101	Wells John	201	Rangers	04	GE 2001	10.03	301	Britain

Η σχέση ή ο πίνακας που επεξεργαζόμαστε βρίσκεται τώρα στην 1η κανονική μορφή (1η NF). Ο ορισμός λέει ότι μια σχέση (πίνακας) βρίσκεται στην **1η κανονική μορφή** όταν περιέχει σταθερό και όχι μεταβλητό αριθμό πεδίων (στηλών) και κάθε πεδίο της σχέσης δεν περιέχει επαναλαμβανόμενες τιμές. Επίσης, κάθε κελί της σχέσης (διασταύρωση γραμμής και στήλης) θα πρέπει να περιέχει μία μόνο τιμή. Τα μειονεκτήματα που βλέπουμε αμέσως ότι προκύπτουν από τη νέα μορφή που πήρε ο πίνακας είναι ότι έχουμε τώρα περισσότερες γραμμές για να απεικονίσουμε τα ίδια ακριβώς δεδομένα και φυσικά έχουμε περιττή επανάληψη τιμών.

Αν είμαστε στην 1η κανονική μορφή, για να μπορέσουμε να προχωρήσουμε στην 2η και στην 3η κανονική μορφή, θα πρέπει να ορίσουμε πρώτα ένα πρωτεύον κλειδί, δηλ. ένα πεδίο ή έναν συνδυασμό από δύο ή περισσότερα πεδία (σύνθετο κλειδί) για να μπορούμε να προσδιορίζουμε μονοσήμαντα την κάθε γραμμή (εγγραφή ή και πλειάδα).

Στο παραπάνω παράδειγμα, παρατηρούμε ότι ο πιο κατάλληλος συνδυασμός πεδίων για να προσδιορίσει μονοσήμαντα την κάθε γραμμή είναι τα πεδία **Κωδικός_Αθλητή** και **Κωδικός_Αγώνα**. Τώρα, αν το κλειδί που έχουμε ορίσει είναι σύνθετο, δηλ. αποτελείται από δύο ή περισσότερα πεδία, θα πρέπει να συνεχίσουμε με την 2η κανονική μορφή (2η NF), αλλιώς θα πρέπει να συνεχίσουμε με την 3η κανονική μορφή (3η NF).

Συνεχίζοντας τώρα με την 2η κανονική μορφή, ψάχνουμε να βρούμε τα πεδία εκείνα που να συσχετίζονται με (αφορούν, εξαρτώνται από) ολόκληρο το σύνθετο κλειδί. Παίρνουμε τα πεδία που συγκροτούν το σύνθετο κλειδί και από τα πεδία αυτά δημιουργούμε έναν καινούργιο πίνακα.

Στον πίνακά μας, το μόνο πεδίο που έχει σχέση με τον συνδυασμό των πεδίων που συγκροτούν το σύνθετο κλειδί, δηλ. με τα πεδία **Κωδικός_Αθλητή** και **Κωδικός_Αγώνα**, είναι προφανώς το πεδίο Επίδοση. Δημιουργούμε τώρα τον παρακάτω πίνακα.

Κωδικός_Αθλητή	**Κωδικός_Αγώνα**	**Επίδοση**
100	01	10.08
100	02	10.04
100	03	10.07
101	01	10.04
101	03	10.05
101	04	10.03

Συνεχίζοντας με την 2η κανονική μορφή, προσπαθούμε τώρα να βρούμε ποια από τα υπόλοιπα πεδία του πίνακα εξαρτώνται από κάθε ξεχωριστό πεδίο του σύνθετου κλειδιού. Παίρνουμε αυτό το ξεχωριστό πεδίο ως πρωτεύον κλειδί και με τα πεδία που έχουν σχέση μ' αυτό δημιουργούμε και από έναν καινούργιο πίνακα κάθε φορά.

Στο παράδειγμά μας, θα προκύψουν οι εξής δύο καινούργιοι πίνακες :

Κωδικός_Αθλητή	**Επώνυμο_Όνομα**	**Κωδικός_Συλλόγου**	**Ονομασία_Συλλόγου**	**Κωδικός_Χώρας**	**Ονομασία_Χώρας**
100	Lewis Carl	200	America	300	USA
100	Lewis Carl	200	America	300	USA
100	Lewis Carl	200	America	300	USA
101	Wells John	201	Rangers	301	Britain
101	Wells John	201	Rangers	301	Britain
101	Wells John	201	Rangers	301	Britain

Κωδικός_Αγώνα	**Αγώνας**
01	LA 2004
02	MO 2003
03	RO 2002
01	LA 2004
03	RO 2002
04	GE 2001

Φυσικά, στον δεύτερο πίνακα που έχει σχέση με τους Αγώνες, θα μπορούσαμε να είχαμε συμπεριλάβει και πεδία που να αφορούν έναν συγκεκριμένο αγώνα, όπως Πόλη, Χώρα, Ημερομηνία, Διεθνής ή Φιλικός κ.ά. Βλέπουμε ότι έχουμε ήδη εξαλείψει ένα μεγάλο μέρος του πλεονασμού δεδομένων που είχαμε στον αρχικό πίνακα. Η σχέση που επεξεργαζόμαστε βρίσκεται τώρα στην **2η κανονική μορφή** (2η NF). Ο ορισμός λέει ότι μια σχέση βρίσκεται στην 2η κανονική μορφή όταν έχει προέλθει από σχέση της 1ης κανονικής μορφής και ακόμη τα πεδία που δεν ανήκουν στο κλειδί έχουν σχέση μόνο με το κλειδί.

Βλέπουμε, όμως, ότι υπάρχουν ακόμη πλεονασμοί δεδομένων, όπως συμβαίνει με τα πεδία **Ονομασία_Συλλόγου** και **Ονομασία_Χώρας** στον πίνακα που έχει ως πεδίο κλειδί τον **Κωδικό_Αθλητή**. Θα πρέπει συνεπώς να προχωρήσουμε σε δύο ακόμη διασπάσεις του πίνακα αυτού για να αποφύγουμε αυτές τις επαναλήψεις τιμών.

Κωδικός_Αθλητή	**Επώνυμο_Όνομα**	**Κωδικός_Συλλόγου**
100	Lewis Carl	200
100	Lewis Carl	200
100	Lewis Carl	200
101	Wells John	201
101	Wells John	201
101	Wells John	201

Κωδικός_Συλλόγου	**Ονομασία_Συλλόγου**	**Κωδικός_Χώρας**
200	America	300
200	America	300
200	America	300
201	Rangers	301
201	Rangers	301
201	Rangers	301

Κωδικός_Χώρας	**Ονομασία_Χώρας**
300	USA
300	USA
300	USA
301	Britain
301	Britain
301	Britain

Έχουμε φθάσει σ' ένα σημείο που δεν χρειάζεται περαιτέρω διάσπαση των πινάκων καθώς στους πίνακες που έχουμε καταλήξει δεν υπάρχουν πεδία που να περιγράφουν κάτι που να έχει σχέση με κάποιο άλλο πεδίο εκτός από το πεδίο κλειδί. Η σχέση που επεξεργαζόμαστε βρίσκεται τώρα στην 3η κανονική μορφή (3η NF). Ο ορισμός λέει ότι μια σχέση βρίσκεται στην 3η κανονική μορφή όταν ικανοποιεί τις απαιτήσεις της 1ης και της 2ης κανονικής μορφής και ακόμη δεν υπάρχει κάποιο πεδίο στον πίνακα που να εξαρτάται από κάποιο άλλο πεδίο διαφορετικό του πρωτεύοντος κλειδιού.

Εισαγωγή στις Βάσεις Δεδομένων (DataBases)

Παράδειγμα 1 - Τι Είναι οι Βάσεις Δεδομένων (DataBases)

Μια Βάση Δεδομένων (DataBase) είναι ένας οργανωμένος τρόπος αποθήκευσης πληροφοριών και πρόσβασής τους με πολλούς τρόπους με διάφορα προγράμματα. Μια βάση δεδομένων είναι κάτι παραπάνω από μια απλή συλλογή αποθηκευμένων στοιχείων.

Ενας άλλος ορισμός είναι ότι μια βάση δεδομένων είναι ένα ολοκληρωμένο σύστημα που αποτελείται από δεδομένα (data) και από το κατάλληλο λογισμικό (software), τα οποία χρησιμοποιώντας το υλικό (hardware) βοηθούν στην ενημέρωση και πληροφόρηση των χρηστών (users).

Ενα πρόγραμμα που διαχειρίζεται βάσεις δεδομένων αποκαλείται Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS, DataBase Management System) και με την βοήθειά του μπορούμε να αποθηκεύσουμε, προσθέσουμε, τροποποιήσουμε, εμφανίσουμε ή και διαγράψουμε τα αποθηκευμένα δεδομένα.

Τα δεδομένα που υπάρχουν στις βάσεις δεδομένων πρέπει να είναι :

- **Ολοκληρωμένα (Integrated)**, δηλ. τα δεδομένα πρέπει να είναι αποθηκευμένα σε ομοιόμορφα οργανωμένα σύνολα αρχείων όπου δεν πρέπει να υπάρχει επανάληψη ή πλεονασμός (redundancy) των ίδιων στοιχείων.
- **Καταμεριζόμενα (Shared)**, δηλ. να μπορούν περισσότεροι του ενός χρήστες να βλέπουν και να μοιράζονται τα ίδια δεδομένα την ίδια χρονική στιγμή.

Παράδειγμα 2 - Τι Είναι το DBMS

Το Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS, DataBase Management System) είναι ένα σύνολο από προγράμματα που επιτρέπουν τον χειρισμό των δεδομένων μιας ή περισσοτέρων βάσεων δεδομένων που ανήκουν στο ίδιο σύστημα. Το DBMS περιέχει κάποια εργαλεία γενικής χρήσης για να μπορούμε να δημιουργούμε και να χειριζόμαστε τα δεδομένα.

Στα νεώτερα DBMS, όπως είναι η Oracle και η Informix, μπορούμε να έχουμε άμεση πληροφόρηση χωρίς να απαιτείται η παρουσία ενός προγραμματιστή. Τα δεδομένα ενός DBMS μπορούν να χρησιμοποιηθούν σε κάθε μορφής ερώτημα (query) για να πάρουμε ό,τι πληροφορίες θέλουμε.

Παράδειγμα 3 - Οι Στόχοι μιας Βάσης Δεδομένων

Οι στόχοι μιας βάσης δεδομένων είναι οι εξής :

- Ο περιορισμός της πολλαπλής αποθήκευσης των ίδιων στοιχείων (redundancy).
- Ο καταμερισμός (sharing) των ίδιων στοιχείων σ' όλους τους χρήστες.
- Η ομοιομορφία (uniformity) στον χειρισμό και την αναπαράσταση των δεδομένων.
- Η επιβολή κανόνων ασφαλείας (security).
- Η διατήρηση της ακεραιότητας (integrity) και της αξιοπιστίας (reliability) των δεδομένων.
- Η ανεξαρτησία των δεδομένων (data independence) και των προγραμμάτων από τον φυσικό τρόπο αποθήκευσης των δεδομένων.

Παράδειγμα 4 - Τα Στοιχεία μιας Βάσης Δεδομένων

Τα δεδομένα μιας βάσης δεδομένων αποθηκεύονται (οργανώνονται) στις εξής στοιχειώδεις μορφές :

- **Πεδίο (Field)**, είναι το μικρότερο κομμάτι δεδομένων στο οποίο μπορούμε να αναφερθούμε και περιέχει ένα μόνο χαρακτηριστικό ή ιδιότητα ενός στοιχείου της βάσης δεδομένων.
- **Εγγραφή (Record)**, είναι ένα σύνολο από διαφορετικά πεδία που περιέχει όλες τις πληροφορίες για ένα στοιχείο της βάσης δεδομένων.
- **Αρχείο (File)**, είναι ένα σύνολο από πολλά παρόμοια στοιχεία (εγγραφές) της βάσης δεδομένων.
- **Πρωτεύον Κλειδί (Primary Key)**, είναι ένα πεδίο ή συνδυασμός πεδίων που χαρακτηρίζει μοναδικά μια εγγραφή.
- **Κλειδί (Key)**, είναι ένα πεδίο που δεν έχει κατ' ανάγκη μοναδική τιμή και που μπορούμε να το χρησιμοποιήσουμε για να κάνουμε αναζήτηση σ' ένα αρχείο.
- **Ξένο Κλειδί (Foreign Key)**, είναι ένα πεδίο που έχει το ίδιο σύνολο τιμών με το πρωτεύον κλειδί ενός άλλου αρχείου.

Παράδειγμα 5 - Τα Εργαλεία Χειρισμού Πληροφοριών

Τα εργαλεία χειρισμού πληροφοριών μιας βάσης δεδομένων είναι γνωστά και σαν "Γλώσσες Εντολών" και με τη βοήθειά τους μπορούμε να δώσουμε εντολές χειρισμού των δεδομένων. Η πιο γνωστή και ευρέως διαδεδομένη γλώσσα εντολών για τις σύγχρονες βάσεις δεδομένων είναι η Δομημένη Γλώσσα Ερωτήσεων SQL (Structured Query Language), η οποία αποτελείται από τα εξής μέρη :

- **DDL** (Data Definition Language, Γλώσσα Ορισμού Δεδομένων), με την οποία καθορίζουμε τις δομές και τα τμήματα μιας βάσης δεδομένων.
- **DML** (Data Manipulation Language, Γλώσσα Χειρισμού Δεδομένων), με την οποία επεξεργαζόμαστε τα δεδομένα μιας βάσης δεδομένων.
- **DCL** (Data Control Language, Γλώσσα Ελέγχου Δεδομένων), με την οποία εξασφαλίζουμε την ασφάλεια και την ακεραιότητα των δεδομένων μιας βάσης δεδομένων.

Παράδειγμα 6 - Οι Γλώσσες 4ης Γενιάς (4GL)

Οι Γλώσσες 4ης Γενιάς **4GL** (4th Generation Languages) είναι γλώσσες προγραμματισμού υψηλού επιπέδου, δηλ. έχουμε την δυνατότητα να κάνουμε πολλές και σύνθετες λειτουργίες με την βοήθειά τους και με λίγες μόνο εντολές.

Οι γλώσσες αυτές στηρίζουν την αποθήκευση των δεδομένων στην θεωρία των βάσεων δεδομένων και έτσι εξασφαλίζεται η ακεραιότητα των πληροφοριών ενώ παρέχουν τα βασικά εργαλεία αναζήτησης και χειρισμού των δεδομένων. Για την Oracle υπάρχει το Accell και για την Informix το Informix-4GL.

Παράδειγμα 7 - Οι Ιεραρχικές Βάσεις Δεδομένων

Στις **Ιεραρχικές (Hierarchical)** βάσεις δεδομένων τα δεδομένα αναπαρίστανται με δενδρικής μορφής δομές δεδομένων και συνδέονται μεταξύ τους με συνδέσμους (links). Η κάθε εγγραφή μπορεί να συνδέεται προς τα πάνω μόνο με μία άλλη εγγραφή (γονέας), ενώ μπορεί να έχει έως δύο εγγραφές που να εξαρτώνται απ' αυτήν (παιδιά). Υπάρχει μία μόνο εγγραφή ρίζα (root), απ' την οποία εξαρτώνται όλες οι άλλες εγγραφές της βάσης δεδομένων. Εχουν το μειονέκτημα ότι είναι πολύπλοκες στην επεξεργασία των εγγραφών τους (προσθήκη, διαγραφή, τροποποίηση).

Παράδειγμα 8 - Οι Δικτυωτές Βάσεις Δεδομένων

Στις **Δικτυωτές (Network)** βάσεις δεδομένων τα δεδομένα αναπαρίστανται με δενδρικής μορφής δομές δεδομένων και συνδέονται μεταξύ τους με συνδέσμους (links),

όπως ακριβώς και στις ιεραρχικές βάσεις δεδομένων, με την διαφορά ότι μια εγγραφή μπορεί να συνδέεται προς τα πάνω με περισσότερες από μία πατρικές εγγραφές (parent records). Είναι λογικά πιο δύσχρηστες αλλά και πιο γρήγορες από τις ιεραρχικές βάσεις δεδομένων.

Παράδειγμα 9 - Οι Σχεσιακές Βάσεις Δεδομένων

Στις **Σχεσιακές (Relational)** βάσεις δεδομένων, τα δεδομένα συνδέονται μεταξύ τους με σχέσεις (relations), οι οποίες προκύπτουν από τα κοινά πεδία που υπάρχουν σε διαφορετικά αρχεία. Τα αρχεία αποκαλούνται πίνακες (tables), οι εγγραφές γραμμές (rows) και τα πεδία στήλες (columns). Η ύπαρξη μιας κοινής τιμής στα πεδία δύο αρχείων καθορίζει και μια σχέση μεταξύ των γραμμών διαφορετικών πινάκων.

Οι σχεσιακές βάσεις δεδομένων έχουν το πλεονέκτημα ότι είναι λογικά κατανοητές και πολύ ευέλικτες και δεκτικές σε αλλαγές.

Παράδειγμα 10 - Τι Είναι ο DBA

Ο Διαχειριστής μιας Βάσης Δεδομένων (**DBA**, DataBase Administrator) είναι αυτός που έχει την ευθύνη για τον σωστό, αποδοτικό και αξιόπιστο τρόπο δημιουργίας και λειτουργίας μια βάσης δεδομένων. Οι αρμοδιότητές του είναι οι εξής

- Η απόφαση για το είδος των πληροφοριών που πρέπει να αποθηκευθούν.
- Η απόφαση για τον τρόπο αποθήκευσης και πρόσβασης στις πληροφορίες αυτές.
- Η συνεργασία με τους τελικούς χρήστες.
- Η απόφαση για τον τρόπο εξασφάλισης των πληροφοριών.
- Η απόφαση για το κάθε πότε θα γίνονται αντίγραφα ασφαλείας (backup) των αρχείων.
- Η παρακολούθηση της σωστής λειτουργίας της βάσης δεδομένων και η απαιτούμενη προσαρμογή της.

Εισαγωγή στη Σχεσιακή Άλγεβρα

Εισαγωγή στη Σχεσιακή Άλγεβρα

Με τη βοήθεια του σχεσιακού μοντέλου δεδομένων είχαμε μια επανάσταση στον σχεδιασμό των βάσεων δεδομένων, καθώς για πρώτη φορά μπορέσαμε να περιγράψουμε τις βάσεις δεδομένων με αυστηρές μαθηματικές έννοιες. Το σχεσιακό μοντέλο είναι ένα σύστημα, τα δομικά (θεμελιώδη) στοιχεία του οποίου είναι οι σχέσεις, οι οποίες μας είναι γνωστές ως πίνακες (tables).

Οι βασικές λειτουργίες ή πράξεις που μπορούμε να κάνουμε στα στοιχεία (δεδομένα) των πινάκων ενός σχεσιακού μοντέλου είναι οι εξής :

- Επιλογή (Selection)
- Προβολή (Projection)
- Καρτεσιανό γινόμενο (Cartesian product)
- Ένωση (Union)
- Διαφορά (Difference)

Εκτός από τις παραπάνω βασικές πράξεις, υπάρχουν και τρεις ακόμα σύνθετες πράξεις, οι οποίες εκφράζονται με τη βοήθεια των βασικών πράξεων και είναι οι εξής:

- Σύνδεση (Join)
- Τομή (Intersection)
- Διαίρεση (Division)

Η Πράξη της Επιλογής (Selection)

Με την πράξη της Επιλογής επιλέγουμε να εμφανίσουμε κάποιες συγκεκριμένες εγγραφές από μια σχέση, οι οποίες ικανοποιούν κάποια κριτήρια που έχουμε θέσει. Η πράξη της Επιλογής συμβολίζεται με το **σ** και μπορούμε να δώσουμε τον ορισμό ότι **Επιλογή** είναι η θεμελιώδης πράξη που εφαρμόζεται σε μια σχέση και έχει ως αποτέλεσμα μια καινούργια σχέση, η οποία περιέχει ορισμένες μόνο πλειάδες (γραμμές ή εγγραφές) της αρχικής σχέσης που ικανοποιούν μια ορισμένη συνθήκη (κατηγόρημα).

Για παράδειγμα, η αλγεβρική πράξη $\sigma_{\text{βαθμός>10}}$**(ΜΑΘΗΤΗΣ)**, εμφανίζει εκείνους μόνο τους μαθητές, μ' όλα τα στοιχεία (πεδία) του καθενός βέβαια, που έχουν βαθμό μεγαλύτερο από 10.

Η αλγεβρική πράξη $\sigma_{\text{υπόλοιπο>1000}}$**(ΠΕΛΑΤΗΣ)**, εμφανίζει εκείνους μόνο τους πελάτες μιας επιχείρησης, μ' όλα τα στοιχεία (πεδία) του καθενός βέβαια, που έχουν υπόλοιπο μεγαλύτερο από 1000 €.

Η αλγεβρική πράξη $\sigma_{\text{πόλη="ΦΛΩΡΙΝΑ"}}$**(ΠΕΛΑΤΗΣ)**, εμφανίζει εκείνους μόνο τους πελάτες μιας επιχείρησης, μ' όλα τα στοιχεία (πεδία) του καθενός βέβαια, που μένουν στην πόλη της Φλώρινας.

Η Πράξη της Προβολής (Projection)

Με την πράξη της Προβολής επιλέγουμε να εμφανίσουμε κάποιες συγκεκριμένες στήλες (πεδία) από μια σχέση. Η πράξη της Προβολής συμβολίζεται με το **Π** και μπορούμε να δώσουμε τον ορισμό ότι **Προβολή** είναι η θεμελιώδης πράξη που εφαρμόζεται σε μια σχέση και έχει ως αποτέλεσμα μια καινούργια σχέση, η οποία περιέχει ορισμένες μόνο στήλες (πεδία) της αρχικής σχέσης.

Για παράδειγμα, η αλγεβρική πράξη $\Pi_{\text{επώνυμο, όνομα, τάξη}}$**(ΜΑΘΗΤΗΣ)**, εμφανίζει μόνο τα πεδία επώνυμο, όνομα και τάξη όλων των μαθητών.

Η αλγεβρική πράξη $\mathbf{\Pi_{επώνυμο,\ όνομα,\ πόλη}(ΠΕΛΑΤΗΣ)}$, εμφανίζει μόνο τα πεδία επώνυμο, όνομα και πόλη όλων των πελατών μιας επιχείρησης.

Η Πράξη του Καρτεσιανού Γινομένου

Η πράξη του Καρτεσιανού Γινομένου είναι ανάλογη με την γνωστή πράξη του πολλαπλασιασμού και μπορούμε να την εφαρμόσουμε σε δύο ή και σε περισσότερες σχέσεις για να συνδυάσουμε τα δεδομένα τους.

Για παράδειγμα, ας θεωρήσουμε έναν πίνακα (σχέση) που περιέχει στοιχεία για 6 διεθνείς αγώνες στίβου, με πεδία ΚωδικόςΑγώνα, Πόλη, Χώρα, Ημερομηνία και έναν άλλον πίνακα (σχέση) που περιέχει στοιχεία για 8 αθλητές, με πεδία ΚωδικόςΑθλητή, Ονοματεπώνυμο, Χώρα.

Για να δημιουργήσουμε το καρτεσιανό γινόμενο του πίνακα των Αγώνων Στίβου με τον πίνακα των Αθλητών, θα πρέπει να δημιουργήσουμε μια καινούργια σχέση με 6 Χ 8 = 48 πλειάδες (εγγραφές), όπου ο κάθε Αγώνας Στίβου θα συνδυάζεται μ' όλους τους αθλητές, και 6 + 8 = 14 στήλες (πεδία), όπου απλά θα παρατάξουμε τα πεδία στη σειρά και θα χρησιμοποιήσουμε ένα κατάλληλο πρόθεμα για το πεδίο Χώρα, που είναι κοινό στους δύο πίνακες.

Η πράξη του Καρτεσιανού Γινομένου ανάμεσα σε δύο σχέσεις R και S, συμβολίζεται με Χ (RXS) και δημιουργεί μια καινούργια σχέση η οποία αποτελείται απ' όλους τους δυνατούς συνδυασμούς κάθε εγγραφής της σχέσης R με κάθε εγγραφή της σχέσης S.

Η Πράξη της Ένωσης (Union)

Η πράξη της Ένωσης ανάμεσα σε δύο σχέσεις R και S, συμβολίζεται με U (RUS) και δημιουργεί μια καινούργια σχέση η οποία αποτελείται απ' όλες τις εγγραφές και των δύο σχέσεων, με την προϋπόθεση φυσικά να μην υπάρχουν διπλότυπες εγγραφές και οι δύο σχέσεις να έχουν τα ίδια ακριβώς πεδία και σε πλήθος και σε τύπο δεδομένων.

Αν δύο σχέσεις δεν είναι συμβατές κατά την ένωση (union-compatible), μπορούμε να χρησιμοποιήσουμε την πράξη της προβολής για να απομονώσουμε (αφαιρέσουμε) όσα πεδία χρειασθεί, έτσι ώστε να μπορέσει να γίνει η ένωση των δύο σχέσεων.

Για παράδειγμα, αν μια σχέση με όνομα ΓιατροίΠαθολογικήςΚλινικής περιέχει τα πεδία ΚωδικόςΓιατρού, Επώνυμο, Όνομα, Ειδικότητα, Βαθμός, Ημερομηνία Πρόσληψης, τότε για να μπορέσει να γίνει ένωση αυτής της σχέσης με μια σχέση με όνομα ΓιατροίΚαρδιολογικήςΚλινικής, θα πρέπει και η δεύτερη σχέση να περιέχει τα ίδια ακριβώς πεδία. Αν τυχόν υπάρχουν Γιατροί που είναι γραμμένοι και στις δύο Κλινικές, τα στοιχεία τους θα εμφανισθούν μία μόνο φορά στην καινούργια σχέση της ένωσης.

Η Πράξη της Διαφοράς (Difference)

Η πράξη της Διαφοράς ανάμεσα σε δύο σχέσεις R και S, συμβολίζεται με – (R–S) και δημιουργεί μια καινούργια σχέση η οποία αποτελείται απ' όλες τις εγγραφές που βρίσκονται στην πρώτη αλλά όχι και στη δεύτερη σχέση, με την προϋπόθεση φυσικά και οι δύο σχέσεις να έχουν τα ίδια ακριβώς πεδία και σε πλήθος και σε τύπο δεδομένων.

Αν δύο σχέσεις δεν είναι συμβατές κατά τη διαφορά, μπορούμε να χρησιμοποιήσουμε την πράξη της προβολής για να απομονώσουμε (αφαιρέσουμε) όσα πεδία χρειασθεί, έτσι ώστε να μπορέσει να γίνει η διαφορά των δύο σχέσεων.

Για παράδειγμα, αν μια σχέση με όνομα ΓιατροίΚλινικής περιέχει τα πεδία Κωδικός Γιατρού, Επώνυμο, Όνομα, Ειδικότητα, Βαθμός, Ημερομηνία Πρόσληψης, τότε για να μπορέσει να γίνει η διαφορά αυτής της σχέσης με την σχέση ΓιατροίΚαρδιολογικήςΚλινικής, δηλ. στην ουσία η αφαίρεση των Γιατρών της Καρδιολογικής Κλινικής από το σύνολο των Γιατρών της Κλινικής, θα πρέπει και η δεύτερη

σχέση να περιέχει τα ίδια ακριβώς πεδία. Οι Γιατροί που είναι γραμμένοι στην Καρδιολογική Κλινική, δεν θα περιέχονται στο αποτέλεσμα της πράξης της διαφοράς.

Η Βάση Δεδομένων MySQL

Μάθημα 1 - Τι Είναι οι Βάσεις Δεδομένων (Databases)

Μια **βάση δεδομένων** (database) αποτελείται από έναν ή περισσότερους **πίνακες** (tables), ο καθένας από τους οποίους περιέχει μια λίστα από κάποια πράγματα. Για μια βάση δεδομένων πελατών (clients), είναι φυσικό να ξεκινήσουμε μ' έναν πίνακα με όνομα clients που θα περιέχει μια λίστα από στοιχεία πελατών.

Ο κάθε πίνακας σε μια βάση δεδομένων περιέχει μια ή περισσότερες **στήλες** (columns) ή **πεδία** (fields), όπου η κάθε στήλη περιέχει μια συγκεκριμένη πληροφορία για τον κάθε πελάτη που υπάρχει στην βάση δεδομένων (database).

Ο πίνακας clients μπορεί να περιέχει στήλες για τον κωδικό ενός πελάτη (ID), για το όνομά του (Name) καθώς και για την ημερομηνία γέννησής του (Date). Το κάθε ανέκδοτο που αποθηκεύουμε σ' αυτόν τον πίνακα λέμε ότι αποτελεί μια **γραμμή** (row) ή μια **εγγραφή** (record) του πίνακα. Για παράδειγμα, ας δούμε τον παρακάτω πίνακα :

ID	Name	Date
1	Αντωνιάδης	1970-04-01
2	Παπαδόπουλος	1968-02-22

Εκτός από τις στήλες για το όνομα του πελάτη (Name) και την ημερομηνία γέννησής του (Date), υπάρχει και μια στήλη με όνομα ID, ο σκοπός της οποίας είναι να εκχωρήσει έναν μοναδικό αριθμό στον κάθε πελάτη έτσι ώστε να έχουμε έναν εύκολο τρόπο αναφοράς σ' αυτόν και να μπορούμε να τον ξεχωρίσουμε από τους άλλους πελάτες.

Σαν επισκόπηση, το παραπάνω είναι ένας πίνακας τριών στηλών που περιέχει δύο γραμμές ή καταχωρήσεις. Η κάθε γραμμή του πίνακα περιέχει έναν κωδικό (ID) αναγνώρισης του πελάτη, το όνομά του (text) καθώς και την ημερομηνία γέννησής του (date). Με βάση αυτήν την βασική ορολογία, είμαστε έτοιμοι να αρχίσουμε να χρησιμοποιούμε την MySQL.

Μάθημα 2 - Εκκίνηση (Logging onto) της MySQL

Το standard interface για να δουλέψουμε με τις βάσεις δεδομένων της MySQL είναι να συνδεθούμε με το λογισμικό του MySQL server και να δίνουμε μία εντολή την φορά. Για να κάνουμε αυτήν την σύνδεση με τον server, θα χρειασθούμε το πρόγραμμα πελάτη (client program) της MySQL.

Στο Linux, το πρόγραμμα αποκαλείται **mysql** και βρίσκεται εξ ορισμού στον κατάλογο */usr/local/mysql/bin*, ενώ στα Windows, το πρόγραμμα αποκαλείται **mysql.exe** και βρίσκεται εξ ορισμού στον κατάλογο *C:\mysql\bin*.

Υπάρχουν δύο τρόποι για να μπορέσουμε να συνδεθούμε με τον MySQL server. Ο πρώτος είναι να χρησιμοποιήσουμε το telnet για να συνδεθούμε (log into) στον server του Web host που μας φιλοξενεί και να δώσουμε την εντολή mysql από εκεί.

Ο δεύτερος είναι να φορτώσουμε (download) και να εγκαταστήσουμε το λογισμικό πελάτη (client software) της MySQL από το site http://www.mysql.com/ στον δικό μας υπολογιστή και να το χρησιμοποιήσουμε για να συνδεθούμε με τον MySQL server.

Οποια μέθοδο κι αν επιλέξουμε και όποιο λειτουργικό σύστημα χρησιμοποιούμε, θα καταλήξουμε σε μια γραμμή εντολών (command line), έτοιμοι να εκτελέσουμε το πρόγραμμα πελάτη της MySQL για να συνδεθούμε στον MySQL server. Πρέπει να γράψουμε τα εξής :

mysql -h <hostname> -u <username> -p

Θα πρέπει να αντικαταστήσουμε το <hostname> με το όνομα του host ή την IP διεύθυνση του υπολογιστή στον οποίο εκτελείται ο MySQL server. Αν εκτελούμε το πρόγραμμα πελάτη στον ίδιο υπολογιστή με τον server, μπορούμε να παραλείψουμε το τμήμα -h <hostname> της εντολής αντί να γράψουμε -h localhost, για παράδειγμα. Το <username> πρέπει να είναι το δικό μας όνομα χρήστη στην MySQL.

Αν εγκαταστήσαμε εμείς οι ίδιοι τον MySQL server, αυτό θα είναι το root, ενώ αν χρησιμοποιούμε τον MySQL server του Web host που μας φιλοξενεί, αυτό θα πρέπει να είναι το όνομα χρήστη της MySQL που μας έχει δοθεί.

Το όρισμα -p λέει στο πρόγραμμα να ζητήσει από μας τον κωδικό εισόδου (password), το οποίο θα συμβεί μόλις δώσουμε την παραπάνω εντολή. Αν έχουμε εγκαταστήσει εμείς οι ίδιοι τον MySQL, αυτό το password θα είναι το root password που επιλέξαμε εμείς, ενώ αν χρησιμοποιούμε τον MySQL server του Web host που μας φιλοξενεί, αυτό θα πρέπει να είναι το password της MySQL που μας έχει δοθεί.

Αν τα γράψαμε όλα σωστά, το πρόγραμμα πελάτη της MySQL θα παρουσιάσει τον εαυτό του και θα εμφανίσει την γραμμή εντολών της MySQL, ως εξής :

mysql>

Τώρα, ο MySQL server είναι σε θέση να παρακολουθεί περισσότερες από μία βάσεις δεδομένων, που αυτό σημαίνει ότι Web host μπορεί να στήσει έναν μόνο MySQL server για να χρησιμοποιηθεί από πολλούς από τους συνδρομητές του.

Ετσι, το επόμενο βήμα μας θα πρέπει να είναι να επιλέξουμε την βάση δεδομένων με την οποία θα δουλέψουμε. Πρώτα απ' όλα, θα δούμε μια λίστα των βάσεων δεδομένων που υπάρχουν στον τρέχοντα server. Δίνουμε την επόμενη εντολή και μετά ENTER.

mysql> SHOW DATABASES;

Η MySQL θα εμφανίσει μια λίστα με τις βάσεις δεδομένων που υπάρχουν στον server, ως εξής :

Database
mysql
test
2 rows in set (0.11 sec)

Ο MySQL server χρησιμοποιεί την πρώτη βάση δεδομένων, με όνομα mysql, για να μπορεί να παρακολουθεί τους χρήστες, τα συνθηματικά τους (passwords) καθώς και το τι επιτρέπεται να κάνουν. Θα αφήσουμε για λίγο αυτή την βάση δεδομένων.

Η δεύτερη βάση δεδομένων, με όνομα test αποτελεί ένα δείγμα βάσης δεδομένων. Η διαδικασία της διαγραφής στην MySQL αποκαλείται dropping (απόρριψη) και η εντολή για να διαγράψουμε μια βάση δεδομένων είναι η εξής :

mysql> DROP DATABASE test;

Αν δώσουμε αυτήν την εντολή και πατήσουμε Enter, η MySQL θα διαγράψει την βάση δεδομένων και θα εμφανίσει το μήνυμα Query OK σαν επιβεβαίωση. Επειδή αυτή η εντολή δεν εμφανίζει κάποιο μήνυμα προειδοποίησης, πρέπει να είμαστε πολύ προσεκτικοί όταν την δίνουμε.

Θα δούμε τώρα λίγα πράγματα για την γραμμή εντολών (command line) της MySQL. Όλες οι εντολές στην MySQL τελειώνουν με τον χαρακτήρα ; (semicolon). Έτσι, αν έχουμε ξεχάσει να κλείσουμε μια εντολή με τον χαρακτήρα ;, η MySQL θα νομίζει ότι δεν έχουμε τελειώσει με την εντολή αυτή και θα περιμένει να συνεχίσουμε να γράφουμε και στην επόμενη γραμμή :

mysql> SHOW
-> DATABASES;

Η MySQL δείχνει ότι περιμένει από μας να ολοκληρώσουμε την εντολή, αλλάζοντας την προτροπή (prompt) από mysql> σε ->. Αυτό είναι βολικό όταν έχουμε να γράψουμε μακροσκελείς εντολές, καθώς μπορούμε να επεκτείνουμε τις εντολές μας σε πολλές γραμμές.

Για να ακυρώσουμε την τρέχουσα εντολή και να αρχίσουμε να την γράφουμε από την αρχή, γράφουμε τους χαρακτήρες \c και πατάμε ENTER, ως εξής :

mysql> DROP DATABASE\c

mysql>

Η MySQL θα αγνοήσει την εντολή που είχαμε ξεκινήσει και θα περιμένει να δώσουμε μια άλλη εντολή.

Τέλος, αν θέλουμε να εξέλθουμε από το πρόγραμμα πελάτη της MySQL, μπορούμε απλά να γράψουμε quit ή exit. Είναι οι μόνες εντολές που δεν χρειάζονται τον χαρακτήρα ; (semicolon).

mysql> quit

Bye

Μάθημα 3 - Τι Είναι η SQL

Το σύνολο των εντολών που θα χρησιμοποιούμε από δω και πέρα για να λέμε στην MySQL τι να κάνει, αποτελεί μέρος ενός standard που αποκαλείται Δομημένη Γλώσσα Ερωτημάτων (Structured Query Language) ή SQL. Οι εντολές της SQL αποκαλούνται επίσης και ερωτήματα (queries).

Η SQL αποτελεί την standard γλώσσα για αλληλεπίδραση με τις περισσότερες βάσεις δεδομένων, έτσι ακόμα κι αν αλλάξουμε στο μέλλον από την MySQL σε μια βάση δεδομένων όπως την Microsoft SQL Server, θα διαπιστώσουμε ότι οι περισσότερες από τις εντολές είναι ολόιδιες.

Δεν πρέπει να συγχέουμε την SQL με την MySQL. Η MySQL είναι το λογισμικό του διακομιστή βάσεων δεδομένων (database server software) που χρησιμοποιούμε, ενώ η SQL είναι η γλώσσα που χρησιμοποιούμε για να αλληλεπιδράσουμε με την βάση δεδομένων.

Μάθημα 4 - Δημιουργία μιας Βάσης Δεδομένων (DataBase)

Η δημιουργία μιας βάσης δεδομένων είναι πολύ εύκολη υπόθεση :

mysql> CREATE DATABASE clients;

Τώρα που έχουμε μια βάση δεδομένων, πρέπει να ενημερώσουμε την MySQL ότι θέλουμε να την χρησιμοποιήσουμε. Η εντολή αυτή είναι η εξής :

mysql> USE clients;

Μπορούμε τώρα να αρχίσουμε να χρησιμοποιούμε την βάση δεδομένων.

Μάθημα 5 - Δημιουργία ενός Πίνακα (Table)

Η σύνταξη της εντολής SQL για την δημιουργία ενός πίνακα (table) είναι η εξής :

mysql> CREATE TABLE <table name> (
-> <column 1 name> <col. 1 type> <col. 1 details>,
-> <column 2 name> <col. 2 type> <col. 2 details>,
-> ...
->);

Οσον αφορά το παράδειγμα που είχαμε δει νωρίτερα με τον πίνακα Jokes, είχε τις εξής τρεις στήλες (columns) : ***ID*** (αριθμός, number), ***Name*** (το όνομα του πελάτη) και ***Date*** (την ημερομηνία γέννησης του πελάτη).

Η εντολή για να δημιουργήσουμε αυτόν τον πίνακα είναι η εξής :

```
mysql> CREATE TABLE Clients (
-> ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
-> Name TEXT,
-> Date DATE NOT NULL
-> );
```

Ας το αναλύσουμε τώρα :

- Η πρώτη γραμμή είναι αρκετά απλή : λέει ότι θέλουμε να δημιουργήσουμε έναν νέο πίνακα με όνομα Clients.
- Η δεύτερη γραμμή λέει ότι θέλουμε μια στήλη (column) με όνομα ID που θα περιέχει μια ακέραια τιμή (integer, INT). Ακόμη, αυτή η στήλη δεν μπορεί να είναι κενή (NOT NULL). Επίσης, αν δεν καθορίσουμε κάποια συγκεκριμένη τιμή, όταν κάνουμε μια νέα καταχώρηση, η MySQL θα επιλέξει η ίδια μια τιμή που θα είναι κατά ένα μεγαλύτερη από την μεγαλύτερη τιμή του πίνακα μέχρι τώρα (AUTO_INCREMENT). Τέλος, αυτή η στήλη θα ενεργεί σαν ένα μοναδικό αναγνωριστικό (unique identifier) για τις καταχωρήσεις του πίνακα, έτσι όλες οι τιμές αυτής της στήλης θα πρέπει να είναι μοναδικές (PRIMARY KEY).
- Η τρίτη γραμμή είναι πολύ απλή : λέει ότι θέλουμε μια στήλη με όνομα Name που θα περιέχει κείμενο (TEXT).
- Η τέταρτη γραμμή ορίζει την τελευταία στήλη, με όνομα Date, η οποία θα περιέχει δεδομένα του τύπου DATE και η οποία δεν θα μπορεί να είναι κενή (NOT NULL).

Πρέπει να έχουμε υπόψη μας ότι, ενώ μπορούμε να γράψουμε τις εντολές της SQL με πεζά ή με κεφαλαία γράμματα, ένας MySQL server που εκτελείται σ' ένα σύστημα που είναι βασισμένο στο Unix, θα ξεχωρίζει τα πεζά από τα κεφαλαία γράμματα (case sensitive) όσον αφορά τα ονόματα των βάσεων δεδομένων και των πινάκων, εφόσον αυτά αντιστοιχούν σε καταλόγους (directories) και αρχεία (files) στον κατάλογο δεδομένων (data directory) της MySQL.

Αλλιώς, η MySQL δεν ξεχωρίζει καθόλου τα πεζά από τα κεφαλαία γράμματα (case insensitive) αλλά με μια εξαίρεση : τα ονόματα των πινάκων και των στηλών καθώς και τα άλλα ονόματα πρέπει να γράφονται ακριβώς το ίδιο όταν χρησιμοποιούνται περισσότερες από μία φορές στην ίδια εντολή.

Επίσης, εκχωρήσαμε έναν συγκεκριμένο τύπο δεδομένων σε κάθε στήλη που δημιουργήσαμε. Η ID θα περιέχει ακεραίους (integers), η Name θα περιέχει κείμενο (text) και η Date θα περιέχει ημερομηνίες (dates). Η MySQL απιατεί να καθορίζουμε τον τύπο δεδομένων (data type) για κάθε στήλη από την αρχή.

Αν γράψουμε σωστά την παραπάνω εντολή, η MySQL θα εμφανίσει το μήνυμα Query OK και θα έχουμε έτσι δημιουργήσει τον πρώτο μας πίνακα (table).

Για να βεβαιωθούμε ότι πράγματι δημιουργήθηκε ο πίνακας, δίνουμε την εξής εντολή :

```
mysql> SHOW TABLES;
```

Η απάντηση θα είναι ως εξής :

```
Tables in clients
Clients
1 row in set
```

Αυτή είναι η λίστα όλων των πινάκων που υπάρχουν στην βάση δεδομένων clinents και περιέχει έναν μόνο πίνακα : τον Clients. Ας ρίξουμε τώρα μια πιο αναλυτική ματιά στον πίνακα Clients :

mysql> DESCRIBE Clients;
Field | Type | Null | Key | Default
ID | int(11) | | PRI | 0 | ...
Name | text | YES | | NULL
Date | date | | | 0000-00-00
3 rows in set

Η παραπάνω εντολή εμφανίζει μια λίστα των στηλών (πεδίων) που υπάρχουν στον πίνακα. Όπως μπορούμε να δούμε, υπάρχουν τρεις στήλες σ' αυτόν τον πίνακα που εμφανίζονται σαν τρεις γραμμές στο αποτέλεσμα.

Για να διαγράψουμε έναν πίνακα στην MySQL, η εντολή είναι σχεδόν ολόιδια με την εντολή για την διαγραφή μιας βάσης δεδομένων :

mysql> DROP TABLE <tableName>;

Μάθημα 6 - Εισαγωγή Δεδομένων σε Πίνακα (Table)

Η εντολή για να εισάγουμε δεδομένα σε μια βάση δεδομένων αποκαλείται INSERT και υπάρχουν οι εξής δύο βασικές μορφές αυτής της εντολής :

mysql> INSERT INTO <table name> SET
-> columnName1 = value1,
-> columnName2 = value2,
-> ...
-> ;
mysql> INSERT INTO <table name>
-> (columnName1, columnName2, ...)
-> VALUES (value1, value2, ...);

Ετσι, για να προσθέσουμε έναν πελάτη στον πίνακα, μπορούμε να επιλέξουμε μια από τις εξής εντολές :

mysql> INSERT INTO Clients SET
-> Name = "Αβραμόπουλος",
-> Date = "1950-03-21";
mysql> INSERT INTO Clients
-> (Name, Date) VALUES (
-> "Δημητρίου",
-> "1960-07-14"
->);

Πρέπει να έχουμε υπόψη μας ότι στην δεύτερη μορφή της εντολής INSERT, η σειρά με την οποία γράφουμε τις στήλες πρέπει να ταιριάζει με την σειρά με την οποία γράφουμε τις αντίστοιχες τιμές.

Μάθημα 7 - Εμφάνιση των Αποθηκευμένων Δεδομένων

Η εντολή για να δούμε τα δεδομένα που είναι αποθηκευμένα στους πίνακες μιας βάσης δεδομένων είναι η SELECT και αποτελεί την πιο πολύ χρησιμοποιούμενη και πιο πολύπλοκη εντολή της SQL.

Η επόμενη εντολή θα εμφανίσει ό,τι είναι αποθηκευμένο στον πίνακα Clients :

*mysql> SELECT * FROM Clients;*

Αν θέλουμε να εμφανισθούν οι τιμές ορισμένων μόνο στηλών, δίνουμε την εξής εντολή :

mysql> SELECT ID, Date FROM Clients;

Το αποτέλεσμα θα είναι ως εξής :

ID Date
1 1950-03-21
2 1960-07-14
2 rows in set (0.00 sec)

Εκτός από το να εμφανίσουμε τις στήλες που θέλουμε με την εντολή Select, μπορούμε να τροποποιήσουμε την εμφάνισή τους με συναρτήσεις (functions). Μια συνάρτηση, η LEFT(), εμφανίζει έναν μέγιστο καθορισμένο αριθμό χαρακτήρων για μια στήλη.

Για παράδειγμα, αν θέλουμε να δούμε μόνο τους 10 πρώτους χαρακτήρες της στήλης Name, μπορούμε να δώσουμε την εξής εντολή :

mysql> SELECT ID, LEFT(Name, 10), Date FROM Clients;
ID LEFT(Name, 20) Date
1 Αβραμόπουλ 1950-03-21
2 Δημητρίου 1960-07-14
2 rows in set (0.05 sec)

Μια άλλη χρήσιμη συνάρτηση είναι η COUNT(), η οποία απλά μετράει τον αριθμό των επιστρεφόμενων αποτελεσμάτων. Έτσι, για παράδειγμα, αν θέλουμε να βρούμε πόσοι πελάτες υπάρχουν στον πίνακα, μπορούμε να χρησιμοποιήσουμε την εξής εντολή :

mysql> SELECT COUNT(*) FROM Clients;
COUNT(*)
2
2 rows in set (0.06 sec)

Μάθημα 8 - Η Δήλωση WHERE

Χρησιμοποιώντας την δήλωση (clause) **WHERE** σε μια εντολή SELECT, μπορούμε να περιορίσουμε τα επιστρεφόμενα αποτελέσματα χρησιμοποιώντας κάποια συνθήκη, ως εξής :

mysql> SELECT COUNT(*) FROM Clients
-> WHERE Date >= "1950-01-01";

Το παραπάνω ερώτημα (query) θα μετρήσει τον αριθμό των πελατών που έχουν ημερομηνίες γέννησης μεγαλύτερες από ή ίσες από την 1η Ιανουαρίου 1950.

Μια άλλη παραλλαγή της δήλωσης WHERE που μας δίνει την δυνατότητα να αναζητήσουμε καταχωρήσεις που περιέχουν ένα συγκεκριμένο κομμάτι κειμένου, είναι η εξής :

mysql> SELECT Name FROM Clients
-> WHERE Name LIKE "%ίδη%";

Αυτό το ερώτημα (query) εμφανίζει το κείμενο όλων των ονομάτων των πελατών που περιέχουν το κείμενο ίδη στην στήλη Name. Η λέξη κλειδί (keyword) **LIKE** λέει στην MySQL ότι η συγκεκριμένη στήλη πρέπει να ταιριάζει με το δεδομένο υπόδειγμα, που σ' αυτήν την περίπτωση είναι το "%ίδη%".

Τα σύμβολα % εδώ σημαίνουν ότι το κείμενο ίδη μπορεί να έχει πριν και μετά ένα οποιοδήποτε κείμενο. Κάτι δηλαδή σαν τον γνωστό μας χαρακτήρα μπαλαντέρ (wildcard) * από άλλα προγράμματα.

Μπορούμε επίσης να συμπεριλάβουμε και συνθήκες στην δήλωση WHERE για να περιορίσουμε κι άλλο τα αποτελέσματα. Για παράδειγμα, για να εμφανίσουμε τους πελάτες που περιέχουν το κείμενο ίδη στο όνομά τους και που γεννήθηκαν μόνο τον Απρίλιο του 1961, μπορούμε να χρησιμοποιήσουμε το εξής ερώτημα :

mysql> SELECT Name FROM Clients WHERE
-> Name LIKE "%ίδη%" AND
-> Date >= "1961-04-01" AND
-> Date < "1961-05-01";

Μάθημα 9 - Τροποποίηση των Αποθηκευμένων Δεδομένων

Για να κάνουμε αλλαγές στις τιμές ενός πίνακα μιας βάσης δεδομένων, χρησιμοποιούμε την εντολή UPDATE, η σύνταξη της οποίας είναι ως εξής :

mysql> UPDATE <tableName> SET
-> <col_name>=<new_value>, ...
-> WHERE <where clause>;

Ετσι, για παράδειγμα, αν θελήσουμε να αλλάξουμε την ημερομηνία γέννησης ενός πελάτη, θα πρέπει να δώσουμε την εξής εντολή :

mysql> UPDATE Clients SET Date="1960-04-01" WHERE ID=1;

Η στήλη ID είναι βολική σ' αυτήν την περίπτωση καθώς μας δίνει την δυνατότητα να διαχωρίσουμε εύκολα και σίγουρα τον πελάτη που θέλουμε να τροποποιήσουμε. Μπορούμε να χρησιμοποιήσουμε και την δήλωση WHERE, όπως στην επόμενη εντολή η οποία αλλάζει την ημερομηνία όλων των καταχωρήσεων πελατων που περιέχουν το κείμενο ίδη :

mysql> UPDATE Clients SET Date="1970-04-01"
-> WHERE Name LIKE "%ίδη%";

Μάθημα 10 - Διαγραφή Αποθηκευμένων Δεδομένων

Για να διαγράψουμε γραμμές (εγγραφές) στην SQL, η σύνταξη είναι η εξής :

mysql> DELETE FROM <tableName> WHERE <where clause>;

Ετσι, για παράδειγμα, για να διαγράψουμε όλους τους πελάτες που περιέχουν το κείνενο ίδη από τον πίνακα Clients, πρέπει να δώσουμε το επόμενο ερώτημα :

mysql> DELETE FROM Clients WHERE Name LIKE
"%ίδη%";

Η επόμενη εντολή θα αδειάσει τον πίνακα Clients με μιας :

mysql> DELETE FROM Clients;

Η Βάση Δεδομένων Oracle

Μάθημα 1 - Τι Είναι η Oracle

Η Oracle ανήκει στις Σχεσιακές Βάσεις Δεδομένων (Relational DataBases), δηλ. στηρίζεται σε σχέσεις (relations) που δηλώνονται με βάση τα κοινά πεδία διαφορετικών πινάκων (tables). Αυτές οι σχέσεις, στην ουσία τα κοινά πεδία, μπορούν να καθορισθούν πριν αλλά και μετά από την δημιουργία των αρχείων και με την βοήθειά τους μπορούμε να έχουμε ευέλικτα και εύκολα διαχειρίσιμα συστήματα πληροφόρησης.

Η Oracle διαθέτει την γλώσσα αναζήτησης ή ερωτημάτων (query language) SQL*Plus, με την βοήθεια της οποίας μπορούμε να διαχειριστούμε τις πληροφορίες μιας βάσης δεδομένων της Oracle. Με την SQL*Plus μπορούμε να δημιουργήσουμε πίνακες, εγγραφές, πεδία και σχέσεις και στην συνέχεια να κάνουμε εργασίες ανεύρεσης και ενημέρωσης (τροποποίησης) των αποθηκευμένων δεδομένων, παρέχοντας έτσι ένα δυναμικό εργαλείο διαχείρισης ενός συστήματος πληροφόρησης.

Πριν κάνουμε ο,τιδήποτε, πρέπει να έχουμε υπόψη μας ότι παίζει πολύ μεγάλο ρόλο η σωστή οργάνωση της βάσης δεδομένων και η δημιουργία των σωστών σχέσεων μεταξύ των αρχείων που την αποτελούν (DataBase design).

Αν δουλεύουμε σαν χρήστες της Oracle σ' ένα μεγάλο σύστημα, ο Διαχειριστής της Βάσης Δεδομένων, DataBase Administrator (DBA), θα πρέπει να μας παραχωρήσει ένα όνομα χρήστη (user name) της Oracle για όσο χρόνο χρησιμοποιούμε την Oracle.

Μάθημα 2 - Τα Μέρη της Oracle

Οπως είδαμε είδαμε, η Oracle διαθέτει σαν εργαλείο αναζήτησης και χειρισμού πληροφοριών την γλώσσα ερωτημάτων SQL*Plus, η οποία δέχεται τα εξής δύο είδη εντολών :

- Εντολές SQL για την επεξεργασία των πληροφοριών.
- Εντολές SQL*Plus για την μεταβολή των εντολών SQL.

Για να διακόψουμε μια εκτελούμενη εντολή της Oracle πρέπει να πατήσουμε το πλήκτρο Delete, ενώ για να σταματήσουμε την λειτουργία της Oracle και να επιστρέψουμε στο λειτουργικό σύστημα πρέπει να γράψουμε Exit.

Το prompt όταν η Oracle είναι σε λειτουργία και περιμένει κάποια εντολή της SQL* από μας είναι το

SQL>

Μια εντολή της SQL* δεν τελειώνει με πάτημα του πλήκτρου Enter αλλά με τον χαρακτήρα ; και μπορούμε να έχουμε μια εντολή της SQL* σε πολλές γραμμές.

Μάθημα 3 - Οι Κατηγορίες Εντολών της SQL*

Οι εντολές της SQL* χωρίζονται στις εξής τρεις κατηγορίες, ανάλογα με το αντικείμενο και την λειτουργία τους :

Εντολές για Ερωτήματα (Queries)

- **Select**, για εμφάνιση πληροφοριών.

Εντολές Αλλαγής Στοιχείων (DML)

- **Insert**, για εισαγωγή νέων γραμμών (εγγραφών).
- **Update**, για τροποποίηση γραμμών.
- **Delete**, για διαγραφή γραμμών.

Εντολές Δημιουργίας Αντικειμένων της Βάσης (DDL)

- **Create Table**, για δημιουργία νέου πίνακα.
- **Create View**, για δημιουργία νέας όψης.
- **Alter Table**, για τροποποίηση της σχεδίασης πίνακα.

Εντολές Ελέγχου Ασφάλειας της Βάσης (DCL)

- **Grant**
- **Revoke**
- **Commit**
- **RollBack**

Μάθημα 4 - Δημιουργία Πίνακα (Table)

Για να δημιουργήσουμε έναν πίνακα (table), χρησιμοποιούμε την εντολή **Create Table** και προσδιορίζουμε το όνομα του πίνακα και μέσα σε παρενθέσεις τις στήλες (πεδία) που θα περιέχει, ως εξής :

- Create Table Clients
  (ClientId char(5) not null, Name char(20), Address char(25), Phone char(15), Poso number(10, 2));
- Create Table Orders
  (OrderId char(6) not null, OrderClientId char(5) not null, OrderDate date, OrderValue number);

Μάθημα 5 - Καταχώρηση Δεδομένων σε Πίνακα

Για να καταχωρήσουμε (εισάγουμε) στοιχεία σ' έναν πίνακα, χρησιμοποιούμε την εντολή **Insert Into**, γράφουμε το όνομα του πίνακα, μετά την δήλωση **Values** και τις τιμές που θέλουμε μέσα σε παρενθέσεις και χωρισμένες με κόμμα, ως εξής :

```
Insert Into Clients Values ('100', 'Αντωνιάδης Νικόλαος', 'Π. Μελά 100', '03850-22000', 100000);
Insert Into Orders Values ('200', '100', '11-09-2001', 200000);
```

Μάθημα 6 - Η Εντολή Select

Με την εντολή Select μπορούμε να επιλέξουμε και να εμφανίσουμε κάποια στοιχεία (γραμμές και στήλες) από έναν ή περισσότερους πίνακες μιας βάσης δεδομένων. Η εντολή χρησιμοποιεί την δήλωση From, για να ξέρει από ποιον πίνακα θα πάρει τα στοιχεία, και την δήλωση Where, για να ξέρει ποια συνθήκη θα εφαρμόσει για να απομονώσει τα στοιχεία που μας ενδιαφέρουν.

Η σύνταξη είναι η εξής :

```
Select πεδίο1, πεδίο2, ...
    From πίνακας1, πίνακας2, ...
        Where συνθήκη;
```

Ακολουθεί ένα παράδειγμα.

```
Select Name, Address, Phone
    From Clients
        Where Poso = 100000;
```

Για να μην εμφανισθούν περισσότερες από μία φορές γραμμές που έχουν τις ίδιες τιμές σ' όλα τα πεδία, χρησιμοποιούμε την δήλωση Distinct πριν από τα ονόματα των πεδίων, ως εξής :

```
Select Distinct Name, Address, Phone
    From Clients;
```

Με την εντολή Select * From Clients εμφανίζονται όλα τα πεδία του πίνακα.

Μάθημα 7 - Η Δήλωση Where

Με την δήλωση Where μπορούμε να ορίσουμε συνθήκες επιλογής γραμμών με βάση τις τιμές ορισμένων πεδίων ενός πίνακα. Πρώτα γράφουμε την στήλη στην οποία θα βασιστεί το ερώτημα (query), ύστερα τον τελεστή σύγκρισης και τέλος την τιμή του πεδίου με την οποία θα γίνει η σύγκριση με την τιμή της στήλης.

Οι τελεστές που μπορούμε να χρησιμοποιήσουμε είναι οι εξής :

- **=**, ισότητα.
- **!=**, ανισότητα.
- **>**, μεγαλύτερο από.
- **<**, μικρότερο από.
- **>=**, μεγαλύτερο ή ίσο από.
- **<=**, μικρότερο ή ίσο από.
- **Between** ... And, εύρος τιμών.
- **In** (λίστα), επιλογή από τιμές μιας λίστας.
- **Like**, σύγκριση με ακολουθία τιμών.
- **Is Null**, χωρίς τιμή, αλλά όχι μηδενική τιμή.

Μπορούμε να χρησιμοποιήσουμε τον λογικό τελεστή άρνησης Not για να αντιστρέψουμε όποια συνθήκη θέλουμε και μπορούμε ακόμη να χρησιμοποιήσουμε και τους λογικούς τελεστές σύζευξης And και διάζευξης Or.

Η Oracle χρησιμοποιεί τους εξής χαρακτήρες μπαλαντέρ (wildcards) :

- %, αντικαθιστά πολλούς χαρακτήρες.
- _, αντικαθιστά έναν μόνο χαρακτήρα.

Ακολουθούν παραδείγματα.

```
Select Name, Address, Phone, Poso
    From Clients
        Where (ClientID > 100 And Poso=100000) Or Poso <50000;
Select Name, Address, Phone
    From Clients
        Where Name Like 'AN%';
```

Μάθημα 8 - Η Δήλωση Order By

Με την δήλωση Order By μπορούμε να ταξινομήσουμε τις εγγραφές που έχουμε επιλέξει με βάση την τιμή ενός πεδίου. Εξ ορισμού, η ταξινόμηση είναι αύξουσα αλλά με την δήλωση Desc μπορούμε να ορίσουμε φθίνουσα ταξινόμηση. Μπορούμε να ορίσουμε ταξινόμηση και με βάση τις τιμές περισσότερων από μία στήλες. Ακολουθεί ένα παράδειγμα.

```
Select Name, Address, Phone, Poso
    From Clients
        Where ClientID > 100 Order By Poso (Desc), Name;
```

Μάθημα 9 - Συνδυασμός Πινάκων

Μπορούμε να συνδυάσουμε τις γραμμές δύο πινάκων, εφόσον υπάρχουν κοινές στήλες στους δύο πίνακες. Αυτές οι στήλες μπορούν να έχουν το ίδιο όνομα ή όχι, αλλά οπωσδήποτε πρέπει να έχουν τον ίδιο τύπο δεδομένων. Η συνθήκη Where καθορίζει τον

τρόπο συνδυασμού των πινάκων με βάση τις τιμές της κοινής στήλης των δύο πινάκων. Ακολουθεί ένα παράδειγμα.

```
Select Name, OrderId
    From Clients, Orders
        Where Clients.ClientId = Orders.OrderClientId;
```

Μπορούμε να χρησιμοποιήσουμε και σύντομα ονόματα ή ψευδώνυμα (aliases) για τους πίνακες, ως εξής :

```
Select O.*, Name, Poso
    From Clients C, Orders O
        Where C.ClientId = O.OrderClientId Order By OrderId;
```

Μάθημα 10 - Αριθμητικές Πράξεις σε Ερωτήματα

Μέσα στις εντολές της SQL μπορούμε να κάνουμε και τις τέσσερις βασικές αριθμητικές πράξεις πάνω στις στήλες μιας εντολής Select. Ακολουθούν παραδείγματα.

```
Select Name, Address, Poso
    From Clients
        Where 0.5 * Poso > 100000;
```

Μάθημα 11 - Οι Αριθμητικές Συναρτήσεις

Μπορούμε να χρησιμοποιήσουμε αριθμητικές συναρτήσεις σε μια εντολή Select. Οι συναρτήσεις αυτές είναι οι εξής :

- **Abs()**, απόλυτη τιμή.
- **Greatest()**, η μέγιστη τιμή μιας λίστας.
- **Least()**, η ελάχιστη τιμή μιας λίστας.
- **Round()**, στρογγυλοποίηση δεκαδικών αριθμών.
- **Trunc()**, αποκοπή ψηφίων δεκαδικών αριθμών.
- **To_Number()**, μετατροπή χαρακτήρων σε αριθμούς.

Οι παραπάνω συναρτήσεις επιδρούν σε κάθε επιλεγόμενη γραμμή ενός ερωτήματος, ενώ οι παρακάτω συναρτήσεις επιδρούν σε μια ομάδα επιλεγόμενων γραμμών και επιστρέφουν μία μόνο τιμή για πολλές γραμμές :

- **Avg()**, μέσος όρος τιμών μιας στήλης.
- **Count()**, πλήθος τιμών μιας στήλης.
- **Max()**, μέγιστη τιμή μιας στήλης.
- **Min()**, ελάχιστη τιμή μιας στήλης.
- **Sum()**, άθροισμα τιμών μιας στήλης.

Ακολουθούν παραδείγματα.

```
Select Sum(Poso)
    From Clients;
Select Count(Name)
    From Clients;
```

Η συνάρτηση Count() αγνοεί τις τιμές Null αλλά μετρά τις επαναλαμβανόμενες τιμές, οπότε πρέπει να την χρησιμοποιήσουμε σε συνδυασμό με την δήλωση Distinct για να μπορέσουμε να μετρήσουμε τις διακριτές τιμές ενός πεδίου.

Για να κάνουμε συνδυασμό των δύο παραπάνω τύπων συναρτήσεων, πρέπει να δημιουργήσουμε ένα υποερώτημα (subquery) για να εξάγουμε μια τιμή από μια συνάρτηση ομάδας και να την χρησιμοποιήσουμε για σύγκριση στο βασικό ερώτημα.

Ακολουθεί ένα παράδειγμα.

```
Select Name, Address, Poso
    From Clients
        Where Poso >
          (Select Avg(Poso) From Clients);
```

Μάθημα 12 - Η Δήλωση Group By

Με την δήλωση **Group By** είναι δυνατό να ομαδοποιήσουμε τις γραμμές ενός πίνακα σε ομάδες, έτσι ώστε οι γραμμές που υπάρχουν σε κάθε ομάδα να έχουν την ίδια τιμή για κάποια στήλη του πίνακα. Μπορούμε να ομαδοποιήσουμε και με βάση περισσότερες από μία στήλες και πρέπει να έχουμε υπόψη μας ότι η δήλωση Group By πρέπει να γράφεται τελευταία.

Ακολουθεί ένα παράδειγμα.

```
Select OrderDate, Sum(OrderValue)
    From Orders
        Group By OrderDate;
```

Θα εμφανισθούν οι ημερομηνίες καθώς και το άθροισμα της αξίας των παραγγελιών ανά ημερομηνία. Με την ομαδοποίηση γίνεται αναγκαστικά και ταξινόμηση των ομάδων των γραμμών του πίνακα με βάση την στήλη που έχουμε χρησιμοποιήσει στην δήλωση Group By.

Μάθημα 13 - Η Δήλωση Having

Για να μπορέσουμε να επιλέξουμε (απομονώσουμε) κάποιες γραμμές ενός πίνακα όταν κάνουμε ομαδοποίηση σ' αυτόν, χρησιμοποιούμε την δήλωση **Having**. Η δήλωση Having εφαρμόζεται στις γραμμές μιας ομάδας.

Ακολουθεί ένα παράδειγμα.

```
Select OrderDate, Count(*), Sum(OrderValue), Avg(OrderValue)
        From Orders
            Group By OrderDate
                Having Avg(OrderValue) > 50000;
```

Θα εμφανισθούν το πλήθος των παραγγελιών, το άθροισμα της αξίας των παραγγελιών καθώς και ο μέσος όρος της αξίας των παραγγελιών αλλά μόνο για τις ημερομηνίες εκείνες που οι παραγγελίες είχαν μέσο όρο μεγαλύτερο από 50.000.

Μάθημα 14 - Οι Συναρτήσεις Χαρακτήρων

Η Oracle διαθέτει τις εξής συναρτήσεις για τις αλφαριθμητικές τιμές (strings) :

- InitCap(), Μετατρέπει σε κεφαλαίο το πρώτο γράμμα.
- Instr(), Βρίσκει την θέση ενός χαρακτήρα σ' ένα πεδίο.
- Length(), Εμφανίζει το πλήθος των χαρακτήρων ενός πεδίου (μήκος).
- Lower(), Μετατρέπει τα γράμματα σε πεζά.
- Upper(), Μετατρέπει τα γράμματα σε κεφαλαία.
- Substr(), Εμφανίζει (απομονώνει) ένα κομμάτι ενός string από μια συγκεκριμένη θέση και με δεδομένο μήκος χαρακτήρων.

Μάθημα 15 - Η Τιμή Null

Η τιμή **Null** αποτελεί μια ειδική περίπτωση. Δεν είναι ίση με το 0 ή με το κενό, απλά δηλώνει μια ανύπαρκτή τιμή. Μπορούμε να χρησιμοποιήσουμε τις φράσεις **Is Null** και **Is Not Null** σε δηλώσεις Where για να επιλέξουμε γραμμές που δεν έχουν τιμές ή που έχουν κάποιες τιμές σε κάποια συγκεκριμένη στήλη.

Οταν χρησιμοποιούμε την δήλωση Order By, εμφανίζονται πάντα πρώτες οι γραμμές που έχουν τιμές Null, άσχετα με την ταξινόμηση που έχουμε ορίσει. Οι συναρτήσεις ομάδας, όπως είναι η Count() και η Sum(), δεν λαμβάνουν υπόψη τους τις τιμές Null.

Αν χρησιμοποιηθεί ένα πεδίο που έχει τιμή Null σε μια έκφραση υπολογισμού, τότε το αποτέλεσμα της έκφρασης θα είναι πάντα Null. Για να το αποφύγουμε αυτό, που μπορεί να μας δημιουργήσει και προβλήματα, μπορούμε να χρησιμοποιήσουμε την συνάρτηση **Nvl()**, η οποία δέχεται σαν ορίσματα το όνομα ενός πεδίου και μια τιμή που θα χρησιμοποιηθεί όταν η τιμή του πεδίου είναι ίση με Null. Συνήθως, αυτό το δεύτερο όρισμα το θέτουμε ίσο με 0.

Ακολουθούν παραδείγματα.

```
Select Name, Address, Poso
      From Clients
            Where Poso Is Null;
Select OrderId, OrderClientId, OrderDate, OrderValue
      From Orders
            Where OrderDate Is Not Null;
Select Name, Address, Nvl(Poso, 0)
      From Clients
```

Μάθημα 16 - Τα Υποερωτήματα (Subqueries)

Τα **υποερωτήματα (subqueries)** είναι ερωτήματα που τα χρησιμοποιούμε στην δήλωση Where άλλων ερωτημάτων. Τα χρησιμοποιούμε όταν πρέπει να δημιουργήσουμε ερωτήματα τα οποία χρειάζονται στοιχεία που προκύπτουν από την εκτέλεση άλλων ερωτημάτων. Το αποτέλεσμα από την χρήση ενός υποερωτήματος δεν εμφανίζεται αλλά βοηθάει στην εκτέλεση του κύριου ερωτήματος. Το υποερώτημα επιστρέφει πάντα μία μόνο τιμή.

Για παράδειγμα, αν θέλουμε να εμφανίσουμε τους πελάτες που έχουν μέσο όρο στο πεδίο Poso μεγαλύτερο από τον μέσο όρο όλων των πελατών, θα πρέπει να δημιουργήσουμε ένα υποερώτημα που να υπολογίζει τον μέσο όρο όλων των πελατών και να το χρησιμοποιήσουμε μέσα σε παρενθέσεις στην δήλωση Where του βασικού ερωτήματος.

Ακολουθούν παραδείγματα.

```
Select Name, Address, Poso
   From Clients
    Where Poso > (Select Avg(Poso) From Clients);
Select ClientId, Name, Address, Poso
  From Clients
    Where ClientId = (Select OrderClientId From Orders Where OrderId =
'100');
```

Μάθημα 17 - Οι Δηλώσεις Union, Intersection και Minus

Με τις δηλώσεις αυτές μπορούμε να κάνουμε Ενωση (Union), Τομή (Intersection) και Διαφορά (Minus) στις τιμές που επιλέγονται με βάση κάποιο υποερώτημα από διάφορους πίνακες. Οι λειτουργίες που κάνουν είναι οι εξής :

Union, επιλέγει όλες τις διαφορετικές τιμές και από τα δύο ερωτήματα.

Intersect, επιλέγει όλες τις κοινές τιμές και από τα δύο ερωτήματα.

Minus, επιλέγει τις γραμμές που βρήκε το προηγούμενο ερώτημα αλλά όχι αυτές που βρήκε το επόμενο.

Ακολουθεί ένα παράδειγμα.

```
Select Desc1, Item1, Poso1
    From Table1
    Union
    Select Desc2, Item2, Poso2
        From Table2
```

Μάθημα 18 - Τα Συσχετισμένα Υποερωτήματα

Το χαρακτηριστικό γνώρισμα αυτών των υποερωτημάτων είναι ότι ο πίνακας στον οποίο δημιουργείται το ερώτημα αναφέρεται και με ψευδώνυμο (alias) μέσα στο κύριο ερώτημα.

Ακολουθεί ένα παράδειγμα.

```
Select Name, Address, Poso
    From Clients C
        Where Poso > (Select Avg(Poso) From Clients Where C.Poli = Poli);
```

Εμφανίζεται ο μέσος όρος του πεδίου Poso για τους πελάτες κάθε διαφορετικής πόλης. Το υποερώτημα υπολογίζει τον μέσο όρο για κάθε πόλη και μετά χρησιμοποιείται αυτό το αποτέλεσμα για να εμφανισθούν τα στοιχεία των πελατών που περιέχουν ποσό μεγαλύτερο από τον υπολογισθέντα μέσο όρο.

Μάθημα 19 - Δημιουργία Πίνακα

Για να δημιουργήσουμε έναν πίνακα (table), δηλώνουμε το όνομα του πίνακα, τα πεδία (στήλες) που θα περιέχει, τον τύπο δεδομένων των πεδίων καθώς και το μήκος του κάθε πεδίου, με την εξής σύνταξη :

```
Create Table όνομα_πίνακα
    (όνομα_στήλης1, όνομα_στήλης2, ...);
```

Ακολουθεί ένα παράδειγμα.

```
Create Table Friends
    (FriendId char(5) Not Null, Name char(20) Not Null, Phone char(10),
    BDate date, Poso number(10, 2));
```

Οι τύποι δεδομένων των πεδίων ενός πίνακα είναι οι εξής :

- Char, περιέχει χαρακτήρες, πρέπει να δηλώσουμε το μήκος του πεδίου και δεν μπορεί να περιέχει πάνω από 240 χαρακτήρες.
- Number, περιέχει αριθμούς και είναι προαιρετικό να δηλώσουμε το συνολικό μήκος των ψηφίων καθώς κι αυτό των δεκαδικών.
- Date, περιέχει ημερομηνίες.
- Long, περιέχει χαρακτήρες αλλά με μέγιστο μήκος 65.535 χαρακτήρων και ένα μόνο πεδίο μπορεί να είναι τύπου Long σ' έναν πίνακα.

Για ένα πεδίο μπορούμε να δηλώσουμε να είναι του τύπου Not Null, που σημαίνει ότι θα πρέπει υποχρεωτικά να υπάρχει κάποια τιμή για το πεδίο αυτό.

Μάθημα 20 - Εισαγωγή Νέων Πεδίων σε Πίνακα

Για να εισάγουμε νέα πεδία σ' έναν ήδη υπάρχοντα πίνακα, χρησιμοποιούμε την εξής σύνταξη :

```
Alter Table όνομα_πίνακα
    Add (ορισμοί_πινάκων);
```

Ακολουθούν παραδείγματα.

```
Alter Table Clients
    Add (City Char(10));
```

```
Alter Table Friends
    Add (Mobile Char(10));
```

Τα νέα πεδία προστίθενται στα δεξιά της τελευταίας στήλης του πίνακα και δεν μπορούμε να τα δηλώσουμε σαν Not Null.

Μάθημα 21 - Εισαγωγή Νέων Γραμμών σε Πίνακα

Για να καταχωρήσουμε νέες γραμμές (εγγραφές) σ' έναν πίνακα, χρησιμοποιούμε την εξής σύνταξη :

```
Insert Into όνομα_πίνακα
    Values (λίστα_τιμών);
```

Ακολουθεί ένα παράδειγμα.

```
Insert Into Clients
    Values ('100', 'Παπαδόπουλος', 'Π. Μελά 100', '03850-22222', 1000.00);
```

Η λίστα_τιμών πρέπει να περιέχει τις τιμές των πεδίων του πίνακα με την ίδια σειρά που δηλώθηκαν κατά την δημιουργία του πίνακα ή όπως εμφανίζονται αν δώσουμε την εξής εντολή :

```
Select * From όνομα_πίνακα;
```

Μπορούμε να καταχωρήσουμε νέες εγραφές σ' έναν πίνακα, επιλέγοντας με υποερώτημα (subquery) εγγραφές από έναν άλλον πίνακα, ως εξής :

```
Create Table Friends
    (FriendId char(5) Not Null, Name char(20) Not Null, Phone char(10),
BDate date, Poso number(10, 2));
Insert Into Friends (FriendId, Name, Phone, BDate, Poso) As
    (Select ClientId, Name, Phone, Date, Poso
        From Clients
            Where Poso > 100000);
```

Μάθημα 22 - Τροποποίηση των Τιμών Πεδίου Πίνακα

Για να αλλάξουμε τις τιμές των πεδίων ενός πίνακα, δίνουμε την εξής εντολή :

```
Update όνομα_πίνακα
    Set πεδίο1 = τιμή1, πεδίο2 = τιμή2, ...
        Where συνθήκη;
```

Σε κάθε εγγραφή που ικανοποιεί την συνθήκη, οι τιμές των πεδίων που υπάρχουν στην δήλωση Set αντικαθίστανται από την νέα τιμή.

Ακολουθούν παραδείγματα.

```
Update Clients
    Set Poso = 200000
        Where Poso = 100000;
Update Clients
    Set Poso = Poso * 1.10
        Where Name In (Select Name From Friends);
```

Μάθημα 23 - Διαγραφή Γραμμών Πίνακα

Για να διαγράψουμε τις εγγραφές ενός πίνακα που ικανοποιούν κάποια συνθήκη, χρησιμοποιούμε την εξής σύνταξη :

```
Delete From όνομα_πίνακα
    Where συνθήκη;
```

Αν παραλείψουμε την δήλωση Where, θα διαγραφούν όλες οι εγγραφές του πίνακα.

Ακολουθούν παραδείγματα.

```
Delete From Clients
    Where Poso < 50000;
Delete From Clients
    Where Name In (Select Name From Friends);
```

Μάθημα 24 - Τροποποίηση Πεδίου Πίνακα

Για να αλλάξουμε τα χαρακτηριστικά κάποιας στήλης ενός πίνακα, δίνουμε την εξής εντολή :

```
Alter Table όνομα_πίνακα
    Modify (ορισμοί_στηλών);
```

Μέσα στις παρενθέσεις γράφουμε τα νέα χαρακτηριστικά της στήλης, όπως μήκος, τύπος κ.ά., αλλα δεν μπορούμε να μικραίνουμε το μήκος της στήλης.

Ακολουθεί ένα παράδειγμα.

```
Alter Table Clients
    Modify (Name char(30));
```

Πρέπει να έχουμε υπόψη μας ότι όταν κάνουμε αλλαγές στις τιμές των εγγραφών ή στα χαρακτηριστικά των πεδίων ενός πίνακα, οι αλλαγές αυτές δεν αποθηκεύονται στην βάση δεδομένων και δεν εμφανίζονται έτσι σ' έναν χρήστη της βάσης δεδομένων. Για να γίνει αυτό, πρέπει να δώσουμε την εντολή **Commit** και η Oracle να απαντήσει με commit complete.

Αν δεν θέλουμε να καταχωρηθεί κάποια αλλαγή στην βάση δεδομένων, πρέπει αντί για την εντολή Commit να δώσουμε την εντολή **RollBack** και η Oracle να απαντήσει με rollback complete.

Μάθημα 25 - Αντιγραφή Πίνακα

Αν θέλουμε να αντιγράψουμε έναν πίνακα μ' ένα άλλο όνομα αλλά και να διατηρήσουμε τον αρχικό πίνακα, δηλ. η γνωστή μας λειτουργία Save As, υπάρχουν οι εξής δύο τρόποι για να το κάνουμε :

- Εξωτερικά, με το utility *EXP*.
- Εσωτερικά, με την εντολή *Create Table όνομα_πίνακα As υποερώτημα.*

Μάθημα 26 - Διαγραφή Πίνακα

Για να διαγράψουμε έναν πίνακα, δίνουμε την εξής εντολή :

```
Drop Table όνομα_πίνακα;
```

Ακολουθεί ένα παράδειγμα.

```
Drop Table Friends;
```

Πρέπει να έχουμε υπόψη μας ότι μετά την εκτέλεση της εντολής Drop γίνεται αυτόματα και Commit και έτσι δεν μπορούμε να κάνουμε αναίρεση της διαγραφής ενός πίνακα.

Μάθημα 27 - Οι Δείκτες (Indexes)

Με την χρήση των **δεικτών (indexes**), οι οποίοι είναι παρόμοιοι στην δομή με τους δείκτες άλλων γλωσσών, μπορούμε να επιταχύνουμε την αναζήτηση πληροφοριών. Η αναζήτηση βασίζεται μόνο σε στήλη που έχει δηλωθεί σαν δείκτης (index). Για να δημιουργήσουμε έναν δείκτη, δίνουμε την εξής εντολή :

```
Create Index όνομα_δείκτη
    On πίνακας(στήλη);
```

Ακολουθεί ένα παράδειγμα.

```
Create Index ClId
    On Clients(ClientId);
```

Αν εκτελέσουμε ένα ερώτημα και η δήλωση Where βασίζεται σε μια στήλη που έχουμε δηλώσει σαν index, τότε η Oracle θα κάνει τις αναζητήσεις της στον πίνακα δείκτη (index table) για να έχει έτσι ταχύτερη πρόσβαση στα δεδομένα. Αν έχουμε χρησιμοποιήσει την δήλωση **Unique** πριν από το όνομα του δείκτη, τότε η SQL* θα ελέγχει ώστε οι τιμές της στήλης που έχει δηλωθεί σαν δείκτης να είναι μοναδικές.

Ακολουθεί ένα παράδειγμα.

```
Create Unique Index ClId
    On Clients(ClientId);
```

Για να διαγράψουμε έναν δείκτη, δίνουμε την εξής εντολή :

```
Drop Index Clid;
```

Ο δείκτης παύει να είναι ενεργός ενώ ο πίνακας που τον περιέχει δεν παθαίνει απολύτως τίποτα.

Μάθημα 28 - Οι Οψεις (Views)

Μια **όψη (view)** είναι σαν ένα φίλτρο μέσα από το οποίο μπορούμε να έχουμε μια διαφορετική μορφή των στοιχείων μιας βάσης δεδομένων. Οι όψεις χρησιμοποιούνται όπως και οι πίνακες, αλλά δεν καταλαμβάνουν χώρο στον δίσκο και εξαρτώνται από τον ή τους πίνακες στους βασίζεται μια όψη. Γι' αυτό αποκαλούνται και εικονικοί πίνακες (virtual tables). Με τις όψεις μπορούμε να απομονώσουμε κάποια στοιχεία μιας βάσης δεδομένων.

Για να δημιουργήσουμε μια όψη, η σύνταξη είναι η εξής :

```
Create View όνομα_όψης
    As ερώτημα
```

Κατά την δημιουργία μιας όψης με βάση ένα ερώτημα, δεν μπορούμε να χρησιμοποιήσουμε την δήλωση Order By. Αφού έχουμε δημιουργήσει μια όψη, θα μπορεί να χρησιμοποιηθεί σαν ένας οποιοσδήποτε πίνακας και κάθε αλλαγή που κάνουμε στον πίνακα θα φαίνεται και σ' όλες τις όψεις που εξαρτώνται απ' αυτόν τον πίνακα.

Ακολουθεί ένα παράδειγμα μιας όψης που εμφανίζει (απομονώνει) από τον πίνακα πελατών εκείνους που το όνομά τους περιέχει τους χαρακτήρες 'ίδη' :

```
Create View ClientPodiac
  As Select ClientId, Name, Address, Phone, Poso
     From Clients
        Where Name Like '%ίδη%';
```

Μάθημα 29 - Μορφοποίηση Εκτυπώσεων με την SQL*Plus

Στην SQL*Plus έχουμε την δυνατότητα να παρουσιάζουμε τα αποτελέσματα της επεξεργασίας ενός πίνακα μορφοποιημένα, με τίτλους, επικεφαλίδες, αρίθμηση σελίδων κ.ά. Με την χρήση της SQL*Plus θα μπορούμε να κάνουμε τα εξής :

- Να τοποθετήσουμε τίτλους στην αρχή και στο τέλος μιας εκτύπωσης με τις εντολές TTitle και BTitle αντίστοιχα.
- Να ορίσουμε επικεφαλίδες στις στήλες (columns) με την εντολή Heading.
- Να ομαδοποιήσουμε την εμφάνιση των γραμμών με την εντολή Break.
- Να κάνουμε υπολογισμούς για μερικά και για γενικά σύνολο με την εντολή Compute.

Η παρουσίαση των αποτελεσμάτων των εντολών της SQL σε καθαρή και ευανάγνωστη μορφή λέγεται Report (αναφορά ή έκθεση) και η διαδικασία δημιουργίας μιας Report αποκαλείται Report Generation.

Για να ορίσουμε επικεφαλίδες για τις στήλες μιας αναφοράς, η σύνταξη είναι η εξής :

Column όνομα_στήλης Heading επικεφαλίδα;

Ακολουθεί ένα παράδειγμα :

Column Name Heading 'Ονομα|Πελάτη';

Η χρήση του χαρακτήρα | έχει σαν αποτέλεσμα να εμφανίζεται η επικεφαλίδα σε δύο γραμμές.

Με την δήλωση **Format** μπορούμε να μορφοποιήσουμε την εμφάνιση του περιεχομένου μιας στήλης και με την δήλωση **Like** να αντιγράψουμε μια ήδη υπάρχουσα μορφοποίηση μιας στήλης.

Με την δήλωση **TTitle** μπορούμε να δημιουργήσουμε τίτλο στην κορυφή μιας σελίδας (κεφαλίδα) και με την δήλωση **BTitle** υπότιτλο στην βάση μιας σελίδας (υποσέλιδο). Με τις δηλώσεις Center, Left και Right μπορούμε να ορίσουμε την στοίχιση των τίτλων.

Ακολουθούν παραδείγματα.

TTitle Center 'Πελάτες Εταιρείας ABC';
BTitle Right 'Ακαδημίας 100 - 101 52 Αθήνα';

Με την δήλωση **Break** μπορούμε να ομαδοποιήσουμε (group) τις γραμμές με βάση την τιμή κάποιας στήλης. Η σύνταξή της είναι ως εξής :

Break On όνομα_στήλης;

Ακολουθεί ένα παράδειγμα.

Break On Poli Skip 1;

Με την δήλωση Skip 1 εμφανίζεται μια κενή γραμμή ανάμεσα στις ομάδες.

Με την δήλωση **Compute** μπορούμε να κάνουμε υπολογισμούς στις τιμές κάποιων στηλών για τις γραμμές μιας ομάδας. Η σύνταξή της είναι η εξής :

Compute Sum Of στήλη1, στήλη2, ... On στήλη_ομαδοποίησης;

Κάθε φορά που αλλάζει η τιμή της στήλης που χρησιμοποιείται για την ομαδοποίηση, υπολογίζεται το ενδιάμεσο άθροισμα των τιμών για τα μέλη της ομάδας. Αντί για την εντολή Sum μπορούμε να χρησιμοποιήσουμε και τις εντολές Min, Max, Avg και Count.

Μάθημα 30 - Η Εντολή Grant

Με την εντολή **Grant** μπορούμε να δώσουμε δικαιώματα χρήσης ή προνόμια (privileges) σ' άλλους χρήστες της Oracle. Ο δημιουργός ενός πίνακα γίνεται αυτόματα και ιδιοκτήτης του και έχει το δικαίωμα να παραχωρήσει άδειες (προνόμια) χρήσης του πίνακα με την εντολή Grant, η οποία έχει την εξής σύνταξη :

Grant προνόμια
    On αντικείμενο
        To χρήστης;

Οι άδειες που μπορούμε να εκχωρήσουμε είναι οι εξής : Select, Insert, Update, Delete, Alter, Index και Cluster, ενώ με την επιλογή All μπορούμε να εκχωρήσουμε όλες τις παραπάνω άδειες.

Ακολουθεί ένα παράδειγμα.

Grant Insert, Update, Delete
    On Clients
        To user01, user02

```
With Grant Option;
```

Η δήλωση **With Grant Option** σημαίνει ότι ο χρήστης στον οποίο παραχωρούμε τα δικαιώματα, μπορεί με την σειρά του να τα παραχωρήσει και σ' έναν τρίτο χρήστη. Μπορούμε να εκχωρήσουμε δικαιώματα και σε όψεις (views) εκτός από πίνακες (tables), για να μπορούμε έτσι να περιορίσουμε την πρόσβαση των χρηστών σε συγκεκριμένα στοιχεία ενός πίνακα.

Μάθημα 31 - Η Εντολή Revoke

Με την εντολή Revoke μπορούμε να αφαιρέσουμε (ανακαλέσουμε) τις άδειες που έχουμε εκχωρήσει σ' έναν χρήστη με την εντολή Grant. Η σύνταξη της εντολής Revoke είναι η εξής :

```
Revoke προνόμια
    On πίνακας/όψη
        From χρήστης;
```

Μάθημα 32 - Η Δήλωση Public

Με την δήλωση Public μπορούμε να παραχωρήσουμε άδειες σ' όλους τους χρήστες μιας βάσης δεδομένων, ακόμη και σ' αυτούς που θα προστεθούν μετά την παραχώρηση της άδειας.

Ακολουθούν παραδείγματα.

```
Grant All
    On Clients
        To Public;
Grant Select, Insert
    On Friends
        To user01
            With Grant Option;
Revoke All
    On Clients
        From user02;
```

Εισαγωγή στην SQL

Εισαγωγή στην SQL

Για να μπορέσουμε να δημιουργήσουμε και να διαχειριστούμε μια βάση δεδομένων, μπορούμε να χρησιμοποιήσουμε ειδικές γλώσσες προγραμματισμού, τις λεγόμενες γλώσσες ερωταπαντήσεων (query languages). Είναι γλώσσες μη διαδικαστικές, τέταρτης γενιάς (4th generation languages). Εμείς απλά διατυπώνουμε με απλές και κατανοητές εντολές το τι πληροφορίες ζητάμε και το ΣΔΒΔ (Σύστημα Διαχείρισης Βάσεων Δεδομένων) αναλαμβάνει να μας απαντήσει. Η SQL (Structured Query Language, δηλ. Δομημένη Γλώσσα Ερωταπαντήσεων) είναι σήμερα η πιο δημοφιλής και πιο διαδεδομένη γλώσσα ανάπτυξης και διαχείρισης σχεσιακών βάσεων δεδομένων.

Η SQL αποτελείται από εντολές με τα ορίσματά τους, τις οποίες μπορούμε να χρησιμοποιήσουμε με συγκεκριμένους κανόνες σύνταξης για να πάρουμε τα αποτελέσματα που θέλουμε. Με την SQL μπορούμε να δημιουργήσουμε μια βάση δεδομένων και τους πίνακές της με τα αντίστοιχα πεδία, να καταχωρήσουμε δεδομένα στους πίνακες, να τροποποιήσουμε και να διαγράψουμε τα δεδομένα αυτά, να αλλάξουμε τη δομή των πινάκων με προσθήκη και διαγραφή πεδίων και να εμφανίσουμε πληροφορίες (συνδυασμούς από δεδομένα).

Η SQL έχει δύο τμήματα :

- Τη Γλώσσα Ορισμού Δεδομένων (DDL, Data Definition Language), η οποία περιέχει τις απαραίτητες εντολές για τον ορισμό και την τροποποίηση του σχεσιακού σχήματος καθώς και για τη δημιουργία, την τροποποίηση και τη διαγραφή σχέσεων. Περιέχει ακόμη τις εντολές δημιουργίας και επεξεργασίας όψεων και ορισμού περιορισμών ακεραιότητας.
- Τη Γλώσσα Χειρισμού Δεδομένων (DML, Data Manipulation Language), η οποία περιέχει τις απαραίτητες εντολές για την εμφάνιση (αναζήτηση) δεδομένων καθώς και για την καταχώρηση, τροποποίηση και διαγραφή των εγγραφών (πλειάδων) μιας σχέσης.
- Τέλος, περιέχει εντολές για τον ορισμό και την επεξεργασία συναλλαγών (transactions).

Οι Τύποι Δεδομένων της SQL

Η SQL υποστηρίζει τους εξής τύπους δεδομένων για τα πεδία μιας σχέσης :

- **char(n)**, ένα αλφαριθμητικό (string) με n ακριβώς χαρακτήρες.
- **varchar(n)**, ένα αλφαριθμητικό (string) με μεταβλητό μήκος και με n το πολύ χαρακτήρες.
- **int**, ακέραιος αριθμός.
- **smallint**, ακέραιος αριθμός με μικρές τιμές.
- **Numeric(p, d)**, αριθμός με p ψηφία, από τα οποία τα d είναι δεκαδικά.
- **real**, αριθμός κινητής υποδιαστολής απλής ακρίβειας.
- **double precision**, αριθμός κινητής υποδιαστολής διπλής ακρίβειας.
- **float(n)**, αριθμός κινητής υποδιαστολής με ακρίβεια n ψηφίων.
- **date**, ημερομηνία (ημέρα, μήνας, έτος).
- **time**, ώρα (ώρα, λεπτά, δευτερόλεπτα).

Μπορούμε να δημιουργήσουμε και δικούς μας τύπους δεδομένων με την εντολή **create domain**, ως εξής :

```
Create domain Name char(20);
```

Ο νέος τύπος δεδομένων Name θα μπορεί να χρησιμοποιηθεί εφεξής όπως και οι υπόλοιποι τύποι δεδομένων της SQL. Απλά θα περιέχει strings με μήκος 20 χαρακτήρων.

Δημιουργία, Τροποποίηση και Διαγραφή Σχέσεων

Για να δημιουργήσουμε μια σχέση (πίνακα) χρησιμοποιούμε την εντολή **create table**. Για παράδειγμα, για να δημιουργήσουμε τη σχέση Αθλητής δίνουμε την εξής εντολή :

```
Create table ΑΘΛΗΤΗΣ
    (Κωδικός_Αθλητή      integer not null,
    Επώνυμο              char(20) not null,
    Όνομα                char(15),
    Ρεκόρ                numeric(4, 2),
    Ημνία_Γέννησης       date,
    Κωδικός_Συλλόγου     integer not null,
    primary key (Κωδικός_Αθλητή),
    check(Κωδικός_Αθλητή >= 100));
```

Μετά την εντολή create table γράφουμε το όνομα της σχέσης (πίνακα) και ακολουθεί μια παρένθεση που περιέχει τα ονόματα των πεδίων με τους τύπους δεδομένων τους. Η δήλωση **not null** σημαίνει ότι το συγκεκριμένο πεδίο θα πρέπει να έχει οπωσδήποτε κάποια τιμή. Προσοχή : άλλο πράγμα είναι η τιμή 0 ή η τιμή του κενού χαρακτήρα (space), που είναι κάποια τιμή, και άλλο πράγμα είναι η μη ύπαρξη τιμής.

Με τη δήλωση **primary key** ορίζουμε το πεδίο κλειδί (πρωτεύον κλειδί) της σχέσης. Αν, αργότερα κατά την εισαγωγή τιμών, δώσουμε μια τιμή null στο πρωτεύον κλειδί ή μια τιμή που ήδη υπάρχει σε κάποια άλλη πλειάδα, τότε θα εμφανισθεί ένα μήνυμα λάθους.

Με τον όρο **check** μπορούμε να δηλώσουμε μια συνθήκη για την περιοχή των τιμών ενός πεδίου. Είδαμε τη χρήση του όρου check για να ελέγξει αν η τιμή ενός κωδικού είναι μεγαλύτερη ή ίση από 100. Μπορούμε να χρησιμοποιήσουμε τον όρο check και για να ελέγξουμε αν η τιμή ενός πεδίου ανήκει σ' ένα προκαθορισμένο σύνολο τιμών, ως εξής :

```
check (Νομός in ('Γρεβενών', 'Καστοριάς', 'Κοζάνης', 'Φλώρινας'))
```

Εδώ ελέγχουμε το αν η τιμή του πεδίου Νομός θα ανήκει σε μία από τέσσερις συγκεκριμένες τιμές.

Για να τροποποιήσουμε μια σχέση και να προσθέσουμε ένα πεδίο, δίνουμε την εντολή alter table και το όρισμα add, ως εξής :

```
Alter table ΑΘΛΗΤΗΣ add (Τηλέφωνο char(10));
```

Τα καινούργια πεδία που προστίθενται σε μια σχέση, έχουν αρχικά τιμές ίσες με null. Για να τροποποιήσουμε μια σχέση και να διαγράψουμε ένα πεδίο, δίνουμε την εντολή alter table και το όρισμα drop, ως εξής :

```
Alter table ΑΘΛΗΤΗΣ drop Τηλέφωνο;
```

Για να διαγράψουμε τελείως μια σχέση από τη βάση δεδομένων στην οποία ανήκει, μαζί με τις τιμές των εγγραφών της, δίνουμε την εντολή drop table, ως εξής :

```
Drop table ΑΘΛΗΤΗΣ;
```

Υπάρχει και η εντολή delete from ΑΘΛΗΤΗΣ, με την οποία μπορούμε να διαγράψουμε τις τιμές (περιεχόμενα, εγγραφές) μιας σχέσης, αλλά όχι την ίδια τη σχέση.

Η Ακεραιότητα Αναφορών

Με τον όρο **ακεραιότητα αναφορών (referential integrity)** αναφερόμαστε στην ιδιότητα όπου οι τιμές ορισμένων πεδίων μιας σχέσης υπάρχουν και σε αντίστοιχα πεδία σε κάποια άλλη σχέση. Για παράδειγμα, αν έχουμε τις σχέσεις ΑΘΛΗΤΗΣ και ΑΓΩΝΑΣ,

τότε η συσχέτιση μεταξύ τους υλοποιείται με την τρίτη σχέση ΣΥΜΜΕΤΟΧΗ, όπου εμφανίζεται το ποιοι αθλητές συμμετείχαν σε ποιους αγώνες και φυσικά τι επίδοση κάνανε.

Η ακεραιότητα αναφορών έρχεται να επιλύσει το πρόβλημα της διαγραφής μιας πλειάδας από τη σχέση ΑΓΩΝΑΣ, οπότε οι αθλητές που συμμετείχαν στον συγκεκριμένο αγώνα θα φαίνονται ξεκρέμαστοι, καθώς επίσης και το πρόβλημα της τροποποίησης του πεδίου κλειδιού (Κωδικός_Αγώνα) μιας πλειάδας από τη σχέση ΑΓΩΝΑΣ, οπότε και πάλι οι αθλητές που συμμετείχαν στον συγκεκριμένο αγώνα θα φαίνονται ξεκρέμαστοι.

Για να αποφύγουμε τέτοιες καταστάσεις, καταφεύγουμε στην έννοια του **ξένου κλειδιού (foreign key)**, όπου στη σχέση ΣΥΜΜΕΤΟΧΗ, το πεδίο Κωδικός_Αθλητή είναι ξένο κλειδί αλλά και πρωτεύον κλειδί στη σχέση ΑΘΛΗΤΗΣ, ενώ το πεδίο Κωδικός_Αγώνα είναι ξένο κλειδί στη σχέση ΣΥΜΜΕΤΟΧΗ αλλά και πρωτεύον κλειδί στη σχέση ΑΓΩΝΑΣ.

Με τη δήλωση της ακεραιότητας αναφορών, αν διαγράψουμε μια πλειάδα ενός πεδίου που είναι πρωτεύον κλειδί σε κάποια σχέση, τότε θα διαγραφούν και όλες οι αντίστοιχες εγγραφές (πλειάδες) στη σχέση όπου το ίδιο πεδίο είναι ξένο κλειδί. Αυτό σημαίνει πρακτικά ότι αν διαγράψουμε έναν αθλητή από τη σχέση ΑΘΛΗΤΗΣ, τότε θα διαγραφούν και όλες οι συμμετοχές του σε αγώνες από τη σχέση ΣΥΜΜΕΤΟΧΗ.

Επίσης, αν τροποποιήσουμε την τιμή ενός πεδίου που είναι πρωτεύον κλειδί σε κάποια σχέση, τότε θα ενημερωθούν αυτόματα και όλες οι αντίστοιχες εγγραφές (πλειάδες) στη σχέση όπου το ίδιο πεδίο είναι ξένο κλειδί. Αυτό σημαίνει πρακτικά ότι αν τροποποιήσουμε τον κωδικό ενός αγώνα από τη σχέση ΑΓΩΝΑΣ, τότε θα ενημερωθούν αυτόματα και όλες οι συμμετοχές των αθλητών στον αγώνα αυτό που υπάρχουν στη σχέση ΣΥΜΜΕΤΟΧΗ.

Για να δηλώσουμε ένα ξένο κλειδί και την αναφορά του, χρησιμοποιούμε τους όρους **foreign key** και **references** όταν δημιουργούμε μια σχέση, ως εξής :

```
Create table ΣΥΜΜΕΤΟΧΗ
  Κωδικός_Αθλητή      integer not null,
  Κωδικός_Αγώνα       integer not null,
  Επίδοση             integer,
  primary key(Κωδικός_Αθλητή, Κωδικός_Αγώνα),
  foreign key(Κωδικός_Αθλητή) references ΑΘΛΗΤΗΣ,
  foreign key(Κωδικός_Αγώνα) references ΑΓΩΝΑΣ);
```

Σύμφωνα με τον παραπάνω τρόπο δημιουργίας της σχέσης ΣΥΜΜΕΤΟΧΗ, αν επιχειρήσουμε να διαγράψουμε έναν αθλητή που έχει συμμετάσχει σε κάποιον αγώνα ή έναν αγώνα στον οποίον έχουν συμμετάσχει κάποιοι αθλητές, το σύστημα δεν θα μας αφήσει να το κάνουμε γιατί έχουμε παραβίαση της ακεραιότητας αναφοράς. Το ίδιο πρόβλημα θα παρουσιασθεί και αν προσπαθήσουμε να τροποποιήσουμε τον κωδικό ενός αθλητή που έχει συμμετάσχει σε κάποιον αγώνα ή τον κωδικό ενός αγώνα στον οποίον έχουν συμμετάσχει κάποιοι αθλητές.

Για να μπορέσουμε να αντιμετωπίσουμε την ανάγκη διαγραφής μιας πλειάδας ή τροποποίησης του πεδίου κλειδιού χωρίς να έχουμε παραβίαση της ακεραιότητας αναφοράς, χρησιμοποιούμε τους όρους **on delete cascade** και **on update cascade** αντίστοιχα, ως εξής :

```
Create table ΣΥΜΜΕΤΟΧΗ
  Κωδικός_Αθλητή      integer not null,
  Κωδικός_Αγώνα               integer not null,
  Επίδοση                     integer,
  primary key(Κωδικός_Αθλητή, Κωδικός_Αγώνα),
  foreign key(Κωδικός_Αθλητή) references ΑΘΛΗΤΗΣ
                              on delete cascade
```

```
                                    on update cascade,
    foreign key(Κωδικός_Αγώνα) references ΑΓΩΝΑΣ
                                    on delete cascade
                                    on update cascade);
```

Οι Όψεις (Views)

Η όψη (view) είναι μια σχέση μιας βάσης δεδομένων που δεν έχει δημιουργηθεί με κάποια εντολή create table. Με τις όψεις μπορούμε να εμφανίζουμε ορισμένα μόνο πεδία μιας σχέσης ή και κάποιες άλλες πληροφορίες που αν δεν ήταν οι όψεις θα έπρεπε να δίνουμε κάθε φορά πολύπλοκες εντολές για να δούμε τις πληροφορίες που θέλουμε. Μπορούμε να δημιουργούμε, να τροποποιούμε και να διαγράφουμε όσες όψεις θέλουμε, χωρίς να επηρεάζονται καθόλου τα δεδομένα των σχέσεων στις οποίες αναφέρονται οι όψεις.

Για να δημιουργήσουμε μια όψη, δίνουμε την εντολή **create view as**, ως εξής :

```
Create view ΑΘΛΗΤΗΣ_1 as
    (Select Κωδικός_Αθλητή, Επώνυμο, Ημνία_Γέννησης
    From ΑΘΛΗΤΗΣ);
```

Μετά τον όρο as γράφουμε μέσα σε παρένθεση την εντολή SQL της οποίας το αποτέλεσμα θα δημιουργήσει την όψη. Η παραπάνω όψη εμφανίζει λίγα μόνο από τα πεδία της σχέσης ΑΘΛΗΤΗΣ. Η επόμενη όψη εμφανίζει την μέση τιμή των ατομικών επιδόσεων (ρεκόρ) των αθλητών :

```
Create view ΑΘΛΗΤΗΣ_2 as
    (Select avg(Ρεκόρ)
    From ΑΘΛΗΤΗΣ);
```

Για να διαγράψουμε μια όψη, δίνουμε την εντολή **drop view**, ως εξής :

```
Drop view ΑΘΛΗΤΗΣ_1;
```

Η Εντολή Select

Θα δούμε τώρα τις εντολές της Γλώσσας Χειρισμού Δεδομένων DML (Data Manipulation Language). Η βασικότερη είναι η εντολή **Select**, που χρησιμοποιείται για την εμφάνιση (ανάκληση) δεδομένων από τη βάση δεδομένων. Η γενική σύνταξη της εντολής Select για την αναζήτηση δεδομένων σε μια βάση δεδομένων είναι η εξής :

```
Select Πεδίο1, Πεδίο2, … Πεδίον
From Πίνακας1, Πίνακας2, …, Πίνακαςm
Where συνθήκη;
```

Τα πεδία που δηλώνουμε στην εντολή Select είναι αυτά που θα εμφανίζονται στο αποτέλεσμα. Είναι αντίστοιχο με την πράξη της προβολής (projection) της σχεσιακής άλγεβρας. Αν θέλουμε να εμφανίζονται όλα τα πεδία θα πρέπει να χρησιμοποιήσουμε τον χαρακτήρα *. Ο όρος **From** δηλώνει τις σχέσεις (πίνακες) στους οποίους θα γίνει η αναζήτηση. Είναι αντίστοιχο με την πράξη του καρτεσιανού γινομένου της σχεσιακής άλγεβρας. Ο όρος **Where** είναι προαιρετικός και περιέχει τη συνθήκη που θέλουμε να ικανοποιούν οι εγγραφές (πλειάδες) του αποτελέσματος. Είναι αντίστοιχο με την πράξη της επιλογής (selection) της σχεσιακής άλγεβρας.

Η επόμενη εντολή Select εμφανίζει μόνο τα πεδία Επώνυμο και Όνομα αθλητή καθώς και τον Κωδικό του Συλλόγου στον οποίο ανήκει ο κάθε αθλητής και αυτό γι' όλους ανεξαιρέτως τους αθλητές που υπάρχουν στον πίνακα ΑΘΛΗΤΗΣ :

```
Select Επώνυμο, Όνομα, Κωδικός_Συλλόγου
From ΑΘΛΗΤΗΣ;
```

Επειδή υπάρχει η περίπτωση, ακραία μεν αλλά εφικτή, να υπάρχουν δύο ή και περισσότερες εγγραφές με ίδια τα παραπάνω στοιχεία (πεδία) ενός αθλητή, μπορούμε να

χρησιμοποιήσουμε και τον όρο distinct, ώστε να μην εμφανίζονται οι όμοιες εγγραφές, ως εξής :

```
Select distinct Επώνυμο, Όνομα, Κωδικός_Συλλόγου
From ΑΘΛΗΤΗΣ;
```

Αν θέλουμε να δούμε όλα τα στοιχεία (πεδία) των αθλητών, θα πρέπει να χρησιμοποιήσουμε το σύμβολο *, ως εξής :

```
Select *
From ΑΘΛΗΤΗΣ;
```

Η επόμενη εντολή Select εμφανίζει εκείνους τους αθλητές που έχουν ατομικό ρεκόρ μεγαλύτερο από μια συγκεκριμένη τιμή :

```
Select Επώνυμο, Όνομα, Κωδικός_Συλλόγου, Ρεκόρ
From ΑΘΛΗΤΗΣ
Where Ρεκόρ > 10.05;
```

Η επόμενη εντολή Select εμφανίζει με τη χρήση της διάζευξης (λογικό ή – or) εκείνους τους αθλητές που ανήκουν σ' έναν από δύο συλλόγους :

```
Select Επώνυμο, Όνομα, Κωδικός_Συλλόγου, Ρεκόρ
From ΑΘΛΗΤΗΣ
Where Κωδικός_Συλλόγου = 1 or Κωδικός_Συλλόγου = 2;
```

Η επόμενη εντολή Select εμφανίζει με τη χρήση της σύζευξης (λογικό και – and) εκείνους τους αθλητές που έχουν ατομικό ρεκόρ σε μια περιοχή τιμών :

```
Select Επώνυμο, Όνομα, Κωδικός_Συλλόγου, Ρεκόρ
From ΑΘΛΗΤΗΣ
Where Ρεκόρ > 10.00 and Ρεκόρ < 10.08;
```

Θα δούμε τώρα πώς μπορεί να γίνει η ανάκληση δεδομένων από πεδία που περιέχονται σε περισσότερες από μία σχέσεις. Η αναφορά σε πολλές σχέσεις αντιστοιχεί στην πράξη της σύνδεσης (join) της σχεσιακής άλγεβρας.

Για παράδειγμα, για τη συσχέτιση των σχέσεων ΑΘΛΗΤΗΣ και ΣΥΜΜΕΤΟΧΗ χρησιμοποιήσαμε το κοινό πεδίο Κωδικός_Αθλητή, το οποίο είναι πρωτεύον κλειδί στη σχέση ΑΘΛΗΤΗΣ και ξένο κλειδί στη σχέση ΣΥΜΜΕΤΟΧΗ.

Για να βρούμε τώρα ποιοι αθλητές συμμετείχαν σε ποιους αγώνες, δίνουμε την εξής εντολή :

```
Select Κωδικός_Αθλητή, Επώνυμο, Όνομα, Κωδικός_Αγώνα
From ΑΘΛΗΤΗΣ, ΣΥΜΜΕΤΟΧΗ
Where ΑΘΛΗΤΗΣ.Κωδικός_Αθλητή =
ΣΥΜΜΕΤΟΧΗ. Κωδικός_Αθλητή;
```

Είδαμε στο προηγούμενο παράδειγμα τη χρήση του όρου where με την απαραίτητη συνθήκη για τη δημιουργία της σύνδεσης των δύο σχέσεων (πινάκων). Μπορούμε να ορίσουμε και μια πιο πολύπλοκη συνθήκη για να βρούμε τους αθλητές που συμμετείχαν σε κάποιον συγκεκριμένο αγώνα :

```
Select Κωδικός_Αθλητή, Επώνυμο, Όνομα, Κωδικός_Αγώνα
From ΑΘΛΗΤΗΣ, ΣΥΜΜΕΤΟΧΗ
Where (ΑΘΛΗΤΗΣ.Κωδικός_Αθλητή =
ΣΥΜΜΕΤΟΧΗ. Κωδικός_Αθλητή) and Κωδικός_Αγώνα=100;
```

Ο Όρος As

Με τον όρο as μπορούμε να μετονομάσουμε πεδία ή σχέσεις αλλά μόνο προσωρινά για τις ανάγκες μιας εντολής Select και όχι μόνιμα. Για παράδειγμα, για να βρούμε τη διαφορά στην επίδοση των αθλητών από το όριο των 10.00, μπορούμε να δώσουμε την εξής εντολή :

```
Select Επώνυμο, Όνομα, Κωδικός_Συλλόγου, Ρεκόρ – 10.00 as
Διαφορά
From ΑΘΛΗΤΗΣ;
```

Μετονομάσαμε τη διαφορά Ρεκόρ – 10.00 ως Διαφορά, ώστε να μπορούμε να χειριστούμε το αποτέλεσμα της πράξης αυτής με μεγαλύτερη ευκολία αργότερα. Ο όρος as χρησιμοποιείται αναγκαστικά και για την μετονομασία πινάκων, όταν χρειαζόμαστε δύο μεταβλητές για την ίδια σχέση. Για παράδειγμα, για να μπορέσουμε να βρούμε τα στοιχεία (κωδικός, επώνυμο, όνομα) των αθλητών που ανήκουν στον ίδιο σύλλογο με τον αθλητή που έχει κωδικό ίσο με 106, δίνουμε της εξής εντολή :

```
Select Κωδικός_Αθλητή, Επώνυμο, Όνομα
From ΑΘΛΗΤΗΣ as A, ΑΘΛΗΤΗΣ as B
Where A.Κωδικός_Συλλόγου = B.Κωδικός_Συλλόγου and
B.Κωδικός_Αθλητή = 106;
```

Ο Όρος Order By

Με τον όρο **order by** μπορούμε να ταξινομήσουμε το αποτέλεσμα μιας εντολής Select ως προς κάποιο πεδίο. Για παράδειγμα, για να εμφανίσουμε μια λίστα των αθλητών ταξινομημένη ως προς επώνυμο πρώτα και μετά ως προς όνομα, δίνουμε την εξής εντολή :

```
Select Επώνυμο, Όνομα, Κωδικός_Συλλόγου
From ΑΘΛΗΤΗΣ
Order by Επώνυμο, Όνομα;
```

Η παραπάνω εντολή θα ταξινομήσει πρώτα ως προς το πεδίο Επώνυμο και όπου υπάρχουν ίδια Επώνυμα θα ταξινομήσει ως προς το πεδίο Όνομα.

Εξ ορισμού η ταξινόμηση γίνεται κατ' αύξουσα σειρά. Μπορούμε να χρησιμοποιήσουμε και τους όρους asc για αύξουσα και desc για φθίνουσα ταξινόμηση, ως εξής :

```
Select Επώνυμο, Όνομα, Κωδικός_Συλλόγου
From ΑΘΛΗΤΗΣ
Order by Κωδικός_Συλλόγου desc, Επώνυμο asc;
```

Η παραπάνω εντολή θα ταξινομήσει πρώτα ως προς το πεδίο Κωδικός_Συλλόγου κατά φθίνουσα σειρά και όπου υπάρχουν ίδιοι κωδικοί συλλόγου θα ταξινομήσει ως προς το πεδίο Επώνυμο κατ' αύξουσα σειρά.

Οι Συναρτήσεις Ομαδοποίησης

Η SQL χρησιμοποιεί μερικές πολύ χρήσιμες συναρτήσεις, που ονομάζονται συναρτήσεις ομαδοποίησης (aggregate functions) και οι οποίες δέχονται ένα σύνολο τιμών και επιστρέφουν μία τιμή. Οι συναρτήσεις αυτές είναι οι εξής :

Συνάρτηση Ομαδοποίησης	Αντίστοιχος Όρος στην SQL
Απαρίθμηση	Count
Άθροισμα	Sum
Μέσος Όρος	Avg
Μέγιστη Τιμή	Max
Ελάχιστη Τιμή	Min

Οι συναρτήσεις Sum και Avg εργάζονται μόνο με αριθμητικές τιμές, ενώ οι υπόλοιπες συναρτήσεις μπορούν να δεχθούν και αλφαριθμητικές τιμές. Για να βρούμε τον συνολικό αριθμό των αθλητών, δίνουμε την εξής εντολή :

```
Select count(*)
From ΑΘΛΗΤΗΣ;
```

Το αποτέλεσμα θα είναι ένας μόνο αριθμός, δηλ. ο συνολικός αριθμός των αθλητών (εγγραφών) της σχέσης ΑΘΛΗΤΗΣ. Για να βρούμε τον αθλητή που έχει την καλύτερη επίδοση, δίνουμε την εξής εντολή :

```
Select Κωδικός_Αθλητή, Επώνυμο, Όνομα, max(Ρεκόρ)
From ΑΘΛΗΤΗΣ;
```

Εδώ η συνάρτηση max() δέχεται σαν είσοδο όλες τις αριθμητικές τιμές του πεδίου Ρεκόρ και επιστρέφει την μέγιστη τιμή μαζί με τα στοιχεία του αθλητή που έχει το καλύτερο ρεκόρ. Για να βρούμε τον αθλητή που έχει την χαμηλότερη επίδοση, δίνουμε την εξής εντολή :

```
Select Κωδικός_Αθλητή, Επώνυμο, Όνομα, min(Ρεκόρ)
From ΑΘΛΗΤΗΣ;
```

Εδώ η συνάρτηση min() δέχεται σαν είσοδο όλες τις αριθμητικές τιμές του πεδίου Ρεκόρ και επιστρέφει την ελάχιστη τιμή μαζί με τα στοιχεία του αθλητή που έχει το χειρότερο ρεκόρ. Για να βρούμε τον μέσο όρο των επιδόσεων των αθλητών, δίνουμε την εξής εντολή :

```
Select avg(Ρεκόρ)
From ΑΘΛΗΤΗΣ;
```

Εδώ η συνάρτηση avg() δέχεται σαν είσοδο όλες τις αριθμητικές τιμές του πεδίου Ρεκόρ και επιστρέφει την μέση τιμή των επιδόσεων όλων των αθλητών. Και εδώ το αποτέλεσμα θα είναι ένας μόνο αριθμός.

Ο Όρος Group By

Με τον όρο **group by** μπορούμε να ξεχωρίσουμε (απομονώσουμε) τις εγγραφές μιας σχέσης σε ανεξάρτητα υποσύνολα και μετά να εφαρμόσουμε μια συνάρτηση ομαδοποίησης σε κάθε υποσύνολο. Για παράδειγμα, σ' έναν πίνακα ΠΕΛΑΤΩΝ μπορούμε να κάνουμε ομαδοποίηση σύμφωνα με το πεδίο Πόλη και έτσι να βρούμε πόσοι πελάτες υπάρχουν σε κάθε πόλη ή ποιος είναι ο μέσος όρος του υπολοίπου από τους πελάτες της κάθε πόλης ή ποιο είναι το μέγιστο υπόλοιπο από τους πελάτες της κάθε πόλης κοκ.

Μπορούμε ακόμα να εφαρμόσουμε και συνθήκες στην ομαδοποίηση, όπως αν θέλουμε να βρούμε σε ποιες πόλεις ο μέσος όρος του υπολοίπου των πελατών είναι μεγαλύτερος από 1.000 € κοκ. Παρατηρούμε ότι η ομαδοποίηση μπορεί να εφαρμοσθεί μόνο σε πεδία τα οποία έχουν επαναλαμβανόμενες τιμές σ' έναν πίνακα. Αυτό σημαίνει ότι η ομαδοποίηση έχει νόημα να γίνει με το πεδίο πόλη αλλά προφανώς δεν έχει νόημα να γίνει με το πεδίο επώνυμο, γιατί τα επώνυμα των πελατών θα είναι διαφορετικά μεταξύ τους.

Όταν χρησιμοποιούμε τον όρο group by για να δηλώσουμε ότι θέλουμε να κάνουμε ομαδοποίηση σύμφωνα με κάποιο πεδίο, τότε πρώτα γίνεται αυτόματα αύξουσα ταξινόμηση σύμφωνα με το πεδίο αυτό, δημιουργούνται τα αναγκαία υποσύνολα και μετά εφαρμόζονται στα υποσύνολα αυτά οι συναρτήσεις ομαδοποίησης που έχουμε ορίσει.

Θεωρούμε συνεπώς τον εξής πίνακα ΠΕΛΑΤΗΣ :

```
Create table ΠΕΛΑΤΗΣ
    (Κωδικός_Πελάτη    integer not null,
    Επώνυμο                char(20) not null,
    Όνομα                  char(15),
    Πόλη                   char(15),
    Ημνία_Γέννησης     date,
    Υπόλοιπο           numeric(4, 2),
    primary key        (Κωδικός_Πελάτη));
```

Για να βρούμε τον μέσο όρο του υπολοίπου των πελατών ανά πόλη, δίνουμε την εξής εντολή :

```
Select Πόλη, avg(Υπόλοιπο)
From ΠΕΛΑΤΗΣ
Group by Πόλη;
```

Η παραπάνω εντολή χωρίζει τη σχέση ΠΕΛΑΤΗΣ σε τόσα υποσύνολα όσες είναι οι υπάρχουσες πόλεις του πίνακα ΠΕΛΑΤΗΣ, όπου το κάθε υποσύνολο αναφέρεται σε μια συγκεκριμένη πόλη. Στη συνέχεια, εφαρμόζεται η συνάρτηση ομαδοποίησης avg() (μέσος όρος) σε κάθε υποσύνολο και το αποτέλεσμα θα είναι μια σχέση με τόσες εγγραφές όσες είναι και οι υπάρχουσες πόλεις του πίνακα ΠΕΛΑΤΗΣ.

Κάθε εγγραφή του αποτελέσματος θα περιέχει δύο πεδία, όπου το ένα θα είναι η πόλη και το άλλο ο μέσος όρος του υπολοίπου των πελατών για την κάθε πόλη. Για να βρούμε τις πόλεις στις οποίες ο μέσος όρος του υπολοίπου των πελατών ανά πόλη είναι μεγαλύτερος από 1.000 €, δίνουμε την εξής εντολή :

```
Select Πόλη, avg(Υπόλοιπο)
From ΠΕΛΑΤΗΣ
Group by Πόλη
Having avg(Υπόλοιπο) > 1000;
```

Ο όρος having χρησιμοποιείται στις συνθήκες που εφαρμόζονται σε κάθε υποσύνολο που δημιουργείται από τον όρο group by και όχι σε κάθε εγγραφή, όπως συμβαίνει με τη συνθήκη του όρου where. Η παρακάτω εντολή εμφανίζει το σύνολο των πελατών που βρίσκονται στην πόλη της Λάρισας και στην πόλη της Κοζάνης, ξεχωριστά για κάθε πόλη, ως εξής :

```
Select Πόλη, count(*)
From ΠΕΛΑΤΗΣ
Group by Πόλη
Having Πόλη='Λάρισα' or Πόλη='Κοζάνη';
```

Ο Όρος Intersect (Τομή)

Επειδή οι σχέσεις αντιστοιχούν σε σύνολα, μπορούμε να χρησιμοποιήσουμε τις πράξεις μεταξύ συνόλων και στις σχέσεις. Η πράξη της **τομής** στη σχεσιακή άλγεβρα επιτυγχάνεται στην SQL με τον όρο **Intersect**. Για να μπορέσουμε να εφαρμόσουμε την πράξη της τομής, όπως και τις πράξεις της ένωσης αλλά και της διαφοράς, ανάμεσα σε δύο σχέσεις, θα πρέπει αυτές να είναι **συμβατές**, δηλ. να έχουν τα ίδια πεδία ή πεδία με το ίδιο πεδίο ορισμού.

Για παράδειγμα, για να βρούμε τους κωδικούς των αθλητών που δεν ανήκουν σε κάποιο σύλλογο και που έχουν συμμετάσχει σε κάποιον συγκεκριμένο αγώνα, δίνουμε την εξής εντολή :

```
(Select Κωδικός_Αθλητή
 From ΑΘΛΗΤΗΣ
 Where Κωδικός_Συλλόγου <> 202)
 Intersect
(Select Κωδικός_Αθλητή
 From ΣΥΜΜΕΤΟΧΗ
 Where Κωδικός_Αγώνα = 301);
```

Η πράξη της τομής στο παραπάνω παράδειγμα είναι εφικτή, εφόσον οι δύο αυτές σχέσεις έχουν τα ίδια πεδία και δημιουργεί μια τρίτη σχέση που περιέχει τις κοινές πλειάδες των δύο αυτών σχέσεων.

Ο Όρος Union (Ένωση)

Η πράξη της **ένωσης** στη σχεσιακή άλγεβρα επιτυγχάνεται στην SQL με τον όρο **Union**. Για παράδειγμα, για να βρούμε τους κωδικούς των αθλητών που έχουν ρεκόρ μεγαλύτερο από κάποια συγκεκριμένη τιμή ή που έχουν συμμετάσχει σε κάποιον συγκεκριμένο αγώνα, δίνουμε την εξής εντολή :

```
(Select Κωδικός_Αθλητή
 From ΑΘΛΗΤΗΣ
 Where Ρεκόρ > 10.40)
Union
(Select Κωδικός_Αθλητή
 From ΣΥΜΜΕΤΟΧΗ
 Where Κωδικός_Αγώνα = 302);
```

Ο Όρος Except (Διαφορά)

Η πράξη της διαφοράς στη σχεσιακή άλγεβρα επιτυγχάνεται στην SQL με τον όρο Except. Για παράδειγμα, για να βρούμε τους κωδικούς των αθλητών που δεν έχουν συμμετάσχει σε κάποιον συγκεκριμένο αγώνα, δίνουμε την εξής εντολή :

```
(Select Κωδικός_Αθλητή
 From ΑΘΛΗΤΗΣ)
Except
(Select Κωδικός_Αθλητή
 From ΣΥΜΜΕΤΟΧΗ
 Where Κωδικός_Αγώνα = 302);
```

Οι Όροι In και Not In

Οι εντολές της SQL έχουν την ιδιότητα να εφαρμόζονται σε σχέσεις και το αποτέλεσμά τους να αποτελεί κι αυτό μια σχέση (ιδιότητα της κλειστότητας). Έτσι, μπορούμε να εμφωλιάσουμε εντολές ώστε το αποτέλεσμα (έξοδος) της μιας εντολής να αποτελεί είσοδο για μια άλλη εντολή.

Με τον όρο in μπορούμε να ελέγξουμε αν μια πλειάδα ανήκει σε μια σχέση η οποία δημιουργείται από μια φωλιασμένη εντολή. Ο όρος in είναι κάτι αντίστοιχο με το γνωστό σύμβολο των μαθηματικών ανήκει σε ( ). Για παράδειγμα, για να βρούμε τα ονόματα των αθλητών που έχουν συμμετάσχει σε κάποιον συγκεκριμένο αγώνα, δίνουμε την εξής εντολή :

```
Select Επώνυμο, Όνομα
From ΑΘΛΗΤΗΣ
Where Κωδικός_Αθλητή in (Select Κωδικός_Αθλητή
From ΣΥΜΜΕΤΟΧΗ
Where Κωδικός_Αγώνα = 302);
```

Θα εκτελεσθεί πρώτα η εντολή που βρίσκεται μέσα στις παρενθέσεις και από τη σχέση που θα δημιουργηθεί, η οποία θα περιέχει μια λίστα με τους κωδικούς των αθλητών που συμμετείχαν στον συγκεκριμένο αγώνα (ΣΥΜΜΕΤΟΧΗ), θα γίνει σύγκριση με τους κωδικούς των αθλητών που υπάρχουν στη σχέση ΑΘΛΗΤΗΣ, ώστε να εμφανισθούν τα Επώνυμα και τα Ονόματα των αθλητών που συμμετείχαν στον συγκεκριμένο αγώνα.

Υπάρχει και ο όρος not in για να ελέγξουμε αν μια πλειάδα δεν ανήκει σε μια σχέση η οποία δημιουργείται από μια φωλιασμένη εντολή. Ο όρος not in είναι κάτι αντίστοιχο με το γνωστό σύμβολο των μαθηματικών δεν ανήκει σε ( ). Για παράδειγμα, για να βρούμε τα ονόματα των αθλητών που δεν ανήκουν σε κάποιους συγκεκριμένους συλλόγους, δίνουμε την εξής εντολή :

```
Select Επώνυμο, Όνομα
```

```
From ΑΘΛΗΤΗΣ
Where Κωδικός_Συλλόγου not in (1, 2);
```

Οι Όροι Some και All

Χρησιμοποιούμε τον όρο **some** για να απαντήσουμε σε ερωτήματα όπου η τιμή ενός πεδίου θέλουμε να συγκριθεί με την τιμή ενός πεδίου **μίας τουλάχιστον** πλειάδας μιας σχέσης που προκύπτει από μια φωλιασμένη εντολή. Για παράδειγμα, για να βρούμε τα ονόματα των αθλητών που έχουν ατομικό ρεκόρ μικρότερο από έναν τουλάχιστον αθλητή ενός συγκεκριμένου συλλόγου, δίνουμε την εξής εντολή :

```
Select Επώνυμο, Όνομα
From ΑΘΛΗΤΗΣ
Where Ρεκόρ < some (Select Ρεκόρ
        From ΑΘΛΗΤΗΣ
        Where Κωδικός_Συλλόγου = 3);
```

Χρησιμοποιούμε τον όρο **all** για να απαντήσουμε σε ερωτήματα όπου η τιμή ενός πεδίου θέλουμε να συγκριθεί με την τιμή ενός πεδίου όλων των πλειάδων μιας σχέσης που προκύπτει από μια φωλιασμένη εντολή. Για παράδειγμα, για να βρούμε τα ονόματα των αθλητών που έχουν ατομικό ρεκόρ μικρότερο απ' όλους τους αθλητές ενός συγκεκριμένου συλλόγου, δίνουμε την εξής εντολή :

```
Select Επώνυμο, Όνομα
From ΑΘΛΗΤΗΣ
Where Ρεκόρ < all (Select Ρεκόρ
        From ΑΘΛΗΤΗΣ
        Where Κωδικός_Συλλόγου = 3);
```

Η Εντολή Insert Into

Οι εντολές της SQL που έχουμε συναντήσει μέχρι τώρα είχαν να κάνουν με την αναζήτηση και την εμφάνιση στοιχείων από μια βάση δεδομένων. Υπάρχουν, όμως, και εντολές για να εισάγουμε, διαγράψουμε και τροποποιήσουμε δεδομένα από μια βάση. Με την εντολή **insert into** μπορούμε να καταχωρήσουμε πλειάδες σε μια ήδη υπάρχουσα σχέση. Ακολουθεί ένα παράδειγμα :

```
Insert into ΑΘΛΗΤΗΣ (Κωδικός_Αθλητή, Επώνυμο, Όνομα, Ρεκόρ,
Ημνία_Γέννησης, Κωδικός_Συλλόγου)
 values (126, 'Νταμπίζας', 'Νικόλαος', 10.57, null, 306);
```

Παρατηρούμε ότι πρέπει να τηρήσουμε με προσοχή τη σειρά των πεδίων και τις αντίστοιχες τιμές τους (αλφαριθμητικές, αριθμητικές, ημερομηνίας κοκ). Για τα πεδία που δεν έχουμε κάποια τιμή, μπορούμε να δηλώσουμε τιμή null, εφόσον φυσικά αυτό επιτρέπεται.

Η Εντολή Delete From

Με την εντολή **delete from** μπορούμε να διαγράψουμε ολόκληρες πλειάδες και όχι μεμονωμένα πεδία (στήλες). Για παράδειγμα, για να διαγράψουμε όλους τους αθλητές που έχουν ατομικό ρεκόρ μεγαλύτερο από κάποια συγκεκριμένη τιμή, δίνουμε την εξής εντολή :

```
Delete from ΑΘΛΗΤΗΣ
Where Ρεκόρ > 11.00;
```

Για να διαγράψουμε όλες τις πλειάδες μιας σχέσης, αλλά όχι και την ίδια την σχέση, δίνουμε την εξής εντολή :

```
Delete from ΑΘΛΗΤΗΣ;
```

Η Εντολή Update

Με την εντολή **update** μπορούμε να τροποποιήσουμε την τιμή κάποιων πεδίων από ορισμένες ή και απ' όλες τις πλειάδες μιας σχέσης. Για παράδειγμα, για να αλλάξουμε τον κωδικό συλλόγου σε 320 για όσους αθλητές ανήκουν στον σύλλογο που έχει κωδικό 307, δίνουμε την εξής εντολή :

```
Update ΑΘΛΗΤΗΣ
Set Κωδικός_Συλλόγου = 320
Where Κωδικός_Συλλόγου = 307;
```

Επίσης, για να μηδενίσουμε τα ρεκόρ όλων των αθλητών, δίνουμε την εξής εντολή :

```
Update ΑΘΛΗΤΗΣ
Set Ρεκόρ = 00.00;
```

Η Δομημένη Γλώσσα Ερωτημάτων SQL

Μάθημα 1 - Τι Είναι η SQL

Η SQL αποτελεί μια στάνταρτ γλώσσα του ANSI για να μπορούμε να έχουμε πρόσβαση σε βάσεις δεδομένων.

- Τα αρχικά SQL σημαίνουν **Structured Query Language**, δηλ. **Δομημένη Γλώσσα Ερωτημάτων**.
- Η SQL μάς δίνει τη δυνατότητα να έχουμε πρόσβαση σε μια βάση δεδομένων (database).
- Η SQL αποτελεί μια στάνταρτ γλώσσα του ANSI (ANSI standard language).
- Η SQL μπορεί να εκτελέσει ερωτήματα (queries) σχετικά με μια βάση δεδομένων.
- Η SQL μπορεί να ανακτήσει δεδομένα από μια βάση δεδομένων.
- Η SQL μπορεί να εισαγάγει νέες εγγραφές σε μια βάση δεδομένων.
- Η SQL μπορεί να διαγράψει εγγραφές από μια βάση δεδομένων.
- Η SQL μπορεί να ενημερώσει εγγραφές σε μια βάση δεδομένων.
- Η SQL είναι πολύ εύκολη στην εκμάθηση.

Η SQL αποτελεί ένα στάνταρτ του ANSI (American National Standards Institute) για να μπορούμε να έχουμε πρόσβαση σε συστήματα βάσεων δεδομένων. Οι εντολές της SQL χρησιμοποιούνται για να ανακτήσουμε (retrieve) και να ενημερώσουμε (update) δεδομένα σε μια βάση δεδομένων (database).

Η SQL συνεργάζεται με προγράμματα βάσεων δεδομένων όπως είναι τα εξής : Access, Informix, Microsoft SQL Server, Oracle, Sybase και πολλά άλλα.

Μάθημα 2 - Οι Πίνακες Βάσεων Δεδομένων (Database Tables)

Οι βάσεις δεδομένων (databases) περιέχουν αντικείμενα (objects) που ονομάζονται Πίνακες (Tables). Οι Εγγραφές (Records) των δεδομένων απο-θηκεύονται σ' αυτούς τους πίνακες. Οι Πίνακες αναγνωρίζονται με τα ονόματά τους, όπως "Persons", "Orders", "Suppliers" κ.ά.

Οι Πίνακες περιέχουν Στήλες (Columns) και Γραμμές (Rows) με δε-δομένα. Οι Γραμμές (Rows) περιέχουν εγγραφές (records), όπως μία εγγραφή για κάθε άτομο. Οι Στήλες (Columns) περιέχουν δεδομένα, όπως First Name, Last Name, Address και City.

Ακολουθεί ένα παράδειγμα ενός Πίνακα που ονομάζεται "Persons" :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα
Αντωνιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Φον Κοζάνη 10	Κοζάνη

Τα LastName, FirstName, Address και City είναι οι Στήλες (Columns) του πίνακα. Οι Γραμμές (Rows) περιέχουν τρεις εγγραφές για τρία άτομα.

Μάθημα 3 - Τα Ερωτήματα της SQL (SQL Queries)

Με την SQL, μπορούμε να κάνουμε ένα ερώτημα (Query) σε μια βάση δεδομένων και να έχουμε ένα αποτέλεσμα (Result) σε μορφή πίνακα (tabu-lar form).

Ενα ερώτημα σαν το εξής :

```
SELECT LastName FROM Persons
```

θα δώσει ένα αποτέλεσμα σαν το εξής :

LastName

Παπαδόπουλος
Αντωνιάδης
Γεωργιάδης

Πρέπει να έχουμε υπόψη μας ότι μερικά συστήματα βάσεων δεδομένων απαιτούν το σύμβολο ; (semicolon) στο τέλος μιας εντολής SQL. Δεν θα χρησιμοποιήσουμε εδώ το σύμβολο ;.

Μάθημα 4 - Χειρισμός Δεδομένων της SQL (Data Manipulation)

Οπως υπονοεί και το όνομά της, η SQL είναι μια σύνταξη για την εκτέλεση ερωτημάτων (queries). Αλλά η γλώσσα της SQL περιλαμβάνει επίσης μια σύνταξη για την ενημέρωση εγγραφών, την εισαγωγή νέων εγγραφών και τη διαγραφή υπαρχόντων εγγραφών.

Αυτές οι εντολές ερωτημάτων και ενημέρωσης αποτελούν μαζί τη Γλώσσα Χειρισμού Δεδομένων (**Data Manipulation Language, DML**) που αποτελεί κομμάτι της SQL :

- **SELECT** - εξάγει δεδομένα από μια βάση δεδομένων.
- **UPDATE** - ενημερώνει δεδομένα σε μια βάση δεδομένων.
- **DELETE** - διαγράφει δεδομένα από μια βάση δεδομένων.
- **INSERT** - εισάγει νέα δεδομένα σε μια βάση δεδομένων.

Μάθημα 5 - Ορισμός Δεδομένων της SQL (Data Definition)

Η Γλώσσα Ορισμού Δεδομένων (Data Definition Language, DDL), που αποτελεί μέρος της SQL, επιτρέπει τη δημιουργία και τη διαγραφή πινάκων μιας βάσης δεδομένων. Μπορούμε επίσης να ορίσουμε indexes (keys), να καθορίσουμε συνδέσμους (links) ανάμεσα στους πίνακες και να επιβάλλουμε περιορισμούς ανάμεσα στους πίνακες μιας βάσης δεδομένων.

Οι σημαντικότερες εντολές DDL στην SQL είναι οι εξής :

- **CREATE TABLE** - δημιουργεί έναν νέον πίνακα σε μια βάση δεδομένων.
- **ALTER TABLE** - τροποποιεί έναν πίνακα σε μια βάση δεδομένων.
- **DROP TABLE** - διαγράφει έναν πίνακα από μια βάση δεδομένων.
- **CREATE INDEX** - δημιουργεί έναν index (search key).
- **DROP INDEX** - διαγράφει έναν index.

Μάθημα 6 - Η SQL και οι Ενεργές Σελίδες Διακομιστή

Η SQL αποτελεί ένα σημαντικό κομμάτι της ASP, επειδή το Ενεργό Αντικείμενο Δεδομένων (Active Data Object, ADO) που χρησιμοποιείται στην ASP (Active Server Pages) για να μπορούμε να έχουμε πρόσβαση σε βάσεις δεδομένων, βασίζεται στην SQL για την πρόσβαση στα δεδομένα.

Μάθημα 7 - Η Εντολή Select της SQL

Η εντολή SELECT επιλέγει στήλες (columns) δεδομένων από μια βάση δεδομένων. Το αποτέλεσμα αποθηκεύεται σε μορφή πίνακα και αποκαλείται result set. Την χρησιμοποιούμε για να εμφανίζουμε (επιλέγουμε) πληροφορίες από έναν πίνακα ως εξής :

SELECT ονόματα_στηλών FROM όνομα_πίνακα

Παράδειγμα : Επιλογή Στηλών από έναν Πίνακα

Για να επιλέξουμε τις στήλες "LastName" και "FirstName", χρησιμοποιούμε μια εντολή SELECT, ως εξής :

SELECT LastName, FirstName FROM Persons

Το αποτέλεσμα :

LastName	FirstName
Παπαδόπουλος	Δημήτριος
Αντωνιάδης	Αντώνιος
Γεωργιάδης	Νικόλαος

Παράδειγμα : Επιλογή όλων των Στηλών

Για να επιλέξουμε όλες τις στήλες από τον πίνακα "Person", χρησιμοποιούμε το σύμβολο * αντί για όνομα στήλης, ως εξής :

SELECT * FROM Persons

Το αποτέλεσμα :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα
Αντωνιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Φον Κοζάνη 10	Κοζάνη

Μάθημα 8 - Το Where Clause της SQL

Το WHERE clause χρησιμοποιείται για να καθορίσουμε ένα κριτήριο επιλογής (selection criteria). Για να μπορέσουμε να επιλέξουμε δεδομένα υπό συνθήκη από έναν πίνακα, πρέπει να προσθέσουμε ένα WHERE clause σε μια εντολή SELECT, ως εξής :

SELECT στήλη FROM πίνακα WHERE στήλη συνθήκη τιμή

Με το WHERE clause, μπορούμε να χρησιμοποιήσουμε αυτές τις συνθήκες :

Τελεστής (Operator)	Συνθήκη (Condition)
=	Ίσο
<>	Όχι ίσο
>	Μεγαλύτερο από
<	Μικρότερο από
>=	Μεγαλύτερο από ή ίσο με
<=	Μικρότερο από ή ίσο με
LIKE	Επεξηγείται παρακάτω

Σε μερικές εκδόσεις της SQL, ο τελεστής για το όχι ίσο (<>), μπορεί να γραφεί ως εξής : **!=**.

Παράδειγμα : Επιλογή Ατόμων από μια Πόλη

Για να επιλέξουμε μόνο τα άτομα που κατοικούν στην πόλη Φλώρινα, προσθέτουμε ένα WHERE clause στην εντολή SELECT, ως εξής :

SELECT * FROM Persons WHERE City='Φλώρινα'

Το αποτέλεσμα :

LastName	FirstName	Address	City	Year
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα	1951
Αντωνιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα	1978

Εχουμε χρησιμοποιήσει μονά εισαγωγικά (single quotes) στις τιμές των συνθηκών στα παραδείγματα. Η SQL χρησιμοποιεί μονά εισαγωγικά στις αλφαριθμητικές τιμές (κείμενο). Τα περισσότερα συστήματα βάσεων δεδομένων αποδέχονται και τα διπλά εισαγωγικά (double quotes). Οι αριθμητικές τιμές δεν πρέπει να περικλείονται σε εισαγωγικά.

Για τις τιμές κειμένου (αλφαριθμητικές) :

Αυτό είναι σωστό :

SELECT * FROM Persons WHERE FirstName='Δημήτριος'

Αυτό δεν είναι σωστό :

SELECT * FROM Persons WHERE FirstName=Δημήτριος

Για τις αριθμητικές τιμές :

Αυτό είναι σωστό :

SELECT * FROM Persons WHERE Year>1965

Αυτό δεν είναι σωστό :

SELECT * FROM Persons WHERE Year>'1965'

Μάθημα 9 - Η Συνθήκη LIKE

Η συνθήκη LIKE χρησιμοποιείται για να καθορίσουμε μια αναζήτηση για ένα υπόδειγμα (pattern) σε μια στήλη.

Η σύνταξη είναι ως εξής :

SELECT στήλη **FROM** πίνακα **WHERE** στήλη **LIKE** υπόδειγμα

Μπορούμε να χρησιμοποιήσουμε το σύμβολο "%" για να ορίσουμε χαρακτήρες μπαλαντέρ (wildcards) πριν και μετά από το υπόδειγμα.

Παράδειγμα : Επιλογή Ατόμων με Υπόδειγμα Ονόματος

Η εντολή SQL θα επιστρέψει τα άτομα που το όνομά τους αρχίζει από 'Ο'.

SELECT * FROM Persons WHERE FirstName LIKE 'Ο%'

Η εντολή SQL θα επιστρέψει τα άτομα που το όνομά τους τελειώνει σε 'α'.

SELECT * FROM Persons WHERE FirstName LIKE '%α'

Η εντολή SQL θα επιστρέψει τα άτομα που το όνομά τους περιέχει το 'γα'.

SELECT * FROM Persons WHERE FirstName LIKE '%γα%'

Ολα τα παραπάνω παραδείγματα θα επιστρέψουν το εξής αποτέλεσμα :

LastName	FirstName	Address	City	Year
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα	1951

Μάθημα 10 - Οι Λογικοί Τελεστές And και Or

Τα AND και OR ενώνουν δύο ή περισσότερες συνθήκες σ' ένα WHE-RE clause. Ο τελεστής AND εμφανίζει μια γραμμή αν ΟΛΕΣ οι συνθήκες είναι αληθείς (true), ενώ ο τελεστής OR εμφανίζει μια γραμμή αν ΟΠΟΙΑΔΗΠΟΤΕ από τις συνθήκες είναι αληθής (true).

Ο αρχικός πίνακας είναι ο εξής :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Ι.Καραβίτη 2	Φλώρινα

Παράδειγμα

Χρησιμοποιούμε το AND για να εμφανίσουμε αυτούς που το όνομά τους είναι "Νικόλαος" και το επώνυμό τους είναι "Γεωργιάδης" :

```
SELECT * FROM Persons
        WHERE FirstName='Νικόλαος'
                AND LastName='Γεωργιάδης'
```

Αποτέλεσμα :

LastName	FirstName	Address	City
Γεωργιάδης	Νικόλαος	Ι.Καραβίτη 2	Φλώρινα

Παράδειγμα

Χρησιμοποιούμε το OR για να εμφανίσουμε αυτούς που το όνομά τους είναι "Νικόλαος" ή το επώνυμό τους είναι "Γεωργιάδης" :

```
SELECT * FROM Persons
        WHERE firstname='Νικόλαος'
                OR lastname='Γεωργιάδης'
```

Αποτέλεσμα :

LastName	FirstName	Address	City
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Φον Κοζάνη 10	Κοζάνη

Παράδειγμα

Μπορούμε επίσης να συνδυάσουμε τα AND και OR, χρησιμοποιώντας παρενθέσεις για να σχηματίσουμε πολύπλοκες εκφράσεις :

```
SELECT * FROM Persons WHERE
 (FirstName='Αντώνιος' OR FirstName='Νικόλαος')
   AND LastName='Γεωργιάδης'
```

Αποτέλεσμα :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα
Αντωνιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Φον Κοζάνη 10	Κοζάνη

Μάθημα 11 - Ο Τελεστής Between ... And

Ο τελεστής BETWEEN ... AND επιλέγει μια περιοχή δεδομένων ανάμεσα σε δύο τιμές. Οι τιμές μπορεί να είναι αριθμοί, κείμενο ή ημερομηνίες.

```
SELECT όνομα_στήλης FROM όνομα_πίνακα
  WHERE όνομα_στήλης
    BETWEEN τιμή1 AND τιμή2
```

Αρχικός πίνακας :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Ι.Καραβίτη 2	Φλώρινα
Μαρκόπουλος	Νικόλαος	Φον Καραγιαν 20	Κοζάνη

Παράδειγμα

Για να εμφανίσουμε τα άτομα που βρίσκονται αλφαβητικά ανάμεσα στους "Γεωργιάδης" και "Μαρκόπουλος", αλλά και να τους περιλαμβάνουν :

```
SELECT * FROM Persons WHERE LastName
        BETWEEN 'Γεωργιάδης' AND 'Μαρκόπουλος'
```

Αποτέλεσμα :

LastName	FirstName	Address	City
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Ι.Καραβίτη 2	Φλώρινα

Μαρκόπουλος	Νικόλαος	Φον Καραγιαν 20	Κοζάνη

Παράδειγμα

Για να εμφανίσουμε τα άτομα που βρίσκονται έξω από την περιοχή που χρησιμοποιήσαμε στο προηγούμενο παράδειγμα, χρησιμοποιούμε τον τελεστή NOT :

```
SELECT * FROM Persons WHERE LastName
 NOT BETWEEN 'Γεωργιάδης' AND 'Μαρκόπουλος'
```

Αποτέλεσμα :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα

Μάθημα 12 - Η Λέξη Κλειδί Distinct

Η λέξη κλειδί DISTINCT χρησιμοποιείται για να επιστρέφει μόνο διακριτές (διαφορετικές) (distinct, different) τιμές. Η εντολή SELECT της SQL επιστρέφει στοιχεία από τις στήλες ενός πίνακα, αλλά τι μπορούμε να κάνουμε αν θέλουμε να επιλέξουμε μόνο διακριτά στοιχεία (distinct eleme-nts);

Στην SQL, αυτό που πρέπει να κάνουμε είναι να προσθέσουμε μια λέξη κλειδί DISTINCT στην εντολή SELECT, ως εξής :

SELECT DISTINCT *ονόματα_στηλών* **FROM** *όνομα_πίνακα*

Παράδειγμα

Επιλογή εταιρειών από έναν πίνακα παραγγελιών.

Ενας απλός πίνακας παραγγελιών :

Company	OrderNumber
Line Computers	3412
Sony	2312
Algorithm	4678
Sony	6798

Η επόμενη εντολή SQL :

```
SELECT Company FROM Orders
```

θα δώσει αυτό το αποτέλεσμα :

Company
Line Computers
Sony
Algorithm
Sony

Βλέπουμε ότι η εταιρεία Sony εμφανίζεται δύο φορές στο αποτέλεσμα. Μερικές φορές δεν το θέλουμε αυτό.

Παράδειγμα

Επιλογή ξεχωριστών εταιρειών από έναν πίνακα παραγγελιών.

Η επόμενη εντολή SQL :

```
SELECT DISTINCT Company FROM Orders
```

θα δώσει αυτό το αποτέλεσμα :

Company
Line Computers

Sony
Algorithm

Τώρα η εταιρεία Sony εμφανίζεται μόνο μία φορά στο αποτέλεσμα.

Μάθημα 13 - Η Λέξη Κλειδί Order By

Η λέξη κλειδί ORDER BY χρησιμοποιείται για να ταξινομήσει το αποτέλεσμα. Το ORDER BY clause χρησιμοποιείται για να ταξινομήσει τις γραμμές.

Πίνακας Παραγγελίες :

Company	OrderNumber
Digital Shop	3412
ABC Shop	5678
Sony	2312
Sony	6798

Παράδειγμα

Για να εμφανίσουμε τις εταιρείες σε αλφαβητική σειρά :

```
SELECT Company, OrderNumber FROM Orders
    ORDER BY Company
```

Αποτέλεσμα :

Company	OrderNumber
ABC Shop	5678
Digital Shop	3412
Sony	6798
Sony	2312

Παράδειγμα

Για να εμφανίσουμε τις εταιρείες σε αλφαβητική σειρά ΚΑΙ (AND) τις παραγγελίες σε αριθμητική σειρά :

```
SELECT Company, OrderNumber FROM Orders
    ORDER BY Company, OrderNumber
```

Αποτέλεσμα :

Company	OrderNumber
ABC Shop	5678
Digital Shop	3412
Sony	2312
Sony	6798

Παράδειγμα

Για να εμφανίσουμε τις εταιρείες σε αντίστροφη αλφαβητική σειρά (reverse alphabetical order) :

```
SELECT Company, OrderNumber FROM Orders
    ORDER BY Company DESC
```

Αποτέλεσμα :

Company	OrderNumber
Sony	2312
Sony	6798
Digital Shop	3412
ABC Shop	5678

Μάθημα 14 - Ασκηση σε SQL

Δημιουργείστε έναν πίνακα με όνομα Customers σε μια βάση δεδομένων με τις εξής στήλες :

Company Name	ContactName	Address	City

Καταχωρείστε στοιχεία στον πίνακα και δοκιμάστε τις εξής εντολές της SQL :

```
SELECT * FROM customers
SELECT CompanyName, ContactName
        FROM customers
SELECT * FROM customers
        WHERE companyname LIKE 'α%'
SELECT CompanyName, ContactName
        FROM customers
                WHERE CompanyName > 'γ'
                        AND ContactName > 'γ'
```

Μάθημα 15 - Η Εντολή INSERT INTO

Η εντολή INSERT INTO εισάγει νέες γραμμές σ' έναν πίνακα. Η σύνταξή της είναι ως εξής :

INSERT INTO όνομα_πίνακα
VALUES (τιμή1, τιμή2, ...)

Μπορούμε επίσης να καθορίσουμε τις στήλες για τις οποίες θέλουμε να εισάγουμε δεδομένα :

INSERT INTO όνομα_πίνακα(στήλη1, στήλη2, ...)
VALUES (τμή1, τιμή2, ...)

Ο επόμενος πίνακας "Persons" :

LastName	firstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα

και αυτή η εντολή SQL :

```
INSERT INTO Persons
  VALUES ('Σιάμκουρης', 'Ιωάννης', 'Π. Μελά 90', 'Καστοριά')
```

δίνουν αυτό το αποτέλεσμα :

LastName	firstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά

Εισαγωγή Δεδομένων σε Συγκεκριμένες Στήλες

Ο επόμενος πίνακας "Persons" :

LastName	firstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά

και η επόμενη εντολή SQL :

```
INSERT INTO Persons (LastName, Address)
VALUES ('Νικολάου', 'Ταγμ. Ναούμ 30')
```

δίνουν αυτό το αποτέλεσμα :

LastName	firstName	Address	City

Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου		Ταγμ. Ναούμ 30	Καστοριά

Μάθημα 16 - Η Εντολή Update

Η εντολή UPDATE ενημερώνει ή αλλάζει γραμμές. Η σύνταξή της είναι ως εξής :

UPDATE όνομα_πίνακα SET όνομα_στήλης=νέα_τιμή
WHERE όνομα_στήλης=τιμή

Αρχικός πίνακας Person :

LastName	firstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου		Ταγμ. Ναούμ 30	Καστοριά

Ενημέρωση μίας Στήλης σε μια Γραμμή

Θέλουμε να προσθέσουμε ένα όνομα στο άτομο που έχει το επώνυμο "Νικολάου":

```
UPDATE Person SET FirstName = 'Αθηνά'
WHERE LastName = 'Νικολάου'
```

Ενημέρωση Πολλών Στηλών σε μια Γραμμή

Για το ίδιο άτομο θέλουμε να αλλάξουμε τη διεύθυνση και να προσθέσουμε ένα όνομα για την πόλη :

```
UPDATE Person
  SET Address = 'Μεγαρόβου 12', City ='Φλώρινα'
    WHERE LastName = 'Νικολάου'
```

Αποτέλεσμα :

LastName	firstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου	Αθηνά	Μεγαρόβου 12	Φλώρινα

Μάθημα 17 - Η Εντολή Delete

Η εντολή DELETE χρησιμοποιείται για να διαγράψουμε γραμμές από έναν πίνακα. Η σύνταξή της είναι ως εξής :

DELETE FROM όνομα_πίνακα
WHERE όνομα_στήλης = τιμή

Αρχικός πίνακας Person :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου	Αθηνά	Μεγαρόβου 12	Φλώρινα

Διαγραφή μιας Γραμμής

Θα διαγράψουμε την "Νικολάου Αθηνά" :

```
DELETE FROM Person WHERE LastName = 'Νικολάου'
```

Αποτέλεσμα :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά

Μάθημα 18 - Οι Συναρτήσεις Count της SQL

Η SQL έχει ενσωματωμένες συναρτήσεις για τη μέτρηση (counting) των εγγραφών μιας βάσης δεδομένων.

Η σύνταξη για τις ενσωματωμένες συναρτήσεις COUNT είναι η εξής :

SELECT COUNT(στήλη) FROM πίνακας

Η Συνάρτηση COUNT(*)

Η συνάρτηση COUNT(*) επιστρέφει τον αριθμό των επιλεγμένων γραμμών από μια επιλογή (selection).

Μ' αυτόν τον πίνακα "Persons" :

Name	Age
Αντωνιάδης Ηλίας	34
Μάρκου Ιωάννης	45
Ιωάννου Γεώργιος	19

Το επόμενο παράδειγμα επιστρέφει τον αριθμό των γραμμών του πίνακα :

SELECT COUNT(*) FROM Persons

Αποτέλεσμα : 3

Το επόμενο παράδειγμα επιστρέφει τον αριθμό των ατόμων που είναι πάνω από 20 χρονών :

SELECT COUNT(*) FROM Persons where Age>20

Αποτέλεσμα : 2

Η Συνάρτηση COUNT(column)

Η συνάρτηση COUNT(column) επιστρέφει τον αριθμό των γραμμών χωρίς τιμή NULL στη συγκεκριμένη στήλη.

Μ' αυτόν τον πίνακα "Persons" :

Name	Age
Αντωνιάδης Ηλίας	34
Μάρκου Ιωάννης	45
Ιωάννου Γεώργιος	

Το επόμενο παράδειγμα βρίσκει τον αριθμό των ατόμων που έχουν τιμή στο πεδίο "Age" του πίνακα "Persons" :

SELECT COUNT(Age) FROM Persons

Αποτέλεσμα : 2

Η συνάρτηση COUNT(column) είναι χρήσιμη για να βρίσκουμε τις στήλες που δεν έχουν τιμή. Το αποτέλεσμα είναι κατά ένα λιγότερο από τον αριθμό των γραμμών του αρχικού πίνακα επειδή ένα από τα άτομα δεν έχει τιμή στο πεδίο age.

Μάθημα 19 - Οι Λέξεις Κλειδιά COUNT και DISTINCT

Οι λέξεις κλειδιά DISTINCT και COUNT μπορούν να χρησιμοποιηθούν μαζί για να μετρήσουμε τον αριθμό των διακριτών αποτελεσμάτων (di-stinct results).

Η σύνταξη είναι ως εξής :

SELECT DISTINCT COUNT(στήλες) FROM πίνακας

Με τον επόμενο πίνακα "Orders" :

Company	OrderNumber
Hitachi	3412
Sony	2312
ABC	4678
Sony	6798

Η επόμενη εντολή SQL :

SELECT COUNT(Company) FROM Orders

θα δώσει αυτό το αποτέλεσμα : 4

Η επόμενη εντολή SQL :

SELECT DISTINCT COUNT(Company) FROM Orders

θα δώσει αυτό το αποτέλεσμα : 3

Μάθημα 20 - Οι Συναρτήσεις της SQL

Η SQL έχει πολλές ενσωματωμένες συναρτήσεις για να μπορούμε να κάνουμε μετρήσεις (counting) και υπολογισμούς (calculations).

Η γενική σύνταξη για τις ενσωματωμένες συναρτήσεις της SQL είναι η εξής :

SELECT function(στήλη) FROM πίνακας

Ο αρχικός πίνακας :

Name	Age
Αντωνιάδης Ηλίας	34
Μάρκου Ιωάννης	45
Ιωάννου Γεώργιος	19

Η Συνάρτηση AVG(column)

Η συνάρτηση AVG επιστρέφει τη μέση τιμή μιας στήλης σε μια επιλογή. Οι τιμές NULL δεν περιλαμβάνονται στους υπολογισμούς.

Το επόμενο παράδειγμα επιστρέφει τον μέσο όρο ηλικίας των ατόμων που υπάρχουν στον πίνακα "Persons" :

SELECT AVG(Age) FROM Persons

Αποτέλεσμα : 32.67

Το επόμενο παράδειγμα επιστρέφει τον μέσο όρο ηλικίας των ατόμων που είναι πάνω από 20 χρονών :

SELECT AVG(Age) FROM Persons where Age>20

Αποτέλεσμα : 39.5

Η Συνάρτηση MAX(column)

Η συνάρτηση MAX επιστρέφει την μεγαλύτερη τιμή μιας στήλης. Οι τιμές NULL δεν περιλαμβάνονται στον υπολογισμό.

Παράδειγμα

SELECT MAX(Age) FROM Persons

Αποτέλεσμα : 45

Η Συνάρτηση MIN(column)

Η συνάρτηση MIN επιστρέφει την μικρότερη τιμή μιας στήλης. Οι τιμές NULL δεν περιλαμβάνονται στον υπολογισμό.

Παράδειγμα

SELECT MIN(Age) FROM Persons

Αποτέλεσμα : 19

Οι συναρτήσεις MIN και MAX μπορούν επίσης να χρησιμοποιηθούν σε στήλες που περιέχουν κείμενο (text), για να βρούμε την μεγαλύτερη ή μικρότερη τιμή σε αλφαβητική σειρά.

Η Συνάρτηση SUM(column)

Η συνάρτηση SUM επιστρέφει το άθροισμα μιας στήλης για μια συγκεκριμένη επιλογή (selection). Οι τιμές NULL δεν περιλαμβάνονται στον υπολογισμό.

Παράδειγμα

Το παρακάτω παράδειγμα επιστρέφει το άθροισμα όλων των ηλικιών του πίνακα "person" :

SELECT SUM(Age) FROM Persons

Αποτέλεσμα : 98

Παράδειγμα

Το παρακάτω παράδειγμα επιστρέφει το άθροισμα των ηλικιών του πίνακα "person" για τα άτομα που είναι πάνω από 20 χρονών :

SELECT SUM(Age) FROM Persons where Age>20

Αποτέλεσμα : 79

Μάθημα 21 - Η Λέξη Κλειδί Group By

Η λέξη κλειδί GROUP BY έχει προστεθεί στην SQL επειδή οι αθροιστικές συναρτήσεις (aggregate functions), όπως είναι η SUM, επιστρέφουν το σύνολο όλων των τιμών μιας στήλης κάθε φορά που καλούνται.

Χωρίς την λέξη κλειδί GROUP BY, το να βρούμε το άθροισμα για κά-θε ανεξάρτητη ομάδα τιμών μιας στήλης θα ήταν αδύνατο.

Η σύνταξη της GROUP BY είναι η εξής :

SELECT στήλη, SUM(στήλη) FROM πίνακας GROUP BY στήλη

Παράδειγμα με GROUP BY

Ο παρακάτω πίνακας "Sales" :

Company	Amount
Grundig	5500
IBM	4500
Grundig	7100

και αυτή η εντολή SQL :

SELECT Company, SUM(Amount) FROM Sales

θα δώσουν αυτό το αποτέλεσμα :

Company	SUM(Amount)
Grundig	17100
IBM	17100
Grundig	17100

Ο παραπάνω κώδικας δεν είναι έγκυρος επειδή η στήλη που επιστρέφει δεν αποτελεί μέρος ενός αθροίσματος (aggregate). Ένα GROUP BY clause μπορεί να το διορθώσει αυτό, ως εξής :

```
SELECT Company, SUM(Amount) FROM Sales
GROUP BY Company
```

και θα δώσει αυτό το αποτέλεσμα :

Company	SUM(Amount)
Grundig	12600
IBM	4500

Μάθημα 22 - Η Λέξη Κλειδί HAVING

Η λέξη κλειδί HAVING έχει προστεθεί στην SQL επειδή η λέξη κλειδί WHERE δεν μπορεί να χρησιμοποιηθεί σε αθροιστικές συναρτήσεις, όπως είναι η SUM.

Η σύνταξη της συνάρτησης HAVING είναι η εξής :

```
SELECT στήλη, SUM(στήλη) FROM πίνακας
  GROUP BY στήλη
    HAVING SUM(στήλη) συνθήκη
```

Ο παρακάτω πίνακας "Sales" :

Company	Amount
Grundig	5500
IBM	4500
Grundig	7100

και αυτή η εντολή SQL :

```
SELECT Company, SUM(Amount) FROM Sales
  GROUP BY Company HAVING SUM(Amount)>10000
```

θα δώσουν αυτό το αποτέλεσμα :

Company	SUM (Amount)
Grundig	12600

Μάθημα 23 - Τα Ψευδώνυμα (Aliases)

Στην SQL, τα ψευδώνυμα (aliases) χρησιμοποιούνται για ονόματα στηλών και πινάκων.

Ψευδώνυμο Στήλης (Column Name Alias)

Η σύνταξη είναι ως εξής :

```
SELECT στήλη AS column_alias FROM πίνακας
```

Ψευδώνυμο Πίνακα (Table Name Alias)

Η σύνταξη είναι ως εξής :

```
SELECT στήλη FROM πίνακας AS table_alias
```

Παράδειγμα με Column Alias

Ο επόμενος πίνακας Persons :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου	Αθηνά	Μεγαρόβου 12	Φλώρινα

και η εξής εντολή SQL :

```
SELECT LastName AS Family, FirstName AS Name
      FROM Persons
```

δίνουν αυτό το αποτέλεσμα :

Family	Name
Σουμπάση	Μαρία
Σιάμκουρης	Ιωάννης
Νικολάου	Αθηνά

Παράδειγμα με Table Alias

Ο επόμενος πίνακας Persons :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου	Αθηνά	Μεγαρόβου 12	Φλώρινα

και η εξής εντολή SQL :

```
SELECT LastName, FirstName
        FROM Persons AS Employees
```

δίνουν αυτό το αποτέλεσμα :

LastName	FirstName
Σουμπάση	Μαρία
Σιάμκουρης	Ιωάννης
Νικολάου	Αθηνά

Μάθημα 24 - Ενωση Πινάκων (Join)

Μερικές φορές πρέπει να επιλέξουμε δεδομένα από δύο πίνακες για να δημιουργήσουμε ένα πιο πολύπλοκο αποτέλεσμα. Θα πρέπει να κάνουμε μια **ένωση (join)**.

Οι πίνακες μιας βάσης δεδομένων μπορούν να συσχετιστούν μεταξύ τους με **κλειδιά (keys)**. Ένα **πρωτεύον κλειδί** (**primary key**) είναι μια στήλη με μια μοναδική τιμή στην κάθε γραμμή. Ο σκοπός είναι να ενώσει τα δεδομένα μαζί από διάφορους πίνακες, χωρίς να έχουμε επανάληψη όλων των δεδομένων σε κάθε πίνακα.

Στον πίνακα "Employees" παρακάτω, η στήλη "ID" είναι το πρωτεύον κλειδί (primary key), που σημαίνει ότι δεν μπορούν να υπάρχουν δύο γραμμές που να έχουν το ίδιο ID. Το ID είναι αυτό που ξεχωρίζει δύο άτομα ακόμη κι αν έχουν το ίδιο όνομα.

Στους πίνακες παραδειγμάτων παρακάτω, προσέχουμε τα εξής :

- Η στήλη "ID" είναι το πρωτεύον κλειδί του πίνακα "Employees".
- Η στήλη "ID" του πίνακα "Orders" χρησιμοποιείται για να αναφερόμαστε στα άτομα του πίνακα "Employees", χωρίς να χρησιμοποιούμε τα ονόματά τους.

Πίνακας Employees :

ID	Name
01	Νικολάου Αθηνά
02	Γεωργιάδης Ηλίας
03	Παπαδόπουλος Στέφανος
04	Σουμπάσης Ιωάννης

Πίνακας Orders :

ID	Product
01	Printer
02	Computer
03	Scanner

Παράδειγμα

Ποιοι έχουν παραγγείλει προϊόντα και τι έχουν παραγγείλει;

```
SELECT Employees.Name, Orders.Product
  FROM Employees, Orders
    WHERE Employees.ID = Orders.ID
```

Αποτέλεσμα :

Name	Product
Νικολάου Αθηνά	Printer
Παπαδόπουλος Στέφανος	Computer
Παπαδόπουλος Στέφανος	Scanner

Παράδειγμα

Ποιοι έχουν παραγγείλει εκτυπωτή (printer);

```
SELECT Employees.Name
FROM Employees, Orders
WHERE Employees.ID = Orders.ID
AND Orders.Product = 'Printer'
```

Αποτέλεσμα :

Name
Νικολάου Αθηνά

Μάθημα 25 - Δημιουργία Βάσης Δεδομένων και Πίνακα

Για να δημιουργήσουμε μια βάση δεδομένων, μπορούμε να δώσουμε την εξής εντολή :

```
CREATE DATABASE όνομα_βάσης_δεδομένων
```

Για να δημιουργήσουμε έναν πίνακα σε μια βάση δεδομένων, μπορούμε να δώσουμε την εξής εντολή :

```
CREATE TABLE όνομα_πίνακα
(
όνομα_στήλης_1 τύπος_δεδομένων,
όνομα_στήλης_2 τύπος_δεδομένων,
…
)
```

Παράδειγμα

Θα δημιουργήσουμε έναν πίνακα με όνομα "Person", με τέσσερις στήλες με ονόματα "LastName", "FirstName", "Address" και "Age" :

```
CREATE TABLE Person
(
LastName varchar,
FirstName varchar,
Address varchar,
Age number
)
```

Θα ορίσουμε ένα μέγιστο μήκος για μερικές στήλες :

```
CREATE TABLE Person
(
LastName varchar(30),
FirstName varchar,
```

```
Address varchar,
Age number(3)
)
```

Ο τύπος δεδομένων (data type) καθορίζει τι είδος δεδομένων θα περιέχει η στήλη. Ο παρακάτω πίνακας περιέχει τους πιο συνηθισμένους τύπους δεδομένων της SQL :

Τύπος Δεδομένων (Data Type)	Περιγραφή (Description)
char(size)	Περιέχει ένα string σταθερού μήκους που μπορεί να περιέχει γράμματα, αριθμούς και ειδικούς χαρακτήρες. Το σταθερό μέγεθος καθορίζεται στις παρενθέσεις.
varchar(size)	Περιέχει ένα string μεταβλητού μήκους που μπορεί να περιέχει γράμματα, αριθμούς και ειδικούς χαρακτήρες. Το μέγιστο μέγεθος καθορίζεται στις παρενθέσεις.
number(size)	Περιέχει έναν αριθμό, όπου ο μέγιστος αριθμός των ψηφίων καθορίζεται στις παρενθέσεις.
number(size, d)	Περιέχει έναν αριθμό, όπου ο μέγιστος αριθμός των ψηφίων καθορίζεται στο size και ο μέγιστος αριθμός των ψηφίων στα δεξιά της υποδιαστολής καθορίζεται στο d.
date	Περιέχει μια ημερομηνία.

Μάθημα 26 - Διαγραφή Βάσης Δεδομένων και Πίνακα

Για να διαγράψουμε μια βάση δεδομένων, χρησιμοποιούμε την εξής εντολή :

DROP DATABASE όνομα_βάσης_δεδομένων

Για να διαγράψουμε έναν πίνακα, χρησιμοποιούμε την εξής εντολή :

DROP TABLE όνομα_πίνακα

Για να διαγράψουμε τα δεδομένα ενός πίνακα χωρίς να διαγράψουμε τον πίνακα, χρησιμοποιούμε την εξής εντολή :

DELETE TABLE όνομα_πίνακα

Μάθημα 27 - Η Εντολή Alter Table

Η εντολή ALTER TABLE χρησιμοποιείται για να προσθέσουμε ή να διαγράψουμε στήλες από έναν υπάρχοντα πίνακα.

Η σύνταξή της για τις δύο αυτές περιπτώσεις είναι ως εξής :

ALTER TABLE όνομα_πίνακα ADD όνομα_στήλης τύπος_δεδομένων
ALTER TABLE όνομα_πίνακα DROP όνομα_στήλης

Πίνακας Person :

LastName	FirstName	Address
Σταύρου	Μαρίνα	Σαρανταπόρου 10

Παράδειγμα

Για να προσθέσουμε μια στήλη με όνομα "City" στον πίνακα "Person", δίνουμε την εξής εντολή :

ALTER TABLE Person ADD City varchar(30)

Αποτέλεσμα :

LastName	FirstName	Address	City
Σταύρου	Μαρίνα	Σαρανταπόρου 10	

Παράδειγμα

Για να διαγράψουμε (drop) τη στήλη "Address" του πίνακα "Person", δίνουμε την εξής εντολή :

ALTER TABLE Person DROP Address

Αποτέλεσμα :

LastName	FirstName	City
Σταύρου	Μαρίνα	

Μάθετε την Access με Απλές Ερωτήσεις

ΠΙΝΑΚΕΣ (Tables)

Τι είναι το Πρωτεύον Κλειδί (Primary Key) ενός πίνακα της Access;

Το Πρωτεύον Κλειδί είναι ένα πεδίο ενός πίνακα της Access, που χαρακτηρίζει μοναδικά μία εγγραφή μέσα σ' ολόκληρο τον πίνακα. Δηλ., δεν μπορεί να υπάρχουν δύο ή περισσότερες εγγραφές που να έχουν ίδια τιμή στο πρωτεύον κλειδί ενός πίνακα.

Ακόμη, οι εγγραφές του πίνακα ταξινομούνται αυτόματα με βάση το πρωτεύον κλειδί. Σ' έναν πίνακα, μπορούμε να ορίσουμε σαν πρωτεύον κλειδί και έναν συνδυασμό δύο ή περισσοτέρων πεδίων, όταν ένα πεδίο μόνο του δεν μπορεί να ορίσει μοναδικά μια εγγραφή. Για παράδειγμα, σ' έναν πίνακα μαθητών μπορούμε να ορίσουμε σαν πρωτεύον κλειδί τα πεδία επώνυμο, όνομα και πατρώνυμο μαζί, όταν είμαστε βέβαια απόλυτα σίγουροι ότι δεν υπάρχουν δύο ή περισσότεροι μαθητές με κοινά αυτά τα τρία πεδία.

Τι είναι το Ευρετήριο (Index) και ποια η χρησιμότητά του;

Το Ευρετήριο είναι μια ιδιότητα που δίνουμε σ' όσα πεδία του πίνακά μας θέλουμε, και το οποίο είναι χρήσιμο όταν ο πίνακας έχει πολλές εγγραφές και θέλουμε να κάνουμε γρήγορη αναζήτηση κάποιων στοιχείων.

Για τα πεδία που έχουμε ορίσει να έχουν δικό τους ευρετήριο, η Access δημιουργεί μόνη της έναν κατάλογο που δεν τον βλέπουμε εμείς, όπου σε κάθε γραμμή του καταλόγου υπάρχει ο αριθμός της εγγραφής και η τιμή του πεδίου. Ο κατάλογος αυτός ταξινομείται αυτόματα βάσει της τιμής του πεδίου. Όταν η Access κάνει αναζήτηση, για να βρει π.χ. ποιοι πελάτες έχουν το επώνυμο "Παπαδόπουλος", τότε, εφ' όσον, βέβαια, έχουμε ορίσει το πεδίο επώνυμο να έχει ευρετήριο δικό του, η Access ψάχνει πολύ γρήγορα στο ταξινομημένο ευρετήριο και βρίσκει όσους πελάτες έχουν το επώνυμο "Παπαδόπουλος".

Για τα πεδία που έχουμε ορίσει να έχουν ευρετήριο, μπορούμε να πούμε στην Access να ελέγξει αν το πεδίο αυτό θα έχει μοναδικές τιμές (no duplicates) ή αν θα έχει πολλές ίδιες τιμές (yes duplicates). Για παράδειγμα, σ' έναν πίνακα μαθητών, μπορούμε να έχουμε σαν πρωτεύον κλειδί τον κωδικό μαθητή και να ορίσουμε ευρετήριο, με αποδεκτές πολλαπλές τιμές, για το επώνυμο και ευρετήριο, χωρίς την αποδοχή πολλαπλών τιμών, για τον αριθμό μητρώου μαθητή. Ο αριθμός μητρώου του μαθητή δεν είναι, βέβαια, το πρωτεύον κλειδί, αλλά δεν μπορούν να υπάρχουν δύο ή περισσότεροι μαθητές με τον ίδιο αριθμό μητρώου.

Σε τι διαφέρουν το πρωτεύον κλειδί ενός πίνακα από το ευρετήριο που δεν αποδέχεται πολλαπλές τιμές;

Σ' έναν πίνακα μπορούμε να έχουμε ένα μόνο πρωτεύον κλειδί, το οποίο μπορεί να αποτελείται από ένα ή περισσότερα πεδία, αλλά μπορούμε να έχουμε συγχρόνως και πολλά ευρετήρια πεδίων που δεν αποδέχονται πολλαπλές τιμές. Το πρωτεύον κλειδί είναι εκείνο που χαρακτηρίζει μοναδιαία έναν πίνακα και όχι το ευρετήριο.

Τι είναι ο Κανόνας Επικύρωσης (Validation Rule) και το Κείμενο Επικύρωσης (Validation Text) και ποια η χρησιμότητά τους;

Ο Κανόνας Επικύρωσης είναι προαιρετικός, μπορεί να οριστεί για κάποιο πεδίο και ελέγχει αν τα δεδομένα που εισάγουμε ανήκουν σε κάποια έγκυρη περιοχή τιμών. Αν γράψουμε κάτι που είναι εκτός της περιοχής τιμών που ορίζει ο κανόνας επικύρωσης, τότε η

Access δεν μας αφήνει να προχωρήσουμε, αν προηγουμένως δεν γράψουμε μια αποδεκτή τιμή για το πεδίο.

Για παράδειγμα, αν καταχωρούμε τους βαθμούς μαθητών, τότε ο κανόνας επικύρωσης που θα πρέπει να δώσουμε, είναι ο : >= 0 and <=20 ή between 0 and 20. Αν, βέβαια, ο σωστός βαθμός είναι ο 15 και εμείς γράψουμε 17, τότε αυτό δεν θα μπορέσει να το ελέγξει η Access.

Αν εισάγουμε στοιχεία για πελάτες, και αυτοί προέρχονται μόνο από τις πόλεις Αθήνα, Θεσ/νίκη και Πάτρα, τότε ο κανόνας επικύρωσης θα είναι ο : 'Αθήνα' or 'Θεσ/νίκη' or 'Πάτρα'.

Το Κείμενο Επικύρωσης είναι ένα μήνυμα που εμφανίζει η Access σ' ένα παράθυρο, όταν παραβούμε τον κανόνα επικύρωσης ενός πεδίου. Μπορούμε να γράψουμε ένα δικό μας μήνυμα, που θα υπενθυμίζει στον χρήστη ποιες είναι οι αποδεκτές τιμές.

Η χρησιμότητα του κανόνα επικύρωσης είναι ότι απλά, περιορίζει τα λάθη που μπορεί να κάνουμε κατά την πληκτρολόγηση.

Τι είναι, πώς ορίζεται και ποια είναι η χρησιμότητα μιας Σχέσης (Relationship) ανάμεσα σε δύο πίνακες της Access;

Είναι μια αντιστοίχηση ανάμεσα σ' ένα πεδίο ενός πίνακα, μ' ένα πεδίο ενός άλλου πίνακα. Τα πεδία αυτά πρέπει να είναι του ιδίου τύπου δεδομένων και του ιδίου μεγέθους. Το ένα είναι το πρωτεύον κλειδί στον έναν πίνακα και το άλλο είναι ένα απλό πεδίο στον άλλον πίνακα.

Για να ορίσουμε μια σχέση ανάμεσα σε δύο πίνακες, ανοίγουμε το παράθυρο Show Relationships της Βάσης Δεδομένων μας και επιλέγουμε (προσθέτουμε) τους πίνακες που θέλουμε να συσχετίσουμε. Μετά, πατάμε με το ποντίκι στο πεδίο του ενός πίνακα, το μεταφέρουμε και το αφήνουμε πάνω στο αντίστοιχο πεδίο του άλλου πίνακα. Η Access εμφανίζει τότε μια γραμμή που συνδέει τους δύο πίνακες και που έχει το σύμβολο 1 στο πρωτεύον πεδίο του ενός πίνακα και το σύμβολο του άπειρου στο αντίστοιχο πεδίο του άλλου πίνακα. Αυτό σημαίνει ότι αυτή είναι μια σχέση ένα προς πολλά (one to many).

Αν κάνουμε κλικ με το ποντίκι πάνω στη γραμμή της σχέσης, μπορούμε να την καταργήσουμε πατώντας το πλήκτρο <delete>, ενώ αν κάνουμε διπλό κλικ πάνω της, εμφανίζεται ένα παράθυρο, όπου μπορούμε να επιλέξουμε, αν θέλουμε βέβαια, άλλα πεδία από τους δύο πίνακες για τη διασύνδεση των δύο πινάκων. Ακόμη, μπορούμε να ορίσουμε αν θα υπάρχει ακεραιότητα αναφοράς και αν θα ισχύει η διαδοχική ενημέρωση ή και η διαδοχική διαγραφή.

Η χρησιμότητα μιας σχέσης ανάμεσα σε δύο πίνακες, είναι ότι μέσω αυτής μπορούμε να αντλήσουμε στοιχεία και πληροφορίες που ανήκουν και στους δύο πίνακες. Για παράδειγμα, αν έχω έναν πίνακα πελατών και έναν πίνακα παραγγελιών που κάνουν οι πελάτες μου, τότε ορίζω το πεδίο κωδικός πελάτη (ID_?aeάτη) να υπάρχει και στους δύο πίνακες, στον πίνακα πελατών σαν πρωτεύον κλειδί βέβαια, και διασυνδέω τους δύο πίνακες με το πεδίο αυτό.

Μετά, μπορώ να ζητήσω πληροφορίες, όπως π.χ. ποιοι πελάτες παρήγγειλαν κάποιο προϊόν αυτή τη χρονιά, ποιοι πελάτες δεν παρήγγειλαν κανένα προϊόν τον τελευταίο μήνα κ.ά. Η Access χρησιμοποιεί τον κωδικό του πελάτη σαν το στοιχείο που διασυνδέει τους δύο πίνακες και εμφανίζει τις πληροφορίες στην οθόνη σαν να ανήκαν αυτές στον ίδιο πίνακα.

Τι σημαίνει η Ακεραιότητα Αναφοράς (Referential Integrity);

Η ακεραιότητα αναφοράς αναφέρεται σε δύο συσχετισμένους πίνακες και σημαίνει πρακτικά ότι δεν μπορώ να καταχωρήσω στοιχεία στον πίνακα μιας σχέσης που είναι στη

μεριά του "πολλά", αν προηγουμένως δεν έχω καταχωρήσει την αντίστοιχη τιμή στο πεδίο της σχέσης που είναι στη μεριά του "ένα".

Δηλ., αν έχω ορίσει να υπάρχει ακεραιότητα αναφοράς στη σχέση του πίνακα πελατών με τον πίνακα παραγγελιών, δεν μπορώ να καταχωρήσω μια παραγγελία για έναν πελάτη, αν προηγουμένως δεν έχω γράψει τον κωδικό του πελάτη στον πίνακα πελατών.

Ένα άλλο παράδειγμα. Σε μια Υπηρεσία Μεταφορών-Επικοινωνιών, μπορούμε να ορίσουμε έναν πίνακα υπαλλήλων και έναν πίνακα αδειών κυκλοφορίας αυτοκινήτων, όπου υπάρχει η σχέση ένα προς πολλά, αφού ένας υπάλληλος μπορεί να γράψει πολλές άδειες κυκλοφορίας, αλλά μια άδεια κυκλοφορίας γράφεται μόνο από έναν υπάλληλο. Εφ' όσον υπάρχει ακεραιότητα αναφοράς, για να γράψω μια καινούργια άδεια κυκλοφορίας, πρέπει να δώσω και τον κωδικό του υπαλλήλου που κάνει την καταχώρηση.

Αν αυτός ο κωδικός, όμως, δεν υπάρχει στον πίνακα των υπαλλήλων, η Access δεν θα με αφήσει να καταχωρήσω την άδεια κυκλοφορίας. Αν καταργήσω την ακεραιότητα αναφοράς, τότε θα μπορέσω να καταχωρήσω τα στοιχεία της νέας άδειας κυκλοφορίας, είτε ο κωδικός του υπαλλήλου που δίνω αντιστοιχεί σε κάποιον υπάλληλο στον πίνακα υπαλλήλων, είτε όχι.

Τι σημαίνει η Διαδοχική Ενημέρωση (Cascade Update);

Σημαίνει ότι, αν κάνω κάποια αλλαγή στο πεδίο που συνδέει δύο πίνακες σε μια σχέση, στον πίνακα που έχει τη σχέση "ένα", τότε ενημερώνονται αυτόματα όλες οι εγγραφές που περιέχουν αυτό το πεδίο στον πίνακα που έχει τη σχέση "πολλά".

Για παράδειγμα, αν έχω ορίσει να ισχύει η διαδοχική ενημέρωση στη σχέση του πίνακα πελατών με τον πίνακα παραγγελιών και αλλάξω τον κωδικό ενός πελάτη, τότε αυτόματα ενημερώνονται στον πίνακα παραγγελιών, όλες οι παραγγελίες που έκανε ο συγκεκριμένος πελάτης, όπου αντικαθίσταται ο παλιός κωδικός με τον καινούργιο κωδικό.

Τι σημαίνει η Διαδοχική Διαγραφή (Cascade Delete);

Σημαίνει ότι, αν σε μια σχέση που συνδέει δύο πίνακες, διαγράψω μια εγγραφή στον πίνακα που έχει τη σχέση "ένα", τότε διαγράφονται αυτόματα όλες οι εγγραφές που περιέχουν αυτό το πεδίο στον πίνακα που έχει τη σχέση "πολλά".

Για παράδειγμα, αν έχω ορίσει να ισχύει η διαδοχική διαγραφή στη σχέση του πίνακα πελατών με τον πίνακα παραγγελιών και διαγράψω έναν πελάτη, τότε αυτόματα διαγράφονται στον πίνακα παραγγελιών, όλες οι παραγγελίες που έκανε ο συγκεκριμένος πελάτης. Σημειώστε ότι, στην Access δεν μπορώ να διαγράψω ένα πεδίο, αλλά μια ολόκληρη εγγραφή.

Τι σημαίνει η σχέση ένα προς πολλά (one to many);

Σημαίνει ότι, σε μια εγγραφή του ενός πίνακα, αντιστοιχούν πολλές εγγραφές του άλλου πίνακα, αλλά σε μια εγγραφή του δεύτερου πίνακα, αντιστοιχεί μία μόνο εγγραφή του πρώτου πίνακα.

Παραδείγματα :

Ένας πελάτης κάνει πολλές παραγγελίες, αλλά μια παραγγελία ανήκει μόνο σ' έναν πελάτη.

Σ' ένα νοσοκομείο, ένας γιατρός μπορεί να παρακολουθεί πολλούς ασθενείς, αλλά ένας ασθενής παρακολουθείται μόνο από έναν γιατρό.

Ένας υπάλληλος σε μια διεύθυνση συγκοινωνιών καταχωρεί πολλά διπλώματα, αλλά ένα δίπλωμα καταχωρείται μόνο από έναν υπάλληλο.

Τι σημαίνει η σχέση πολλά προς πολλά (many to many);

Σημαίνει ότι, σε μια εγγραφή του ενός πίνακα, αντιστοιχούν πολλές εγγραφές του άλλου πίνακα, αλλά και σε μια εγγραφή του δεύτερου πίνακα, αντιστοιχούν πολλές εγγραφές του πρώτου πίνακα.

Παραδείγματα :

Ένας καθηγητής διδάσκει σε πολλούς μαθητές, αλλά και ένας μαθητής διδάσκεται από πολλούς καθηγητές.

Ένας ιδιοκτήτης μπορεί να έχει στην κατοχή του πολλά ακίνητα, αλλά και ένα ακίνητο μπορεί να ανήκει σε πολλούς ιδιοκτήτες.

Ένας αθλητής παίρνει μέρος σε πολλούς αγώνες, αλλά και σ' έναν αγώνα παίρνουν μέρος πολλοί αθλητές.

Τι σημαίνει η σχέση ένα προς ένα (one to one);

Σημαίνει ότι, σε μια εγγραφή του ενός πίνακα, αντιστοιχεί μία μόνο εγγραφή του άλλου πίνακα, και αλλά σε μια εγγραφή του δεύτερου πίνακα, αντιστοιχεί μία μόνο εγγραφή του πρώτου πίνακα.

Παραδείγματα :

Ένας οδηγός μπορεί να έχει ένα μόνο δίπλωμα οδήγησης, αλλά και ένα δίπλωμα οδήγησης ανήκει μόνο σ' έναν οδηγό.

Ένα αυτοκίνητο μπορεί να έχει μόνο μία μηχανή, αλλά και μία μηχανή ανήκει μόνο σ' ένα αυτοκίνητο.

Ένας νομός μπορεί να έχει μόνο μία πόλη σαν πρωτεύουσα, αλλά και μία πόλη μπορεί να είναι πρωτεύουσα μόνο ενός νομού.

Η Access υποστηρίζει μόνο σχέσεις ένα προς πολλά. Αν, όμως, έχω μια σχέση πολλά προς πολλά, τότε τι πρέπει να κάνω;

Η λύση είναι η δημιουργία ενός τρίτου πίνακα, ο οποίος θα περιέχει τα πρωτεύοντα κλειδιά των δύο άλλων πινάκων. Οι σχέσεις που θα δημιουργηθούν έτσι, μεταξύ των δύο αρχικών πινάκων και του τρίτου πίνακα, θα είναι σχέσεις ένα προς πολλά.

Στα παραδείγματα που υπάρχουν παραπάνω, θα μπορούσαμε π.χ. για τη σχέση των ιδιοκτητών με τα ακίνητα, να ορίσουμε έναν τρίτο πίνακα, με πεδία τον κωδικό ιδιοκτήτη, τον κωδικό ακινήτου και το ποσοστό που έχει ο ιδιοκτήτης σ' αυτό το οικόπεδο. Έτσι, ο πίνακας των ιδιοκτητών και ο πίνακας των ακινήτων θα έχουν μια σχέση ένα προς πολλά με τον πίνακα αυτόν.

Η Access υποστηρίζει μόνο σχέσεις ένα προς πολλά. Αν, όμως, έχω μια σχέση ένα προς ένα, τότε τι πρέπει να κάνω;

Πολύ απλά, μπορούμε να δημιουργήσουμε έναν μόνο πίνακα, που θα περιέχει τις εγγραφές και των δύο πινάκων ενωμένες μία-μία. Π.χ. για την περίπτωση του πίνακα των νομών και του πίνακα των πόλεων, θα μπορούσαμε να δημιουργήσουμε έναν νέο πίνακα, όπου η κάθε εγγραφή θα περιέχει τα στοιχεία του νομού και τα στοιχεία της πόλης που είναι πρωτεύουσα του νομού.

Τι είναι ένα Ερώτημα (Query);

Είναι μια διαδικασία της Access που μας επιτρέπει να φιλτράρουμε (απομονώνουμε) κάποια πεδία ή/και κάποιες εγγραφές από έναν ή περισσότερους πίνακες. Οι νέες

πληροφορίες εμφανίζονται στην οθόνη σαν ένας καινούργιος πίνακας. Ο παλιός πίνακας παραμένει άθικτος. Αυτό λέγεται Ερώτημα Επιλογής (Select Query).

Για παράδειγμα, μπορεί από τον πίνακα των πελατών να θέλω να δω μόνο εκείνους που μένουν στην Αθήνα ή στην Πάτρα. Μπορεί, ακόμη, να θέλω να δω μόνο εκείνους τους πελάτες που χρωστάνε πάνω από 100.000 δρχ.

Μπορώ, όμως, να κάνω και συνδυασμούς. Π.χ. μπορεί να θέλω να δω μόνο εκείνους τους πελάτες που μένουν στη Θεσ/νίκη και χρωστάνε από 100.000 έως 300.000 δρχ.

Μπορεί, όμως, να θέλω να δω στην οθόνη μου μόνο τα πεδία επώνυμο και όνομα απ' όλους τους πελάτες, χωρίς να βάλω κάποιον περιορισμό.

Τι είναι τα Κριτήρια της Access και πώς κάνω συνδυασμούς Κριτηρίων;

Τα κριτήρια είναι οι περιορισμοί που θέτω στις τιμές των πεδίων για να φιλτράρω (απομονώσω) τα αποτελέσματα, όπως τα θέλω. Μπορώ να κάνω συνδυασμούς κριτηρίων με τους τελεστές And και Or. Ο τελεστής And δεν φαίνεται στην οθόνη, αλλά η χρήση του εννοείται όταν γράφουμε κριτήρια σε διάφορα πεδία στη γραμμή Criteria.

Για να λειτουργήσει ο τελεστής Or πρέπει να γράψω τα κριτήριά μου στη γραμμή Or, που είναι κάτω από τη γραμμή Criteria. Μπορεί, όμως, στη γραμμή Criteria και μέσα σ' ένα πεδίο, να χρησιμοποιήσω τους τελεστές And και Or για να θέσω ό,τι κριτήρια θέλω.

Μπορώ να χρησιμοποιήσω πολλούς τελεστές για να προσδιορίσω τα κριτήριά μου. Η χρήση τους θα φανεί καλύτερα με παραδείγματα :

LIKE 'Αθήνα' ® να είναι ίσο με 'Αθήνα'

LIKE 'Α*' ® να αρχίζει με 'Α'

LIKE '*ίδης' ® να τελειώνει σε 'ίδης'

LIKE '*α*' ® να έχει το 'α' ενδιάμεσα

LIKE '[ΑΒΓ]*' ® να αρχίζει μ' ένα από τα γράμματα ΑΒΓ

IN ('Αθήνα, 'Θεσ/νίκη', 'Πάτρα') ® να είναι μια από τις τρεις πόλεις 'Αθήνα' Or 'Ηράκλειο' ® να είναι ίσο με 'Αθήνα' ή 'Ηράκλειο'>20 and <40 ® να είναι μεταξύ 20 και 40, χωρίς να περιέχει το 20 ή το 40 between 10 and 30 ® να είναι μεταξύ 10 και 30, περιέχοντας το 10 και το 30 <= 100 Or > 200 ® μικρότερο ή ίσο του 100 ή μεγαλύτερο του 200

LIKE '??α*' ® να έχει δύο χαρακτήρες στη αρχή, μετά το 'α' και μετά ο,τιδήποτε

LIKE '##00' ® ένας τετραψήφιος αριθμός που τελειώνει σε 00

LIKE '[!ΑΒ]*' ® να μην αρχίζει από τα γράμματα Α ή Β και μετά να έχει ο,τιδήποτε

Τι σημαίνει η ένδειξη Show στα Ερωτήματα της Access;

Σημαίνει ότι, το πεδίο που έχει την ένδειξη Show σημαδεμένη (Χ), θα εμφανιστεί στην οθόνη του ερωτήματος. Αν δεν είναι σημαδεμένη η ένδειξη, τότε δεν θα εμφανιστεί όταν εκτελέσουμε το ερώτημα.

Υπάρχει, όμως, κάποιος λόγος να μην θέλω να εμφανιστεί ένα πεδίο σ' ένα ερώτημα, εφόσον αυτό το πεδίο το έχω τοποθετήσει στο πλέγμα του ερωτήματος και έχω βάλει κάποια κριτήρια σ' αυτό; Και βέβαια.

Δείτε το παρακάτω παράδειγμα : από τον πίνακα των αθλητών θέλω να δημιουργήσω ένα ερώτημα και να δω μόνο εκείνους τους αθλητές που είναι από την Ελλάδα. Θα πρέπει να επιλέξω το πεδίο χώρα και να βάλω το κριτήριο LIKE 'Ελλάδα'. Τότε, όμως, όταν θα εκτελεστεί το ερώτημα, θα βλέπω σ' όλους τους αθλητές και στη στήλη χώρα τη λέξη Ελλάδα.

Αυτό, όμως, είναι περιττό, αφού επέλεξα μόνο τους Έλληνες αθλητές. Έτσι, μπορώ να επιλέξω να είναι απενεργοποιημένη η ένδειξη Show για το πεδίο χώρα. Το ερώτημα αυτό μπορώ να το αποθηκεύσω και να το ονομάσω Έλληνες Αθλητές.

Τι σημαίνει η ένδειξη Totals στα Ερωτήματα της Access;

Σημαίνει ότι μπορώ να χρησιμοποιήσω κάποιες αριθμητικές συναρτήσεις για να κάνω υπολογισμούς σε πεδία ενός ερωτήματος. Οι σημαντικότερες από τις συναρτήσεις αυτές είναι οι : Sum (Άθροισμα), Count (Μέτρηση), Avg (Μέσος Όρος), Min (Εύρεση μικρότερου) και Max (Εύρεση μεγαλύτερου).

Για να μπορέσω να χρησιμοποιήσω σωστά αυτές τις συναρτήσεις, πρέπει πρώτα να έχω χρησιμοποιήσει την Ομαδοποίηση (Group by). Η ομαδοποίηση ανήκει κι' αυτή στην ομάδα Totals.

Τι σημαίνει η Ομαδοποίηση (Group by) στα Ερωτήματα της Access;

Σημαίνει απλά, ότι η Access συγκεντρώνει μαζί όλες εκείνες τις εγγραφές που έχουν ίδια τιμή, στο πεδίο όπου ορίζω την ομαδοποίηση.

Για παράδειγμα, για να δω πόσες παραγγελίες έχει κάνει ο κάθε πελάτης, ομαδοποιώ (group by) το πεδίο κωδικός πελάτη και χρησιμοποιώ τη συνάρτηση count στο πεδίο κωδικός παραγγελίας.

Για να δω ποιους πελάτες παρακολουθεί ο κάθε πωλητής μου και πόση αξία έχουν οι παραγγελίες του κάθε πελάτη, ομαδοποιώ τον κωδικό πωλητή και τον κωδικό πελάτη και χρησιμοποιώ τη συνάρτηση sum στο πεδίο αξία παραγγελίας.

Τι είναι τα Ερωτήματα Δράσης (Action Queries);

Είναι ερωτήματα της Access με τα οποία μπορώ να κάνω αλλαγές στα στοιχεία των πινάκων όπου εφαρμόζονται αυτά τα ερωτήματα. Για να εφαρμοστούν αυτές οι αλλαγές, μπορώ να ορίσω και κάποια κριτήρια για να μην ισχύσουν οι αλλαγές για όλες τις εγγραφές του πίνακα.

Υπάρχουν *Ερωτήματα Ενημέρωσης (Update Queries)*, όπου μπορώ να αλλάξω την τιμή σε κάποια πεδία του πίνακά μου, *Ερωτήματα Διαγραφής (Delete Queries)*, όπου μπορώ να διαγράψω κάποιες εγγραφές του πίνακα, *Ερωτήματα Προσθήκης (Add Queries)*, όπου μπορώ να προσθέσω στοιχεία ενός πίνακα σ' έναν άλλον πίνακα και *Ερωτήματα Δημιουργίας Πίνακα (Make Table Queries)*, όπου μπορώ να δημιουργήσω έναν καινούργιο πίνακα από τα στοιχεία ενός άλλου πίνακα.

Πώς μπορώ να κάνω υπολογισμούς πάνω σε πεδία της Access;

Για να κάνω υπολογισμούς στα πεδία της Access, χωρίς, όμως, να αλλάξω τα αρχικά στοιχεία ενός πίνακα, πρέπει να δημιουργήσω ένα ερώτημα και σε μια στήλη του ερωτήματος να γράψω το πεδίο, όπου θα γίνουν οι υπολογισμοί, μέσα σε αγκύλες.

Για παράδειγμα, για να δω πόσο θα αλλάξουν οι μισθοί των υπαλλήλων μιας εταιρείας, αν αυτοί αυξηθούν κατά 10%, θα πρέπει να δημιουργήσω ένα ερώτημα και να γράψω σε κάποια στήλη την έκφραση :

*[μισθός] * 1.1*

Μπορώ να δω το αποτέλεσμα στην οθόνη, αλλά δεν μπορώ να αλλάξω μ' αυτόν τον τρόπο τις τιμές των μισθών.

Μπορώ, ακόμη, να γράψω και εκφράσεις που θα περιέχουν περισσότερα από ένα πεδία :

*[επίδομα γάμου] + [επίδομα παραγωγικότητας] * 1.2*

Για να μπορέσω, όμως, να αλλάξω τις τιμές κάποιων πεδίων της Access, πρέπει να χρησιμοποιήσω είτε τα Ερωτήματα Ενημέρωσης (Update Queries) ή να γράψω μια υπομονάδα εντολών (module).

ΦΟΡΜΕΣ (Forms)

Τι είναι οι Φόρμες της Access και ποια η χρησιμότητά τους;

Οι φόρμες της Access είναι ένας όμορφος τρόπος απεικόνισης των περιεχομένων των Πινάκων (Tables) ή των Ερωτημάτων (Queries) μιας Βάσης Δεδομένων. Μια φόρμα αναφέρεται πάντα σ' έναν πίνακα (table) ή σ' ένα ερώτημα (query) της Access απ' όπου και παίρνει τα δεδομένα που απεικονίζει στην οθόνη. Σ' έναν πίνακα μπορούμε να αντιστοιχίσουμε όσες φόρμες θέλουμε, δηλ. διαφορετικούς τρόπους εμφάνισης των δεδομένων μας.

Σε μια φόρμα μπορούμε να βάλουμε δικούς μας τίτλους (επικεφαλίδες) και να τοποθετήσουμε τα πεδία σε όμορφα έγχρωμα πλαίσια, με όποιες γραμματοσειρές και σε όποιο μέγεθος γραμμάτων θέλουμε εμείς.

Μπορούμε να αλλάξουμε το χρώμα του φόντου, των γραμμάτων ή του περιθωρίου σε κάθε πλαίσιο πεδίου και να προσθέσουμε και ειδικά εφέ. Ακόμα, μπορούμε να ζωγραφίσουμε μεμονωμένα πλαίσια και ορθογώνια μέσα στη φόρμα.

Σε μια φόρμα εμφανίζεται συνήθως μία εγγραφή ανά οθόνη και με τα πλήκτρα PageUp και PageDown μπορούμε να μετακινηθούμε από εγγραφή σε εγγραφή. Κατά τα λοιπά, ισχύουν όλοι οι περιορισμοί και οι κανόνες εγκυρότητας που είχαμε θέσει όταν δημιουργήσαμε τον πίνακα στον οποίο βασίζεται η φόρμα.

Ό,τι καταχωρίσεις και διορθώσεις κάνουμε στη φόρμα, θα μπορούμε να τις δούμε και στην άποψη φύλλου δεδομένων (datasheet) του πίνακα και το αντίθετο. Απλούστατα, με τη φόρμα έχουμε έναν ωραίο τρόπο εμφάνισης των περιεχομένων ενός πίνακα, αλλά και άλλα πολλά πλεονεκτήματα.

Ποια είναι τα πλεονεκτήματα που έχει η χρήση των φορμών;

Σε μια φόρμα μπορούμε να εμφανίσουμε τιμές που προκύπτουν από υπολογισμούς των τιμών κάποιων πεδίων του πίνακα στον οποίο αναφέρεται η φόρμα. Για παράδειγμα, μπορεί να θέλουμε να βλέπουμε τον μέσο όρο των βαθμών ενός μαθητή σε μια φόρμα που αναφέρεται σε μαθητές. Δεν θα ήταν, βέβαια, σωστό να δημιουργήσουμε ένα πεδίο, όπου θα υπολογίζαμε και θα καταχωρούσαμε εμείς τον μέσο όρο, αφού ο μέσος όρος προκύπτει από υπολογισμό πάνω στις τιμές κάποιων πεδίων του πίνακα.

Για να το κάνουμε αυτό, θα πρέπει να δημιουργήσουμε ένα ειδικό χειριστήριο (control box) με το εργαλείο πλαισίου κειμένου (ab½), όπου θα γράψουμε τον τύπο : *([βαθμός-1]+[βαθμός-2]+...)/10*, αν ο μαθητής έχει βαθμούς σε δέκα μαθήματα. Ο μέσος όρος θα υπολογίζεται τότε και θα εμφανίζεται σε κάθε εγγραφή μαθητή, χωρίς να αποτελεί ξέχωρο πεδίο.

Σε μια φόρμα μπορούμε να εμφανίσουμε και άλλη μια ή περισσότερες υποφόρμες, δηλ. φόρμες μέσα σε φόρμα, που είναι πάρα πολύ χρήσιμες για να υπάρχει άμεση ενημέρωση όταν έχουμε συσχετισμένους πίνακες "ένα προς πολλά". Περισσότερα για τις υποφόρμες σε παρακάτω ερώτηση.

Μπορούμε, ακόμα, να εμφανίσουμε και εικόνες, ζωγραφιές, φωτογραφίες, ήχους ή και κινούμενες εικόνες (video) από άλλα προγράμματα των Windows μέσα σε ειδικά πλαίσια της φόρμας. Όλα αυτά λέγονται Αντικείμενα ΣΕΑ και περισσότερα γι' αυτά παρακάτω.

Οι φόρμες είναι πολύ χρήσιμες όταν κάποια πεδία παίρνουν τιμές από μια συγκεκριμένη περιοχή τιμών. Για παράδειγμα, αν οι πελάτες μιας εταιρείας προέρχονται κατά το

μεγαλύτερο μέρος τους από τις πόλεις Θεσ/νίκη, Κατερίνη και Λάρισα, τότε, μπορούμε σε μια φόρμα να εμφανίσουμε ένα πλαίσιο στο πεδίο πόλη, όπου θα υπάρχουν οι τρεις αυτές τιμές και θα μπορούμε να επιλέγουμε όποια τιμή θέλουμε, κάνοντας απλά κλικ πάνω της με το ποντίκι.

Αν, βέβαια, ο πελάτης είναι από μια πόλη που δεν ανήκει στη λίστα αυτή, τότε η Access μάς δίνει τη δυνατότητα να καταχωρίσουμε και τιμές εκτός λίστας. Με τον τρόπο αυτό, όμως, γλυτώνουμε από πληκτρολόγηση και αποφεύγουμε και τα λάθη. Περισσότερα, παρακάτω, στην παράγραφο Κατάλογοι και Σύνθετα Πλαίσια.

Το μόνο μειονέκτημα που έχουν οι φόρμες, είναι ότι δεν μπορούμε να δούμε ταυτόχρονα στην οθόνη πολλές εγγραφές μαζί, κάτι που μπορεί να γίνει με την προβολή φύλλου δεδομένων (datasheet).

Τι είναι η Εργαλειοθήκη και τι το Φύλλο Ιδιοτήτων;

Η **εργαλειοθήκη (toolbox)** είναι μια συλλογή εργαλείων με τα οποία μπορούμε να δημιουργήσουμε δεσμευμένα ή αδέσμευτα πλαίσια κειμένου, ομάδες επιλογών, κουμπιά εντολών, πλαίσια καταλόγου, σύνθετα πλαίσια, εικόνες, υποφόρμες, γραμμές και ορθογώνια.

Οι **φύλλο ιδιοτήτων (properties)** περιέχει όλες τις ιδιότητες ενός αντικειμένου και μπορεί να αναφέρεται σ' ολόκληρη τη φόρμα, στο τμήμα λεπτομερειών της (details), σ' ένα χειριστήριο, σ' ένα πλαίσιο, σ' ένα κουμπί εντολής και γενικά σε οποιοδήποτε αντικείμενο. Οι ιδιότητες που εμφανίζονται στο φύλλο ιδιοτήτων, χωρίζονται σε τέσσερις κατηγορίες : Format (Εμφάνιση), Data (Δεδομένα), Event (Συμβάντα), Other (Διάφορα Άλλα) και η επιλογή All (Όλα) έχει όλες τις ιδιότητες συγκεντρωμένες.

Τι είναι οι Ετικέτες και τι τα Πλαίσια Κειμένου;

Σε μια φόρμα, τα κείμενα και οι τιμές των πεδίων εμφανίζονται μέσα σε πλαίσια. Υπάρχουν, όμως, δύο είδη πλαισίων : οι ετικέτες (labels) και τα πλαίσια κειμένου (text boxes).

Οι **ετικέτες** είναι μηνύματα (τίτλοι, επικεφαλίδες, οδηγίες, πληροφορίες), όπου μπορούμε να γράψουμε ό,τι θέλουμε και λέγονται αδέσμευτα, γιατί απλούστατα περιέχουν σταθερό κείμενο και δεν συνδέονται με κάποιο πεδίο του πίνακα, έτσι ώστε να αλλάζουν όταν μετακινούμαστε από εγγραφή σε εγγραφή. Το πλήκτρο (εργαλείο) της εργαλειοθήκης με το οποίο δημιουργούμε ετικέτες είναι αυτό που έχει το γράμμα Α.

Τα **πλαίσια κειμένου** συνδέονται με κάποιο πεδίο ή πεδία του πίνακα στο οποίο βασίζεται η φόρμα. Έτσι, αν αλλάξουμε το περιεχόμενο ενός πλαισίου κειμένου, τότε αλλάζει και η τιμή του πεδίου με το οποίο είναι συνδεδεμένο. Μπορούμε, ακόμη, να χρησιμοποιήσουμε ένα πλαίσιο κειμένου για να υπολογίζουμε τιμές χρησιμοποιώντας αριθμητικές παραστάσεις. Σ' αυτή την περίπτωση, όμως, δεν μπορούμε να αλλάξουμε τιμές, αλλά απλά κάνουμε υπολογισμούς τιμών από τις τιμές άλλων πεδίων. Το πλήκτρο (εργαλείο) της εργαλειοθήκης με το οποίο δημιουργούμε πλαίσια κειμένου είναι αυτό που έχει τα γράμματα **ab½** .

Οι ετικέτες και τα πλαίσια κειμένου αποκαλούνται και **χειριστήρια (control boxes)** και μπορούμε να αλλάξουμε το μέγεθός τους και να τα μετακινήσουμε μαζί ή και ξέχωρα.

Τι είναι το Εργαλείο Ομάδας Επιλογών (Option Group);

Είναι ένα χρήσιμο εργαλείο με το οποίο δημιουργούμε ένα πλαίσιο όπου μπορούμε να τοποθετήσουμε μέσα ένα ή περισσότερα κουμπιά επιλογών (option buttons), πλαίσια ελέγχου (check boxes) ή κουμπιά διακόπτη (toggle buttons). Σε κάθε κουμπί μπορούμε να αντιστοιχίσουμε μια ξεχωριστή αριθμητική τιμή. Τα κουμπιά επιλογών έχουν ένα μαύρο

σημάδι σ' έναν κύκλο όταν είναι επιλεγμένα, τα πλαίσια ελέγχου έχουν ένα σημάδι Ο μέσα σ' ένα τετράγωνο πλαίσιο και τα κουμπιά διακόπτη μοιάζουν με ηλεκτρικό διακόπτη.

Όταν κάνουμε κλικ με το ποντίκι σ' ένα από τα κουμπιά επιλογών, τότε όλη η ομάδα επιλογών παίρνει αυτή την τιμή και ακυρώνονται όλες οι άλλες επιλογές. Την ομάδα αυτή επιλογών την συνδέουμε μ' ένα πεδίο του πίνακα στο οποίο βασίζεται η φόρμα και έτσι το πεδίο αυτό ενημερώνεται με την τιμή που έχουμε επιλέξει από το αντίστοιχο κουμπί επιλογής.

Για παράδειγμα, η εταιρεία κινητής τηλεφωνίας Telestet έχει κατατάξει τους πελάτες της σε τέσσερις κατηγορίες, ανάλογα με το οικονομικό πακέτο που έχουν επιλέξει : economy, business, city και business plus. Ένας πελάτης της Telestet θα ανήκει υποχρεωτικά σε μία από τις τέσσερις παραπάνω κατηγορίες.

Για να διευκολύνουμε, λοιπόν, την καταχώριση των στοιχείων των πελατών και για να μην κάνουμε λάθη κατά την πληκτρολόγηση, μπορούμε να δημιουργήσουμε μια ομάδα επιλογών, όπου θα τοποθετήσουμε τέσσερα κουμπιά επιλογών, που το καθένα θα αντιστοιχεί σε μια από τις κατηγορίες πελατών της Telestet. Έτσι, πατώντας με το ποντίκι στην κατηγορία που θέλουμε, το αντίστοιχο πεδίο παίρνει αυτόματα την τιμή, χωρίς να χρειαστεί να την πληκτρολογήσουμε. Θα πρέπει να τοποθετήσουμε και τις αντίστοιχες ετικέτες (labels) δίπλα στην κάθε επιλογή.

Τι είναι οι Κατάλογοι και τι τα Σύνθετα Πλαίσια (List Box - Combo Box);

Είναι μια διευκόλυνση που μας παρέχει η Access και που μοιάζει με το εργαλείο ομάδας επιλογών που είδαμε στην προηγούμενη παράγραφο. Και εδώ έχουμε να κάνουμε με επιλογές από μια συγκεκριμένη περιοχή τιμών.

Είδαμε σε μια προηγούμενη παράγραφο ένα παράδειγμα με τις τρεις πόλεις (Θεσ/νίκη - Κατερίνη - Λάρισα), απ' όπου επιλέγουμε αυτήν που θέλουμε. Αν, όμως, κάποιος πελάτης δεν μένει σε μια από τις τρεις αυτές πόλεις, τότε μπορούμε να παρακάμψουμε τις τρεις αυτές επιλογές και να δώσουμε μια καινούργια επιλογή, χωρίς να υπάρχει κανένα απολύτως πρόβλημα.

Υπάρχουν δύο ειδών λίστες τιμών : οι κατάλογοι και τα σύνθετα πλαίσια. Οι διαφορές τους είναι ότι στα σύνθετα πλαίσια υπάρχει ένα πτυσσόμενο πλαίσιο όπου μπορούμε να καταχωρίσουμε και δικές μας τιμές εκτός από τις τιμές του καταλόγου, ενώ στους καταλόγους δεν υπάρχει πτυσσόμενο πλαίσιο και οι τιμές που μπορούμε να επιλέξουμε είναι καθορισμένες.

Δημιουργούμε πρώτα τον κατάλογο ή το σύνθετο πλαίσιο και μετά το συνδέουμε με το πεδίο που θέλουμε. Η λίστα των τιμών που θα ανήκει στο σύνθετο πλαίσιο, μπορεί να προέρχεται είτε από έναν άλλον πίνακα ή να είναι μια λίστα τιμών που θα τη δημιουργήσουμε εμείς. Αν θέλουμε να περιοριστούμε μόνο στις επιλογές της λίστας επιλέγουμε Limit To List - Yes, αλλιώς επιλέγουμε Limit To List - No και μπορούμε να γράψουμε και άλλες τιμές εκτός λίστας.

Παράδειγμα με δική μας λίστα τιμών είδαμε προηγουμένως. Υπάρχει, όμως, και περίπτωση να πάρουμε τιμές από έναν άλλον πίνακα; Και βέβαια. Αν καταχωρούμε τις παραγγελίες των πελατών μας, τότε πρέπει να δίνουμε σε κάθε παραγγελία και τον κωδικό ή το επώνυμο του πελάτη. Δεν μπορούμε να θυμόμαστε, βέβαια, όλους τους κωδικούς ή όλα τα επώνυμα των πελατών.

Έτσι, επιλέγουμε για λίστα τιμών τις τιμές του πίνακα πελατών και στο πλαίσιο όπου θα πρέπει να γράψουμε τον κωδικό του πελάτη μέσα στη φόρμα των παραγγελιών, επιλέγουμε τον πελάτη που θέλουμε από τη λίστα των πελατών, χωρίς να ανησυχούμε αν γράψαμε

σωστά τον κωδικό του ή το επώνυμό του. Σ' αυτή την περίπτωση, η τιμή που θα επιλέξουμε πρέπει οπωσδήποτε να είναι από τη λίστα των πελατών (Limit To List - Yes).

Τι είναι τα Αντικείμενα ΣΕΑ (OLE Objects);

Είναι ειδικά πλαίσια που μπορούμε να ορίσουμε μέσα σε μια φόρμα, όπου μπορούμε να εμφανίσουμε εικόνες, οι οποίες να είναι σταθερές, δηλ. η εικόνα να είναι η ίδια για όλες τις εγγραφές, ή για κάθε εγγραφή του πίνακα να υπάρχει μια ξεχωριστή εικόνα. Αντί για εικόνα, μπορεί να υπάρχει μια ζωγραφιά, ένα εικονίδιο ήχου ή ακόμα και μια καταχώριση video (κινούμενη εικόνα).

Τι είναι τα Κουμπιά Εντολών (Command Buttons);

Είναι ειδικά πλήκτρα στα οποία μπορούμε να αντιστοιχίσουμε μια μακροεντολή ή μια διαδικασία της γλώσσας προγραμματισμού Visual Basic for Applications (VBA) που έχει η Access. Δημιουργούμε αυτό το πλήκτρο μέσα στη φόρμα, του δίνουμε ένα χαρακτηριστικό όνομα και όταν κάνουμε κλικ πάνω του με το ποντίκι, τότε εκτελείται η αντίστοιχη μακροεντολή ή η διαδικασία. Το ποια μακροεντολή ή διαδικασία θα εκτελεστεί, το ορίζουμε στον πίνακα των ιδιοτήτων του κουμπιού εντολών.

Μια μακροεντολή ή μια διαδικασία μπορεί να ανοίγει μια φόρμα ή μια αναφορά, να κάνει μια εκτύπωση ή έναν έλεγχο εγκυρότητας δεδομένων κ.ά.

Τι είναι η Υποφόρμα (SubForm);

Η υποφόρμα είναι μια φόρμα που είναι ενσωματωμένη μέσα σε μια κύρια φόρμα. Για παράδειγμα, αν έχουμε δημιουργήσει μια φόρμα για τους πελάτες μιας εταιρείας και θέλουμε συγχρόνως να βλέπουμε στην οθόνη μας και ποιες παραγγελίες έχει κάνει ο κάθε πελάτης, τότε χρησιμοποιούμε το ειδικό εικονίδιο για τη δημιουργία υποφόρμας, επιλέγουμε τον πίνακα των παραγγελιών και έτσι βλέπουμε στην οθόνη μας για τον κάθε πελάτη και όλες τις παραγγελίες του.

Για να μπορέσουμε να δημιουργήσουμε, βέβαια, μια υποφόρμα, θα πρέπει να υπάρχει μια σχέση "ένα προς πολλά" ανάμεσα στον πίνακα της κύριας φόρμας και στον πίνακα που θα δώσει τα στοιχεία του στην υποφόρμα.

ΑΝΑΦΟΡΕΣ (Reports)

Τι είναι οι Αναφορές της Access και ποια η χρησιμότητά τους;

Οι **αναφορές (reports)** της Access είναι ένας ωραίος τρόπος εκτύπωσης των δεδομένων που υπάρχουν σ' έναν πίνακα (table) ή σ' ένα ερώτημα (query) της Access. Τα δεδομένα ενός πίνακα μπορούμε να τα εκτυπώσουμε και με την αντίστοιχη εντολή (εικονίδιο) που υπάρχει στις επιλογές ενός πίνακα, αλλά με τις αναφορές μπορούμε να κάνουμε πολλά περισσότερα πράγματα, με κυριότερο την ομαδοποίηση και την εμφάνιση αθροισμάτων (sum) για τα αριθμητικά πεδία.

Για να μπορέσουμε να δουλέψουμε πιο αποτελεσματικά με τις αναφορές, είναι πάρα πολύ χρήσιμοι οι **Οδηγοί Αναφορών (Report Wizards)**. Αν, παρ' όλα αυτά, αποφασίσουμε να δημιουργήσουμε μόνοι μας μια αναφορά, θα συναντήσουμε αρκετές δυσκολίες στη δημιουργία της.

Τι σημαίνει και πώς δηλώνεται η Ομαδοποίηση (Grouping);

Η ομαδοποίηση και η ταξινόμηση δηλώνονται μαζί και σημαίνουν ότι θέλουμε η Access να μας εμφανίσει τα δεδομένα στην εκτύπωση ομαδοποιημένα και ταξινομημένα σύμφωνα με κάποιο ή κάποια πεδία. Όταν πατάμε στο κουμπί ομαδοποίησης/ταξινόμησης,

εμφανίζεται ένα παράθυρο όπου μπορούμε να επιλέξουμε το πεδίο ή τα πεδία σύμφωνα με τα οποία θα γίνει η ταξινόμηση και η ομαδοποίηση.

Στο πάνω μέρος του παραθύρου αυτού επιλέγουμε τα πεδία με βάση τα οποία θα γίνει η ταξινόμηση και στο κάτω μέρος υπάρχουν κάποιες επιλογές για το κάθε πεδίο, όπου μπορούμε να επιλέξουμε αν θα υπάρχει **κεφαλίδα ομάδας (group header)** ή/και **υποσέλιδο ομάδας (group footer)**.

Αν επιλέξουμε να υπάρχει στην εκτύπωση κεφαλίδα ομάδας, τότε ουσιαστικά δηλώνουμε στην Access ότι θέλουμε να κάνει και ομαδοποίηση για το συγκεκριμένο πεδίο, εκτός από ταξινόμηση. Ακόμα, μπορούμε να δηλώσουμε αν η ομαδοποίηση θα γίνει με βάση ολόκληρη την τιμή του πεδίου ή με βάση τα αρχικά του.

Σαν να μην κατάλαβα πολύ καλά τι είναι η Ομαδοποίηση. Μήπως μπορώ να έχω ένα παράδειγμα;

Βέβαια. Ας υποθέσουμε ότι έχουμε έναν πίνακα πελατών και έναν πίνακα παραγγελιών που έχουν μεταξύ τους σχέση "ένα προς πολλά". Μπορώ να κάνω ομαδοποίηση ανά πελάτη, δηλ. να εκτυπώσω τους πελάτες με αλφαβητική ταξινόμηση, να έχω μια κεφαλίδα ομάδας με τα στοιχεία του πελάτη (επώνυμο, όνομα, τηλέφωνο κ.ά.) και από κάτω να έχω μια αλφαβητική ή ανά ημερομηνία παρουσίαση των παραγγελιών του πελάτη αυτού. Μόλις τελειώσουν οι παραγγελίες του κάθε πελάτη, μπορώ, αν θέλω, να δημιουργήσω και υποσέλιδο ομάδας, όπου θα εμφανιστούν τα σύνολα των παραγγελιών (συνολική αξία παραγγελιών του πελάτη κ.ά.).

Αν δεν είχα ορίσει να υπάρχει κεφαλίδα ομάδας για το επώνυμο του πελάτη, τότε θα είχε γίνει μόνο η ταξινόμηση ανά επώνυμο πελάτη και όχι και η ομαδοποίηση.

Μπορώ, ακόμη, να κάνω ομαδοποίηση και σύμφωνα με το πρώτο γράμμα του κάθε πελάτη, δηλ. να είναι μαζί αυτοί που αρχίζουν από Α, μετά αυτοί που αρχίζουν από Β, κοκ.

Ακόμα, μπορώ να κάνω ομαδοποίηση με βάση δύο πεδία. Για παράδειγμα, μπορώ να ομαδοποιήσω πρώτα σύμφωνα με την πόλη που μένει ο πελάτης και μετά να ομαδοποιήσω ή απλά να ταξινομήσω σύμφωνα και με το επώνυμό του. Έτσι, θα εμφανιστούν πρώτα οι πόλεις με αλφαβητική σειρά και σε κάθε πόλη θα εμφανιστούν οι πελάτες της πόλης αυτής πάλι με αλφαβητική σειρά. Για κάθε πελάτη μπορώ, αν θέλω, να εκτυπώσω και τις παραγγελίες που έχει κάνει.

Σ' ένα άλλο παράδειγμα, αν έχουμε μια βάση δεδομένων με σχολεία, τότε μπορώ να ομαδοποιήσω πρώτα ανά σχολείο και μετά ανά τάξη σχολείου και έτσι να βλέπω στην εκτύπωση το 1ο Γυμνάσιο, τις τρεις τάξεις του μ' όλα τα στοιχεία των μαθητών ανά τάξη, μετά το 2ο Γυμνάσιο με τις τρεις τάξεις του και όλα τα στοιχεία των μαθητών κοκ.

ΜΑΚΡΟΕΝΤΟΛΕΣ (Macros)

Τι είναι οι Μακροεντολές της Access και ποια η χρησιμότητά τους;

Οι **μακροεντολές (macros)** είναι ένα σύνολο εντολών της Access που εκτελείται κάθε φορά που το καλούμε. Μπορούμε, μέσα στις εντολές που υπάρχουν σε μια μακροεντολή, να προσθέσουμε και συνθήκες (conditions), ώστε η μακροεντολή να μην εκτελείται πάντα, παρά μόνο όταν είναι αληθείς οι συνθήκες που έχουμε θέσει.

Η Access έχει ένα συγκεκριμένο σύνολο εντολών για μακροεντολές, απ' όπου μπορούμε να επιλέξουμε αυτές που θέλουμε να εκτελέσουμε. Δηλ., δεν μπορούμε να δημιουργήσουμε δικές μας εντολές, αλλά περιοριζόμαστε σ' αυτές που έχει έτοιμες η Access.

Μερικές απ' αυτές τις εντολές είναι οι εξής :

- Close (κλείνει μια φόρμα, μια αναφορά ή μια βάση δεδομένων),

- OpenForm (ανοίγει μια φόρμα),
- OpenReport (ανοίγει μια αναφορά),
- RunCode (εκτελεί μια συνάρτηση της VBA),
- RunMacro (εκτελεί μια άλλη μακροεντολή),
- StopMacro (σταματά την τρέχουσα μακροεντολή),
- GoToRecord (πηγαίνει σε μια εγγραφή),
- GoToControl (ενεργοποιεί ένα χειριστήριο),
- Maximize (μεγιστοποιεί το ενεργό παράθυρο),
- Beep (παράγει ήχο),
- Msgbox (εμφανίζει ένα μήνυμα).

Μια μακροεντολή μπορεί να αποτελείται από μία ή περισσότερες από τις εντολές που είδαμε παραπάνω και μπορούμε ακόμη να ομαδοποιήσουμε και κάποιες εντολές μέσα σε μια μακροεντολή και να τις δώσουμε ένα όνομα. Έτσι, μια μακροεντολή μπορεί να αποτελείται από ομάδες εντολών και μπορούμε να καλέσουμε μόνο αυτή την ομάδα εντολών της μακροεντολής ως εξής :

[ΌνομαΜακροεντολής].[ΟμάδαΜακροεντολής]

Για να έχουμε πρόσβαση στις τιμές των πεδίων (χειριστηρίων) μιας φόρμας, μπορούμε να χρησιμοποιήσουμε τον εξής τύπο :

Forms![ΌνομαΦόρμας]![ΌνομαΧειριστηρίου]

Μπορούμε να συνδέσουμε μια μακροεντολή με ένα κουμπί εντολής μιας φόρμας με τους εξής δύο τρόπους : ή σύρουμε το εικονίδιο της μακροεντολής με το ποντίκι και το αφήνουμε μέσα στη φόρμα στο σημείο που θέλουμε, οπότε το κουμπί εντολής δημιουργείται αυτόματα, ή δημιουργούμε μέσα στη φόρμα ένα κουμπί εντολής και από τις ιδιότητες του κουμπιού εντολής και από τον κατάλογο Event (Συμβάντα) επιλέγουμε την ιδιότητα συμβάντος OnClick και γράφουμε εκεί το όνομα της μακροεντολής που θέλουμε να εκτελεστεί με το πάτημα του ποντικιού.

Η **χρησιμότητα** των μακροεντολών είναι ότι μπορούν να αντικαταστήσουν πολλές ενέργειες που θα έπρεπε να κάναμε αν δεν υπήρχε η μακροεντολή. Έτσι, ενέργειες που τις κάνουμε συχνά, όπως π.χ. άνοιγμα φόρμας, άνοιγμα αναφοράς, έλεγχος των τιμών κάποιων πεδίων, κ.ά., μπορούμε να τις ενσωματώσουμε μέσα σε μακροεντολές και να τις εκτελούμε μ' ένα απλό πάτημα του ποντικιού.

ΥΠΟΜΟΝΑΔΕΣ (Modules)

Τι είναι οι Υπομονάδες της Access και ποια η χρησιμότητά τους;

Οι **υπομονάδες (modules)** της Access είναι στην ουσία η γλώσσα προγραμματισμού που έχει η Access. Επειδή οι μακροεντολές της Access είναι περιορισμένες σε κάποιες συγκεκριμένες εντολές, υπάρχουν και οι υπομονάδες που έχουν απεριόριστες δυνατότητες.

Με τις υπομονάδες μπορούμε να δημιουργήσουμε όσες μεταβλητές και σταθερές θέλουμε και να ελέγξουμε και να τροποποιήσουμε όποια πεδία των πινάκων της Access θέλουμε. Μπορούμε, ακόμη, πολύ εύκολα, να δημιουργήσουμε συνθήκες, εντολές ροής, εντολές επανάληψης (βρόχους), συναρτήσεις (functions) και υπορουτίνες (subroutines).

Η **Visual Basic for Applications (VBA)**, η γλώσσα προγραμματισμού της Access, δεν διαφέρει και δεν υστερεί σε τίποτα από τις άλλες γλώσσες προγραμματισμού. Κάθε πρόγραμμα (κώδικας) της VBA αποθηκεύεται σε υπομονάδες (modules). Υπάρχουν δύο

τρόποι δημιουργίας υπομονάδων : σαν αντικείμενο υπομονάδας ή σαν μέρος ενός αντικειμένου φόρμας ή αναφοράς.

Τι είναι το Αντικείμενο Υπομονάδας;

Είναι ένας κώδικας προγράμματος που περιέχει διαδικασίες που θα μπορούν να χρησιμοποιηθούν από ένα ή περισσότερα ερωτήματα, φόρμες ή αναφορές. Μια διαδικασία που έχει οριστεί σ' ένα αντικείμενο υπομονάδας, μπορεί να κληθεί από οποιοδήποτε σημείο της εφαρμογής μας.

Μπορούμε να δούμε τα αντικείμενα υπομονάδων μιας βάσης δεδομένων πατώντας στην καρτέλα Modules (Υπομονάδες).

Τι είναι οι Υπομονάδες Φορμών και Αναφορών;

Είναι υπομονάδες που ανήκουν αποκλειστικά σε κάποια φόρμα ή αναφορά. Μπορούμε να διορθώσουμε την υπομονάδα μιας φόρμας ή αναφοράς ανοίγοντας τη φόρμα ή την αναφορά σε Άποψη Σχεδιασμού και πατώντας στο κουμπί Code (Εώδικας) της γραμμής εργαλείων ή επιλέγοντας την εντολή Code από το μενού View.

Περιγράψτε το Παράθυρο Υπομονάδας

Το παράθυρο υπομονάδας δημιουργείται όταν ανοίξουμε μια υπομονάδα σε Άποψη Σχεδιασμού. Η Access μάς τοποθετεί αυτόματα στο τμήμα Declarations (Δηλώσεις), όπου μπορούμε να ορίσουμε εκείνες τις μεταβλητές που θα τις μοιράζονται όλες οι διαδικασίες της υπομονάδας.

Στη γραμμή τίτλου του παραθύρου βλέπουμε το όνομα της υπομονάδας και ακριβώς από κάτω υπάρχουν δύο πλαίσια πτυσσόμενου καταλόγου :

Πλαίσιο Καταλόγου Αντικειμένων (Object)

Όταν δουλεύουμε μ' ένα αντικείμενο υπομονάδας, εδώ εμφανίζεται μόνο η επιλογή General. Όταν δουλεύουμε με μια υπομονάδα φόρμας ή αναφοράς, επιλέγουμε τη φόρμα ή την αναφορά από τον κατάλογο και τότε στο πλαίσιο καταλόγου διαδικασιών (Proc) εμφανίζονται οι διαθέσιμες διαδικασίες συμβάντων για το αντικείμενο που έχουμε επιλέξει.

Πλαίσιο Καταλόγου Διαδικασιών (Proc)

Όταν δουλεύουμε μ' ένα αντικείμενο υπομονάδας, εδώ εμφανίζονται με αλφαβητική σειρά όλες οι διαδικασίες που έχει η υπομονάδα. Όταν δουλεύουμε με μια υπομονάδα φόρμας ή αναφοράς, εδώ εμφανίζονται οι διαθέσιμες διαδικασίες συμβάντων που ανήκουν στη φόρμα ή την αναφορά.

Πώς δημιουργώ μια νέα Διαδικασία σε μια Υπομονάδα;

Μέσα στο παράθυρο υπομονάδας, γράφουμε την εντολή Function (Συνάρτηση) ή Sub (Υπορουτίνα) και μετά πατάμε στο κουμπί Εισαγωγής Διαδικασίας (Insert Procedure) της γραμμής εργαλείων ή επιλέγουμε την εντολή Procedure από το μενού Insert. Η Access θα ξεκινήσει τότε αυτόματα μια νέα διαδικασία και θα βάλει στο τέλος την εντολή End Function ή End Sub αντίστοιχα.

Τύποι Δεδομένων της VBA

Οι τύποι δεδομένων της VBA είναι παρόμοιοι με τους τύπους δεδομένων των πεδίων μιας βάσης δεδομένων.

Οι τύποι δεδομένων είναι : Byte, Integer (%), Long (&), Single (!), Double (#), Currency (@), String ($), Boolean, Date Object, Variant και User-defined. Τα σύμβολα

μέσα στην παρένθεση είναι ο χαρακτήρας που τοποθετείται μετά το όνομα της μεταβλητής για να προσδιορίσει τον τύπο της.

Εμβέλεια Μεταβλητών και Σταθερών

Μια μεταβλητή ή μια σταθερά μπορεί να είναι "ορατή" σε μία μόνο διαδικασία (τοπική εμβέλεια), σ' όλες τις διαδικασίες μιας υπομονάδας ή σ' όλες τις διαδικασίες της βάσης δεδομένων μας (καθολική εμβέλεια).

Για να δηλώσουμε μια καθολική μεταβλητή, χρησιμοποιούμε την εντολή **Public** στο τμήμα Declarations ενός αντικειμένου υπομονάδας. Για να δηλώσουμε μια καθολική σταθερά, χρησιμοποιούμε τη λέξη Public μαζί με μια εντολή Const στο τμήμα Declarations.

Για να δηλώσουμε μια μεταβλητή ή σταθερά που θα μπορεί να χρησιμοποιηθεί απ' όλες τις διαδικασίες μιας υπομονάδας, την ορίζουμε στο τμήμα Declarations της υπομονάδας.

Για να δηλώσουμε μια μεταβλητή ή σταθερά που θα μπορεί να χρησιμοποιηθεί μόνο σε μια συγκεκριμένη διαδικασία, την ορίζουμε σαν μέρος της διαδικασίας.

Σε μια βάση δεδομένων, μπορούμε να δώσουμε ίδια ονόματα σε μεταβλητές που ανήκουν σε διαφορετικές υπομονάδες καθώς και σε μεταβλητές που έχουν καθολική ή τοπική εμβέλεια. Στην πρώτη περίπτωση, για να ξεχωρίσουμε τις μεταβλητές, γράφουμε πριν απ' το όνομά τους, το όνομα της υπομονάδας που ανήκουν. Π.χ., οι μεταβλητές module1.intx και module2.intx έχουν ίδιο όνομα (intx) αλλά ανήκουν σε διαφορετικές υπομονάδες.

Στη δεύτερη περίπτωση, ισχύει ό,τι και σ' άλλες γλώσσες προγραμματισμού, δηλ. μέσα σε μια διαδικασία υπερισχύουν οι τοπικές μεταβλητές και έξω από τη διαδικασία υπερισχύουν οι καθολικές μεταβλητές.

Εντολές της VBA

Εντολές για Ορισμό Σταθερών και Μεταβλητών

Const

Με την εντολή αυτή ορίζουμε σταθερές.
Π.χ.
Public Const PI = 3.14159

Dim

Δηλώνουμε μια μεταβλητή ή έναν πίνακα μεταβλητών στο τμήμα Declarations μιας υπομονάδας που θα μπορεί να χρησιμοποιηθεί σ' όλες τις διαδικασίες της υπομονάδας. Μπορούμε, ακόμη, να χρησιμοποιήσουμε την εντολή Dim μέσα σε μια διαδικασία για να δηλώσουμε μια τοπική μεταβλητή στη διαδικασία αυτή.

Π.χ.

```
Dim intMyInteger As Integer
Dim dbMyDatabase As Database
Dim strMyString (51 To 100) As String * 20
```

Public

Με την εντολή αυτή δηλώνουμε καθολικές μεταβλητές στο τμήμα Declarations μιας υπομονάδας.

Π.χ.

```
Public lngMyNumber As Long
```

Υπάρχει και η εντολή Private, με την οποία δηλώνουμε μια μεταβλητή ή σταθερή που θα είναι διαθέσιμη μόνο μέσα στην υπομονάδα όπου γίνεται η δήλωση. Η δήλωση Private είναι default.

ReDim

Δηλώνουμε δυναμικά έναν πίνακα μέσα σε μια διαδικασία ή αλλάζουμε τις διαστάσεις ενός δηλωμένου πίνακα μέσα σε μια διαδικασία κατά το χρόνο εκτέλεσης.

Π.χ.

```
ReDim strProductNames(20) As String * 25
```

Static

Δηλώνουμε μια μεταβλητή που θα χρησιμοποιηθεί μόνο μέσα σε μια διαδικασία και στην οποία η Access δεν θα επαναφέρει την αρχική της τιμής για όσο διάστημα θα είναι ανοιχτή η υπομονάδα που περιέχει τη διαδικασία αυτή.

Π.χ.

```
Static intMyInteger As Integer
Static strMyString (51 To 100) As String * 20
```

Type

Την χρησιμοποιούμε σ' ένα τμήμα Declarations για να δημιουργήσουμε μια δομή δεδομένων ορισμένη από το χρήστη που θα περιέχει μία ή περισσότερες μεταβλητές. Μπορούμε επίσης να χρησιμοποιήσουμε την εντολή Type για να δηλώσουμε μια δομή εγγραφής.

Αφού δηλώσουμε μια δομή δεδομένων ορισμένη από το χρήστη, θα μπορούμε να χρησιμοποιήσουμε το όνομα τύπου σε οποιαδήποτε επόμενη εντολή Dim, Public ή Static για να δημιουργήσουμε μια μεταβλητή αυτού του τύπου.

Μπορούμε να αναφερόμαστε στις μεταβλητές μιας τέτοιας δομής δεδομένων, εισάγοντας το όνομα της μεταβλητής, μια τελεία και το όνομα της μεταβλητής που είναι μέσα στη δομή.

Π.χ.

```
Type MyRecord
   lngID As Long
   strLast As String
   strFirst As String
   strMid As String
End Type
……
Dim usrContacts As MyRecord
usrContacts.strLast = 'Jones'
```

Οι Πίνακες στην Access

Πίνακες μπορούμε να δηλώσουμε με τις εντολές Dim, ReDim και Static. Σ' έναν πίνακα μπορούμε να έχουμε μέχρι και 60 διαστάσεις. Αν δεν ορίσουμε το κάτω όριο σε μια διάσταση ενός πίνακα, η προεπιλογή είναι το 0. Μπορούμε να αλλάξουμε την προεπιλεγμένη τιμή του κάτω ορίου με την εντολή Option Base στο τμήμα Declarations της υπομονάδας :

```
Option Base 1
```

Το κάτω όριο μιας διάστασης ενός πίνακα θα πρέπει να είναι μεγαλύτερο από τον αριθμό -32.768 και το πάνω όριο θα πρέπει να είναι μικρότερο από τον αριθμό 32.767.

Η Εντολή Set

Με την εντολή αυτή μπορούμε να αντιστοιχίσουμε μεταβλητές αντικειμένου σε ονόματα βάσεων δεδομένων, πινάκων και πεδίων για να κάνουμε ευκολότερα τη δουλειά μας. Δείτε τα παρακάτω παραδείγματα :

```
Dim dbMyDB As Database
Set dbMyDB = CurrentDb()
```

Η μεταβλητή dbMyDB αναφέρεται στην τρέχουσα βάση δεδομένων.

```
Dim tblMyTable As TableDef
Set tblMyTable = dbMyDB.TableDefs![tblClubs]
```

Η μεταβλητή tblMyTable αναφέρεται στον πίνακα tblClubs της τρέχουσας βάσης δεδομένων.

```
Dim fldMyField As Field
Set fldMyField = tblMyTable![Notes]
```

Η μεταβλητή fldMyField αναφέρεται στο πεδίο Notes του πίνακα tblClubs της τρέχουσας βάσης δεδομένων.

Επεξεργασία των Εγγραφών μιας Βάσης Δεδομένων

Η Access χρησιμοποιεί πολλές εντολές με πολλές επιλογές για να επεξεργαστεί τις εγγραφές ενός πίνακα μιας βάσης δεδομένων. Εδώ θα δούμε με απλά παραδείγματα το πώς μπορούμε να επεξεργαστούμε τις εγγραφές ενός πίνακα.

```
Dim dbEntSched As Database
Dim rcdClubs As RecordSet
Set dbEntSched = CurrentDb()
Set rcdClubs = dbEntSched.OpenRecordSet('tblClubs', dbOpenTable)
```

Μπορούμε να χρησιμοποιήσουμε μια από τις μεθόδους Move για να μετακινηθούμε σε μια συγκεκριμένη εγγραφή. Π.χ. recordset.MoveFirst για να μετακινηθούμε στην πρώτη εγγραφή. Άλλες επιλογές είναι οι MoveLast, MoveNext και MovePrevious.

Για να μετακινηθούμε σε μια άλλη εγγραφή, πρέπει να χρησιμοποιήσουμε τη μέθοδο Find με κάποια κριτήρια. Για παράδειγμα, για να βρούμε το πρώτο στοιχείο ενός συνόλου εγγραφών του οποίου η τιμή AmountOwed είναι μικρότερη από 100$, δίνουμε :

```
Dim dbEntSched As Database
Dim rcdAPContracts As RecordSet
Set dbEntSched = CurrentDb()
rcdAPContracts.FindFirst 'AmountOwed > 100'
```

Για να διαγράψουμε μια εγγραφή, πρώτα μετακινούμαστε στην εγγραφή αυτή και μετά χρησιμοποιούμε τη μέθοδο Delete. Για παράδειγμα, για να διαγράψουμε το πρώτο στοιχείο ενός συνόλου εγγραφών του οποίου η τιμή AmountOwed είναι ίση με 0, δίνουμε :

```
rcdAPContracts.FindFirst 'AmountOwed = 0'
If Not rcdAPContracts.NoMatch Then
   rcdAPContracts.Delete
End If
```

Για να ενημερώσουμε κάποιες εγγραφές, πρώτα πηγαίνουμε στην πρώτη εγγραφή που θέλουμε να ενημερώσουμε και μετά χρησιμοποιούμε τη μέθοδο Edit. Τέλος, θα πρέπει να χρησιμοποιήσουμε και τη μέθοδο Update για να ισχύσουν οι αλλαγές. Για παράδειγμα, για

να αυξήσουμε κατά 10% την καταχώριση AmountOwed της πρώτης εγγραφής με τιμή μεγαλύτερη από 100, δίνουμε :

```
rcdAPContracts.FindFirst 'AmountOwed > 100'
If Not rcdAPContracts.NoMatch Then
   rcdAPContracts.Edit
   rcdAPContracts![AmountOwed] = rcdAPContracts![AmountOwed] * 1.1
   rcdAPContracts.Update
End If
```

Για να εισάγουμε μια νέα εγγραφή σ' ένα σύνολο εγγραφών, πρέπει να χρησιμοποιήσουμε τη μέθοδο AddNew. Δίνουμε όλες τις τιμές που θέλουμε στα πεδία και τέλος χρησιμοποιούμε και τη μέθοδο Update για να γίνει η αποθήκευση.

```
rcdClubs.AddNew
rcdClubs![ClubName] = 'Winthrop Brewing Co.'
rcdClubs![StreetAddress] = '155 Riverside ave.'
rcdClubs![City] = 'Winthrop'
rcdClubs![State] = 'WA'
rcdClubs![ZipCode] = '98862'
rcdClubs![PhoneNumber] = '(509) 996-3183'
rcdClubs.Update
```

Συναρτήσεις και Υπορουτίνες

Στη VBA υπάρχουν, όπως και στις άλλες γλώσσες προγραμματισμού, δύο είδη διαδικασιών : οι Συναρτήσεις (Functions) και οι Υπορουτίνες (Subroutines). Και οι τύποι διαδικασιών μπορούν να δεχθούν παραμέτρους. Οι συναρτήσεις μπορούν να επιστρέψουν μια τιμή δεδομένων, ενώ οι υπορουτίνες όχι.

Function

Μπορούμε να χρησιμοποιήσουμε μια εντολή Function για να δηλώσουμε μια νέα συνάρτηση, τις παραμέτρους που θα δέχεται, τον τύπο μεταβλητής που θα επιστρέφει και τον κώδικα που θα εκτελεί τη διαδικασία συνάρτησης.

Θα πρέπει να δηλώσουμε τον τύπο δεδομένων όλων των ορισμάτων που μπορεί να δεχθεί η συνάρτηση στη λίστα παραμέτρων της. Αν χρησιμοποιήσουμε τη δεσμευμένη λέξη ByVal για να δηλώσουμε ένα όρισμα, τότε αυτή θα είναι κλήση με τιμή. Δηλαδή, οποιαδήποτε αλλαγή κάνουμε σ' ένα όρισμα ByVal μέσα στη συνάρτηση, δεν θα αλλάξει την αρχική μεταβλητή.

Αν χρησιμοποιήσουμε τη δεσμευμένη λέξη ByRef για να δηλώσουμε ένα όρισμα, τότε αυτή θα είναι κλήση με αναφορά. Δηλαδή, οποιαδήποτε αλλαγή κάνουμε σ' ένα όρισμα ByRef μέσα στη συνάρτηση, θα αλλάξει και την αρχική μεταβλητή.

Μπορούμε να χρησιμοποιήσουμε την εντολή Exit Function σε οποιοδήποτε σημείο της συνάρτησης, για να βγούμε από τη συνάρτηση ομαλά και να επιστρέψουμε στη διαδικασία κλήσης.

Για παράδειγμα, για να δημιουργήσουμε μια συνάρτηση με όνομα MyFunction που θα δέχεται ένα ακέραιο και ένα αλφαριθμητικό όρισμα και θα επιστρέφει μια τιμή διπλής ακρίβειας, γράφουμε :

```
Function MyFunction (intArg1 As Integer, strArg2 As String) As Double
   <εντολές συνάρτησης>
End Function
```

Sub

Μπορούμε να χρησιμοποιήσουμε μια εντολή Sub για να δηλώσουμε μια νέα υπορουτίνα, τις παραμέτρους που θα δέχεται και τον κώδικά της.

Μπορούμε να χρησιμοποιήσουμε την εντολή Exit Sub σε οποιοδήποτε σημείο της υπορουτίνας, για να βγούμε από την υπορουτίνα ομαλά και να επιστρέψουμε στη διαδικασία κλήσης.

Για παράδειγμα, για να δημιουργήσουμε μια υπορουτίνα με όνομα MySub που θα δέχεται δύο αλφαριθμητικά ορίσματα, αλλά θα μπορεί να τροποποιήσει μόνο το δεύτερο, γράφουμε :

```
Sub MySub (ByVal strArg1 As String, ByRef strArg2 As String)
   <εντολές υπορουτίνας>
End Sub
```

Έλεγχος της Ροής των Εντολών

Call

Μεταφέρει τον έλεγχο σε μια υπορουτίνα.

Π.χ.

```
Call MySub (intMyInteger, curPrice * intQty)
```

Do ... Loop

Εκτελεί ένα σύνολο εντολών πολλές φορές μέχρις ότου γίνει ψευδής ή αληθής μια συνθήκη. Δέχεται και τις δύο εκφράσεις While και Until, τις οποίες μπορούμε να βάλουμε στην αρχή ή στο τέλος του βρόχου.

Π.χ.

```
Do Until rcdClubs.EOF
   <εντολές διαδικασίας>
   rcdClubs.MoveNext
Loop
```

For ... Next

Εκτελεί ένα σύνολο εντολών συγκεκριμένες φορές.

Π.χ.

```
For intI = 0 To 4
   Debug.Print dbEntSched.QueryDefs(intI).Name
Next intI
```

For Each ... Next

Εκτελεί ένα σύνολο εντολών για κάθε στοιχείο μιας συλλογής ή μήτρας.

Π.χ.

```
For Each qdf In dbEntSched.QueryDefs
   Debug.Print qdf.Name
Next qdf
```

GoTo

Πηγαίνει, χωρίς περιορισμό, σε κάποια άλλη εντολή.

Π.χ.

```
GoTo SkipOver
```

If … Then ... Else

Εκτελεί κάποιες εντολές ανάλογα με την τιμή που επιστρέφει μια συνθήκη.

Π.χ.

```
Dim strMyString As String, strFirst As String, intVal As Integer
strFirst = Ucase$(Mid$(strMyString, 1, 1))
If strFirst >= 'A' And strFirst <= 'F' Then
   intVal = 1
ElseIf strFirst >= 'G' And strFirst <= 'N' Then
   intVal = 2
ElseIf strFirst >= 'O' And strFirst <= 'Z' Then
   intVal = 3
Else
   intVal = 0
End If
```

Select Case

Εκτελεί εντολές υπό συνθήκη, με βάση το αποτέλεσμα της σύγκρισης της τιμής μιας παράστασης μ' έναν κατάλογο ή ένα εύρος τιμών.

Π.χ.

```
Dim strMyString As String, intVal As Integer
Select Case Ucase$(Mid$(strMyString, 1, 1))
   Case 'A' To 'F'
      intVal = 1
   Case 'G' To 'N'
      intVal = 2
   Case 'O' To 'Z'
      intVal = 3
   Case Else
      intVal = 0
End Select
```

Stop

Αναστέλλει προσωρινά την εκτέλεση μιας διαδικασίας.

While ... Wend

Εκτελεί συνέχεια ένα σύνολο εντολών για όσο διάστημα είναι αληθής μια συνθήκη.

Π.χ.

```
While Not rcdClubs.EOF
   <εντολές διαδικασίας>
   rcdClubs.MoveNext
Wend
```

With

Είναι παρόμοια με την εντολή With της Pascal.

Π.χ.

```
Dim rst As Recordset, db As Database
Set db = CurentDb()
Set rst = db.OpenRecordset('MyTable', dbOpenDynaset, dbAppendOnly)
```

```
With rst
   .Addnew
   ![FieldOne] = '1'
   ![FieldTwo] = 'John'
   ![FieldThree] = 'Viescas'
   .Update
   .Close
End With
```

Εκτέλεση Ενεργειών Μακροεντολών

Στη VBA μπορούμε να εκτελέσουμε τις περισσότερες από τις ενέργειες μακροεντολών. Για όσες, όμως, ενέργειες μακροεντολών δεν μπορούν να εκτελεστούν μέσα σε μια διαδικασία της VBA, υπάρχουν ισοδύναμες εντολές της VBA. Για να εκτελέσουμε μια ενέργεια μακροεντολής μέσα από τη VBA, χρησιμοποιούμε το αντικείμενο DoCmd.

DoCmd

Π.χ. για να ανοίξουμε τη φόρμα Customer στην Άποψη Φόρμας για να εισαγάγουμε δεδομένα, γράφουμε :

```
DoCmd.OpenForm 'Customer', acNormal, , ,acAdd
```

Για να κλείσουμε τη φόρμα Supplier, γράφουμε :

```
DoCmd.Close acForm, 'Supplier'
```

Προγραμματισμός σε Access

ΥΠΟΜΟΝΑΔΕΣ (Modules)

Τι είναι οι Υπομονάδες της Access και ποια η χρησιμότητά τους;

Οι **υπομονάδες (modules)** της Access είναι στην ουσία η γλώσσα προγραμματισμού που έχει η Access. Επειδή οι μακροεντολές της Access είναι περιορισμένες σε κάποιες συγκεκριμένες εντολές, υπάρχουν και οι υπομονάδες που έχουν απεριόριστες δυνατότητες.

Με τις υπομονάδες μπορούμε να δημιουργήσουμε όσες μεταβλητές και σταθερές θέλουμε και να ελέγξουμε και να τροποποιήσουμε όποια πεδία των πινάκων της Access θέλουμε. Μπορούμε, ακόμη, πολύ εύκολα, να δημιουργήσουμε συνθήκες, εντολές ροής, εντολές επανάληψης (βρόχους), συναρτήσεις (functions) και υπορουτίνες (subroutines).

Η **Visual Basic for Applications (VBA)**, η γλώσσα προγραμματισμού της Access, δεν διαφέρει και δεν υστερεί σε τίποτα από τις άλλες γλώσσες προγραμματισμού. Κάθε πρόγραμμα (κώδικας) της VBA αποθηκεύεται σε υπομονάδες (modules). Υπάρχουν δύο τρόποι δημιουργίας υπομονάδων : σαν αντικείμενο υπομονάδας ή σαν μέρος ενός αντικειμένου φόρμας ή αναφοράς.

Τι είναι το Αντικείμενο Υπομονάδας;

Είναι ένας κώδικας προγράμματος που περιέχει διαδικασίες που θα μπορούν να χρησιμοποιηθούν από ένα ή περισσότερα ερωτήματα, φόρμες ή αναφορές. Μια διαδικασία που έχει οριστεί σ' ένα αντικείμενο υπομονάδας, μπορεί να κληθεί από οποιοδήποτε σημείο της εφαρμογής μας.

Μπορούμε να δούμε τα αντικείμενα υπομονάδων μιας βάσης δεδομένων πατώντας στην καρτέλα Modules (Υπομονάδες).

Τι είναι οι Υπομονάδες Φορμών και Αναφορών;

Είναι υπομονάδες που ανήκουν αποκλειστικά σε κάποια φόρμα ή αναφορά. Μπορούμε να διορθώσουμε την υπομονάδα μιας φόρμας ή αναφοράς ανοίγοντας τη φόρμα ή την αναφορά σε Άποψη Σχεδιασμού και πατώντας στο κουμπί Code (Εώδικας) της γραμμής εργαλείων ή επιλέγοντας την εντολή Code από το μενού View.

Περιγράψτε το Παράθυρο Υπομονάδας

Το παράθυρο υπομονάδας δημιουργείται όταν ανοίξουμε μια υπομονάδα σε Άποψη Σχεδιασμού. Η Access μάς τοποθετεί αυτόματα στο τμήμα Declarations (Δηλώσεις), όπου μπορούμε να ορίσουμε εκείνες τις μεταβλητές που θα τις μοιράζονται όλες οι διαδικασίες της υπομονάδας.

Στη γραμμή τίτλου του παραθύρου βλέπουμε το όνομα της υπομονάδας και ακριβώς από κάτω υπάρχουν δύο πλαίσια πτυσσόμενου καταλόγου :

Πλαίσιο Καταλόγου Αντικειμένων (Object)

Όταν δουλεύουμε μ' ένα αντικείμενο υπομονάδας, εδώ εμφανίζεται μόνο η επιλογή General. Όταν δουλεύουμε με μια υπομονάδα φόρμας ή αναφοράς, επιλέγουμε τη φόρμα ή την αναφορά από τον κατάλογο και τότε στο πλαίσιο καταλόγου διαδικασιών (Proc) εμφανίζονται οι διαθέσιμες διαδικασίες συμβάντων για το αντικείμενο που έχουμε επιλέξει.

Πλαίσιο Καταλόγου Διαδικασιών (Proc)

Όταν δουλεύουμε μ' ένα αντικείμενο υπομονάδας, εδώ εμφανίζονται με αλφαβητική σειρά όλες οι διαδικασίες που έχει η υπομονάδα. Όταν δουλεύουμε με μια υπομονάδα

φόρμας ή αναφοράς, εδώ εμφανίζονται οι διαθέσιμες διαδικασίες συμβάντων που ανήκουν στη φόρμα ή την αναφορά.

Πώς δημιουργώ μια νέα Διαδικασία σε μια Υπομονάδα;

Μέσα στο παράθυρο υπομονάδας, γράφουμε την εντολή Function (Συνάρτηση) ή Sub (Υπορουτίνα) και μετά πατάμε στο κουμπί Εισαγωγής Διαδικασίας (Insert Procedure) της γραμμής εργαλείων ή επιλέγουμε την εντολή Procedure από το μενού Insert. Η Access θα ξεκινήσει τότε αυτόματα μια νέα διαδικασία και θα βάλει στο τέλος την εντολή End Function ή End Sub αντίστοιχα.

Τύποι Δεδομένων της VBA

Οι τύποι δεδομένων της VBA είναι παρόμοιοι με τους τύπους δεδομένων των πεδίων μιας βάσης δεδομένων.

Οι τύποι δεδομένων είναι : Byte, Integer (%), Long (&), Single (!), Double (#), Currency (@), String ($), Boolean, Date Object, Variant και User-defined. Τα σύμβολα μέσα στην παρένθεση είναι ο χαρακτήρας που τοποθετείται μετά το όνομα της μεταβλητής για να προσδιορίσει τον τύπο της.

Εμβέλεια Μεταβλητών και Σταθερών

Μια μεταβλητή ή μια σταθερά μπορεί να είναι "ορατή" σε μία μόνο διαδικασία (τοπική εμβέλεια), σ' όλες τις διαδικασίες μιας υπομονάδας ή σ' όλες τις διαδικασίες της βάσης δεδομένων μας (καθολική εμβέλεια).

Για να δηλώσουμε μια καθολική μεταβλητή, χρησιμοποιούμε την εντολή **Public** στο τμήμα Declarations ενός αντικειμένου υπομονάδας. Για να δηλώσουμε μια καθολική σταθερά, χρησιμοποιούμε τη λέξη Public μαζί με μια εντολή Const στο τμήμα Declarations.

Για να δηλώσουμε μια μεταβλητή ή σταθερά που θα μπορεί να χρησιμοποιηθεί απ' όλες τις διαδικασίες μιας υπομονάδας, την ορίζουμε στο τμήμα Declarations της υπομονάδας.

Για να δηλώσουμε μια μεταβλητή ή σταθερά που θα μπορεί να χρησιμοποιηθεί μόνο σε μια συγκεκριμένη διαδικασία, την ορίζουμε σαν μέρος της διαδικασίας.

Σε μια βάση δεδομένων, μπορούμε να δώσουμε ίδια ονόματα σε μεταβλητές που ανήκουν σε διαφορετικές υπομονάδες καθώς και σε μεταβλητές που έχουν καθολική ή τοπική εμβέλεια. Στην πρώτη περίπτωση, για να ξεχωρίσουμε τις μεταβλητές, γράφουμε πριν απ' το όνομά τους, το όνομα της υπομονάδας που ανήκουν. Π.χ., οι μεταβλητές module1.intx και module2.intx έχουν ίδιο όνομα (intx) αλλά ανήκουν σε διαφορετικές υπομονάδες.

Στη δεύτερη περίπτωση, ισχύει ό,τι και σ' άλλες γλώσσες προγραμματισμού, δηλ. μέσα σε μια διαδικασία υπερισχύουν οι τοπικές μεταβλητές και έξω από τη διαδικασία υπερισχύουν οι καθολικές μεταβλητές.

Εντολές της VBA

Εντολές για Ορισμό Σταθερών και Μεταβλητών

Const

Με την εντολή αυτή ορίζουμε σταθερές.

Π.χ.

```
Public Const PI = 3.14159
```

Dim

Δηλώνουμε μια μεταβλητή ή έναν πίνακα μεταβλητών στο τμήμα Declarations μιας υπομονάδας που θα μπορεί να χρησιμοποιηθεί σ' όλες τις διαδικασίες της υπομονάδας. Μπορούμε, ακόμη, να χρησιμοποιήσουμε την εντολή Dim μέσα σε μια διαδικασία για να δηλώσουμε μια τοπική μεταβλητή στη διαδικασία αυτή.

Π.χ.

```
Dim intMyInteger As Integer
Dim dbMyDatabase As Database
Dim strMyString (51 To 100) As String * 20
```

Public

Με την εντολή αυτή δηλώνουμε καθολικές μεταβλητές στο τμήμα Declarations μιας υπομονάδας.

Π.χ.

```
Public lngMyNumber As Long
```

Υπάρχει και η εντολή **Private**, με την οποία δηλώνουμε μια μεταβλητή ή σταθερή που θα είναι διαθέσιμη μόνο μέσα στην υπομονάδα όπου γίνεται η δήλωση. Η δήλωση Private είναι default.

ReDim

Δηλώνουμε δυναμικά έναν πίνακα μέσα σε μια διαδικασία ή αλλάζουμε τις διαστάσεις ενός δηλωμένου πίνακα μέσα σε μια διαδικασία κατά το χρόνο εκτέλεσης.

Π.χ.

```
ReDim strProductNames(20) As String * 25
```

Static

Δηλώνουμε μια μεταβλητή που θα χρησιμοποιηθεί μόνο μέσα σε μια διαδικασία και στην οποία η Access δεν θα επαναφέρει την αρχική της τιμής για όσο διάστημα θα είναι ανοιχτή η υπομονάδα που περιέχει τη διαδικασία αυτή.

Π.χ.

```
Static intMyInteger As Integer
Static strMyString (51 To 100) As String * 20
```

Type

Την χρησιμοποιούμε σ' ένα τμήμα Declarations για να δημιουργήσουμε μια δομή δεδομένων ορισμένη από το χρήστη που θα περιέχει μία ή περισσότερες μεταβλητές. Μπορούμε επίσης να χρησιμοποιήσουμε την εντολή Type για να δηλώσουμε μια δομή εγγραφής.

Αφού δηλώσουμε μια δομή δεδομένων ορισμένη από το χρήστη, θα μπορούμε να χρησιμοποιήσουμε το όνομα τύπου σε οποιαδήποτε επόμενη εντολή Dim, Public ή Static για να δημιουργήσουμε μια μεταβλητή αυτού του τύπου.

Μπορούμε να αναφερόμαστε στις μεταβλητές μιας τέτοιας δομής δεδομένων, εισάγοντας το όνομα της μεταβλητής, μια τελεία και το όνομα της μεταβλητής που είναι μέσα στη δομή.

Π.χ.

```
Type MyRecord
   lngID As Long
```

```
    strLast As String
    strFirst As String
    strMid As String
End Type
...
Dim usrContacts As MyRecord
usrContacts.strLast = 'Jones'
```

Οι Πίνακες στην Access

Πίνακες μπορούμε να δηλώσουμε με τις εντολές Dim, ReDim και Static. Σ' έναν πίνακα μπορούμε να έχουμε μέχρι και 60 διαστάσεις. Αν δεν ορίσουμε το κάτω όριο σε μια διάσταση ενός πίνακα, η προεπιλογή είναι το 0. Μπορούμε να αλλάξουμε την προεπιλεγμένη τιμή του κάτω ορίου με την εντολή Option Base στο τμήμα Declarations της υπομονάδας :

```
Option Base 1
```

Το κάτω όριο μιας διάστασης ενός πίνακα θα πρέπει να είναι μεγαλύτερο από τον αριθμό -32.768 και το πάνω όριο θα πρέπει να είναι μικρότερο από τον αριθμό 32.767.

Η Εντολή Set

Με την εντολή αυτή μπορούμε να αντιστοιχίσουμε μεταβλητές αντικειμένου σε ονόματα βάσεων δεδομένων, πινάκων και πεδίων για να κάνουμε ευκολότερα τη δουλειά μας. Δείτε τα παρακάτω παραδείγματα :

```
Dim dbMyDB As Database
Set dbMyDB = CurrentDb()
```

Η μεταβλητή dbMyDB αναφέρεται στην τρέχουσα βάση δεδομένων.

```
Dim tblMyTable As TableDef
Set tblMyTable = dbMyDB.TableDefs![tblClubs]
```

Η μεταβλητή tblMyTable αναφέρεται στον πίνακα tblClubs της τρέχουσας βάσης δεδομένων.

```
Dim fldMyField As Field
Set fldMyField = tblMyTable![Notes]
```

Η μεταβλητή fldMyField αναφέρεται στο πεδίο Notes του πίνακα tblClubs της τρέχουσας βάσης δεδομένων.

Επεξεργασία των Εγγραφών μιας Βάσης Δεδομένων

Η Access χρησιμοποιεί πολλές εντολές με πολλές επιλογές για να επεξεργαστεί τις εγγραφές ενός πίνακα μιας βάσης δεδομένων. Εδώ θα δούμε με απλά παραδείγματα το πώς μπορούμε να επεξεργαστούμε τις εγγραφές ενός πίνακα.

```
Dim dbEntSched As Database
Dim rcdClubs As RecordSet
Set dbEntSched = CurrentDb()
Set rcdClubs = dbEntSched.OpenRecordSet('tblClubs', dbOpenTable)
```

Μπορούμε να χρησιμοποιήσουμε μια από τις μεθόδους **Move** για να μετακινηθούμε σε μια συγκεκριμένη εγγραφή. Π.χ. recordset.MoveFirst για να μετακινηθούμε στην πρώτη εγγραφή. Άλλες επιλογές είναι οι MoveLast, MoveNext και MovePrevious.

Για να μετακινηθούμε σε μια άλλη εγγραφή, πρέπει να χρησιμοποιήσουμε τη μέθοδο **Find** με κάποια κριτήρια. Για παράδειγμα, για να βρούμε το πρώτο στοιχείο ενός συνόλου εγγραφών του οποίου η τιμή AmountOwed είναι μικρότερη από 100$, δίνουμε :

```
Dim dbEntSched As Database
Dim rcdAPContracts As RecordSet
Set dbEntSched = CurrentDb()
rcdAPContracts.FindFirst 'AmountOwed > 100'
```

Για να διαγράψουμε μια εγγραφή, πρώτα μετακινούμαστε στην εγγραφή αυτή και μετά χρησιμοποιούμε τη μέθοδο **Delete**. Για παράδειγμα, για να διαγράψουμε το πρώτο στοιχείο ενός συνόλου εγγραφών του οποίου η τιμή AmountOwed είναι ίση με 0, δίνουμε :

```
rcdAPContracts.FindFirst 'AmountOwed = 0'
If Not rcdAPContracts.NoMatch Then
   rcdAPContracts.Delete
End If
```

Για να ενημερώσουμε κάποιες εγγραφές, πρώτα πηγαίνουμε στην πρώτη εγγραφή που θέλουμε να ενημερώσουμε και μετά χρησιμοποιούμε τη μέθοδο **Edit**. Τέλος, θα πρέπει να χρησιμοποιήσουμε και τη μέθοδο **Update** για να ισχύσουν οι αλλαγές. Για παράδειγμα, για να αυξήσουμε κατά 10% την καταχώριση AmountOwed της πρώτης εγγραφής με τιμή μεγαλύτερη από 100, δίνουμε :

```
rcdAPContracts.FindFirst 'AmountOwed > 100'
If Not rcdAPContracts.NoMatch Then
   rcdAPContracts.Edit
   rcdAPContracts![AmountOwed] = rcdAPContracts![AmountOwed] * 1.1
   rcdAPContracts.Update
End If
```

Για να εισάγουμε μια νέα εγγραφή σ' ένα σύνολο εγγραφών, πρέπει να χρησιμοποιήσουμε τη μέθοδο **AddNew**. Δίνουμε όλες τις τιμές που θέλουμε στα πεδία και τέλος χρησιμοποιούμε και τη μέθοδο **Update** για να γίνει η αποθήκευση.

```
rcdClubs.AddNew
rcdClubs![ClubName] = 'Winthrop Brewing Co.'
rcdClubs![StreetAddress] = '155 Riverside ave.'
rcdClubs![City] = 'Winthrop'
rcdClubs![State] = 'WA'
rcdClubs![ZipCode] = '98862'
rcdClubs![PhoneNumber] = '(509) 996-3183'
rcdClubs.Update
```

Συναρτήσεις και Υπορουτίνες

Στη VBA υπάρχουν, όπως και στις άλλες γλώσσες προγραμματισμού, δύο είδη διαδικασιών : οι Συναρτήσεις (Functions) και οι Υπορουτίνες (Subroutines). Και οι τύποι διαδικασιών μπορούν να δεχθούν παραμέτρους. Οι συναρτήσεις μπορούν να επιστρέψουν μια τιμή δεδομένων, ενώ οι υπορουτίνες όχι.

Function

Μπορούμε να χρησιμοποιήσουμε μια εντολή Function για να δηλώσουμε μια νέα συνάρτηση, τις παραμέτρους που θα δέχεται, τον τύπο μεταβλητής που θα επιστρέφει και τον κώδικα που θα εκτελεί τη διαδικασία συνάρτησης.

Θα πρέπει να δηλώσουμε τον τύπο δεδομένων όλων των ορισμάτων που μπορεί να δεχθεί η συνάρτηση στη λίστα παραμέτρων της. Αν χρησιμοποιήσουμε τη δεσμευμένη λέξη

ByVal για να δηλώσουμε ένα όρισμα, τότε αυτή θα είναι **κλήση με τιμή**. Δηλαδή, οποιαδήποτε αλλαγή κάνουμε σ' ένα όρισμα ByVal μέσα στη συνάρτηση, δεν θα αλλάξει την αρχική μεταβλητή.

Αν χρησιμοποιήσουμε τη δεσμευμένη λέξη **ByRef** για να δηλώσουμε ένα όρισμα, τότε αυτή θα είναι **κλήση με αναφορά**. Δηλαδή, οποιαδήποτε αλλαγή κάνουμε σ' ένα όρισμα ByRef μέσα στη συνάρτηση, θα αλλάξει και την αρχική μεταβλητή.

Μπορούμε να χρησιμοποιήσουμε την εντολή **Exit Function** σε οποιοδήποτε σημείο της συνάρτησης, για να βγούμε από τη συνάρτηση ομαλά και να επιστρέψουμε στη διαδικασία κλήσης.

Για παράδειγμα, για να δημιουργήσουμε μια συνάρτηση με όνομα MyFunction που θα δέχεται ένα ακέραιο και ένα αλφαριθμητικό όρισμα και θα επιστρέφει μια τιμή διπλής ακρίβειας, γράφουμε :

```
Function MyFunction (intArg1 As Integer, strArg2 As String) As Double
   <εντολές συνάρτησης>
End Function
```

Sub

Μπορούμε να χρησιμοποιήσουμε μια εντολή Sub για να δηλώσουμε μια νέα υπορουτίνα, τις παραμέτρους που θα δέχεται και τον κώδικά της.

Μπορούμε να χρησιμοποιήσουμε την εντολή **Exit Sub** σε οποιοδήποτε σημείο της υπορουτίνας, για να βγούμε από την υπορουτίνα ομαλά και να επιστρέψουμε στη διαδικασία κλήσης.

Για παράδειγμα, για να δημιουργήσουμε μια υπορουτίνα με όνομα MySub που θα δέχεται δύο αλφαριθμητικά ορίσματα, αλλά θα μπορεί να τροποποιήσει μόνο το δεύτερο, γράφουμε :

```
Sub MySub (ByVal strArg1 As String, ByRef strArg2 As String)
   <εντολές υπορουτίνας>
End Sub
```

Έλεγχος της Ροής των Εντολών

Call

Μεταφέρει τον έλεγχο σε μια υπορουτίνα.

Π.χ.

```
Call MySub (intMyInteger, curPrice * intQty)
```

Do ... Loop

Εκτελεί ένα σύνολο εντολών πολλές φορές μέχρις ότου γίνει ψευδής ή αληθής μια συνθήκη. Δέχεται και τις δύο εκφράσεις While και Until, τις οποίες μπορούμε να βάλουμε στην αρχή ή στο τέλος του βρόχου.

Π.χ.

```
Do Until rcdClubs.EOF
   <εντολές διαδικασίας>
   rcdClubs.MoveNext
Loop
```

For ... Next

Εκτελεί ένα σύνολο εντολών συγκεκριμένες φορές.

Π.χ.

```
For intI = 0 To 4
   Debug.Print dbEntSched.QueryDefs(intI).Name
Next intI
```

For Each ... Next

Εκτελεί ένα σύνολο εντολών για κάθε στοιχείο μιας συλλογής ή μήτρας.

Π.χ.

```
For Each qdf In dbEntSched.QueryDefs
   Debug.Print qdf.Name
Next qdf
```

GoTo

Πηγαίνει, χωρίς περιορισμό, σε κάποια άλλη εντολή.

Π.χ.

```
GoTo SkipOver
```

If ... Then ... Else

Εκτελεί κάποιες εντολές ανάλογα με την τιμή που επιστρέφει μια συνθήκη.

Π.χ.

```
Dim strMyString As String, strFirst As String, intVal As Integer
strFirst = Ucase$(Mid$(strMyString, 1, 1))
If strFirst >= 'A' And strFirst <= 'F' Then
   intVal = 1
ElseIf strFirst >= 'G' And strFirst <= 'N' Then
   intVal = 2
ElseIf strFirst >= 'O' And strFirst <= 'Z' Then
   intVal = 3
Else
   intVal = 0
End If
```

Select Case

Εκτελεί εντολές υπό συνθήκη, με βάση το αποτέλεσμα της σύγκρισης της τιμής μιας παράστασης μ' έναν κατάλογο ή ένα εύρος τιμών.

Π.χ.

```
Dim strMyString As String, intVal As Integer
Select Case Ucase$(Mid$(strMyString, 1, 1))
   Case 'A' To 'F'
      intVal = 1
   Case 'G' To 'N'
      intVal = 2
   Case 'O' To 'Z'
      intVal = 3
   Case Else
      intVal = 0
End Select
```

Stop

Αναστέλλει προσωρινά την εκτέλεση μιας διαδικασίας.

While ... Wend

Εκτελεί συνέχεια ένα σύνολο εντολών για όσο διάστημα είναι αληθής μια συνθήκη.

Π.χ.

```
While Not rcdClubs.EOF
   <εντολές διαδικασίας>
   rcdClubs.MoveNext
Wend
```

With

Είναι παρόμοια με την εντολή With της Pascal.

Π.χ.

```
Dim rst As Recordset, db As Database
Set db = CurentDb()
Set rst = db.OpenRecordset('MyTable', dbOpenDynaset, dbAppendOnly)
With rst
   .Addnew
   ![FieldOne] = '1'
   ![FieldTwo] = 'John'
   ![FieldThree] = 'Viescas'
   .Update
   .Close
End With
```

Εκτέλεση Ενεργειών Μακροεντολών

Στη VBA μπορούμε να εκτελέσουμε τις περισσότερες από τις ενέργειες μακροεντολών. Για όσες, όμως, ενέργειες μακροεντολών δεν μπορούν να εκτελεστούν μέσα σε μια διαδικασία της VBA, υπάρχουν ισοδύναμες εντολές της VBA. Για να εκτελέσουμε μια ενέργεια μακροεντολής μέσα από τη VBA, χρησιμοποιούμε το αντικείμενο DoCmd.

DoCmd

Π.χ. για να ανοίξουμε τη φόρμα Customer στην Άποψη Φόρμας για να εισαγάγουμε δεδομένα, γράφουμε :

```
DoCmd.OpenForm 'Customer', acNormal, , ,acAdd
```

Για να κλείσουμε τη φόρμα Supplier, γράφουμε :

```
DoCmd.Close acForm, 'Supplier'
```