



Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

**Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής**



Πτυχιακή Εργασία

Τίτλος:

Βάση Δεδομένων για Super Market

Νίκος Μαθιουδάκης (Α.Μ. : 1678)

Γιώργος Μάρκογλου (Α.Μ. : 1776)

Επιβλέπων καθηγητής: Νικόλαος Παπαδάκης

Επιτροπή Αξιολόγησης:

Ημερομηνία Παρουσίασης: 27/10/2015

Θέμα πτυχιακής εργασίας

Θέλουμε να αποθηκεύσουμε την παρακάτω πληροφορία για μια αλυσίδα supermarket.

- Τα στοιχεία κάθε προϊόντος. Όνομα, είδος και κατασκευάστης / παραγωγός και τιμή.
- Τις αποθήκες που έχουμε. Όνομα, διεύθυνση.
- Τα υποκαταστήματα, Όνομα., διεύθυνση
- Τους υπαλλήλους, Ονοματεπώνυμο, διεύθυνση τηλέφωνο, μισθό και αριθμό ταυτότητας.
- Τους προμηθευτές, Ονοματεπώνυμο, διεύθυνση τηλέφωνο
- Τις παραγγελίες κωδικό και ποσότητα.

Επίσης θέλουμε να αποθηκεύσουμε:

Μια αποθήκη μπορεί να έχει πολλά προϊόντα και ένα προϊόν μπορεί να είναι σε πολλές αποθήκες. Για κάθε προϊόν θέλουμε να αποθηκεύουμε τα αποθέματα που υπάρχουν στην αποθήκη. Ένα υποκατάστημα εφοδιάζεται από μια αποθήκη και μια αποθήκη μπορεί να εφοδιάζει πολλά υποκαταστήματα. Κάθε υπάλληλος δουλεύει σε ένα και μόνο ένα υποκατάστημα και ένα υποκατάστημα έχει πολλούς υπαλλήλους. Κάθε προμηθευτής συνδέεται με πολλά προϊόντα (τουλάχιστον 1) και κάθε προϊόν παρέχεται από πολλούς προμηθευτές (τουλάχιστον 1). Κάθε προμηθευτής έχει μια τιμή πώλησης η οποία αλλάζει συχνά. Κάθε προμηθευτής συνδέεται με πολλές αποθήκες και αντίστροφα. Διαδικασίες που πρέπει να υποστηρίζονται. Εισαγωγή και διαγραφή για κάθε μια από τις πιο πάνω κατηγορίες. Για κάθε υποκατάστημα θέλουμε να αποθηκεύουμε τα αποθέματα που υπάρχουν για κάθε προϊόν. Για κάθε προϊόν σε κάθε υποκατάστημα θέλουμε να έχουμε ένα ελάχιστο απόθεμα για κάθε προϊόν (για το ίδιο προϊόν αυτό είναι διαφορετικό σε κάθε υποκατάστημα).

Στο τέλος κάθε μέρα γίνεται ενημέρωση με τις πωλήσεις και γίνεται η αντίστοιχη μείωση των αποθεματικών. Όταν τα αποθεματικά κάποιο προϊόντος βγαίνει το αντίστοιχο μήνυμα. Διαδικασία εφοδιασμού. Μεταφορά προϊόντων από την αποθήκη προς κάποιο υποκατάστημα με αυτόματη μείωση και αύξηση των αποθεμάτων αντίστοιχα. Η διαδικασία πρέπει να ενεργοποιείται αυτόματα κάθε φορά που τα αποθέματα κάποιο προϊόντος σε κάποιο υποκατάστημα πέσουν κάτω από το ελάχιστο θα γίνεται εφοδιασμός έτσι ώστε τα αποθέματα να γίνουν το διπλάσιο του ελαχίστου. Διαδικασία αυτόματης παραγγελίας. Κάθε φορά που τα αποθέματα κάποιο προϊόντος σε κάποια αποθήκη πέσουν κάτω από το ελάχιστο θα γίνεται παραγγελία προς το προμηθευτή με την χαμηλότερη τιμή έτσι ώστε τα αποθέματα να γίνουν το διπλάσιο του ελαχίστου.

Περίληψη

Η εργασία όπως είδαμε και παραπάνω ασχολείται με την υλοποίηση μιας βάσης δεδομένων για Σούπερ Μάρκετ η οποία διαχειρίζεται τα προϊόντα του, τα αποθέματα του είτε στο κατάστημα είτε σε κάποια από τις αποθήκες από τις οποίες τα προμηθεύετε. Επίσης αρχειοθετεί τους εργαζόμενους του καταστήματος και κρατά προσωπικά στοιχεία για τον κάθε ένα αλλά και δεδομένα για τις οικονομικές τους απολαβές. Μια τέτοια βάση δεδομένων είναι πολύ σημαντική για μεγάλα καταστήματα ή μεγάλες πολυεθνικές εταιρίες οι οποίες διαχειρίζονται εισαγωγή και εξαγωγή προϊόντων εντός και εκτός της εταιρίας και απασχολούν πολλούς εργαζόμενους. Είναι το κύριο εργαλείο όχι μόνο γιατί μπορούν να ελέγχουν τότε ένα προϊόν βρίσκεται σε έλλειμμα αλλά και τι αποθέματα υπάρχουν στις αποθήκες. Σε περίπτωση που κάποιο προϊόν βρίσκεται σε έλλειψη και δεν πέσει στην αντίληψη του εργαζόμενου η βάση τον ενημερώνει για τα τελευταία τρία αποθέματα και για αυτόματη αποστολή παραγγελίας για την συμπλήρωση στην αρχική τους ποσότητα. Αν το προϊόν δεν βρίσκεται σε έλλειμμα μόνο στην δικιά τους αποθήκη αλλά και σε κάποιο προμηθευτή τότε ενημερώνει και άλλους προμηθευτές με τους οποίους συνεργάζεται για την εύρεση του προϊόντος που χρειάζεται για να συμπληρώσει τα αποθέματα του.

Η διαδικασία που προαναφέραμε όσον αφορά τους προμηθευτές του καταστήματος συνδέεται με τα προϊόντα που μπορεί να εφοδιάσει ο κάθε ένας έτσι ώστε να γνωρίζει ποιός πρέπει να ενημερωθεί κάθε φορά για την κατάλληλη παραγγελία του προϊόντος. Γίνετε δηλαδή αυτόματη αντιστοίχιση προμηθευτών και προϊόντων.

Έπειτα λοιπόν την κατανόηση της λειτουργικότητας της βάσης μας αρχίσαμε να γράφουμε τον κώδικα υλοποίησης της. Με τον οποίο και δημιουργήσαμε την βάση μας.

Παρακάτω αναφερόμαστε με ποιο τρόπο φτάσαμε σε αυτό το αποτέλεσμα.

Αρχικά αναφέρουμε κάποια ιστορικά στοιχεία από όπου γίνεται κατανοητό πως έγινε η επιλογή της SQL για την υλοποίηση τέτοιου είδους προγραμμάτων. Προγράμματα δηλαδή τα οποία καταγράφουν και συγκρατούν διαφόρων ειδών δεδομένα και τα αρχειοθετούν. Επειδή όμως το περιβάλλον της SQL δεν είναι και τόσο ευχάριστο στο ανθρώπινο μάτι το παρουσιάσαμε με την βοήθεια της JAVA η οποία προσφέρει ένα πιο κατανοητό περιβάλλον στο χρήστη.

Η διαδικασία που ακολουθήσαμε αναφέρεται παρακάτω παρουσιάζοντας και κάποιες βασικές εντολές που χρησιμοποιούνται σε κάθε μια από τις γλώσσες προγραμματισμού που χρησιμοποιήθηκαν οι οποίες αποτελούν βασικές αρχές για την υλοποίηση πιο απλών προγραμμάτων.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ - 1	2
ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ.....	2
1.1. Εισαγωγή στις βάσεις δεδομένων	2
1.2. Ιστορική αναδρομή των βάσεων δεδομένων	3
1.3. Είδη βάσεων.....	4
1.4. Πλεονεκτήματα και Μειονεκτήματα των βάσεων.....	6
1.4.1. Πλεονεκτήματα	6
1.4.2. Μειονεκτήματα	7
ΚΕΦΑΛΑΙΟ - 2	9
ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΒΑΣΕΩΝ	9
2.1. Σχεδιασμός βάσης.....	9
2.2. Πίνακες δεδομένων.....	9
ΚΕΦΑΛΑΙΟ - 3	11
ΟΡΟΙ & ΕΝΝΟΙΕΣ – ΘΕΩΡΙΑ & ΟΡΙΣΜΟΙ	11
3.1 Η έννοια του Portal	11
3.2 Η γλώσσα HTML	11
3.3 Cascading Style Sheets (CSS)	12
3.4 Javascript	12
3.5 Asynchronous JavaScript and XML.....	13
3.6 PHP	13
3.7 MySQL	15
ΚΕΦΑΛΑΙΟ - 4	16
ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ SQL	16
4.1. Ιστορικά στοιχεία	16
4.2. Συσχετισμοί	16
4.3. Δυνατότητες της SQL.....	17
4.4. Ορισμός δεδομένων.....	17
4.5. Κατάλογος (Catalog).....	17
4.6. Βασικές εντολές της SQL	18
4.7. Περιορισμοί.....	19
4.8. Δημιουργία πίνακα	20
4.9. DML: Insert	20
4.10. DML: DELETE.....	21

4.11.	DML: UPDATE	21
4.12.	Ερωτήματα SQL (Queries)	22
4.12.1.	Απλά SQL Queries.....	22
4.12.2.	Σύνθετα SQL queries	23
4.12.3.	Εμφωλευμένα Ερωτήματα	29
ΚΕΦΑΛΑΙΟ - 5		33
ΣΧΕΣΙΑΚΑ ΜΟΝΤΕΛΑ & ΠΙΝΑΚΕΣ		33
5.1.	Πληθικότητα.....	33
5.2.	Σχέσεις.....	34
5.2.1.	Εισαγωγή στις σχέσεις.....	35
5.2.2.	Δημιουργία σχέσης	36
5.2.3.	Διαγραφή σχέσης	37
6.1.	Προβολή βάσης δεδομένων σε περιβάλλον Java	39
6.2.	Στιγμιότυπα ιστοσελίδας	45

ΚΕΦΑΛΑΙΟ - 1

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

1.1. Εισαγωγή στις βάσεις δεδομένων

Η αλματώδης ανάπτυξη της επιστήμης της πληροφορικής και των επικοινωνιών τα τελευταία χρόνια έχει καταστήσει την πληροφορία ως ένα από τα βασικότερα και πολυτιμότερα αγαθά. Είναι κοινός τόπος σήμερα η εκτίμηση ότι το αγαθό της πληροφορίας είναι επιθυμητό απ' όλους τους εργαζόμενους αλλά και τους εκπαιδευόμενους, ώστε να είναι πιο αποδοτικοί, ανταγωνιστικοί αλλά και παραγωγικοί στην εργασία τους.

Οι βάσεις δεδομένων αποτελούν πλέον σήμερα ένα αναπόσπαστο κομμάτι της καθημερινής ζωής των ανθρώπων και των επιχειρήσεων, παρόλο που αυτό είναι σχεδόν άγνωστο για την πλειοψηφία των ατόμων, που αγνοούν την ύπαρξη τους. Ο λόγος που συμβαίνει αυτό είναι διότι, οι τελικοί χρήστες ενός αντικειμένου ή μιας υπηρεσίας η οποία στηρίζεται σε μια βάση δεδομένων γνωρίζουν μόνο τον εξωτερικό ορισμό αυτής και το αποτέλεσμα της και έχουν άγνοια για το πώς έχει οριστεί αυτό το αντικείμενο και πως εν τέλει λειτουργεί.

Για παράδειγμα αν πάμε στην τράπεζα για κατάθεση ή ανάληψη χρημάτων, αν κάνουμε κράτηση ξενοδοχείου ή αεροπορικού εισιτηρίου, αν ψάχνουμε βιβλιογραφικά στοιχεία από ένα κατάλογο βιβλιοθήκης, υπάρχει πιθανότητα η δραστηριότητα μας να περιλαμβάνει προσπέλαση δεδομένων, τα οποία εμείς δε βλέπουμε, αλλά αντιλαμβανόμαστε το αποτέλεσμα τους.

Μια βάση δεδομένων (database) χρησιμοποιείται για την συλλογή, αποθήκευση, οργάνωση και επεξεργασία δεδομένων με σκοπό την εξαγωγή πληροφοριών από αυτά, τα οποία σχετίζονται με ένα συγκεκριμένο θέμα. Μια βάση δεδομένων χρησιμοποιείται σε διάφορες πρακτικές, οι οποίες είναι μια σειρά αλυσιδωτών αντιδράσεων με σκοπό την εξαγωγή πληροφοριών. Χαρακτηριστικά παραδείγματα δεδομένων που απαιτούν σωστή και αποδοτική οργάνωση είναι ενδεικτικά 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.

Τα δεδομένα που εισάγονται σε μια βάση δεδομένων δεν περιέχουν πλεονασμούς και ο τρόπος οργάνωσης τους είναι ανεξάρτητος από τις εφαρμογές που χρησιμοποιούν, ενώ μπορούν να χρησιμοποιηθούν ταυτόχρονα από πολλούς, διαφορετικούς χρήστες και για διαφορετικούς σκοπούς. Σημειώνεται λοιπόν, σύμφωνα με τα παραπάνω ότι μέσω της χρήσης μια βάσης δεδομένων, αντί να έχουμε ξεχωριστά αρχεία με πλεονάζουσα πληροφορία για κάτι το οποίο μας ενδιαφέρει, η βάση δεδομένων αποτελεί μία διαμοιρασμένη συλλογική πηγή της πληροφορίας. Μία βάση δεδομένων δηλαδή κρατάει εκτός από τα στοιχεία κάποιου οργανισμού και μία περιγραφή των στοιχείων αυτών.

Μια βάση δεδομένων έχει τις παρακάτω ιδιότητες:

1. Αναπαριστά κάποια άποψη του πραγματικού κόσμου, η οποία μερικές φορές λέγεται και μικρόκοσμος. Είναι μια λογικά συνεκτική συλλογή δεδομένων, που έχει κάποια

εγγενή σημασία. Μια τυχαία διευθέτηση δεδομένων δεν είναι σωστό να αναφέρεται ως βάση δεδομένων.

2. Σχεδιάζεται και γεμίζει με δεδομένα για κάποιο συγκεκριμένο σκοπό.
3. Προορίζεται για μια συγκεκριμένη ομάδα χρηστών και για κάποιες προκαθορισμένες εφαρμογές.

Μια βάση δομείται σε τρία επίπεδα, το εξωτερικό, το εσωτερικό και το νοητό επίπεδο. Στο εξωτερικό επίπεδο, αποτελείται από εκείνα τα στοιχεία της βάσης που ενδιαφέρουν το χρήστη, το νοητό (ή εννοιολογικό) επίπεδο, το οποίο περιέχει τις λογικές δομές ολόκληρης της βάσης, όπως τις βλέπει ο διαχειριστής και το εσωτερικό τέλος επίπεδο, το οποίο καλύπτει την φυσική υλοποίηση της βάσης με σκοπό την επίτευξη της βέλτιστης απόδοσης και χρησιμοποίησης χώρου. Το εσωτερικό επίπεδο της βάσης δεδομένων, στην ουσία καλύπτει τις δομές των δεδομένων και την οργάνωση των αρχείων που χρησιμοποιούνται για την αποθήκευση των δεδομένων στους δίσκους του συστήματος.

Από τα παραπάνω συνάδει ότι μια βάση δεδομένων αποτελεί μια ολοκληρωμένη συλλογή δεδομένων, αποθηκευμένη και οργανωμένη με τρόπο, ο οποίος διευκολύνει τη διαχείριση τους και παρέχει αξιοπιστία, ευελιξία και ασφάλεια. Βασίζεται δε, στη συγκέντρωση των δεδομένων και στην ελαχιστοποίηση των πλεοναζόντων δεδομένων, υποστηρίζοντας και εξυπηρετώντας την ανάπτυξη ποικίλων εφαρμογών.

1.2. Ιστορική αναδρομή των βάσεων δεδομένων

Η εξέλιξη των βάσεων δεδομένων μπορεί να θεωρηθεί ότι ξεκίνησε τη δεκαετία του 1950, όπου για τη διαχείριση και επεξεργασία μεμονωμένων αρχείων χρησιμοποιούνταν οι κάρτες και οι ταινίες. Οι εξελίξεις σε συσκευές μαζικής αποθήκευσης τυχαίας πρόσβασης και αύξηση υπολογιστικής ισχύος έθεσαν την περίοδο εκείνη τις απαραίτητες προϋποθέσεις για την ανάπτυξη συστημάτων διαχείρισης δεδομένων σε αντικατάσταση των συστημάτων διαχείρισης αρχείων.

Τα πρώτα συστήματα διαχείρισης βάσεων δεδομένων δημιουργήθηκαν τη δεκαετία του 1960 με σκοπό ένα κοινό οργανωτικό πλαίσιο για την διαχείριση δεδομένων τα οποία μέχρι τότε αποθηκεύονταν σε μεμονωμένα αρχεία. Το 1964, ο Charles Bachman της General Electric πρότεινε ένα δικτυωτό μοντέλο δεδομένων (network data model) στο οποίο οι εγγραφές δεδομένων ήταν συνδεδεμένες μεταξύ τους με τέτοιο τρόπο ώστε να σχηματίζουν τεμνόμενα σύνολα δεδομένων. Τα πρώτα συστήματα διαχείρισης βάσεων δεδομένων στηρίχθηκαν σε αυτό το δικτυωτό μοντέλο.

Το 1970 έγινε για πρώτη φορά ο ορισμός του σχεσιακού μοντέλου από τον Edgar Codd σε ένα ερευνητικό έντυπο της IBM με τίτλο 'System R4 Relational'. Στην αρχή βέβαια δεν ήταν ξεκάθαρο κατά πόσο ένα σχεσιακό σύστημα που θα βασιζόταν στο σχεσιακό μοντέλο θα μπορούσε να πετύχει εμπορικά. Έτσι, μέχρι και το 1979 όλες οι εμπορικές υλοποιήσεις βάσεων δεδομένων βασίζονταν είτε στην δικτυωτή είτε στην ιεραρχική προσέγγιση. Άρχισαν βέβαια να αναπτύσσονται βέβαια τα ερευνητικά προγράμματα σχεσιακών συστημάτων System R(IBM) και INGRESS, καθώς και σχεσιακές γλώσσες SEQUEL, QBE και QUEL .

Αργότερα, στις αρχές της δεκαετίας του 1980, η σχεσιακή γλώσσα SQL (μέρος του system R) αντικατέστησε την QUEL στο σύστημα INGRESS. Αναπτύχθηκαν οι έννοιες της διαχείρισης συναλλαγών (transaction management) από τον Jim Gray. Οι τάσεις που άρχιζαν να εμφανίζονται εκείνη την περίοδο αφορούσαν τα αντικειμενοστραφή συστήματα, την αρχιτεκτονική πελάτη-διακομιστή και τις καταναεμημένες βάσεις.

Οι εγκαταστάσεις των σχεσιακών συστημάτων αυξάνουν με γοργούς ρυθμούς με πρώτα τα συστήματα Oracle, Server, SQL, Sybase, Informix, DB2. Εμφανίζονται τα σχεσιακά

συστήματα διαχείρισης βάσεων δεδομένων και σε προσωπικούς υπολογιστές: Dbase εξελίχθηκε μέχρι τις μέρες μας σε Paradox και η πιο γνωστή Microsoft Access.

1.3. Είδη βάσεων

Τα διαθέσιμα είδη βάσεων δεδομένων που συναντάμε σήμερα παρουσιάζουν διαφορές τόσο στον τρόπο που διαχειρίζονται την πληροφορία, όσο και στις δυνατότητες αποθήκευσης-περιγραφής, πρόσβασης και ανάκτησής της. Υπάρχουν τέσσερα πρότυπα με τα οποία σχεδιάζεται μια βάση δεδομένων. Τα πρότυπα αυτά είναι τα εξής:

- Ιεραρχικό πρότυπο δεδομένων
- Δικτυακό πρότυπο δεδομένων
- Σχεσιακό πρότυπο δεδομένων
- Αντικειμενοστραφές πρότυπο δεδομένων

Στο Ιεραρχικό πρότυπο (Hierarchical) τα δεδομένα της βάσης παριστάνονται με μια απλή δενδρική δομή. Ένα παράδειγμα ιεραρχικής βάσης δεδομένων για το πληροφοριακό σύστημα ενός τμήματος ανθρώπινων πόρων μιας τυχαίας επιχείρησης αναπαριστάται στην Εικόνα 1.

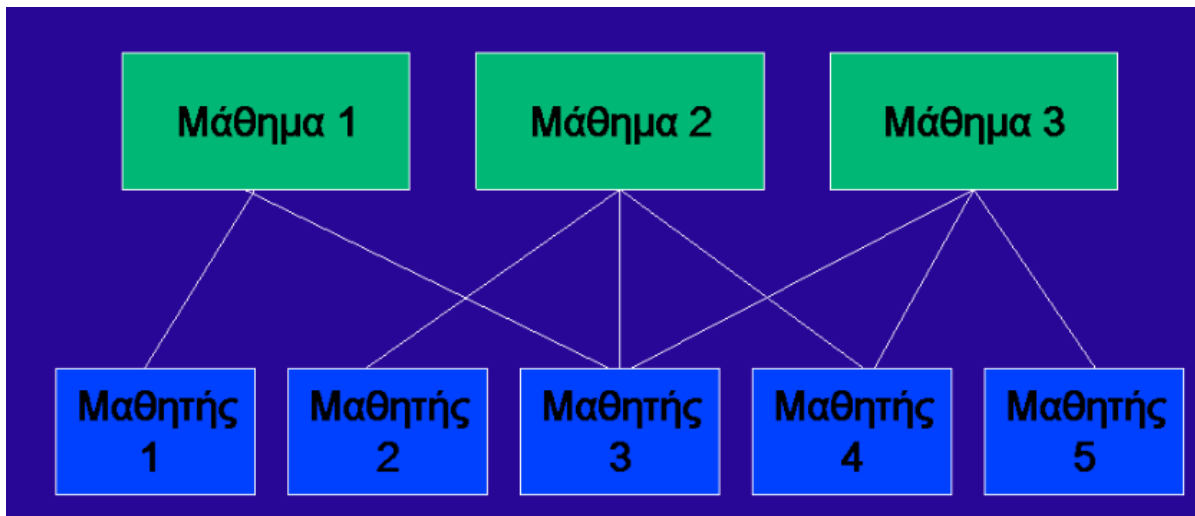


Εικόνα 1: Ιεραρχικό πρότυπο οργάνωσης βάσης δεδομένων.

Στο συγκεκριμένο πρότυπο, κάθε δένδρο περιλαμβάνει μια ανώτερη ιεραρχικά εγγραφή ή record (ανώτερο επίπεδο) και ένα σύνολο κατώτερων ιεραρχικά εγγραφών (κατώτερο επίπεδο). Κάθε εγγραφή περιέχει στοιχεία δεδομένων οργανωμένα σε τμήματα εγγραφών (segments). Το ανώτερο επίπεδο μιας εγγραφής συνδέεται με το κατώτερο με μια λογική σχέση γονέα-απογόνου, ενώ κάθε εγγραφή είναι δυνατό να έχει οποιοδήποτε πλήθος κατώτερων ιεραρχικά εγγραφών (απογόνους). Ο τύπος της εγγραφής που αντιστοιχεί στην κορυφή της ιεραρχίας (τμήμα ανώτατου επιπέδου) ονομάζεται ρίζα (root).

Στο δικτυακό πρότυπο (Network) τα δεδομένα παρουσιάζονται με λογικές συσχετίσεις many-to-many, πολλά προς πολλά, με την έννοια ότι οι «γονείς» έχουν πολλούς διαφορετικούς απογόνους και αντιστρόφως, δηλαδή τα «τέκνα» έχουν πολλούς διαφορετικούς «γονείς». Για

παράδειγμα, όπως φαίνεται στην Εικόνα 2, υπάρχουν στο σύστημα της Γραμματείας μιας σχολής πέντε μαθητές που χρωστάνε εκ των τριών μαθημάτων που εμφανίζονται στο σχήμα.



Εικόνα 2: Σχηματική απεικόνιση δικτυακού προτύπου δεδομένων.

Στο δικτυακό πρότυπο τα δεδομένα παριστάνονται με εγγραφές, όπως και στο ιεραρχικό πρότυπο και συνδυαστικούς κρίκους. Το δικτυακό πρότυπο βέβαια, είναι γενικότερο από το ιεραρχικό, επειδή ένας τύπος εγγραφής μπορεί να έχει περισσότερες από μια ιεραρχικά ανώτερες εγγραφές (γονείς).

Το δικτυακό πρότυπο επιτρέπει τη σχεδίαση βάσεων δεδομένων με πολλαπλές, αμφίδρομες αντιστοιχίες μεταξύ των δεδομένων και υπερέρχει του ιεραρχικού. Βέβαια, τόσο το ιεραρχικό, όσο και το δικτυακό πρότυπο χρησιμοποιούνται σήμερα ελάχιστα πλέον (έχουν σχεδόν απαξιωθεί), εκτός συγκεκριμένων περιπτώσεων που ανταποκρίνονται σε πολύ συγκεκριμένες απαιτήσεις. Τα πρότυπα αυτά θεωρείται ότι υστερούν σε ευελιξία και υποστήριξη και δε διαθέτουν μια εύχρηστη γλώσσα για υποβολή ερωτημάτων.

Αντιθέτως, το πιο διαδεδομένο σύστημα διαχείρισης βάσης δεδομένων είναι το σχεσιακό πρότυπο (Relational Data Base Management System), το οποίο παριστάνει τα δεδομένα ως πίνακες δυο διαστάσεων (στήλες και γραμμές) που ονομάζονται σχέσεις (relations). Η κάθε γραμμή χρησιμοποιείται για την αποθήκευση της πληροφορίας που αφορά μια εγγραφή, ενώ η κάθε στήλη αποθηκεύει τα ξεχωριστά τμήματα της πληροφορίας.

Οι πίνακες μοιάζουν με επίπεδα αρχεία, αλλά το σχεσιακό πρότυπο υποστηρίζει τη συνδυασμένη εξαγωγή πληροφοριών από περισσότερους από έναν πίνακες. Στο πρότυπο αυτό, συχνά οι πίνακες αναφέρονται και ως «αρχεία».

Το σχεσιακό πρότυπο συσχετίζει τα δεδομένα δυο πινάκων με βάση συγκεκριμένα δεδομένα, που είναι κοινά στους δυο πίνακες. Το σχεσιακό πρότυπο υποστηρίζει τις τρεις ακόλουθες βασικές πράξεις που παράγουν τα ζητούμενα σύνολα δεδομένων:

- ✓ **Επιλογή (select).** Η πράξη επιλογή δίνει τη δυνατότητα στο χρήστη να επιλέξει ένα υποσύνολο εγγραφών (γραμμών) από ένα πίνακα, σύμφωνα με συγκεκριμένα κριτήρια.
- ✓ **Προβολή (project).** Η πράξη προβολή δίνει τη δυνατότητα στο χρήστη να ορίσει ένα υποσύνολο των στηλών (πεδίων) ενός πίνακα.
- ✓ **Ένωση (join).** Η ένωση δίνει τη δυνατότητα στο χρήστη να συνδυάσει δυο ή και περισσότερους πίνακες και να ορίσει ένα πίνακα με τα δεδομένα που επιθυμεί.

Στο σχεσιακό πρότυπο, απαιτείται μονάχα η βάση να γίνεται αντιληπτή από τον τελικό χρήστη με τη μορφή πινάκων. Αυτό σημαίνει ότι η φυσική δομή της βάσης μπορεί να είναι υλοποιημένη χρησιμοποιώντας διάφορες τεχνικές αποθήκευσης. Η παραπάνω άποψη είναι εφαρμόσιμη μονάχα στη λογική δομή της βάσης δηλαδή στο εξωτερικό και το νοητό επίπεδο.

1.4. Πλεονεκτήματα και Μειονεκτήματα των βάσεων

1.4.1. Πλεονεκτήματα

Τα πλεονεκτήματα των βάσεων δεδομένων είναι τα παρακάτω:

➤ **Έλεγχος του πλεονασμού δεδομένων:** Όπως αναφέρθηκε παραπάνω τα παραδοσιακά συστήματα αρχείων σπαταλούσαν αρκετό χώρο με το να αποθηκεύουν τα ίδια δεδομένα σε περισσότερα από ένα αρχεία . Αντιθέτως, τα συστήματα βάσεων δεδομένων προσπαθούν να εξαλείψουν τον πλεονασμό τελείως ενσωματώνοντας τα αρχεία έτσι ώστε να μην υπάρχουν πολλά αντίγραφα των ιδίων δεδομένων .Παρ'όλα αυτά οι βάσεις δεδομένων δεν εξαφανίζουν τελείως τον πλεονασμό των δεδομένων, αφού σε πολλές περιπτώσεις χρειάζεται να έχουμε επανάληψη των ιδίων δεδομένων όπως για παράδειγμα στην υλοποίηση σύνθετων σχέσεων (relationships) ανάμεσα στα στοιχεία της βάσης.

➤ **Συνεκτικότητα των Δεδομένων :** Με την εξαφάνιση ή τον έλεγχο του πλεονασμού των δεδομένων ελαττώνουμε τον κίνδυνο εμφάνισης μη συνεκτικών δεδομένων .Εάν τα δεδομένα είναι αποθηκευμένα μονάχα μία φορά στη βάση, οποιαδήποτε ενημέρωση στις τιμές τους εκτελείται μία φορά και η νέα τιμή είναι κατευθείαν διαθέσιμη σε όλους τους τελικούς χρήστες . Εάν πάλι τα ίδια δεδομένα είναι αποθηκευμένα περισσότερες από μία φορές στη βάση και το σύστημα διαχείρισης είναι ενήμερο, μπορεί να εγγυηθεί ότι όλα τα αντίγραφα θα κρατηθούν ενημερωμένα .Δυστυχώς όμως μέχρι και σήμερα δεν μπορούν όλα τα υπάρχοντα στο εμπόριο συστήματα διαχείρισης βάσεων να εγγυηθούν αυτή τη συνεκτικότητα των δεδομένων.

➤ **Επιπλέον Πληροφορίες Από Τα Ίδια Δεδομένα :** Μέσω της ενσωμάτωσης των δεδομένων καθίσταται δυνατό για έναν οργανισμό να αντλήσει από τα δεδομένα της βάσης επιπλέον πληροφορίες, είτε μέσω συναρτήσεων στατιστικών του συστήματος διαχείρισης της βάσης, είτε μέσω της συνένωσης πινάκων .

➤ **Κοινοποίηση Δεδομένων :** Τυπικά, τα αρχεία ανήκουν σε όλους τους εξουσιοδοτημένους χρήστες και έτσι οι περισσότεροι χρήστες μπορούν να μοιραστούν τα δεδομένα .Επιπλέον οι εφαρμογές μπορούν να επεκτείνουν τα υπάρχοντα δεδομένα προσθέτοντας απλά τα νέα δεδομένα στη βάση, χωρίς να χρειάζεται να ορίσουν ξανά όλα τα δεδομένα. Οι εφαρμογές επίσης μπορούν να βασίζονται στις συναρτήσεις του συστήματος διαχείρισης χωρίς να χρειάζεται να έχουν τις δικές τους συναρτήσεις.

➤ **Βελτιωμένη Ακεραιότητα Δεδομένων :** Η ακεραιότητα εκφράζει συνήθως τους διάφορους περιορισμούς , οι οποίοι είναι στην ουσία κανόνες, τους οποίους η βάση δεν πρέπει να παραβαίνει . Οι περιορισμοί αυτοί μπορεί να εφαρμόζονται στα δεδομένα ενός πεδίου (γνώρισμα), ενός πίνακα, ή μπορεί να εφαρμόζονται και στις σχέσεις μεταξύ των πινάκων .Για παράδειγμα, στο πεδίο (γνώρισμα) μιας email διεύθυνσεως θα θέλαμε να υπάρχει το σύμβολο @ υποχρεωτικά.

➤ **Βελτιωμένη Ασφάλεια :** Η ασφάλεια μίας βάσης δεδομένων αποτελεί την προστασία της απέναντι σε μη εξουσιοδοτημένους χρήστες .Χωρίς τα απαραίτητα μέτρα η συνένωση των αρχείων κάνει τα δεδομένα ακόμα πιο επιρρεπή και ευάλωτα σε σχέση με τα συστήματα αρχείων .Έτσι τα συστήματα διαχείρισης βάσεων επιτρέπουν στον administrator να ορίσει και να επιβάλλει την ασφάλεια της βάσης . Αυτό μπορεί να γίνει με τη μορφή ονόματος χρήστη και κωδικού έτσι ώστε να ορισθούν οι εξουσιοδοτημένοι χρήστες . Επιπλέον ορίζονται και τα δικαιώματα που μπορεί να έχει ένας χρήστης ή ένα γκρουπ χρηστών στους διάφορους πίνακες της βάσης. Αξίζει διευκρινιστεί ότι δίνεται και η δυνατότητα ορισμού διαφορετικών δικαιωμάτων για τον ίδιο χρήστη σε κάθε πίνακα της βάσης .

➤ **Βελτιωμένη Διαθεσιμότητα και Απόκριση :** Σαν αποτέλεσμα της ενσωμάτωσης των αρχείων τα δεδομένα είναι απευθείας προσβάσιμα από τον τελικό χρήστη . Τα περισσότερα συστήματα διαχείρισης βάσεων παρέχουν στον τελικό χρήστη γλώσσες υποβολής ερωτήσεων στη βάση, έτσι ώστε ο κάθε χρήστης να μπορεί να λάβει τα στοιχεία που αυτός θέλει, χωρίς να είναι απαραίτητη η παρουσία κάποιου προγραμματιστή ο οποίος θα γράψει κάποια εφαρμογή για την εξαγωγή στοιχείων από τη βάση .

➤ **Αυξημένη Παραγωγικότητα :** Όπως αναφέρθηκε και πριν τα διάφορα συστήματα διαχείρισης παρέχουν έτοιμες συναρτήσεις στους προγραμματιστές εφαρμογών ώστε να μην χρειάζεται να ανησυχούν για πολύ χαμηλού επιπέδου λεπτομέρειες . Αυτό έχει ως αποτέλεσμα την αύξηση της παραγωγικότητας των προγραμματιστών και την μείωση του χρόνου ανάπτυξης των διαφόρων εφαρμογών με τελικό αποτέλεσμα και την μείωση του κόστους μίας τέτοιας εφαρμογής.

➤ **Βελτιωμένη Συντήρηση :** Στα παλαιότερα συστήματα αρχείων η περιγραφή των δεδομένων ήταν ενσωματωμένη μέσα σε κάθε εφαρμογή , κάνοντας έτσι την κάθε εφαρμογή να εξαρτάτε από τα δεδομένα. Μία οποιαδήποτε αλλαγή στη δομή των δεδομένων απαιτούσε και την ανάλογη αλλαγή και στα προγράμματα εφαρμογών που επηρεάζονταν από αυτήν. Αντίθετα στα συστήματα διαχείρισης απομονώνεται η περιγραφή των δεδομένων από τις εφαρμογές με αποτέλεσμα αυτές να μένουν απρόσβλητες από οποιαδήποτε αλλαγή.

➤ **Αυξημένος Συγχρονισμός :** Σε πολλά από τα παλιά συστήματα αρχείων όταν δύο ή περισσότεροι χρήστες προσπαθούσαν να έχουν πρόσβαση στο ίδιο αρχείο συγχρόνως ήταν πιθανό οι προσβάσεις αυτές να ανακατεύονταν με αποτέλεσμα την απώλεια των πληροφοριών ή ακόμα και την απώλεια της ακεραιότητας. Τα σημερινά συστήματα διαχείρισης όμως εξασφαλίζουν ότι κάτι τέτοιο δεν θα συμβεί.

1.4.2. Μειονεκτήματα

Από την άλλη πλευρά, τα μειονεκτήματα των βάσεων δεδομένων είναι τα παρακάτω:

➤ **Πολυπλοκότητα :** Η παροχή όλων των λειτουργιών που απαιτούμε από ένα καλό σύστημα διαχείρισης γίνεται από ένα πολύ σύνθετο πρόγραμμα . Οι σχεδιαστές , οι προγραμματιστές , οι διαχειριστές, ακόμα και οι τελικοί χρήστες θα πρέπει να αντιληφθούν τις λειτουργίες το συστήματος διαχείρισης για να μπορέσουν να το εκμεταλλευτούν.

Αποτυχία στο να μπορέσουν να αντιληφθούν τις λειτουργίες του συστήματος διαχείρισης θα μπορούσε να οδηγήσει σε λανθασμένες αποφάσεις σχεδίασης με πολλαπλές συνέπειες.

➤ **Μέγεθος :** Η πολυπλοκότητα και το εύρος των λειτουργιών του συστήματος διαχείρισης το κάνουν ένα πολύ μεγάλο πρόγραμμα με αρκετές απαιτήσεις σε αποθηκευτικό χώρο και μνήμης για να τρέξει ικανά.

➤ **Κόστος Αγοράς Συστήματος Διαχείρισης :** Το κόστος ενός τέτοιου συστήματος μπορεί να κυμαίνεται σήμερα από 1.500 € για έναν προσωπικό υπολογιστή έως και 60.000 € για ένα σύστημα πολλών χρηστών που θα εκτελείται κάποιον υπολογιστή που ο σκοπός του είναι η εξυπηρέτηση χιλιάδων χρηστών.

➤ **Επιπρόσθετο Κόστος Υλικού (Hardware) :** Οι απαιτήσεις σε αποθηκευτικό χώρο για το σύστημα διαχείρισης είναι πιθανόν να αυξηθούν με αποτέλεσμα την αγορά επιπλέον δίσκων για την κάλυψη των αναγκών της βάσης . Επιπλέον πολλές φορές για να επιτύχουμε την επιθυμητή απόδοση σε χρόνους απόκρισης ίσως χρειαστεί η αγορά νέου υπολογιστή.

➤ **Κόστος Μετατροπής :** Πολλές φορές το κόστος αλλαγής hardware εξ' αιτίας των παραπάνω λόγων είναι ασήμαντο συγκριτικά με το κόστος μετατροπής των διαφόρων προγραμμάτων εφαρμογών έτσι ώστε αυτά να μπορούν να συνεργαστούν με ένα καινούργιο σύστημα διαχείρισης της βάσης ή με καινούργιο hardware .

➤ **Επιδόσεις Συστήματος :** Τυπικά ένα παλιό σύστημα αρχείων είναι γραμμένο για μία συγκεκριμένη εφαρμογή με αποτέλεσμα να έχει καλές επιδόσεις . Αντιθέτως ένα σύστημα διαχείρισης είναι γραμμένο πιο γενικά με σκοπό να καλύπτει τις ανάγκες πολλών εφαρμογών και όχι μίας μονάχα . Αυτό έχει ως αποτέλεσμα οι εφαρμογές να μην τρέχουν τόσο γρήγορα όπως θα έτρεχαν σε ένα σύστημα αρχείων.

➤ **Μεγαλύτερες Επιπτώσεις Σε Αποτυχία :** Η συγκέντρωση όλων των πόρων έχει ως αποτέλεσμα να γίνεται στο σύστημα πιο ευάλωτο . Από τη στιγμή που όλοι οι χρήστες και οι εφαρμογές βασίζονται στην διαθεσιμότητα του συστήματος διαχείρισης η αποτυχία οποιουδήποτε μέρους μπορεί να οδηγήσει το σύστημα σε προσωρινή παύση.

ΚΕΦΑΛΑΙΟ - 2

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΒΑΣΕΩΝ

2.1. Σχεδιασμός βάσης

Ο σχεδιασμός μιας βάσης δεδομένων πρέπει να γίνεται κατά τέτοιο τρόπο, έτσι ώστε η βάση να ανταποκρίνεται στις σημερινές ανάγκες, τις οποίες καλείται να καλύψει και γι' αυτό δημιουργείται, αλλά και σε μελλοντικές ανάγκες. Επομένως, είναι απαραίτητο, η ομάδα σχεδίασης της βάσης, να αφιερώσει αρκετό χρόνο στη μελέτη της δομής της και των δεδομένων που θα χρησιμοποιηθούν γι' αυτήν.

Πριν ξεκινήσουμε να σχεδιάζουμε μια βάση δεδομένων θα πρέπει να απαντήσουμε αρχικά στα εξής ερωτήματα:

- α) Ποιους και πόσους πίνακες θα έχει η βάση που θα δημιουργήσουμε;
- β) Ποια πεδία θα έχει ο κάθε πίνακας και τι τύπου δεδομένα θα έχει το κάθε πεδίο.
- γ) Ποιο θα είναι το πρωτεύον κλειδί του κάθε πίνακα;
- δ) Ποιες σχέσεις θα έχει η βάση και ποιος θα είναι ο τύπος τους;
- ε) Ποιες φόρμες, ποια ερωτήματα και ποιες εκθέσεις θα χρειαστεί να δημιουργηθούν;
- ς) Ποιες εργασίες θα αυτοματοποιηθούν ώστε να ανατεθούν σε μακροεντολές;

2.2. Πίνακες δεδομένων

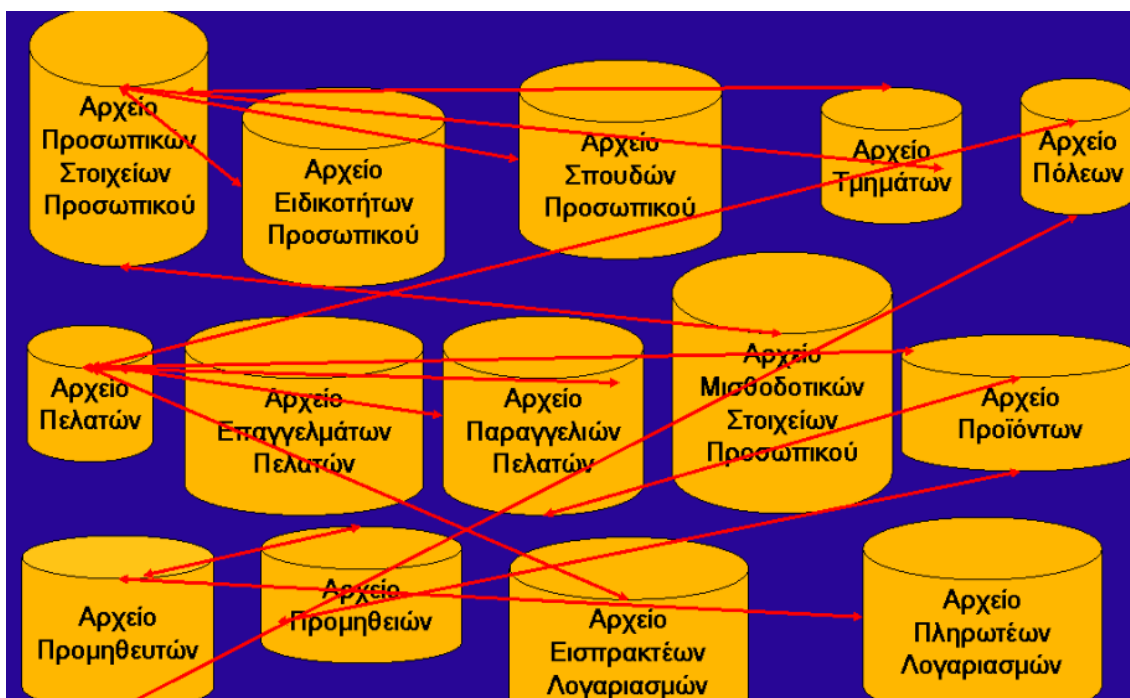
Ο πίνακας δεδομένων περιέχει πληροφορίες σχετικά με ένα συγκεκριμένο θέμα, όπως είναι οι πελάτες ή τα προϊόντα. Η κάθε εγγραφή σε έναν πίνακα περιέχει πληροφορίες σχετικά με ένα στοιχείο, όπως έναν συγκεκριμένο πελάτη και αποτελείται από πεδία (γνώρισμα) όπως Εταιρεία, Όνομα και Επώνυμο. Η εγγραφή αποκαλείται συνήθως γραμμή και το πεδίο αποκαλείται συνήθως στήλη. Ένας πίνακας ονομάζεται αλλιώς και οντότητα της βάσης δεδομένων. Άλλα χαρακτηριστικά είναι:

- **Κλειδί (key):** Ένα γνώρισμα μιας οντότητας την προσδιορίζει με μοναδικό τρόπο και την διακρίνει από τις άλλες. Το χαρακτηριστικό αυτό ονομάζεται κύριο κλειδί.
- **Ξένο Κλειδί:** Ένα γνώρισμα μιας οντότητας που είναι κύριο κλειδί σε μια οντότητα αλλά υπάρχει και σε μια άλλη οντότητα σαν απλό γνώρισμα, λέγεται ξένο κλειδί. Τα ξένα κλειδιά είναι απαραίτητα για να μπορέσουμε να κάνουμε τις συσχετίσεις (συνδέσεις, επικοινωνίες) ανάμεσα στις οντότητες.

Ονομασίες Πεδίων:

Η access παρέχει ένα μεγάλο βαθμό ελευθερίας όσον αφορά τις ονομασίες των πεδίων. Υπάρχουν όμως και κάποιοι περιορισμοί τους οποίους θα πρέπει να αναφέρουμε. Το ίδιο επίσης υσχύει και ως προς τις ονομασίες των στοιχείων ελέγχου και και διάφορων αντικειμένων. Έτσι το όνομα ενός πεδίου:

- Μπορεί να έχει μήκος έως μέχρι και 64 χαρακτήρες και να περιέχει γράμματα, αριθμούς ή κενά διαστήματα. Για παράδειγμα μπορούμε να γράφουμε ως όνομα πεδίου: Αριθμός Τηλεφώνου 1.
- Δεν επιτρέπεται να οριστεί ως πρώτος χαρακτήρας ενός πεδίου το κενό διάστημα.
- Δεν επιτρέπεται να χρησιμοποιηθεί η τελεία, το θαυμαστικό και οι αγκύλες.
- Δεν επιτρέπεται να εισάγουμε χαρακτήρες ελέγχου (τιμές ASCII από το 0 μέχρι το 31).
- Πρέπει να αποφεύγονται οι δεσμευμένες λέξεις όπως DATE, NOW, TIME, διότι αποτελούν συναρτήσεις της Access.
- Επίσης όταν ένα πεδίο εμφανίζεται σε δύο πίνακες για να αποθηκευθεί η σύγκυση θα πρέπει να χρησιμοποιείται το ίδιο όνομα και στους δύο πίνακες.



Εικόνα 3: Σύνδεση πινάκων δεδομένων μέσω πεδίων.

Οι πίνακες των δεδομένων μιας βάσης έχουν ορισμένα χαρακτηριστικά, τα οποία είναι τα εξής:

- Κάθε πίνακας μίας βάσης θα πρέπει να έχει όνομα το οποίο θα είναι μοναδικό για κάθε πίνακα της ίδιας βάσης.
- Κάθε κελί θα πρέπει να περιέχει μία και μοναδική τιμή.
- Κάθε στήλη ενός πίνακα θα πρέπει να έχει μοναδικό όνομα.
- Οι τιμές κάθε κελιού της ίδιας στήλης θα πρέπει να ανήκουν στον ίδιο τομέα.
- Η σειρά των στηλών δεν παίζει κανένα ρόλο.
- Κάθε σειρά ενός πίνακα είναι μοναδική, δηλ. δεν υπάρχουν στον ίδιο πίνακα δύο σειρές με όλα τα δεδομένα ίδια.
- Η σειρά των σειρών ενός πίνακα θεωρητικά δεν έχει καμία σημασία παρόλο που πρακτικά ίσως να έχει επίπτωση στην ταχύτητα προσπέλασης της βάσης.

ΚΕΦΑΛΑΙΟ - 3

ΟΡΟΙ & ΕΝΝΟΙΕΣ – ΘΕΩΡΙΑ & ΟΡΙΣΜΟΙ

3.1 Η έννοια του Portal

Portal είναι μια ειδικά σχεδιασμένη ιστοσελίδα η οποία συγκεντρώνει πληροφορίες από διάφορες πηγές με ένα ενιαίο τρόπο. Συνήθως, κάθε πηγή πληροφόρησης παίρνει ειδικό χώρο στη σελίδα για την εμφάνιση πληροφοριών (ένα συστατικό στοιχείο πύλης).

Συχνά, ο χρήστης μπορεί να ρυθμίσει ποια να εμφανιστεί. Παραλλαγές του portal αποτελούν το Mashup (υβριδική web εφαρμογή) και το intranet “dashboards” για εταιρικά στελέχη και διευθυντές. Η έκταση στην οποία εμφανίζεται το περιεχόμενο του portal με “ομοιόμορφο τρόπο” μπορεί να εξαρτάται από το σκοπό και τους χρήστες που προορίζονται να εξυπηρετήσουν, καθώς και την ποικιλομορφία του περιεχομένου.

Πρόκειται για ένα site βασισμένο πάνω σε έναν κατάλογο πληροφοριών Internet το οποίο προσπαθεί να κάνει τους χρήστες να το επισκέπτονται κάθε φορά που ζητούν κάτι μέσα από το δίκτυο και δεν γνωρίζουν πού θα το βρουν. Αν ζητούν κάτι εξειδικευμένο τότε το Portal τους κατευθύνει στο ανάλογο site (αν υπάρχει). Αν όμως ζητούν κάτι κοινότυπο ή δημοφιλές (π.χ. ειδήσεις) τότε το Portal προσπαθεί να τους οδηγήσει στις υπηρεσίες που προσφέρει το ίδιο.

Η κατασκευή Portal για μια επιχείρηση μπορεί να αποτελέσει τον ακρογωνιαίο λίθο όχι μόνο της διαδικτυακής της παρουσίας, αλλά με τη κατασκευή του Portal μια εταιρία θα μπορούσε να χαράξει μια νέα δυναμική προσέγγιση στις υπηρεσίες ή και προϊόντα τα οποία διαθέτει. Παραδείγματα των πρώτων δημόσιων διαδικτυακών πυλών ήταν AOL, Excite, Netvibes, iGoogle, MSN, Naver, Indiatimes, Rediff, Sify και το Yahoo.

3.2 Η γλώσσα HTML

HTML είναι μια γλώσσα προγραμματισμού που επινοήθηκε για να επιτρέψει τη δημιουργία ιστοσελίδων. Οι ιστοσελίδες αυτές μπορεί στη συνέχεια να προβληθούν από οποιονδήποτε άλλον που σχετίζεται με το Διαδίκτυο. Είναι σχετικά εύκολη στην εκμάθηση, καθώς τα βασικά στοιχεία της γίνονται κατανοητά για τους περισσότερους ανθρώπους σε μία συνεδρίαση, αλλά και αρκετά ισχυρή σε ό, τι σας επιτρέπει να δημιουργήσετε. Είναι συνεχώς υπό αναθεώρηση και εξέλιξη για να ανταποκριθεί στις απαιτήσεις και τις ανάγκες της αυξανόμενης χρήσης του Διαδικτύου, υπό τη διεύθυνση του W3C, του οργανισμού που είναι επιφορτισμένος με το σχεδιασμό και τη διατήρηση της γλώσσας.

Ο ορισμός της HTML είναι HyperText Markup Language. HyperText είναι η μέθοδος με την οποία οι χρήστες κινούνται στο διαδίκτυο (κάνοντας κλικ στο ειδικό κείμενο που ονομάζεται υπερ-συνδέσεις που θα φέρει στην επόμενη σελίδα). Το γεγονός ότι είναι υπερ απλά σημαίνει ότι δεν είναι γραμμική δηλαδή, μπορούμε να πάμε σε οποιαδήποτε θέση στο Internet όποτε θέλουμε, κάνοντας κλικ σε συνδέσμους, δεν υπάρχει καθορισμένη σειρά για

να κάνει την περιήγηση του ο χρήστης. Markup είναι αυτό που κάνουν οι ετικέτες HTML στο κείμενο που περικλείεται ανάμεσά τους. Μπορούν να του δώσουν μια συγκεκριμένη μορφή (πλαγιαστά κείμενο, για παράδειγμα). HTML είναι μια γλώσσα, καθώς έχει κωδικοποιημένες λέξεις (εντολές) και σύνταξη, όπως οποιαδήποτε άλλη γλώσσα.

Η HTML αποτελείται από μια σειρά σύντομων κωδικοποιημένων λέξεων δακτυλογραφημένων σε ένα αρχείο κειμένου από τον συγγραφέα της ιστοσελίδας, αυτές είναι οι ετικέτες. Το κείμενο στη συνέχεια αποθηκεύεται ως αρχείο html, για να προβληθεί μέσω ενός browser, όπως ο Internet Explorer ή το Mozilla Firefox. Αυτό το πρόγραμμα περιήγησης διαβάζει το αρχείο και μετατρέπει το κείμενο σε μια ορατή μορφή, ελπίζουμε καθιστώντας τη σελίδα όπως ο συγγραφέας είχε την πρόθεση να την παρουσιάσει. Για να δημιουργήσει κάποιος χρήστης την δική του HTML σελίδα προϋποθέτει την σωστή χρήση κατάλληλων ετικετών για να επιτευχθεί το επιθυμητό αποτέλεσμα. Μπορείτε να χρησιμοποιήσετε από ένα υποτυπώδες επεξεργαστή κειμένου μέχρι ένα ισχυρό γραφικό επεξεργαστή κειμένου για να δημιουργήσετε σελίδες HTML.

Οι ετικέτες είναι αυτό που διαχωρίζει το κανονικό κείμενο από τον κώδικα HTML. Μπορείτε να τους αναγνωρίσετε, όπως τα λόγια μεταξύ <angle-brackets>. Διαφορετικές ετικέτες εκτελούν διάφορες λειτουργίες. Οι ίδιες οι ετικέτες δεν εμφανίζονται όταν προβάλετε τη σελίδα σας μέσω ενός browser, αλλά τα αποτελέσματά τους εμφανίζονται. Η απλούστερες ετικέτες δεν κάνουν τίποτα περισσότερο από απλή μορφοποίηση σε κείμενο.

Δεδομένου ότι οι ιστότοποι έγιναν πιο δημοφιλής και οι ανάγκες αυξήθηκαν τόσο πολύ που η HTML δεν επαρκούσε για να τις καλύψει δημιουργήθηκαν πολλές άλλες γλώσσες που υποστηρίζουν την HTML και επιτρέπουν την δημιουργία νέων πραγμάτων που πολλαπλασιάζουν τις δυνατότητές της. Τέλος η HTML τροποποιείται κάθε λίγα χρόνια για να ανοίξει ο δρόμος για βελτιώσεις.

3.3 Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) είναι πλέον ο συνήθης τρόπος για να καθορίσουμε την παρουσίαση των σελίδων HTML, από τις γραμματοσειρές και τα χρώματα στην πλήρη διάταξη μιας σελίδας. Είναι μια γλώσσα προγραμματισμού που ανήκει στην κατηγορία των γλωσσών φύλλων στυλ που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης.

Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε σε γλώσσα HTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστοτόπου. Είναι πολύ πιο αποτελεσματική από τη χρήση HTML σε κάθε σελίδα για να ορίσετε την εμφάνιση του site σας. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη.

3.4 Javascript

Η JavaScript είναι μια γλώσσα δέσμης ενεργειών (scripting language) που σχεδιάστηκε κατά κύριο λόγο για την προσθήκη διαδραστικότητας σε ιστοσελίδες και τη δημιουργία εφαρμογών Web. Η γλώσσα εφαρμόστηκε για πρώτη φορά από την Netscape Communications Corp στις Netscape Navigator 2 beta (1995). Η JavaScript είναι διαφορετική από τη γλώσσα Java (που αναπτύχθηκε στη δεκαετία του 1990 στο Sun Microsystems).

Ωστόσο, οι δύο γλώσσες μπορούν να συνεργάζονται καλά. Προγράμματα JavaScript από την πλευρά του client(client side), ή δέσμες ενεργειών(scripts), μπορεί να ενσωματωθούν άμεσα στον πηγαίο κώδικα HTML των ιστοσελίδων. Ανάλογα με την πρόθεση του προγραμματιστή, scripts μπορούν να τρέξουν όταν ο χρήστης ανοίγει τη σελίδα Web, κάνει κλικ ή σέρνει κάποιο στοιχείο της σελίδας με το ποντίκι, πιέζει ένα κουμπί στο πληκτρολόγιο, υποβάλλει μια φόρμα, ή αφήνει τη σελίδα.

3.5 Asynchronous JavaScript and XML

Ajax (Asynchronous JavaScript and XML) είναι μια ομάδα αλληλένδετων τεχνικών ανάπτυξης ιστοσελίδων που χρησιμοποιούνται στην πλευρά του πελάτη για να δημιουργήσουμε ασύγχρονες εφαρμογές web. Με την τεχνολογία Ajax, web εφαρμογές μπορούν να στείλουν δεδομένα και να ανακτήσουν δεδομένα από ένα διακομιστή ασύγχρονα (στο παρασκήνιο), χωρίς να παρεμβαίνει με την εμφάνιση και τη συμπεριφορά της υπάρχουσας σελίδας. Τα δεδομένα μπορούν να ανακτηθούν χρησιμοποιώντας το αντικείμενο XMLHttpRequest. Παρά το όνομα, η χρήση της XML δεν απαιτείται (JSON χρησιμοποιείται συχνά αντ' αυτού), καθώς και τα αιτήματα δεν χρειάζεται να είναι ασύγχρονα.

Η τεχνολογία Ajax δεν είναι μια ενιαία τεχνολογία, αλλά μια ομάδα τεχνολογιών. HTML και CSS μπορεί να χρησιμοποιηθεί σε συνδυασμό για τη δημιουργία και τις πληροφορίες στυλ μιας σελίδας. Το DOM είναι προσβάσιμο με JavaScript για να εμφανίζει δυναμικά και να επιτρέπει στο χρήστη να αλληλεπιδρά με τις πληροφορίες που παρουσιάζονται. Η JavaScript και το αντικείμενο XMLHttpRequest παρέχει μια μέθοδο για την ανταλλαγή δεδομένων ασύγχρονα μεταξύ browser και server για να αποφύγει την πλήρη επαναφόρτωση της σελίδας.

Η τεχνολογία AJAX αυτή τη στιγμή αποτελεί τη πιο σύγχρονη τεχνολογία στον προγραμματισμό στο internet, δίνοντας διαδραστικές δυνατότητες σε ένα δυναμικό site, μετατρέποντας το από ένα απλό site σε μια διαδικτυακή εφαρμογή.

3.6 PHP

Η PHP είναι μια γλώσσα προγραμματισμού γενικής χρήσης υψηλού επιπέδου, που σχεδιάστηκε για τη δημιουργία δυναμικών ιστοσελίδων. Με τον όρο δυναμική εννοείται μια ιστοσελίδα που μπορεί να τροποποιούνται από το διαχειριστή τους online ή να διαφοροποιείται, ανάλογα με τα χαρακτηριστικά του χρήστη που τις προβάλλει, όπως για παράδειγμα, το λειτουργικό του σύστημα, η διεύθυνση IP του κ.ά.. Η σύνταξη της έχει δανειστεί στοιχεία από γλώσσες όπως C, Java και Perl με κάποιες φυσικά αυθεντικές προσθήκες και χαρακτηριστικά.

Το 1994 ήταν η χρονιά που δημιουργήθηκε η PHP, όπου ήταν αρχικά η δουλειά ενός ατόμου και συγκεκριμένα του Rasmus Lerdorf. Το 2011 ήταν σε χρήση περίπου σε δέκα εκατομμύρια τομείς παγκόσμια και αυτός ο αριθμός αυξάνεται γρήγορα. Το όνομα PHP/FI, όπως αρχικά ονομάστηκε η PHP, σήμαινε *Personal Home Page/Form Interpreter* (προσωπική αρχική σελίδα), αλλά στη συνέχεια άλλαξε σύμφωνα με μια σύμβαση, τη GNU και τώρα τα αρχικά PHP σημαίνουν *Hypertext Preprocessor* (προεπεξεργαστής κειμένων

PHP). Η σημερινή βασική έκδοση που χρησιμοποιείται κατά κόρον, όπως φαίνεται και παρακάτω είναι η PHP 5.

Η PHP χρησιμοποιείται για τον χειρισμό των λειτουργιών και εργασιών που θα διεκπεραιώνει μια ιστοσελίδα. Ο κώδικας σε γλώσσα PHP που γράφεται για μια ιστοσελίδα δεν γίνεται άμεσα αντιληπτός αλλά μετά από την επέμβαση του χρήστη στην ιστοσελίδα. Παραδείγματα χρήσης PHP για τον χειρισμό ιστοσελίδων είναι η εγγραφή χρηστών (user registration), τα forum, τα wiki κ.ά..

Μια σελίδα PHP περνά από επεξεργασία από ένα συμβατό διακομιστή του Παγκόσμιου Ιστού (π.χ. Apache), ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο, που θα σταλεί στο πρόγραμμα περιήγησης των επισκεπτών σε μορφή κώδικα HTML. Μπορεί να χρησιμοποιηθεί για την διαχείριση δεδομένων μέσα σε βάσεις σε συνδυασμό με την MySQL.

Μπορεί δηλαδή να συνδέεται σε μια βάση δεδομένων με κατάλληλες συναρτήσεις και να εισάγει, να διαγράφει, να επεξεργάζεται και να παρουσιάζει εγγραφές. Ο συνδυασμός Linux-Apache-PHP-MySQL, είναι η πιο δημοφιλής πλατφόρμα εκτέλεσης ιστοσελίδων και είναι γνωστός με το ακρωνύμιο LAMP. Παρόμοια, ο συνδυασμός */Apache/PHP/MySQL ονομάζεται *AMP, όπου το πρώτο αρχικό αντιστοιχεί στην πλατφόρμα, στην οποία εγκαθίστανται ο Apache, η PHP και η MySQL (π.χ. Windows, Mac OS X).

Η PHP μπορεί να συγκριθεί με τους βασικούς ανταγωνιστές της όπως η Perl, τα Microsoft Active Server Pages (ASP) και τα Java Server Pages (JSP). Σε σχέση λοιπόν με αυτά έχει πολλά πλεονεκτήματα:

✓ **Αποτελεσματικότητα και απόδοση**

Το γεγονός ότι με έναν απλό διακομιστή μπορεί να εξυπηρετηθεί ένας μεγάλος αριθμός επισκεπτών καθημερινά την καθιστά πολύ αποτελεσματική και καλύτερη από τους ανταγωνιστές της σύμφωνα με δοκιμές που δημοσιεύθηκαν από την Zend Technologies (<http://www.zend.com>).

✓ **Έτοιμες ενσωματωμένες βιβλιοθήκες**

Διαθέτει πολλές χρήσιμες λειτουργίες σχετικές με το WEB η οποίες είναι ενσωματωμένες σε βιβλιοθήκες. Δημοφιλής λειτουργίες του διαδικτύου όπως ηλεκτρονικό ταχυδρομείο, δημιουργία PDF, δημιουργία εικόνων δυναμικά, σύνδεση με άλλες υπηρεσίες δικτύου και πολλές άλλες γίνονται με λίγες γραμμές κώδικα.

✓ **Διασύνδεση με βάσεις δεδομένων**

Η PHP δίνει την δυνατότητα στον προγραμματιστή να συνδεθεί γρήγορα και απλά με πολλά συστήματα βάσεων δεδομένων όπως MySQL, PostgreSQL, Oracle, Sybase καθώς και με τις βάσεις δεδομένων της Microsoft με την χρήση ενός προγράμματος οδήγησης Open Database Connectivity Standard (ODBC).

✓ **Ευκολία μάθησης**

Όπως αναφέρθηκε και παραπάνω η σύνταξη της PHP έχει δανειστεί στοιχεία από γλώσσες όπως C, Java και Perl γεγονός που την κάνει εύκολη στην εκμάθηση και την χρήση από προγραμματιστές που γνωρίζουν αυτές της γλώσσες προγραμματισμού.

✓ **Χαμηλό κόστος**

Η PHP είναι γλώσσα ανοιχτού κώδικα και είναι δωρεάν . Μπορείτε να κατεβάσετε την τελευταία έκδοση από το <http://www.php.net> , χωρίς χρέωση.

✓ **Μεταφερσιμότητα**

Ο κώδικας που γράφει ένας προγραμματιστής σε PHP δουλεύει τις περισσότερες φορές χωρίς αλλαγές σε οποιοδήποτε σύστημα τρέχει PHP. Αυτό γίνεται γιατί η PHP είναι διαθέσιμη σε πολλά λειτουργικά συστήματα όπως LINUX (Solaris, Ubuntu κ.α.) αλλά και σε εμπορικά λειτουργικά όπως Microsoft Windows.

✓ **Ανοιχτός κώδικας προέλευσης**

Η PHP είναι γλώσσα ανοιχτού κώδικα όπως έχει αναφερθεί παραπάνω και ως εκ τούτου ότι αλλαγές θέλει να κάνει η προγραμματιστής για να εξυπηρετήσει τις ανάγκες της εφαρμογής του μπορεί να τις κάνει και να επεκτείνει τις δυνατότητες της γλώσσας χωρίς να περιμένει τον κατασκευαστή για διορθώσεις, προσθήκες και αναβαθμίσεις.

3.7 MySQL

Η MySQL είναι ένα πολύ δυνατό και γρήγορο, σύστημα διαχείρισης βάσεων δεδομένων. Μια βάση δεδομένων επιτρέπει να αποθηκεύει ,να αναζητάει, να ταξινομεί και να ανακαλεί τα δεδομένα που επιθυμεί ένας χρήστης πολύ αποτελεσματικά.

Ο MySQL διακομιστής ελέγχει την πρόσβαση στα δεδομένα, για να μπορούν να δουλεύουν πολλοί χρήστες ταυτόχρονα , για να παρέχει γρήγορη πρόσβαση και να διασφαλίζει ότι μόνο πιστοποιημένοι χρήστες μπορούν να έχουν πρόσβαση. Άρα, η MySQL είναι ένας πολυνηματικός διακομιστής πολλαπλών χρηστών.

Χρησιμοποιεί την SQL (Structured Query Language) την τυπική γλώσσα ερωτημάτων για βάσεις δεδομένων. Η MySQL μετρά πάνω από 11 εκατομμύρια χρήστες παγκοσμίως, είναι διαθέσιμη από το 1996 αν και η ιστορία της ξεκινά από το 1979.

Οι κυριότεροι ανταγωνιστές της MySQL είναι οι PostgreSQL, Microsoft SQL και Oracle. Η MySQL έχει σαφώς πολλά πλεονεκτήματα, όπως εύκολη διαμόρφωση και μάθηση, χαμηλό κόστος και ο κώδικας προέλευσης είναι διαθέσιμος. Η MySQL είναι χωρίς αμφιβολία πολύ γρηγορότερη από τους ανταγωνιστές της.

ΚΕΦΑΛΑΙΟ - 4

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ SQL

4.1. Ιστορικά στοιχεία

- Sequel: Η SQL σχεδιάστηκε από την IBM και αρχικά ονομαζόταν SEQUEL (Structured English Query Language)
- SQL: Σήμερα το SQL σημαίνει Structured Query Language και αποτελεί την κοινή γλώσσα όλως των Σχεσιακών ΣΔΒΔ
- Προφορά: Τα αρχικά SQL διαβάζονται διεθνώς.
- SEQUEL ... και στην Ελλάδα S-Q-L.
- Υλοποιήσεις της SQL
 - SQL-1 SQL-86
 - SQL-2 SQL-92
 - SQL-3 SQL-99

4.2. Συσχετισμοί

Ένας από τους βασικούς λόγους της απόλυτης επικράτησης των Σχεσιακών ΒΔ είναι ότι εξασφαλίζει μεταφερσιμότητα μεταξύ των Σχεσιακών ΣΔΣΒΔ. Υπάρχουν βέβαια διαφοροποιήσεις μεταξύ των υλοποιήσεων της SQL στα εμπορικά ΣΔΣΒΔ όμως με λίγο προσοχή αυτά ξεπερνιούνται. Επίσης μπορούμε να γράφουμε SQL εντολές (προγράμματα) και αυτές να εκτελούνται σε πολλές και διαφορετικές μεταξύ τους πλατφόρμες.

Η γλώσσα SQL σχετίζεται με τη σχεσιακή άλγεβρα. Οι πράξεις της Σχεσιακής Άλγεβρας παίζουν σημαντικό ρόλο για την κατανόηση των διαφόρων αιτημάτων προς το Σχεσιακό ΣΔΒΔ αλλά επιπλέον και για την επεξεργασία και τη βελτιστοποίηση ερωτήσεων σε ένα Σχεσιακό ΣΔΒΔ.

Οι διατυπώσεις σχεσιακών πράξεων στην SQL είναι πιο απλή υπόθεση από ότι στην σχεσιακή άλγεβρα που πρέπει να ορίσουμε ακριβώς με ποια σειρά θα εκτελεστούν οι πράξεις μιας σύνθετης έκφρασης, ενώ στην SQL μπορούμε να περιγράψουμε σε πιο ψηλό επίπεδο την ερώτηση/πράξη και αναλαμβάνει το ΣΔΒΔ να την εκτέλεση με αποδοτικό τρόπο.

Στην SQL δηλώνουμε απλά τι θέλουμε να κάνουμε και βασίζεται στο σχεσιακό λογισμό πλειάδων, μια άλλη τυπική γλώσσα .

4.3. Δυνατότητες της SQL

Η SQL είναι μια πλήρης γλώσσα βάσεων δεδομένων που περιλαμβάνει εντολές για:

- ορισμό δεδομένων (ΓΟΔ – DDL Data Definition Language)
- ενημέρωση (ΓΧΔ – DML Data Manipulation Language)
- ερωτήσεις (ΓΧΔ – Query Language).

Επιπλέον, υποστηρίζει τον:

- ορισμό όψεων (views)
- προσδιορισμό ασφάλειας, δικαιωμάτων κτλ.
- ορισμό περιορισμών ακεραιότητας (integrity constraints)
- έλεγχο δοσοληψιών (transactions).

Επίσης σε πολλές γλώσσες προγραμματισμού γενικού σκοπού επιτρέπεται η εμφύτευση εντολών SQL, όπως είναι η Java, C/C++, C#, Pascal/Delphi, Visual Basic, php, ...

4.4. Ορισμός δεδομένων

Ορολογία:

- πίνακας (table) – σχέση
- γραμμή (row) – πλειάδα
- στήλη (column) - γνώρισμα

Επιπλέον έννοιες:

- Σχήμα (schema): Ομαδοποιούνται πίνακες και άλλα δομικά στοιχεία που αναφέρονται στην ίδια βάση δεδομένων, όπως όψεις, δικαιώματα, ...
- Κάθε σχήμα έχει ένα όνομα και κάποια ταυτότητα εξουσιοδότησης.

Στην ORACLE: Για κάθε λογαριασμό ομάδας ορίζεται αυτόματα και ένα δικό της σχήμα

- Κάθε ομάδα δουλεύει σε ένα δικό της σχήμα, ώστε να δουλεύει σε μια δικιά της ΒΔ.

4.5. Κατάλογος (Catalog)

Το Σύστημα Διαχείρισης Βάσεων Δεδομένων ένα σύνολο από μηχανισμούς και δεδομένα για τη διαχείριση των βάσεων δεδομένων. Μία κεντρική δομή είναι ο

ΚΑΤΑΛΟΓΟΣ (catalog) που περιλαμβάνει μια συλλογή σχημάτων. Αφορούν τον διαχειριστή της βάσης δεδομένων (DBA – Database Administrator).

4.6. Βασικές εντολές της SQL

- Αλλαγή Κωδικού (Password)
-Εντολή SQL για την αλλαγή password (στην oracle):

```
ALTER USER nikos  
IDENTIFIED BY m8yted;
```

- Διαχείριση Σχήματος ΒΔ

```
CREATE  
ALTER  
DROP
```

- Διαχείριση Δεδομένων

```
INSERT  
UPDATE  
DELETE
```

- Ερωτήματα (Queries)

```
SELECT
```

- DDL:CREATE TABLE

```
CREATE TABLE EMPLOYEE (  
    NAME VARCHAR(15)  
);
```

- DDL: DROP TABLE

```
CREATE TABLE EMPLOYEE  
(  
    NAME VARCHAR(15)  
);
```

```
DROP TABLE EMPLOYEE;
```

```
DROP TABLE EMPLOYEE CASCADE  
CONSTRAINTS;
```

- DDL: ALTER TABLE

```
CREATE TABLE EMPLOYEE
(
  ID INTEGER,
  NAME VARCHAR(15)
);
```

```
ALTER TABLE EMPLOYEE
ADD AGE INTEGER;
ALTER TABLE EMPLOYEE
DROP COLUMN AGE;
```

- **CONSTRAINTS**

```
CREATE TABLE EMPLOYEE
(
  ID INTEGER PRIMARY KEY,
  NAME VARCHAR(15) NOT NULL
);
```

```
ALTER TABLE ADD AGE;
DROP TABLE EMPLOYEE CASCADE
CONSTRAINTS;
```

- **Δημιουργία/Διαγραφή Πίνακα**

```
CREATE TABLE EMPLOYEE
(
  NAME VARCHAR(15)
);
```

```
DROP TABLE EMPLOYEE;
```

```
DROP TABLE EMPLOYEE CASCADE
CONSTRAINTS;
```

4.7. Περιορισμοί

Οι περιορισμοί του σχεσιακού μοντέλου δεδομένων:

- **Πεδίο Ορισμού:** Ο τύπος δεδομένων του πεδίου/στήλης του πίνακα.
- **Κλειδί**
 - Primary Key
 - Unique
- **Ακεραιότητα Οντοτήτων**
- **Αναφορική Ακεραιότητα**

```
ALTER TABLE employee ADD CONSTRAINT
FK_PROIST FOREIGN KEY(employer)
References employee(ID);
```

4.8. Δημιουργία πίνακα

```
CREATE TABLE supermarket.EMPLOYEE
(
NAME VARCHAR(15)
);

CREATE TABLE warehouse.SUPPLIER (
NAME VARCHAR(15)
);
```

- Για να δει ένας χρήστης όλους τους πίνακές του:

```
SELECT TABLE_NAME
FROM USER_TABLES;
```

- Για να δει ο DBA όλους τους πίνακες ενός χρήστη:

```
SELECT * FROM DBA_TABLES
WHERE OWNER = 'supermarket';
```

4.9. DML: Insert

Στην απλή της μορφή, χρησιμοποιείται για την εισαγωγή μιας ή περισσότερων πλειάδων (γραμμών/εγγραφών) σε μια σχέση/πίνακα. Η σειρά των γνωρισμάτων πρέπει να είναι ίδια με τη σειρά των γνωρισμάτων/πεδίων/στηλών του πίνακα. Παράδειγμα:

```
INSERT INTO EMPLOYEE
VALUES ('Mathiou','Nikos', '694578622', '12-JUL-86',
Papanastasiou,Heraklion,Crete', 'M', 71305,
'54635424', 6 );
```

Εναλλακτικά η Insert μπορεί να χρησιμοποιηθεί δηλώνοντας ρητά τα γνωρίσματα τα οποία αντιστοιχούν στις τιμές της νέας πλειάδας:

Γνωρίσματα με τιμή NULL μπορούν να παραλειφθούν. Παράδειγμα:

- Εισαγωγή πλειάδας για έναν νέο εργαζόμενο EMPLOYEE για τον οποίο γνωρίζουμε μόνο το μικρό όνομα, το επίθετο και τον ΑΤ.

```
INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)
VALUES ('Mathiou', 'Nikos', '694578622'
);
```

4.10. DML: DELETE

Διαγράφει πλειάδες από μια σχέση. Με χρήση του όρου WHERE μπορούμε να προσδιορίζουμε τις πλειάδες που θα διαγραφούν. Μια εντολή delete διαγράφει πλειάδες από έναν πίνακα (εκτός εάν υπάρχουν περιορισμοί αναφορικής ακεραιότητας και η δοθεί συνθήκη CASCADE).

Το πλήθος των γραμμών που διαγράφει η εντολή delete εξαρτάται από τη συνθήκη where. Εάν δεν οριστεί συνθήκη WHERE-clause τότε η εντολή διαγράφει όλες τις γραμμές του πίνακα.

Σε αυτή την περίπτωση ο πίνακας θα δεν θα περιέχει καμία γραμμή μετά την εντολή delete (ο πίνακας όμως δεν θα έχει διαγραφεί !) Προσοχή στους περιορισμούς αναφορικής ακεραιότητας.

Παραδείγματα:

```
DELETE FROM EMPLOYEE
WHERE LNAME='Mathiou';

DELETE FROM EMPLOYEE
WHERE SSN='987654321';

DELETE FROM EMPLOYEE
WHERE DNO IN
    (SELECT DNUMBER
     FROM DEPARTMENT
     WHERE DNAME='Research'
    );

DELETE FROM EMPLOYEE;
```

4.11. DML: UPDATE

Μεταβολή των τιμών των πεδίων σε μία ή περισσότερες γραμμές του πίνακα.

- where: Μια συνθήκη where καθορίζει ποιες γραμμές θα τροποποιηθούν

- set: Το τμήμα SET ορίζει ποια γνωρίσματα θα ενημερωθούν και ποιες θα είναι οι νέες τιμές τους

Κάθε εντολή ενημερώνει πλειάδες που ανήκουν στην ίδια σχέση. Η αναφορική ακεραιότητα θα πρέπει να τηρηθεί.

Παράδειγμα 1:

Αλλαγή του πεδίου location και του υπεύθυνου τμήματος του project με αριθμό:

```
UPDATE PROJECT
SET PLOCATION = Papanastasiou',
DNUM = 5
WHERE PNUMBER=10;
```

Παράδειγμα 2:

Αύξηση 10% σε όλους τους εργαζόμενους του τμήματος 'Research':

```
U6: UPDATE EMPLOYEE
SET SALARY = SALARY *1.1
WHERE DNO IN (SELECT DNUMBER
FROM DEPARTMENT
WHERE DNAME='Research');
```

Παρατήρηση: Η νέα τιμή του πεδίου μισθός εξαρτάται από την αρχική τιμή του πεδίου. Η αναφορά στο πεδίο SALARY δεξιά από το “=” αναφέρεται στην αρχική τιμή αριστερά από το “=” αναφέρεται στην νέα τιμή.

4.12. Ερωτήματα SQL (Queries)

- Απλά ερωτήματα SQL
- Σύνθετα SQL Queries
- Εμφωλιασμένα SQL Queries

4.12.1. Απλά SQL Queries

Τα βασικά ερωτήματα SQL αντιστοιχούν στις πράξεις SELECT, PROJECT, και JOIN της σχεσιακής άλγεβρας. Ένα απλό παράδειγμα ερωτήματος σε έναν πίνακα:

- Query 0: Ανάκτηση της ημερομηνίας γέννησης και της διεύθυνσης του εργαζόμενου με όνομα: 'Nick M. Mathiou'.

```
Q0: SELECT      BDATE, ADDRESS
FROM            EMPLOYEE
WHERE           FNAME='Nick' AND MINIT='M'
               AND LNAME='Mathiou'
```

Μοιάζει με το ζευγάρι των πράξεων SELECT-PROJECT της σχεσιακής άλγεβρας. Το SELECT καθορίζει τα γνωρίσματα για το projection και το WHERE καθορίζει τη συνθήκη για το selection

Σημείωση: Στην Σ:Α το αποτέλεσμα επιτρέπεται να έχει πολλαπλές πλειάδες.

- Query 1: Ανάκτηση του ονόματος και της διεύθυνσης όλων των εργαζομένων του τμήματος 'Research':

```
SELECT          FNAME, LNAME, ADDRESS
FROM            EMPLOYEE, DEPARTMENT
WHERE           DNAME='Research'
AND             DNUMBER=DNO;
```

Μοιάζει με τις πράξεις SELECT-PROJECT-JOIN της σχεσιακής άλγεβρας

(DNAME='Research') είναι η συνθήκη επιλογής (selection condition)

(DNUMBER=DNO) είναι η συνθήκη join condition.

- Query 2: Για κάθε έργο με τοποθεσία in 'Heraklion', να βρεθούν ο κωδικός του έργου, το υπεύθυνο τμήμα, το όνομα, διεύθυνση, ημερ. γέννησης του διευθυντή του υπεύθυνου τμήματος.

```
SELECT          PNUMBER, DNUM, LNAME, BDATE, ADDRESS
FROM            PROJECT, DEPARTMENT, EMPLOYEE
WHERE           DNUM=DNUMBER AND MGRSSN=SSN
AND             PLOCATION='Heraklion';
```

Στο ερώτημα υπάρχουν 2 συνθήκες συνένωσης.

Η συνθήκη συνένωσης DNUM=DNUMBER σχετίζει το έργο με το υπεύθυνο τμήμα.

Η συνθήκη συνένωσης MGRSSN=SSN σχετίζει το υπεύθυνο τμήμα με τον εργαζόμενο που διευθύνει το τμήμα.

4.12.2. Σύνθετα SQL queries

Ασάφεια και Πλήρη Ονόματα

Ονόματα Πεδίων: Στην SQL, τα ονόματα των πεδίων σε κάθε σχέση είναι τοπικά.

Ιδια Ονόματα: Επομένως δύο πεδία που ανήκουν σε διαφορετικές σχέσεις (πίνακες), μπορούν να έχουν το ίδιο όνομα, χωρίς αυτό να σημαίνει ότι σχετίζονται.

Πρόθεμα: Όταν σε μια εντολή SQL υπάρχει περίπτωση να προκύψει ασάφεια μπορούμε να χρησιμοποιήσουμε στο όνομα του πεδίου ως πρόθεμα το όνομα της σχέσης στην οποία ανήκει.

Παράδειγμα:

Εάν υπάρχει πεδίο NAME στον πίνακα EMPLOYEE και πεδίο NAME στον πίνακα DEPARTMENT και εμφανίζονται και οι δύο πίνακες σε ένα ερώτημα SQL, μπορούμε να γράψουμε:

```
SELECT      EMPLOYEE.NAME, DEPARTMENT.NAME
FROM ...
```

Ψευδώνυμα (Aliases)

Μία σχέση είναι πιθανό να εμφανίζεται περισσότερες από μία φορές σε ένα ερώτημα SQL.

Ψευδώνυμα: Στην περίπτωση αυτή (αλλά και κάθε άλλη περίπτωση) μπορούμε να χρησιμοποιήσουμε ψευδώνυμα για το όνομα της σχέσης

Παράδειγμα:

Για κάθε εργαζόμενο, να βρεθεί το όνομα του εργαζομένου και το όνομα του προϊσταμένου του.

```
SELECT      E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM        EMPLOYEE E, EMPLOYEE S
WHERE       E.SUPERSSN=S.SSN
```

Παράδειγμα:

Για κάθε εργαζόμενο, να βρεθεί το όνομα του εργαζομένου και το όνομα του προϊσταμένου του

```
SELECT      E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM        EMPLOYEE E, EMPLOYEE S
WHERE       E.SUPERSSN=S.SSN
```

Τα ονόματα E και S είναι ψευδώνυμα για τη σχέση EMPLOYEE.

Μπορούμε να θεωρήσουμε το E και το S ως δύο διαφορετικά (υποθετικά) αντίγραφα της σχέσης EMPLOYEE. Το E αναπαριστά την οντότητα EMPLOYEE στο ρόλο του εργαζομένου. Το S αναπαριστά την οντότητα EMPLOYEE στο ρόλο του προϊσταμένου.

Ασαφή Ονόματα και Μετονομασία

```
SELECT      ONOMA, ERGAZOMENOS.ONOMA, DIEYHYNSH
FROM        ERGAZOMENOS, TMHMA
WHERE       TMHMA.ONOM = 'RESEARCH'
AND         TMHMA.KOD_TMHM=ERGAZOMENOS.KOD_TMHM;
SELECT      E.ONOMA, E.EPITHETO, S.ONOMA, S.EPITHETO
FROM        ERGAZOMENOS E, ERGAZOMENOS S
```

```
WHERE      E.PROISTAMENOS = S.AT;
SELECT     S.NAME AS ONOMA
FROM       STUDENT S, STUDENT T
WHERE      S.AM = T.AM;
```

Ερώτημα χωρίς τον όρο WHERE

WHERE: Όταν λείπει ο όρος WHERE από ένα ερώτημα, δεν ορίζεται καμία συνθήκη για την επιλογή των εγγραφών (πλειάδων) της σχέσης. Επομένως επιλέγονται όλες οι πλειάδες που παράγονται από τον όρο FROM.

WHERE TRUE: Ουσιαστικά είναι σα να χρησιμοποιούμε τη συνθήκη WHERE TRUE

Παράδειγμα:

Να ανακτηθούν οι αριθμοί ταυτότητας όλων των εργαζομένων.

```
SELECT     SSN
FROM       EMPLOYEE;
```

Καρτεσιανό Γινόμενο:

Εάν ορισθούν 2 ή περισσότερες σχέσεις στον όρο FROM και δεν υπάρχει όρος WHERE, το αποτέλεσμα του ερωτήματος θα είναι το καρτεσιανό γινόμενο

Παράδειγμα:

```
SELECT     SSN, DNAME
FROM       EMPLOYEE, DEPARTMENT;
```

Το σύμβολο *

Όλα τα γνωρίσματα: Δηλώνοντας το σύμβολο * στον όρο SELECT ζητούμε να εμφανισθούν όλα τα γνωρίσματα από τις πλειάδες που θα επιλεγούν

Παραδείγματα:

```
Q1C:      SELECT * FROM EMPLOYEE
          WHERE      DNO=5 ;

Q1D:      SELECT * FROM EMPLOYEE, DEPARTMENT
          WHERE      DNAME='Research'
          AND         DNO=DNUMBER ;
```

Αστερίσκος και Εντολή χωρίς Where

```
SELECT      NAME, AM
FROM        STUDENTS;
SELECT * FROM STUDENTS;
```

Distinct

Η SQL σε αντίθεση με το σχεσιακό μοντέλο δεδομένων δεν επιβάλλει να είναι κάθε σχέση ένα σύνολο. Επομένως επιτρέπεται να εμφανίζονται διπλές και γενικότερα πολλαπλές πλειάδες. Εάν θέλουμε να εξαλειφθούν τυχόν διπλές πλειάδες στο αποτέλεσμα ενός query μπορούμε να χρησιμοποιήσουμε τη λέξη κλειδί DISTINCT.

Παράδειγμα:

Η λίστα των μισθών όλων των εργαζομένων:

```
Q:11      SELECT      SALARY
          FROM        EMPLOYEE;

Q:11A:    SELECT      DISTINCT SALARY
          FROM        EMPLOYEE
```

Συνολοθεωρητικές Πράξεις

Η SQL υποστηρίζει βασικές συνολοθεωρητικές πράξεις

UNION: Η πράξη της ένωσης

INTERSECT: Τομή συνόλων

MINUS: Διαφορά συνόλων

Τα αποτελέσματα των πράξεων αυτών είναι σύνολα πλειάδων: Επομένως διπλές πλειάδες εξαλείφονται! Συμβατότητα: Οι συνολοθεωρητικές πράξεις εφαρμόζονται μόνο μεταξύ συμβατών σχέσεων. Οι σχέσεις θα πρέπει να έχουν τα ίδια γνωρίσματα και τα γνωρίσματα πρέπει να εμφανίζονται με την ίδια σειρά.

Σύνολα: Ένωση

```
SELECT      S.NAME AS ONOMA
FROM        STUDENT S
WHERE       ETOS = 3 UNION
SELECT      T.NAME AS ONOMA
FROM        STUDENT T
WHERE       ETOS = 4;
```

Με διπλότυπες γραμμές:

```
SELECT      S.NAME AS ONOMA FROM STUDENT S
```

```
WHERE          ETOS = 3
UNION ALL
SELECT         T.NAME AS ONOMA
FROM           STUDENT T
WHERE          ETOS = 4;
```

Σύνολα: Αφαίρεση

```
SELECT         DISTINCT S.NAME AS ONOMA
FROM           STUDENT S MINUS
SELECT         T.NAME AS ONOMA
FROM           STUDENT T
WHERE          T.SUPERVISOR='4897';
```

Σύνολα: Τομή

```
SELECT         DISTINCT S.NAME AS ONOMA
FROM           STUDENT S INTERSECT
SELECT         T.NAME AS ONOMA
FROM           STUDENT T
WHERE          T.SUPERVISOR = '4897';
```

Not In, Exists, Not Exists

Παράδειγμα:

Λίστα όλων των έργων στα οποία απασχολείται εργαζόμενος με επίθετο 'Smith' ή το τμήμα που έχει την ευθύνη του έργου έχει διευθυντή με επίθετο 'Smith'.

```
(SELECT        PNAME
FROM           PROJECT, DEPARTMENT, EMPLOYEE
WHERE          DNUM=DNUMBER
AND            MGRSSN=SSN
AND            LNAME='Smith')

UNION

SELECT        PNAME
FROM           PROJECT, WORKS_ON, EMPLOYEE
WHERE          PNUMBER=PNO AND ESSN=SSN
AND            LNAME='Smith')
```

Τιμές NULL στην SQL

Η SQL επιτρέπει ερωτήματα που ελέγχουν εάν μια τιμή είναι NULL (δεν υπάρχει, δεν προσδιορίζεται ή δεν εφαρμόζεται)

Σημείωση: Η τιμή NULL δεν ισούται με καμία άλλη τιμή. Μάλιστα δύο τιμές NULL δεν θεωρείται ότι είναι ίσες, δηλαδή η συνθήκη $NULL = NULL$ δεν είναι αληθής

IS (NOT) NULL: Για τον έλεγχο τιμών NULL η SQL χρησιμοποιεί τις συνθήκες IS NULL και IS NOT NULL.

Παράδειγμα:

Ανάκτηση των ονομάτων όλων των εργαζομένων που δεν έχουν επιβλέποντα

```
Q14:      SELECT      FNAME, LNAME
           FROM        EMPLOYEE
           WHERE        SUPERSSN IS NULL
```

Σημείωση: Εάν σε μια συνθήκη συνένωσης υπάρχουν πλειάδες με τιμές NULL στα πεδία συνένωσης, οι πλειάδες αυτές δεν θα περιλαμβάνονται στο αποτέλεσμα.

Τιμές NULL

```
SELECT * FROM      ERGAZOMENOS
WHERE             PROISTAMENOS IS NULL;
SELECT * FROM      ERGAZOMENOS
WHERE             PROISTAMENOS IS NOT NULL;
```

Between

```
SELECT      E.SSN ONOMA
FROM        EMPLOYEE E
WHERE       E.SALARY
BETWEEN    1000 AND 2000;
```

Ρητά σύνολα τιμών

Η SQL επιτρέπει στον where τη δήλωση ενός συγκεκριμένου συνόλου τιμών:

Παράδειγμα:

Ανάκτηση των αριθμών ταυτότητας των εργαζομένων που απασχολούνται σε ένα τουλάχιστον από τα έργα με κωδικό 1, 2 ή 3.

```
Q13:      SELECT      DISTINCT      ESSN
           FROM        WORKS_ON
           WHERE        PNO IN      (1, 2, 3)
```

Διάταξη των Αποτελεσμάτων

```
SELECT      T.NAME, T.AM
FROM        STUDENT T ORDER
```

```

BY          T.AM;

SELECT      T.NAME, T.AM
FROM        STUDENT T
ORDER BY    T.NAME ASC, T.AM DESC;

```

4.12.3. Εμφωλευμένα Ερωτήματα

Ένα ερώτημα SELECT το οποίο δηλώνεται στον όρο WHERE ενός άλλου εξωτερικού ερωτήματος.

Παράδειγμα:

Ανάκτηση των ονομάτων και των διευθύνσεων όλων των εργαζομένων του τμήματος με όνομα 'Research'.

```

SELECT      FNAME, LNAME, ADDRESS
FROM        EMPLOYEE
WHERE       DNO IN
            (SELECT      DNUMBER FROM DEPARTMENT
             WHERE       DNAME='Research')

```

Το εμφωλευμένο ερώτημα επιλέγει τον κωδικό τμήματος με όνομα 'Research'

Το εξωτερικό ερώτημα επιλέγει μια πλειάδα από τη σχέση EMPLOYEE εάν ο κωδικός τμήματος DNO της πλειάδας ανήκει στο αποτέλεσμα του εμφωλευμένου ερωτήματος. Ο τελεστής σύγκρισης IN συγκρίνει μια τιμή ν με ένα σύνολο ή multi-set τιμών V, και επιστρέφει TRUE εάν το ν είναι ένα από τα στοιχεία του V.

Γενικά, μπορούμε να έχουμε ερωτήματα SELECT με πολλά επίπεδα εμφωλιασμού. Σε περίπτωση αμφισημίας κατά την αναφορά σε γνώρισμα για το οποίο που δεν προσδιορίζεται επακριβώς η σχέση, η αναφορά γίνεται στο πιο εμφωλιασμένο επίπεδο του ερωτήματος.

Εμφωλευμένα Ερωτήματα SQL

```

SELECT      T.NAME, T.AM
FROM        STUDENT T
WHERE       T.SUPERVISOR IN
            (SELECT      P.AM
             FROM        PROFESSOR P

```

```
WHERE      P.NAME LIKE '%SHA'
);
```

Συσχετισμένα Εμφωλευμένα Ερωτήματα

Εάν μια συνθήκη στον όρο WHERE ενός εμφωλιασμένου ερωτήματος αναφέρεται σε γνώρισμα μιας σχέσης που δηλώνεται σε εξωτερικό ερώτημα, τα δύο ερωτήματα λέμε ότι είναι συσχετισμένα (correlated). Το αποτέλεσμα του εμφωλιασμένου ερωτήματος προσδιορίζεται για κάθε πλειάδα του εξωτερικού ερωτήματος. Η εσωτερική σχέση αποτιμάται μια φορά για κάθε πλειάδα της εξωτερικής σχέσης (λογική ερμηνεία του αποτελέσματος).

Παράδειγμα:

Εργαζόμενοι που έχουν συνονόματο εξαρτώμενο.

```
SELECT      E.ONOMA, E.EPITHETO
FROM        ERGAZOMENOS E
WHERE       E.AR_TAYT IN
            (SELECT      E_ARTAYT
              FROM        EKSARTWMENOS K
              WHERE       E.ONOMA=K.ONOMA_EKSARTWMENOY
              AND         E.FYLO = K.FYLO
            );
```

Παράδειγμα:

```
SELECT      E.ONOMA, E.EPITHETO
FROM        ERGAZOMENOS E
WHERE       E.AR_TAYT IN
            (SELECT      E_ ARTAYT
              FROM        EKSARTWMENOS K
              WHERE       E.ONOMA=K.ONOMA_EKSARTWMENOU
            ) ;
```

Στο παράδειγμα, το εμφωλιασμένο ερώτημα δίνει ένα διαφορετικά αποτέλεσμα για κάθε πλειάδα του εξωτερικού ερωτήματος.

Ένα ερώτημα που γράφεται με εμφωλιασμένους SELECT... FROM... WHERE... όρους και που χρησιμοποιεί τους τελεστές σύγκρισης = ή IN μπορεί πάντοτε να γραφεί ως ερώτημα χωρίς εμφωλιασμό.

Παράδειγμα:

Το προηγούμενο εμφωλιασμένο μπορεί να γραφεί ως

```
SELECT      E.Onoma, E.Epitheto
FROM        Ergazomenos E, Eksartwmenos D
WHERE       E. AR_TAYT=D.E_ ARTAYT
AND         E.Onoma=D.ONOMA_EKSARTWMENOU
```

Η αρχική SQL είχε και έναν τελεστή σύγκρισης CONTAINS για συσχετισμένα εμφωλιασμένα ερωτήματα. Ο τελεστής αυτός αφαιρέθηκε από τη γλώσσα, πιθανόν λόγω της δυσκολίας υλοποίησής του.

Παράδειγμα:

Οι Διευθυντές με έναν τουλάχιστον εξαρτώμενο:

```
SELECT      E.ONOMA, E.EPITHETO
FROM        ERGAZOMENOS E
WHERE EXISTS
            (SELECT * FROM      EKSARTWMENOS K
              WHERE            E.AR_TAYT=K.E_ ARTAYT)
AND EXISTS
            (SELECT * FROM      TMHMA T
              WHERE            T.DIEYTHYNTHS=E.AR_TAYT
            ) ;
```

Η συνάρτηση EXISTS

Η συνάρτηση EXISTS ελέγχει εάν το αποτέλεσμα ενός συσχετισμένου εμφωλευμένου ερωτήματος είναι κενό (δεν περιέχει πλειάδες) ή όχι. Μπορούμε να διατυπώσουμε το ερώτημα για εργαζόμενους με συνονόματο εξαρτώμενο με χρήση της συνάρτησης EXISTS:

```
SELECT      ONOMA, EPITHETO
FROM        ERGAZOMENOS E

WHERE       EXISTS

            (SELECT * FROM EKSARTWMENOS
WHERE       AR_TAYT=D.E_ARTAYT
AND         E.Όνομα=D.ΟΝΟΜΑ_EKSARTWMENΟΥ
            );
```

Παράδειγμα:

Ανάκτηση των ονομάτων των εργαζομένων που δεν έχουν εξαρτώμενα μέλη.

```
SELECT      ONOMA, EPITHETO
FROM        ERGAZOMENOS

WHERE       NOT EXISTS

            (SELECT * FROM EKSARTWMENOS
WHERE       AR_TAYT=D.E_ARTAYT);
```

Το συσχετισμένο εμφωλιασμένο ερώτημα ανακτά όλες τις πλειάδες της σχέσης EKSARTWMENOS που σχετίζονται με μία πλειάδα της σχέσης ERGAZOMENOS. Η συνάρτηση EXISTS είναι απαραίτητη για την εκφραστική ισχύ της γλώσσας SQL.

ΚΕΦΑΛΑΙΟ - 5

ΣΧΕΣΙΑΚΑ ΜΟΝΤΕΛΑ & ΠΙΝΑΚΕΣ

5.1. Πληθικότητα

Η πληθικότητα, περιγράφει τον αριθμό στιγμιότυπων ενός τύπου οντοτήτων που μπορούν σε μια συσχέτιση να αντιστοιχίζονται με μία οντότητα ενός άλλου τύπου. Έτσι μπορούμε να εντοπίσουμε τις συσχετίσεις μεταξύ των οντοτήτων ως μια αντίστοιχη σχέση των αντικειμένων στην αλυσίδα supermarket ως εξής:

- Μια αποθήκη μπορεί να έχει πολλά προϊόντα και ένα προϊόν μπορεί να είναι σε πολλές αποθήκες. Για κάθε προϊόν θέλουμε να αποθηκεύουμε τα αποθέματα που υπάρχουν στην αποθήκη. Είναι μια σχέση πολλά προς πολλά (N-M). Με πίνακα σύνδεσης Αποθέματα και πεδία Κωδικό Αποθήκης και Κωδικό Προϊόντος και Αποθέματα Προϊόντος.
- Ένα υποκατάστημα εφοδιάζεται από μια αποθήκη και μια αποθήκη μπορεί να εφοδιάζει πολλά υποκαταστήματα. Είναι μια σχέση πολλά προς ένα (N-1)
- Κάθε υπάλληλος δουλεύει σε ένα και μόνο ένα υποκατάστημα και ένα υποκατάστημα έχει πολλούς υπαλλήλους. Είναι μια σχέση πολλά προς ένα (N-1)
- Κάθε προμηθευτής συνδέεται με πολλά προϊόντα (τουλάχιστον 1) και κάθε προϊόν παρέχεται από πολλούς προμηθευτές (τουλάχιστον 1). Κάθε προμηθευτής έχει μια τιμή πώλησης η οποία αλλάζει συχνά. Είναι μια σχέση πολλά προς πολλά (N-M). Με πίνακα σύνδεσης Τιμή Πώλησης και πεδία Κωδικός Προμηθευτή, Κωδικός Προϊόντος, Τιμή Πώλησης, Κατασκευαστής και Τιμή Προϊόντος.
- Κάθε προμηθευτής συνδέεται με πολλές αποθήκες και αντίστροφα. Είναι μια σχέση πολλά προς πολλά (N-M). Με πίνακα σύνδεσης Προμήθεια Αποθήκης και πεδία Κωδικός Προμηθευτή και Κωδικός Αποθήκης.

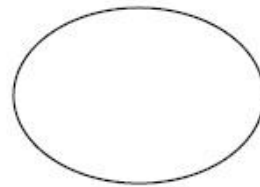
5.2. Σχέσεις

Σκοπός του κεφαλαίου είναι να δούμε τα διάφορα είδη σχέσεων και πως μπορούμε να τις δημιουργήσουμε. Ακόμα πως δημιουργείτε μια σχέση «ένα προς ένα», «ένα προς πολλά» και «πολλά προς πολλά».

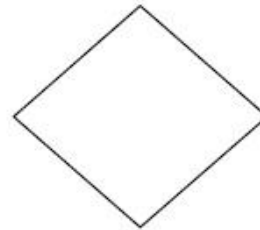
Οντότητα ----->



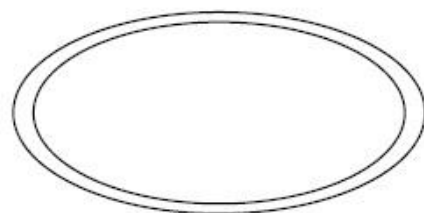
Γνώρισμα----->



Συσχέτιση----->



Πλειότιμο πεδίο ----->



Σχήματα οντοτήτων 1

5.2.1. Εισαγωγή στις σχέσεις

Μετά την δημιουργία των πινάκων, πρέπει να τους συνδέσετε μεταξύ τους ώστε να μπορείτε να φτιάξετε ερωτήματα, φόρμες και εκθέσεις. Αυτού του είδους η σύνδεση ονομάζεται σχέση. Προσθέστε πεδία σε πίνακες ή δημιουργήστε νέους πίνακες για να διευκρινίσετε τις απαιτούμενες σχέσεις.

Υπάρχουν τρεις τύποι σχέσεων:

- Ένα-προς-ένα (**one to one**): Αντιστοιχίζεται μια οντότητα ενός τύπου με το πολύ ή ακριβώς μια οντότητα ενός άλλου τύπου.
- 1-N Ένα-προς- πολλά (**one to many**): Αντιστοιχίζεται μια οντότητα ενός τύπου με κανένα, ένα ή πολλά στιγμιότυπα ενός άλλου τύπου.
- M-N Πολλά-προς- πολλά (**many to many**): Αντιστοιχίζεται κάθε στιγμιότυπο του ενός τύπου με ένα, κανένα ή πολλά στιγμιότυπα του άλλου τύπου

Σχέση ένα προς ένα (one to one):

Σε μια σχέση ένα προς ένα, μια εγγραφή του πρώτου πίνακα πρέπει να ταιριάζει σε μια εγγραφή του δεύτερου, και αντιστρόφως. Για παράδειγμα σε ένα σχολείο ένας καθηγητής διδάσκει μόνο ένα μάθημα και ένα μάθημα διδάσκει μόνο από έναν καθηγητή. Σχέση ένα προς ένα δημιουργείται αν και οι δύο οι σχετιζόμενες στήλες αποτελούν πρωτεύοντα κλειδιά ή να έχουν μοναδικές τιμές, δηλαδή η ιδιότητα Ευρετήριο των πεδίων να είναι αληθής χωρίς να επιτρέπονται διπλότυπα. Αυτός ο τύπος σχέσης δεν χρησιμοποιείται συχνά διότι τα δεδομένα που συσχετίζονται με τον τρόπο αυτόν συνήθως περιέχονται σε ένα μόνο πίνακα. Αυτός ο τύπος σχέσης μπορεί να χρησιμοποιηθεί:

- Αν έχετε ένα μεγάλο, πολύστηλο πίνακα και θέλετε να τον χωρίσετε σε δύο μικρότερους, τοποθετώντας μερικές στήλες στον έναν και μερικές στον άλλο πίνακα.
- Αν έχετε ένα πίνακα του οποίου κάποιες στήλες δεν θέλετε είναι ορατές για όλους τους χρήστες, οπότε πρέπει να τοποθετηθούν σε διαφορετικό πίνακα.

Σχέση ένα προς πολλά (one to many):

Μια σχέση ένα προς πολλά είναι ο πιο συνηθισμένος τύπος σχέσης. Για παράδειγμα σε ένα σχολείο μια αίθουσα ανήκει σε πολλούς μαθητές και αλλά οι μαθητές ανήκουν μόνο σε μία αίθουσα. Σε αυτόν τον τύπο σχέσης, μια γραμμή του πρώτου πίνακα μπορεί να έχει πολλές γραμμές που ταιριάζουν στον δεύτερο πίνακα, αλλά μια γραμμή του δεύτερου πίνακα μπορεί να έχει μόνο μία γραμμή που ταιριάζει του πρώτου. Για να είναι δυνατή η δημιουργία μιας τέτοιας σχέσης θα πρέπει μόνο το ένα από τα σχετιζόμενα πεδία να είναι πρωτεύον κλειδί ή να δέχεται μοναδικές τιμές.

Σχέση πολλά προς πολλά (many to many):

Σε μια σχέση πολλά προς πολλά, μια εγγραφή του πρώτου πίνακα μπορεί να ταιριάζει με πολλές εγγραφές του δεύτερου πίνακα. Με τον ίδιο τρόπο μία εγγραφή του δεύτερου πίνακα μπορεί να ταιριάζει με πολλές εγγραφές του πρώτου πίνακα. Για παράδειγμα σε μία βιβλιοθήκη ένα εγγεγραμμένος μπορεί να δανειστεί πολλούς τίτλους βιβλίων αλλά

και κάθε τίτλος να ενοικιάζεται από πολλούς εγγεγραμμένους. Για να είναι δυνατή η υλοποίηση μια τέτοιας σχέσης στην Access πρέπει να δημιουργηθεί ένας τρίτος πίνακας, που ονομάζεται πίνακας σύνδεσης. Του οποίου το πρωτεύον κλειδί πρέπει να αποτελείται από δύο πεδία, τα πρωτεύοντα κλειδιά των δύο σχετιζόμενων πινάκων. Στην ουσία μια σχέση πολλά προς πολλά αποτελείται από δύο σχέσεις ένα προς πολλά, μία μεταξύ του πρώτου πίνακα και του πίνακα σύνδεσης και μία του δεύτερου πίνακα και του πίνακα σύνδεσης.

5.2.2. Δημιουργία σχέσης

Στην συνέχεια, για να δημιουργηθεί μια σχέση θα πρέπει να βεβαιωθείτε ότι οι πίνακες που θα χρησιμοποιηθούν είναι κλειστές διότι το πρόγραμμα δεν επιτρέπει την δημιουργία σχέσεων ανάμεσα σε πίνακες που είναι ανοιχτοί. Στην συνέχεια επιλέγετε την εντολή «σχέσεις» και προσθέτετε τους πίνακες τους οποίους θέλετε να δημιουργήσετε την σχέση. Εφόσον εμφανίσετε τους πίνακες, επιλέγετε ποιόν τύπο σχέσης θέλετε.

Ένα προς πολλά:

- Σύρετε το πεδίο που θέλετε να συσχετίσετε από έναν πίνακα στο σχετικό πεδίο του άλλου πίνακα..
- Στις περισσότερες περιπτώσεις, μπορείτε να σύρετε το πεδίο πρωτεύοντος κλειδιού (το οποίο εμφανίζεται με έντονη γραφή) από έναν πίνακα σε ένα παρόμοιο πεδίο (συνήα με το ίδιο όνομα) που ονομάζεται ξένο κλειδί του άλλου πίνακα.
- Εμφανίζεται το παράθυρο διαλόγου Επεξεργασία σχέσεων. Βεβαιωθείτε ότι είναι σωστά τα ονόματα πεδίων που εμφανίζονται στις δύο στήλες. Μπορείτε να τις αλλάξετε, αν είναι απαραίτητο.
- Κάντε κλικ στο κουμπί δημιουργία για να δημιουργήσετε τη σχέση.

Ένα προς ένα:

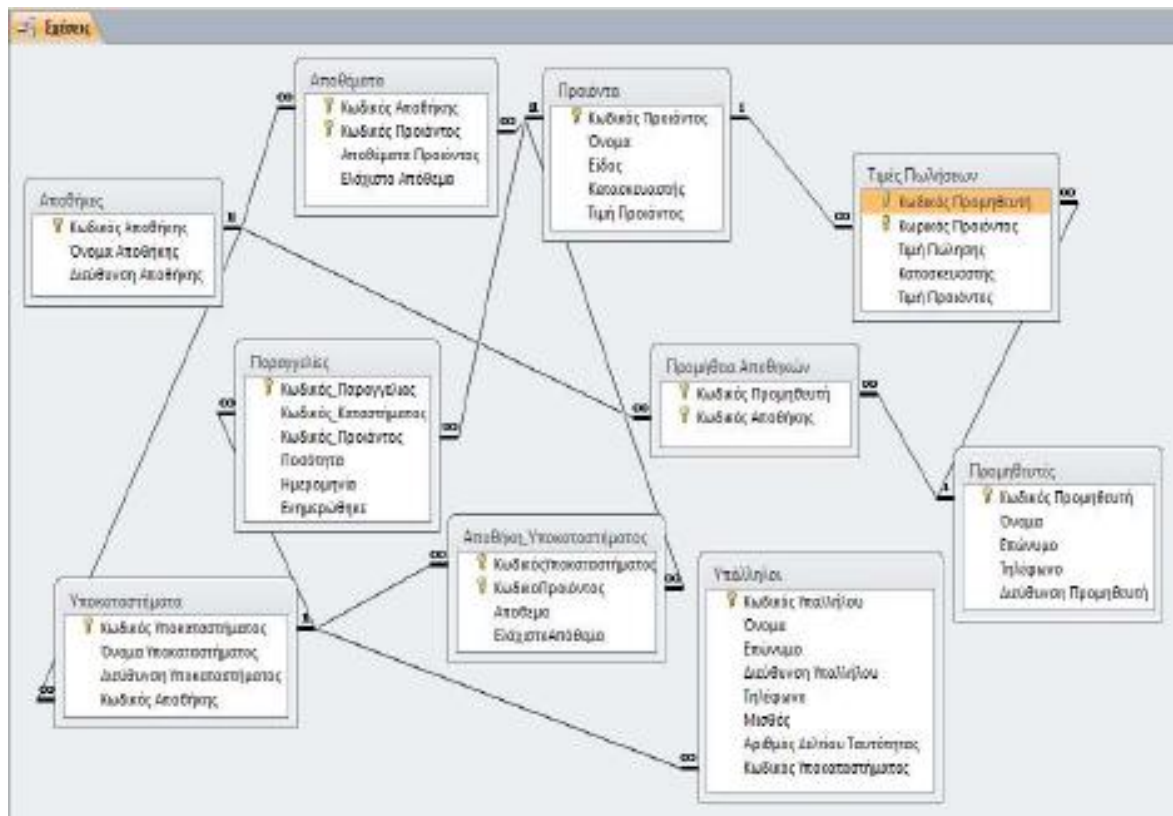
Για να δημιουργηθεί μια σχέση «ένα προς ένα» πρέπει να ακολουθήσετε την ίδια διαδικασία όπως με την δημιουργία μιας σχέσης «ένα προς πολλά» με την διαφορά ότι και τα δύο πεδία πρέπει να έχουν μοναδικές τιμές.

Πολλά προς πολλά:

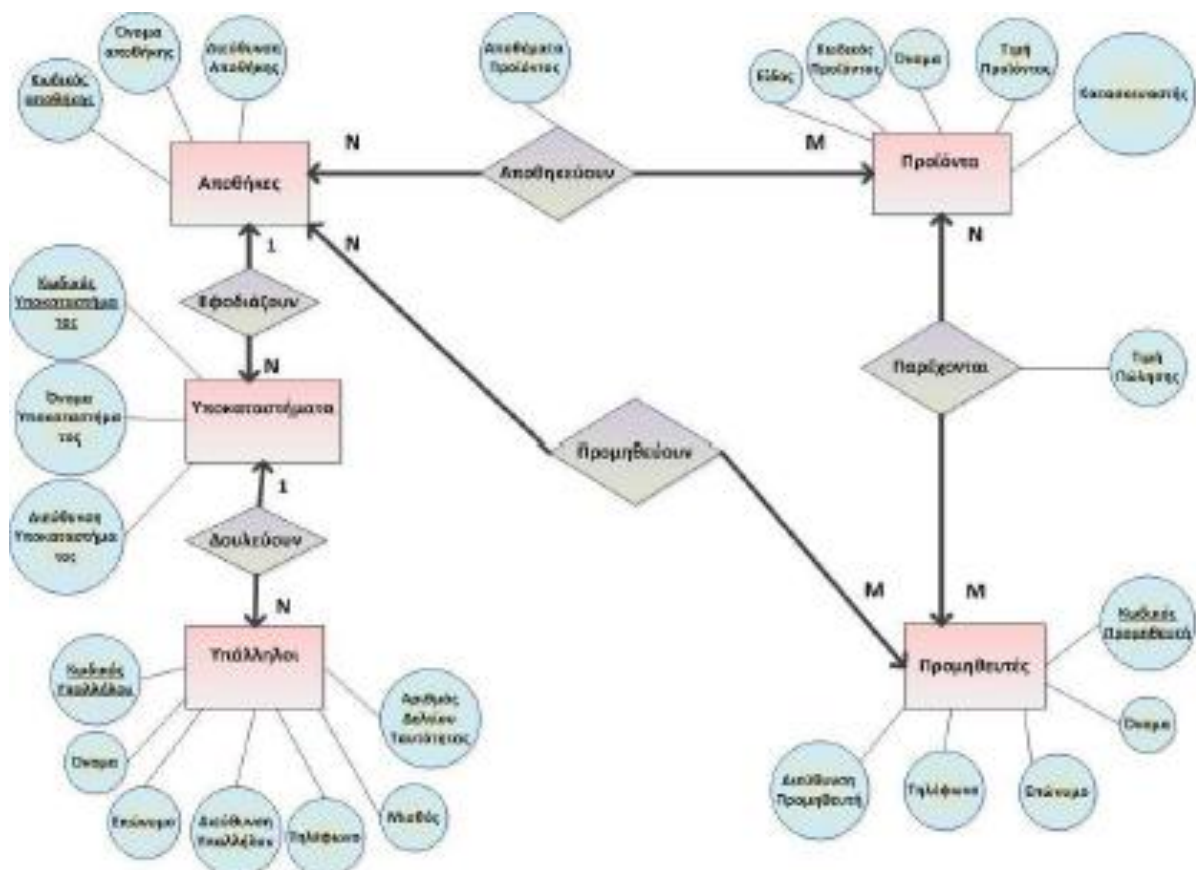
- Εμφανίστε τους δύο πίνακες που θα έχουν μια σχέση πολλά-προς-πολλά.
- Δημιουργήστε έναν τρίτο πίνακα, που ονομάζεται πίνακας σύνδεσης, και στη συνέχεια να προσθέσετε τα νέα πεδία πίνακα σύνδεσης με τους ίδιους ορισμούς όπως τα πεδία πρωτεύοντος κλειδιού από καθέναν από τους δύο άλλους πίνακες.
 - Στον πίνακα σύνδεσης πρωτεύον κλειδί είναι τα πρωτεύοντα κλειδιά των άλλων δύο πινάκων.
 - Σύρετε το πεδίο που θέλετε να συσχετίσετε από τον πρώτο πίνακα στο σχετικό πεδίο του πίνακα σύνδεσης και από τον πίνακα σύνδεσης σύρετε το πεδίο που θέλετε να συσχετίσετε στο σχετικό πεδίο του δεύτερου πίνακα.

5.2.3. Διαγραφή σχέσης

Για να διαγράψετε μια σχέση που έχετε δημιουργήσει μεταξύ δύο πινάκων, επιλέγετε την γραμμή της σχέσης στο παράθυρο Σχέσεις, πατάτε delete ή διαγραφή από το μενού Επεξεργασία και επιβεβαιώνετε την διαγραφή της σχέσης.



Σχέσεις μεταξύ οντοτήτων 1



Σχεσιακό μοντέλο βάσης δεδομένων 1

6.1. Προβολή βάσης δεδομένων σε περιβάλλον Java

Για να χρησιμοποιήσουμε μια βάση δεδομένων από μια γλώσσα προγραμματισμού χρειαζόμαστε έναν “driver”.

- **JDBC** είναι το API για τη Java και καθορίζει πως ένας πελάτης μπορεί να συνδεθεί με μια βάση δεδομένων. Σύνδεση με τη βδ, εκτέλεση ερωτήσεων και τροποποιήσεων και συλλογή των αποτελεσμάτων. Τεχνικά: είναι ένα σύνολο από κλάσεις στο java.sql package.

- Υπάρχει ένα JDBC για κάθε διαφορετικό ΣΔΒΔ

Υπάρχουν ορισμένα βήματα που θα πρέπει ν' ακολουθηθούν για την ανάπτυξη κώδικα Java που έχει πρόσβαση σε σχεσιακές βάσεις δεδομένων.

1. Φόρτωμα ενός οδηγού συμβατού με την προς επεξεργασία βάση δεδομένων. Αυτό μπορεί να γίνει προγραμματιστικά ή μπορεί να γίνει μέσω ενός αντικειμένου Properties που εγκαθίσταται από τον χρήστη. Στα παραδείγματα του βιβλίου και στις συνοδευτικές ασκήσεις ακολουθείται η πρώτη μέθοδος.
2. Εγκατάσταση σύνδεσης με την βάση.
3. Συσχετισμός μιας δήλωσης SQL με την σύνδεση.
4. Εκτέλεση της SQL δήλωσης.
5. Η δήλωση SQL θα παράγει έναν πίνακα που είναι αποθηκευμένος σε ένα αντικείμενο ResultSet. Το αντικείμενο θα περιέχει αναφορές στις σειρές του πίνακα που έχει σχηματιστεί από την εκτέλεση της δήλωσης SQL. Διατρέχονται οι σειρές του αντικειμένου ResultSet και εκτελείται η απαραίτητη επεξεργασία.
6. Πιθανώς εκτέλεση κι άλλων δηλώσεων SQL με τη διαδικασία που περιγράφηκε.
7. Όταν η επεξεργασία της βάσης δεδομένων τελειώσει και δεν χρειάζεται να εκτελεστούν άλλες SQL δηλώσεις, η σύνδεση με την βάση τερματίζεται.

Ο κώδικας για την εκτέλεση μιας απλής SELECT στον πίνακα Employees φαίνεται παρακάτω. Η δήλωση SELECT που έχουμε εδώ επιλέγει όλους τους υπαλλήλους που ο μισθός τους ξεπερνά τις £35000 τον χρόνο. Επεξεργαζόμαστε σειρά προς σειρά τον πίνακα που επιστρέφεται ως αποτέλεσμα αυτού του ερωτήματος και εκτυπώνουμε το όνομα του υπαλλήλου και τον μισθό του στο System.out.

```
import java.sql.*;

public class JDBCCode
{
    public static void main(String args[])
    {
        //Set the name of the file that is to be accessed
        //and the name of the driver
        String fileURL = "...";
        String driverName = "...";
        try
        {
            //Load the driver programmatically
            Class.forName(driverName);
        }
        catch (ClassNotFoundException cfn)
        {
            //Problem with driver, display error messages
            //and return to operating
            //system with status value 1
            System.out.println("Problem loading driver");
            System.exit(1);
        }
        try
        {
            //Establish a connection to the database,
            //second argument is the name
            //of the user and the third
            //argument is a password (blank)
            Connection con =
                DriverManager.getConnection(fileURL, "Darrel", "");
            //Create a statement object
            Statement selectStatement =
                con.createStatement();
            //Execute the SQL select statement
            ResultSet rs =
                selectStatement.executeQuery
                ("SELECT name, salary FROM
                 employees WHERE salary >35000");
            String employeeName;
            int employeeSalary;
            while(rs.next())
            {
                employeeName = rs.getString(1);
                employeeSalary = rs.getInt(2);
                System.out.println
                ("Name = "+ employeeName
                 +"Salary = "+ employeeSalary);
            }
            //Close down the database connection, result
```

```
set
                                //and the SELECT statement
                                selectStatement.close();
                                con.close();
                                rs.close();
                                }
                                catch (Exception e)
                                {
                                System.out.println("Problems with access to
database");
                                e.printStackTrace();
                                System.exit(2);
                                }
                                }
}
```

Ας δούμε όμως κομμάτια του κώδικα με περισσότερες λεπτομέρειες. Η static μέθοδος `Class.forName` (static σημαίνει εδώ πως δεν χρειάζεται να φτιάξουμε ένα πραγματικό αντικείμενο της κλάσης `Class` για να χρησιμοποιήσουμε τη μέθοδο αυτή) χρησιμοποιείται για το φόρτωμα του οδηγού. Το όνομα της κλάσης αποτελείται συνήθως από συμβολικά ονόματα χωρισμένα με τελείες.

Για παράδειγμα `my.SQL.Driver`. Σημειώστε πως πρέπει να βρείτε ποιος συγκεκριμένος οδηγός είναι ο κατάλληλος για τη βάση δεδομένων που χρησιμοποιείτε. Αν η κλάση δεν βρεθεί τότε προκαλείται μία `ClassNotFoundException` και το μπλοκ `catch` θα εμφανίσει ένα μήνυμα λάθους και θα τερματίσει το πρόγραμμα.

Αν το φόρτωμα του οδηγού ήταν επιτυχές στη συνέχεια θα γίνει σύνδεση με την βάση δεδομένων. Στο παραπάνω παράδειγμα αυτό επιτυγχάνεται με την δήλωση:

```
Connection con =
DriverManager.getConnection(fileURL, "Darrel", "");
```

Δηλώνεται ένα νέο αντικείμενο `Connection` και χρησιμοποιείται η μέθοδος `getConnection`, η οποία έχει τρία ορίσματα. Το δεύτερο είναι το όνομα χρήστη της βάσης δεδομένων και το τρίτο είναι ο αντίστοιχος κωδικός πρόσβασης (σ' αυτήν την περίπτωση δεν υπάρχει κωδικός πρόσβασης). Το πρώτο όρισμα είναι ένα Ενιαίος Εντοπιστής Πόρων (Uniform Resource Locator, URL): το URL είναι μία μορφή διεύθυνσης για αντικείμενα ή υπηρεσίες που υπάρχουν στον Παγκόσμιο Ιστό. Παρακάτω παρουσιάζεται ένα URL μιας βάσης δεδομένων:

```
jdbc:odbc://open.ac.uk/myFile
```

Το πρώτο τμήμα του URL δηλώνει το πρωτόκολλο επικοινωνίας με τον διακομιστή βάσης δεδομένων (εδώ δηλώνεται το `jdbc`). Το επόμενο τμήμα προσδιορίζει σε τί τύπο βάσης δεδομένων αναφέρεται, στην περίπτωση του παραπάνω URL η βάση δεδομένων είναι μία συμβατή με το πρότυπο ODBC βάση δεδομένων. Στη συνέχεια προσδιορίζεται ο υπολογιστής στον οποίο βρίσκεται η βάση δεδομένων (χρησιμοποιείται η σημειογραφία πεδίου που

παρουσιάστηκε στο Κεφάλαιο 2, εδώ ήταν το open.ac.uk). Τέλος υπάρχει το ίδιο το όνομα της βάσης δεδομένων, που στο παραπάνω παράδειγμα είναι myFile.

Στο επόμενο κομμάτι του κώδικα η μέθοδος createStatement δημιουργεί μία δήλωση που μπορεί να χρησιμοποιηθεί για την αποστολή δηλώσεων SQL στην βάση δεδομένων και η μέθοδος executeQuery εκτελεί την δήλωση SQL που αποτελεί το όρισμά της.

Στην συνέχεια μπορούμε να επεξεργαστούμε την απάντηση του ερωτήματος. Όπως είπαμε, η απάντηση θα έχει τη μορφή πίνακα. Αυτός θα έχει δύο στήλες: η πρώτη θα περιλαμβάνει τα ονόματα των υπαλλήλων που ο μισθός τους ξεπερνά τις £35000, η δεύτερη τον μισθό του υπάλληλου.

Ο πίνακας αυτός αντιπροσωπεύεται στον κώδικα Java από ένα αντικείμενο ResultSet και όπως αναφέραμε πρέπει να διατρέξουμε τις γραμμές του μια - μια. Η μέθοδος next του αντικειμένου ResultSet θα προχωράει το δείκτη του ResultSet στην επόμενη εγγραφή και θα επιστρέφει την τιμή true όταν υπάρχει επόμενη διαθέσιμη εγγραφή. Πάντα μόνο μια εγγραφή του ResultSet είναι διαθέσιμη, συνεπώς πρέπει να χρησιμοποιήσουμε την next και να προχωράμε εγγραφή - εγγραφή μέχρι να φτάσουμε στην τελευταία.

Μέσα στον κύκλο επεξεργασίας από την πρώτη ως την τελευταία εγγραφή χρησιμοποιούνται δύο μέθοδοι, η getInt και η getString. Καθεμιά από αυτές παίρνει μια ακέραιη παράμετρο που δηλώνει τον αριθμό στήλης του πίνακα του αποτελέσματος και θα επιστρέψει τα περιεχόμενα της τρέχουσας εγγραφής στην στήλη που έχει δηλωθεί. Ο τύπος δεδομένων που επιστρέφεται από την κλήση της μεθόδου είναι ο τύπος που δηλώνεται από το όνομα της μεθόδου.

Για παράδειγμα, η getString(1) θα επιστρέψει τα περιεχόμενα της πρώτης στήλης της τρέχουσας εγγραφής του ResultSet με τη μορφή string. Παρόμοια, η getInt(2) θα επιστρέψει τον ακέραιο που βρίσκεται στην δεύτερη (τελευταία) στήλη της τρέχουσας σειράς του ResultSet. Πρέπει να σημειώσουμε πως πρέπει να προσέχετε να ταιριάζουν οι συναρτήσεις που χρησιμοποιείτε με τους τύπους δεδομένων που υπάρχουν στη βάση δεδομένων. Επίσης εκτός από αυτές τις δυο υπάρχουν και άλλες αντίστοιχες μέθοδοι όπως οι getFloat, getDate κ.λ.π.

Τέλος, η σύνδεση και το result set κλείνουν και το πρόγραμμα τερματίζεται.

Ο παρακάτω κώδικας αποτελεί τμήμα ενός άλλου προγράμματος Java που ενημερώνει μια βάση δεδομένων με την εισαγωγή μίας σειράς:

```
String fileURL = "...";
String driverName = "...";
try
{
    //Load in the driver programmatically
    Class.forName(driverName);
}
catch (ClassNotFoundException cfn)
{
    //Problem with driver, error message and return
    //to operating system with status value 1
    System.out.println("Problem loading driver");
    System.exit(1);
}
```

```
try
{
    //Establish a connection to the database, second
    //argument is the name of the user and the third
    //argument is a password (blank)
    Connection con =
    DriverManager.getConnection(fileURL,"Darrel","");
    Statement updateStatement =
    con.createStatement();
    String employeeName = "...",
        employeeSalary = "...",
        employeeDepartment = "...";
    //Execute the SQL insert statement
    int noOfRows =
    updateStatement.executeUpdate
    ("INSERT INTO employees (name, salary, department)" +
    "VALUES (" + "'" + employeeName + "', '" +
    employeeSalary + "', '" + employeeDepartment + "')");
}

catch(SQLException sqe)
{
    System.out.println("Problem updating database");
}
```

Ο κώδικας αυτός μοιάζει πολύ μ' αυτόν που παρουσιάστηκε νωρίτερα. Η διαφορά τους έγκειται στον τρόπο εκτέλεσης της εισαγωγής. Πρώτα εκτελείται η μέθοδος `executeUpdate` του αντικειμένου `Statement`. Υπενθυμίζουμε ότι η `SELECT` εκτελέστηκε με τη μέθοδο `executeQuery`. Η μέθοδος `executeUpdate` θα επιστρέψει έναν ακέραιο αντί για ένα αντικείμενο `ResultSet`. Αυτός ο ακέραιος αντιπροσωπεύει τον αριθμό σειρών στους πίνακες της βάσης δεδομένων που επηρεάζονται από την εκτέλεση της SQL εντολής. Φυσικά, μια άλλη διαφορά τους είναι η χρήση διαφορετικής δηλώσεως SQL. Χρησιμοποιήθηκε η `INSERT INTO`. Στο παραπάνω παράδειγμα, η εκτέλεση του κώδικα Java θα είχε ως αποτέλεσμα να εκτελεστεί η ακόλουθη SQL εντολή:

```
INSERT INTO employees(name,salary,department)
VALUES(name of the employee, salary of the employee, department of the employee)
```

Αυτό θα εισάγει μία νέα σειρά / εγγραφή με τις τιμές που προσδιορίζονται από τα αλφαριθμητικά που θα υπάρχουν μέσα στα μονά εισαγωγικά.

Το τελευταίο παράδειγμα παρουσιάζει την χρήση των `commit` και `rollback` κατά την ενημέρωση μίας βάσης δεδομένων.

```
try

{

    //Establish a connection to the database, second
    //argument is the name of the user and the third
    //argument is a password (blank)
    Connection con =
        DriverManager.getConnection(fileURL,"Darrel","");
    //We have to program the commits, the statement below
    //means that commits are not automatically applied
    con.setAutoCommit(false);
    updateStatement = con.createStatement();
    //Carry out some update operation on the database
    //described by fileURL
    //...
    updateStatement.close();
    //If this point is reached then the update on
    //the database has been successful, commit by
    //sending the commit message to con.
    con.commit();
}
catch(SQLException sqe1)
{
    //There was a problem, for example no database was
    //found, insert recovery code or error code here
}
```

Η commit θα ενημερώσει την βάση δεδομένων ότι όλες οι πράξεις που έχουν γίνει στην βάση δεδομένων μετά το τελευταίο commit, θα πρέπει να πραγματοποιηθούν τώρα.

Στον παραπάνω κώδικα η βάση δεδομένων τοποθετείται σε κατάσταση χειροκίνησης δέσμευσης μεταβολών κατά την οποία ο προγραμματιστής πρέπει να εισάγει τον κώδικα που πράγματι κάνει τις αλλαγές μόνος του. Αν ο κώδικας στο μπλοκ try εκτελεστεί σωστά τότε εφαρμόζεται το commit που είναι η τελευταία εντολή του μπλοκ. Αν όμως δεν γίνει σωστά, τότε παράγεται μία SQLException και εκτελείται ο κώδικας του μπλοκ catch. Ποια θα είναι στην συνέχεια η επεξεργασία του προβλήματος εξαρτάται από την εφαρμογή.

6.2. Στιγμιότυπα ιστοσελίδας

Ο κώδικας της SQL όπως προαναφέραμε χρειάζεται ένα πιο οικείο περιβάλλον παρουσιάσεις για τους χρήστες. Γι' αυτό και το περιβάλλον της Java μπορεί με απλή μεθοδολογία να παρουσιάσει μια βάση δεδομένων SQL.

Παρακάτω παρουσιάζουμε μερικά screenshots από την δική μας βάση δεδομένων όπως παρουσιάζεται σε περιβάλλον της JAVA.

Αρχικά παρουσιάζουμε το περιβάλλον που παρουσιάζεται στο χρήστη μόλις φορτώσει τον κώδικα στο υπολογιστή του. Βλέπουμε ότι δίνει στο χρήστη επιλογές για τις μετέπειτα εργασίες που θα κάνει στο περιβάλλον αυτό. Επίσης του παρουσιάζονται και κάποιες μικρές οδηγίες βοηθητικές για τις επόμενες κινήσεις του.



Στη συνέχεια μεταβαίνει στο χώρο που μπορεί να κάνει τις παραγγελίες του ή να δει σε τι κατάσταση βρίσκονται τα αποθέματα στην αποθήκη του. Έτσι ώστε αν κάποιο προϊόν βρίσκεται σε έλλειμμα να δώσει παραγγελία για να συμπληρωθεί το απόθεμα της αποθήκης. Σε αυτό το σημείο ενημερώνεται και αν κάποιο προϊόν δεν υπάρχει καθόλου στην αποθήκη.

[Κεντρική](#)[Καταστήματα](#)[Διαχείριση](#)

SUPERMARKET

shop1

Προϊόν: product1 - Διαθέσιμα: 500

Ποσότητα:

Παραγγείλτε

Προϊόν: product2 - Διαθέσιμα: 500

Ποσότητα:

Παραγγείλτε

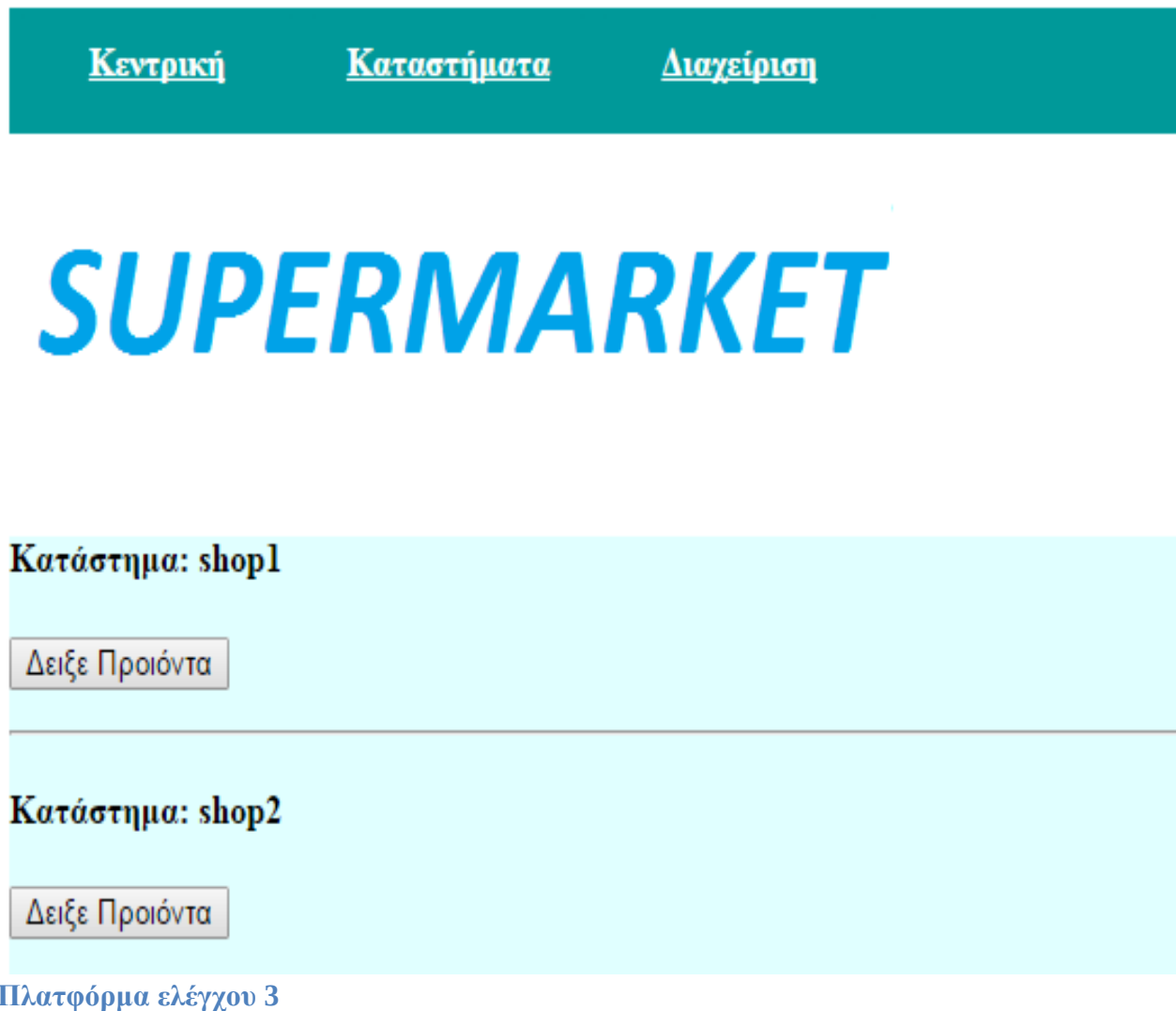
Προϊόν: product3 - Διαθέσιμα: 500

Ποσότητα:

Παραγγείλτε

Πλατφόρμα ελέγχου 2

Στη συγκεκριμένη πλατφόρμα ο χρήστης μπορεί να δει τα προϊόντα που βρίσκονται είτε στο κατάστημα 1 είτε στο κατάστημα 2. Και σε τι απόθεμα βρίσκονται. Απο όπου ενημερώνεται κάθε φορά όταν ένα προϊόν βρίσκεται στα τελευταία πέντε κομμάτια.



Πλατφόρμα ελέγχου 3

```
mysql> select * from shop_product;
```

product_name	shop_name	product_left	less_product_number
product1	shop1	100	5
product1	shop2	100	5
product2	shop1	100	5
product3	shop1	100	5

```
4 rows in set (0.00 sec)
```



```
mysql> select * from warehouse_product;
```

product_name	warehouse_name	product_quantity	less_product_number
product1	warehouse1	1000	5
product1	warehouse2	1000	5
product2	warehouse2	1000	5
product3	warehouse1	1000	5

```
4 rows in set (0.00 sec)
```

Βάσεις δεδομένων & Φίλτρα 1

Η επόμενη πλατφόρμα μας παρουσιάζει τα προϊόντα σε περιβάλλον SQL τους τύπους τους, τις τιμές τους και τα αποθέματα τους σε καταστήματα και σε αποθήκες.

```
mysql> select * from product
-> ;
```

pr_name	pr_type	pr_constructor	pr_price	pr_warehouse	pr_supplier
product1	type1	constructor1	10E	warehouse1	supplier1
product2	type1	constructor2	20E	warehouse2	supplier2
product3	type2	constructor2	10E	warehouse1	supplier2
product4	type2	constructor1	90E	warehouse2	supplier1

```
4 rows in set (0.00 sec)
```

```
mysql> select * from employee;
```

id_number	name	salary	address	phone_number	shop
AA30	employee1	1000E	employee1_adress	6900000011	shop1
AA31	employee2	3000E	employee2_adress	6900000022	shop2

```
2 rows in set (0.01 sec)
```

```
mysql> select * from shop;
```

shp_name	shp_address	shp_warehouse
shop1	shop1_address	warehouse1
shop2	shop2_address	warehouse2

```
2 rows in set (0.00 sec)
```

```
mysql> select * from ordering;
```

or_code	or_quantity	or_product	or_supplier	or_shop
111product1shop1	111	product1	supplier1	shop1
124product1shop1	124	product1	supplier1	shop1
213product1shop1	213	product1	supplier1	shop1
250product1shop1	250	product1	supplier1	shop1
300product1shop1	300	product1	supplier1	shop1
400product1shop1	400	product1	supplier1	shop1

```
6 rows in set (0.00 sec)
```

```
mysql> select * from supplier;
```

splr_name	splr_address	splr_phone_number
supplier1	supplier1_adress	6900000000
supplier2	supplier2_adress	6900000002

```
2 rows in set (0.01 sec)
```

```
mysql> select * from supplier_storage;
```

spst_supplier_name	spst_storage_name
supplier1	warehouse1
supplier2	warehouse2

```
2 rows in set (0.01 sec)
```

Βάσεις δεδομένων & Φίλτρα 2

```
mysql> select * from warehouse;
+-----+-----+
| wr_name | wr_adress |
+-----+-----+
| warehouse1 | warehouse1_adress |
| warehouse2 | warehouse2_adress |
+-----+-----+
2 rows in set (0.00 sec)
```

Βάσεις δεδομένων & Φίλτρα 3

```
mysql> show tables;
+-----+
| Tables_in_supermarket |
+-----+
| employee               |
| ordering                |
| product                |
| shop                   |
| shop_product           |
| supplier                |
| supplier_storage       |
| warehouse               |
| warehouse_product      |
+-----+
9 rows in set (0.00 sec)
```

Βάσεις δεδομένων & Φίλτρα 4

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Aho A., Sagiv Y. and Ullman J. D. (1979). Equivalence among Relational Expressions. *SIAM Journal of Computing*.
- Alsberg P. A. and Day J. D. (1976). A principle for resilient sharing of distributed resources. In *Proc. 2nd Int. Conf. Software Engineering, San Francisco, CA*, 562-570.
- American National Standards Institute (1975). ANSI/X3/SPARC Study Group on Data Base Management Systems. Interim Report, FDT. *ACM SIGMOD Bulletin*.
- Anahory S. and Murray D. (1997). *Data Warehousing in the Real World: A Practical Guide For Building Decisions Support Systems*.
- Astrahan M. M., Blasgen m. W., Chamberlin D. D., Eswaran K. P., Gray J. N., Griffith P. P., King W. F., Lorie R. A., McJones P. R., Mehl J. W., Putsolu G. R., Traiger I. L., Wade B. W., and Watson V. (1976). System R: Relational Approach to Database Management. *ACM Trans. Database Systems*.
- Atkinson M. P., and Buneman P. (1989). Type and persistence in database programming languages.
- Bancilhon F. and Buneman P. (1990). *Advances In Database Programming Languages*.
- Batini C., Cesi S. and Navathe S. (1992). *Conceptual Database Design: An Entity-Relationship Approach*.
- Boyce R., Chamberlin D., King W. and Hammer M. (1975). Specifying Queries In Relational Expressions.
- Cannan S. and Otten G. (1993). *SQL – The Standard Handbook*. Maidenhead: McGraw – Hill International.
- Chen P. P. (1976). The Entity-Relationship model – Towards a Unified View Of Data.
- Codd E. F. (1972b). Further normalization of the data base model. In *Data Base Systems* (Rustin R., ed.), Englewood Cliffs, NJ: Prentice-Hall
- Codd E. F. (1974). Recent Investigations in relational data base systems. In *Proc. IFIP Congress*.
- Codd E. F. (1979). Extending the data base relational model to capture more meaning. *ACM Trans. Database Systems*.
- Database Architecture Framework Task Group. (1986). Reference Model for DBMS standardization. *SIGMOD Record*.
- MySql Documentation Command Reference and Downloads: <http://www.mysql.org>. The Official Site.
- PHP Documentation, Command Reference and Downloads: <http://www.php.net>.