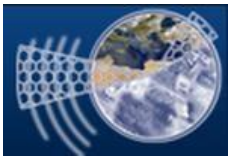
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής



Πτυχιακή Εργασία

Τίτλος: «Σύστημα διαχείρισης ξενοδοχείων»

Αλέξανδρος Φραγκάκης (ΑΜ:3183)
Εμμανουήλ Παρτσακουλάκης(ΑΜ:3226)
Γεώργιος Γλυκοκόκαλος (ΑΜ:3094)

Επιβλέπων Καθηγητής : Νικόλαος Παπαδάκης

ΗΡΑΚΛΕΙΟ
2016

Ευχαριστίες

Σε αυτό το σημείο θα θέλαμε να ευχαριστήσουμε τις οικογένειες μας για την υποστήριξη που μας παρείχαν όλα αυτά τα χρόνια σπουδών. Η βοήθεια τους ήταν απαραίτητη και συνάμα καθοριστική για την ολοκλήρωση των σπουδών μας.

Επιπλέον θα θέλαμε να ευχαριστήσουμε τους καθηγητές του τμήματος και ιδιαίτερα τον επιβλέπων, Κύριο Παπαδάκη Νικόλαο, για την πολύτιμη βοήθειά του καθόλη την διάρκεια διεκπεραίωσης της εργασίας.

Abstract

Our thesis on a comprehensive hotel management system. It provides the user the ability to seek the best and most economical room and wants to make online booking. All information is stored in the database and the manager on the other hand has full control of the content having the ability to add and remove information. Some of the features it has is to add a hotel room that has such and similar photos. Even the administrator can view statistics for every hotel, for example completeness. The environment is created for user friendly and easy to use, enriched with as many technologies as possible.

Σύνοψη

Η πτυχιακή μας εργασία αφορά ένα ολοκληρωμένο σύστημα διαχείρισης ξενοδοχείων. Παρέχει την δυνατότητα στον χρήστη να αναζητήσει το καλύτερο και οικονομικότερο δωμάτιο που επιθυμεί και να κάνει online κράτηση. Όλες οι πληροφορίες είναι αποθηκευμένες στην βάση δεδομένων και ο διαχειριστής από την άλλη μεριά έχει τον πλήρη έλεγχο του περιεχομένου έχοντας την δυνατότητα να προσθέσει και να αφαιρέσει πληροφορίες. Μερικές από τις δυνατότητες που έχει είναι να προσθέσει ένα ξενοδοχείο, τα δωμάτια που έχει όπως και τις ανάλογες φωτογραφίες. Ακόμα ο διαχειριστής μπορεί να βλέπει στατιστικά στοιχεία για κάθε ξενοδοχείο, όπως για παράδειγμα την πληρότητα. Το περιβάλλον που δημιουργήσαμε για τον χρήστη είναι φιλικό και εύκολο στην χρήση, εμπλουτισμένο με όσο το δυνατόν περισσότερες τεχνολογίες.

Πίνακας περιεχομένων

Πίνακας Εικόνων

1 Εισαγωγή

1.1 Περίληψη

Η εργασία αυτή αποσκοπεί στην μελέτη και ανάλυση των τεχνολογιών για την δημιουργία ενός σύγχρονου διαδικτυακού συστήματος διαχείρισης ξενοδοχείων. Πρώτα θα αναφερθούμε σε ένα από τα κύρια μέρη αυτής της εφαρμογής, τον τρόπο διαχείρισης της βάσης δεδομένων που χρησιμοποιήσαμε.

Συγκεκριμένα χρησιμοποιήσαμε την αρχιτεκτονική MVC, που σημαίνει Model-View-Controller.Στη συνέχεια σας παρουσιάσουμε τις τεχνολογίες-γλώσσες προγραμματισμού διαδικτύου που χρησιμοποιήσουμε για να εκπληρώσουμε την εργασία αυτή.

1.2 Κίνητρο για τη διεξαγωγή της εργασίας

Ο τουρισμός αποτελεί μια πολύ μεγάλη βιομηχανία και σημαντικότατη πηγή εσόδων για την χώρα μας. Ο τουριστικός τομέας απασχολεί διαφορετικές κατηγορίες επαγγελμάτων με σημαντικότερο κλάδο αυτό της πληροφορικής. Η συνεχής ανάπτυξη τεχνολογιών έχει βοηθήσει τον τουρισμό σε πολλούς τομείς. Με τα σύγχρονα μέσα κοινωνικής δικτύωσης προβάλλονται τουριστικοί προορισμοί δίνοντας έτσι την ευκαιρία στον άνθρωπο να γνωρίσει ακόμα πιο πολλά μέρη. Με την χρήση του διαδικτύου να γίνεται όλο και πιο απλή, ο καθένας έχει πλέον πρόσβαση σε τουριστικές επιχειρήσεις για να επιλέξει το πακέτο που του ταιριάζει.

Με τον ανταγωνισμό που υπάρχει ανάμεσα στις τουριστικές επιχειρήσεις, η ποιότητα των υπηρεσιών γίνεται καλύτερη. Οι ολοκληρωμένες διαδικτυακές εφαρμογές που συνεχώς αυξάνονται είναι ικανές να εξυπηρετούν όλους τους χρήστες σε πολύ λίγο χρόνο. Ο χρήστης πλέον μπορεί να κάνει κράτηση για αεροπορικά εισιτήρια, δωμάτια ξενοδοχείων, ενοικίαση αυτοκινήτου ακόμα και για τραπέζι σε ένα εστιατόριο. Η προσέγγιση των προβλημάτων, ο τρόπος σχεδίασης και υλοποίησης και οι απαιτήσεις που έχει ο χρήστης να αποτελούν τα βασικά κίνητρα αυτής της εργασίας.

1.3 Σκοπός και στόχοι της εργασίας

Η συγκεκριμένη εργασία έχει ως σκοπό την ανάπτυξη ενός σύγχρονου συστήματος διαχείρισης ξενοδοχείων. Τα δεδομένα και οι πληροφορίες του συστήματος είναι αποθηκευμένες σε μια βάση δεδομένων PostgreSQL, όπου αντλούνται για να εμφανιστούν στον χρήστη. Βέβαια χρησιμοποιήθηκαν πολλές ακόμα τεχνολογίες όπως για παράδειγμα οι JSP, HTML, Javascript, AJAX και JSON. Σκοπός μας είναι η προσαρμογή όλων αυτών για την ανάπτυξη της εφαρμογής και προπάντων η εκμάθηση των παραπάνω τεχνολογιών.

1.4 Δομή της εργασίας

Στο πρώτο κεφάλαιο βρίσκεται το περιεχόμενο, το κίνητρο, ο σκοπός και οι στόχοι της εργασίας αυτής. Στο δεύτερο κεφάλαιο γίνεται μια σύντομη αναφορά στην αρχιτεκτονική την οποία υλοποιήσαμε και τις τεχνολογίες που χρησιμοποιήσαμε. Το τρίτο κεφάλαιο ασχολείται με την διαχείριση των βάσεων δεδομένων. Στο τέταρτο κεφάλαιο αναλύουμε τον σχεδιασμό μιας βάσης δεδομένων και δίνουμε μια διαγραμματική αναπαράσταση συσχετίσεων οντοτήτων. Στο πέμπτο και κύριο κεφάλαιο αναλύουμε την εφαρμογή με ανάλυση του κώδικα. Στο τελευταίο κεφάλαιο υπάρχουν τα συμπεράσματα και οι μελλοντικές επεκτάσεις της εργασίας αυτής.

2 Μεθοδολογία Υλοποίησης

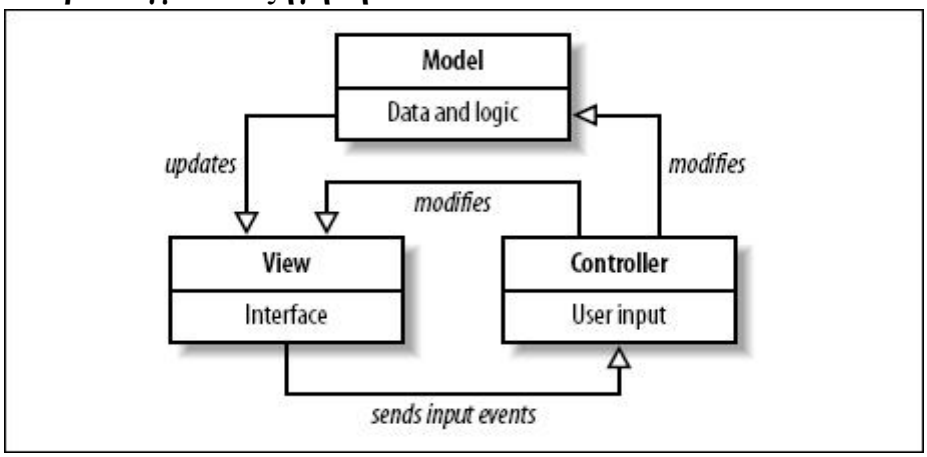
2.1 Αρχιτεκτονική MVC

Η εργασία είναι χτισμένη πάνω στο μοντέλο Model-View-Controller (**MVC**).Είναι μια αρχιτεκτονική η οποία χρησιμοποιείται για την ανάπτυξη εφαρμογών webμε διαφορετικό τρόπο από ό,τι η παραδοσιακή ανάπτυξη ASP.NETweb. Θα μπορούσαμε να πούμε ότι το MVC είναι ένα είδος αρχιτεκτονικής, είναι δηλαδή ένας συγκεκριμένος τρόπος με τον οποίο «χτίζουμε» ένα application.Είναι μια από τα πιο συχνά χρησιμοποιούμενα πρότυπα ανάπτυξης ιστοσελίδων [1].

2.2Τι σημαίνουν τα αρχικά του MVC

- **Model**
  Στο μοντέλο τοποθετούμε τις λειτουργίες της εφαρμογής που σχετίζονται με την πρόσβαση στη βάση δεδομένων. Συνήθως οι κλάσεις του μοντέλου εμπεριέχουν συναρτήσεις που χρησιμοποιούνται για ανάκτηση, εισαγωγή ή ανανέωση των δεδομένων στη βάση δεδομένων [1].
- **View**
  Είναι η πληροφορία που εμφανίζεται στο χρήστη. Είναι συνήθως μια ιστοσελίδα, αυτό που βλέπουμε δηλαδή. Τις περισσότερες φορές μία View μιλάει με ένα controller και αφού ο controller κάνει τις διάφορες επεξεργασίες των δεδομένων στέλνει στη View συγκεκριμένα δεδομένα να εμφανίσει [1].
- **Controller**
  Ο controller λειτουργεί ως διαμεσολαβητής μεταξύ Model και της View. Ελέγχει το πώς «τρέχει» η εφαρμογή. Μιλάει με το Model, παίρνει τα δεδομένα που ζητά και εν συνεχεία και αφού τα επεξεργαστεί τα στέλνει πίσω στη View για να τα απεικόνιση [1].

Ένα παράδειγμα-επεξήγηση



Εικόνα 1 - Σχεδίαση μοντέλου MVC

Έχουμε τον χρήστη ο οποίος δίνει κάποια στοιχεία του σε ένα web-application. Για παράδειγμα συμπληρώνει μία φόρμα με το ονομά του και πατάει το κουμπί εγγραφή. Στη συνέχεια ο ελεκτής ο controller δηλαδή αφού έχει λάβει τα στοιχεία που έχει δώσει ο χρήστης μιλάει με το μοντέλο χρησιμοποιώντας τα στοιχεία του χρήστη σαν μεταβλητή και ζητάει δεδομένα από το μοντέλο τα οποία όταν τα λαμβάνει τα προσαρμόζει ανάλογα με αυτό που ζήτησε ο χρήστης. Έπειτα ο controller με τα νεα δεδομένα που έχει αλλάζει την view [1].

2.3 Βασικά Πλεονεκτήματα MVC

- **Διαχωρισμός Προβλημάτων**

  ¨Ένα απο τα βασικότερα πλεονέκτηματα του MVC. Διασπάται η εφαρμογή σε τρία επίπεδα, το επίπεδο των models,το επίπεδο των controllers και το επίπεδο των views και το κάθε επίπεδο κάνει διαφορετικά πράγματα και ταυτόχρονα συνεργάζεται με τα άλλα επίπεδα. Μία εφαρμογή για να που θεωρείται σωστά δομημένη είναι εκείνη που τα τρία επίπεδα είναι ξεκάθαρα καθορισμένα και δεν συμπλέκονται. Για παράδειγμα θα ήταν λάθος αν στο επίπεδο των View υπάρχει κώδικας που συνδέεται με την βάση δεδομένων και «τραβάει» δεδομένα [1].

- **Επεκτασιμότητα**

  Ένα άλλο πλεονέκτημα της MVC αρχιτεκτονικής είναι η «Επεκτασιμότητα». Μας δίνεται η δυνατότητα σε μία εφαρμογή να μπορούμε μελλοντικά να προσθέσουμε λειτουργίες σε αυτή ή να αλλάξουμε κάποιες από τις ήδη υπάρχουσες λειτουργίες και να έχουμε άλλα αποτελέσματα. Ένα παράδειγμα, η πλατφόρμα WordPress. Είναι επεκτάσιμη με τη χρήση των διαφόρων plug-ins διότι προσθέτουμε στις ήδη υπάρχουσες λειτουργίες και άλλες λειτουργίες. Τα προγράμματα που είναι φτιαγμένα με MVC αρχιτεκτονική έχουν βασικό χαρακτηριστικό ότι είναι επεκτάσιμα [1].

- **Ελεγξιμότητα**

  Αυτό το χαρακτηριστικό είναι ένα πολύ κρίσιμο. Οι MVC εφαρμογές έχουν την δυνατότητα να είναι ελέγξιμες και με τον τρόπο αυτό συντηρούνται πιο εύκολα. Έστω ότι έχουμε μία εφαρμογή η οποία διαθέτει μία λειτουργία login, δηλαδή ζητά από τον χρήστη να πληκτρολογήσει κάποια στοιχεία σε μία φόρμα και ενα συνέχεια τον εισάγει μέσα στο σύστημα. Αυτή τη λειτουργία την ελέγχει κάποιος login-Controller ο οποίος περιέχει κώδικα που διαχειρίζεται τα δεδομένα αυτά που εισήχθησαν από τον χρήστη. Αυτός ο controller θεωρείται μία «μονάδα». Στα MVC frameworks μπορούμε με πολλή ευκολία να γράψουμε απλό κώδικα με τον οποίο τεστάρουμε αυτόν τον controller αλλά και κάθε μία από τις λειτουργίες του. Παίρνουμε τα αποτελέσματα και βλέπουμε η συγκεκριμένη μονάδα της εφαρμογής μας λειτουργεί σωστά [1].

2.4 Οντότητες-Entities στην MVC Αρχιτεκτονική

Οι οντότητες ή αλλιώς «entities» εμφανίζονται συχνά στον προγραμματισμό. Ο όρος «οντότητα» προκύπτει κατά την μοντελοποίηση ενός συστήματος, έτσι ο developer έχει την ικανότητα να δει τις οντότητες από τις οποίες αποτελείται η εφαρμογή του. Για παράδειγμα στην εφαρμογή που υλοποιήσαμε με ξενοδοχεία, δωμάτια χρήστες οι οντότητες είναι ο χρήστης, το δωμάτιο, το ξενοδοχείο κλπ. Σε αυτά τα πράγματα που υπάρχουν δηλαδή, βασίζουμε τις λειτουργίες της εφαρμογής μας. Τις περισσότερες φορές με τον όρο «entity» αναφέρουμε και ένα πίνακα από την βάση δεδομένων. Με την δημιουργία οντοτήτων ολοκληρώνουμε την εφαρμογή μας με μεγαλύτερη ευκολία και σιγουριά [1].

2.5 MVC frameworks

Χρησιμοποιώντας κάποιο framework μπορούμε να δημιουργήσουμε μια εφαρμογή σε αρχιτεκτονική MVC, πιο γρήγορα και με μεγαλύτερη ασφάλεια επειδή σε ένα framework υπάρχουν αρκετές προκαθορισμένες λειτουργίες, έτσι εξοικονομούμε γράψιμο και χρόνο φυσικά. Στην Java ένα από τα καλύτερα Javaframeworks για αρχάρειους είναι το Springframework.Υπάρχουν και άλλα βέβαια όπως το Playframework, το Grailsframework, το Strutsframework και το Jsf frameworkκα [1].

2.6 Προγράμματα ανάπτυξης

Η παρούσα πτυχιακή εργασία υλοποιήθηκε σε γλώσσα προγραμματισμού Java. Το πρόγραμμα που χρησιμοποιήσαμε για την ανάπτυξης της εφαρμογής ήταν το NetBeans [3]. Είναι ένα ενσωματωμένο περιβάλλον ανάπτυξης εφαρμογών (Integrated Development Environment - IDE). Στην αρχή χρησιμοποιήθηκε για την ανάπτυξη εφαρμογών Java αλλά πλέον χρησιμοποιείται και για άλλες γλώσσες όπως PHP, C/C++, και HTML5. Με τις τελευταίες εκδόσεις έχει εξελιχθεί σε ένα ισχυρό πρόγραμμα επεξεργασίας για web scripting. Για την δημιουργία της βάσης χρησιμοποιήσαμε το πρόγραμμα pgAdmin [4].

2.7 Τεχνολογίες που χρησιμοποιήθηκαν

2.7.1 PostgreSQL

Η PostgreSQL αποτελεί μια σχεσιακή βάση δεδομένων ανοικτού κώδικα με πολλές δυνατότητες. Η ανάπτυξη της διαρκεί ήδη πάνω από δύο δεκαετίες και βασίζεται σε μια αποδεδειγμένα καλή αρχιτεκτονική η οποία έχει δημιουργήσει μια ισχυρή αντίληψη των χρηστών της γύρω από την αξιοπιστία, την ακεραιότητα δεδομένων και την ορθή λειτουργία [18].

Η PostgreSQL τρέχει σε όλα τα βασικά λειτουργικά συστήματα, στα οποία περιλαμβάνονται το Linux, το MACOSX, τοSolaris τα Windows και άλλα πολλά. Είναι συμβατή με ACID και συμπεριλαμβάνει τους περισσότερους SQL92 και SQL99 τύπους δεδομένων συμπεριλαμβανομένων INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL και TIMESTAMP. Επίσης υποστηρίζει αποθήκευση μεγάλων δυαδικών αντικειμένων (binary), όπως εικόνες, ήχοι ή βίντεο. Διαθέτει επίσης περιβάλλοντα προγραμματισμού για τις γλώσσες προγραμματισμού C,C++,Java,Perl,Python,Ruby,Tcl και υποστήριξη για την πλατφόρμα .NETκαι το πρότυπο ODBC.

Μια βάση δεδομένων, PostgreSQL υλοποιεί εξελιγμένα χαρακτηριστικά, όπως Multi-Version Concurrency Control (MVCC), point in time recovery, tablespaces, asynchronous replication, nested transactions (savepoints), online/hot backups, a sophisticated query planner/optimizer, write ahead logging for fault tolerance. Υποστηρίζει διεθνή σετ χαρακτήρων, κωδικοποιήση χαρακτήρων σε πολλάbyte, Unicode, καθώς και δυνατότητα ταξινόμισης δεδομένων ανεξάρτητα απο το locale. Μπορεί να διαχειριστεί εύκολα μεγάλους αριθμούς ταυτόχρονων χρηστών καθώς και μεγάλο όγκο δεδομένων. Υπάρχουν ενεργά συστήματα PostgreSQL σε περιβάλλοντα παραγωγής που διαχειρίζονται πάνω από 4 terabytes δεδομένων.

2.7.2 Java

Η **Java** είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού που σχεδιάστηκε από την εταιρεία πληροφορικής *Sun Microsystems*. Ένα από τα βασικά πλεονεκτήματα της Java έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας [11]. Τα προγράμματα που είναι γραμμένα σε *Java* τρέχουνε ακριβώς το ίδιο σε Windows, Linux, Unix και Macintosh χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή

να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα. Για να επιτευχθεί όμως αυτό χρειαζόταν κάποιος τρόπος έτσι ώστε τα προγράμματα γραμμένα σε Java να μπορούν να είναι «κατανοητά» από κάθε υπολογιστή ανεξάρτητα του είδους επεξεργαστή (Intel x86, IBM, Sun SPARC, Motorola) αλλά και λειτουργικού συστήματος (Windows, Unix, Linux, BSD, MacOS). Ο λόγος είναι ότι κάθε κεντρική μονάδα επεξεργασίας κατανοεί διαφορετικό κώδικα μηχανής. Ο συμβολικός κώδικας (*assembly*) που μεταφράζεται και εκτελείται σε Windows είναι διαφορετικός από αυτόν που μεταφράζεται και εκτελείται σε έναν υπολογιστή Macintosh. Η λύση δόθηκε με την ανάπτυξη της *Εικονικής Μηχανής* (*Virtual Machine* ή VM ή ΕΜ στα ελληνικά).

Ακόμα μία ιδέα που βρίσκεται πίσω από τη *Java* είναι η ύπαρξη του συλλέκτη απορριμμάτων (*Garbage Collector*). Συλλογή απορριμμάτων είναι μία κοινή ονομασία που χρησιμοποιείται στον τομέα της πληροφορικής για να δηλώσει την ελευθέρωση τμημάτων μνήμης από δεδομένα που δε χρειάζονται και δε χρησιμοποιούνται άλλο. Αυτή η απελευθέρωση μνήμης στη Java είναι αυτόματη και γίνεται μέσω του συλλέκτη απορριμμάτων. Υπεύθυνη για αυτό είναι και πάλι η εικονική μηχανή η οποία μόλις «καταλάβει» ότι ο σωρός (heap) της μνήμης (στη Java η συντριπτική πλειοψηφία των αντικειμένων αποθηκεύονται στο σωρό σε αντίθεση με τη C++ όπου αποθηκεύονται κυρίως στη στοίβα) κοντεύει να γεμίσει ενεργοποιεί το συλλέκτη απορριμμάτων. Έτσι ο προγραμματιστής δε χρειάζεται να ανησυχεί για το πότε και αν θα ελευθερώσει ένα συγκεκριμένο τμήμα της μνήμης, ούτε και για σφάλματα δεικτών. Αυτό είναι ιδιαίτερα σημαντικό γιατί είναι κοινά τα σφάλματα προγραμμάτων που οφείλονται σε λανθασμένο χειρισμό της μνήμης.

Όλα τα εργαλεία που χρειάζεται κάποιος για να γράψει Java προγράμματα έρχονται δωρεάν, από το περιβάλλον ανάπτυξης μέχρι εργαλεία *build* όπως το Apache Ant και βιβλιοθήκες, ενώ υπάρχουν πολλές διαφορετικές υλοποιήσεις της *Εικονικής Μηχανής* και του *μεταγλωττιστή* (πχ the GNU Compiler for Java) της *Java*. Για να γράψει κάποιος κώδικα Java δε χρειάζεται τίποτα άλλο παρά έναν επεξεργαστή κειμένου, όπως το Σημειωματάριο (Notepad) των Windows ή ο vi (γνωστός στο χώρο του Unix). Παρ' όλα αυτά, ένα ολοκληρωμένο περιβάλλον ανάπτυξης (*IDE*) βοηθάει πολύ, ιδιαίτερα στον εντοπισμό σφαλμάτων (debugging). Υπάρχουν αρκετά διαθέσιμα, κάποια από αυτά είναι το NetBeans,Eclipse,jEdit,jCreatorκαι jBuilder ενώ κάποια από αυτά έρχονται δωρεάν.

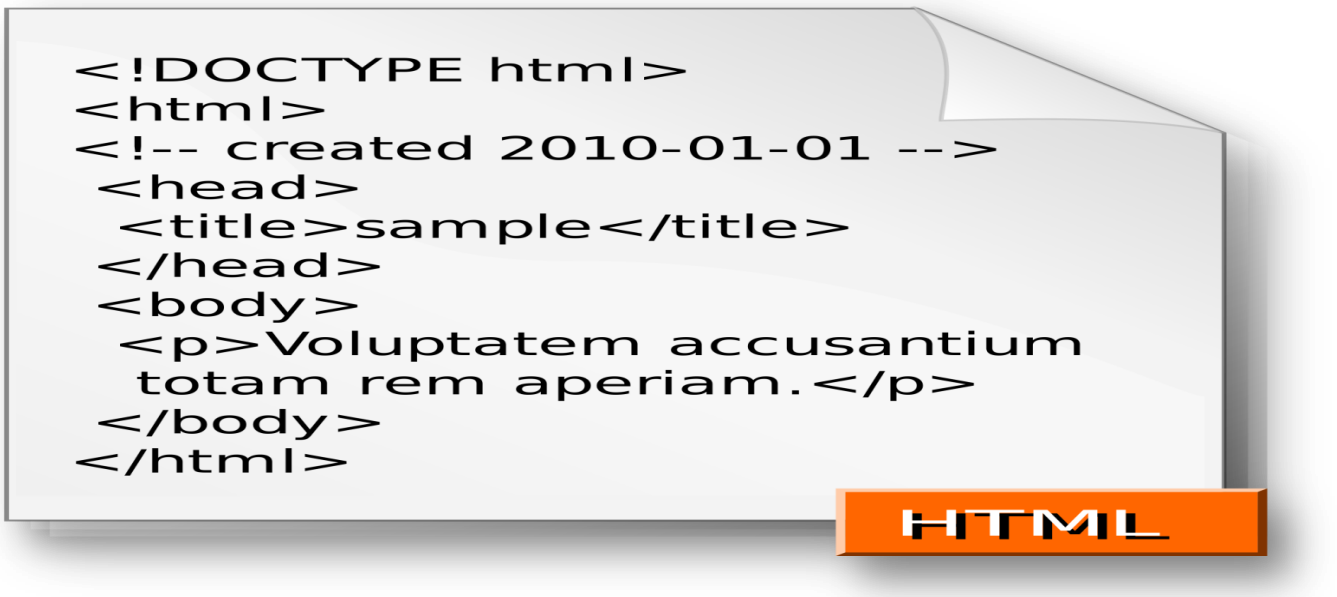
2.7.3 HTML5 και CSS

Η HTML (αρχικοποίηση του αγγλικού **H**yper**T**ext **M**arkup **L**anguage, είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων [8].

Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από *ετικέτες* (tags), οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα `<html>`), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα `<h1>` και `</h1>`), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης (ή σε άλλες περιπτώσεις ετικέτα ανοίγματος και ετικέτα κλεισίματος αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ. Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και τα συνθέτει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να ερμηνεύσει το περιεχόμενο της σελίδας [8].

Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML [8].

Οι Web browsers μπορούν επίσης να αναφέρονται σε στυλ μορφοποίησης CSS για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού. Ο οργανισμός W3C, ο οποίος δημιουργεί και συντηρεί τα πρότυπα για την HTML και τα CSS, ενθαρρύνει τη χρήση των CSS αντί διαφόρων στοιχείων της HTML για σκοπούς παρουσίασης του περιεχομένου [8].



Εικόνα 2 - Παράδειγμα html

2.7.4 HTML5

Η **HTML5** είναι μια υπό ανάπτυξη γλώσσα σήμανσης για τον Παγκόσμιο Ιστό που όταν ετοιμαστεί θα είναι η επόμενη μεγάλη έκδοση της HTML (Γλώσσα Υπερκειμένου, HyperText Markup Language). Η ομάδα Web Hypertext Application Technology Working Group (WHATWG) άρχισε δουλειά σε αυτή την έκδοση τον Ιούνιο του 2004 με το όνομα Web Applications 1.0. Το Φεβρουάριο του 2010 το πρότυπο ήταν ακόμη σε κατάσταση "Last Call" στο WHATWG [9].

Η HTML5 προορίζεται για αντικατάσταση της HTML 4.01, της XHTML 1.0, και της DOM Level 2 HTML. Ο σκοπός είναι η μείωση της ανάγκης για ιδιόκτητα plug-in και πλούσιες διαδικτυακές εφαρμογές (RIA) όπως το Adobe Flash, το Microsoft Silverlight, το Apache Pivot, και η Sun JavaFX.

Οι ιδέες πίσω από την HTML5 εμφανίστηκαν αρχικά το 2004 από την ομάδα WHATWG. Η HTML5 εμπεριέχει το πρότυπο *Web Forms 2.0* που είναι επίσης της WHATWG.

Το πρότυπο HTML5 υιοθετήθηκε ως αρχικό βήμα για τις εργασίες της νέας ομάδας εργασίας HTML του W3C το 2007. Αυτή η ομάδα εργασίας δημοσίευσε το Πρώτο Δημόσιο Working Draft του προτύπου στις 22 Ιανουαρίου 2008. Το πρότυπο είναι ακόμη υπό ανάπτυξη, και αναμένεται να παραμείνει έτσι για πολλά χρόνια, παρόλο που μέρη της HTML5 θα τελειώσουν και θα υποστηριχτούν από περιηγητές πριν το όλο πρότυπο φτάσει στη τελική κατάσταση Recommendation. Οι συντάκτες της HTML5 είναι ο Ίαν Χίκσον της εταιρίας Google και ο Ντέιβ Χίατ της εταιρίας Apple

2.7.5 Css

Επικαλυπτόμενα φύλλα στυλ (CSS) είναι μια γλώσσα που χρησιμοποιείται για την περιγραφή της παρουσίασης ενός εγγράφου γραμμένο σε μια γλώσσα σήμανσης. Παρά το γεγονός ότι χρησιμοποιείται πιο συχνά για να ρυθμίσετε το οπτικό στυλ των ιστοσελίδων και των διεπαφών χρηστών γραμμένο σε HTML και XHTML, η γλώσσα μπορεί να εφαρμοστεί σε οποιοδήποτε έγγραφο XML, συμπεριλαμβανομένων των πεδιάδα XML, SVG και XUL, και ισχύει για απόδοση στην ομιλία, ή σε άλλα μέσα. Μαζί με την HTML και JavaScript, η CSS αποτελεί τον ακρογωνιαίο λίθο της τεχνολογίας που χρησιμοποιείται από περισσότερες ιστοσελίδες για να δημιουργήσουν οπτικά ελκυστικό ιστοσελίδες, διεπαφών χρήστη για εφαρμογές web, και διεπαφές χρήστη για πολλές κινητές εφαρμογές [12].

Η CSS έχει σχεδιαστεί κυρίως για να επιτρέψει το διαχωρισμό του περιεχομένου εγγράφου από την παρουσίαση του εγγράφου, συμπεριλαμβανομένων πτυχών όπως η διάταξη, τα χρώματα και τις γραμματοσειρές. Αυτός ο διαχωρισμός μπορεί να βελτιώσει την προσβασιμότητα του περιεχομένου, παρέχοντας μεγαλύτερη ευελιξία και έλεγχο στις προδιαγραφές των χαρακτηριστικών παρουσίασης, επιτρέποντας πολλαπλές σελίδες HTML να μοιραστούν τη μορφοποίηση, καθορίζοντας τη σχετική CSS σε ένα ξεχωριστό αρχείο .css ώστε να μειώσουν την πολυπλοκότητα και την επανάληπτικότητα στο δομικό περιεχόμενο [12].

Αυτός ο διαχωρισμός της μορφοποίησης και του περιεχομένου καθιστά δυνατό να παρουσιάσει την ίδια σελίδα σήμανσης σε διαφορετικά στυλ για τις διάφορες μεθόδους απόδοσης, όπως επί της οθόνης, σε έντυπη μορφή, με τη φωνή (όταν διαβάζονται από ένα πρόγραμμα περιήγησης ή οθόνη αναγνώστη ομιλία-based) και σε Braille με βάση, αφής συσκευές. Μπορεί επίσης να χρησιμοποιηθεί για να εμφανίσει την ιστοσελίδα με διαφορετικό τρόπο, ανάλογα με το μέγεθος της οθόνης ή της συσκευής στην οποία προβάλλεται. Οι αναγνώστες μπορούν επίσης να καθορίσουν ένα διαφορετικό φύλλο (cascading style sheet) , όπως ένα αρχείο CSS που αποθηκεύεται στον υπολογιστή τους, για να παρακάμψει το φύλλο που έχει καθορίσει ο συγγραφέας [12].

Αλλαγές στο graphic-design ενός εγγράφου (ή εκατοντάδες έγγραφα) μπορούν να εφαρμοστούν γρήγορα και εύκολα, με επεξεργασία λίγων γραμμών στο αρχείο CSS που χρησιμοποιούν, και όχι από την αλλαγή σήμανσης στα έγγραφα [12].

Εικόνα 3 - Παράδειγμα css

2.7.6 Javascript

Η **JavaScript** (**JS**) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται [13]

Είναι μια γλώσσα σεναρίων που βασίζεται στα πρότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Σε ότι αφορά την σύνταξή της είναι επηρεασμένη σε μεγάλο βαθμό απο τη C. Αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java, αλλά γενικά αυτές οι δύο γλώσσες δε σχετίζονται και έχουν διαφορετική σημασιολογία. Οι βασικές αρχές σχεδιασμού της JavaScript προέρχονται από τις γλώσσες προγραμματισμού Self και Scheme. Βασίζεται σε διαφορετικά σε διαφορετικά προγραμματιστικά παραδείγματα, υποστηρίζοντας αντικειμενοστρεφές, προστακτικό και συναρτησιακό στύλ προγραμματισμού.

Χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων — τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers) και οι μικρές εφαρμογές της επιφάνειας εργασίας (desktop widgets). Οι νεότερες εικονικές μηχανές και πλαίσια ανάπτυξης για JavaScript (όπως το Node.js) έχουν επίσης κάνει τη JavaScript πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού στην πλευρά του διακομιστή (server-side).

```
<html>
    <body>
        <script type="text/javascript">
            var d = new Date()
            var time = d.getHours()
            if (time < 10) {
                document.write("<b> Καλημέρα </b>")
            }
            else {
                document.write("<b> Καλό Μεσημέρι </b>")
            }
        </script>
        <p> Αυτό το παράδειγμα δείχνει την εντολή If ... Else. </p>
        <p> Αν η ώρα στον φυλλομετρητή είναι πριν τις 10, θα λάβουμε τον χαιρετισμό "Καλημέρα", αλλιώς τον χαιρετισμό
        "Καλό Μεσημέρι". </p>
    </body>
</html>
```

Εικόνα 4 - Παράδειγμα JavaScript

2.7.7 Ajax

Ajax είναι ένα σύνολο τεχνικών ανάπτυξης Ιστού χρησιμοποιώντας πολλές τεχνολογίες web στην πλευρά του πελάτη για τη δημιουργία ασύγχρονης εφαρμογής Web. Με Ajax,οι διαδικτυακές εφαρμογές μπορούν να στείλουν και να ανακτήσουν δεδομένα από ένα διακομιστή ασύγχρονα (στο παρασκήνιο) χωρίς να παρεμβαίνει με την οθόνη και τη συμπεριφορά της υπάρχουσας σελίδας. Με την αποσύνδεση του στρώματος ανταλλαγής δεδομένων από το στρώμα παρουσίασης, οAjax

επιτρέπει στις ιστοσελίδες, καθώς και στις εφαρμογές web, να αλλάξει το περιεχόμενο δυναμικά, χωρίς την ανάγκη να φορτώσετε εκ νέου ολόκληρη τη σελίδα. Παρά το όνομα, η χρήση της XML δεν απαιτείται (JSON χρησιμοποιείται συχνά στην παραλλαγή ajaj), και τα αιτήματα δεν χρειάζεται να είναι ασύγχρονα[14].

Δεν είναι μια τεχνολογία, αλλά μια ομάδα τεχνολογιών. Η HTML και CSS μπορεί να χρησιμοποιηθούν σε συνδυασμό με τη σήμανση και τις πληροφορίες στυλ. Το DOM είναι προσβάσιμο με JavaScript για να εμφανιστεί δυναμικά και επιτρέπει στο χρήστη να αλληλεπιδράσει 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.

2.7.8 JSON

JSON (JavaScriptNotationObject) είναι μια ελαφριά μορφή δεδομένων-ανταλλαγής. Είναι εύκολο για τους ανθρώπους να διαβάζουν και να γράφουν. Είναι εύκολο για τις μηχανές να αναλύσουν και να δημιουργήσουν. Βασίζεται σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript, πρότυπο ECMA-262 3rd Edition - Δεκεμβρίου 1999. Στην ουσία είναι μια μορφή κειμένου που είναι εντελώς ανεξάρτητη γλώσσα, αλλά χρησιμοποιεί συμβάσεις που είναι εξοικειωμένες με τους προγραμματιστές της C-οικογένεια γλωσσών, συμπεριλαμβανομένων των C , C ++, C #, Java, JavaScript, Perl, Python, και πολλοί άλλοι. Αυτές οι ιδιότητες καθιστούν την JSON μια ιδανική γλώσσα δεδομένων ανταλλαγής [15].

Η JSON είναι χτισμένη σε δύο δομές:

- Μια συλλογή από ζεύγη ονόματος / τιμής. Σε διάφορες γλώσσες, αυτό υλοποιείται ως ένα αντικείμενο, ρεκόρ, struct, λεξικό, πίνακα hash, πληκτρολογηθεί λίστα ή associative array.
- Μια ταξινομημένη λίστα τιμών. Στις περισσότερες γλώσσες, αυτή πραγματοποιείται ως μια σειρά, διάνυσμα, λίστα, ή ακολουθία.

Αυτές είναι καθολικές δομές δεδομένων. Σχεδόν όλες οι σύγχρονες γλώσσες προγραμματισμού τους στηρίξει σε μία ή την άλλη μορφή. Είναι λογικό ότι μια μορφή δεδομένων που είναι ανταλλάξιμο με γλώσσες προγραμματισμού επίσης να βασίζονται σε αυτές τις δομές.

```
function roomsearch()
{
    var data = $("#superform").serialize();
    //alert(data);
    $.ajax({
        type: 'POST',
        data: data,
        dataType: "json",
        url: 'RoomSearchServlet',
        success: function (result) {

            drawTable(result);

        }
    });
}
```

Εικόνα 5 - Παράδειγμα JSON

2.7.9 jQuery

Η **jQuery** είναι μια βιβλιοθήκη JavaScript σχεδιασμένη να απλοποιήσει την υλοποίηση σεναρίων (scripting) στη πλευρά του πελάτη (client-side) της HTML και υποστηρίζει πολλαπλούς φυλλομετρητές Ιστού. Κυκλοφόρησε τον Ιανουάριο του 2006 από τον Τζον Ρέριγκ (John Resig). Χρησιμοποιείται σε πάνω από το 65% των 10.000 ιστοτόπων με τη μεγαλύτερη επισκεψιμότητα [16].

Κάποια από τα χαρακτηριστικά της είναι τα παρακάτω :

- DOM element επιλογές χρησιμοποιώντας την ανοιχτού κώδικα μηχανή επιλογής πολλαπλών φυλλομετρητών *Sizzle*.
- DOM διάσχιση και τροποποίηση (υποστηρίζοντας CSS 1-3)
- χειρισμός DOM βασισμένος σε CSS επιλογείς που χρησιμοποιεί τα id και class σαν κριτήρια για να κατασκευάσει επιλογείς.
- Events
- Εφέ και κινητά στοιχεία
- AJAX
- Επεκτασιμότητα μέσω plug-ins
- Εργαλεία όπως πληροφορίες user-agent, ανίχνευση χαρακτιριστικώ.
- Μεθόδους συμβατότητας που είναι εγγενώς διαθέσιμα σε σύγχρονα προγράμματα περιήγησης.
- Υποστήριξη πολλαπλών φυλλομετρητών.

Η jQuery είναι ένα αρχείο JavaScript, που περιέχει όλες τις λειτουργίες. Μπορεί να συμπεριληφθεί σε μια ιστοσελίδα παρέχοντας το αρχείο τοπικά

```
<script type="text/javascript" src="jquery.js"></script>
```

ή έχοντας ένα σύνδεσμο σε ένα από τους πολλούς διακομιστές που τη φιλοξενούν.

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
```

2.7.10 Bootstrap

Το Bootstrap είναι ένα web design framework. Μας δίνει την δυνατότητα να δημιουργήσουμε μια web εφαρμογή γρήγορα και χωρίς να ασχοληθούμε ιδιαίτερα με το σχεδιασμό του. Περιλαμβάνει πολλά στοιχεία τα οποία μπορούμε να τα χρησιμοποιήσουμε στην εφαρμογή μας. Ο κώδικας περιέχει HTML και CSS καθώς επίσης και προαιρετικές επεκτάσεις JavaScript. Είναι συμβατό με όλους τους περιηγητές διαδικτύου και υποστηρίζει ανταποκρίσιμο σχεδιασμό. Δηλαδή λαμβάνει τα χαρακτηριστικά της συσκευής την οποία χρησιμοποιούμε (Υπολογιστής, κινητό τηλέφωνο, tablet) και προσαρμόζει δυναμικά την διάταξη της ιστοσελίδας. Οι λόγοι που επιλέξαμε να χρησιμοποιήσουμε το framework αυτό είναι α) ότι είναι δωρεάν β) περιλαμβάνει πολλά στοιχεία που μας βοήθησαν στην εφαρμογή μας και γ) είναι εύκολη στην χρήση [17].

3 Από τα Δεδομένα στις Βάσεις Δεδομένων

3.1 Δεδομένα και Πληροφορίες

Στην σημερινή εποχή της πληροφορίας χαρακτηρίζεται από την ικανότητα των ανθρώπων να μπορούν να ανταλλάσσουν και να μεταφέρουν πληροφορίες με μεγάλη ελευθερία έχοντας πρόσβαση σε γνώσεις που θα ήταν δύσκολο ή και αδύνατο να αποκτηθούν στο παρελθόν. Στην καθημερινότητα μας διαχειριζόμαστε ένα πλήθος πληροφοριών, οι οποίες μας διευκολύνουν να κατανοήσουμε οτιδήποτε συμβαίνει στο κοντινό περιβάλλον μας και να αντιμετωπίσουμε τα προβλήματα που μας απασχολούν. Η έννοια της πληροφορίας σημαίνει ειδήσεις, γεγονότα, ιδέες, κάτι που μπορεί να αποκτηθεί και να θεωρηθεί γνώση. Στις αρχές του 20ου αιώνα όπου εμφανίστηκε η θεωρία των πληροφοριών καθορίστηκε σαφώς και το πλαίσιο της με αυστηρούς μαθηματικούς νόμους. Μερικές από τις επιτυχημένες πρακτικές της θεωρίας των πληροφοριών είναι η σχεδίαση συστημάτων ανίχνευσης πλοίων αεροπλάνων κλπ, οι εφαρμογές στη μετάδοση των εικόνων, συλλογή στοιχείων από δορυφόρους και η επικοινωνία υπολογιστών και κινητών τηλεφώνων [2].

Στοιχεία πληροφορίας ή Δεδομένα είναι οποιαδήποτε παράσταση, όπως χαρακτήρες ή αριθμητικές ποσότητες, σύμβολα κτλ., στην οποία δίνεται ή είναι δυνατόν να δοθεί μια σημασία.

Πληροφορία είναι η σημασία που δίνει ο άνθρωπος σε ένα σύνολο δεδομένων, τα οποία επεξεργάζεται με τη βοήθεια προκαθορισμένων συμφωνιών που έχουν θεσπιστεί από τον ίδιο.

Σαν συμπέρασμα από τους παραπάνω ορισμούς προκύπτει ότι τα δεδομένα μπορεί να είναι αποσπασματικά και ακατέργαστα. Η συλλογή και ο συσχετισμός των δεδομένων δίνει ως αποτέλεσμα την πληροφορία.

3.2 Σημασία της Διαχείρισης Δεδομένων

Στις μέρες μας τα δεδομένα καταγράφονται και χρησιμοποιούνται σε καθημερινή βάση σε διάφορες δραστηριότητες όπως επιστημονική, εμπορική, κυβερνητική, στρατιωτική κλπ. Υπάρχουν μεγάλες ποσότητες δεδομένων που επιβάλουν την εύρεση αποτελεσματικών μεθόδων αποθήκευσης. Για να υπάρχει ευκολία στην αναζήτηση και ενημέρωση των δεδομένων, πρέπει να είναι οργανωμένα. Η ταχύτητα, η ακρίβεια και η ασφάλεια με την οποία αποθηκεύουν τον τεράστιο όγκο δεδομένων είναι οι βασικοί λόγοι που αναθέτουμε στους υπολογιστές την επίλυση προβλημάτων [2].

Αποτέλεσμα αυτού ήταν η ανάπτυξη συστημάτων όπου τα δεδομένα αποθηκεύονται σε ψηφιακή μορφή, όπως μαγνητικές ταινίες , μαγνητικούς δίσκους, με τη χρήση πάντα κατάλληλων εφαρμογών λογισμικού. Οι εφαρμογές αυτές, εκτός από αποθήκευση δεδομένων, έδιναν στον χρήστη επιπλέον δυνατότητες όπως ενημέρωση, διαγραφή και επεξεργασία των δεδομένων.

3.3 Εξέλιξη της Διαχείρισης Δεδομένων

Τα αρχικά συστήματα εφαρμογών με ηλεκτρονική τήρηση δεδομένων που χρησιμοποιήθηκαν για την αποθήκευση και εκμετάλλευση πληροφοριακών δεδομένων, ήταν οργανωμένα σε ανεξάρτητα και μεμονωμένα αρχεία. Τα συστήματα αυτά επεκτείνουν τη λογική οργάνωσης των πληροφοριών, αξιοποιώντας στο έπακρο την δύναμη των υπολογιστών. Αυτός ο τρόπος διαχείρισης δίνει την δυνατότητα στον προγραμματιστή να υλοποίηση τις μεθόδους που επιθυμεί, αυξάνοντας όμως τον όγκο δεδομένων και την πολυπλοκότητα της πληροφορίας [2].

Ένα χαρακτηριστικό αυτών των εφαρμογών είναι ότι για να επικοινωνήσουν με τα αρχεία, πρέπει να γνωρίζουν τον ακριβή τρόπο οργάνωσης των δεδομένων μέσα σ' αυτά. Επομένως η εφαρμογή πρέπει να περιλαμβάνει στον κώδικάς της ένα κομμάτι που περιγράφει την σειρά αποθήκευσης των δεδομένων, για να είναι σε θέση να διαβάσει, εγγράψει, μεταβάλει ή διαγράψει δεδομένα. Καταλήγουμε στο συμπέρασμα ότι υπάρχει ανάγκη συμβατότητας της λογικής συγγραφής του κώδικα με τον τρόπο οργάνωσης των δεδομένων των αρχείων με τα οποία επικοινωνεί. Η απευθείας επικοινωνία της εφαρμογής με τα δεδομένα που πραγματοποιείται από τον κώδικα της κάθε εφαρμογής, ονομάζεται φυσική διεπαφή (interface). Επειδή κάθε εφαρμογή μπορεί να επικοινωνήσει μόνο με τα δικά της αρχεία αυτό έχει σαν αποτέλεσμα οι εφαρμογές να εξαρτώνται από τα δεδομένα (data-dependent) [2].

3.4 Εφαρμογές Βάσεων Δεδομένων

- **Κρατήσεις θέσεων σε αεροπορικές εταιρίες.** Στις εφαρμογές αυτές υποστηρίζονται πολλές λειτουργίες, όπως κράτηση θέσης για μια συγκεκριμένη πτήση, αναζήτηση πληροφοριών για πτήσεις με συγκεκριμένες αφετηρίες και προορισμούς, αναζήτηση πληροφοριών για τις τιμές και τη διαθεσιμότητα των εισιτηρίων [2].

- **Τραπεζικές Συναλλαγές.** Η πληροφορία αποτελείται από ονόματα πελατών, διευθύνσεις, τραπεζικούς λογαριασμούς, πιστωτικές κάρτες και ότι άλλο αφορά μια τράπεζα. Ο στόχος των εφαρμογών αυτών είναι η ταχύτερη εξυπηρέτηση των πελατών. Με τις νέες υπηρεσίες που βασίζονται στο διαδίκτυο καταργείτε η φυσική παρουσία στο κατάστημα υποστηρίζοντας ταυτόχρονες προσπελάσεις χρηστών [2].

- **Διαχείριση Εταιρικών Δεδομένων.** Οι μεγάλες εταιρείες χρειάζονται οργάνωση των δεδομένων της για να επιτύχει ουσιαστική και δυναμική συμβολή στην αποδοτική λειτουργία της. Σε όλα τα τμήματα της εταιρείας χρειάζεται η μεθοδική οργάνωση των δεδομένων, για να επιτρέπει γρήγορη και άμεση ενημέρωση των δεδομένων [2].

- **Πολυμεσικές Εφαρμογές.** Στις μέρες μας μεγάλο ποσοστό της πληροφορίας είναι στην μορφή του ήχου, της εικόνας και του video, η υποστήριξη των τύπων αυτών απαιτεί μεγαλύτερο χώρο αποθήκευσης και ταχύτητα στην επεξεργασία γιατί υπάρχει μεγάλος όγκος δεδομένων. Οι Βάσεις Δεδομένων πολυμέσων χρησιμοποιούν σύγχρονες και βελτιωμένες τεχνικές για την διαχείριση σύνθετων τύπων δεδομένων [2].

- **Ανάκτηση πληροφορίας στο διαδίκτυο.** Στο διαδίκτυο υπάρχει διάχυτη πληροφορία κάνοντας την αναζήτηση τον επόμενο στόχο. Η έρευνα εστιάζει στη μελέτη αποτελεσματικών και αποδοτικών μεθόδων αναζήτησης πληροφορίας. Οι μηχανές αναζήτησης όπως Google,Bing, Yahoo!, κ.λπ. κάνουν χρήση εξελιγμένων τεχνικών ικανοποιώντας εκατομμύρια ερωτήματα που δίνει ο χρήστης [2].

- **Αποθήκες Δεδομένων και Αναλυτική Επεξεργασία.** Μία άλλη σύγχρονη εφαρμογή που προέκυψε ως ανάγκη λόγω της ύπαρξης τεράστιου όγκου δεδομένων σε μεγάλες επιχειρήσεις και οργανισμούς είναι τα Συστήματα Στήριξης Αποφάσεων (decision support systems). Για τη λήψη αποφάσεων σε θέματα ανάλυσης αγοράς, οικονομικού σχεδιασμού, marketing κ.λπ., ένας αναλυτής μίας επιχείρησης ενδιαφέρεται να υποβάλλει πολύπλοκα ερωτήματα περισσότερο σε συγκεντρωτικά δεδομένα παρά σε ατομικές συναλλαγές. Η υποβολή και η επεξεργασία τέτοιου τύπου ερωτημάτων είναι μία διαδικασία από δύσκολη μέχρι και αδύνατη, καθώς συχνά τα δεδομένα των μεγάλων φορέων είναι διασκορπισμένα σε πολλά ετερογενή συστήματα, που μπορεί να είναι και γεωγραφικά κατανεμημένα. Στις περιπτώσεις αυτές, τα πρωτογενή δεδομένα με ειδικές διαδικασίες ομογενοποιούνται και αποθηκεύονται σε ειδικές βάσεις με συνοπτικό τρόπο χρησιμοποιώντας διαφορετικές οπτικές γωνίες και διαφορετικές κλίμακες. Αυτές οι ειδικές βάσεις λέγονται Αποθήκες Δεδομένων (data

warehouses), ενώ η επεξεργασία τους έχει την ειδική ονομασία Άμεση Αναλυτική Επεξεργασία (On-Line Analytical Processing, OLAP) σε αντίθεση με τα παραδοσιακά Συστήματα Βάσεων Δεδομένων, όπου μιλούμε για Άμεση Επεξεργασία Συναλλαγών [2].

- **Εξόρυξη Δεδομένων.** Η Εξόρυξη Δεδομένων (data mining) είναι μία άλλη σύγχρονη περιοχή των Βάσεων Δεδομένων, που πρόκυψε ως ανάγκη λόγω του τεράστιου όγκου δεδομένων, που συλλέγονται είτε αυτοματοποιημένα από δορυφόρους, κάμερες παρακολούθησης, συσκευές μέτρησης είτε με τον κλασσικό παραδοσιακό τρόπο από ένα Σύστημα Βάσεων Δεδομένων. Ο τεράστιος όγκος δεδομένων αποτρέπει την χρήση συμβατικών μεθόδων επεξεργασίας και έτσι εφαρμόζουμε ειδικές αυτοματοποιημένες μεθόδους επεξεργασίας, οι οποίες αναλαμβάνουν να "σκάψουν" στα δεδομένα αναζητώντας κάποια χρήσιμη πληροφορία, προηγουμένως άγνωστη και κυρίως ενδιαφέρουσα, με την οποία να μπορούν ληφθούν αποφάσεις και να δρομολογηθούν διαδικασίες. Η αποδοτική διαχείριση τεράστιων όγκων δεδομένων με τη βοήθεια κλασικών τεχνικών των ΒΔ είναι το χαρακτηριστικό της. Τυπικές μέθοδοι επεξεργασίας κατά την εξόρυξη δεδομένων είναι ο χαρακτηρισμός (characterization), η αντιδιαστολή (discrimination), η κατηγοριοποίηση (classification), η εξαγωγή κανόνων συσχέτισης (association rules), η ομαδοποίηση (clustering) κ.α [2].
- **Διαχείριση Κινούμενων Αντικειμένων** είναι μια ακόμη εφαρμογή με ραγδαία ανάπτυξη. Εφαρμογές όπως η καταγραφή της κίνησης οχημάτων, η παρακολούθηση μίας πετρελαιοκηλίδας, η παρακολούθηση εξέλιξης παγετώνων, απαιτούν την αποθήκευση της θέσης των αντικειμένων σε σχέση με το χρόνο. Η μελέτη της κίνησης επιτρέπει να απαντούμε σε καίρια ερωτήματα που σχετίζονται κύρια με: α) την εξέλιξη της κίνησης στο παρελθόν και β) την πρόβλεψη για τη θέση των αντικειμένων στο κοντινό ή μακρινό μέλλον. Η επεξεργασία ερωτημάτων τέτοιου τύπου είναιεξαιρετικά δύσκολη και απαιτεί την κατάλληλη οργάνωση και διαχείριση των αντικειμένων με μοντέρνες ευρηματικές τεχνικές έτσι ώστε να υλοποιείται αποτελεσματικά και αποδοτικά [2].

3.5 Λειτουργίες Συστήματος Διαχείρισης Βάσης Δεδομένων

Το Συστήματος Διαχείρισης Βάσης Δεδομένων γνωστό και ως **ΣΔΒΔ**, έχει σαν βασικό στόχο τον γενικό χειρισμό της βάσης, όσον αφορά τη δημιουργία, συντήρηση, επεξεργασία στοιχείων, ελέγχους ασφαλείας κτλ, και την εξυπηρέτηση των χρηστών σε όλα τα επίπεδα. Ουσιαστικά, θα μπορούσαμε να ισχυριστούμε, το ΣΔΒΔ είναι ένας **μεσάζων** μεταξύ του χρήστη και της βάσης δεδομένων και μόνο μέσω αυτού ο χρήστης μπορεί να ζητήσει πληροφορίες από τη βάση. Μπορεί το ΣΔΒΔ να είναι εγκατεστημένο σε ένα Η/Υ και να χρησιμοποιείται από ένα χρήστη (single user system) ή να είναι εγκατεστημένο σε ένα σύνολο ηλεκτρονικών υπολογιστών, που επικοινωνούν μεταξύ τους μέσω κάποιου (τοπικού ή απομακρυσμένου) δικτύου, και να χρησιμοποιείται από πολλούς χρήστες (multi user system) [2]. Οι βασικές λειτουργίες τις οποίες αναλαμβάνει ένα ΣΔΒΔ είναι όπως είδαμε παραπάνω:

- Οργανώνει τη βάση δεδομένων στο μέσο αποθήκευσης (σκληροί δίσκοι, οπτικοί δίσκοι, νέφος κλπ.).
- Διαθέτει τους μηχανισμούς για τη διαχείριση των δεδομένων. Τροφοδοτεί τις εφαρμογές με δεδομένα στη μορφή που αυτές τα ζητούν.

3.6 Χαρακτηριστικά ενός ΣΔΒΔ

Την ανάγκη για αξιολόγηση της αξιοπιστίας σε οτιδήποτε αφορά τις συναλλαγές στη βάση δεδομένων, έρχεται να εγγυηθεί η αποδοχή των απαιτήσεων **ACID** (Atomicity, Consistency, Isolation, Durability) - (Ατομικότητα, Συνέπεια, Απομόνωση, Μονιμότητα). Σαν **συναλλαγή** θα θεωρήσουμε οποιαδήποτε λογική ενέργεια που σχετίζεται με τα δεδομένα. Π.χ. μια μεταφορά κεφαλαίου από ένα λογαριασμό μιας τράπεζας σε έναν άλλο, όσες κινήσεις και να χρειάζεται τελικά είναι μια συναλλαγή [2].

- Η **Ατομικότητα** απαιτεί η τροποποίηση που θα γίνει στην Βάση Δεδομένων να τηρεί τον κανόνα **όλα ή τίποτα**, αν δηλαδή ένα μέρος της συναλλαγής αποτύχει, αποτυγχάνει όλη η συναλλαγή καιη ΒΔ μένει όπως ήταν πριν εκτελεστεί η συναλλαγή. Η Ατομικότητα σημαίνει ότι οι χρήστες είναι απαλλαγμένοι από τον φόβο μη ολοκληρωμένων συναλλαγών.
- Η **Συνέπεια** διασφαλίζει ότι η ΒΔ διατηρείται σε μια συνεπή κατάσταση, συγκεκριμένα λέει ότι κάθε συναλλαγή θα οδηγεί την βάση δεδομένων από τη μια συνεπή κατάσταση στην άλλη. Σε περίπτωση που μια συναλλαγή παραβιάζει κάποιο κανόνα της συνέπειας, ανακαλείται προκειμένου η ΒΔ να έχει μόνο έγκυρα δεδομένα. Π.χ. αν σε μια ΒΔ ένα πεδίο είναι μόνο για ακέραιους αριθμούς τότε το ΣΔΒΔ μπορεί είτε να απορρίψει απόπειρες για είσοδο δεκαδικών αριθμών είτε να τους στρογγυλοποιήσει. Και οι δυο αυτές ενέργειες διατηρούν την συνέπεια. Υπάρχουν τρία είδη συνέπειας, ισχυρή, ασθενής και ενδεχόμενη, η μελέτη τους όμως ξεφεύγει από το πλαίσιο του παρόντος βιβλίου.
- Η **Απομόνωση** αναφέρεται στην απαίτηση ότι όλες οι ενέργειες δεν μπορούν να έχουν πρόσβαση ή να δουν δεδομένα τα οποία τροποποιούνται εκείνη την στιγμή από μια συναλλαγή η οποία δεν έχει ακόμα ολοκληρωθεί.
- Η **Μονιμότητα** εγγυάται στον χρήστη ότι αν τελειώσει μια συναλλαγή επιτυχώς τότε τα αποτελέσματα της δεν θα χαθούν. Οι αλλαγές που έχει κάνει η συναλλαγή δεν θα χαθούν σε περίπτωση απώλειας ρεύματος ή άλλης καταστροφής.

Όταν έχουμε ένα πολύπλοκα διαμοιρασμένο σύστημα αποθήκευσης πληροφοριών είναι φυσικό να αναπτυχθεί και αντίστοιχος προβληματισμός σε θεωρητικό επίπεδο. Αποτέλεσμα αυτής της θεωρητικής μελέτης είναι το **θεώρημα CAP** (Consistency, Availability, Partition tolerance) (Συνέπεια, Διαθεσιμότητα, ανοχή Διαμερισμού), γνωστό επίσης και ως θεώρημα Brewer. Σύμφωνα με αυτό είναι **αδύνατο** για ένα κατανεμημένο σύστημα υπολογιστών να παρέχει **ταυτόχρονα** και τις τρεις εγγυήσεις:

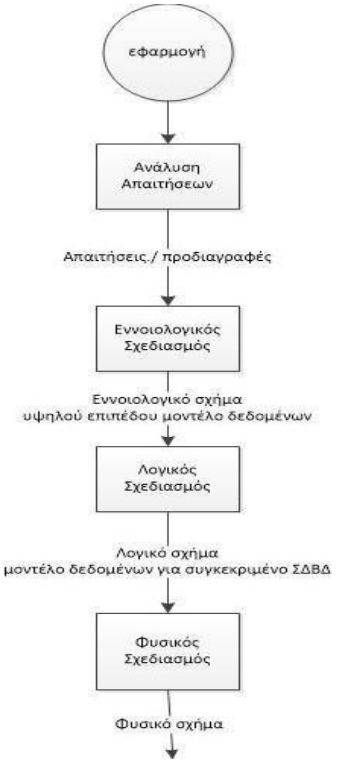
- **Συνέπεια**, αναπτύχθηκε βέβαια προηγουμένως αλλά κατά τον Brewer η κεντρική ιδέα είναι ότι όλοι οι κόμβοι βλέπουν τα ίδια δεδομένα την ίδια στιγμή.
- **Διαθεσιμότητα**, εγγύηση ότι κάθε αίτημα παίρνει μια απάντηση σχετικά με το αν πέτυχε ή απέτυχε.
- **ανοχή Διαμερισμού**, εγγύηση ότι το σύστημα συνεχίζει να λειτουργεί παρά τον διαμερισμό που οφείλεται σε βλάβες του δικτύου.

4 Σχεδιασμός Βάσεων Δεδομένων

4.1 Εισαγωγή

Τα συστήματα βάσεων δεδομένων έχουν σχεδιαστεί για να διαχειρίζονται μεγάλους όγκους πληροφοριών. Οι μεγάλοι αυτοί όγκοι πληροφοριών αποτελούν μέρος της λειτουργίας μίας επιχείρησης ή ενός οργανισμού. Το τελικό προϊόν της επιχείρησης ή του οργανισμού μπορεί να είναι η παροχή πληροφοριών από τη βάση δεδομένων ή κάποια συσκευή ή υπηρεσία. Στην τελευταία περίπτωση η βάση δεδομένων έχει μόνο υποστηρικτικό ρόλο. Το κεφάλαιο αυτό επικεντρώνεται στον σχεδιασμό βάσεων δεδομένων και ειδικότερα στον εννοιολογικό και λογικό σχεδιασμό της βάσης δεδομένων [2].

Για να είναι δυνατή η υλοποίηση μιας εφαρμογής με μεγάλη πολυπλοκότητα απαιτείται ο σωστός σχεδιασμός της. Για παράδειγμα για τον σωστό σχεδιασμό ενός αυτοκινήτου απαιτείται να προηγηθούν αρκετά σχέδια τα οποία απεικονίζουν διαφορετικές απόψεις του αντικειμένου. Οι βάσεις δεδομένων αποτελούν και αυτές πολύπλοκα συστήματα τα οποία μπορούν να περιγράφουν με διαφορετικούς σχεδιασμούς. Για τη Σχεδίαση μιας Βάσης δεδομένων απαιτούνται τα παρακάτω βήματα:



Εικόνα 6 - Τα βήματα για τη σχεδίαση μιας Βάσης Δεδομένων

4.1.1 Ανάλυση Απαιτήσεων

Απαίτηση είναι η περιγραφή μιας υπηρεσίας που θα πρέπει να παρέχει ένα σύστημα, μιας διεργασίας που θα πραγματοποιεί ή μιας συνθήκης που θα πρέπει να ικανοποιεί. Η απαίτηση απαντάει στο ερώτημα «τι κάνει το σύστημα;» και δεν περιλαμβάνει τον τρόπο που θα γίνει αυτό. Αποτελεί μια προσπάθεια αποσαφήνισης του προβλήματος που στοχεύει να λύσει η εφαρμογή που θα δημιουργηθεί και όχι στην περιγραφή κάποιας λύσης [2].

Η απαίτηση καταγράφεται και τεκμηριώνεται με τέτοιο τρόπο ώστε να είναι κατανοητή από όλα τα ενδιαφερόμενα μέρη στην ανάπτυξη της εφαρμογής. Επιπλέον η απαίτηση είναι μια σημαντική σταθερά που παραμένει ως ένας παράγοντας αξιολόγησης σε όλη τη διάρκεια του κύκλου ζωής ενός προϊόντος. Είναι δηλαδή ένας είδος συμβολαίου που ελέγχεται ακόμα και μετά το τέλος της ανάπτυξης του προϊόντος [2].

Η ανάλυση απαιτήσεων είναι μια διαδικασία αποτύπωσης όλων των απαιτήσεων, που πρέπει να πληροί η εφαρμογή. Η διαδικασία προσδιορισμού απαιτήσεων αποτελεί ένα δύσκολο έργο καθώς σε πολλές περιπτώσεις η επιτυχία μίας εφαρμογής στηρίζεται στην σωστή ανάλυση των απαιτήσεων. Για το σκοπό αυτό ο αναλυτής του συστήματος προετοιμάζει ένα έγγραφο απαιτήσεων που περιλαμβάνει πληροφορίες όπως:

- Ποια δεδομένα αποθηκεύονται και που
- Ποιες εφαρμογές θα χτιστούν πάνω από τα δεδομένα;
- Ποιες διαδικασίες πρέπει να υποστηριχθούν
- Ποιες είναι οι απαιτήσεις επίδοσης

Εφόσον έχει ολοκληρωθεί η διαδικασία προσδιορισμού απαιτήσεων το επόμενο βήμα αποτελεί ο σχεδιασμός της βάσης δεδομένων. Σε αυτό το βήμα είναι βασικό στοιχείο η περιγραφή της βάσης μέσα από τρεις διαφορετικούς σχεδιασμούς.

4.1.2 Εννοιολογικός, Λογικός και Φυσικός Σχεδιασμός

Εννοιολογικός Σχεδιασμός

Στον Εννοιολογικό Σχεδιασμό περιγράφονται οι οντότητες του πραγματικού κόσμου καθώς και οι συσχετίσεις μεταξύ τους. Για παράδειγμα σε μια εφαρμογή μαθητολογίου σε ένα σχολείο οντότητες μπορεί να είναι οι μαθητές, οι καθηγητές, τα μαθήματα, οι βαθμοί κ.α. Η οντότητα μαθητής συσχετίζεται με την οντότητα μάθημα και την οντότητα βαθμοί. Η οντότητα μαθητής έχει ιδιότητες όπως αριθμός μητρώου, επώνυμο, όνομα κ.α. Επίσης, περιγράφονται οι περιορισμοί που υπάρχουν στα δεδομένα (π.χ. δεν επιτρέπονται βαθμοί μικρότεροι του 0) και οι συσχετίσεις μεταξύ τους. Για κάθε οντότητα καθορίζεται το πρωτεύον κλειδί που η τιμή του είναι μοναδική για παράδειγμα στην οντότητα μαθητής πρωτεύον κλειδί μπορεί να είναι ο αριθμός μητρώου [2].

Για τον σωστό εννοιολογικό σχεδιασμό έχουν αναπτυχθεί διαγράμματα όπως τα Διαγράμματα Οντοτήτων Συσχετίσεων (ΔΟΣ). Για παράδειγμα, σε αυτή τη φάση προσδιορίζονται οι οντότητες και οι συσχετίσεις, στη συνέχεια καθορίζονται οι ιδιότητες των οντοτήτων και συσχετίσεων, οι τύποι των συσχετίσεων και προσδιορίζονται τα πρωτεύοντα κλειδιά κάθε οντότητας Η φάση αυτή ολοκληρώνεται με την κατασκευή και τον έλεγχο του ΔΟΣ.

Ο εννοιολογικός σχεδιασμός είναι ανεξάρτητος από τη φυσική οργάνωση των δεδομένων, με αποτέλεσμα να μην εξαρτάται από κάποιο συγκεκριμένο τρόπο αποθήκευσης.

Λογικός Σχεδιασμός

Ο Λογικός Σχεδιασμός αφορά τον σχεδιασμό του Σχεσιακού Μοντέλου το οποίο προκύπτει από την μεταφορά του ΔΟΣ που έχει κατασκευαστεί από τον εννοιολογικό σχεδιασμό. Για την σωστή σχεδίαση του Σχεσιακού Μοντέλου εφαρμόζονται μέθοδοι κανονικοποίησης και γίνεται έλεγχος πλεονασμού και ακεραιότητας δεδομένων. Ο σχεδιασμός ολοκληρώνεται με την ανασκόπηση όλης της διαδικασίας και την πρόβλεψη μελλοντικών αναγκών [2].

Φυσικός Σχεδιασμός

Ο Φυσικός σχεδιασμός περιγράφει τη φυσική οργάνωση της βάσης. Καθορίζεται ο τρόπος με τον οποίο αποθηκεύονται τα δεδομένα στο φυσικό μέσο αποθήκευσης (π.χ. σκληρό δίσκο) και ο τρόπος με τον οποίο πραγματοποιείται η προσπέλαση στα δεδομένα. Σε αυτή τη φάση γίνεται έλεγχος ότι η βάση ικανοποιεί τις απαιτήσεις επίδοσης/φόρτου κ.α, τα οποία επιβάλλουν οι απαιτήσεις του εγγράφου απαιτήσεων [2].

4.2 Μοντέλα Δεδομένων

Η περιγραφή των δεδομένων μιας βάσης δεδομένων πραγματοποιείται από το μοντέλο δεδομένων (data models). Αποτελεί ένα σύνολο εργαλείων για την περιγραφή των σχέσεων των δεδομένων, της σημασίας των δεδομένων και των περιορισμών συνέπειας [2].

Τα μοντέλα δεδομένων περιγράφουν το σχήμα της Βάσης Δεδομένων σε αρκετά υψηλό επίπεδο, χωρίς λεπτομέρειες υλοποίησης. Θα πρέπει να τονιστεί ότι λάθη ή παραλήψεις στο μοντέλο δεδομένων έχουν άμεσο αντίκτυπο στα αποθηκευμένα δεδομένα όσο και στις λειτουργίες επεξεργασίας δεδομένων. Η αλλαγή του μοντέλου δεδομένων συνεπάγεται αλλαγές σε πολλά τμήματα της Βάσης Δεδομένων.Έχουν αναπτυχθεί δύο μοντέλα δεδομένων τα οποία θα περιγραφούν σε αυτό το κεφάλαιο, το Μοντέλο Οντοτήτων Συσχετίσεων και το Σχεσιακό Μοντέλο [2].

4.2.1. Εννοιολογικό, Λογικό και Φυσικό Σχήμα (Μοντέλο)

Για να είναι χρήσιμη μια βάση δεδομένων πρέπει να μπορεί να ανακαλεί τα δεδομένα της αποτελεσματικά. Οι σχεδιαστές των βάσεων για την αποτελεσματική ανάκληση δεδομένων χρησιμοποιούν πολύπλοκες δομές δεδομένων, οι οποίες χρησιμοποιούνται για την αναπαράσταση των δεδομένων της βάσης δεδομένων. Όμως, οι χρήστες μιας Βάσης Δεδομένων δεν είναι ανάγκη να είναι έμπειροι προγραμματιστές, για αυτό το λόγο οι σχεδιαστές κρύβουν την πολυπλοκότητα από τους χρήστες μέσω διαφόρων επιπέδων αφαιρετικότητας. Για να επιτευχθεί η απόκρυψη της πολυπλοκότητας από τους χρήστες χρησιμοποιούνται τρία σχήματα (βλέπε εικόνα 7) το εννοιολογικό σχήμα, το λογικό σχήμα και το φυσικό σχήμα. Στη βιβλιογραφία το εννοιολογικό και το λογικό σχήμα αναφέρονται πολλές φορές ως ένα σχήμα για αυτό το λόγο στην παρούσα ενότητα το λογικό σχήμα αναφέρεται και ως λογικό (εννοιολογικό) σχήμα [2].

Εικόνα 7 - Εννοιολογικό, Λογικό και Φυσικό σχήμα

Το χαμηλότερο επίπεδο αφαιρετικότητας (**φυσικό σχήμα**) περιγράφει τον τρόπο αποθήκευσης των δεδομένων . Σε αυτό περιγράφονται με λεπτομέρεια οι πολύπλοκες δομές που χρησιμοποιούνται για την αποθήκευση και την ανάκτηση των δεδομένων.

Το **λογικό (εννοιολογικό) σχήμα** έχει υψηλότερο επίπεδο αφαιρετικότητας και περιγράφει ποια δεδομένα αποθηκεύονται και ποιες είναι οι σχέσεις μεταξύ τους. Στην ουσία, το λογικό σχήμα περιγράφει τη βάση δεδομένων με απλές δομές. Αυτό εξασφαλίζει ότι αν και μπορεί να απαιτούνται πολύπλοκες δομές στο φυσικό σχήμα ο χρήστης της Βάσης Δεδομένων δεν χρειάζεται να τις γνωρίζει.

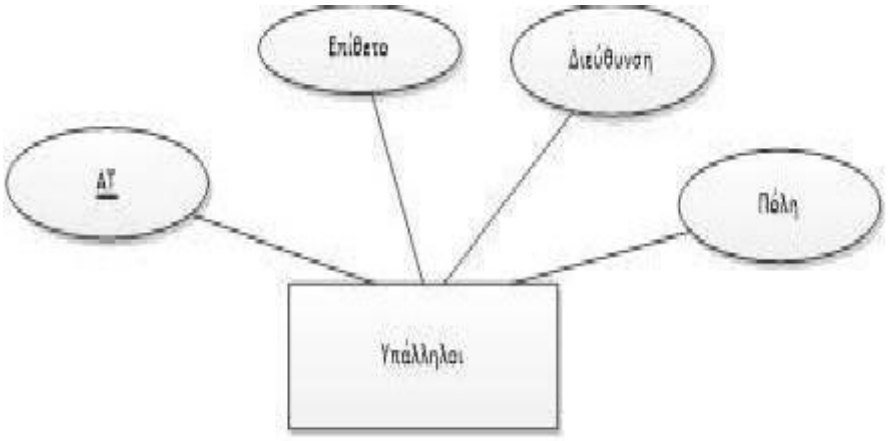
Το υψηλότερο επίπεδο αφαιρετικότητας (**εννοιολογικό σχήμα**) περιγράφει μόνο ένα κομμάτι της Βάσης Δεδομένων. Αν και στο λογικό σχήμα τα δεδομένα περιγράφονται με απλές δομές ο χρήστης δεν χρειάζεται να τις γνωρίζει. Με αυτόν τον τρόπο η συνδιαλλαγή με το σύστημα της Βάσης Δεδομένων γίνεται πιο απλή. Διαφορετικοί χρήστες έχουν προσπέλαση σε διαφορετικά δεδομένα της Βάσης Δεδομένων, ανάλογα με τους περιορισμούς και τους κανόνες προσπέλασης που εφαρμόζονται.

Ένας από τους πιο σημαντικούς λόγους για την χρήση ΣΔΒΔ είναι η υποστήριξη της **ανεξαρτησίας** των δεδομένων. Αυτό σημαίνει ότι ο κώδικας δεν επηρεάζεται από μεταβολές στον τρόπο οργάνωσης και καταχώρισης των δεδομένων. Η ανεξαρτησία των δεδομένων γίνεται μέσα από τα τρία επίπεδα αφαιρετικότητας που έχουν περιγραφεί. Ο χρήστης προστατεύεται από αλλαγές στη λογική δομή των δεδομένων μέσω της λογικής ανεξαρτησίας των δεδομένων. Με ανάλογο τρόπο το λογικό σχήμα απομονώνει τον χρήστη από αλλαγές που συμβαίνουν στο φυσικό σχήμα (για παράδειγμα, αλλαγές που αφορούν την αποθήκευση των δεδομένων στην περιφερειακή μνήμη, επιλογές ευρετηρίων, κ.α). Όσο το εννοιολογικό σχήμα μένει ανεπηρέαστο, η υλοποίηση των παραπάνω μπορεί να αλλάζει χωρίς να χρειάζεται αλλαγή στον κώδικα. Βέβαια, αυτές οι αλλαγές μπορεί να έχουν αρνητικές επιπτώσεις στις επιδόσεις του συστήματος.

4.2.2. Μοντέλο Οντοτήτων Συσχετίσεων

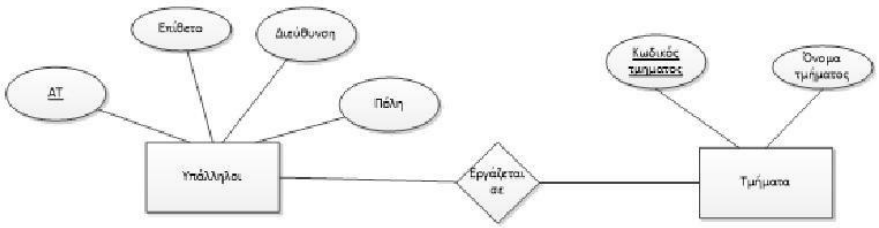
Το μοντέλο οντοτήτων συσχετίσεων βασίζεται στο πως αντιλαμβάνεται κάποιος τον φυσικό κόσμο. Αποτελείται από ένα σύνολο αντικειμένων του φυσικού κόσμου που ονομάζονται οντότητες στο συγκεκριμένο μοντέλο και από τις σχέσεις των αντικειμένων. Εκτός από τις οντότητες και τις σχέσεις, κάθε μοντέλο οντοτήτων συσχετίσεων αντιπροσωπεύει και κάποιους περιορισμούς που υπάρχουν και πρέπει τα περιεχόμενα της βάσης να υπακούν. Μία οντότητα περιγράφεται από ένα σύνολο γνωρισμάτων/χαρακτηριστικών που την χαρακτηρίζουν [2].

Το μοντέλο οντοτήτων συσχετίσεων αποτελεί το στάδιο του εννοιολογικού σχεδιασμού μιας βάσης δεδομένων και περιγράφει το λογικό (εννοιολογικό) σχήμα της βάσης δεδομένων. Εμείς στο παρόν κεφάλαιο θα παρουσιάσουμε μόνο το σχεσιακό μοντέλο ΣΔΒΔ, οπότε ο λογικός σχεδιασμός σημαίνει μετασχηματισμό του διαγράμματος οντοτήτων συσχετίσεων σε σχήμα σχεσιακής βάσης δεδομένων. Το μοντέλο οντοτήτων συσχετίσεων περιγράφεται από το διάγραμμα οντοτήτων. Παράδειγμα διαγραμματικής απεικόνισης του μοντέλου παρουσιάζεται στην εικόνα 3.2 που αφορά το σύνολο οντοτήτων Υπάλληλοι με γνωρίσματα όνομα, Αριθμός Αστυνομικής ταυτότητας ΑΤ, διεύθυνση και πόλη κατοικίας [2].



Εικόνα 8 - Σχηματική αναπαράσταση της οντότητας Υπάλληλοι

Οι συσχετίσεις ανάμεσα στις οντότητες αναπαριστώνται με ρόμβο. Παράδειγμα παρουσιάζεται στην εικόνα 3.3.



Εικόνα 9 - Σχηματική αναπαράσταση της συσχέτισης

Στην εικόνα 3.3 περιγράφονται οι οντότητες Υπάλληλοι και Τμήματα. Τα τμήματα αντιπροσωπεύουν τα τμήματα που εργάζονται οι υπάλληλοι στον φυσικό κόσμο (Λογιστήριο, Τμήμα προσωπικού, τμήμα πωλήσεων κ.α.). Η σχέση ανάμεσα στις δύο οντότητες περιγράφεται από τη συσχέτιση "Εργάζεται σε". Το μοντέλο οντοτήτων συσχετίσεων χρησιμοποιείται ευρέως στην ανάλυση των απαιτήσεων μιας βάσης δεδομένων [2].

Το χαρακτηριστικό το οποίο προσδιορίζει μοναδικά μία οντότητα καλείται πρωτεύον κλειδί. Το πρωτεύον κλειδί σημειώνεται σε ένα διάγραμμα ΟΣ με υπογράμμιση. Στην οντότητα Μαθητής ο αριθμός μητρώου προσδιορίζει μοναδικά τον κάθε μαθητή, επομένως, το πρωτεύον κλειδί είναι το χαρακτηριστικό ΑΜ. Σε πολλές περιπτώσεις όμως, δεν υπάρχει κάποιο χαρακτηριστικό που από μόνο του να αποτελεί κλειδί. Εναλλακτικά μπορούμε να χρησιμοποιήσουμε δύο η περισσότερα χαρακτηριστικά (εάν υπάρχουν) που στο σύνολό τους προσδιορίζουν μοναδικά μία οντότητα. Στην περίπτωση αυτή το κλειδί καλείται σύνθετο κλειδί [2].

Όταν δεν υπάρχει κάποιο χαρακτηριστικό που να μπορεί να χρησιμοποιηθεί ως κλειδί, τότε μπορούμε να ορίσουμε κάποιο τεχνητό χαρακτηριστικό που να προσδιορίζει μοναδικά κάθε οντότητα. Το χαρακτηριστικό αυτό συνήθως ονομάζεται κωδικός ή έχει κάποιο άλλο προσδιοριστικό [2].

Στο παράδειγμα της εικόνας 3.3. έχουμε χρησιμοποιήσει την ονομασία κωδικός τμήματος για κλειδί της οντότητας Τμήματα. Σημειώνεται ότι ένα σύνολο οντοτήτων μπορεί να έχει περισσότερα από ένα κλειδιά. Ένα από τα κλειδιά χαρακτηρίζεται ως πρωτεύον κλειδί ενώ τα υπόλοιπα ως δευτερεύοντα κλειδιά. Δευτερεύον κλειδί της οντότητας μαθητής μπορεί να είναι το επίθετο.

4.2.3 Σχεσιακό Μοντέλο

Το σχεσιακό μοντέλο χρησιμοποιείται ευρέως και πολλά συστήματα βάσης δεδομένων βασίζονται σε αυτό. Είναι σε χαμηλότερο επίπεδο αφαιρετικότητας από το μοντέλο οντοτήτων συσχετίσεων. Χαρακτηρίζεται από απλότητα και είναι ελκυστικό στη χρήση του. Το σχεσιακό μοντέλο δεδομένων (relational data model) αναπτύχθηκε από τον Codd το 1970. Τα περισσότερα εμπορικά ΣΔΒΔ υποστηρίζουν το σχεσιακό μοντέλο δεδομένων ή επεκτάσεις αυτού [2]. Οι βασικοί στόχοι του σχεσιακού μοντέλου είναι :

- Η υποστήριξη της ανεξαρτησίας των δεδομένων, έτσι ώστε αλλαγές στη φυσική δομή να μην απαιτούν αλλαγές στα προγράμματα εφαρμογής,
- Η αποφυγή της επανάληψης δεδομένων, όταν τα ίδια δεδομένα αποθηκεύονται πολλές φορές σε διαφορετικές περιοχές της Βάσης Δεδομένων.

Το σχεσιακό μοντέλο αποτελείται από πίνακες που κάθε πίνακας έχει ένα μοναδικό όνομα και προσδιορίζεται από ένα σύνολο γραμμών και ένα σύνολο στηλών. Κάθε γραμμή του πίνακα αναπαριστά μία εγγραφή δεδομένων, ενώ οι στήλες του πίνακα ορίζουν τα χαρακτηριστικά ή ιδιότητες της κάθε εγγραφής. Κάθε πίνακας περιέχει εγγραφές ενός συγκεκριμένου τύπου και κάθε τύπος εγγραφής ορίζει ένα σταθερό αριθμό πεδίων ή ιδιοτήτων. Ένας πίνακας μπορεί να χρησιμοποιηθεί τόσο για τις εγγραφές που περιγράφουν οντότητες, όσο και για τις εγγραφές που περιγράφουν συσχετίσεις. Συνήθως, θεωρούμε ότι οι στήλες είναι διατεταγμένες και συνηθίζουμε να αναφέρουμε πρώτα τις στήλες που αντιστοιχούν σε χαρακτηριστικά που είναι κλειδιά. Στην εικόνα 3.4 παρουσιάζεται παράδειγμα πίνακα σχεσιακού μοντέλου [2].

ΑΤ Υπαλλήλου	Επίθετο Υπαλλήλου	Διεύθυνση κατοικίας	Πόλη
ΑΑ 14355	Παπαδόπουλος	Βενιζέλου 4	Θεσσαλονίκη
ΑΒ 78155	Χαρισίδης	Τσιμισκή 19	Θεσσαλονίκη
ΑΒ 15889	Κερκινάκης	Εγνατία 151	θεσσαλονίκη

Εικόνα 10 - Πίνακας υπάλληλος

Στην εικόνα 3.4 παρουσιάζεται ο πίνακας υπαλλήλων μιας εταιρείας όπου ο υπάλληλος που προσδιορίζεται από τον αριθμό της Αστυνομικής ταυτότητας ΑΒ 15889 ονομάζεται Κερκινάκης, κατοικεί στην Θεσσαλονίκη και μένει στην διεύθυνση Εγνατίας 151.

Για κάθε χαρακτηριστικό/ιδιότητα υπάρχει ένα σύνολο επιτρεπτών τιμών, το οποίο καλείται πεδίο ορισμού του χαρακτηριστικού/ ιδιότητας. Μπορούμε να δημιουργήσουμε διατάξεις στο σχεσιακό μοντέλο όπου υπάρχει πρόβλημα δηλαδή όπου εμφανίζονται διπλές πληροφορίες. Για παράδειγμα, αν θεωρήσουμε ότι στον πίνακα υπάλληλος θα πρέπει να συμπεριλάβουμε τα ονόματα των παιδιών του κάθε υπαλλήλου και ο υπάλληλος Κερκινάκης έχει δύο παιδιά, θα έπρεπε να αποθηκεύσουμε 2 γραμμές στον πίνακα όπως φαίνεται στην εικόνα 3.5. Όμως, με αυτή τη διαδικασία δημιουργούνται διπλές πληροφορίες όπως το επίθετο, η διεύθυνση και η πόλη που είναι οι ίδιες και στις δύο γραμμές. Στην ενότητα 3.4 θα μελετήσουμε πως μπορούμε να ξεχωρίσουμε τις καλές σχεδιάσεις διατάξεων από τις "κακές" [2].

ΑΤ Υπαλλήλου	Επίθετο	Διεύθυνση	Πόλη	Όνομα παιδιού
ΑΑ 14355	Παπαδόπουλος	Βενιζέλου 4	Θεσσαλονίκη	**NULL**
ΑΒ 78155	Χαρισίδης	Τσιμισκή 19	Θεσσαλονίκη	Άννα
ΑΒ 15889	Κερκινάκης	Εγνατία 151	Θεσσαλονίκη	Κωνσταντίνος
ΑΒ 15889	Κερκινάκης	Εγνατία 151	Θεσσαλονίκη	Γεώργιος

Εικόνα 11 - Πίνακας υπαλλήλων με διπλές πληροφορίες

Υπάρχουν περιπτώσεις στις οποίες δε γνωρίζουμε την τιμή ενός χαρακτηριστικού. Στην εικόνα 3.5, παρατηρούμε ότι εφόσον ο Παπαδόπουλος δεν έχει παιδιά, η ιδιότητα Όνομα παιδιού έχει την τιμή **NULL** (κενή τιμή).Η κενή τιμή είναι διαφορετική από 0 ή από τον χαρακτήρα ≪SPACE≫. Μία κενή τιμή συνήθως δε δημιουργεί πρόβλημα, ωστόσο καλό είναι οι τιμές NULL να αποφεύγονται διότι δημιουργούνται δυσκολίες στη διατύπωση ερωτημάτων. Τονίζεται ότι δεν επιτρέπονται να υπάρχουν κενές τιμές στα κλειδιά ενός πίνακα.

Τα μοντέλα οντοτήτων συσχετίσεων και το σχεσιακό μοντέλο παρουσιάζουν πολλές ομοιότητες. Για παράδειγμα, μία οντότητα του μοντέλου ΟΣ αντιστοιχεί σε μία γραμμή ενός πίνακα στο σχεσιακό μοντέλο. Ένα σύνολο οντοτήτων αντιστοιχεί αντίστοιχα σε έναν πίνακα του σχεσιακού μοντέλου. Ως συνήθως το μοντέλο οντοτήτων συσχετίσεων μεταφράζεται σε σχεσιακό μοντέλο από τους σχεδιαστές μιας βάσης δεδομένων.

4.3 Διαγραμματική Αναπαράσταση Συσχετίσεων Οντοτήτων

Το μοντέλο οντοτήτων συσχετίσεων (ERM) αποτελεί την βάση του διαγράμματος οντοτήτων συσχετίσεων (ERD). Το διάγραμμα οντοτήτων συσχετίσεων αναπαριστά με εννοιολογικό τρόπο τις βάσεις δεδομένων. Απεικονίζει τα κύρια συστατικά των βάσεων δεδομένων: οντότητες (entities), ιδιότητες - γνωρίσματα (attributes) και συσχετίσεις (relationships). Η οντότητα αντιπροσωπεύει τόσο συγκεκριμένα αντικείμενα όσο και αφηρημένα γεγονότα και καταστάσεις [2].

4.3.1. Οντότητες, Στιγμιότυπα, Γνωρίσματα - Ιδιότητες και Κλειδιά

Ένα **στιγμιότυπο (instance)** είναι ένα «αντικείμενο» του πραγματικού κόσμου το οποίο ξεχωρίζει από τα άλλα αντικείμενα. Για παράδειγμα, στο σχολικό περιβάλλον κάθε μαθητής ή κάθε καθηγητής είναι ένα στιγμιότυπο. Ο μαθητής «Ανδρέας Παπαϊωάννου» και ο καθηγητής «Στέφανος Λαζάρου» αποτελούν στιγμιότυπα. Κάθε στιγμιότυπο έχει ένα σύνολο από ιδιότητες, όπως όνομα, επίθετο, διεύθυνση και αριθμός μητρώου ή ταυτότητας. Οι τιμές για κάποιες από τις ιδιότητες αυτές είναι δυνατό να είναι διαφορετικές για κάθε στιγμιότυπο. Για παράδειγμα ο αριθμός μητρώου είναι μοναδικός για κάθε μαθητή. Δεν θα μπορούσαμε όμως να ισχυριστούμε το ίδιο και για το όνομα, το επίθετο ή την διεύθυνση [2].

Μία **οντότητα (entity)** είναι ένα σύνολο από στιγμιότυπα του ίδιου τύπου που έχουν κοινές ιδιότητες ή γνωρίσματα. Για παράδειγμα, το σύνολο των καθηγητών ενός σχολείου αποτελούν την οντότητα Καθηγητής. Ο καθηγητής «Στέφανος Λαζάρου» ανήκει στην οντότητα Καθηγητής. Αποτελεί δηλαδή ένα στιγμιότυπο της οντότητας Καθηγητής. Παρόμοια, η οντότητα Μαθητής αναφέρεται στο σύνολο όλων των μαθητών ενός σχολείου. Ο μαθητής «Ανδρέας Παπαϊωάννου» είναι ένα στιγμιότυπο της οντότητας Μαθητής [2].

Μία οντότητα μπορεί να είναι κάτι που έχει μία φυσική υπόσταση όπως ένα πρόσωπο ή ένα βιβλίο ή μπορεί να είναι κάτι πιο γενικό, πιο αφηρημένο όπως ένα μάθημα, μία συναυλία ή μία δεξιότητα. Μία οντότητα είναι κάτι που είναι «σημαντικό» για μία επιχείρηση ή έναν οργανισμό και για το οποίο η επιχείρηση ή ο οργανισμός θέλει να ξέρει ορισμένα βασικά πράγματα. Για παράδειγμα, ένα σχολείο χρειάζεται να αποθηκεύσει δεδομένα σχετικά με τους Μαθητές, τους Καθηγητές, τα

Μαθήματα και τους Βαθμούς. Οι Μαθητές, οι Καθηγητές, τα Μαθήματα και οι Βαθμοί για το σχολείο θεωρούνται οντότητες, βασικά δηλαδή πράγματα για τα όποια υπάρχει ανάγκη να αποθηκεύσουμε κάποιες πληροφορίες. Πρέπει να γνωρίζουμε δηλαδή ποιοι είναι οι μαθητές μας και ποιά μαθήματα παρακολουθούν. Η οντότητα μπορεί να θεωρηθεί και ως σύνολο από στιγμιότυπα. Σε ένα σχολείο η οντότητα Μαθητής μπορεί να έχει ως στιγμιότυπα τον Παύλο, την Ελένη, τον Νίκο και τον Άλεξ [2].

Στον παρακάτω πίνακα παρουσιάζονται παραδείγματα οντοτήτων και αντίστοιχων στιγμιότυπων.

Οντότητες	**Στιγμιότυπα**
Πρόσωπο	Μέγας Αλέξανδρος, Θόδωρος Κολοκοτρώνης, Μαρία Παπαπέτρου
Προϊόν	Nikes Tzampadan
Είδος Προϊόντος	Αθλητικά
Εργασία	Μηχανικός, Γιατρός
Αγορά	Το βιβλίο που αγόρασα χθες
Ζώο	Γάτα, Σκύλος, Λιοντάρι
Εκλογές	Για το ευρωκοινοβούλιο το επόμενο φθινόπωρο

Εικόνα 12 - Παραδείγματα οντοτήτων και αντίστοιχων στιγμιότυπων

Θα πρέπει να σημειώσουμε ότι το προσδιοριστικό που δίνουμε σε ένα αντικείμενο για παράδειγμα αν θα είναι οντότητα ή στιγμιότυπο εξαρτάται κάθε φορά από το πλαίσιο στο οποίο αναφερόμαστε. Για παράδειγμα, ο σκύλος είναι οντότητα ή στιγμιότυπο; Εξαρτάται. Αν αναφερόμαστε στα διάφορα είδη ζώων, τότε έχει νόημα να ορίσουμε την οντότητα Ζώο που θα έχει ως στιγμιότυπα τον Σκύλο, την Γάτα και άλλα. Αν όμως αναφερόμαστε σε μία επιχείρηση εκτροφής και αναπαραγωγής σκύλων, τότε είναι φυσικό να ορίσουμε την οντότητα Σκύλος η οποία μπορεί να έχει ως στιγμιότυπα το Τεριέ, το Κανίς, το Λαμπραντόρ και άλλα [2].

Κάθε οντότητα περιγράφεται από ένα σύνολο από **γνωρίσματα (attributes) ή ιδιότητες.** Για παράδειγμα, η οντότητα Μαθητής έχει ως γνωρίσματα τον αριθμό μητρώου, το όνομα, το επίθετο, την διεύθυνση και το τμήμα. Όταν τα γνωρίσματα αυτά πάρουν συγκεκριμένες τιμές, τότε αναφερόμαστε στα στιγμιότυπα της αντίστοιχης οντότητας. Για παράδειγμα, ένας συγκεκριμένος μαθητής, δηλαδή ένα στιγμιότυπο της οντότητας Μαθητής, μπορεί να έχει ως αριθμό μητρώου το 4589543, ως όνομα το Ανδρέας, ως επίθετο το Παπαϊωάννου, ως διεύθυνση το Εγνατίας 12 και ως τμήμα το Β1. Δηλαδή, κάθε στιγμιότυπο μιας οντότητας έχει συγκεκριμένες **τιμές (values)** για τα αντίστοιχα γνωρίσματα της οντότητας [2].

Τα γνωρίσματα για τα οποία είναι υποχρεωτικό να έχουν τιμή, ονομάζονται **υποχρεωτικά γνωρίσματα (mandatory attributes)**. Για παράδειγμα, στις περισσότερες επιχειρήσεις είναιαπαραίτητο να είναι γνωστά τα ονοματεπώνυμα των ατόμων που διαδραματίζουν κάποιο ρόλο στις διαδικασίες της επιχείρησης. Άλλα γνωρίσματα, όπως είναι για παράδειγμα το κινητό τηλέφωνο, είναι προαιρετικό να έχουν μία τιμή, εκτός και αν πρόκειται για εταιρείες κινητής τηλεφωνίας. Τα γνωρίσματα αυτά λέγονται **προαιρετικά γνωρίσματα (optional attributes) [2].**
Στον παρακάτω πίνακα παρουσιάζονται παραδείγματα οντοτήτων και αντίστοιχων γνωρισμάτων.

Οντότητες	**Γνωρίσματα**
Αυτοκίνητο	Μοντέλο, Ιπποδύναμη, Τιμή
Εργασία	Τίτλος, Περιγραφή
Συναλλαγή	Ποσό, Ημερομηνία
Εργασία	Μηχανικός, Γιατρός
Υπάλληλος	Ονοματεπώνυμο, Διεύθυνση, Κινητό, Αριθμός Ταυτότητας, email

Εικόνα 13 - Παραδείγματα οντοτήτων και αντίστοιχων γνωρισμάτων

Αν τιμές που μπορεί να πάρει ένα γνώρισμα είναι μοναδικές για κάθε στιγμιότυπο, τότε το γνώρισμα αυτό λέγεται **πρωτεύον κλειδί (unique identifier)**. Για παράδειγμα, ο αριθμός μητρώου για την οντότητα Μαθητής αποτελεί το κλειδί της οντότητας αυτής, διότι δεν υπάρχουν δύο μαθητές με τον ίδιο αριθμό μητρώου.

4.3.2. Συσχετίσεις μεταξύ Οντοτήτων - Τύποι Συσχετίσεων

Ποια είναι η σχέση ανάμεσα σε εσάς και την θεία σας, τον θείο σας, τα ξαδέλφια σας, τη γιαγιά σας κλπ; Πώς θα μπορούσατε να εξηγήσετε σε κάποιον τρίτο ότι η Ιωάννα είναι ξαδέλφη σας; Είναι απαραίτητο για την κοινωνία μας να κατηγοριοποιεί τις σχέσεις μεταξύ των ανθρώπων ή της αρκεί να αναγνωρίζει τα άτομα μόνο με το όνομά τους [2];

Για παράδειγμα, φανταστείτε ότι θέλετε να συστήσετε σε κάποιον την ξαδέλφη σας την Ιωάννα, χωρίς την αναφορά της λέξης «ξαδέλφη» ή κάποια άλλη παρεμφερή. Θα λέγατε τότε «Αυτή είναι η Ιωάννα, το παιδί της γυναίκας που έχει τους ίδιους γονείς με τους γονείς μου». Πολύ μεγάλη εισαγωγή και κουραστική. Δεν θα ήταν προτιμότερο να πείτε «Αυτή είναι η Ιωάννα, η κόρη της θείας μου» ή πιο απλά ακόμη «Αυτή είναι η Ιωάννα, ξαδέλφη μου» [2];

Το παραπάνω καθημερινό παράδειγμα των οικογενειακών σχέσεων συναντάται και στον χώρο των επιχειρήσεων και των οργανισμών. Γενικά, οι **σχέσεις (relationships)** μας βοηθούν να δούμε πώς τα διάφορα μέρη ενός συστήματος επηρεάζουν το ένα το άλλο. Για παράδειγμα, οι οντότητες ΜΑΘΗΤΗΣ (STUDENT) και ΜΑΘΗΜΑ (COURSE) συνδέονται μεταξύ τους. Κάθε μαθητής μπορεί να επιλέξει, για παράδειγμα το μάθημα «Εφαρμογές Πληροφορικής». Για να μοντελοποιήσουμε με ακρίβεια την επιχείρηση ή τον οργανισμό, οι σχέσεις μεταξύ των οντοτήτων είναι εξίσου σημαντικές με τις ίδιες τις οντότητες [2].

Στη συνέχεια παραθέτουμε κάποια παραδείγματα σχέσεων μεταξύ διαφόρων οντοτήτων:

- Οι ΕΡΓΑΖΟΜΕΝΟΙ έχουν ΘΕΣΕΙΣ ΕΡΓΑΣΙΑΣ Οι ΘΕΣΕΙΣ ΕΡΓΑΣΙΑΣ κατέχονται από ΕΡΓΑΖΟΜΕΝΟΥΣ
- Τα ΠΡΟΪΟΝΤΑ κατατάσσονται σύμφωνα με τον ΤΥΠΟ ΤΟΥ ΠΡΟΪΟΝΤΟΣ Ο ΤΥΠΟΣ ΤΟΥ ΠΡΟΪΟΝΤΟΣ αποτελεί μια κατηγοριοποίηση για ένα ΠΡΟΪΟΝ
- Οι ΑΝΘΡΩΠΟΙ κάνουν ΚΡΑΤΗΣΕΙΣ ΕΙΣΙΤΗΡΙΩΝ Οι ΚΡΑΤΗΣΕΙΣ ΕΙΣΙΤΗΡΙΟΥ γίνονται από ΑΝΘΡΩΠΟΥΣ

Οι σχέσεις είναι χαρακτηρίζονται ως **υποχρεωτικές(mandatory)** ή **προαιρετικές(optional).** Θεωρήστε τις δύο οντότητες ΕΡΓΑΖΟΜΕΝΟΣ (EMPLOYEE) και ΘΕΣΗ ΕΡΓΑΣΙΑΣ (JOB). Για να μπορέσουμε να καθορίσουμε το είδος της σχέσης που διέπει τις παραπάνω οντότητες θα πρέπει να απαντήσουμε στα παρακάτω δύο ερωτήματα [2]:

- Πρέπει κάθε εργαζόμενος να έχει μια θέση εργασίας; Με άλλα λόγια, αυτό είναι ένα υποχρεωτικό ή προαιρετικό για κάθε εργαζόμενο;
- Πρέπει κάθε θέση εργασίας να αντιστοιχεί σε έναν υπάλληλο; Με άλλα λόγια, αυτό είναι

ένα υποχρεωτικό ή προαιρετικό για κάθε θέση εργασίας;

Σε μία σχέση ένα ή περισσότερα στιγμιότυπα μιας οντότητας μπορούν ή πρέπει να συσχετισθούν με ένα στιγμιότυπο της άλλης οντότητας. Το αν θα είναι ένα ή περισσότερα στιγμιότυπα εκφράζει αυτό που ονομάζουμε ως **πληθάριθμος (cardinality)** μιας σχέσης. Ο πληθάριθμος μιας σχέσης μετράει την ποσότητα από κάτι, απαντάει δηλαδή στην ερώτηση «Πόσα;». Για παράδειγμα:

- Σε πόσες θέσεις εργασίας μπορεί ένας υπάλληλος να εργάζεται; Σε μία θέση εργασίας μόνο ή σε περισσότερες από μία;
- Πόσοι υπάλληλοι μπορούν να κατέχουν μια συγκεκριμένη θέση εργασίας; Ένας υπάλληλος μόνο ή περισσότεροι από έναν εργαζόμενο;

Ο πληθάριθμος μιας σχέσης απαντάει στο ερώτημα αν το πλήθος είναι ένα ή περισσότερα από ένα. Δεν δίνει απάντηση με ένα συγκεκριμένο πλήθος για παράδειγμα τέσσερα (4). Οι παραπάνω ερωτήσεις είναι πολύ σημαντικές και έχουν αντίκτυπο στον σχεδιασμό της βάσης δεδομένων και συγκεκριμένα στο μοντέλο συσχετίσεων οντοτήτων. Στο προαναφερθέν παράδειγμα επιλέγουμε τις παρακάτω παραδοχές για τις οποίες θα δούμε στην επόμενη ενότητα πώς απεικονίζονται με διαγραμματικό τρόπο [2]:

- Κάθε εργαζόμενος **υποχρεωτικά (mandatory)** πρέπει να έχει **ακριβώς μία(cardinality)** θέση εργασίας.
- Μια θέση εργασίας **μπορεί (optionality**) να κατέχεται από **περισσότερους από έναν εργαζόμενους (cardinality)**. Δηλαδή για μία θέση εργασίας μπορεί να έχουμε κανέναν, έναν ή περισσότερους από έναν εργαζόμενους.

Γενικά, τα μοντέλα οντοτήτων συσχετίσεων χρησιμοποιούν τρεις τύπους σχέσεων:

- **Ένα προς πολλά 1:M**
- **Πολλά προς πολλά M:M**
- **Ένα προς ένα 1:1**

Ένα προς πολλά 1:M

Ένας ζωγράφος ζωγραφίζει πολλούς πίνακες, αλλά κάθε πίνακας αποτελεί δημιούργημα ενός μόνο ζωγράφου. Η σχέση «Ο ζωγράφος ζωγραφίζει πίνακες» είναι μία σχέση «ένα προς πολλά». Το «ένα» είναι από την πλευρά της οντότητας «ΖΩΓΡΑΦΟΣ», ενώ το «πολλά» είναι από την πλευρά της οντότητας «ΠΙΝΑΚΑΣ» [2].

Πολλά προς πολλά M:M

Ένας εργαζόμενος μπορεί να μάθει πολλές δεξιότητες και κάθε δεξιότητα μπορεί να κατακτηθεί από πολλούς εργαζόμενους. Η σχέση «Ο εργαζόμενος μαθαίνει μία δεξιότητας» είναι μία σχέση «πολλά προς πολλά». Το «πολλά» είναι και από την πλευρά της οντότητας «ΕΡΓΑΖΟΜΕΝΟΣ», αλλά και από την πλευρά της οντότητας «ΔΕΞΙΟΤΗΤΑ» [2].

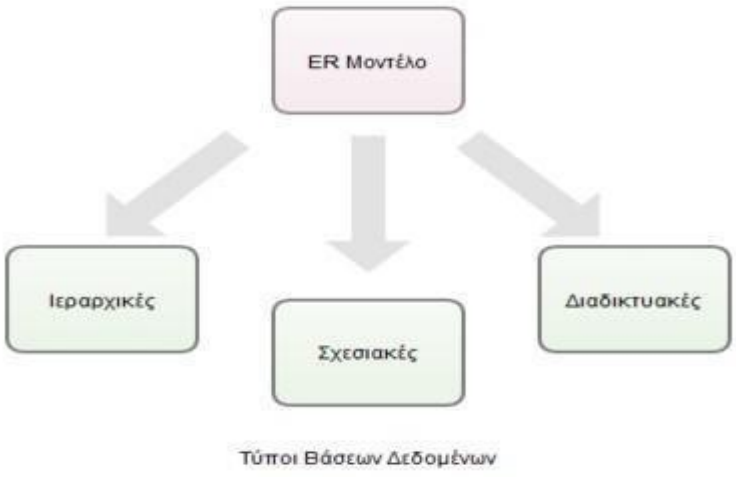
Ένα προς ένα 1:1

Μία εταιρεία λιανικού εμπορίου αναθέτει την διαχείριση για κάθε ένα από τα καταστήματά της σε ένα υπάλληλο. Κάθε υπεύθυνος καταστήματος διευθύνει ένα μόνο κατάστημα. Η σχέση «Ο εργαζόμενος διευθύνει ένα κατάστημα» είναι μία σχέση «ένα προς ένα». Το «ένα» είναι και από την πλευρά της οντότητας «ΕΡΓΑΖΟΜΕΝΟΣ», αλλά και από την πλευρά της οντότητας «ΚΑΤΑΣΤΗΜΑ» [2].

4.3.3 Μοντελοποίηση Συσχετίσεων Οντοτήτων και ER Διάγραμμα

Το **μοντέλο συσχετίσεων οντοτήτων(Entity Relationship model)** είναι ένα εννοιολογικό μοντέλο το οποίο χρησιμοποιείται ευρέως στον σχεδιασμό βάσεων δεδομένων για την κατανόηση και αναπαράσταση των απαιτήσεων δεδομένων μιας επιχείρησης ή ενός οργανισμού [2].

Η **μοντελοποίηση συσχετίσεων οντοτήτων (Entity Relationship modeling)** είναι ανεξάρτητη από το υλικό ή το λογισμικό που χρησιμοποιείται ή θα χρησιμοποιηθεί στην υλοποίηση, όπως φαίνεται στο παρακάτω σχήμα. Παρόλο που το μοντέλο συσχετίσεων οντοτήτων (Entity Relationship Model) αποτελεί τον θεμελιώδη λίθο για την δημιουργία κάθε τύπου βάσης δεδομένων, όπως των ιεραρχικών (hierarchical databases), των διαδικτυακών (network databases) και των σχεσιακών βάσεων δεδομένων (relational databases) συνδέεται περισσότερο όμως με τις σχεσιακές βάσεις δεδομένων [2].



Εικόνα 14 - Το μοντέλο οντοτήτων συσχετίσεων είναι ανεξάρτητο από την υλοποίηση

Ένα **μοντέλο συσχετίσεων οντοτήτων (ERM)** είναι μία λίστα οντοτήτων καθώς και σχέσεων μεταξύ των οντοτήτων αυτών. Παρέχει βασικές πληροφορίες, όπως περιγραφές οντοτήτων, τύπους δεδομένων και περιορισμούς. Το **διάγραμμα συσχετίσεων οντοτήτων (ERD)** αποτελεί γραφική αναπαράσταση του μοντέλου συσχετίσεων οντοτήτων. Χρησιμοποιεί μία σειρά από γραφικά στοιχεία τα οποία θα μελετήσουμε στη συνέχεια. Τα γραφικά αυτά στοιχεία διαφέρουν σε ορισμένα σημεία από τα αντίστοιχα που παρουσιάστηκαν στις προηγούμενες ενότητες, όπως είναι αυτό της αναπαράστασης της συσχέτισης. Και στις δύο όμως περιπτώσεις η μεθοδολογία και η προσέγγιση παραμένει η ίδια [2].

Απεικόνιση Οντότητας

Στο ER διάγραμμα οι οντότητες αναπαρίστανται από πλαίσια μέσα στα οποία αναγράφεται το όνομα της οντότητας, όπως φαίνεται στο παρακάτω σχήμα [2]:

Εικόνα 15 - Απεικόνιση οντοτήτων σε ERδιάγραμμα

Απεικόνιση Ιδιοτήτων - Γνωρισμάτων Οντότητας

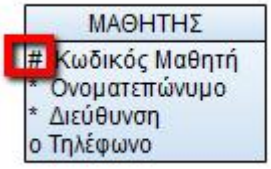
Τα γνωρίσματα απεικονίζονται μέσα στο πλαίσιο της οντότητας, όπως φαίνεται στο παρακάτω σχήμα:

Εικόνα 16 - Απεικόνιση ιδιοτήτων-γνωρισμάτων οντοτήτων σε ERδιάγραμμα

Τα σύμβολα * και **o** μπροστά από τα γνωρίσματα υποδηλώνουν την αναγκαιότητα ή μη ύπαρξης τιμής αντίστοιχα στα γνωρίσματα, όταν δημιουργούνται τα στιγμιότυπα της οντότητας. Για παράδειγμα, όταν εισάγεται στο πληροφοριακό σύστημα ένας νέος εργαζόμενος θα πρέπει να είναι γνωστό το ονοματεπώνυμό του και η διεύθυνσή του. Η ημερομηνία γέννησης και το email του μπορούν να συμπληρωθούν και αργότερα. Οι ανάγκες της επιχείρησης απαιτούν την καταγραφή του ονοματεπώνυμού και της διεύθυνσης του εργαζόμενου με την εισαγωγή ενός νέου εργαζομένου [2].

Απεικόνιση Κλειδιού

Εκείνο το γνώρισμα σε μία οντότητα που έχει τον ρόλο του κλειδιού, δηλαδή προσδιορίζει μονοσήμαντα την οντότητα στο ER διάγραμμα σημειώνεται με το σύμβολο «#», όπως φαίνεται στο παρακάτω σχήμα [2]:



Εικόνα 17 - Απεικόνιση κλειδιού οντότητας σε ERδιάγραμμα

Απεικόνιση Συσχετίσεων Οντοτήτων

Οι συσχετίσεις ή σχέσεις μεταξύ των οντοτήτων απεικονίζονται από μια γραμμή, που συνδέει τις δύο οντότητες. Στα δύο άκρα της γραμμής αναγράφονται οι σχέσεις μεταξύ των οντοτήτων, δηλαδή πώς «βλέπει» η κάθε οντότητα τον εαυτό της σε σχέση με την άλλη οντότητα. Στο παρακάτω σχήμα απεικονίζονται οι εξής σχέσεις [2]:

- Κάθε εργαζόμενος έχει μία θέση εργασίας
- Κάθε θέση εργασίας απασχολεί εργαζόμενους

Εικόνα 18 - Απεικόνιση συσχετίσεων οντοτήτων από το χώρο των επιχειρήσεων

Απεικόνιση Πληθάριθμου Σχέσης

Οι σχέσεις μεταξύ των οντοτήτων είναι «ένα» ή «πολλά». Το «ένα» σημαίνει ακριβώς ένα. Τα «πολλά» σημαίνουν ένα ή περισσότερα. Το «ένα» απεικονίζεται με ένα ευθύγραμμο τμήμα στην «απέναντι» οντότητα. Τα «πολλά» απεικονίζεται με ένα τρίποδο στην «απέναντι» οντότητα. Το παράδειγμα που απεικονίζεται στο παρακάτω σχήμα διαβάζεται ως εξής [2]:

- Κάθε εργαζόμενος κατέχει ακριβώς μία θέση εργασίας.

Εικόνα 19 - Απεικόνιση πληθάριθμου σχέσης «ένα»

Παρόμοια, το παράδειγμα που απεικονίζεται στο παρακάτω σχήμα διαβάζεται ως εξής:

- Σε κάθε θέση εργασίας εργάζονται περισσότεροι από ένας εργαζόμενοι.

Εικόνα 20 - Απεικόνιση πληθάριθμου σχέσης «πολλά»

Απεικόνιση Υποχρεωτικής/Προαιρετικής Σχέσης

Οι σχέσεις, όπως έχουμε δει, μπορεί να είναι υποχρεωτικές ή προαιρετικές, όπως συμβαίνει και με τα γνωρίσματα. Οι υποχρεωτικές σχέσεις απεικονίζονται από μία συμπαγή γραμμή, ενώ οι προαιρετικές από μία διακεκομμένη γραμμή. Το παρακάτω σχήμα διαβάζεται ως εξής [2]:

- Κάθε εργαζόμενος έχει οπωσδήποτε μία θέση εργασίας.
- Κάθε εργασίας μπορεί να απασχολεί εργαζόμενους. Δεν είναι υποχρεωτικό δηλαδή για μία θέση εργασίας να υπάρχουν οπωσδήποτε εργαζόμενοι».

Εικόνα 21 - Υποχρεωτικές και προαιρετικές συσχετίσεις από το χώρο των επιχειρήσεων

Παρόμοια παραδείγματα συναντάμε και στον σχολικό χώρο με τα μαθήματα επιλογής. Κάθε μαθητής υποχρεούται να πάρει ένα μάθημα επιλογής. Υπάρχουν όμως μαθήματα επιλογής που δεν τα έχει επιλέξει κανείς. Η περίπτωση αυτή μπορεί να αναπαρασταθεί από το παρακάτω σχήμα [2]:

Εικόνα 22 - Υποχρεωτικές και προαιρετικές συσχετίσεις από τον σχολικό χώρο

Ολοκληρωμένη Απεικόνιση Συσχετίσεων Οντοτήτων

Αν συνδυάσουμε όλα τα παραπάνω σε ένα διάγραμμα θα πάρουμε το παρακάτω σχήμα, το οποίο διαβάζεται ως εξής [2]:

- Κάθε εργαζόμενος υποχρεωτικά (mandatory) πρέπει να έχει ακριβώς μία (cardinality) θέση εργασίας.
- Μια θέση εργασίας μπορεί (optionality) να κατέχεται από περισσότερους από έναν εργαζόμενους (cardinality). Δηλαδή για μία θέση εργασίας μπορεί να έχουμε κανέναν, έναν ή περισσότερους από έναν εργαζόμενους.

Εικόνα 23- ERδιάγραμμα

4.3.4 Επίλυση Συσχετίσεων Πολλά προς Πολλά

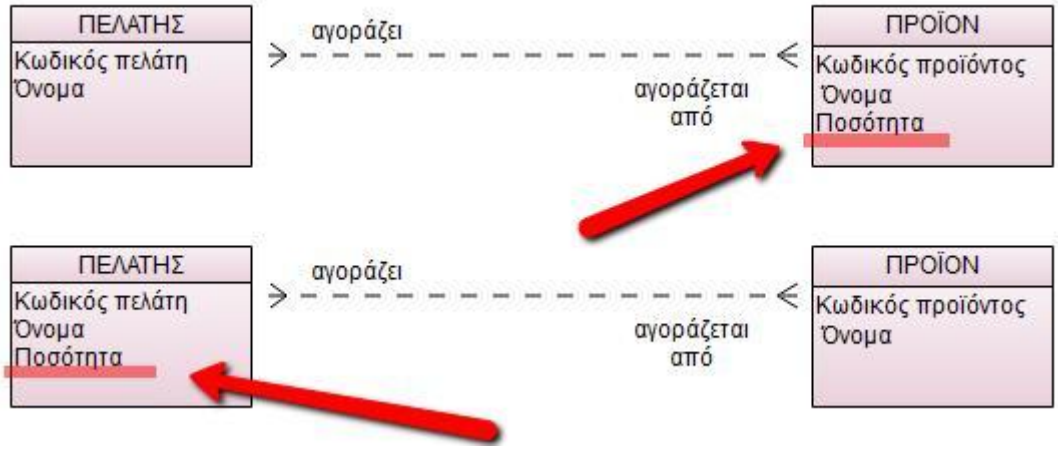
Κατά την διαδικασία της μοντελοποίησης είναι δυνατό να ορίσετε αρκετές σχέσεις μεταξύ των οντοτήτων του τύπου «πολλά προς πολλά» (M:M). Οι περισσότερες από αυτές τις σχέσεις «πολλά προς πολλά» «κρύβουν» μέσα τους οντότητες τις οποίες θα χρειαστούμε στη συνέχεια για να τις προσδώσουμε συγκεκριμένα γνωρίσματα. Γνωρίσματα τα οποία απαραίτητα για μία πληρέστερη εννοιολογική απόδοση των αναγκών μίας επιχείρησης ή ενός οργανισμού [2].

Ένα τυπικό παράδειγμα, αποτελεί αυτό της σχέσης μεταξύ ΠΕΛΑΤΗ και ΠΡΟΪΟΝ, όπως παρουσιάζεται στο παρακάτω διάγραμμα οντοτήτων-συσχετίσεων. Μία εταιρεία λιανικού εμπορίου πουλάει διάφορα προϊόντα σε πελάτες της. Ας υποθέσουμε ότι γίνονται δεκτοί στο σύστημα και μελλοντικοί πελάτες. Το ER διάγραμμα διαβάζεται ως εξής [2]:

- Ένας πελάτης μπορεί να αγοράσει ένα ή περισσότερα προϊόντα.
- Ένα προϊόν μπορεί να αγοραστεί από έναν ή περισσότερους πελάτες.

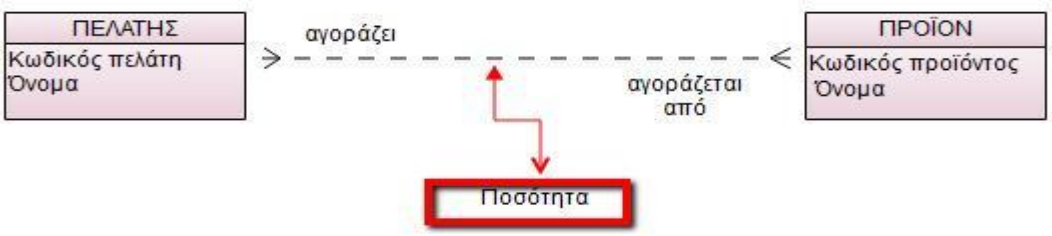
Εικόνα 24 - Οι σχέσεις «πολλά προς πολλά» κάτι κρύβουν

Για παράδειγμα, ο πελάτης Νίκος Χατζηπέτρου αγοράζει δύο μπλούζες. Το όνομα «Νίκος Χατζηπέτρου» είναι το όνομα του πελάτη και «μπλούζα» είναι το όνομα του προϊόντος. Τίθεται το εξής ερώτημα: Η ποσότητα 2 ως πληροφορία σε ποια από τις δύο οντότητες θα «αποθηκευτεί»; Στον ΠΕΛΑΤΗ ή στο ΠΡΟΪΟΝ [2];



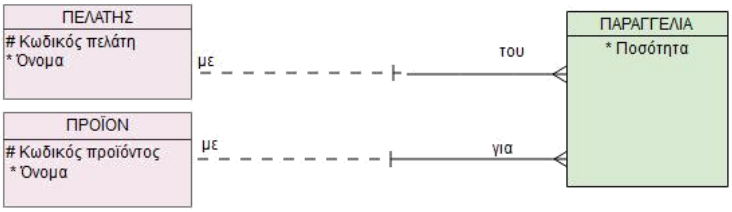
Εικόνα 25 - Η ποσότητα είναι γνώρισμα της...

Είναι ξεκάθαρο ότι η πληροφορία «Ποσότητα» δεν αποτελεί γνώρισμα ούτε της οντότητας ΠΕΛΑΤΗ αλλά ούτε και της οντότητας ΠΡΟΪΟΝ. Περισσότερο φαίνεται να είναι η πληροφορία «Ποσότητα» γνώρισμα της σχέσης που συνδέει τον ΠΕΛΑΤΗ με το ΠΡΟΪΟΝ [2].



Εικόνα 26 - Η ποσότητα είναι γνώρισμα της σχέσης

Οι συσχετίσεις όμως δεν έχουν και δεν μπορούν να έχουν γνωρίσματα. Προφανώς λείπει μία οντότητα της οποίας ένα γνώρισμα είναι η «Ποσότητα». Εισάγεται στο ER διάγραμμα η οντότητα «ΠΑΡΑΓΓΕΛΙΑ», την οποία ονομάζουμε οντότητα «τομής» (intersection entity) [2].



Εικόνα 27 - Επίλυση συσχέτισης «πολλά προς πολλά». Εισάγουμαι μια καινούργια οντότητα η «Παραγγελία»

Με την δημιουργία μία τρίτης βοηθητικής οντότητας, όπως είναι η οντότητα «ΠΑΡΑΓΓΕΛΙΑ» στο προηγούμενο παράδειγμα, επιλύεται το πρόβλημα των συσχετίσεων «πολλά προς πολλά». Η συσχέτιση «πολλά προς πολλά» παραχωρεί την θέση της σε δύο συσχετίσεις «ένα προς πολλά». Το μοναδικό αναγνωριστικό ή κλειδί της τρίτης οντότητας προκύπτει από τον συνδυασμό των κλειδιών των δύο αρχικών οντοτήτων. Τώρα, στο ER διάγραμμα οι συσχετίσεις που ενώνουν τις δύο αρχικές οντότητες με την τρίτη βοηθητική οντότητα είναι «ένα προς πολλά» [2].

4.4 Μοναδικά Κλειδιά και Κανονικοποίηση (Normalization)

Ένα βασικό θέμα στον σχεδιασμό μιας βάσης δεδομένων είναι ο προσδιορισμός των κλειδιών με τα οποία γίνεται η αναζήτηση πληροφοριών στη βάση δεδομένων. Τα κλειδιά μπορεί να είναι πρωτεύοντα (μοναδικά για κάθε εγγραφή), δευτερεύοντα (για παράδειγμα το επίθετο ενός υπαλλήλου, μαθητή) ή σύνθετα (ένωση δύο ή και περισσότερων χαρακτηριστικών). Στην παρούσα ενότητα θα αναφερθούμε στον προσδιορισμό των κλειδιών μέσω παραδειγμάτων [2].

Ένα δεύτερο θέμα που θα μας απασχολήσει στην ενότητα αυτή, είναι η **κανονικοποίηση** (normalization), που αποτελεί ένα πολύ σημαντικό βήμα στον σχεδιασμό μίας βάσης δεδομένων. Οι πίνακες που προκύπτουν μετά τη μετατροπή του μοντέλου ΟΣ σε σχεσιακό, όπως αναφέρθηκε στην ενότητα 3.2, περιέχουν ως συνήθως πληροφορίες που επαναλαμβάνονται.. Η επαναλαμβανόμενες πληροφορίες οδηγούν σε πολλαπλές γραμμές στον σχεδιασμό του αντίστοιχου πίνακα, γεγονός που

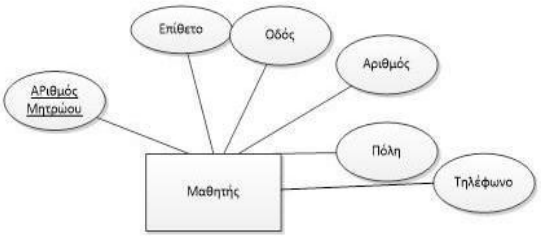
οδηγεί σε σημαντικά προβλήματα. Η κανονικοποίηση σχέσεων αφορά στην απλοποίηση των σχέσεων με στόχο την εξάλειψη της επαναληπτικότητας των δεδομένων.

4.4.1 Πρωτεύον, Σύνθετο και Δευτερεύον κλειδί

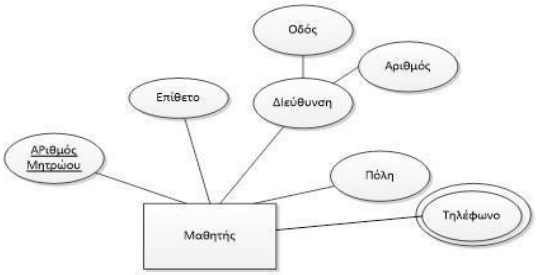
Κάθε οντότητα έχει ένα σύνολο χαρακτηριστικών/ ιδιοτήτων. Αναλυτικά, έχουμε τους εξής τύπους χαρακτηριστικών:

Απλό ή **Σύνθετο.**

Ένα απλό χαρακτηριστικό δεν μπορεί να διαχωριστεί σε μικρότερα τμήματα, ενώ τα σύνθετα χαρακτηριστικά μπορούν να χωριστούν σε μικρότερα τμήματα σε άλλα απλούστερα χαρακτηριστικά. Η χρήση συνθέτων χαρακτηριστικών είναι καλύτερα να γίνεται όταν επιθυμούμε σε κάποιες περιπτώσεις να αναφερόμαστε σε ολόκληρο το χαρακτηριστικό, ενώ σε κάποιες άλλες περιπτώσεις επιθυμούμε να αναφερόμαστε στα επιμέρους τμήματά του. Για παράδειγμα στην εικόνα 28 3.4.1 , θα μπορούσαμε να αντικαταστήσουμε τα χαρακτηριστικά οδός και αριθμός του συνόλου οντοτήτων Μαθητής με το σύνθετο χαρακτηριστικό διεύθυνση όπως παρουσιάζεται στην εικόνα 29. Τα σύνθετα χαρακτηριστικά μας βοηθούν να συγκεντρώσουμε συσχετιζόμενα χαρακτηριστικά σε ομάδες, διευκολύνοντας το σχεδιασμό της Βάσης Δεδομένων [2].



Εικόνα 28 - Οντότητα μαθητής



Εικόνα 29- Οντότητα μαθητή

Απλής Τιμής ή Πολλαπλής Τιμής.

Τα χαρακτηριστικά απλής τιμής έχουν μία μόνο τιμή. Για παράδειγμα, το σύνολο Μαθητής έχει το χαρακτηριστικό επίθετο το οποίο είναι ένα για κάθε οντότητα . Υπάρχουν όμως χαρακτηριστικά με δυο ή περισσότερες τιμές. Αυτά ονομάζονται χαρακτηριστικά πολλαπλής τιμής.

Για παράδειγμα, θεωρείστε το σύνολο Μαθητής το οποίο έχει το χαρακτηριστικό τηλέφωνο. Το χαρακτηριστικό τηλέφωνο μπορεί να έχει δύο ή και περισσότερες τιμές εφόσον ο μαθητής μπορεί να έχει σταθερό και κινητό τηλέφωνο. Στα χαρακτηριστικά πολλαπλής τιμής μπορούμε να ορίσουμε κατώτερο και ανώτερο όριο τιμών. Για παράδειγμα, μπορούμε να περιορίσουμε τον αριθμό των τηλεφωνικών αριθμών που επιθυμούμε να καταγράψουμε σε δύο μόνο (π.χ. ένα αριθμό τηλεφώνου σπιτιού και ένα για κινητό). Τα χαρακτηριστικά αυτά αναπαριστώνται με διπλή έλλειψη σε ένα διάγραμμα οντοτήτων συσχετίσεων [2].

Κενό.

Μια **κενή** τιμή χρησιμοποιείται όταν κάποια οντότητα δεν έχει τιμή για κάποιο χαρακτηριστικό. Για παράδειγμα, εάν κάποιος μαθητής δηλώσει ότι δεν έχει τηλέφωνο, τότε η τιμή του χαρακτηριστικού Τηλέφωνο για τον συγκεκριμένο μαθητή θα είναι Κενό.

Ένα χαρακτηριστικό/ιδιότητα ενός πίνακα ονομάζεται **πρωτεύον κλειδί** εάν μπορεί να διαχωρίζει τις διαφορετικές γραμμές του πίνακα. Για παράδειγμα, ο αριθμός αστυνομικής ταυτότητας ή ο αριθμός φορολογικού μητρώου μπορούν να θεωρηθούν ως πρωτεύοντα κλειδιά για τον πίνακα Υπάλληλος εφόσον είναι μοναδικά για κάθε υπάλληλο. Αντίθετα, το επίθετο δε θεωρείται πρωτεύον κλειδί, διότι μπορεί να υπάρχουν πολλοί υπάλληλοι που να έχουν το ίδιο επίθετο (για παράδειγμα το επίθετο Παπαδόπουλος είναι κοινό και μπορεί να το έχουν πολλοί υπάλληλοι μιας εταιρείας ή ενός οργανισμού) [2].

Ένας πίνακας μπορεί να έχει περισσότερα από ένα πρωτεύοντα κλειδιά (για παράδειγμα ο αριθμός αστυνομικής ταυτότητας και ο αριθμός φορολογικού μητρώου). Ωστόσο, ένα μόνο επιλέγεται να χρησιμοποιηθεί. Το κλειδί που επιλέγεται καλείται **πρωτεύον κλειδί(primarykey)** ενώ τα υπόλοιπα καλούνται **δευτερεύοντα (secondary).** Ως δευτερεύον κλειδί στον πίνακαυπάλληλος μπορεί να επιλεχθεί το επίθετο έτσι ώστε να διευκολύνεται ο χρήστης στην αναζήτηση κάποιου υπαλλήλου με τη χρήση του επίθετου. Σε πολλές περιπτώσεις το κλειδί ενός πίνακα μπορεί να είναι **σύνθετο (composite**) δηλαδή να αποτελείται από δύο ή και περισσότερα χαρακτηριστικά. Για παράδειγμα, στον πίνακα 3.4.3 παρουσιάζεται ένα σύνθετο κλειδί αποτελούμενο από τα χαρακτηριστικά, Επίθετο, ¨Όνομα και ¨Όνομα Πατρός. Με αυτόν τον τρόπο διαχωρίζονται οι γραμμές του πίνακα και εύκολα ξεχωρίζει η εγγραφή Παπαδόπουλος Ιωάννης του Ιωάννη από την εγγραφή Παπαδόπουλος Ιωάννης του Χρήστου [2].

Επίθετο	Όνομα	Όνομα Πατρός	Διεύθυνση	Πόλη
Παπαδόπουλος	Ιωάννης	Δημήτριος	Βενιζέλου 4	Θεσσαλονίκη
Παπαδόπουλος	Ιωάννης	Χρήστος	Τσιμισκή 19	Θεσσαλονίκη
Χαρισίδης	Νικόλαος	Μενέλαος	Εγνατία 151	Θεσσαλονίκη
Κερκινάκης	Δημήτριος	Γεώργιος		

Εικόνα30 - Σύνθετο κλειδί

4.4.2 Κανονικοποίηση και Πρώτη Κανονική Μορφή

Η επαναληπτικότητα των δεδομένων δημιουργεί προβλήματα στους πίνακες που προκύπτουν από το σχεσιακό μοντέλο. Αυτό καθιστούν ανεπιθύμητη την επαναληπτικότητα για τους εξής κύριους λόγους [2] :

Πρόβλημα Ενημέρωσης Δεδομένων.

Αν χρειαστεί να γίνει αλλαγή των δεδομένων κατά τη χρήση της ΒΔ, αυτά πρέπει να αλλάξουν όπου εμφανίζονται. Για παράδειγμα, η αλλαγή διεύθυνσης ενός μαθητή οδηγεί σε ενημέρωση δύο γραμμών του πίνακα Μαθητής (Εικόνα 3.4.4). Οι ενημερώσεις, δημιουργούν καθυστερήσεις και αν για κάποιους λόγους δεν ενημερωθούν όλες οι γραμμές του πίνακα οδηγούμαστε σε ασυνέπεια δεδομένων (άλλες γραμμές να έχουν τη νέα διεύθυνση και άλλες την παλιά) [2].

Πρόβλημα Διαγραφής Δεδομένων.

Αν για κάποιους λόγους, ένας μαθητής καταργήσει όλους τους τηλεφωνικούς του αριθμούς, δηλώνοντας ότι δεν έχει τηλέφωνο, τι μπορούμε να κάνουμε [2];

Οι εναλλακτικές λύσεις που έχουμε αλλά καμία δεν μπορεί να εφαρμοστεί είναι οι εξής:

- να θέσουμε κενό (NULL) σε όλες τις τιμές της στήλης τηλέφωνο, κάτι το οποίο δεν μπορεί να γίνει διότι η στήλη μπορεί να συμμετέχει στο πρωτεύον κλειδί του πίνακα, και
- να διαγράψουμε τις γραμμές του πίνακα, που επίσης δεν μπορεί να γίνει διότι θα χάσουμε τις υπόλοιπες πληροφορίες για το συγκεκριμένο μαθητή.

Πρόβλημα Εισαγωγής Δεδομένων.

Κάθε φορά που εισάγουμε στη βάση δεδομένων έναν νέο μαθητή, πρέπει να εισάγουμε και κάποιον αριθμό τηλεφώνου, κάτι το οποίο δεν είναι απαραίτητο να γίνει στην αρχή [2].

ΑΜ Μαθητή	Επίθετο	Διεύθυνση	Πόλη	Τηλέφωνο
14355	Παπαδόπουλος	Βενιζέλου 4	Θεσσαλονίκη	23105555555
14355	Παπαδόπουλος	Βενιζέλου 4	Θεσσαλονίκη	69999999999
14582	Χαρισίδης	Τσιμισκή 19	Θεσσαλονίκη	23106666666
14695	Κερκινάκης	Εγνατία 151	Θεσσαλονίκη	23107777777

Εικόνα 31 - Πίνακας με επαναλαμβανόμενες πληροφορίες

Η λύση που υπάρχει για την επίλυση των παραπάνω προβλημάτων είναι η κατασκευή δύο διαφορετικών πινάκων. Στην παραπάνω εικόνα 31 παρουσιάζεται ο πίνακας Μαθητής με επαναλαμβανόμενες πληροφορίες λόγω της ύπαρξης του χαρακτηριστικού πολλαπλής τιμής Τηλέφωνο Μετά την κατασκευή δύο πινάκων, η νέα μορφή της βάσης δεδομένων που προκύπτει παρουσιάζεται στην εικόνα 32 [2].

ΑΜ Μαθητή	Επίθετο	Διεύθυνση	Πόλη
14355	Παπαδόπουλος	Βενιζέλου 4	Θεσσαλονίκη
14582	Χαρισίδης	Τσιμισκή 19	Θεσσαλονίκη
14695	Κερκινάκης	Εγνατία 151	Θεσσαλονίκη

ΑΜ Μαθητή	Τηλέφωνο
14355	23105555555
14355	69999999999
14582	23106666666
14695	23107777777

Εικόνα 32 - Κατασκευή δύο πινάκων για την αποφυγή της επαναληπτηκότητας των δεδομένων

Βασικός στόχος της κανονικοποίησης είναι ο προσδιορισμός ενός συνόλου πινάκων με στόχο την ελαχιστοποίηση της επαναληπτικότητας δεδομένων. Με τη μέθοδο της κανονικοποίησης πραγματοποιούνται διαδοχικές διασπάσεις των πινάκων σε πιο

απλές μορφές. Η διάσπαση ακολουθεί κάποιους κανόνες. Από τον αρχικό πίνακα απομακρύνονται κάποια χαρακτηριστικά και τοποθετούνται σε νέους πίνακες ώστε να είναι δυνατή η ανασύνδεση των αρχικών πληροφοριών μέσω των νέων σχέσεων [2].

Έχουν προταθεί πέντε κανονικές μορφές που όμως δεν θα αναλυθούν στην παρούσα ενότητα εκτός από την πρώτη κανονική μορφή.

Πρώτη Κανονική Μορφή (1NF)

Ένας πίνακας βρίσκεται στην πρώτη κανονική μορφή όταν οι τιμές που λαμβάνει κάθε στήλη του πίνακα είναι ατομικές. Επομένως, η πρώτη κανονική μορφή απαγορεύει την ύπαρξη στηλών οι οποίες περιέχουν χαρακτηριστικά πολλαπλής τιμής. Υπάρχουν δύο τρόποι για να μετασχηματίσουμε έναν πίνακα που περιέχει πολλαπλές τιμές στην πρώτη κανονική μορφή [2].

Ο ένας τρόπος είναι να επαναλάβουμε τα δεδομένα πολλές φορές, έτσι ώστε τελικά ο πίνακας που θα προκύψει να περιέχει μόνο ατομικές τιμές στις στήλες. Ο δεύτερος τρόπος είναι να χρησιμοποιήσουμε ξεχωριστό πίνακα για τα χαρακτηριστικά πολλαπλών τιμών. Και οι δύο τρόποι θεωρούνται σωστοί, όμως ο δεύτερος μειώνει σημαντικά την επαναληπτικότητα των δεδομένων [2].

Παράδειγμα πρώτης κανονικής μορφής

Θεωρείστε ότι σχεδιάζεται το Μαθητολόγιο του σχολείου σας. Κάθε μαθητής παρακολουθεί μαθήματα ανάλογα με την τάξη. Για κάθε μαθητή θα υπάρχουν τα στοιχεία του καθώς και τα μαθήματα τα οποία παρακολουθεί [2].

ΑΜ Μαθητή	Επίθετο	¨Όνομα	Όνομα Πατρός	Διεύθυνση	Μαθήματα
14355	Παπαδόπουλος	Ιωάννης	Δημήτριος	Βενιζέλου 4	ΣΔΒΔ
14666	Παπαδόπουλος	Ιωάννης	Χρήστος	Εθν. Αμύνης 8	Μαθηματικά
14582	Χαρισίδης	Νικόλαος	Μενέλαος	Τσιμισκή 19	Φυσική Αγωγή
14695	Κερκινάκης	Δημήτριος	Γεώργιος	Εγνατία 151	Φυσική

Εικόνα 33 - Πίνακας πρώτης κανονικής μορφής

Παρατηρούμε ότι το χαρακτηριστικό Μαθήματα είναι πολλαπλής τιμής και θα αναγκαστούμε να διασπάσουμε τον πίνακα ώστε να κανονικοποιηθεί [2].

ΑΜ Μαθητή	Επίθετο	¨Όνομα	Όνομα Πατρός	Διεύθυνση
14355	Παπαδόπουλος	Ιωάννης	Δημήτριος	Βενιζέλου 4
14666	Παπαδόπουλος	Ιωάννης	Χρήστος	Εθν. Αμύνης 8
14582	Χαρισίδης	Νικόλαος	Μενέλαος	Τσιμισκή 19
14695	Κερκινάκης	Δημήτριος	Γεώργιος	Εγνατία 151

ΑΜ Μαθητή	Μαθήματα
14355	ΣΔΒΔ
14355	Μαθηματικά
14355	Φυσική Αγωγή κ.α
14355	ΣΔΒΔ
14666	Νέα Ελληνικά
14582	Φυσική
14695	Μαθηματικά

Εικόνα 34 - Διάσπαση πίνακα για να κανονικοποιηθεί

4.5 Μετασχηματισμός από το Εννοιολογικό Μοντέλο στο Σχεσιακό Μοντέλο

Το επόμενο βήμα στον σχεδιασμό των βάσεων δεδομένων, πριν την φυσική υλοποίηση, αποτελεί ο μετασχηματισμός του μοντέλου οντοτήτων συσχετίσεων στο σχεσιακό μοντέλο. Αυτό σημαίνει ότι τα «αντικείμενα» του ER διαγράμματος που γνωρίσαμε στις προηγούμενες υποενότητες θα μετατραπούν στα αντίστοιχα «αντικείμενα» του σχεσιακού μοντέλου [2].

Αυτή η ενότητα περιγράφει μερικές βασικές αρχές των σχεσιακών βάσεων δεδομένων (relational databases) και παρουσιάζει διάφορες τεχνικές που μπορούμε να χρησιμοποιήσουμε για να μετατρέψουμε το μοντέλο οντοτήτων συσχετίσεων (Entity Relationship model) στο σχεσιακό μοντέλο [2].

4.5.1 Εισαγωγή στις έννοιες των Σχεσιακών Βάσεων Δεδομένων

Το επόμενο βήμα στον σχεδιασμό των βάσεων δεδομένων αποτελεί ο μετασχηματισμός του μοντέλου οντοτήτων συσχετίσεων στο σχεσιακό μοντέλο. Αυτό σημαίνει ότι οι οντότητες, τα γνωρίσματα, οι συσχετίσεις και τα μοναδικά αναγνωριστικά του ER διαγράμματος που γνωρίσαμε στις προηγούμενες υπό-ενότητες θα μετατραπούν στα αντίστοιχα «αντικείμενα» του σχεσιακού μοντέλου [2].

Θεωρείστε τον μετασχηματισμό αυτόν παρόμοιο με αυτόν που ακολουθεί ο σχεδιαστής ρούχων όταν με βάση το σχέδιο του στο χαρτί δημιουργεί το αντίστοιχο ρούχο από ύφασμα. Η μετατροπή αυτή απαιτεί γνώσεις ραπτικής από την πλευρά του σχεδιαστή. Έτσι και εμείς στις επόμενες υπό-ενότητες θα εφοδιαστούμε με γνώσεις σχετικές με την δομή του σχεσιακού μοντέλου. Το σχεσιακό μοντέλο αποτελεί ένα βήμα πριν την υλοποίηση [2].

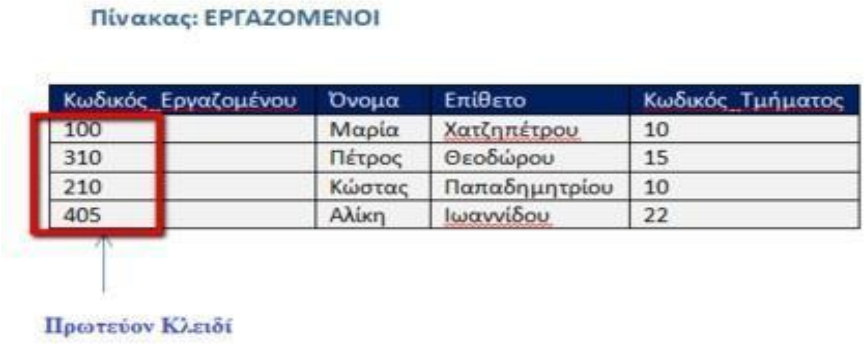
Ας μελετήσουμε τώρα την δομή του σχεσιακού μοντέλου. Ένα σχεσιακό μοντέλο μπορεί να ειδωθεί ως ένα σύνολο από δισδιάστατους πίνακες. Στους δισδιάστατους αυτούς πίνακες οργανώνονται και αποθηκεύονται τα δεδομένα. Στο παρακάτω σχήμα παρουσιάζεται ο πίνακας ΕΡΓΑΖΟΜΕΝΟΙ, όπου αποθηκεύονται πληροφορίες σχετικά με τους εργαζόμενους [2].

Κωδικός_Εργαζομένου	Όνομα	Επίθετο	Κωδικός_Τμήματος
100	Μαρία	Χατζηπέτρου	10
310	Πέτρος	Θεοδώρου	15
210	Κώστας	Παπαδημητρίου	10
405	Αλίκη	Ιωαννίδου	22

Εικόνα 35 - Πίνακας εργαζόμενοι

Κάθε πίνακας περιέχει γραμμές και στήλες. Κάθε γραμμή περιγράφει και έναν εργαζόμενο ή ένα στιγμιότυπο της οντότητας ΕΡΓΑΖΟΜΕΝΟΣ. Οι γραμμές του πίνακα είναι τέσσερις. Όσοι και οι εργαζόμενοι. Κάθε στήλη χρησιμεύει για να αποθηκεύει συγκεκριμένο τύπο τιμών όπως είναι ο κωδικός του εργαζόμενου, το ονοματεπώνυμό του και ο κωδικός του τμήματος στο οποίο εργάζεται [2].

Το **πρωτεύον κλειδί (primary key (PK))** είναι μία στήλη ή ένας συνδυασμός από στήλες που προσδιορίζει με μοναδικό τρόπο κάθε γραμμή του πίνακα. Στο παρακάτω σχήμα, το πρωτεύον κλειδί του πίνακα ΕΡΓΑΖΟΜΕΝΟΙ είναι ο κωδικός του εργαζόμενου δηλαδή η στήλη Κωδικός_Εργαζομένου. Αυτό σημαίνει ότι δεν μπορούμε να έχουμε δύο εργαζόμενους με τον ίδιο κωδικό. Κάθε εργαζόμενος προσδιορίζεται από ένα μη μηδενικό κωδικό ο οποίος είναι μοναδικός για τον εργαζόμενο αυτόν [2].



Εικόνα 36 - Το πρωτεύον κλειδί του πίνακα εργαζόμενοι είναι η στήλη κωδικός_εργαζόμενου

Αντίθετα στον πίνακα ΛΟΓΑΡΙΑΣΜΟΙ το πρωτεύον κλειδί είναι ο συνδυασμός του κωδικού της τράπεζας και του κωδικού του λογαριασμού. Στην περίπτωση αυτή, δεν αρκεί μόνο μία στήλη για να προσδιοριστεί μοναδικά κάθε λογαριασμός. Διότι είναι δυνατό σε δύο διαφορετικές τράπεζες όπως συμβαίνει στην πρώτη και πέμπτη γραμμή του πίνακα, ο κωδικός λογαριασμού να είναι ο ίδιος. Άρα, χρειαζόμαστε και μία δεύτερη στήλη για να προσδιορίσουμε το πρωτεύον κλειδί. Αυτή η δεύτερη στήλη είναι ο κωδικός της τράπεζας [2].

Εικόνα 37 - Το πρωτεύον κλειδί του πίνακα ΛΟΓΑΡΙΑΣΜΟΙ είναι ο συνδυασμός των στηλών Κωδικός_Τράπεζας και Κωδικός_Λογαριασμού

Ένα **ξένο κλειδί (foreign key (FK))** είναι μία στήλη ή ένας συνδυασμός από στήλες ενός πίνακα που μας επιτρέπει να «συνδεθούμε» με μία γραμμή ενός άλλου πίνακα.Για παράδειγμα στο παρακάτω σχήμα από τον πίνακα ΕΡΓΑΖΟΜΕΝΟΙ (EMPLOYESS) γνωρίζουμε ότι Jennifer Whalen εργάζεται στο τμήμα με κωδικό 10. Αν θέλουμε να μάθουμε περισσότερα για το τμήμα στο οποίο εργάζεται η Jennifer Whalen, θα αναζητήσουμε στον πίνακα ΤΜΗΜΑΤΑ (DEPARTMENTS) εκείνη την γραμμή του πίνακα που έχει DEPARTMENT_ID το 10. Το τμήμα με κωδικό 10 έχει όνομα Administration. Άρα, η Jennifer Whalen εργάζεται στο τμήμα Administration [2].

Πρέπει να σημειώσουμε ότι το DEPARTMENT_ID ορίζεται ως το πρωτεύον κλειδί για τον πίνακα ΤΜΗΜΑΤΑ (DEPARTMENTS) ενώ για τον πίνακα ΕΡΓΑΖΟΜΕΝΟΙ (EMPLOYEES) ο κωδικός τμήματος DEPARTMENT_ID ορίζεται ως το ξένο κλειδί. Κάθε τιμή της στήλης DEPARTMENT_ID του πίνακα ΕΡΓΑΖΟΜΕΝΟΙ (EMPLOYEES) πρέπει να «ταιριάζει» με μία τιμή από την στήλη DEPARTMENT_ID του πίνακα ΤΜΗΜΑΤΑ (DEPARTMENTS) [2].

Οι δύο αυτές λοιπόν συσχετιζόμενες μεταξύ τους πληροφορίες «Jennifer Whalen» και «Administration» βρίσκονται σε διαφορετικούς πίνακες και η εύρεση της συσχέτισής τους γίνεται μέσω των ξένων και πρωτευόντων κλειδιών των αντίστοιχων πινάκων [2].

Εικόνα 38 - Το ξένο κλειδί ενός πίνακα «συνδέεται» με το πρωτεύον κλειδί ενός άλλου πίνακα

4.5.2 Διαδικασία Μετασχηματισμού από το Εννοιολογικό Μοντέλο στο Σχεσιακό Μοντέλο

Στην υποενότητα αυτή θα μελετήσουμε τους κανόνες που διέπουν τον μετασχηματισμό του μοντέλου οντοτήτων συσχετίσεων στο σχεσιακό μοντέλο. Η υλοποίηση του σχεσιακού μοντέλου είναι μία σχεσιακή βάση δεδομένων, όπως φαίνεται στο παρακάτω σχήμα, με τον (σχεσιακό) πίνακα ΜΑΘΗΤΕΣ [2].

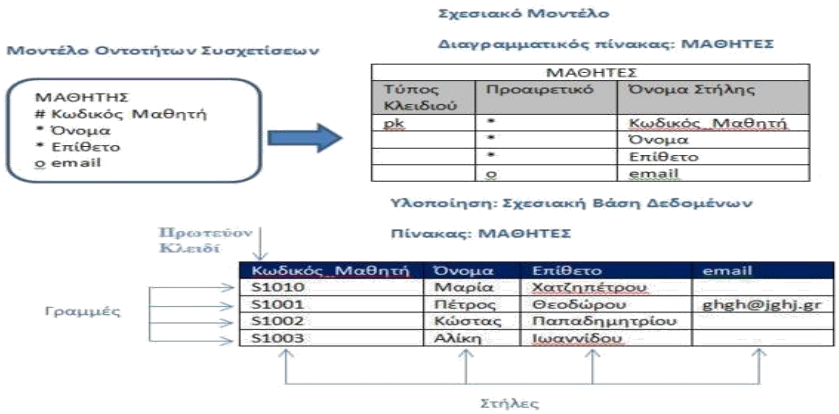
Ο (σχεσιακός) πίνακας ΜΑΘΗΤΕΣ είναι ένας δισδιάστατος πίνακας. Αποτελείται από γραμμές και στήλες. Έχει τέσσερις στήλες Κωδικός_Μαθητή, Όνομα, Επίθετο και email. Κάθε

γραμμή του πίνακα αυτού περιγράφει ένα στιγμιότυπο της οντότητας ΜΑΘΗΤΗΣ. Για παράδειγμα, υπάρχει καταχωρημένος στον πίνακα ο μαθητής με όνομα «Πέτρος», επίθετο «Θεοδώρου», email «ghgh@jghj.gr» και κωδικό μαθητή «S1001». Η στήλη Κωδικός_Μαθητή αποτελεί το πρωτεύον κλειδί του πίνακα. Κάθε μαθητής έχει ένα μοναδικό κωδικό. Δεν μπορούμε, δηλαδή, να συναντήσουμε δύο μαθητές με τον ίδιο κωδικό.

Όλες οι παραπάνω αυτές πληροφορίες σχετικά με τον (σχεσιακό) πίνακα ΜΑΘΗΤΕΣ, όπως αυτές καθορίζονται από το αντίστοιχο μοντέλο οντοτήτων συσχετίσεων, αποθηκεύονται με έναν συγκεκριμένο τρόπο στον διαγραμματικό πίνακα ΜΑΘΗΤΕΣ. Ο διαγραμματικός αυτός πίνακας ονομάζεται ΜΑΘΗΤΕΣ όπως και η αντίστοιχη οντότητα. Συνήθως προτιμούμε τον πληθυντικό αριθμό. Δηλαδή ΜΑΘΗΤΕΣ αντί για ΜΑΘΗΤΗΣ. Ο διαγραμματικός πίνακας αποτελείται από τρεις στήλες. Τον «Τύπο Κλειδιού», το «Προαιρετικό» και το «Όνομα Στήλης».

Ο διαγραμματικός αυτός πίνακας ΜΑΘΗΤΕΣ μας λέει ότι για την οντότητα ΜΑΘΗΤΗΣ θα δημιουργηθεί κατά την υλοποίηση ο (σχεσιακός) πίνακας ΜΑΘΗΤΕΣ που θα έχει τέσσερις στήλες: Κωδικός_Μαθητή, Όνομα, Επίθετο και email, όπως φαίνεται από την στήλη «Όνομα Στήλης» του διαγραμματικού πίνακα ΜΑΘΗΤΕΣ. Οι τρεις πρώτες πληροφορίες για τον μαθητή είναι υποχρεωτικές, όπως φαίνεται από την στήλη «Προαιρετικό» του διαγραμματικού πίνακα ΜΑΘΗΤΕΣ. Δηλαδή, κάθε φορά που θα εισάγουμε έναν μαθητή πρέπει υποχρεωτικά να καταγράφουμε τον κωδικό του, το όνομα του και το επίθετό του. Η καταγραφή του email του μαθητή είναι προαιρετική. Η στήλη Κωδικός_Μαθητή θα αποτελεί το πρωτεύον κλειδί του (σχεσιακού) πίνακα ΜΑΘΗΤΕΣ όπως φαίνεται από την στήλη «Τύπος κλειδιού» του διαγραμματικού πίνακα ΜΑΘΗΤΕΣ. Δηλαδή, στον σχεσιακό πίνακα ΜΑΘΗΤΕΣ όλα τα κελιά της στήλης Κωδικός_Μαθητή θα έχουν διαφορετικές τιμές.

Ας μελετήσουμε στη συνέχεια πιο αναλυτικά τον μετασχηματισμό του μοντέλου οντοτήτων συσχετίσεων, που έχει ως αποτέλεσμα την δημιουργία του διαγραμματικού πίνακα.



Εικόνα 39 - Μετασχηματισμός μοντέλου οντοτήτων συσχετίσεων

Οι κανόνες που ακολουθούνται για τον μετασχηματισμό ενός ER διαγράμματος στο σχεσιακό μοντέλο είναι οι εξής:

Κάθε οντότητα γίνεται ένας πίνακας Κάθε γνώρισμα γίνεται στήλη

Κάθε μοναδικό αναγνωριστικό γίνεται πρωτεύον κλειδί Κάθε συσχέτιση μετασχηματίζεται σε ξένο κλειδί.

Κάθε στιγμιότυπο γίνεται γραμμή του πίνακα (στην φυσική υλοποίηση)

Στηνεικόνα 3.5.7. παρουσιάζεται ο μετασχηματισμός της οντότητας ΜΑΘΗΤΗΣ με βάση τους παραπάνω κανόνες με την μορφή του διαγραμματικού πίνακα ΜΑΘΗΤΕΣ. Ο διαγραμματικός πίνακας ΜΑΘΗΤΕΣ αποτελεί την αναπαράσταση της δομής μιας σχεσιακής βάσης. Ο (σχεσιακός) πίνακας ΜΑΘΗΤΕΣ αποτελεί την μετέπειτα υλοποίηση του διαγραμματικού πίνακα ΜΑΘΗΤΕΣ.

Όπως έχουμε εξηγήσει και παραπάνω ένας διαγραμματικός πίνακας αποτελείται από τρεις στήλες. Τον «Τύπο Κλειδιού», το «Προαιρετικό» και το «Όνομα Στήλης». Η πρώτη γραμμή του διαγραμματικού πίνακα περιέχει το όνομα του σχεσιακού πίνακα. Τα κελιά στην στήλη «Όνομα Στήλης» (δηλαδή Κωδικός_Μαθητή, Όνομα, Επίθετο και email) αντιστοιχούν στα γνωρίσματα της αντίστοιχης οντότητας, καθώς και στις στήλες του αντίστοιχου σχεσιακού πίνακα. Για κάθε κελί που περιέχεται στη στήλη «Όνομα Στήλης» σημειώνεται στο αντίστοιχο κελί της στήλης «Τύπος Κλειδί» το σύμβολο «pk» ή «fk» ανάλογα με το αν το αντίστοιχο γνώρισμα αποτελεί πρωτεύον ή ξένο κλειδί. Κατά επέκταση και η αντίστοιχη στήλη του σχεσιακού πίνακα θα αποτελεί πρωτεύον ή ξένο κλειδί. Η στήλη «Προαιρετικό» περιέχει τα σύμβολα «*» ή «ο» ανάλογα με το αν το αντίστοιχο κελί στην στήλη «Όνομα Στήλης» αναφέρεται σε γνώρισμα που είναι υποχρεωτικό ή προαιρετικό [2].

Ας θεωρήσουμε ένα δεύτερο παράδειγμα, την συσχέτιση Εργαζόμενος – Τμήμα και ας προσπαθήσουμε να την μετασχηματίσουμε ακολουθώντας τους κανόνες, όπως φαίνεται στο παρακάτω σχήμα.

Κάθε οντότητα γίνεται ένας πίνακας

Η οντότητα ΕΡΓΑΖΟΜΕΝΟΣ γίνεται διαγραμματικός πίνακας με το όνομα ΕΡΓΑΖΟΜΕΝΟΙ. Παρόμοια, η οντότητα ΤΜΗΜΑ γίνεται διαγραμματικός πίνακας με το όνομα ΤΜΗΜΑΤΑ.

Κάθε γνώρισμα γίνεται στήλη (του αντίστοιχου σχεσιακού πίνακα)

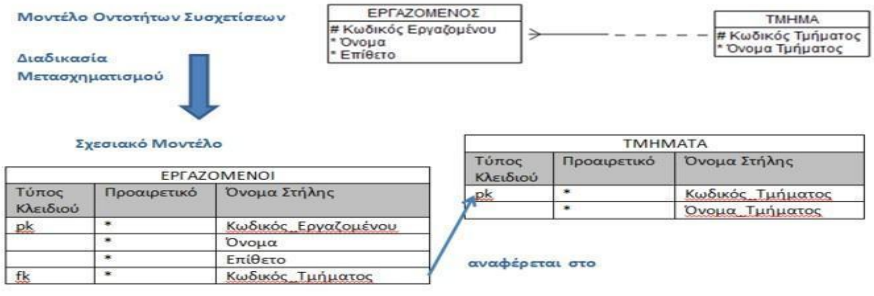
Τα γνωρίσματα Κωδικός Εργαζομένου, Όνομα και Επίθετο της οντότητας ΕΡΓΑΖΟΜΕΝΟΣ μετατρέπονται στα αντίστοιχα κελιά Κωδικός_Εργαζομένου, Όνομα και Επίθετο της στήλης «Όνομα Στήλης» του διαγραμματικού πίνακα ΕΡΓΑΖΟΜΕΝΟΙ. Παρόμοια και για τα γνωρίσματα της οντότητας ΤΜΗΜΑ. Τα σύμβολα του υποχρεωτικού (*) ή του προαιρετικού (ο) των γνωρισμάτων ακολουθούν και τα αντίστοιχα κελιά στην στήλη «Προαιρετικό».

Κάθε μοναδικό αναγνωριστικό γίνεται πρωτεύον κλειδί

Αν ένα γνώρισμα αποτελεί μοναδικό αναγνωριστικό (πρωτεύον κλειδί) για την οντότητα, τότε στο αντίστοιχο κελί της στήλης «Τύπος Κλειδί» σημειώνεται το σύμβολο pk. Στο παράδειγμά μας δηλαδή, τα κελιά Κωδικός_Εργαζομένου και Κωδικός_Τμήματος θα είναι πρωτεύοντα κλειδιά αντίστοιχα για τους (σχεσιακούς) πίνακες ΕΡΓΑΖΟΜΕΝΟΙ και ΤΜΗΜΑΤΑ.

Κάθε συσχέτιση μετασχηματίζεται σε ξένο κλειδί.

Το ξένο κλειδί τοποθετείται σε εκείνον τον διαγραμματικό πίνακα που βρίσκεται στη θέση του «πολλά». Η συσχέτιση μεταξύ ΤΜΗΜΑ και ΕΡΓΑΖΟΜΕΝΟΣ είναι «ένα προς πολλά». Δηλαδή σε ένα τμήμα είναι δυνατό να εργάζονται περισσότεροι από ένας εργαζόμενοι. Το «πολλά» αντιστοιχεί στην οντότητα ΕΡΓΑΖΟΜΕΝΟΣ. Για αυτό και στον διαγραμματικό πίνακα ΕΡΓΑΖΟΜΕΝΟΙ δημιουργείται ένα επιπλέον κελί με το όνομα Κωδικός_Τμήματος που έχει τον ρόλο του ξένου κλειδιού. Η στήλη «Τύπος Κλειδί» στο αντίστοιχο κελί περιέχει το σύμβολο «fk» για να δηλώσουμε ότι είναι ξένο κλειδί. Τα δύο κλειδιά ξένο και πρωτεύον στους δύο διαγραμματικούς πίνακες συνδέονται μεταξύ τους με ένα βέλος για να δείξουμε την μεταξύ τους σχέση.



Εικόνα40 - Ο μετασχηματισμός ενός ER διαγράμματος σε σχεσιακό μοντέλο.

5.Κύριο μέρος Πτυχιακής

Το κεφάλαιο αυτό ασχολείται με την ανάλυση του προβλήματος, την σχεδίαση της λύσης και την υλοποίηση.

5.1 Ανάλυση Προβλήματος

Στόχος της εργασίας είναι η ανάπτυξη ενός συστήματος διαχείρισης ξενοδοχείων. Το κάθε ξενοδοχείο έχει ξεχωριστό interface και στη βάση δεδομένων θέλουμε να αποθηκεύουμε το όνομα του, την κατηγορία, την διεύθυνση, το τηλέφωνο, τον διευθυντή, φωτογραφίες (εσωτερικές-εξωτερικές) και τις τιμές για τα δωμάτια. Για τα δωμάτια από την άλλη μεριά θα αποθηκεύουμε το είδος, την τοποθεσία και την κατηγορία. Επίσης για τους χρήστες του συστήματος θα αποθηκεύουμε το όνομα του, την διεύθυνση, το τηλέφωνο του, τον αριθμό πιστωτικής και ένα μοναδικό κωδικό ο οποίος θα ανατίθεται στον πελάτη αυτόματα. Ο κάθε χρήστης έχει ένα username και ένα password.

5.1.2 Απαιτήσεις Συστήματος

Το σύστημα πρέπει να υποστηρίζει τις παρακάτω διεργασίες:

- Εγγραφή ενός νέου χρήστη. Ο χρήστης θα δίνει όλα τα αναγκαία στοιχεία και θα επιλέγει ένα δικό του username/password. Το σύστημα θα ελέγχει αν είναι μοναδικό το username και αν δεν είναι θα ζητάει από τον χρήστη να δώσει ένα νέο.
- Να επιτρέπεται on-line κρατήσεις και ακυρώσεις κρατήσεων μέσω διαδικτύου. Θα πρέπει να επιτρέπεται στον χρήστη να ζητάει αν μπορεί να κάνει κράτηση η οποία θα περιλαμβάνει δωμάτια(π.χ. 1 δίκλινα, 3 μονόκλινα κτλ) . Το σύστημα θα κάνει την κράτηση μόνο αν όλα όσα ζητάει ο χρήστης είναι διαθέσιμα.
- Σε περίπτωση που δεν υπάρχουν διαθέσιμα αυτά που ζητάει ο χρήστης (τότε και μόνο τότε) το σύστημα θα εμφανίζει/οδηγεί τον χρήστη σε κάποιο άλλο ξενοδοχείο. Σε αυτήν την περίπτωση θα γίνονται τα ακόλουθα.

    - Αν ο χρήστης κάνει τελικά κράτηση στο άλλο ξενοδοχείο τότε 10% των¬ εισπράξεων θα γίνεται στο πρώτο ξενοδοχείο (από εδώ και κάτω σε αυτά τα ποσά θα αναφερόμαστε ως χρέος του ενός ξενοδοχείου προς το άλλο).Η δρομολόγηση σε άλλο ξενοδοχείο θα γίνεται με τα εξής κριτήρια :

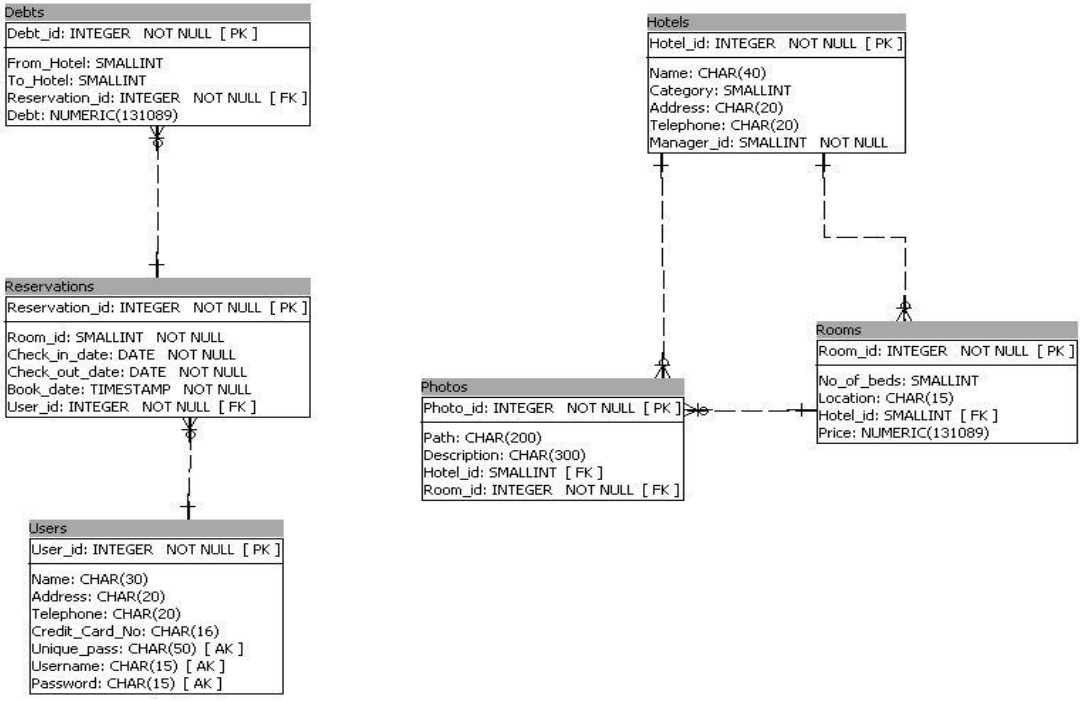
        - Αν υπάρχουν δύο άλλα ξενοδοχεία (που μπορούν να εξυπηρετήσουν την κράτηση που ζητήθηκε (ξενοδοχείο_2 και ξενοδοχείο_3) τότε αν το πρώτο ξενοδοχείο (αυτό που αδυνατεί να εξυπηρετήσει την κράτηση) είναι το ξενοδοχείο_1 υπολογίζουμε τα
          Α= Ποσό_που χρωστάει_το ξενοδοχείο_1_στο ξενοδοχείο_2 - Ποσό_που χρωστάει_το ξενοδοχείο_2_στο ξενοδοχείο_1
          Β= Ποσό_που χρωστάει_το ξενοδοχείο_1_στο ξενοδοχείο_3 - Ποσό_που χρωστάει_το ξενοδοχείο_3_στο ξενοδοχείο_1
          Αν Α>Β επιλέγουμε το ξενοδοχείο_2 αλλιώς το ξενοδοχείο_3. Στόχος είναι να μηδενίζονται όσο το δυνατόν τα χρέη.

Τέλος το σύστημα θα πρέπει να εκτυπώνει και συγκεντρωτικές αναφορές
1. Στο τέλος κάθε χρόνου τι οφείλει το κάθε ξενοδοχείο στο άλλο.
2. Την πληρότητα κάθε μήνα.

5.2 Σχεδιασμός Υλοποίησης

5.2.1 Ανάλυση της Βάσης Δεδομένων postgreSQL

Στην εικόνα παρακάτω απεικονίζεται το σχεσιακό μοντέλο της εργασίας και οι συσχετίσεις μεταξύ των πινάκων στην βάση δεδομένων. Οι πίνακες που δημιουργήσαμε είναι τα **ξενοδοχεία**, τα **δωμάτια**, οι **φωτογραφίες**, τα **χρέη**, οι **κρατήσεις** και οι **χρήστες**. Μπορούμε να διακρίνουμε τα πρωτεύοντα κλειδιά με την χαρακτηριστική γραμμή που τα ξεχωρίζει από τα υπόλοιπα πεδία με την κωδική ονομασία [PK]. Τα ξένα κλειδιά διακρίνονται με την κωδική ονομασία [FK].



Εικόνα 41 - Σχεσιακό μοντέλο της Βάσης Δεδομένων της εφαρμογής μας

5.2.2 Πίνακες

5.2.2.1 Πίνακας Hotels

Στον πίνακα αυτό αποθηκεύουμε τις πληροφορίες του κάθε ξενοδοχείου. Έχει πρωτεύων κλειδί το **Hotel_id** και περιέχει τα πεδία :**Name,Category, Address, Telephone, Manager_id.**

	Hotel_id [PK] integer	Name character(40)	Category smallint	Address character(20)	Telephone character(20)	Manager_id smallint
1	19	Chania 3	3	Chania	2824022812	1
2	20	Chania 1	4	Chania	2821038600	1
3	21	Rethimno 1	4	Rethimnon	2831027401	10
4	22	Rethimno 2	5	Rethimnon	2831027401	10
5	23	Heraklion 1	5	Heraklion	2810238812	1
6	24	Heraklion 2	4	Heraklion	2810228103	10
7	25	St. Nikolas	4	Agios Nikolaos	2841090330	10
8	26	St. Nikolas	5	Agios Nikolaos	2841090200	1
9	27	Heraklion 3	3	Heraklion	2810300019	2
10	28	Chania 2	3	Chania	2821090181	10

Εικόνα 42 - Πίνακας των ξενοδοχείων

5.2.2.2 Πίνακας Rooms

Στον πίνακα αυτό αποθηκεύουμε τα χαρακτηριστικά του κάθε δωματίου. Έχει πρωτεύων κλειδί το**Room_id** και περιέχει τα πεδία: **No_of_beds,Location,Hotel_id,Price.**

	Room_id [PK] integer	No_of_beds smallint	Location character(15)	Hotel_id smallint	Price numeric
1	10	1	sea view	19	30
2	11	2	garden view	19	50.0
3	12	3	sea view	19	70.0
4	13	1	city view	20	30.0
5	14	2	city view	20	50.0
6	15	3	city view	20	70.0
7	16	1	indoor view	21	30.0
8	17	2	sea view	21	50.0
9	18	3	sea view	21	70.0
10	19	1	sea view	22	30.0
11	20	2	sea view	22	50.0
12	21	3	sea view	22	70.0
13	22	1	city view	23	30.0
14	23	2	city view	23	50.0
15	24	3	city view	23	70.0

Εικόνα 43 - Πίνακας των δωματίων

5.2.2.3 Πίνακας Photos

Σε αυτόν τον πίνακα αποθηκεύουμε τις φωτογραφίες που ανεβάζουμε. Έχει πρωτεύων κλειδί το **Photo_id**και περιέχει τα πεδία:**Path,Description,Hotel_id,Room_id.**

	Photo_id [PK] integer	Path character(200)	Description character(300)	Hotel_id smallint	Room_id integer
1	49	logo_chrysana_logo.gif	Chryssana_logo	19	
2	50	Chrysana_single_1.jpg	Chrysana_single_1	19	10
3	51	Chrysana_single_2.jpg	Chrysana_single_2	19	10
4	52	Chrysana_single_3.jpg	Chrysana_single_3	19	10
5	53	Chrysana_double_1.jpg	Chrysana_double_1	19	11
6	54	Chrysana_double_2.jpg	Chrysana_double_2	19	11
7	55	Chrysana_double_3.jpg	Chrysana_double_3	19	11
8	56	Chrysana_triple_1.jpg	Chrysana_triple_1	19	12
9	57	Chrysana_triple_2.jpg	Chrysana_triple_2	19	12
10	58	Chrysana_outside_1.png	Chrysana_outside_1	19	12
11	59	Chrysana_outside_2.png	Chrysana_outside_2	19	

Εικόνα 44 - Πίνακας των φωτογραφιών

5.2.2.4 Πίνακας Debts

Στον πίνακα αυτό αποθηκεύουμε τα στοιχεία των χρεώσεων ανάμεσα στα ξενοδοχεία . Έχει πρωτεύων κλειδί το **Debt_id** και περιέχει τα πεδία: **From_Hotel, To Hotel, Reservasion_id, Debt.**

	Debt_id [PK] integer	From_Hotel smallint	To_Hotel smallint	Reservation_ smallint	Debt numeric
1	1	19	20	13	40
2	2	22	23	15	30
3	3	19	20	13	30
4	5	20	19	13	90
5	6	24	23	34	15.00
6	7	19	20	36	7.000
7	8	23	27	36	10
*					

Εικόνα 45 - Πίνακας των χρεών

5.2.2.5Πίνακας Reservations

Στον πίνακα αυτό αποθηκεύουμε τα στοιχεία των κρατήσεων . Έχει πρωτεύων κλειδί το **Reservation_id** και περιέχει τα πεδία : **Room_id, Check_in_date, Check_out_date, Book_date, User_id.**

	Reservation_ [PK] integer	Room_id smallint	Check_in_dat date	Check_out_d date	Book_date timestamp w	User_id smallint
1	13	22	2016-04-13	2016-04-15	2016-04-04	1
2	15	26	2016-04-10	2016-04-12	2016-04-04	3
3	16	10	2016-04-26	2016-04-27	2016-04-04	1
4	17	17	2016-04-26	2016-04-26	2016-04-26	1
5	18	17	2016-05-01	2016-05-31	2016-04-30	1
6	19	23	2016-05-01	2016-05-31	2016-04-30	1
7	20	23	2016-05-01	2016-05-31	2016-04-30	1
8	34	26	2016-05-15	2016-05-17	2016-05-15	1
9	35	15	2016-05-15	2016-05-31	2016-05-15	1
10	36	12	2016-05-16	2016-05-17	2016-05-15	1
11	46	20	2016-05-19	2016-05-20	2016-05-19	1

Εικόνα 46 - Πίνακας των κρατήσεων

5.2.2.6 Πίνακας Users

Στον πίνακα αυτό αποθηκεύουμε τα στοιχεία των χρηστών . Έχει πρωτεύων κλειδί το **User_id** και περιέχει τα πεδία:**Name, Address, Telephone, Credit_Card_No, Unique_pass, Username, Password.**

	User_id [PK] integer	Name character(30	Address character(20)	Telephone character(20)	Credit_Card_No character(16)	Unique_pass character(50)	Username character(15	Password character(15)
1	1	Alex	Iwnias 45	87656787	4444555566668888	78654567887	Alex83	alex83
2	2	Giorgos	Kapodistrias	385938345	352345756634564	465534545645	giorgosss	3456765
3	3	Tester	werwer4	3234234234234	23445345345345	2398ec233-24	test	test
4	10	Manos	Super 76	8291298719287	848749393748948	2602e068-012	Manos_zid	oskso09eu1x2eu9
*								

Εικόνα 47 - Πίνακας των χρηστών

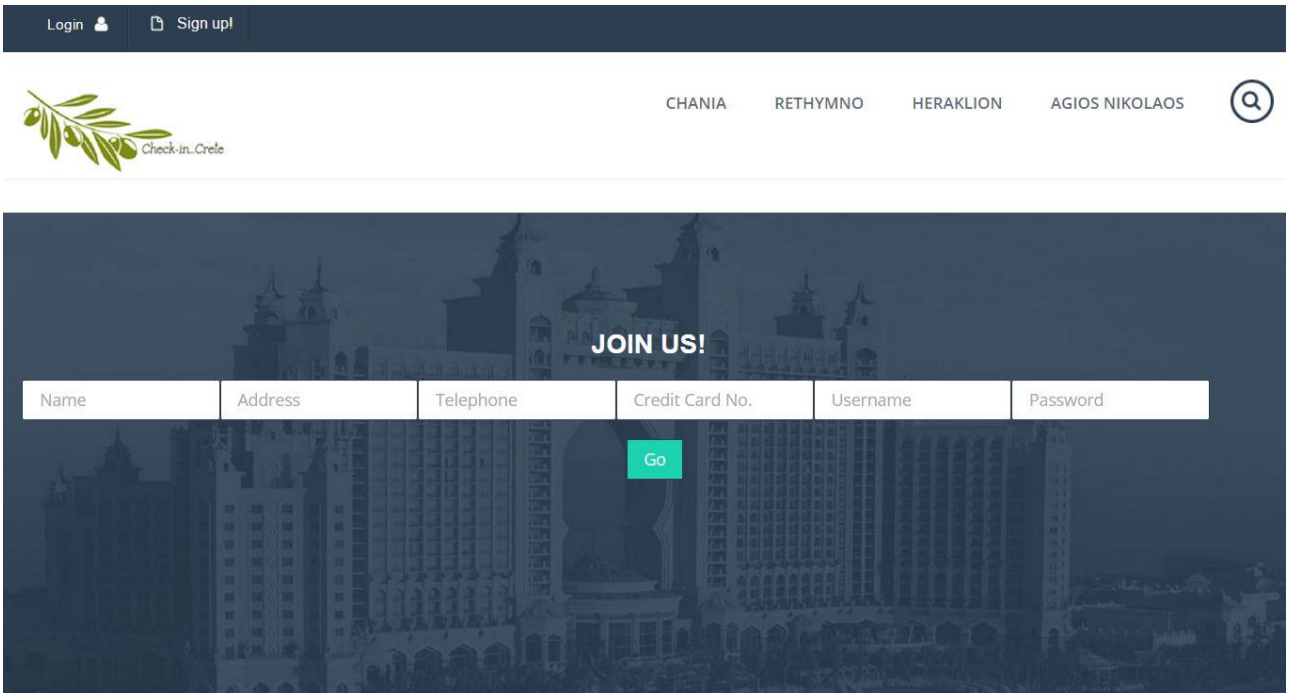
5.3 Υλοποίηση

Αυτή η ενότητα περιλαμβάνει την υλοποίηση της λύσης που σχεδιάσαμε, πλήρες εγχειρίδιο χρήσης και εγχειρίδιο συστήματος.

5.3.1 Εγγραφή ενός νέου χρήστη

Στην κεντρική σελίδα της εφαρμογής υπάρχει ένας slider με φωτογραφίες από τις τέσσερις Περιφερειακές ενότητες της Κρήτης. Ο χρήστης μπορεί να δει πληροφορίες σχετικά με τον νομό που τον ενδιαφέρει όπως τα μέσα μαζικής μεταφοράς που μπορεί να χρησιμοποιήσει, αλλά και να περιηγηθεί στην πόλη μέσω την βοήθεια του GoogleMaps. Επίσης στην κεντρική σελίδα υπάρχει αναζήτηση όπου μπορείς να αναζητήσεις δωμάτιο με τις ημερομηνίες και τον νομό που επιθυμείς. Ακόμα υπάρχει και ένας slider με τα ξενοδοχεία, όπου μπορείς να επιλέξεις απευθείας το ξενοδοχείο και να αναζητήσεις διαθέσιμα δωμάτια.

Ο χρήστης για να κάνει μια κράτηση πρέπει πρώτα να δημιουργήσει έναν λογαριασμό. Αυτό μπορεί να επιτευχθεί πατώντας Signup, όπου και εμφανίζεται η σελίδα εγγραφής νέου χρήστη. Αυτά που πρέπει να συμπληρώσει ο χρήστης είναι το όνομα του, η διεύθυνσή του, το τηλέφωνο του, ο αριθμός πιστωτικής κάρτας, ένα usernameκαι έναν κωδικό.



Εικόνα 48 - Σελίδα Signup

Όταν ο χρήστης συμπληρώσει το πεδίο με το username και τοποθετήσει τον κέρσορα στο δίπλα πεδίο για να γράψει τον κωδικό που θέλει, στέλνεται μέσω κώδικα Ajax στο Servlet"IsUserNameAvailable" το username και ταυτόχρονα ελέγχεται από πόσους χαρακτήρες αποτελείται. Tousername πρέπει να έχει τουλάχιστον τρεις (3) χαρακτήρες ειδάλλως εμφανίζεται ένα ενημερωτικό μήνυμα. Το Servlet από την μεριά του παραλαμβάνει και αποθηκεύει σε μια μεταβλητή το username που πληκτρολόγησε ο χρήστης και με την βοήθεια της συνάρτησης "IsUserNameAvailable" ελέγχουμε αν το username είναι μοναδικό. Η συνάρτηση, δέχεται σαν όρισμα ένα String το username στην περίπτωση μας, κάνει την σύνδεση με την βάση και έπειτα εκτελείται το SQLερώτημα **"select * from \"Users\" where \"Username\"=uname".** Αν δεν υπάρχει το username ήδη στην βάση τότε φορτώνει ένα μικρό εικονίδιο ότι είναι εντάξει. Σε περίπτωση όμως που υπάρχει εγγραφή στην βάση και έχει το ίδιο username, μας εμφανίζει ένα προειδοποιητικό μήνυμα να το αλλάξουμε.

```
<script type="text/javascript">
    $(document).ready(function ()
    {
        $("#uname").change(function ()
        {
            var uname = $(this).val();

            if (uname.length > 3)
            {
                $("#ustatus").html('<img src="images/loading.gif" align="absmiddle"> <font color=#1DD2AF>Checking Username...</font>');
                $.ajax
                        ({
                            type: "POST",
                            url: "IsUserNameAvailable",
                            data: "uname=" + uname,
                            success: function (msg)
                            {
                                $("#ustatus").ajaxComplete(function (event, request, settings)
                                {
                                    $("#ustatus").html(msg);
                                });
                            }
                        });
            } else
            {
                $("#ustatus").html("<p><font color=#1DD2AF>Username should be at least 3 character or more!</font></p>");
            }

        });
    });

</script>
```

Εικόνα 49 - Έλεγχος του username-ajax

```
public class IsUserNameAvailable extends HttpServlet {


    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {

            try {
                Thread.sleep(2000); // makes the thread delay by n second...
            } catch (InterruptedException ex) {
                Logger.getLogger(IsUserNameAvailable.class.getName()).log(Level.SEVERE, null, ex);
            }

            String uname = request.getParameter("uname");
            if(User_DAO.IsUserNameAvailable(uname))
            {
                out.println("<img src='images/tick.gif' align='absmiddle'>");
            }
            else
            {
                out.println("<font color=#1DD2AF>" + uname +
                        ", is already used, please try another...</font>");
            }
        }
    }
```

Εικόνα 50 - Το servlet που ελέγχει το username στην βάση

```
public static boolean IsUserNameAvailable(String uname) {

    try {

        try (Connection conn = MyDb.getConn()) {
            PreparedStatement statement = conn.prepareStatement("select * from \"Users\" where \"Username\"=?");
            statement.setString(1, uname);
            ResultSet result = statement.executeQuery();
            if (!result.next())
            {
               return true;
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
```

Εικόνα 51 - Η συνάρτηση "IsUserNameAvailable"

5.3.2 On-line κράτηση και ακύρωση

Στην κεντρική σελίδα υπάρχει διαθέσιμη η αναζήτηση με την οποία μπορεί ο χρήστης να ψάξει ξενοδοχείο με τις ημερομηνίες, τον προορισμό και τα κρεβάτια που επιθυμεί.

Εικόνα 52 - Η κεντρική αναζήτηση

```
<div class="p-ww">
    <form id="searchform" action="GeneralSearchServlet" method ="GET" onsubmit="return validateForm()">
        <span></span>
        <input class="date" id="datepicker_from"  name="datepicker_from" type="text" value="Check in" onfocus="this.value = '';" onblur="if (this.value === '') {
                this.value = 'Check in';
            }">
        <span></span>
        <input class="date" id="datepicker_till" name="datepicker_till" type="text" value="Check out" onfocus="this.value = '';" onblur="if (this.value === '') {
                this.value = 'Check out';
            }">
        <span></span>
        <input class="dest" type="text" name ="destination" value="Destination" onfocus="this.value = '';" onblur="if (this.value === '') {
                this.value = 'Destination';
            }">
        <input class="bed" type="text" name="beds" value="Beds" onfocus="this.value = '';" onblur="if (this.value === '') {
                this.value = 'Beds';
            }">
        <input type="submit" id="ResultsButton"  value="Results" />
        <!--change CSS in Button  -->
        <br><br>
    </form>
</div>
```

Εικόνα 53 - Η φόρμα της κεντρικής αναζήτησης

Για γίνει η αναζήτηση πρέπει να έχουμε συμπληρώσει όλα τα πεδία της φόρμας. Ο έλεγχος ακεραιότητας γίνεται μέσω Javascript. Μετά τον έλεγχο τα στοιχεία της φόρμας αποστέλλονται με την μέθοδο GETστο servlet "GeneralSearchServlet".

```
<script>
    function validateForm() {
        var dest = document.forms["searchform"]["destination"].value;
        var checkin = document.forms["searchform"]["datepicker_from"].value;
        var checkout = document.forms["searchform"]["datepicker_till"].value;
        var beds = document.forms["searchform"]["beds"].value;
        if (dest === 0 || dest === "Destination" || checkin === 0 || checkin === "Check in" || checkout === 0 || checkout === "Check out" || beds === 0 || beds === "Beds")
        {
            alert("Field must be filled out");
            return false;
        }
    }
</script>
```

Εικόνα 54 - Έλεγχος ακεραιότητας

Στο servlet, αποθηκεύουμε τα στοιχεία που ήρθαν από την φόρμα και με την βοήθεια της συνάρτησης "getAvailableHotelsForPeriod" παίρνουμε τα διαθέσιμα ξενοδοχεία. Δημιουργούμε cookies με τα στοιχεία που έχουμε πάρει από την φόρμα τα οποία είναι ενεργά για πέντε (5) λεπτά. Στη συνέχεια μέσω της συνάρτησης "**getAvailableHotelsForPeriod**" αποθηκεύουμε σε μια μεταβλητή όλα τα διαθέσιμα ξενοδοχεία για την περίοδο και τα κρεβάτια που ζήτησε ο χρήστης. Μετά στην σελίδα "**search_results.jsp**" εμφανίζονται τα διαθέσιμα ξενοδοχεία. Το SQL ερώτημα που χρησιμοποιεί η συνάρτηση είναι το εξής:

"SELECT DISTINCT ON (\"Hotels\".\"Name\") * FROM public.\"Rooms\" INNER JOIN public.\"Hotels\" ON \"Hotels\".\"Hotel_id\" = \"Rooms\".\"Hotel_id\" AND \"Rooms\".\"No_of_beds\" = ? AND \"Hotels\".\"Address\" ILIKE ? AND \"Rooms\".\"Room_id\" NOT IN (SELECT \"Reservations\".\"Room_id\" FROM public.\"Reservations\" WHERE \"Reservations\".\"Check_in_date\" <= ? AND \"Reservations\".\"Check_out_date\" >= ?) ORDER BY \"Hotels\".\"Name\", \"Rooms\".\"Price\";"

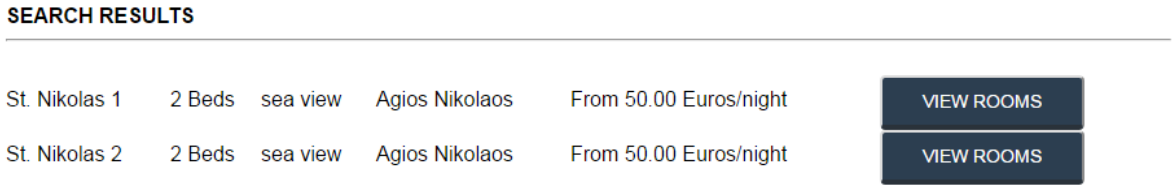
```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException, ParseException {

    try {
        String datepicker_from = request.getParameter("datepicker_from");
        String datepicker_till = request.getParameter("datepicker_till");

        String destination = request.getParameter("destination");
        Integer beds = Integer.parseInt(request.getParameter("beds"));

        //creating cookies for all the above parameters
        Cookie from_date_cookie = new Cookie("date_from", datepicker_from);
        Cookie till_date_cookie = new Cookie("date_till", datepicker_till);
        Cookie destination_cookie = new Cookie("destination", destination);
        Cookie bed_cookie = new Cookie("beds", String.valueOf(beds)); //converts int into string

        //time that cookies expire // 5 minutes
        from_date_cookie.setMaxAge(60 * 15);
        till_date_cookie.setMaxAge(60 * 15);
        destination_cookie.setMaxAge(60 * 15);
        bed_cookie.setMaxAge(60 * 15);

        //add cookies to response
        response.addCookie(from_date_cookie);
        response.addCookie(till_date_cookie);
        response.addCookie(destination_cookie);
        response.addCookie(bed_cookie);

        ArrayList<Result> availableHotels = Result_DAO.getAvailableHotelsForPeriod(datepicker_from, datepicker_till,destination, beds);

        String url = "/search_results.jsp";

        ServletContext sc = getServletContext();

        RequestDispatcher rd = sc.getRequestDispatcher(url);
        request.setAttribute("availableHotels", availableHotels);
        rd.forward(request, response);

    } catch (NumberFormatException | SQLException | ServletException | IOException ex) {

        response.sendRedirect("index.jsp");
    }
}
```

Εικόνα 55 - Το servlet "GeneralSearchServlet"

```
public static ArrayList<Result> getAvailableHotelsForPeriod(String checkindate, String checkoutdate, String destination, int no_beds)
        throws SQLException, ParseException {

    ArrayList<Result> results = new ArrayList<>();

    String takenformat = "MM/dd/yyyy";
    String sendformat = "YYYY-MM-dd";

    SimpleDateFormat takenformatter = new SimpleDateFormat(takenformat, Locale.ENGLISH);
    SimpleDateFormat sendformatter = new SimpleDateFormat(sendformat, Locale.ENGLISH);
    java.util.Date from_temp = takenformatter.parse(checkindate);
    java.util.Date till_temp = takenformatter.parse(checkoutdate);
    String from = sendformatter.format(from_temp);
    String till = sendformatter.format(till_temp);

    Connection conn = MyDb.getConn();
    PreparedStatement pStatement = conn.prepareStatement("SELECT DISTINCT ON (\"Hotels\".\"Name\") * FROM public.\"Rooms\" INNER JOIN
    pStatement.setInt(1, no_beds);
    pStatement.setString(2, "%" + destination + "%");
    pStatement.setDate(3, java.sql.Date.valueOf(from));
    pStatement.setDate(4, java.sql.Date.valueOf(till));
    ResultSet SQLresult = pStatement.executeQuery();

    while (SQLresult.next()) {
        long room_id = SQLresult.getLong("Room_id");
        int beds = SQLresult.getInt("No_of_beds");
        String location = SQLresult.getString("Location");  //gets the third column of the table Hotels
        long hotel_id = SQLresult.getLong("Hotel_id");
        BigDecimal price = SQLresult.getBigDecimal("Price");
        String hotel_name = SQLresult.getString("Name");
        int category = SQLresult.getInt("Category");
        String address = SQLresult.getString("Address");
        String telephone = SQLresult.getString("Telephone");
        long manager_id = SQLresult.getInt("Manager_id");

        Result res = new Result(beds, price, hotel_name, room_id, location, hotel_id, category, address, telephone, manager_id);

        results.add(res);
    }
    SQLresult.close();
    conn.close();

    return results;
}
```

Εικόνα 56 - Η συνάρτηση "getAvailableHotelsForPeriod"

Τα αποτελέσματα της αναζήτησης εμφανίζονται στην σελίδα **"search_results.jsp",** την οποία βλέπουμε παρακάτω**.**



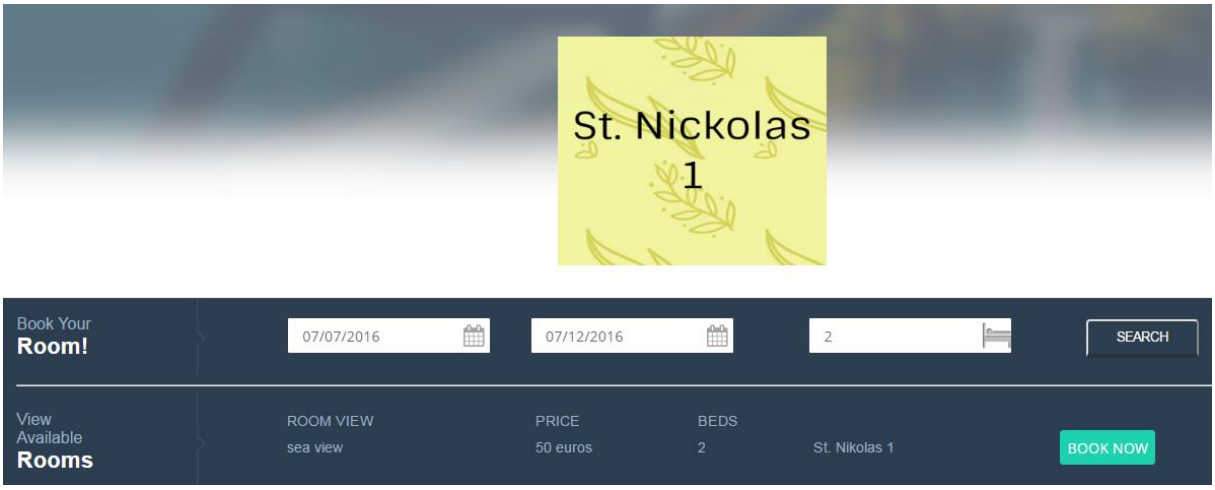
Εικόνα 57- Εμφάνιση διαθέσιμων ξενοδοχείων

Επιλέγοντας το ξενοδοχείο που θέλουμε μεταφερόμαστε στο interface του ξενοδοχείου που επιλέξαμε και μας εμφανίζονται τα διαθέσιμα δωμάτια. Μπορούμε να κάνουμε την κράτηση αλλά και να διαβάσουμε σχετικές πληροφορίες για το ξενοδοχείο .Στον παρακάτω κώδικα μπορούμε να δούμε την **"search_results.jsp"**με την οποία μεταφερόμαστε στο ξενοδοχείο που θέλουμε.

```
<table  border="1" style="width:100%">
        <%
            ArrayList<Result> availableHotels = (ArrayList<Result>) request.getAttribute("availableHotels");

            if (availableHotels.size() > 0)
            {

                for (Result res : availableHotels)
                {
        %>

        <!--use the above code to fetch attributes (available hotels from method)-->

    <tbody>
        <tr>
            <td><a href="hotel_interface.jsp?hotel_id=<% out.print(res.getHotel_id());%>" target="_blank"></td>
            <td><%out.print(res.getHotel_name());%></td>
            <td><%out.print(res.getBeds());%>  Beds</td>
            <td><%out.print(res.getLocation());%></td>
            <td><%out.print(res.getAddress());%></td>
            <td>From <%out.print(res.getPrice());%>0 Euros/night</td>
            <td><input type="button" value="View Rooms" class="btn" onclick="window.open('hotel_interface.jsp?hotel_id=<% out.print(res.getHotel_id());%>')"></td>
        </tr>
        <%
                }
            }
        %>
    </tbody>
</table>
```

Εικόνα 58- Σελίδα"search_results.jsp"

Στην σελίδα του ξενοδοχείου μας εμφανίζονται οι ημερομηνίες και τα κρεβάτια που είχαμε επιλέξει στην αρχική μας αναζήτηση. Υπάρχει η δυνατότητα αν θέλουμε να αλλάξουμε τις ημερομηνίες και τον αριθμό των κρεβατιών. Έπειτα για να κάνουμε την κράτηση δεν αρκεί παρά να πατήσουμε το κουμπί "Book now".



Εικόνα 59 - Εμφάνιση διαθέσιμων δωματίων

Σε περίπτωση που το ξενοδοχείο δεν έχει διαθέσιμο δωμάτιο για τις ημερομηνίες που θέλουμε τότε μας προτείνει κάποιο άλλο στον ίδιο νομό. Επειδή υπάρχουν χρέη μεταξύ των ξενοδοχείων και στόχος μας είναι να μηδενίζονται όσο το δυνατόν τα χρέη, υλοποιούμε τον παρακάτω κώδικα που βρίσκεται στην σελίδα **"RoomSearchServlet"**.

Αποθηκεύουμε στις μεταβλητές τις ημερομηνίες, το ξενοδοχείο και τα κρεβάτια που θέλει ο χρήστης. Έπειτα με την συνάρτηση **"getAvailableRoomsForHotel"**βρίσκουμε τα ξενοδοχεία που έχουν διαθέσιμα δωμάτια για τις συγκεκριμένες ημερομηνίες και τα αποθηκεύουμε σε ένα ArrayList. Μετά ελέγχουμε αν το ArrayList **δεν** είναι άδειο, το οποίο σημαίνει ότι έχει βρει ξενοδοχείο και το εμφανίζει.

```
public class RoomSearchServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException, ParseException {

        try {
            String datepicker_from = request.getParameter("datepicker_from");
            String datepicker_till = request.getParameter("datepicker_till");

            Long hotel_id = Long.parseLong(request.getParameter("hotel_id"));
            Integer beds = Integer.parseInt(request.getParameter("beds"));

            PrintWriter out = response.getWriter();

            ArrayList<Result> availablerooms = Result_DAO.getAvailableRoomsForHotel(datepicker_from, datepicker_till, hotel_id, beds);

            if(!availablerooms.isEmpty()) //if to availablerooms IS NOT empty...Emfanise ta apotelesmata se jsonArray
            {
                JSONArray jsonArray = new JSONArray();
                for(Result r: availablerooms)
                {
                    jsonArray.add(r.toJson());
                }
                out.print(jsonArray.toString()); //kanei print to JsonArray stin String morfi tou
            }
```

Εικόνα 60 - Το Servlet "RoomSearchServlet"

Σε περίπτωση που δεν βρει ξενοδοχείο, δηλαδή το ArrayList είναι άδειο τότε ξαναψάχνει ξενοδοχεία με τις ημερομηνίες και την διεύθυνση του αρχικού ξενοδοχείου για να βρει κάποιο κοντινό. Για να μειωθεί και το χρέος ανάμεσα στα ξενοδοχεία κάνουμε αναζήτηση στο ArrayList που υπάρχουν τα χρέη. Αν το ArrayList είναι άδειο σημαίνει ότι δεν χρωστάς πουθενά και εμφανίζει τα ξενοδοχεία που βρήκε.

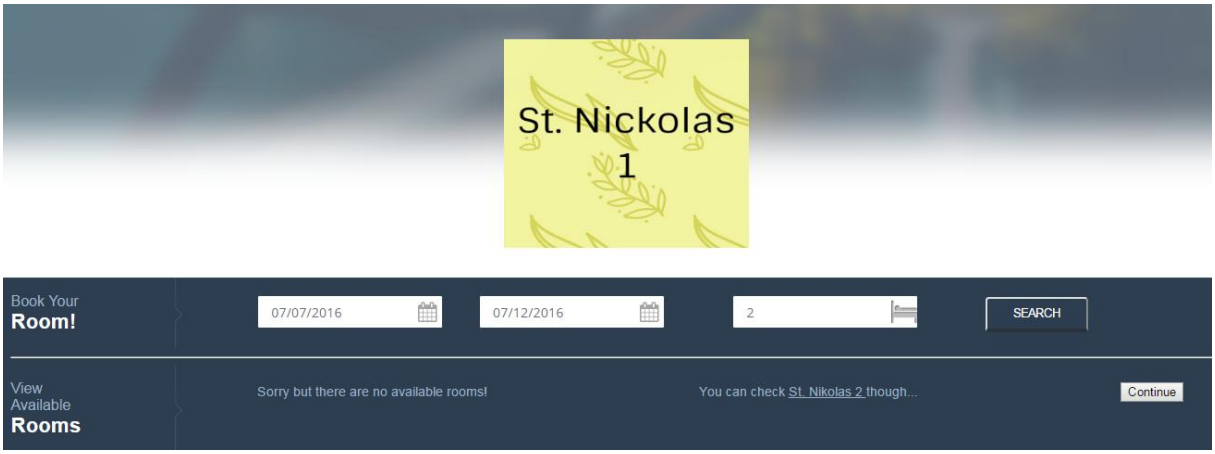
```
else //to hotel DEN exei available dwmatia
{
    Hotel h = Hotel_DAO.getHotelbyId(hotel_id);
    ArrayList<Result> availableHotelsWithAvailableRoomsNearLocation = Result_DAO.getAvailableHotelsForPeriod(datepicker_from, datepicker_till,h.getAddress(), beds);

    //thewroume oti ta xreh einai panta topika!!! Diladi aforoun to idio location/city
    ArrayList<Debt> HotelDebts = Debt_DAO.getDebtsForHotel(hotel_id);

    if(HotelDebts.isEmpty())
    {
        //de xrwstas pouthena. Piase ena apo ta availableHotelsWithAvailableRoomsNearLocation
        //kai epestrepse to.
        JSONArray jsonArray = new JSONArray();

        for(Result r: availableHotelsWithAvailableRoomsNearLocation)
        {
            jsonArray.add(r.toJson());
        }

        out.print(jsonArray.toString());

    }
```

Εικόνα 61 - Το Servlet "RoomSearchServlet"

Αν το ArrayList είναι γεμάτο τότε το ξενοδοχείο χρωστάει σε κάποιο άλλο. Κάνουμε ταξινόμηση κατά φθίνουσα σειρά, έτσι στην πρώτη θέση βρίσκεται το ξενοδοχείο που χρωστάμε τα ποιό πολλά. Αναζητούμε στο ξενοδοχείο αυτό διαθέσιμα δωμάτια με τις ημερομηνίες που έχει επιλέξει ο χρήστης και τα εμφανίζει.

```
        else //alliws na xrwstas, sortare ta apotelesmata kai sti thesi 0 vrisketai to megalytero xreos!
        {
            Collections.sort(HotelDebts);
            //logika kai efoson egine sort me fthinousa seira shmainei oti to megalutero Dept einai sth thesi 0

            //kanw iteration enos by enos ta xreh ypotithetai me th swsth seira (apo to megalutero pros to mikrotero)
            //for(int i=tixrwstawpou.size(); i >=0; i--)
            for(int i=0; i < HotelDebts.size(); i++)
            {
                Debt d = HotelDebts.get(i);

                int hotelIdPouTouXrwstawPolla = d.getTo_Hotel();
                //prepei na elegkw an auto to hotel exei diathesima dwmatia
                //1os tropos: na kalesw thn getAvailableRoomsForHotel
                //2os tropos: na tsekarw an yparxei mesa sto availableHotelsWithAvailableRoomsNearLocation

                //pame me ton 2o
                for(Result r: availableHotelsWithAvailableRoomsNearLocation)
                {
                    if(r.getHotel_id() == hotelIdPouTouXrwstawPolla)
                    {
                        JSONArray jsonArray = new JSONArray();
                        jsonArray.add(r.toJson());
                        out.print(jsonArray.toString());

                        return;
                    }
                }
            }
        }
    } catch (SQLException ex) {
        Logger.getLogger(RoomSearchServlet.class.getName()).log(Level.SEVERE, null, ex);
```

Εικόνα 62 - Το Servlet "RoomSearchServlet"



Εικόνα 63 - Μήνυμα που σε παραπέμπει σε άλλο ξενοδοχείο

Παρακάτω παρουσιάζεται ο κώδικας για την υλοποίηση μιας κράτησης, ο οποίος βρίσκεται στην σελίδα **"ReservationServlet".**

```
String hotel_id_unable_to_serve = request.getParameter("hotel_id_unable_to_serve");
//remember: when hotel_id_unable_to_serve is "null" it means there is no hotel_id_unable_to_serve

String datepicker_from = request.getParameter("datepicker_from");
String datepicker_till = request.getParameter("datepicker_till");

int userid = Integer.parseInt(request.getParameter("user_id"));

int hotel_id = Integer.parseInt(request.getParameter("hotel_id"));
Integer beds = Integer.parseInt(request.getParameter("beds"));

// create rule if fields are empty!
Room room = Room_DAO.getRoombyBedsAndHotelId(beds, hotel_id);

Long room_id = room.getId();
int roomid = toIntExact(room_id); // converts Long into Int


    Reservation_DAO.insertReservation(roomid, datepicker_from, datepicker_till, userid);

ArrayList<Reservation> reservations = Reservation_DAO.getAllReservations(); //gia na doume tin teleutaia kratisi

Long res_id = reservations.get(reservations.size() - 1).getReservation_id();
int resid = toIntExact(res_id);
```

Εικόνα 64 - Το Servlet "ReservationServlet "

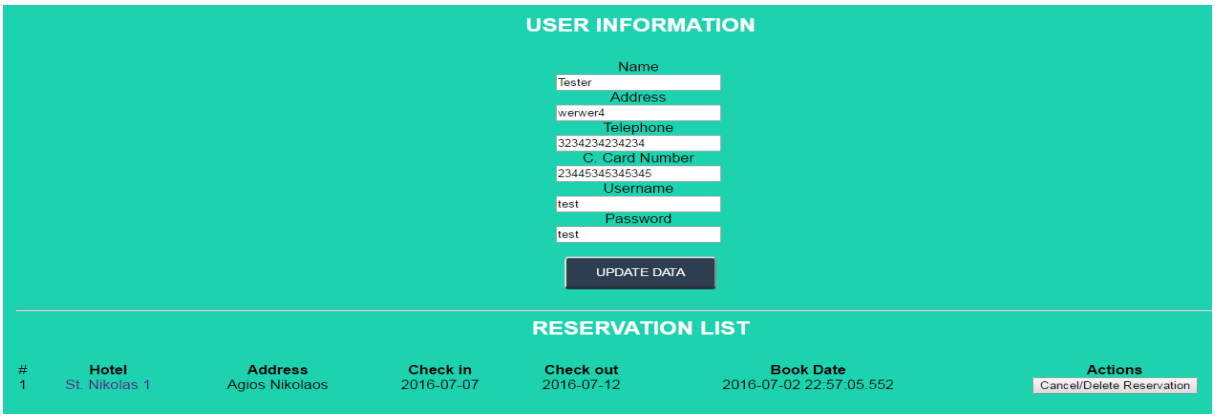
```
if (!hotel_id_unable_to_serve.equals("null")) {

    MathContext mc = new MathContext(4); // 4 precision
    BigDecimal percentage = new BigDecimal("0.100");
    BigDecimal roomprice = room.getPrice();

    int to_hotel = Integer.parseInt(hotel_id_unable_to_serve);
    BigDecimal debt_10 = roomprice.multiply(percentage, mc); //the 1 day Debt

    //to debt prepei na pollaplasiazetai me ta book days
    //allazoume format sto date, gia na ginei i afairesi twn imerwn
    String checkin = "MM/dd/yyyy";
    SimpleDateFormat check_in = new SimpleDateFormat(checkin, Locale.ENGLISH);
    String checkout = "MM/dd/yyyy";
    SimpleDateFormat check_out = new SimpleDateFormat(checkout, Locale.ENGLISH);
    // to kanoume typou Date
    java.util.Date from_temp = check_in.parse(datepicker_from);
    java.util.Date till_temp = check_out.parse(datepicker_till);
    //kanoume tin afairesi
    int days;
    long date = from_temp.getTime();
    long date2 = till_temp.getTime();
    long dif = date2 - date;
    long difdays = dif / (24 * 60 * 60 * 1000) + 1;
    days = (int) difdays - 1;

    BigDecimal daysBD = new BigDecimal(days); // int into BigDecimal
    BigDecimal final_debt = daysBD.multiply(debt_10, mc);
    Debt_DAO.insertDebt(hotel_id, to_hotel, resid, final_debt);
}

RequestDispatcher rs = request.getRequestDispatcher("user_account.jsp");
//the following line, sends the message to the jsp...
request.setAttribute("Bookmessage", "<font color=#1DD2AF>" + "Reservation was a success!" + "</font>");
rs.forward(request, response);

catch (NumberFormatException | SQLException | ServletException | IOException ex) {

response.sendRedirect("index.jsp");
```

Εικόνα 65 - Το Servlet "ReservationServlet "

Για να ελέγξουμε το ξενοδοχείο έχει διαθέσιμα δωμάτια και δεν παραπέμψει τον χρήστη σε κάποιο άλλο ξενοδοχείο χρησιμοποιούμε την μεταβλητή **"hotel_id_unable_to_serve".** Ελέγχουμε την μεταβλητή αυτή και αν δεν είναι **null**, σημαίνει ότι δεν υπάρχουν διαθέσιμα δωμάτια και πήγε σε άλλο ξενοδοχείο και τότε κάνουμε τα παρακάτω. Επειδή πήγε σε άλλο ξενοδοχείο πρέπει να

υπολογίσουμε το χρέος που θα δώσει το δεύτερο ξενοδοχείο στο πρώτο. Υπολογίζουμε το χρέος για μια ημέρα και έπειτα βρίσκουμε πόσες μέρες έχει γίνει η κράτηση για να υπολογίσουμε όλο το χρέος.

Μετά που θα κάνουμε την κράτηση εμφανίζεται ο λογαριασμός μας και οι κρατήσεις που έχουμε κάνει. Σε περίπτωση που έχουμε κάνει κάποιο λάθος και θέλουμε να ακυρώσουμε την κράτηση υπάρχει επιλογή αρκεί να πατήσουμε το κουμπί.



Εικόνα 66 - Σελίδα λογαριασμού χρήστη

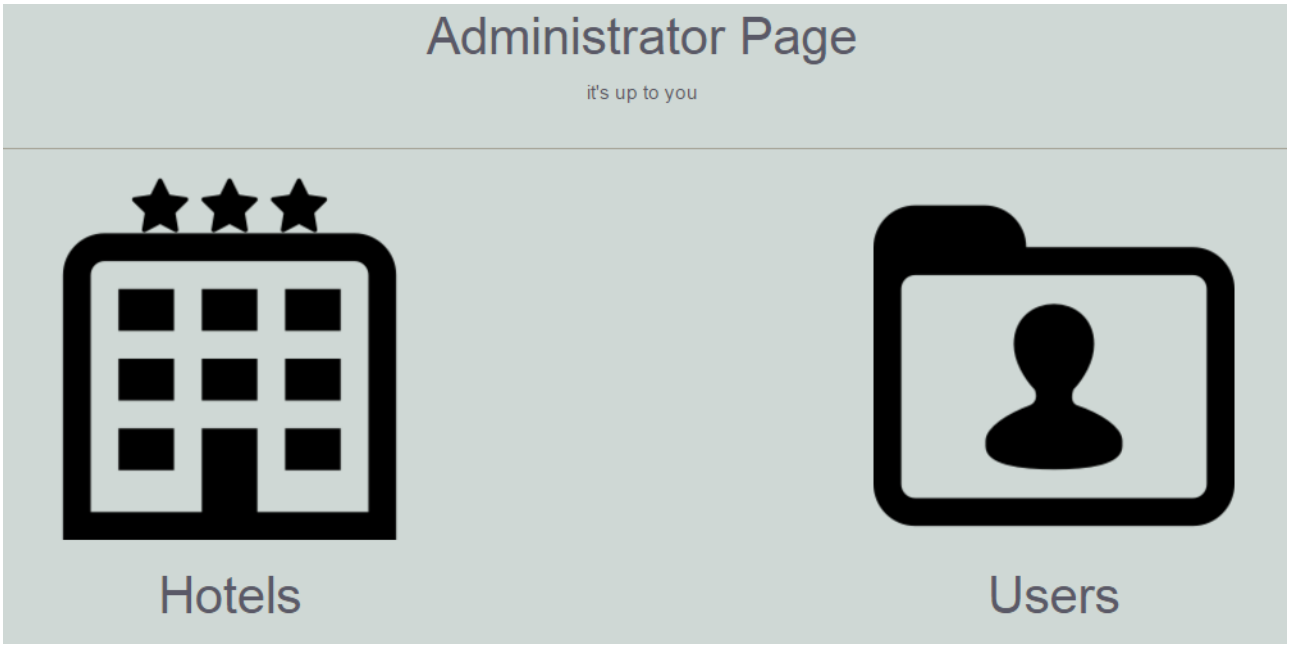
```
public class DeleteReservationServlet extends HttpServlet {
    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     * @throws java.text.ParseException
     * @throws java.sql.SQLException
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException, ParseException, SQLException {

        try {

            Long resid = Long.parseLong(request.getParameter("reservation_id"));
            int res_id = resid.intValue();
            int debt_res = 0;
            ArrayList<Debt> debts = Debt_DAO.getAllDebts();

            for (int i = 0; i < debts.size(); i++) {
                if (debts.get(i).getReservation_id() == res_id) {
                    debt_res = debts.get(i).getReservation_id();
                }
            }
            if (debt_res == res_id) {

                Debt_DAO.deleteDebt(res_id);
                Reservation_DAO.deleteReservation(resid);

            } else {
                Reservation_DAO.deleteReservation(resid);
            }
            RequestDispatcher rs = request.getRequestDispatcher("user_account.jsp");
            //the following line, sends the message to the jsp...
            request.setAttribute("Deletemessage", "<font color=#1DD2AF>" + "Reservation was deleted!" + "</font>");
            rs.forward(request, response);
        } catch (NumberFormatException | SQLException | ServletException | IOException ex) {
            response.sendRedirect("index.jsp");
        }
    }
    HttpServlet methods. Click on the + sign on the left to edit the code.
}
```

Εικόνα 67 - Το Servlet "DeleteReservationServlet"

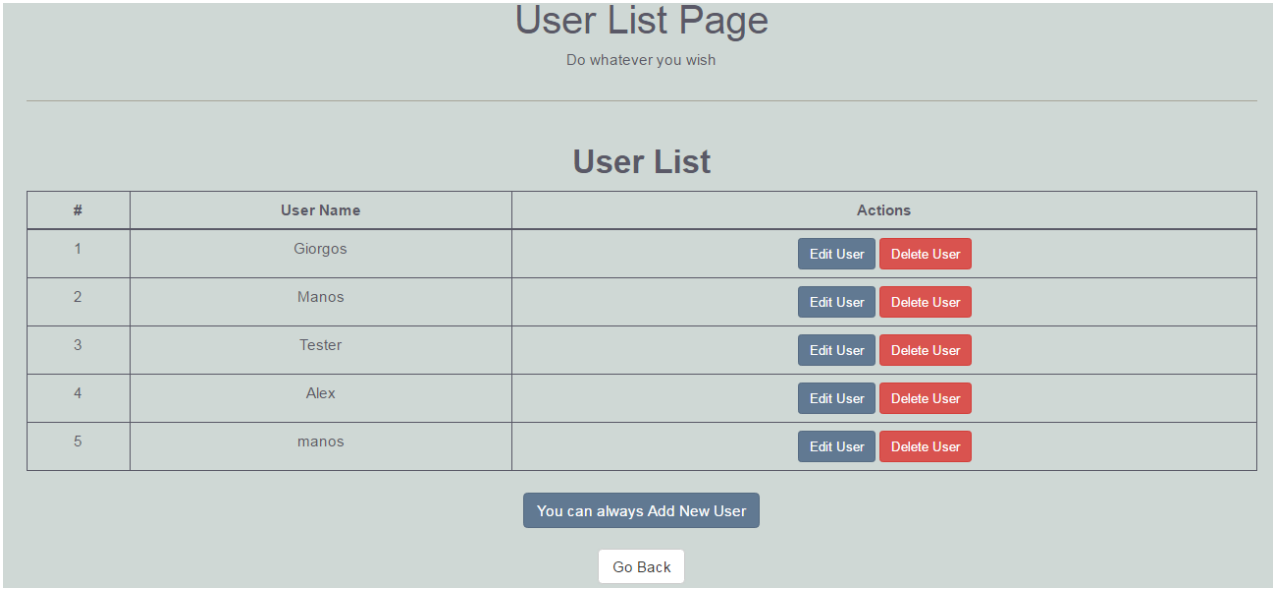
Όταν γίνεται μια ακύρωση πρέπει να ελέγξουμε αν έχει δημιουργηθεί χρέος ανάμεσα στα ξενοδοχεία. Έτσι με την ακύρωση της κράτησης διαγράφεται και το χρέος.

5.3.3 Administrator Page

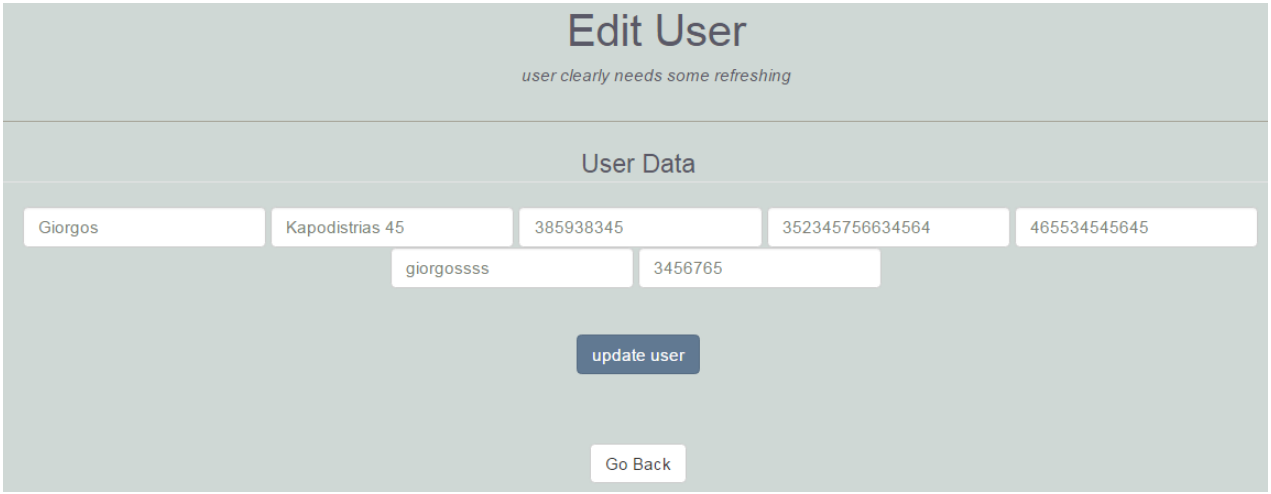
Η σελίδα του διαχειριστή διαθέτει επιπλέον δικαιώματα από έναν απλό χρήστη. Οι δυνατότητες που έχεις σαν διαχειριστής του συστήματος είναι πολλές. Κατ' αρχάς μπορείς να επεξεργαστείς τα στοιχεία των χρηστών, όπως για παράδειγμα να προσθέσεις, να διορθώσεις στοιχεία ή και να διαγράψεις κάποιον χρήστη.



Εικόνα 68 - Σελίδα διαχείρισης



Εικόνα 69 - Σελίδα χρηστών



Εικόνα 70 - Σελίδα επεξεργασίας χρήστη

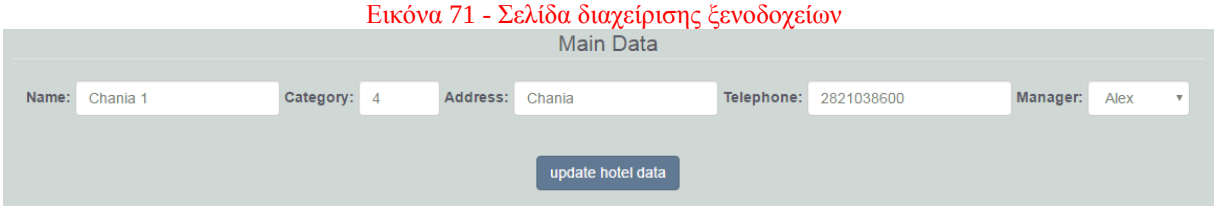
Στην σελίδα του διαχειριστή επίσης έχεις την δυνατότητα της προσθήκης καινούργιου ξενοδοχείου, επεξεργασία στοιχείων όπως πχ προσθήκη φωτογραφιών δωματίων και τροποποίηση τιμών.

Hotel List

#	Hotel Name	Actions
1	Chania 1	Edit Hotel Hotel Reports Delete Hotel
2	Heraklion 1	Edit Hotel Hotel Reports Delete Hotel
3	St. Nikolas 1	Edit Hotel Hotel Reports Delete Hotel
4	Chania 2	Edit Hotel Hotel Reports Delete Hotel
5	Chania 3	Edit Hotel Hotel Reports Delete Hotel
6	Heraklion 2	Edit Hotel Hotel Reports Delete Hotel
7	Heraklion 3	Edit Hotel Hotel Reports Delete Hotel
8	St. Nikolas 2	Edit Hotel Hotel Reports Delete Hotel
9	Rethimno 1	Edit Hotel Hotel Reports Delete Hotel
10	Rethimno 2	Edit Hotel Hotel Reports Delete Hotel

You can always Add New Hotel

Εικόνα 71 - Σελίδα διαχείρισης ξενοδοχείων



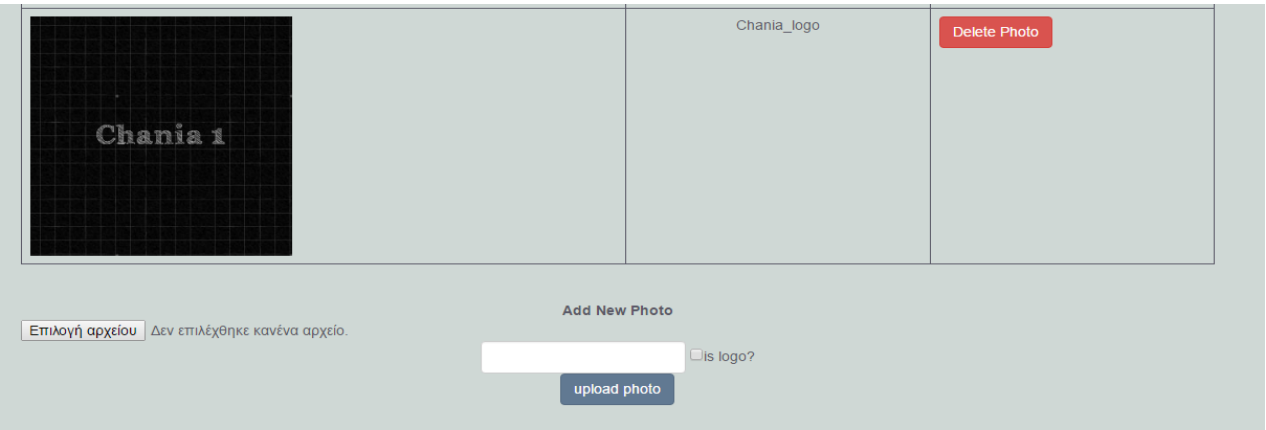
Εικόνα 72 - Σελίδα επεξεργασίας στοιχείων ξενοδοχείου

Rooms

#	Room ID	No of beds	Location	Hotel ID	Price in Euro	Actions
1	13	1	city view	20	30.0	Edit Room Delete Room
2	14	2	city view	20	50.0	Edit Room Delete Room
3	15	3	city view	20	70.0	Edit Room Delete Room
4	34	2	Sea	20	100.0	Edit Room Delete Room
5	35	2	Sea	20	200.0	Edit Room Delete Room

You can always Add New Room

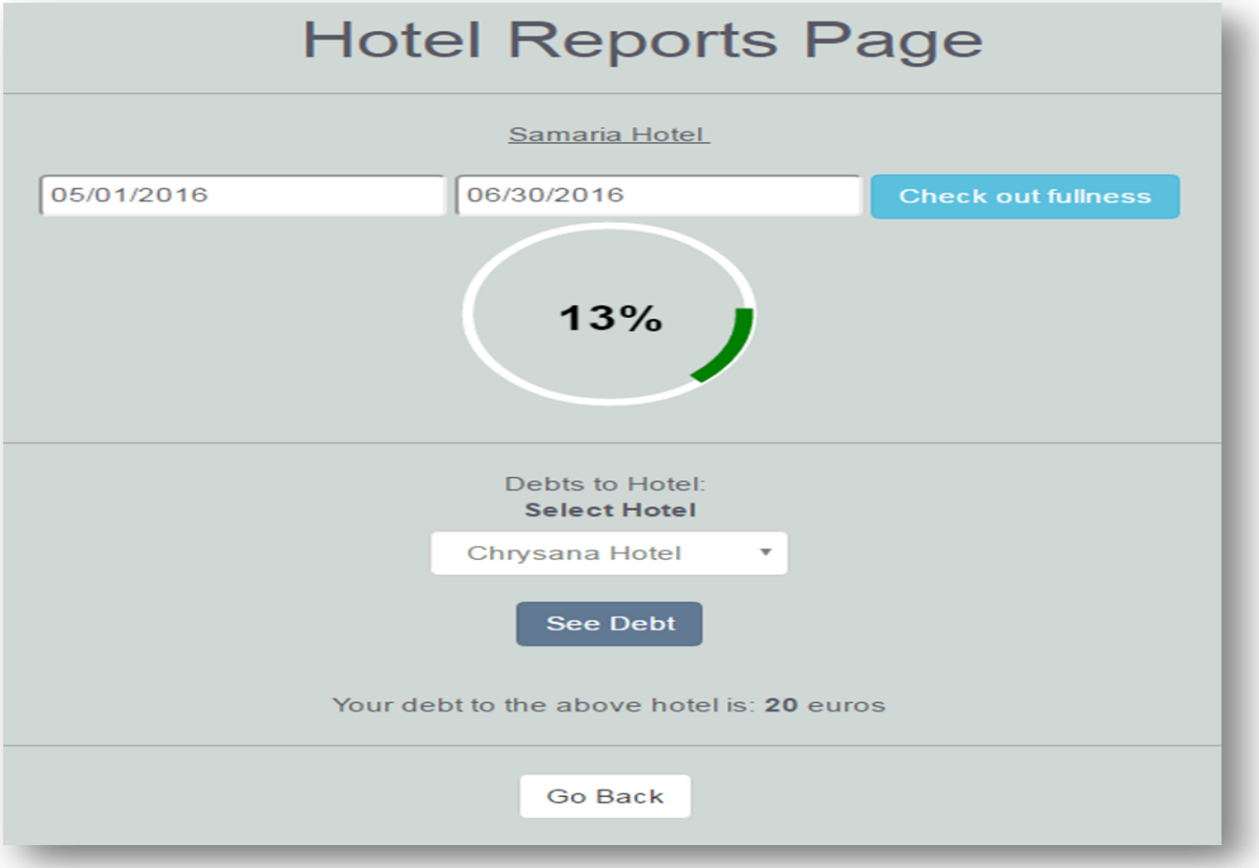
Εικόνα 73 - Σελίδα επεξεργασίας δωματίων



Εικόνα 74 - Σελίδα προσθήκης-αφαίρεσης φωτογραφιών

5.3.4 Συγκεντρωτικές αναφορές

Η εφαρμογή διαθέτει δυο κατηγορίες αναφορών και παρουσιάζονται με γραφικό τρόπο. Η μία κατηγορία έχει να κάνει με τις οφειλές του κάθε ξενοδοχείου στο άλλο. Η δεύτερη κατηγορία αφορά την πληρότητα του εκάστοτε ξενοδοχείου. Πρόσβαση στις αναφορές υπάρχει από την σελίδα του διαχειριστή με την λίστα των ξενοδοχείων όπου μπορεί να επιλέξει για ποιο ξενοδοχείο επιθυμεί να δει τις αναφορές.



Εικόνα 75 - Σελίδα αναφορών

6 Αποτελέσματα

6.1 Συμπεράσματα

Το διαδίκτυο και ειδικά τα ολοκληρωμένα συστήματα κράτησης έχουν επηρεάσει θετικά τη ανάπτυξη του τουρισμού. Κάθε χρόνο όλο και περισσότερες επιχειρήσεις αναπτύσσονται για την δημιουργία νέων υπηρεσιών και καινοτομιών που αφορούν τον τουριστικό τομέα, με σκοπό την μείωση του κόστους και την αύξηση των κερδών. Τα σύγχρονα συστήματα διαχείρισης κρατήσεων έχουν βελτίωση στα μέγιστα την εξυπηρέτηση πολλών χρηστών ταυτόχρονα χάρις την ικανότητά τους στην υποστήριξη μεγάλου όγκου δεδομένων και πληροφοριών.

6.2 Μελλοντική Εργασία και Επεκτάσεις

Εν κατακλείδι θα θέλαμε να επισημάνουμε κάποιες μελλοντικές αλλαγές και επεκτάσεις οι οποίες κρίνονται απαραίτητες. Μια σημαντική επέκταση στο μέλλον θα ήταν η προσθήκη αναζήτησης με βάση την επιλογή παροχών που επιθυμεί ο πελάτης. Με αυτό τον τρόπο έχει την δυνατότητα να αναζήτηση ξενοδοχείο με βάση την κατηγορία, τις παροχές του ξενοδοχείου αλλά και την τιμή ανά βράδυ. Αυτό διευκολύνει τον πελάτη να βρει αυτό που χρειάζεται πιο γρήγορα, για την καλύτερη εξυπηρέτηση του. Πλέον όλο και περισσότεροι άνθρωποι χρησιμοποιούν την πιστωτική τους κάρτα για online αγορές. Γι' αυτό προτείνουμε την προσθήκη πληρωμής μέσω της πιστωτικής κάρτας. Επιπρόσθετα μια εξίσου αξιόλογη επέκταση σε επίπεδο ασφάλειας είναι η προσθήκη πρωτόκολλου SSL για την ασφαλή σύνδεση μεταξύ της ιστοσελίδας και του χρήστη. Έτσι εξασφαλίζουμε την ασφαλή ανταλλαγή δεδομένων, αλλά και βοηθούμε τον χρήστη να επιβεβαιώσει την ταυτότητα της ιστοσελίδας που επισκέπτεται.

Βιβλιογραφία

[1] «Αρχιτεκτονική MVC» [Ηλεκτρονικό]. Available : http://webapptester.com/mvc-framework-first-impression/
[2] «Συστήματα Διαχείρισης Βάσεων Δεδομένων και Εφαρμογές τους στο Διαδίκτυο» [Ηλεκτρονικό]. Available : http://tinyurl.com/zpfqv88
[3] «NetBeans» [Ηλεκτρονικό]. Available : https://netbeans.org/
[4] «NetBeans |Opensoft»[Ηλεκτρονικό].Available: http://www.opensoft.gr/software/netbeans
[5] «PHP.net» [Ηλεκτρονικό]. Available : http://php.net/
[6] «C» [Ηλεκτρονικό].Available : https://en.wikipedia.org/wiki/C_(programming_language)
[7] «C++» [Ηλεκτρονικό].Available : http://www.cplusplus.com/doc/tutorial/
[8] «HTML» [Ηλεκτρονικό].Available : http://www.w3schools.com/html/
[9] «HTML5» [Ηλεκτρονικό].Available : http://www.w3schools.com/html/html5_intro.asp
[10] «pgAdmin» [Ηλεκτρονικό].Available : https://www.pgadmin.org/
[11] «Java» [Ηλεκτρονικό].Available : https://www.java.com/en/
[12] «Css» [Ηλεκτρονικό].Available : https://www.w3.org/Style/CSS/
[13] «JavaScript» [Ηλεκτρονικό].Available : https://www.javascript.com/
[14] «Ajax» [Ηλεκτρονικό].Available : http://www.w3schools.com/ajax/
[15] «Json» [Ηλεκτρονικό].Available : http://www.json.org/
[16] «jQuery» [Ηλεκτρονικό].Available : https://jquery.com/
[17] «Bootstrap» [Ηλεκτρονικό].Available : http://getbootstrap.com/
[18] «PostgreSQL» [Ηλεκτρονικό]. Available : https://www.postgresql.org/
[19] «JavaServerPages» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/JavaServer_Pages

Παράρτημα

Περίληψη

Στην παρούσα πτυχιακή εργασία ασχοληθήκαμε με την υλοποίηση ενός ολοκληρωμένου συστήματος διαχείρισης ξενοδοχείων, όπου το κάθε ξενοδοχείο διαθέτει ξεχωριστό interface. Ο χρήστης έχει την δυνατότητα εγγραφής στο σύστημα και αναζήτησης δωματίων με βάση την ημερομηνία και τον προορισμό που επιθυμεί.

Η παρούσα πτυχιακή εργασία υλοποιήθηκε σε γλώσσα προγραμματισμού Java. Το πρόγραμμα που χρησιμοποιήσαμε για την ανάπτυξης της εφαρμογής ήταν το NetBeans. Είναι ένα ενσωματωμένο περιβάλλον ανάπτυξης εφαρμογών (Integrated Development Environment - IDE). Στην αρχή χρησιμοποιήθηκε για την ανάπτυξη εφαρμογών Java αλλά πλέον χρησιμοποιείται και για άλλες γλώσσες όπως PHP, C/C++, και HTML5. Με τις τελευταίες εκδόσεις έχει εξελιχθεί σε ένα ισχυρό πρόγραμμα επεξεργασίας για web scripting. Για την δημιουργία της βάσης χρησιμοποιήσαμε το πρόγραμμα pgAdmin.

Στο κεφάλαιο 2 περιγράφουμε την λογική με την οποία υλοποιήσαμε αυτή την εφαρμογή και ιδιαίτερα την αρχιτεκτονική την οποία χρησιμοποιήσαμε. Επίσης αναφέρουμε τα βασικά πλεονεκτήματα τα οποία προσφέρει η συγκεκριμένη αρχιτεκτονική. Το κεφάλαιο 2 ακόμα περιλαμβάνει τις τεχνολογίες τις οποίες χρησιμοποιήσαμε με αναλυτική περιγραφή για την κάθε μια.

Στο 3ο κεφάλαιο αναφέρουμε την έννοια της πληροφορίας και πως μας διευκολύνει για να αντιμετωπίσουμε τα προβλήματα που μας απασχολούν. Γίνετε αναφορά στην σημασία και την εξέλιξη της διαχείρισης των δεδομένων, δίνοντας και κάποια παραδείγματα που έχουν εφαρμογή οι βάσεις δεδομένων.

Στο κεφάλαιο 4 αναφέρετε στον τρόπο με τον οποίο σχεδιάσαμε την βάση δεδομένων. Περιγράφονται αναλυτικά τα βήματα τα οποία χρειάζονται για να υλοποιήσεις για τον μετασχηματισμό του διαγράμματος οντοτήτων συσχετίσεων σε σχήμα σχεσιακής βάσης δεδομένων.

Στο πέμπτο και προτελευταίο κεφάλαιο γίνετε ανάλυση της εφαρμογής, ξεκινώντας από την βάση δεδομένων και στη συνέχεια περιγράφουμε τις δυνατότητες που έχει ο χρήστης. Στην κεντρική σελίδα υπάρχει η δυνατότητα να δημιουργήσει καινούργιο λογαριασμό με τα στοιχεία του. Επιπρόσθετα στην κεντρική σελίδα μπορεί να αναζητήσει δωμάτιο με τις ημερομηνίες και τον προορισμό που θέλει. Επιλέγοντας το ξενοδοχείο που επιθυμεί ο χρήστης είναι στην διάθεση του και οι πληροφορίες σχετικά με τα δωμάτια και τις παροχές τις οποίες προσφέρει. Ο χρήστης διαθέτει σελίδα διαχείρισης κρατήσεων και έχει την δυνατότητα να ακυρώσει μια κράτηση αλλά και να αλλάξει τα στοιχεία του σε περίπτωση που έχει κάνει κάποιο λάθος. Ο διαχειριστής από την άλλη μεριά έχει πλήρη έλεγχο του συστήματος. Μερικές από τις δυνατότητες που έχει είναι η διαχείριση ξενοδοχείων. Μπορεί να προσθέσει και να αφαιρέσει ένα ξενοδοχείο, να επεξεργαστεί τις πληροφορίες με τα ήδη υπάρχοντα δωμάτια, καθώς και να πρόσθεση μια νέα φωτογραφία ή ακόμα και την διαγράψει.

Το 6ο και τελευταίο κεφάλαιο περιλαμβάνει το συμπέρασμα και τις μελλοντικές επεκτάσεις που χρειάζονται στην παρούσα πτυχιακή εργασία.