Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Πανεπιστήμιο Θεσσαλίας

Διπλωματική Εργασία με τίτλο:

Ανάπτυξη διαδικτυακού εργαλείου μοντελοποίησης βάσεων δεδομένων

Development of a web tool for database modeling

Ειρήνη Παπαδοπούλου

Επιβλέπων:

Μιχαήλ Βασιλακόπουλος, Αναπληρωτής Καθηγητής

Βόλος, 2017

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Πανεπιστήμιο Θεσσαλίας

Διπλωματική Εργασία με τίτλο:

Ανάπτυξη διαδικτυακού εργαλείου μοντελοποίησης βάσεων δεδομένων

Development of a web tool for database modeling

Ειρήνη Παπαδοπούλου

Επιβλέπων Α'	Επιβλέπων Β'
Μιχαήλ Βασιλακόπουλος	Εμμανουήλ Βάβαλης
Αναπληρωτής Καθηγητής	Καθηγητής

Βόλος, 2017

Στην οικογένειά μου

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Μιχαήλ Βασιλακόπουλο ως επιβλέποντα καθηγητή της παρούσας εργασίας, για τις χρήσιμες συμβουλές και την συνεχή καθοδήγησή του μέχρι την ολοκλήρωση της εργασίας. Επιπλέον, θέλω να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή κ. Εμμανουήλ Βάβαλη για την πολύτιμη βοήθειά του.

Ευχαριστώ την οικογένειά μου, τους φίλους, και τον σύντροφό μου που ήταν στο πλευρό μου όχι μόνο κατά την υλοποίηση της εργασίας, αλλά καθ' όλη την διάρκεια των φοιτητικών μου χρόνων.

Επίσης, θα ήθελα να ευχαριστήσω τους συμφοιτητές μου σε αυτό το τμήμα, με τους οποίους μοιράστηκα κοινές εμπειρίες, χαρές και εμπόδια κατά την διάρκεια των σπουδών μας.

Περίληψη

Διανύουμε μια εποχή που χαρακτηρίζεται από την ύπαρξη ενός τεράστιου όγκου πληροφορίας σε διάφορους τομείς της καθημερινής ζωής, από ψυχαγωγία μέχρι και επιστήμη. Επομένως, έχει δημιουργηθεί η ανάγκη οργάνωσης και διαχείρισης αυτής της πληροφορίας από τα υπολογιστικά συστήματα. Με την εισαγωγή του παγκόσμιου ιστού στην σημερινή κοινωνία, η ανάγκη για την διαχείριση της πληροφορίας είναι ακόμη μεγαλύτερη.

Για την οργάνωση μεγάλου όγκου πληροφοριών χρησιμοποιούνται πλέον οι βάσεις δεδομένων, που αποτελούν ξεχωριστό κομμάτι της επιστήμης των υπολογιστικών συστημάτων, καθώς και εργαλεία για την διαχείρισή τους.

Το αντικείμενο της παρούσας εργασίας είναι η ανάλυση του μοντέλου Οντοτήτων - Συσχετίσεων που χρησιμοποιείται για την μοντελοποίηση μιας βάσης δεδομένων, καθώς και η ανάπτυξη ενός διαδικτυακού εργαλείου που βασίζεται στο μοντέλο αυτό και επιτρέπει την δημιουργία διαγραμμάτων οντοτήτων - συσχετίσεων. Επιπλέον, το εργαλείο αυτό είναι σε θέση να μετατρέψει κάποιο διάγραμμα σε κώδικα PostgreSQL, ένα σύστημα διαχείρισης βάσεων δεδομένων βασιζόμενο στο αντικειμενο - σχεσιακό μοντέλο.

Abstract

We currently live in an age which is identified by the existence of a considerable amount of information that plays an important role in various areas of everyday life. As a result, there is always the need to find ways so that this amount of information can be organized and managed by computers. Ever since the world wide web allowed ease of communication throughout the world, the need for such solutions has become even greater.

Databases are now used to store and manage this great amount of information, forming a separate field in computer science. In addition, various tools have been developed in order to manage these databases.

The main subject of this thesis is the analysis of the Entity - Relationship model, which is used during the phase of designing a database, as well as the development of an online tool which is based on this model and allows the users to create their own entity - relationship diagrams. This tool is also able to convert such a diagram into PostgreSQL code, an object-relational database management system.

Περιεχόμενα

1.Εισαγωγή

1.1 Αντικείμενο Εργασίας

Η σημερινή εποχή χαρακτηρίζεται από την ταχεία ανάπτυξη της τεχνολογίας, ειδικότερα στον τομέα των υπολογιστικών συστημάτων και στις τεχνολογίες παγκόσμιου ιστού. Άνθρωποι κάθε ηλικίας χρησιμοποιούν τα προϊόντα της τεχνολογίας για ψυχαγωγία, επικοινωνία ή ενημέρωση και σε πολλές περιπτώσεις βασίζονται σε αυτά για την διεκπεραίωση απλών ή περίπλοκων διαδικασιών. Η παρουσία των υπολογιστών και έξυπνων συστημάτων είναι πλέον αναπόσπαστο κομμάτι των σύγχρονων κοινωνιών, καθώς αποτελεί απαραίτητη προϋπόθεση για την ομαλή καθημερινή ζωή. Επιπλέον, η ανάπτυξη αυτή έχει ως αποτέλεσμα νέες δυνατότητες που παλιότερα αποτελούσαν σενάρια επιστημονικής φαντασίας. Τέτοιες δυνατότητες γίνονται αντιληπτές σε περιπτώσεις όπως η χρήση βάσεων δεδομένων στον τομέα της βιολογίας, ή ακόμα και η χρήση μέσων κοινωνικής δικτύωσης που επιτρέπουν στους χρήστες να επικοινωνήσουν με άτομα από ολόκληρο τον κόσμο.

Από τον καιρό της κατασκευής των πρώτων ακόμη υπολογιστών, δημιουργήθηκε η ανάγκη της συλλογής, αποθήκευσης και διαχείρισης πληροφοριών. Μία βάση δεδομένων αποτελεί μια συλλογή δεδομένων οι οποίες είναι οργανωμένες με τέτοιο τρόπο ώστε να μπορούν να προσπελαστούν, να ενημερωθούν και να διαχειριστούν. Είναι λοιπόν εύκολο να διαπιστώσει κανείς την σημασία των βάσεων δεδομένων σε μια εποχή που χαρακτηρίζεται από μια τεράστια ποσότητα δεδομένων που ανανεώνεται καθημερινά και είναι διαθέσιμη στον κάθε άνθρωπο. Συνεπώς δημιουργείται η ανάγκη αποτελεσματικής διαχείρισης αυτού του όγκου δεδομένων με απλό τρόπο. Αυτόν τον ρόλο αναλαμβάνει ένα σύστημα διαχείρισης βάσεων δεδομένων (DataBase Management System), λογισμικό που έχει ως σκοπό να κάνει την διαχείριση μεγάλου όγκου δεδομένων πιο εύκολη.

Πριν από την αποθήκευση και διαχείριση της πληροφορίας, είναι απαραίτητο να μπορούμε να απεικονίσουμε μια βάση δεδομένων κατά την διάρκεια του σχεδιασμού της. Το μοντέλο Οντοτήτων-Συσχετίσεων (Entity-Relationship ή ER model) χρησιμεύει στην απεικόνιση των πληροφοριών που θα αποθηκευτούν σε μια βάση δεδομένων, συγκεκριμένα περιγράφει τις οντότητες και τις συσχετίσεις μεταξύ τους. Το μοντέλο αυτό είναι αρκετά διαδεδομένο κατά τον αρχικό σχεδιασμό μιας βάσης δεδομένων, καθώς επιτρέπει σε μια επιχείρηση να περιγράψει με μεγαλύτερη λεπτομέρεια τις απαιτήσεις των χρηστών της, όσον αφορά την βάση δεδομένων. Αποτέλεσμα της σχεδίασης μιας βάσης δεδομένων χρησιμοποιώντας αυτό το μοντέλο είναι το διάγραμμα οντοτήτων-συσχετίσεων (ER diagram), το οποίο περιγράφει τις οντότητες και συσχετίσεις με μορφή σχημάτων, καθώς και συνδέσεις μεταξύ τους.

Το αντικείμενο της παρούσας εργασίας είναι η ανάπτυξη και επεξήγηση της διαδικτυακής εφαρμογής SQL Draw, η οποία βασίζεται στο Μοντέλο Οντοτήτων-Συσχετίσεων (Entity Relationship Model ή ER model) για την μοντελοποίηση μιας βάσης δεδομένων. Πιο συγκεκριμένα, η εφαρμογή SQL Draw επιτρέπει την δημιουργία διαγραμμάτων οντοτήτων-συσχετίσεων από τον χρήστη για την απεικόνιση της πληροφορίας και στην συνέχεια προσφέρει την επιλογή παραγωγής εντολών για την δημιουργία της συγκεκριμένης βάσης δεδομένων στο σύστημα PostgreSQL (Object-Relational DataBase Management System ή ORDBMS). Τόσο η PostgreSQL όσο και το EER model θα αναλυθούν περαιτέρω σε επόμενα κεφάλαια.

1.2 Διάρθρωση Εργασίας

Στο κεφάλαιο 1 της παρούσας διπλωματικής παρουσιάζεται μια εισαγωγή στο αντικείμενο της εργασίας. Στο κεφάλαιο 2 συζητάμε τις σημαντικές έννοιες του μοντέλου Οντοτήτων - Συσχετίσεων και του σχεσιακού μοντέλου. Στο κεφάλαιο 3 παρουσιάζεται με λεπτομέρεια η λειτουργία της εφαρμογής SQL Draw, ενώ στο κεφάλαιο 4 αναλύονται οι τεχνολογίες που χρησιμοποιήθηκαν στην ανάπτυξη της εφαρμογής. Το κεφάλαιο 5 περιέχει την ανάλυση του κώδικα της εφαρμογής. Τέλος,

στο κεφάλαιο 6 γίνεται μια σύντομη αναφορά των μελλοντικών σχεδίων αναφορικά με την εφαρμογή SQL Draw.

2.Θεωρητικό Υπόβαθρο

2.1 Βάσεις Δεδομένων και DBMS

Οι βάσεις δεδομένων είναι άμεσα συνδεδεμένες με την σημερινή ανάπτυξη των πληροφοριακών συστημάτων, καθώς εφαρμόζονται σε σχεδόν όλους τους τομείς της σύγχρονης ζωής, όπως για παράδειγμα το εμπόριο και η επιστήμη. Επιχειρήσεις και οργανισμοί διαθέτουν έναν μεγάλο όγκο πολύτιμων πληροφοριών που καλούνται να διαχειριστούν. Η επιτυχία μιας επιχείρησης μπορεί να εξαρτάται άμεσα από την αποτελεσματική και έξυπνη διαχείριση αυτών των πληροφοριών.

Ένας γενικός ορισμός της βάσης δεδομένων είναι μια συλλογή από δεδομένα που συσχετίζονται μεταξύ τους [1]. Συνήθως, μια βάση δεδομένων ακολουθεί κάποιους κανόνες και εξυπηρετεί κάποιο σκοπό. Μπορούμε να θεωρήσουμε ότι μια βάση δεδομένων περιέχει δεδομένα που προέρχονται από τον πραγματικό κόσμο και αλλαγή στον πραγματικό κόσμο σημαίνει αλλαγή και στην βάση δεδομένων. Επιπλέον, αυτή η συλλογή δεδομένων αφορά μια ομάδα ατόμων που ενδιαφέρονται για αυτά τα δεδομένα. Ένα απλό παράδειγμα μιας βάσης δεδομένων μπορεί να είναι το σύνολο των φοιτητών ενός ιδρύματος, τα μαθήματα που παρακολουθεί ο κάθε φοιτητής και ο βαθμός που έχει καταχωρηθεί στο κάθε μάθημα.

Η ανάγκη για την διαχείριση αυτού του όγκου πληροφορίας δημιούργησε λογισμικά ειδικά σχεδιασμένα για αυτόν το σκοπό [2], που ονομάζονται DBMS (DataBase Management System). Ο σχεδιασμός ενός DBMS περιλαμβάνει διάφορους υποτομείς και τεχνικές προγραμματισμού που συνεργάζονται μεταξύ τους για το τελικό προϊόν. Το πρώτο DBMS με όνομα IDS (Integrated Data Storage) κατασκευάστηκε από τον Charles Bachman, στις αρχές του 1960. Σήμερα, πολύ σημαντικό ρόλο παίζουν τα DBMS στον παγκόσμιο ιστό, διότι όλο και περισσότερες ιστοσελίδες ανακτούν τα δεδομένα τους μέσω ενός DBMS.

Η χρήση ενός τέτοιου λογισμικού έχει πολλά προτερήματα σε σχέση με την απλή αποθήκευση δεδομένων σε αρχεία. Σύμφωνα με τους Ramakrishna και Gehrke [2] αυτά τα πλεονεκτήματα περιλαμβάνουν (επιγραμματικά) :

- Ανεξαρτησία δεδομένων από τα προγράμματα που τα χρησιμοποιούν.
- Γρήγορη πρόσβαση στα δεδομένα.
- Έλεγχος καταχωρήσεων και ασφάλεια.
- Διαχείριση δεδομένων σε περιπτώσεις όπου πολλοί χρήστες έχουν πρόσβαση στα ίδια δεδομένα.
- Επαναφορά από βλάβες.
- Ελαχιστοποίηση του χρόνου που απαιτείται για την ολοκλήρωση μιας εφαρμογής.

Όσο η δικτύωση των υπολογιστών επιτρέπει την διαθεσιμότητα πληροφοριών σε μεγάλο όγκο, τόσο μεγαλύτερη γίνεται και η ανάγκη για την διαχείριση της. Συνεπώς η σχεδίαση των DBMS είναι ένας τομέας που συνεχώς αναπτύσσεται.

2.2 Σχεδιασμός Βάσεων Δεδομένων

Προκειμένου να βρίσκεται σε θέση κανείς να κατανοήσει την χρησιμότητα του μοντέλου ER, θα πρέπει να αποκτήσει πρώτα μια γενική εικόνα της διαδικασίας που απαιτείται για να σχεδιαστεί μια εφαρμογή βάσεων δεδομένων. Όπως και στην γενική περίπτωση του σχεδιασμού λογισμικού, η διαδικασία μπορεί να χωριστεί σε κάποια στάδια ανάλογα με το είδος του λογισμικού που κατασκευάζεται. Σε αυτήν την υποενότητα θα γίνει μια σύντομη επεξήγηση αυτών των σταδίων.

Στάδιο 1, Ανάλυση Απαιτήσεων: Σε αυτό το στάδιο γίνεται η συλλογή πληροφοριών που αφορούν τις απαιτήσεις που έχουν από την εφαρμογή οι μελλοντικοί χρήστες της [2]. Δηλαδή, η εταιρία που έχει αναλάβει την ανάπτυξη της εφαρμογής πρέπει να καταγράψει τα δεδομένα που θα περιέχει η βάση δεδομένων, τον τύπο τους, και τον τρόπο με τον οποίο θα γίνεται διαχείριση αυτών των δεδομένων. Επίσης πρέπει σε αυτό το στάδιο να είναι γνωστές οι λειτουργίες που θα εκτελούνται συχνά, για λόγους απόδοσης. Αυτό το στάδιο περιλαμβάνει συνεννόηση με τους πελάτες ή διάφορες ομάδες χρηστών, και υπάρχουν πολλές μέθοδοι για να γίνει κάτι τέτοιο, μερικές από τις οποίες είναι αυτοματοποιημένες [2]. Είναι σημαντικό να τονίσουμε ότι οι απαιτήσεις αυτές πρέπει να είναι όσο το δυνατόν πιο σαφείς, για την επιτυχημένη σχεδίαση του λογισμικού [3].

Στάδιο 2, Εννοιολογικός ή Ιδεατός Σχεδιασμός: Αφότου η ανάλυση απαιτήσεων έχει ολοκληρωθεί, γίνεται να προχωρήσουμε στο επόμενο στάδιο. Το σύνολο των απαιτήσεων του λογισμικού που έχει στην διάθεσή της η εταιρία, θα χρησιμοποιηθεί για την περιγραφή σε υψηλό επίπεδο της βάσης δεδομένων, και των περιορισμών που πρόκειται να υπάρξουν [2]. Επίσης περιγράφονται και οι δομές δεδομένων που θα χρησιμοποιηθούν στην εφαρμογή, δηλαδή οι τρόποι με τους οποίους η εφαρμογή θα οργανώσει τα δεδομένα. Σε αυτήν τη φάση του σχεδιασμού παίζει σημαντικό ρόλο το μοντέλο ER, καθώς αποτελεί ένα υψηλού επιπέδου εννοιολογικό μοντέλο δεδομένων και είναι η πιο διαδεδομένη μέθοδος που χρησιμοποιείται συχνά. Σκοπός του μοντέλου ER είναι η γραφική απεικόνιση των δεδομένων που έχουν συλλεχθεί, με όσο πιο σαφή και απλό τρόπο γίνεται. Η περιγραφή αυτή βοηθάει τόσο τους προγραμματιστές αλλά και τους πελάτες / χρήστες να κατανοήσουν τις απαιτήσεις και να επικοινωνήσουν μεταξύ τους. Σημαντικό πλεονέκτημα της περιγραφής μέσω ER μοντέλου είναι το γεγονός ότι μπορεί να γίνει κατανοητή ακόμη και από άτομα που δεν διαθέτουν τεχνικές γνώσεις [2].

Στάδιο 3, Σχεδιασμός της Λογικής Βάσης Δεδομένων: Σε αυτό το στάδιο επιλέγεται το συγκεκριμένο DBMS που θα χρησιμοποιηθεί για την υλοποίηση της βάσης δεδομένων. Έπειτα, χρησιμοποιώντας την ιδεατή βάση δεδομένων που προκύπτει

από το στάδιο 2, δημιουργείται η αναπαράσταση της βάσης δεδομένων σύμφωνα με το DBMS που επιλέγεται [2]. Για την αναπαράσταση αυτή χρησιμοποιούνται σχέσεις (πίνακες) στις οποίες αποθηκεύονται τα δεδομένα με συγκεκριμένο τρόπο. Με άλλα λόγια, στην περίπτωση που χρησιμοποιηθεί το μοντέλο ER, το διάγραμμα ER μετατρέπεται σε σχήμα σχεσιακής (συνήθως) βάσης δεδομένων. Το αποτέλεσμα αυτής της μετατροπής ονομάζεται λογικό σχήμα [2].

Στάδιο 4, Τελειοποίηση Σχήματος: Στην συνέχεια, το λογικό σχήμα που προκύπτει μπορεί να βελτιωθεί περισσότερο, με μια λεπτομερέστερη ανάλυση. Αυτό το στάδιο ασχολείται με την αντιμετώπιση προβλημάτων που σχετίζονται με το σχεσιακό σχήμα που δημιουργήθηκε. Πιο συγκεκριμένα, γίνεται ανάλυση των πινάκων του σχεσιακού σχήματος με στόχο την κανονικοποίησή τους, και την τελειοποίηση του σχήματος [2].

Στάδιο 5, Σχεδιασμός Φυσικού Σχήματος: Σε αυτό το στάδιο μας ενδιαφέρει η αποδοτικότητα του συνολικού συστήματος. Για αυτόν το λόγο, εξετάζονται διάφορα σενάρια στα οποία η επεξεργασία της βάσης δεδομένων ενδέχεται να επιβαρύνει την απόδοση της εφαρμογής [2]. Για την αντιμετώπιση των επιβαρύνσεων μπορεί να δημιουργηθούν κάποια ευρετήρια ή ακόμα και να τροποποιηθούν κομμάτια του σχεσιακού σχήματος που έχει ήδη παραχθεί.

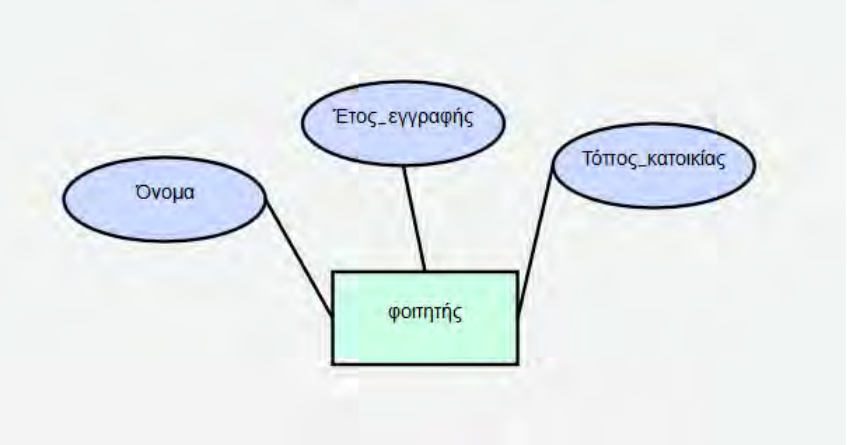
Στάδιο 6, Σχεδιασμός της Πολιτικής Προστασίας της Ιδιωτικότητας και των Λειτουργιών Εφαρμογής: Το συγκεκριμένο στάδιο αφορά όχι μόνο την βάση δεδομένων αλλά ολόκληρη την εφαρμογή. Αρχικά, πρέπει να γίνει καταγραφή των διαφόρων ομάδων χρηστών καθώς και των λειτουργιών της εφαρμογής. Για κάθε ομάδα χρηστών προσδιορίζονται τα μέρη της βάσης δεδομένων στα οποία επιτρέπεται η πρόσβαση, και οι επιτρεπόμενες λειτουργίες. Φυσικά, αφού ολοκληρωθεί αυτό το βήμα πρέπει να δημιουργηθούν κανόνες πρόσβασης και να τεθούν σε λειτουργία [2]. Ένα DBMS βοηθάει στην τήρηση αυτών των κανόνων, καθώς περιέχει έτοιμες λειτουργίες που αφορούν την ασφάλεια δεδομένων.

Μετά την ολοκλήρωση της διαδικασίας σχεδιασμού της εφαρμογής, περνάμε στο στάδιο της υλοποίησης όπου και διαλέγουμε την γλώσσα προγραμματισμού που θα χρησιμοποιηθεί σε συνδυασμό με το DBMS που επιλέχτηκε.

2.3 Μοντέλο Οντοτήτων - Συσχετίσεων

Το μοντέλο οντοτήτων - συσχετίσεων χρησιμοποιεί ως βασικά αντικείμενα τις οντότητες, τα γνωρίσματα τους, και τις συσχετίσεις μεταξύ οντοτήτων. Σε αυτήν την υποενότητα θα ασχοληθούμε με την επεξήγηση του μοντέλου.

2.3.1 Οντότητες και γνωρίσματα



Σχήμα 2.1: Οντότητα και γνωρίσματα

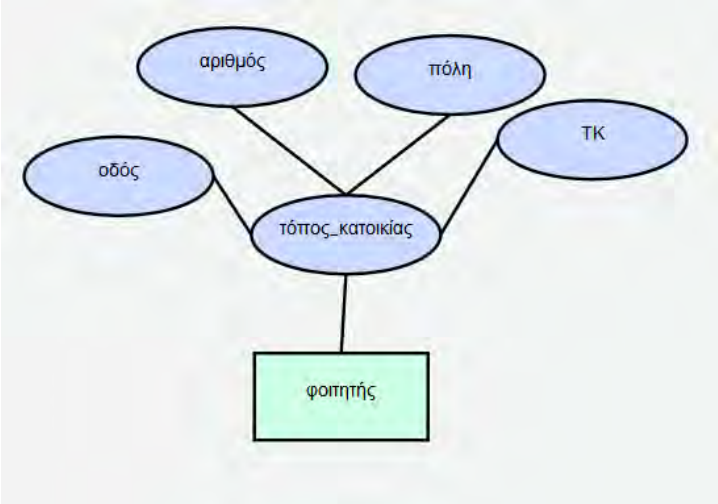
Το βασικότερο αντικείμενο του μοντέλου ER είναι η **οντότητα** (entity) , η οποία περιγράφει ένα αντικείμενο που είτε έχει φυσική υπόσταση στον κόσμο είτε όχι [1]. Για παράδειγμα, μια οντότητα μπορεί να είναι ο φοιτητής ενός πανεπιστημίου.

Μια οντότητα πρέπει να περιγράφεται από κάποιες ιδιότητες, τα **γνωρίσματά** της. Έχοντας πάλι ως παράδειγμα την οντότητα "φοιτητής", γνωρίσματα αυτής της οντότητας θα μπορούσαν να είναι: το όνομα, το έτος εγγραφής, και ο τόπος κατοικίας. Σε κάθε ένα από τα γνωρίσματα μιας οντότητας καταχωρείται μια τιμή, η οποία αντιπροσωπεύει κάποιο δεδομένο. Στο παράδειγμά μας, πιθανές τιμές για τα γνωρίσματα είναι 'Ειρήνη Παπαδοπούλου', '2011', 'Βόλος, Μαγνησία'. Υπάρχουν διάφορα είδη γνωρισμάτων, τα οποία θα αναλυθούν παρακάτω.

Όλες οι οντότητες που διαθέτουν κοινά γνωρίσματα αποτελούν έναν **τύπο οντοτήτων**. Για παράδειγμα, όλοι οι φοιτητές ενός πανεπιστημίου έχουν τα γνωρίσματα που αναφέρθηκαν παραπάνω και αποτελούν έναν τύπο οντοτήτων, όμως οι τιμές αυτών των γνωρισμάτων για την κάθε οντότητα είναι διαφορετικές. Επιπλέον, το σύνολο όλων των τύπων οντοτήτων μιας βάσης δεδομένων ονομάζεται **σύνολο οντοτήτων** [1]. Για παράδειγμα, εάν είχαμε μόνο τους δυο τύπους οντοτήτων 'φοιτητής' και 'πανεπιστήμιο' στην βάση δεδομένων, το σύνολο οντοτήτων αποτελείται από αυτούς τους τύπους. Στο μοντέλο ER, ο τύπος οντοτήτων περιγράφεται από ένα ορθογώνιο σχήμα, ενώ το γνώρισμα (attribute) με μια έλλειψη.

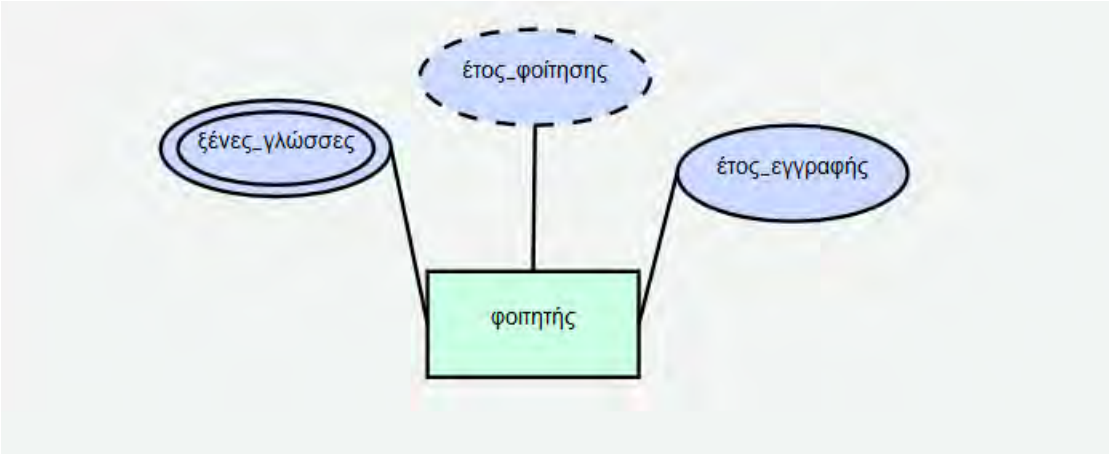
Απλά και σύνθετα γνωρίσματα: Με τον όρο **σύνθετο γνώρισμα** (composite attribute), εννοούμε ένα γνώρισμα που μπορεί να διαιρεθεί σε πιο βασικά γνωρίσματα, με ξεχωριστή έννοια το καθένα [1]. Σαν παράδειγμα θεωρήστε το γνώρισμα 'τόπος κατοικίας', όμως αυτή τη φορά διαιρείται στα εξής επιπλέον γνωρίσματα: Οδός, αριθμός, πόλη, ταχυδρομικός κώδικας. Αντίστοιχα, οι τιμές θα μπορούσαν να είναι: 'Ιάσονος', '26', 'Βόλος', '38221'. Τα σύνθετα γνωρίσματα χρησιμεύουν σε περιπτώσεις που χρειάζεται πρόσβαση είτε σε ολόκληρο το γνώρισμα, είτε σε ξεχωριστά του τμήματα [1].

Απλά γνωρίσματα ονομάζονται αυτά που δεν έχουν υποδιαιρέσεις, δηλαδή τα γνωρίσματα που δεν είναι σύνθετα.



Σχήμα 2.2 : Σύνθετο γνώρισμα

Μονότιμα και πλειότιμα γνωρίσματα: Μονότιμο (single-valued) γνώρισμα ονομάζεται το γνώρισμα που αποθηκεύει μόνο μία τιμή. Για παράδειγμα το έτος εγγραφής ενός φοιτητή παίρνει μια μοναδική τιμή. **Πλειότιμο** γνώρισμα (multivalued attribute) είναι ένα γνώρισμα που μπορεί να περιέχει μία ή παραπάνω τιμές [1]. Για παράδειγμα, ένα γνώρισμα με όνομα 'ξένες γλώσσες' που περιγράφει τις ξένες γλώσσες που έχει μάθει ένας φοιτητής, μπορεί να περιέχει μόνο την τιμή 'Αγγλικά' ή ίσως και παραπάνω τιμές. Ένα πλειότιμο γνώρισμα μπορεί να έχει άνω και κάτω όρια, τα οποία προσδιορίζουν τον αριθμό των τιμών που μπορούν να πάρουν [1]. Το πλειότιμο γνώρισμα παριστάνεται με διπλή γραμμή στο περίγραμμα.



Σχήμα 2.3: Πλειότιμο γνώρισμα, αποθηκευμένο και παραγόμενο γνώρισμα.

Αποθηκευμένα και παραγόμενα γνωρίσματα: Παραγόμενο (derived) γνώρισμα ορίζεται εκείνο το γνώρισμα η τιμή του οποίου μπορεί να παραχθεί χρησιμοποιώντας ένα άλλο **αποθηκευμένο** (stored) γνώρισμα [1]. Για παράδειγμα, το γνώρισμα 'έτος φοίτησης' της οντότητας 'φοιτητής', μπορεί να παραχθεί αφαιρώντας την τιμή του γνωρίσματος 'έτος εγγραφής' από την σημερινή χρονολογία. Πολλές φορές το παραγόμενο γνώρισμα μπορεί να εξαρτάται από ένα αποθηκευμένο γνώρισμα διαφορετικής αλλά συσχετιζόμενης οντότητας. Το παραγόμενο γνώρισμα παριστάνεται με διακεκομμένη γραμμή στο περίγραμμα όπως φαίνεται στο σχήμα 2.3.

Γνωρίσματα NULL: Μερικές φορές τυχαίνει η τιμή κάποιου γνωρίσματος να μην μπορεί να προσδιοριστεί [1]. Για παράδειγμα το γνώρισμα 'ξένες γλώσσες' που περιγράφει τις ξένες γλώσσες που έχει μάθει κάποιος ισχύει μόνο για άτομα που έχουν γνώσεις ξένων γλωσσών. Μια άλλη περίπτωση είναι να υπάρχει τιμή αλλά να μην είναι γνωστή, για παράδειγμα να μην γνωρίζουμε την διεύθυνση ηλεκτρονικού ταχυδρομείου ενός συγκεκριμένου ατόμου ώστε να την καταχωρήσουμε στο γνώρισμα 'email address'. Στις δυο αυτές περιπτώσεις δημιουργείται η τιμή NULL η οποία σημαίνει 'δεν εφαρμόζεται' ή 'άγνωστο' [1].

2.3.2 Κλειδιά και πεδίο ορισμού

Συνήθως κάποιος τύπος οντοτήτων έχει ένα γνώρισμα (ή περισσότερα) η τιμή του οποίου είναι μοναδική για κάθε οντότητα που ανήκει στον τύπο αυτό. Η τιμή ενός τέτοιου γνωρίσματος μπορεί να χρησιμοποιηθεί για να προσδιορίσει μια οντότητα με μοναδικό τρόπο, και αυτό το γνώρισμα ονομάζεται **κλειδί** [1]. Όταν ένα γνώρισμα είναι κλειδί, εισάγει έναν περιορισμό που αναγκάζει όλες τις οντότητες που ανήκουν στον ίδιο τύπο να έχουν διαφορετική τιμή στο γνώρισμα αυτό. Για παράδειγμα, στον τύπο οντοτήτων 'φοιτητής' μπορεί να προστεθεί το γνώρισμα 'ΑΕΜ' που δηλώνει τον αριθμό μητρώου του φοιτητή. Αυτό το γνώρισμα αποτελεί κλειδί διότι δεν επιτρέπεται δυο φοιτητές να έχουν ίδιο αριθμό ΑΕΜ. Σε κάποιες περιπτώσεις όπου ένα σύνολο γνωρισμάτων χρησιμοποιείται για την διάκριση μια οντότητας από τις υπόλοιπες, τότε πρέπει το σύνολο των γνωρισμάτων να ανήκει σε ένα σύνθετο γνώρισμα [1]. Είναι δυνατό ένας τύπος οντοτήτων να διαθέτει παραπάνω από ένα κλειδί, ή καθόλου κλειδιά (ασθενής οντότητα) όπως θα δούμε παρακάτω. Σε ένα διάγραμμα ER, ένα γνώρισμα-κλειδί έχει υπογραμμισμένο όνομα.

Κάθε απλό γνώρισμα (δηλαδή το γνώρισμα που δεν διαιρείται) μπορεί να πάρει κάποιες συγκεκριμένες τιμές. Το σύνολο αυτών των τιμών αποτελεί το **πεδίο ορισμού** του γνωρίσματος [1]. Για παράδειγμα, το γνώρισμα 'έτος εγγραφής' του τύπου οντοτήτων 'φοιτητής' μπορεί να πάρει τιμές από '2000' έως '2017', θεωρώντας ότι το

πανεπιστήμιο ιδρύθηκε το έτος 2000. Στα διαγράμματα ER δεν παριστάνεται αυτή η πληροφορία αλλά μπορεί να διευκρινιστεί στο στάδιο υλοποίησης χρησιμοποιώντας μια γλώσσα προγραμματισμού και κατάλληλους τύπους δεδομένων [1].

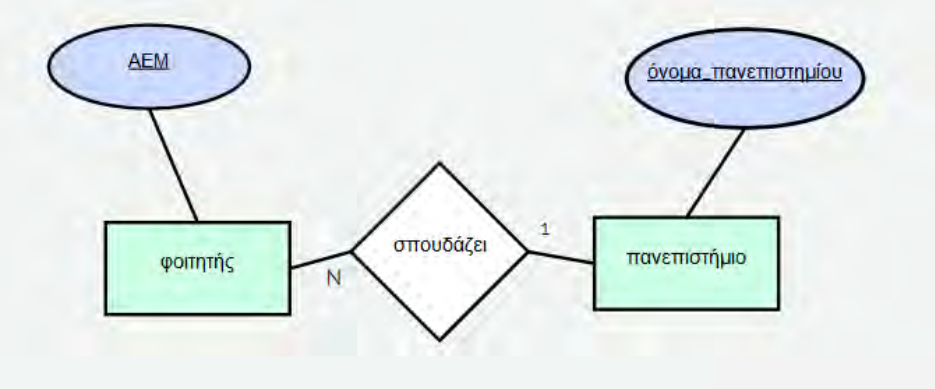
2.3.3 Συσχετίσεις

Στο μοντέλο ER, όποτε κάποιο γνώρισμα ενός τύπου οντοτήτων αναφέρεται σε κάποιον άλλο τύπο οντοτήτων, τότε λέμε πως υπάρχει μια συσχέτιση [1]. Για παράδειγμα, εάν έχουμε τους τύπους οντοτήτων 'φοιτητής' και 'πανεπιστήμιο', και ο τύπος οντοτήτων 'φοιτητής' έχει ένα γνώρισμα με όνομα 'πανεπιστήμιο' που δηλώνει σε ποιο πανεπιστήμιο ανήκει, τότε υπάρχει συσχέτιση μεταξύ των δυο αυτών τύπων οντοτήτων. Συνεπώς, είναι πιο σωστό να μην αναπαρασταθεί αυτή η πληροφορία χρησιμοποιώντας ένα γνώρισμα, αλλά μια συσχέτιση.

Ένας **τύπος συσχέτισης** (relationship type) R [1], μεταξύ n τύπων οντοτήτων E_1, E_2,...,E_n ορίζει ένα σύνολο συσχετίσεων ανάμεσα στις οντότητες αυτών των τύπων οντοτήτων. Χρησιμοποιώντας μαθηματική διατύπωση, το σύνολο R είναι ένα σύνολο που αποτελείται από i συσχετίσεις r_i όπου κάθε r_i συσχετίζει n ξεχωριστές οντότητες (e_1, e_2,...,e_n). Κάθε οντότητα e_j στο r_i ανήκει στον τύπο οντοτήτων E_j, όπου $1 \leq j \leq n$.

Στα διαγράμματα ER, η συσχέτιση παριστάνεται ως ένας ρόμβος, και συνδέονται με τις οντότητες που συμμετέχουν στην συσχέτιση μέσω ευθύγραμμων τμημάτων.

Βαθμός ενός τύπου συσχέτισης: Βαθμός ενός τύπου συσχέτισης [1] ονομάζεται ο αριθμός των τύπων οντοτήτων που συμμετέχουν στην συσχέτιση. Μια συσχέτιση με βαθμό 2 ονομάζεται δυαδική, ενώ μια συσχέτιση με βαθμό 3 ονομάζεται τριαδική. Ένας τύπος συσχέτισης μπορεί να συσχετίζει οποιοδήποτε αριθμό οντοτήτων μεταξύ τους, όμως οι πιο συχνές είναι οι δυαδικές συσχετίσεις, καθώς εκείνες που έχουν μεγαλύτερο βαθμό είναι αρκετά πολύπλοκες.



Σχήμα 2.4: Δυαδική συσχέτιση

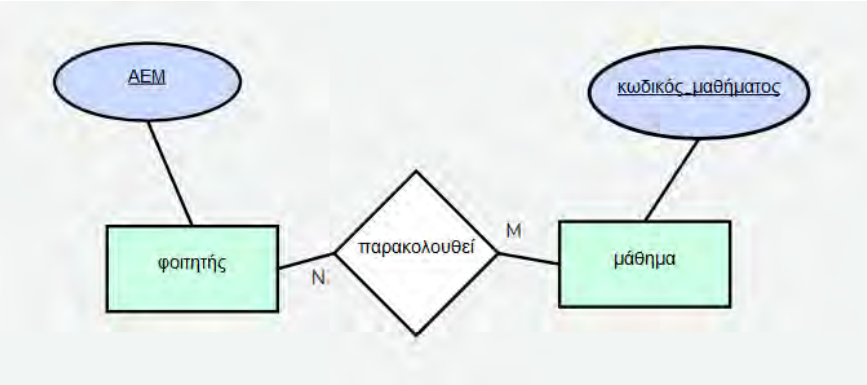
Συσχέτιση ως γνώρισμα: Σε περιπτώσεις που θέλουμε να αναπαραστήσουμε μια δυαδική συσχέτιση με την μορφή γνωρίσματος [1], υπάρχουν δυο τρόποι να γίνει αυτό. Σαν παράδειγμα θεωρούμε την περίπτωση του σχήματος 2.4. Ο πρώτος τρόπος είναι να δημιουργήσουμε ένα γνώρισμα με όνομα 'πανεπιστήμιο' που ανήκει στον τύπο οντοτήτων 'φοιτητής', και το πεδίο ορισμού του γνωρίσματος να περιλαμβάνει το σύνολο οντοτήτων του τύπου οντοτήτων 'πανεπιστήμιο'. Ο δεύτερος τρόπος είναι να δημιουργήσουμε ένα **πλειότιμο** γνώρισμα 'φοιτητές' (επειδή σε κάθε πανεπιστήμιο φοιτούν πολλά άτομα), που ανήκει στον τύπο οντοτήτων 'πανεπιστήμιο' και παίρνει τιμές από το σύνολο των οντοτήτων των φοιτητών. Αυτοί οι δυο τρόποι είναι ισοδύναμοι.

Αναδρομικές συσχετίσεις: Κάθε τύπος οντοτήτων που συμμετέχει σε μια συσχέτιση έχει κάποιο συγκεκριμένο ρόλο [1], και του δίνεται ένα όνομα ρόλου (role name) για να διευκρινιστεί η λειτουργία του. Συνήθως το όνομα ρόλου παραλείπεται επειδή ο ρόλος κάθε οντότητας είναι προφανής. Σε κάποιες περιπτώσεις όμως, ο ίδιο τύπος οντοτήτων συμμετέχει δυο φορές στην ίδια συσχέτιση, και οι ρόλοι πρέπει να διευκρινίζονται. Επεκτείνοντας το παράδειγμα του φοιτητή, ας θεωρήσουμε ότι το πανεπιστήμιο αναθέτει σε κάποιον φοιτητή μεγάλου έτους ή μεταπτυχιακό να βοηθήσει κάποιον πρωτοετή φοιτητή με τα μαθήματα του πρώτου εξαμήνου. Έτσι δημιουργείται μια συσχέτιση (ας της δώσουμε όνομα 'βοηθός') που συσχετίζει τον τύπο οντοτήτων 'φοιτητής' με τον εαυτό του. Από την μια πλευρά ο τύπος οντοτήτων 'φοιτητής' συμμετέχει ως βοηθός, ενώ από την άλλη πλευρά συμμετέχει ως 'πρωτοετής φοιτητής'.

2.3.4 Περιορισμοί Συσχετίσεων

Υπάρχουν δυο είδη περιορισμών όσον αφορά τις συσχετίσεις, που επηρεάζουν τον τρόπο με τον οποίο οι οντότητες συμμετέχουν σε μία συσχέτιση. Για λόγους απλότητας, θα ασχοληθούμε με δυαδικές συσχετίσεις.

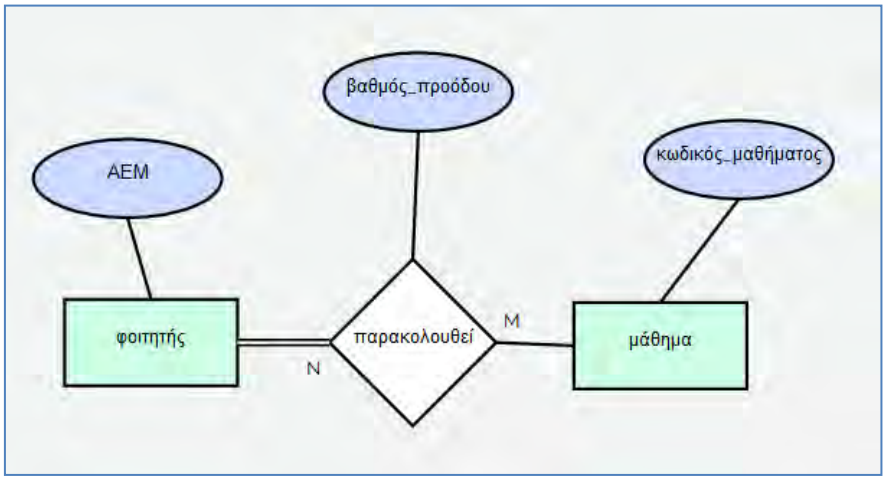
Λόγος πληθικότητας: Ο λόγος πληθικότητας (cardinality ratio) [1] σε μια δυαδική συσχέτιση δηλώνει τον μέγιστο αριθμό στιγμιότυπων συσχέτισης στα οποία έχει την δυνατότητα να ανήκει μια οντότητα. Αυτό γίνεται καλύτερα κατανοητό με ένα παράδειγμα. Στο σχήμα 2.4 ο τύπος συσχετίσεων έχει λόγο πληθικότητας N:1 (φοιτητές προς πανεπιστήμια) και αυτό σημαίνει ότι μια οντότητα του τύπου οντοτήτων 'πανεπιστήμιο' μπορεί να συσχετιστεί με οποιοδήποτε αριθμό οντοτήτων του τύπου οντοτήτων 'φοιτητής'. Με άλλα λόγια πολλοί φοιτητές σπουδάζουν σε ένα πανεπιστήμιο. Ο λόγος πληθικότητας μπορεί να είναι 1:1, 1:N (ή N:1), N:M. Όταν έχουμε λόγο πληθικότητας 1:1 σημαίνει ότι μία οντότητα του ενός τύπου οντοτήτων συσχετίζεται με ακριβώς μία οντότητα του άλλου τύπου. Λόγος πληθικότητας N:M δηλώνει ότι πολλές οντότητες του ενός τύπου μπορούν να συσχετιστούν με πολλές οντότητες του άλλου τύπου. Για παράδειγμα στο σχήμα 2.5 βλέπουμε ότι πολλοί φοιτητές μπορούν να παρακολουθούν πολλά μαθήματα.



Σχήμα 2.5: Λόγος πληθικότητας

Ο λόγος πληθικότητας που θα επιλεχτεί εξαρτάται από τις οντότητες και τι αντιπροσωπεύουν αυτές στον πραγματικό κόσμο. Στο διάγραμμα ER περιγράφεται με την ύπαρξη του χαρακτήρα '1', 'N', ή 'M' στον ρόμβο, δίπλα στην αντίστοιχη γραμμή σύνδεσης της συσχέτισης.

Περιορισμός συμμετοχής: Ο δεύτερος περιορισμός που ονομάζεται περιορισμός συμμετοχής (participation) [1], δηλώνει τον ελάχιστο αριθμό οντοτήτων που πρέπει να συμμετέχουν στα στιγμιότυπα ενός τύπου συσχετίσεων. Η συμμετοχή ενός τύπου οντοτήτων μπορεί να είναι είτε ολική, είτε μερική. Μια **ολική** συμμετοχή δείχνει ότι όλες οι οντότητες ενός τύπου οντοτήτων πρέπει υποχρεωτικά να συμμετέχουν στον τύπο συσχέτισης. Για παράδειγμα, θεωρήστε ότι κάθε οντότητα του τύπου 'φοιτητής' πρέπει να παρακολουθεί τουλάχιστον ένα μάθημα για να ανήκει στους φοιτητές. Σε αυτήν την περίπτωση η συμμετοχή του τύπου οντοτήτων 'φοιτητής' είναι ολική, και συμβολίζεται με διπλή γραμμή στην σύνδεση (σχήμα 2.6). Αντίθετα, εάν δεν χρειάζεται όλοι οι φοιτητές να παρακολουθούν απαραίτητα κάποιο μάθημα, η συμμετοχή είναι **μερική** και η αντίστοιχη γραμμή σύνδεσης στο διάγραμμα θα ήταν μονή.



Σχήμα 2.6: Συμμετοχή

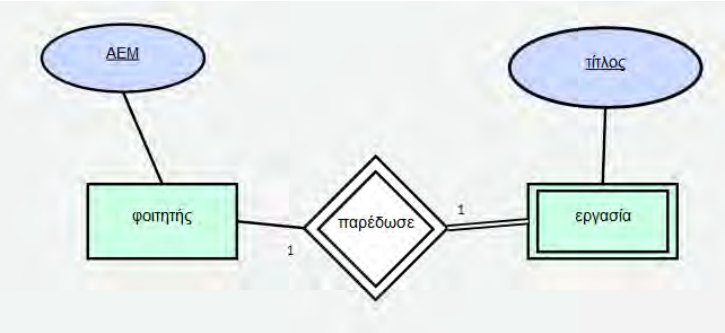
Η ολική συμμετοχή δηλώνει ότι μια οντότητα 'φοιτητής' δεν μπορεί να υπάρχει εάν δεν συμμετέχει σε κάποιο στιγμιότυπο του τύπου συσχέτισης 'παρακολουθεί'. Γι' αυτό και η ολική συμμετοχή ονομάζεται και **περιορισμός ύπαρξης**.

Γνωρίσματα συσχέτισης: Ένας τύπος συσχέτισης μπορεί να έχει γνωρίσματα [1], όπως και ένας τύπος οντοτήτων. Για παράδειγμα στο σχήμα 2.6 ο τύπος συσχέτισης 'παρακολουθεί' έχει γνώρισμα με όνομα 'βαθμός προόδου' που δηλώνει τον βαθμό που πήρε ένας συγκεκριμένος φοιτητής σε ένα συγκεκριμένο μάθημα.

2.3.5 Ασθενείς Οντότητες

Ένας τύπος οντοτήτων ονομάζεται **ασθενής** [1], όταν δεν έχει δικά του γνωρίσματα - κλειδιά και επομένως δεν γίνεται να ξεχωρίσουμε τις οντότητες του. Αυτή η περίπτωση αντιμετωπίζεται με το να συσχετίσουμε έναν ασθενή τύπο οντοτήτων με έναν **ισχυρό** τύπο οντοτήτων (κανονικές οντότητες που μπορούν να προσδιοριστούν μοναδικά από μόνες τους). Για να προσδιοριστούν οι οντότητες του ασθενή τύπου χρησιμοποιούμε έναν συνδυασμό από ένα γνώρισμα του ασθενή τύπου και το κλειδί του ισχυρού τύπου. Το είδος συσχέτισης που συνδέει τον ασθενή τύπο οντοτήτων με τον ισχυρό, ονομάζεται **προσδιορίζουσα** συσχέτιση. Επίσης ο ισχυρός τύπος οντοτήτων που συμμετέχει στην συσχέτιση ονομάζεται **ιδιοκτήτης**. Συνήθως επιλέγεται κάποιο γνώρισμα του ασθενή τύπου οντοτήτων για να χρησιμοποιηθεί στον συνδυασμό ταυτοποίησης, και ονομάζεται **μερικό κλειδί**.

Καθώς μια ασθενής οντότητα δεν είναι δυνατό να προσδιοριστεί από μόνη της, πρέπει υποχρεωτικά να συμμετέχει στην προσδιορίζουσα συσχέτιση. Οπότε ένας ασθενής τύπος οντοτήτων έχει πάντα ολική συμμετοχή στην συσχέτιση.



Σχήμα 2.7: Ασθενής οντότητα, προσδιορίζουσα συσχέτιση

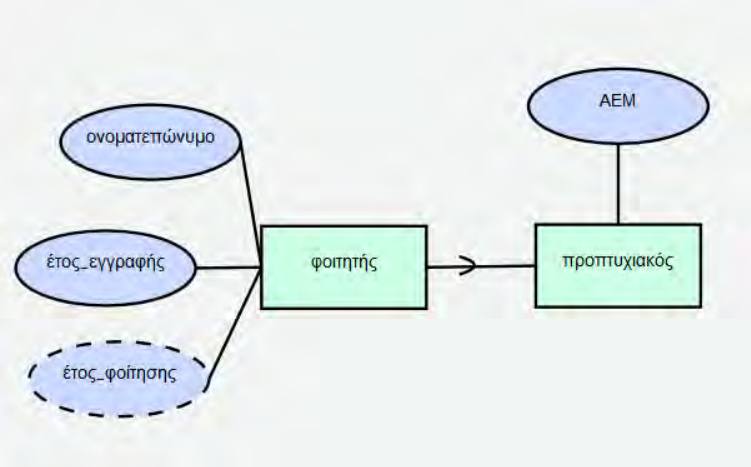
Στο σχήμα 2.7 η ασθενής οντότητα 'εργασία' δεν μπορεί να προσδιοριστεί από το γνώρισμα 'τίτλος' επειδή μπορεί δυο εργασίες να έχουν τον ίδιο τίτλο. Επομένως οι οντότητες αυτού του τύπου συσχετίζονται με τον ισχυρό τύπο οντοτήτων 'φοιτητής' ώστε να μπορούν να προσδιοριστούν με μοναδικό τρόπο.

2.4 Εκτεταμένο Μοντέλο Οντοτήτων - Συσχετίσεων

Το μοντέλο ER που εξηγήθηκε στην ενότητα 2.3 επαρκεί για τον σχεδιασμό των περισσότερων απλών βάσεων δεδομένων. Σε αυτήν την ενότητα θα δούμε το εκτεταμένο μοντέλο Οντοτήτων - Συσχετίσεων, το οποίο αποτελεί επέκταση του απλού μοντέλου και περιέχει αντικείμενα που προσθέτουν νέες δυνατότητες στον σχεδιασμό.

2.4.1 Υπερκλάσεις

Η πρώτη σημαντική έννοια του εκτεταμένου μοντέλου είναι η έννοια της υπερκλάσης [1]. Ένας τύπος οντοτήτων αποτελείται από όλες τις οντότητες που μοιράζονται τα ίδια γνωρίσματα και συσχετίσεις, όμως σε κάποιες περιπτώσεις είναι χρήσιμο να υπάρχουν υποομάδες κάποιου τύπου οντοτήτων που πρέπει να συμπεριληφθούν στον σχεδιασμό. Για παράδειγμα, ο τύπος οντοτήτων 'φοιτητής' μπορεί να διακρίνεται σε 'προπτυχιακός'. Αυτή η ομάδα είναι υποσύνολο του συνόλου οντοτήτων 'φοιτητής', και ονομάζεται **υποκλάση**, ενώ ο τύπος οντοτήτων 'φοιτητής' ονομάζεται **υπερκλάση**. Επειδή η κάθε οντότητα μιας υποκλάσης ανήκει και στην υπερκλάση, εισάγεται η έννοια της **κληρονομικότητας τύπου**. Μια οντότητα που ανήκει σε υποκλάση κληρονομεί τα γνωρίσματα της υπερκλάσης και τις συσχετίσεις στις οποίες ανήκει η υπερκλάση.

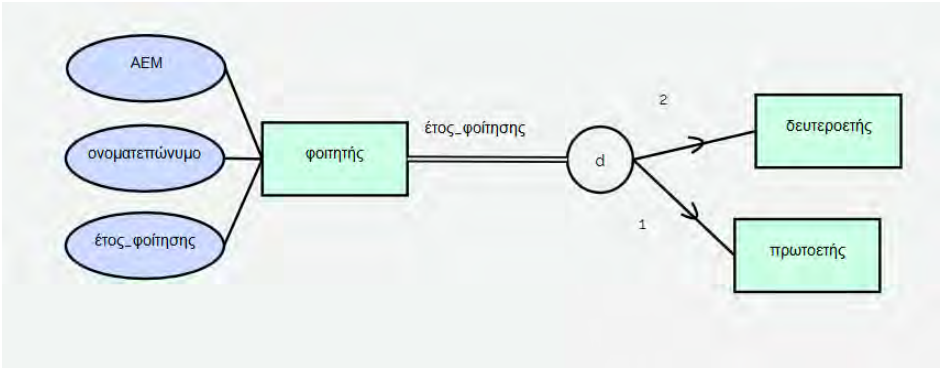


Σχήμα 2.8: Υπερκλάση και υποκλάση

Η σχέση υπερκλάσης - υποκλάσης σε ένα διάγραμμα παριστάνεται με την σύνδεση των δυο τύπων οντοτήτων με ευθύγραμμο τμήμα και το σύμβολο υποσυνόλου που δείχνει προς την υποκλάση.

2.4.2 Εξειδικεύσεις και Γενικεύσεις

Εξειδίκευση [1] ονομάζεται η διαδικασία δημιουργίας ενός συνόλου υποκλάσεων ενός τύπου οντοτήτων. Η διάκριση αυτή βασίζεται σε κάποιο χαρακτηριστικό της υπερκλάσης. Για παράδειγμα οι τύποι οντοτήτων 'προπτυχιακός' και 'μεταπτυχιακός' είναι υποκλάσεις του τύπου 'φοιτητής' και η διάκριση γίνεται με βάση το είδος σπουδών. Οι φοιτητές θα μπορούσαν να διακριθούν επίσης και σε 'πρωτοετής', 'δευτεροετής', ανάλογα με την τιμή του γνωρίσματος 'έτος φοίτησης' της υπερκλάσης. Δεν χρειάζεται να χρησιμοποιηθεί εξειδίκευση όταν πρόκειται να δημιουργήσουμε μόνο μια υποκλάση.



Σχήμα 2.9: Εξειδίκευση

Στο σχήμα 2.9 βλέπουμε την αναπαράσταση της εξειδίκευσης σε διάγραμμα ER. Η υπερκλάση συνδέεται με έναν κύκλο, και οι υποκλάσεις συνδέονται με τον κύκλο, χρησιμοποιώντας το σύμβολο του υποσυνόλου.

Το είδος του ευθύγραμμου τμήματος που συνδέει την υπερκλάση με τον κύκλο δηλώνει **μερική** (απλή γραμμή) ή **ολική** (διπλή γραμμή) εξειδίκευση. Η ολική εξειδίκευση σημαίνει ότι όλες οι οντότητες του τύπου οντοτήτων 'φοιτητής' πρέπει να ανήκουν σε μια τουλάχιστον υποκλάση. Στο κέντρο του κύκλου τοποθετείται το γράμμα 'd' ή 'o' που προέρχονται από τις λέξεις disjoint και overlapping αντίστοιχα. Όταν μια εξειδίκευση χαρακτηρίζεται ως **επικαλυπτόμενη** (overlapping) σημαίνει ότι οντότητες από την μια υποκλάση της εξειδίκευσης μπορούν να ανήκουν και σε άλλες υποκλάσεις (της ίδιας εξειδίκευσης). Η **μη επικαλυπτόμενη** εξειδίκευση διακρίνει την υπερκλάση σε υποκλάσεις που δεν περιέχουν κοινές οντότητες μεταξύ τους, είναι δηλαδή ξένα σύνολα.

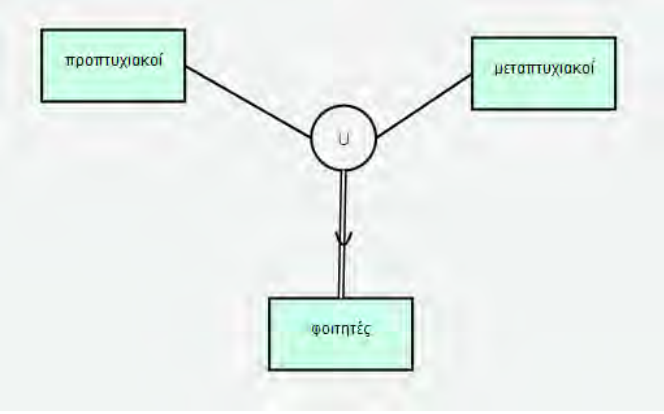
Όταν η διάκριση των υποκλάσεων γίνεται βάσει κάποιο γνωρίσματος της υπερκλάσης, τότε μπορούμε (προαιρετικά) να προσθέσουμε αυτήν την συνθήκη στο διάγραμμα βάζοντας το όνομα του γνωρίσματος κοντά στην γραμμή που ενώνει την υπερκλάση με τον κύκλο (σχήμα 2.9). Το γνώρισμα αυτό ονομάζεται **ορίζον γνώρισμα** (defining attribute). Στην συνέχεια πρέπει να ορίσουμε την τιμή που πρέπει να πάρει αυτό το γνώρισμα, ώστε η οντότητα να ανήκει στην αντίστοιχη υποκλάση. Στο σχήμα 2.9, για να ανήκει ένας φοιτητής στους πρωτοετείς φοιτητές πρέπει η τιμή του γνωρίσματος 'έτος φοίτησης' να είναι ίση με 1, με παρόμοιο τρόπο διακρίνονται και οι δευτεροετείς φοιτητές.

Η αντίστροφη διαδικασία της εξειδίκευσης [1] ονομάζεται **γενίκευση**, κατά την οποία αναγνωρίζουμε κοινά γνωρίσματα που έχουν κάποιοι τύπου οντοτήτων και δημιουργούμε μια υπερκλάση με αυτά τα κοινά γνωρίσματα, στην οποία ανήκουν ως υποκλάσεις οι πρώτοι τύποι οντοτήτων. Για παράδειγμα, εάν είχαμε τους τύπους οντοτήτων 'πρωτοετής φοιτητής', 'δευτεροετής φοιτητής' και ο κάθε τύπος είχε τα γνωρίσματα 'ΑΕΜ', 'ονοματεπώνυμο', 'έτος φοίτησης', θα ήταν δυνατό να δημιουργήσουμε την υπερκλάση 'φοιτητής' με αυτά τα γνωρίσματα και να καταλήξουμε πάλι στο σχήμα 2.9.

2.4.3 Ενώσεις

Μερικές φορές στον σχεδιασμό μιας βάσης δεδομένων, χρειάζεται ένας τύπος δεδομένων να κληρονομεί χαρακτηριστικά από δυο ή και παραπάνω τύπους οντοτήτων. Αυτή είναι μια περίπτωση διαφορετική από την μοναδική υπερκλάση που είδαμε στην ενότητα 2.4.1 και την εξειδίκευση που είδαμε στην ενότητα 2.4.2. Όταν λοιπόν ένας τύπος δεδομένων έχει παραπάνω από μια υπερκλάσεις [1], ονομάζουμε τον τύπο αυτό **ένωση** (union) ή **κατηγορία**. Αυτός ο τύπος οντοτήτων αποτελεί υποσύνολο της ένωσης των οντοτήτων των υπερκλάσεων του. Θεωρήστε ότι χρειάζεται ένας τύπος οντοτήτων που περιέχει το σύνολο των φοιτητών που είναι εγγεγραμμένοι σε ένα μάθημα. Οι φοιτητές μπορεί να είναι είτε προπτυχιακοί είτε μεταπτυχιακοί, οπότε θέλουμε να περιλαμβάνεται η ένωση των δυο αυτών τύπων οντοτήτων στον τύπο οντοτήτων 'φοιτητές'.

Στο μοντέλο ER η ένωση παριστάνεται με έναν κύκλο που περικλείει το γράμμα 'U'. Ο κύκλος συνδέεται με τις υπερκλάσεις μέσω ευθύγραμμου τμήματος. Η υποκλάση συνδέεται με τον κύκλο χρησιμοποιώντας απλή ή διπλή γραμμή (μερική ή ολική συμμετοχή) και το σύμβολο του υποσυνόλου. Μια ολική ένωση περιέχει ολόκληρο το σύνολο οντοτήτων της ένωσης των υπερκλάσεων, ενώ μια μερική ένωση περιέχει ένα υποσύνολο της ένωσης.



Σχήμα 2.10: Ένωση

2.5 Σχεσιακό Μοντέλο και PostgreSQL

2.5.1 Σχεσιακό Μοντέλο

Το σχεσιακό μοντέλο δεδομένων δημιουργήθηκε από τον Codd το 1970 [2], και γνώρισε μεγάλη επιτυχία, ξεπερνώντας γρήγορα τα μοντέλα που βρίσκονταν σε χρήση εκείνη την εποχή. Το σχεσιακό μοντέλο δεδομένων διαθέτει πλεονεκτήματα που ευθύνονται για το γεγονός ότι είναι πολύ διαδεδομένο, όμως το βασικό του πλεονέκτημα είναι ότι μπορεί να περιγραφεί με μαθηματικό τρόπο, χρησιμοποιώντας την Θεωρία Συνόλων [4]. Ακόμη ένα σημαντικό πλεονέκτημα αποτελεί η απλή και εύκολα κατανοητή δομή του. Το σχεσιακό μοντέλο δημιουργήθηκε έχοντας τους εξής στόχους [4]:

- Ανεξαρτησία δεδομένων από την εφαρμογή που τα χρησιμοποιεί. Αυτό σημαίνει πως διάφορες αλλαγές στην οργάνωση της βάσης δεδομένων δεν επηρεάζουν την εφαρμογή που έχει πρόσβαση στην βάση δεδομένων.

- Να μην υπάρχει πλεονασμός δεδομένων, σε περιπτώσεις που τα ίδια δεδομένα αποθηκεύονται πολλές φορές.

- Ύπαρξη και διατήρηση ακεραιότητας δεδομένων.

- Η δυνατότητα ανάπτυξης γλωσσών που χρησιμοποιούν την θεωρία συνόλων για να διαχειριστούν δεδομένα, με αποτέλεσμα την διευκόλυνση διατύπωσης ερωτημάτων προς το DBMS.

Μετά την άφιξη του σχεσιακού μοντέλου δημιουργήθηκαν διάφορα DBMS που βασίζονται σε αυτό, βοηθώντας με αυτόν το τρόπο ακόμη περισσότερο στην διάδοσή του μοντέλου. Συνεπώς, σήμερα έχουν δημιουργηθεί σχεσιακά συστήματα που βασίζονται σε αυτό ή σε επεκτάσεις του και προσφέρουν πολλές δυνατότητες και ακόμη μεγαλύτερη ευκολία στην χρήση [4].

Το βασικό αντικείμενο του σχεσιακού μοντέλου είναι η **σχέση** (ή πίνακας). Ο κάθε πίνακας έχει έναν αριθμό γραμμών που ονομάζονται **πλειάδες** (tuples), και έναν αριθμό στηλών που ονομάζονται **γνωρίσματα** ή χαρακτηριστικά (attributes). Σαν ένα απλό παράδειγμα θα χρησιμοποιήσουμε το σχήμα 6.1 του βιβλίου "*Συστήματα Βάσεων Δεδομένων*" των Μανωλόπουλος Ι. και Παπαδόπουλος Α.

κωδικός	*όνομα*	*τηλέφωνο*	*διεύθυνση*	*ΑΠΚ*
12	Γιώργος	6977333222	Π. Κορομηλά 12	12345
44	Μαρία	6945123456	Β. Όλγας 13	54321
55	Αλέξανδρος	6937222999	Γ. Λαμπράκη	11111

Ο πίνακας αυτός αποτελεί στιγμιότυπο της σχέσης 'Συνδρομητής'. Ο αριθμός των στηλών ονομάζεται **βαθμός** (degree) και ο αριθμός των γραμμών (εγγραφών) ονομάζεται **πληθικότητα** (cardinality).

Επιπλέον, κάθε γνώρισμα μπορεί να πάρει τιμές που προέρχονται από ένα σύνολο τιμών, το πεδίο ορισμού του γνωρίσματος. Για να είναι δυνατό όμως να προσδιοριστεί με ακρίβεια το πεδίο ορισμού ενός γνωρίσματος, πρέπει να είναι γνωστός ο τύπος δεδομένων που περιέχει, και η μορφοποίησή του (format). Για κάθε σχέση, πρέπει το πεδίο ορισμού κάθε γνωρίσματος να είναι ατομικό, δηλαδή οι τιμές του πεδίου να θεωρούνται αδιαίρετες μονάδες [4]. Το αν θα θεωρήσουμε τις τιμές αδιαίρετες μονάδες ή όχι, εξαρτάται από την ερμηνεία που θα τους δοθεί.

Είναι σημαντικό να διευκρινιστεί η διαφορά μεταξύ **σχήματος** βάσης δεδομένων (schema) και **στιγμιότυπου** (instance). Το σχήμα της βάσης δεδομένων είναι αποτέλεσμα της διαδικασίας λογικού σχεδιασμού, ενώ το στιγμιότυπο αποτελεί το σύνολο όλων των δεδομένων που βρίσκονται καταχωρημένα στην βάση δεδομένων κάποια συγκεκριμένη στιγμή [4].

Είναι σημαντικό να αναφέρουμε μερικές ιδιότητες των σχέσεων, για καλύτερη κατανόηση του σχεσιακού μοντέλου [4]:

- Οι σχέσεις (πίνακες) έχουν μοναδικά ονόματα, δηλαδή δεν επιτρέπεται δυο σχέσεις να έχουν το ίδιο όνομα.

- Τα γνωρίσματα μπορούν να λάβουν μόνο ατομικές τιμές.

- Γνωρίσματα που ανήκουν στην ίδια σχέση έχουν μοναδικά ονόματα.

- Δυο πλειάδες της ίδιας σχέσης, δεν πρέπει να έχουν ίδιες τιμές σε όλα τα γνωρίσματα. Δηλαδή πρέπει να υπάρχει τουλάχιστον ένα γνώρισμα, η τιμή του οποίου διαφοροποιεί τις δυο πλειάδες.

Γνωρίσματα - Κλειδιά: Κάθε πίνακας πρέπει να έχει ένα **πρωτεύον κλειδί** (primary key), δηλαδή ένα γνώρισμα που για κάθε πλειάδα έχει καταχωρημένη μια μοναδική τιμή. Με άλλα λόγια, το πρωτεύον κλειδί ενός πίνακα χρησιμοποιείται για να ξεχωρίζουν με μοναδικό τρόπο οι πλειάδες [4]. Για παράδειγμα, σε μια σχέση με όνομα 'Καταστήματα' που περιγράφει το σύνολο των καταστημάτων μιας αλυσίδας, πρωτεύον κλειδί μπορεί να είναι ένας αριθμητικός κωδικός, μοναδικός για κάθε κατάστημα. Αντιθέτως δεν προτείνεται να χρησιμοποιηθεί ως πρωτεύον κλειδί το γνώρισμα 'Πόλη' επειδή δυο καταστήματα μπορεί να βρίσκονται στην ίδια πόλη. Σε περίπτωση που ένας πίνακας έχει παραπάνω από ένα γνωρίσματα που μπορούν να είναι πρωτεύοντα, είναι αναγκαίο να επιλεχθεί μόνο ένα πρωτεύον κλειδί, και τα υπόλοιπα ονομάζονται **υποψήφια κλειδιά**. Όταν ένα πρωτεύον κλειδί αποτελείται από περισσότερα από ένα γνωρίσματα, τότε ονομάζεται **σύνθετο κλειδί** (composite). Τέλος, τα γνωρίσματα που δεν παίρνουν απαραίτητα μοναδικές τιμές μεταξύ των πλειάδων, ονομάζονται **δευτερεύοντα** (secondary) κλειδιά [4].

Είναι εύκολο να διαπιστώσει κανείς ότι το μοντέλο ER και το σχεσιακό μοντέλο έχουν κάποια κοινά χαρακτηριστικά [4]. Η βασική ομοιότητα είναι ότι ο τύπος οντοτήτων (στο μοντέλο ER) που αναλύσαμε σε προηγούμενη ενότητα, μπορεί να μετατραπεί στο σχεσιακό μοντέλο σε έναν πίνακα. Μια οντότητα που ανήκει σε έναν τύπο οντοτήτων στο μοντέλο ER, στο σχεσιακό μοντέλο αποτελεί μια εγγραφή (πλειάδα) του πίνακα. Τα γνωρίσματα ενός τύπου οντοτήτων στο μοντέλο ER αντιστοιχούν στις στήλες ενός πίνακα στο σχεσιακό μοντέλο.

2.5.2 Αντικειμενο-Σχεσιακά DBMS

Καθώς σημαντικό κομμάτι αυτής της εργασίας αποτελεί το σύστημα διαχείρισης βάσεων δεδομένων PostgreSQL, κρίνεται χρήσιμο να γίνει μια αναφορά στα αντικειμενο - σχεσιακά DBMS.

Καθώς τα σχεσιακά DBMS αποτέλεσαν χρήσιμα εργαλεία στην διαχείριση βάσεων δεδομένων και έλυσαν πολλά προβλήματα που είχαν προκύψει, έγιναν προσπάθειες για την επέκταση του σχεσιακού μοντέλου και την προσθήκη νέων δυνατοτήτων στα σχεσιακά DBMS. Αποτέλεσμα της επέκτασης του σχεσιακού μοντέλου ήταν το αντικειμενο - σχεσιακό μοντέλο, και τα αντίστοιχα DBMS ονομάστηκαν αντικειμενο - σχεσιακά DBMS. Τα νέα αυτά DBMS προέκυψαν από την ενίσχυση των σχεσιακών DBMS με αντικειμενοστρεφείς ιδιότητες [4].

Για να αποκτήσουν αντικειμενοστρεφείς ιδιότητες τα σχεσιακά DBMS, χρειάστηκαν αρκετές αλλαγές στον τρόπο που γινόταν αποθήκευση και διαχείριση δεδομένων [4]. Για να υποστηριχτούν οι νέες ιδιότητες, χρειάστηκαν επίσης διαφορετικοί τρόποι υπολογισμού απόδοσης του συστήματος, και συνεπώς νέοι μηχανισμοί βελτιστοποίησης. Φυσικά, για να μπορέσουν τα προγράμματα που χρησιμοποιούσαν αντικειμενο - σχεσιακά DBMS να επωφεληθούν από τις νέες ιδιότητες, χρειάστηκε επιπλέον ανάπτυξη της γλώσσας SQL [4].

2.5.3 PostgreSQL

Η **PostgreSQL** [5] είναι ένα open source αντικειμενο - σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων, που είναι γνωστό για την σωστή λειτουργία του, την αξιοπιστία του, και την ακεραιότητα δεδομένων που υπόσχεται. Είναι πλήρως συμβατή με το σύνολο ιδιοτήτων ACID (Atomicity, Consistency, Isolation, Durability), και εγγυάται αξιόπιστες συναλλαγές σε μια βάση δεδομένων ακόμη και σε περιπτώσεις λάθους. Υποστηρίζει επίσης αποθηκευμένες διαδικασίες, ξένα κλειδιά, όψεις, ενώσεις και triggers. Triggers ονομάζονται οι συναρτήσεις που καλούνται αυτόματα από το σύστημα όταν συμβαίνει κάποιο γεγονός στην βάση δεδομένων (π.χ. εισαγωγή νέων

δεδομένων σε πίνακα). Διαθέτει τους περισσότερους τύπους δεδομένων του standard SQL:2008, μερικοί από τους οποίους είναι οι εξής: INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, και TIMESTAMP. Επιπλέον, υποστηρίζει την αποθήκευση και διαχείριση εικόνων, βίντεο και ήχων. Η PostgreSQL διαθέτει native προγραμματιστικά περιβάλλοντα για γλώσσες υψηλού επιπέδου όπως C/C++, Java, .Net, Perl, Python και άλλες, καθώς και πολύ καλά εγχειρίδια χρήσης. Τέλος, βρίσκεται σε ανάπτυξη για τουλάχιστον 15 χρόνια και μπορεί να εγκατασταθεί στα περισσότερα σύγχρονα λειτουργικά συστήματα (Unix-based, Linux, Windows).

Ένα από τα πολλά πλεονεκτήματα της PostgreSQL [5] είναι η δυνατότητα εξατομίκευσης των λειτουργιών της. Εκτός από διάφορες διαδικασίες (μαθηματικές συναρτήσεις, επεξεργασία συμβολοσειρών, κρυπτογραφία) που υπάρχουν ήδη στην βιβλιοθήκη της, triggers και νέες διαδικασίες μπορούν να γραφτούν σε C και να φορτωθούν ως βιβλιοθήκες, προσφέροντας έτσι μεγάλη ευελιξία στην προσθήκη νέων δυνατοτήτων. Επίσης, η PostgreSQL διαθέτει framework που επιτρέπει στους προγραμματιστές να δημιουργήσουν δικούς τους τύπους δεδομένων, όπως και την υποστήριξη συναρτήσεων και τελεστών που ορίζουν την συμπεριφορά τους.

Σημαντικό πλεονέκτημα της PostgreSQL αποτελεί το γεγονός ότι είναι open source λογισμικό (ανοιχτού κώδικα). Σύμφωνα με το PostgreSQL License, ο χρήστης μπορεί να χρησιμοποιήσει, να τροποποιήσει, ή ακόμη και να διαμοιράσει το σύστημα με όποιον τρόπο επιθυμεί.

Εγκατάσταση: Μπορεί να γίνει εγκατάσταση του συστήματος της PostgreSQL ακολουθώντας τις οδηγίες της επίσημης ιστοσελίδας, με επίσκεψη στον σύνδεσμο https://www.postgresql.org/download/. Για λόγους απλότητας, σε παραδείγματα που θα ακολουθήσουν θα χρησιμοποιηθεί η έκδοση PostgreSQL 9.5.9 για Windows.

3.Η Εφαρμογή SQL Draw

3.1 Απαιτήσεις Εφαρμογής

Η εφαρμογή SQL Draw σχεδιάστηκε έχοντας ως βάση τις παρακάτω απαιτήσεις:

- Συμβατότητα με σύγχρονους περιηγητές διαδικτύου ώστε να μπορεί να χρησιμοποιηθεί online χωρίς καμία επιπλέον διαδικασία εγκατάστασης, και ως επακόλουθο, χωρίς περιορισμό στο λειτουργικό σύστημα του χρήστη.
- Παροχή κατάλληλου μενού για την σχεδίαση διαγραμμάτων οντοτήτων - συσχετίσεων καθώς και δυνατότητα μετακίνησης των γραφικών σχημάτων στον χώρο.
- Δυνατότητα εξαγωγής εικόνας .PNG που απεικονίζει το διάγραμμα που σχεδιάστηκε. Επίσης, ο χρήστης πρέπει να είναι σε θέση να μπορεί να αποθηκεύσει το διάγραμμα που δημιούργησε σε μορφή αρχείου .txt, το οποίο χρησιμοποιείται για την επεξεργασία του διαγράμματος σε περίπτωση που ανανεωθεί ή κλείσει η σελίδα της εφαρμογής.
- Έλεγχος για τυχόν λάθη που υπάρχουν στο διάγραμμα και εμφάνιση κατάλληλου μηνύματος λάθους για κάθε ένα από αυτά.
- Τέλος, η εφαρμογή SQL Draw πρέπει να έχει την δυνατότητα μετατροπής του διαγράμματος σε εντολές PostgreSQL, οι οποίες παράγουν το αντικειμενο-σχεσιακό σχήμα που αντιστοιχεί στο διάγραμμα.

3.2 Περιγραφή της Εφαρμογής SQL Draw

3.2.1 Γενικά

Για την χρήση της εφαρμογής δεν χρειάζεται καμία εγκατάσταση εφόσον υπάρχει πρόσβαση στο internet. Ο χρήστης μπορεί να δει την εφαρμογή με μια επίσκεψη στον σύνδεσμο: https://vg-draw-app.herokuapp.com/. Ενδέχεται να υπάρχει καθυστέρηση φόρτωσης της σελίδας την πρώτη φορά. Σε περίπτωση που ο σύνδεσμος δεν δουλεύει ή δεν έχετε τον πηγαίο κώδικα, μπορείτε να επικοινωνήσετε με e-mail στην διεύθυνση eirini.p09@gmail.com.

Για την σωστή χρήση της εφαρμογής, καλό είναι ο περιηγητής να έχει αναβαθμιστεί στην πιο πρόσφατη έκδοση, και να υπάρχει ποντίκι συνδεδεμένο στην συσκευή (η χρήση μέσω συσκευών αφής ίσως δεν είναι εύκολη). Επίσης δεν προτείνεται η χρήση των περιηγητών Internet Explorer και Safari καθώς δεν είναι συμβατοί με τις βιβλιοθήκες που χρησιμοποιούνται.

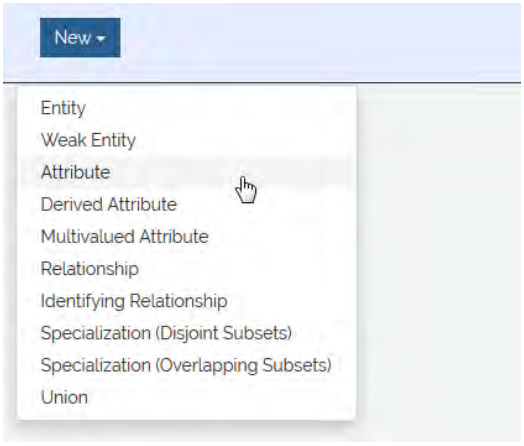
Σε περίπτωση που δεν υπάρχει σύνδεση στο internet ή για οποιοδήποτε λόγο πρέπει να τρέξει η εφαρμογή offline, αυτό γίνεται εύκολα με την χρήση του πηγαίου κώδικα που συνοδεύει την παρούσα εργασία. Μέσα στον φάκελο με όνομα web υπάρχει το αρχείο index.html. Για να τρέξει η εφαρμογή αρκεί να γίνει δεξί κλικ στο αρχείο αυτό, και στη συνέχεια από το μενού 'Open with' να επιλεχτεί κάποιος περιηγητής διαδικτύου (για παράδειγμα Firefox ή Google Chrome).

3.2.2 Κεντρική Οθόνη

Σχήμα 3.1: Κεντρική Οθόνη

Η κεντρική οθόνη διαθέτει απλή μορφή με σκοπό την βελτίωση της λειτουργικότητας. Το μεγαλύτερο μέρος της οθόνης καταλαμβάνει η περιοχή σχεδίασης, η οποία έχει παραλληλόγραμμο σχήμα και προσαρμόζεται αυτόματα στο μέγεθος οθόνης της συσκευής. Επάνω αριστερά υπάρχει το μενού 'New' το οποίο χρησιμοποιείται για την δημιουργία νέων αντικειμένων. Στο κάτω μέρος υπάρχουν οι επιλογές αποθήκευσης / φόρτωσης, ελέγχου λαθών και μετατροπής διαγράμματος. Οι λειτουργίες αυτές θα εξηγηθούν σε επόμενη ενότητα. Ακριβώς από κάτω ο σύνδεσμος 'HOME' ανακατευθύνει προς την αρχική σελίδα, ο σύνδεσμος 'HOW TO USE' περιέχει πληροφορίες και οδηγίες για την δημιουργία διαγραμμάτων, και ο σύνδεσμος 'SOURCE CODE' κατευθύνει στο GitHub repository της εφαρμογής.

3.2.3 Δημιουργία Διαγράμματος ER



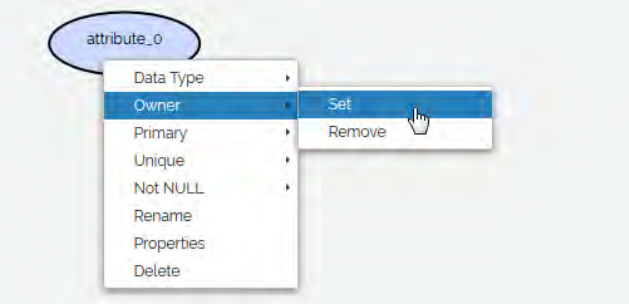
Σχήμα 3.2: Μενού αρχικοποίησης

Με απλό κλικ στο κουμπί 'New' εμφανίζεται το drop-down μενού που επιτρέπει στον χρήστη να επιλέξει κάποιο αντικείμενο. Μετά την επιλογή του, το αντίστοιχο σχήμα εμφανίζεται στην περιοχή σχεδίασης με συμβολισμό όπως ακριβώς περιγράφεται από το μοντέλο ER. Το drop-down μενού χρησιμεύει στην εξοικονόμηση χώρου, και παραμένει κρυμμένο όσο δεν χρειάζεται.

Αφού έχει δημιουργηθεί ένα σχήμα, μπορεί να μετακινηθεί οπουδήποτε μέσα στο πλαίσιο μέσω drag & drop. Δηλαδή ο χρήστης έχοντας τον δείκτη του ποντικιού επάνω στο σχήμα, κρατώντας πατημένο το αριστερό κλικ μετακινεί το σχήμα στον χώρο.

Επίσης, μπορεί να γίνει προσαρμογή του μεγέθους ενός σχήματος. Αφού γίνει διπλό αριστερό κλικ πάνω σε ένα σχήμα, πάλι έχοντας πατημένο το αριστερό κλικ και μετακινώντας τον δείκτη προς κάποια κατεύθυνση, γίνεται αύξηση ή μείωση του μεγέθους σχήματος. Με ακόμη ένα διπλό αριστερό κλικ, τερματίζεται η συγκεκριμένη λειτουργία όταν το σχήμα έχει το επιθυμητό μέγεθος.

Εφόσον κάποια σχήματα έχουν δημιουργηθεί, οι ιδιότητες και οι συνδέσεις μεταξύ τους γίνονται χρησιμοποιώντας μενού που αντιστοιχεί στο κάθε αντικείμενο ξεχωριστά, και εμφανίζεται με δεξί κλικ επάνω στο αντικείμενο.



Σχήμα 3.3: Μενού γνωρίσματος

Στο σχήμα 3.3 παρουσιάζεται το μενού ενός απλού γνωρίσματος, όμως στην συνέχεια θα γίνει λεπτομερής επεξήγηση του μενού όλων των αντικειμένων.

Οντότητα (Entity)

Το μενού των αντικειμένων τύπου οντότητα αποτελείται από τις επιλογές:

- Superclass: Αυτή η επιλογή χωρίζεται σε δυο ακόμη επιλογές, Set και Remove. Η πρώτη επιλογή ενώνει το αντικείμενο οντότητα με κάποια άλλη ισχυρή οντότητα που γίνεται υπερκλάση της. Όταν γίνει κλικ στην επιλογή πρέπει στη συνέχεια να γίνει κλικ στην οντότητα-υπερκλάση και τα δυο σχήματα θα ενωθούν. Ένας τύπος οντοτήτων μπορεί να έχει μόνο μια υπερκλάση. Η επιλογή Remove διαγράφει αυτήν την σύνδεση, εφόσον υπάρχει.

- Specialization: Αυτή η επιλογή χωρίζεται σε δυο ακόμη επιλογές, Set και Remove. Η επιλογή Set δουλεύει ακριβώς όπως στην επιλογή Superclass. Ένα αντικείμενο οντότητα μπορεί να είναι συνδεδεμένο με πολλές εξειδικεύσεις. Οπότε, στην επιλογή Remove πρέπει να προσδιοριστεί ποια σύνδεση θα διαγραφεί με ένα ακόμη κλικ στο αντίστοιχο αντικείμενο της εξειδίκευσης.

- Defining Criteria: Η επιλογή Defining Criteria εισάγει τιμή για ένα ορίζον γνώρισμα σε μια σύνδεση τύπου οντοτήτων - εξειδίκευση. Χωρίζεται σε δυο ακόμη επιλογές, Set και Remove. Εφόσον επιλεχτεί η επιλογή Set πρέπει να γίνει κλικ σε μια εξειδίκευση με την οποία η οντότητα έχει ήδη συνδεθεί. Θα εμφανιστεί παράθυρο που ζητάει την εισαγωγή μιας τιμής. Η επιλογή Remove λειτουργεί με ίδιο τρόπο όπως στην επιλογή Specialization.

- Union: Αυτή η επιλογή χωρίζεται σε τρεις ακόμη επιλογές, Set (Partial), Set (Total) και Remove. Οι δύο πρώτες λειτουργούν ακριβώς όπως η επιλογή Set της επιλογής Specialization, απλά εδώ έχουμε μερική ή ολική συμμετοχή (partial / total). Αφού γίνει η σύνδεση, το αντικείμενο οντότητα θεωρείται μια από τις υπερκλάσεις της εξειδίκευσης. Η επιλογή Remove λειτουργεί όπως στο υπο-μενού Specialization.

- Rename: Η επιλογή Rename εμφανίζει ένα παράθυρο που προτρέπει τον χρήστη να πληκτρολογήσει ένα νέο όνομα για το αντικείμενο. Με κλικ στο OK το νέο όνομα παίρνει την θέση του στο σχήμα.

- Delete: Αυτή η επιλογή διαγράφει το αντικείμενο και όλες τις συνδέσεις στις οποίες συμμετέχει.

Οι επιλογές Rename και Delete εκτελούν την ίδια ακριβώς λειτουργία σε όλα τα αντικείμενα οπότε από εδώ και πέρα απλώς θα αναφέρονται επιγραμματικά.

Ασθενής Οντότητα (Weak Entity)

Το μενού των αντικειμένων τύπου ασθενής οντότητα αποτελείται από τις επιλογές, καθώς μια ασθενής οντότητα δεν μπορεί να είναι υπερκλάση ή υποκλάση:

- Rename
- Delete

Γνώρισμα (Attribute)

Το μενού των αντικειμένων τύπου γνώρισμα αποτελείται από τις επιλογές:

- Data Type: Ορισμός του τύπου δεδομένων ενός γνωρίσματος. Ο χρήστης μπορεί να επιλέξει ανάμεσα σε INTEGER, BIGINT, SMALLINT, REAL, DOUBLE PRECISION, DECIMAL (i, j), CHARACTER (n), CHARACTER VARYING (n), BIT (n), BIT VARYING (n), BOOLEAN, XML, TIME, DATE, TIMESTAMP, INTERVAL, Other (Add Type). Ο αριθμός n δηλώνει το μήκος που πρέπει να έχει μια τιμή CHARACTER ή BIT, και το μέγιστο μήκος μιας τιμής BIT VARYING ή CHARACTER VARYING. Στον τύπο δεδομένων DECIMAL (i, j) ο αριθμός i [1] (precision), δηλώνει το πλήθος ψηφίων του δεκαδικού αριθμού ενώ ο αριθμός j (scale) δηλώνει τον αριθμό ψηφίων που ακολουθούν την υποδιαστολή. Επομένως, εάν ο χρήστης επιλέξει κάποιο τύπο που περιγράφεται από n, i ή j τότε η συνάρτηση ζητάει να προσδιοριστούν οι τιμές τους. Η τελευταία επιλογή Other επιτρέπει στον χρήστη να εισάγει κάποιον τύπο δεδομένων που δεν περιλαμβάνεται στην λίστα. Όλα τα γνωρίσματα δημιουργούνται έχοντας ως τύπο δεδομένων INTEGER.

- Owner: Αυτή η επιλογή χωρίζεται σε δυο ακόμη επιλογές, Set και Remove. Η επιλογή Set ορίζει σε ποιο τύπο οντοτήτων / συσχετίσεων ανήκει το γνώρισμα. Αφού επιλεγεί, πρέπει να γίνει κλικ σε ένα σχήμα οντότητας / συσχέτισης για να ολοκληρωθεί η σύνδεση. Επίσης, ένα γνώρισμα μπορεί να έχει ως ιδιοκτήτη ένα άλλο γνώρισμα, όπως είδαμε στα σύνθετα γνωρίσματα. Η επιλογή Remove αφαιρεί την σύνδεση. Ένα γνώρισμα επιτρέπεται να ανήκει σε μοναδικό αντικείμενο.

- Primary: Αυτή η επιλογή χωρίζεται σε δυο ακόμη επιλογές, Yes και No. Χαρακτηρίζει ένα γνώρισμα ως πρωτεύον κλειδί. Πρέπει να χρησιμοποιηθεί σε περίπτωση που ένας τύπος οντότητας έχει 2 ή παραπάνω κλειδιά (unique attributes).

- Unique: Αυτή η επιλογή χωρίζεται σε δυο ακόμη επιλογές, Yes και No. Χαρακτηρίζει ένα γνώρισμα ως κλειδί.

- Not NULL: Αυτή η επιλογή χωρίζεται σε δυο ακόμη επιλογές, Yes και No. Καθορίζει το εάν επιτρέπεται το γνώρισμα να πάρει τιμές NULL.

- Properties: Εμφανίζει παράθυρο στο οποίο καταγράφονται οι ιδιότητες του γνωρίσματος για χαρακτηριστικά που δεν είναι απαραίτητα ορατά στο διάγραμμα, όπως το εάν χαρακτηρίζεται ως primary, unique και not NULL.

- Rename

- Delete

Παραγόμενο Γνώρισμα (Derived Attribute)

Το μενού των αντικειμένων τύπου γνώρισμα αποτελείται από τις επιλογές (όπως εξηγήθηκαν παραπάνω):

- Owner
- Rename
- Delete

Πλειότιμο Γνώρισμα (Multivalued Attribute)

Το μενού των αντικειμένων τύπου γνώρισμα αποτελείται από τις επιλογές (όπως εξηγήθηκαν παραπάνω):

- Data Type
- Owner
- Unique
- Rename

- Properties
- Delete

Συσχέτιση (Relationship) και Προσδιορίζουσα Συσχέτιση (Identifying Relationship)

Ο κάθε τύπος συσχέτισης μπορεί να συνδέεται με τέσσερις οντότητες, το πολύ (βαθμού 4). Ο περιορισμός αυτός υπάρχει επειδή οι συσχετίσεις βαθμού μεγαλύτερου του 2 είναι σπάνιες, και το να υπάρχει περιορισμένος αριθμός συνδέσεων διευκολύνει κατά πολύ την διαχείρισή τους. Η κάθε σύνδεση λοιπόν αντιστοιχεί σε μια γωνία του ρόμβου, και επομένως έχουμε 4 συνδέσεις: Top, Bottom, Left, Right. Το μενού για τα αντικείμενα τύπου συσχέτιση αποτελείται από τις επιλογές:

- Top: Η επιλογή αυτή χωρίζεται στις εξής υπο-επιλογές: Set (Partial), Set (Total), Remove, Ratio: 1//N/M/L/K. Οι πρώτες δυο επιλογές συνδέουν την συσχέτιση με έναν τύπο οντοτήτων, χρησιμοποιώντας μερική ή ολική συμμετοχή αντίστοιχα. Αφού γίνει κλικ σε μια από τις δυο επιλογές, πρέπει να γίνει κλικ στο σχήμα της οντότητας. Γίνεται να αλλάξει το είδος συμμετοχής χωρίς να διαγραφεί η σύνδεση, αρκεί να γίνει η ίδια διαδικασία επιλέγοντας σωστή συμμετοχή, και η σύνδεση θα αντικατασταθεί. Η επιλογή Remove αφαιρεί την σύνδεση. Οι επιλογές Ratio καθορίζουν τον λόγο πληθικότητας. Σε μια δυαδική συσχέτιση χρειαζόμαστε μόνο τα γράμματα M και N για να δείξουμε συσχέτιση πολλά προς πολλά, όμως όταν έχουμε τέσσερις συνδέσεις χρειαζόμαστε τέσσερα γράμματα (N,M,K,L).

- Bottom: Όμοια με παραπάνω.

- Left: Όμοια με παραπάνω.

- Right: Όμοια με παραπάνω.

- Rename

- Delete

Εξειδικεύσεις (Specialization), Disjoint & Overlapping

Το μενού για τις εξειδικεύσεις αποτελείται από τις επιλογές:

- Superclass: Η επιλογή αυτή χωρίζεται σε Set (Partial), Set (Total) και Remove, οι οποίες λειτουργούν με παρόμοιο τρόπο με όσες εξηγήθηκαν παραπάνω. Μια εξειδίκευση μπορεί να συνδεθεί με μόνο έναν ισχυρό τύπο οντοτήτων, που θα μετατραπεί σε υπερκλάση της εξειδίκευσης. Όμως μπορεί να έχει απεριόριστο αριθμό υποκλάσεων.

- Defining Attribute: Η επιλογή αυτή χωρίζεται σε Set και Remove. Η επιλογή Set απαριθμεί τα γνωρίσματα της υπερκλάσης (εφόσον υπάρχει) και προτρέπει τον χρήστη να επιλέξει ένα ως ορίζον γνώρισμα της εξειδίκευσης.

- Delete

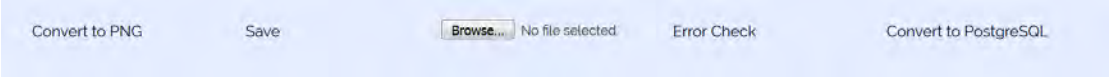
Ένωση (Union)

Τέλος, η ένωση έχει το παρακάτω μενού:

- Superclass: Η επιλογή αυτή χωρίζεται σε Add και Remove. Μια ένωση μπορεί να έχει πολλές υπερκλάσεις, όμως μόνο μια υποκλάση. Η επιλογή Add συνδέει την ένωση με έναν ισχυρό τύπο οντότητας, ενώ η επιλογή Remove αφαιρεί την σύνδεση. Για να διευκρινιστεί ποια σύνδεση από όλες πρέπει να αφαιρεθεί, ο χρήστης πρέπει να κάνει κλικ στο αντίστοιχο σχήμα.

- Delete

3.2.4 Αποθήκευση και άνοιγμα αρχείου



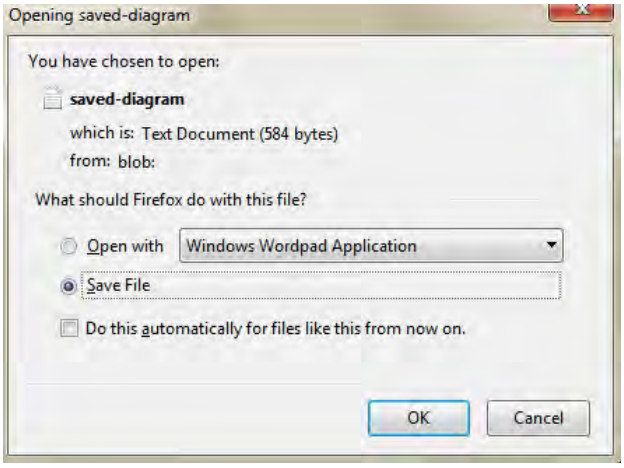
Σχήμα 3.4: Περισσότερες λειτουργίες

Ακριβώς κάτω από το πλαίσιο σχεδίασης υπάρχουν οι επιλογές αποθήκευσης και φόρτωσης αρχείου διαγράμματος, όπως και η μετατροπή σε εικόνα PNG.

Εικόνα PNG: Το κουμπί με όνομα 'Convert to PNG' χρησιμεύει στην μετατροπή του πλαισίου σχεδίασης σε εικόνα. Η εφαρμογή δημιουργεί μια νέα καρτέλα στον περιηγητή η οποία περιέχει την εικόνα. Από εκεί ο χρήστης μπορεί εάν θέλει να την αποθηκεύσει κάνοντας δεξί κλικ επάνω της και επιλέγοντας 'Save Image As...'. Η λεπτομερής λειτουργία του θα εξηγηθεί στην ενότητα ανάλυσης του κώδικα της εφαρμογής.

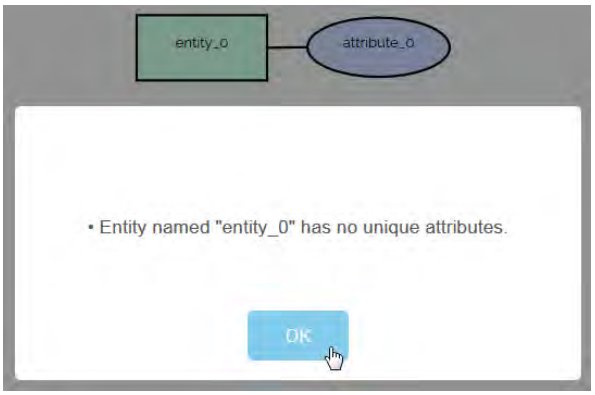
Αποθήκευση διαγράμματος: Μια ακόμη χρήσιμη λειτουργία είναι αυτή της αποθήκευσης. Σε οποιοδήποτε στάδιο δημιουργίας του διαγράμματος, ο χρήστης μπορεί να αποθηκεύσει το διάγραμμα σε μορφή αρχείου .txt το οποίο βασίζεται στην γλώσσα σήμανσης XML για να περιγράψει τα αντικείμενα που δημιουργήθηκαν και όλες τις συσχετίσεις μεταξύ τους. Με αυτόν τον τρόπο, ακόμη και αν γίνει ανανέωση ή κλείσιμο της σελίδας της εφαρμογής και χαθούν τα προσωρινά δεδομένα, το διάγραμμα θα βρίσκεται αποθηκευμένο. Η εφαρμογή εμφανίζει ένα παράθυρο που επιτρέπει στον χρήστη να επιλέξει όνομα για το αρχείο και τοποθεσία αποθήκευσης.

Άνοιγμα αρχείου: Εφόσον υπάρχει αποθηκευμένο αρχείο διαγράμματος SQL Draw, μπορεί να γίνει άνοιγμα του αρχείου από την εφαρμογή, χρησιμοποιώντας το κουμπί 'Browse'. Το SQL Draw θα διαβάσει το αρχείο .txt και θα αρχικοποιήσει ξανά το διάγραμμα. Περισσότερες λεπτομέρειες για τον αλγόριθμο αποθήκευσης - φόρτωσης και τα αρχεία αυτά θα δοθούν σε επόμενο κεφάλαιο.



Σχήμα 3.5: Διαδικασία αποθήκευσης διαγράμματος

3.2.5 Έλεγχος σφαλμάτων



Σχήμα 3.6: Ειδοποίηση σφάλματος απουσίας κλειδιού

Το κουμπί με όνομα 'Error Check' όπως φαίνεται στο σχήμα 3.4 πραγματοποιεί τον έλεγχο για σφάλματα σε ένα διάγραμμα. Η διαδικασία αυτή έχει ως σκοπό την προετοιμασία για μετατροπή σε αντικειμενο-σχεσιακό σχήμα PostgreSQL, οπότε προτρέπει τον χρήστη να διορθώσει το διάγραμμα σε περίπτωση που υπάρχουν λάθη. Με το πάτημα του κουμπιού 'Error Check' εμφανίζεται μήνυμα που ειδοποιεί σχετικά με την ύπαρξη λαθών τα οποία απαριθμεί. Σε περίπτωση που δεν εντοπιστούν σφάλματα, εμφανίζεται το μήνυμα 'No errors found'. Στην συνέχεια, θα γίνει αναφορά σε όλα τα μηνύματα λάθους που είναι δυνατό να εμφανιστούν, και στις περιπτώσεις κατά τις οποίες εμφανίζονται.

"No objects found."
Το συγκεκριμένο μήνυμα εμφανίζεται όταν δεν υπάρχει κανένα αντικείμενο στο πλαίσιο.

"Entity named (name) has no attributes."
Εμφανίζεται όταν ένας τύπος οντοτήτων δεν είναι συνδεδεμένο με γνωρίσματα.

"Entity named (name) has no unique attributes."
Εμφανίζεται όταν δεν υπάρχει κανένα γνώρισμα ενός τύπου οντοτήτων που να χαρακτηρίζεται ως κλειδί (unique ή primary).

"Entity named (name) has multiple unique attributes, please specify the primary key."
Εμφανίζεται όταν ένας τύπος οντοτήτων διαθέτει 2 ή παραπάνω υποψήφια κλειδιά. Επειδή μόνο ένα μπορεί να γίνει πρωτεύον κλειδί, ο χρήστης καλείται να το επιλέξει.

"Entity named (name) has multiple primary keys, please specify only one key."
Όμοια με παραπάνω.

"Composite primary keys are not supported yet. "
Εμφανίζεται για λόγους δυσκολίας στην υλοποίηση, όταν ένα σύνθετο κλειδί που διαιρείται σε 2 ή παραπάνω γνωρίσματα αποτελεί πρωτεύον κλειδί.

"Weak entity named (name) must be part of an identifying relationship."
Όταν υπάρχει ασθενής τύπος οντοτήτων στο διάγραμμα, πρέπει υποχρεωτικά να συμμετέχει σε προσδιορίζουσα συσχέτιση.

"Weak entities cannot be superclasses."

Όταν μια ασθενής οντότητα χρησιμοποιείται ως υπερκλάση.

"Weak entities cannot be subclasses."

Όταν μια ασθενής οντότητα χρησιμοποιείται ως υποκλάση.

" Multiple entities cannot have the same name."

Οι τύποι οντοτήτων πρέπει να έχουν μοναδικά ονόματα, σε περίπτωση που υπάρχουν 2 ή περισσότεροι τύποι με το ίδιο όνομα, εμφανίζεται αυτό το μήνυμα.

"Attribute named (name) does not belong to an element."

Το συγκεκριμένο μήνυμα εμφανίζεται όταν κάποιο γνώρισμα δεν έχει ιδιοκτήτη.

"Attribute named (name) needs to be NOT NULL."

Όταν ένα γνώρισμα ανήκει σε σύνθετο γνώρισμα το οποίο χαρακτηρίζεται ως Not NULL. Σε αυτήν την περίπτωση πρέπει και το πρώτο να χαρακτηριστεί ως Not NULL.

"Multiple attributes belonging to the same element cannot have the same name."

Όταν ένας τύπος οντοτήτων έχει 2 ή περισσότερα γνωρίσματα με το ίδιο όνομα.

"Relationship named (name) must be connected to at least 2 entities."

Όταν μια συσχέτιση έχει λιγότερες από 2 συνδέσεις.

" Multiple relationships cannot have the same name."

Όταν 2 ή παραπάνω συσχετίσεις έχουν το ίδιο όνομα.

"Identifying relationships cannot have attributes."

Όταν μια προσδιορίζουσα συσχέτιση έχει γνωρίσματα.

"Identifying relationship named (name) must be connected to exactly one weak entity."
Όταν μια προσδιορίζουσα συσχέτιση δεν είναι συνδεδεμένη με καμία ασθενή οντότητα, ή είναι συνδεδεμένη με περισσότερες από μια.

"Identifying relationship named (name) must be connected to at least one strong entity."
Όταν μια προσδιορίζουσα συσχέτιση δεν είναι συνδεδεμένη με καμία ισχυρή οντότητα.

"Identifying relationship named (name) must be connected to a weak entity through total participation."
Όταν μια ασθενής οντότητα συμμετέχει σε προσδιορίζουσα συσχέτιση, όμως με μερική συμμετοχή. Όπως εξηγήθηκε σε προηγούμενο κεφάλαιο, η ολική συμμετοχή είναι υποχρεωτική.

"Identifying relationship named (name) must be connected to all owner entities using '1' cardinality ratio."
Όταν η προσδιορίζουσα συσχέτιση χρησιμοποιεί λόγο πληθικότητας διαφορετικό του '1' για σύνδεση με τις ισχυρές οντότητες.

"Specializations must have a superclass."
Όταν το σχήμα μιας εξειδίκευσης δεν είναι συνδεδεμένο με οντότητα-υπερκλάση.
"Specializations must have at least 2 subclasses."
Όταν η εξειδίκευση έχει λιγότερες από 2 υποκλάσεις, καθώς δεν υπάρχει λόγος για την ύπαρξή της στην περίπτωση μόνο μιας υποκλάσης.

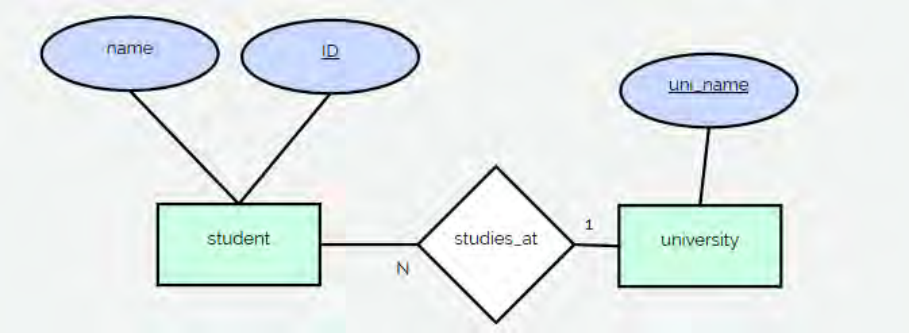
"Unions must have a subclass."

Όταν μια ένωση δεν διαθέτει υποκλάση.

" Unions must have at least 2 superclasses."

Όταν μια ένωση έχει λιγότερες από 2 υπερκλάσεις.

3.2.6 Μετατροπή σε κώδικα PostgreSQL




```
CREATE TABLE university (
      uni_name   CHARACTER(50) NOT NULL,
PRIMARY KEY (uni_name) );

CREATE TABLE student (
      FK_2_uni_name    CHARACTER(50),
      name  CHARACTER(60) NOT NULL,
      ID    INTEGER NOT NULL,
PRIMARY KEY (ID) );


ALTER TABLE student ADD FOREIGN KEY (FK_2_uni_name) REFERENCES
university (uni_name) ON DELETE SET NULL ON UPDATE CASCADE;
```

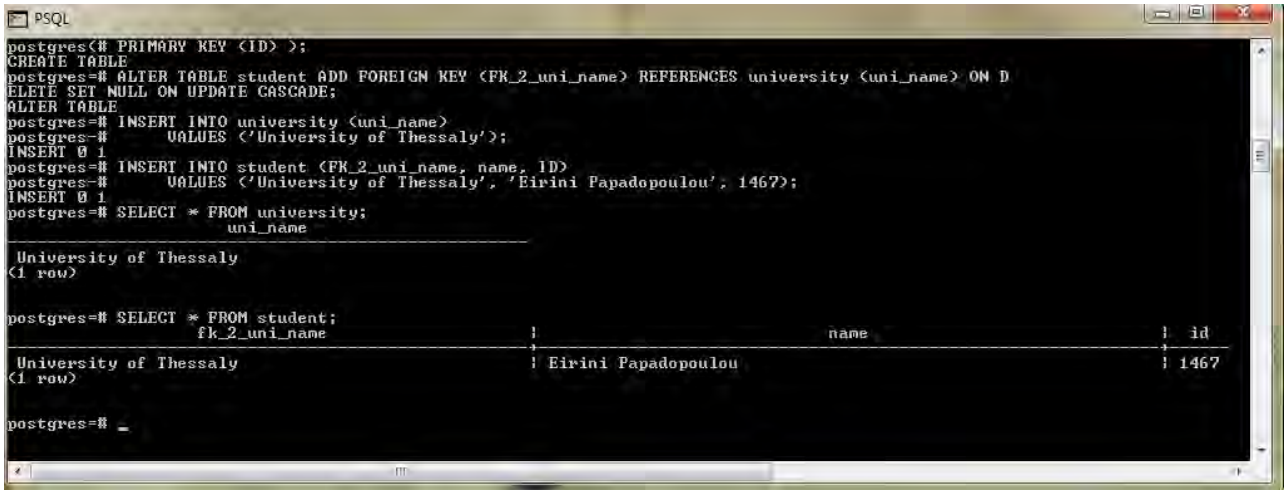
Σχήμα 3.7: Μετατροπή διαγράμματος ER σε PostgreSQL

Το κουμπί 'Convert to PostgreSQL' μετατρέπει το τρέχον διάγραμμα σε κώδικα PostgreSQL, ο οποίος γράφεται σε αρχείο κειμένου .txt και ο χρήστης στη συνέχεια μπορεί να το αποθηκεύσει. Η διαδικασία μετατροπής αυτόματα πραγματοποιεί τον έλεγχο για σφάλματα, και η μετατροπή ξεκινάει μόνο εφόσον δεν υπάρχουν σφάλματα. Παράδειγμα της μετατροπής παρουσιάζεται στο σχήμα 3.7, ενώ επεξήγηση του αλγορίθμου θα γίνει σε διαφορετικό κεφάλαιο.

Στην συνέχεια, δοκιμάζουμε τον κώδικα στο σύστημα PostgreSQL 9.5.9 για Windows.

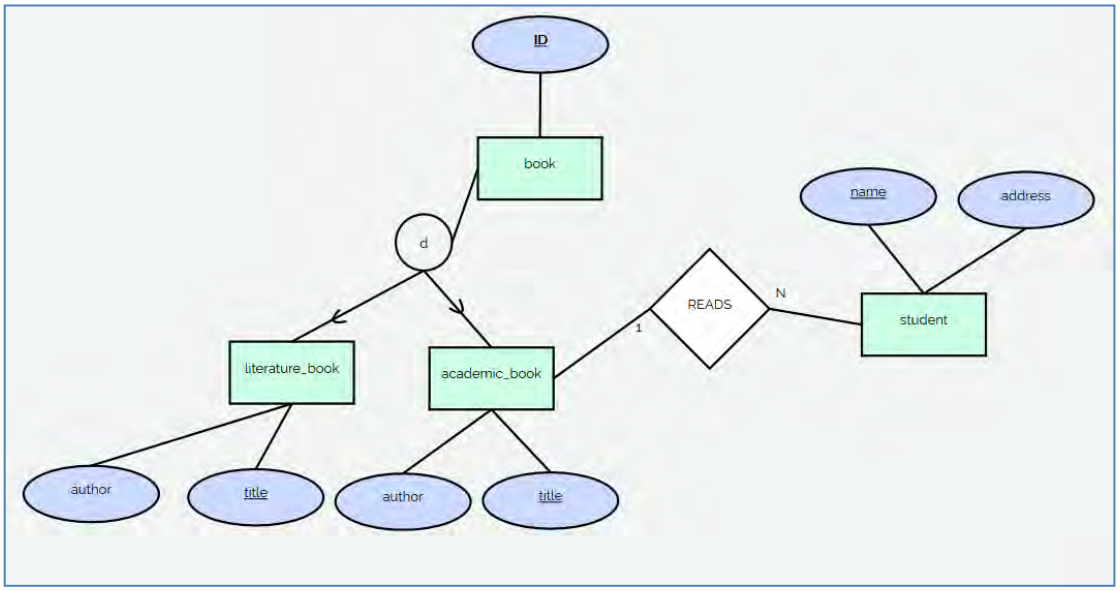
Σχήμα 3.8: Χρήση του παραγόμενου κώδικα

Όπως φαίνεται στο σχήμα 3.8, όταν έχει ολοκληρωθεί η εγκατάσταση του συστήματος, μπορεί να γίνει χρήση του console για την εισαγωγή εντολών. Στο σχήμα 3.8 έχει γίνει αντιγραφή και επικόλληση κάθε εντολής ξεχωριστά. Στη συνέχεια θα εισάγουμε κάποια στοιχεία στους πίνακες, χρησιμοποιώντας την εντολή INSERT και θα γίνει προβολή των πινάκων χρησιμοποιώντας την εντολή SELECT.



Σχήμα 3.9: Τελικό αποτέλεσμα

3.2.7 Παράδειγμα μετατροπής



Σχήμα 3.9.1: Διάγραμμα προς μετατροπή

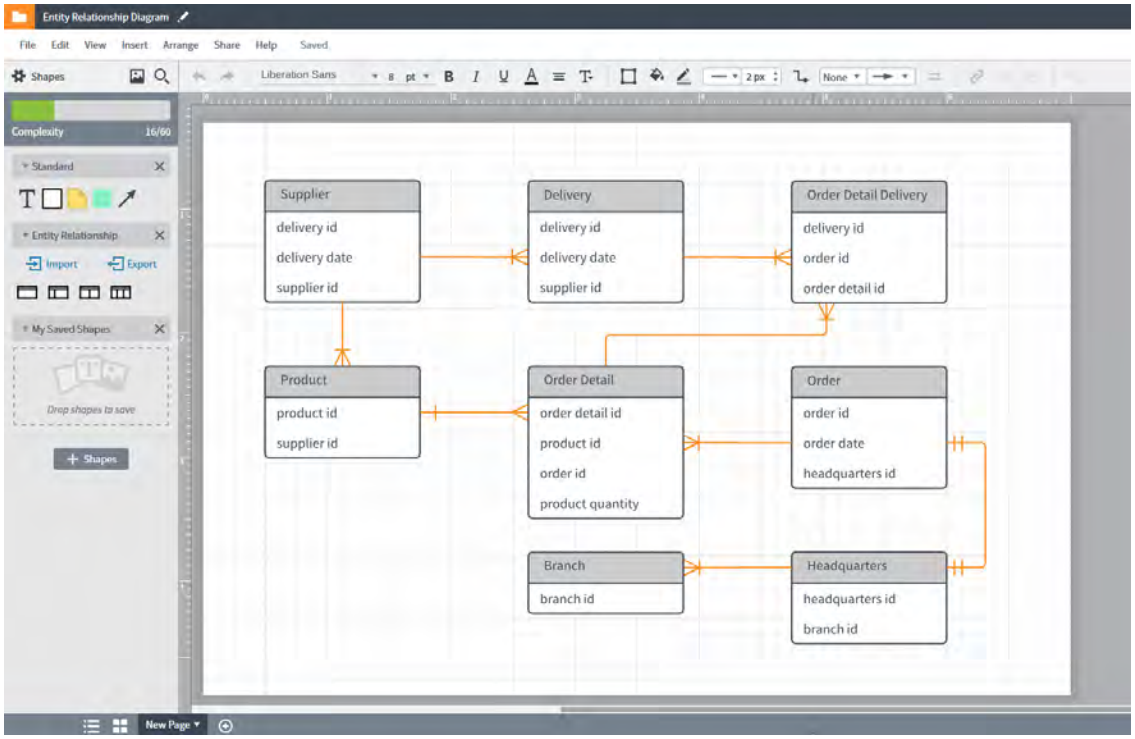
Ας δούμε πως μετατρέπεται ένα μεγαλύτερο διάγραμμα που περιέχει μια εξειδίκευση. Το διάγραμμα του σχήματος 3.9.1 μας δίνει τον παρακάτω κώδικα PostgreSQL.

```
CREATE TABLE academic_book (
        author   CHARACTER(20),
        title      CHARACTER(20) NOT NULL,
UNIQUE (title),
UNIQUE (ID)
 )INHERITS (book);

CREATE TABLE student (
        FK_2_ID         INTEGER,
        address CHARACTER VARYING(25),
        name     CHARACTER(20) NOT NULL,
PRIMARY KEY (name) );

CREATE TABLE book (
        ID       INTEGER NOT NULL,
PRIMARY KEY (ID) );

CREATE TABLE literature_book (
        title      CHARACTER(20) NOT NULL,
        author   CHARACTER(20),
UNIQUE (title),
UNIQUE (ID)
 )INHERITS (book);
```

ALTER TABLE student ADD FOREIGN KEY (FK_2_ID) REFERENCES academic_book (ID) ON DELETE SET NULL ON UPDATE CASCADE;

3.3 Υπάρχουσες Εφαρμογές

3.3.1 Lucidchart



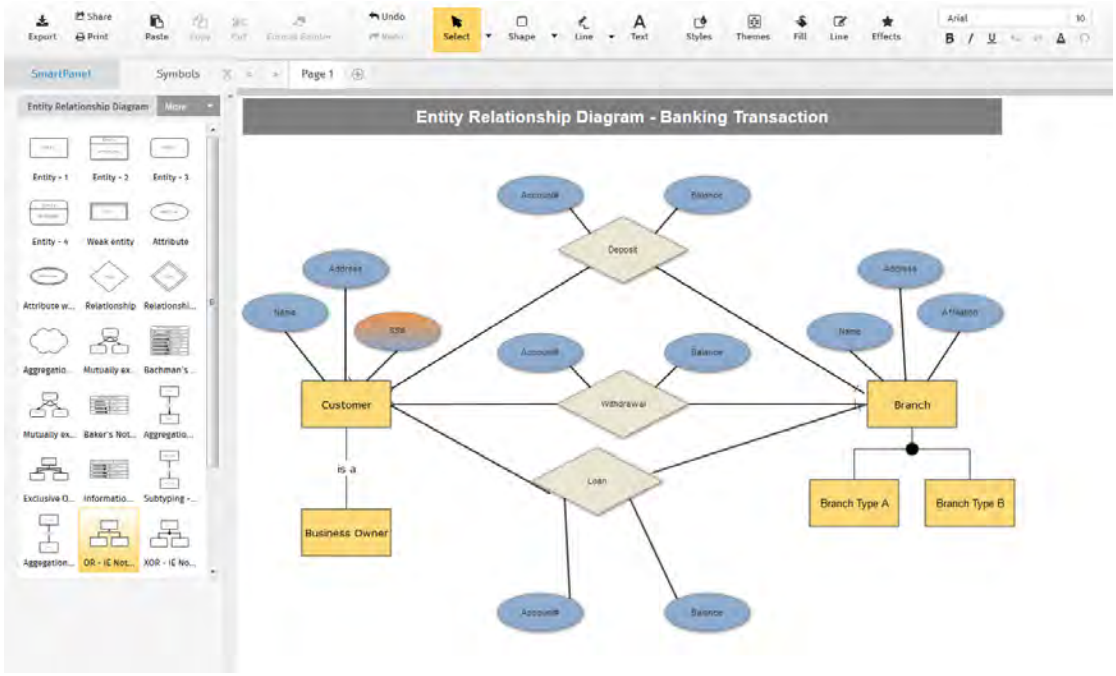
Σχήμα 3.10: Lucidchart

Το Lucidchart [6], είναι μια online εφαρμογή σχεδίασης διαγραμμάτων που βασίζεται στην τεχνολογία HTLM5 και είναι συμβατή με όλες τις σύγχρονες συσκευές που έχουν την δυνατότητα περιήγησης στο internet. Πρόκειται για υπηρεσία που χρειάζεται συνδρομή και διαθέτει διάφορα πακέτα τιμολόγησης, όμως προσφέρει δωρεάν δοκιμή συγκεκριμένων λειτουργιών της μέσω free trial. Το Lucidchart υποστηρίζει την σχεδίαση διαγραμμάτων οντοτήτων-συσχετίσεων χρησιμοποιώντας μορφή UML. Η διαδικασία σχεδίασης είναι αρκετά απλή και βασίζεται σε drag & drop μηχανισμούς. Δεν υποστηρίζεται αυτόματη δημιουργία κώδικα SQL. Επιπλέον είδη διαγραμμάτων μπορούν να σχεδιαστούν όπως Flowchart, Venn Diagram, Network και

άλλα. Υπάρχει μηχανισμός που διατηρεί το ιστορικό αλλαγών σε ένα διάγραμμα με δυνατότητα επαναφοράς. Το διάγραμμα μπορεί να αποθηκευτεί τοπικά σε διάφορες μορφές όπως PDF, PNG, SVG και άλλες. Επιπλέον, είναι δυνατή η ενσωμάτωση του διαγράμματος σε ιστοσελίδα ή blog, και όποιες αλλαγές γίνονται στο αρχικό διάγραμμα αυτόματα εμφανίζονται και στο ενσωματωμένο.

Τα αρχεία που δημιουργούνται αποθηκεύονται απευθείας στον server και ο χρήστης μπορεί να τα οργανώσει σε φακέλους, αλλά και να μοιραστεί την πρόσβαση σε αυτά με μια ομάδα χρηστών.

3.3.2 SmartDraw



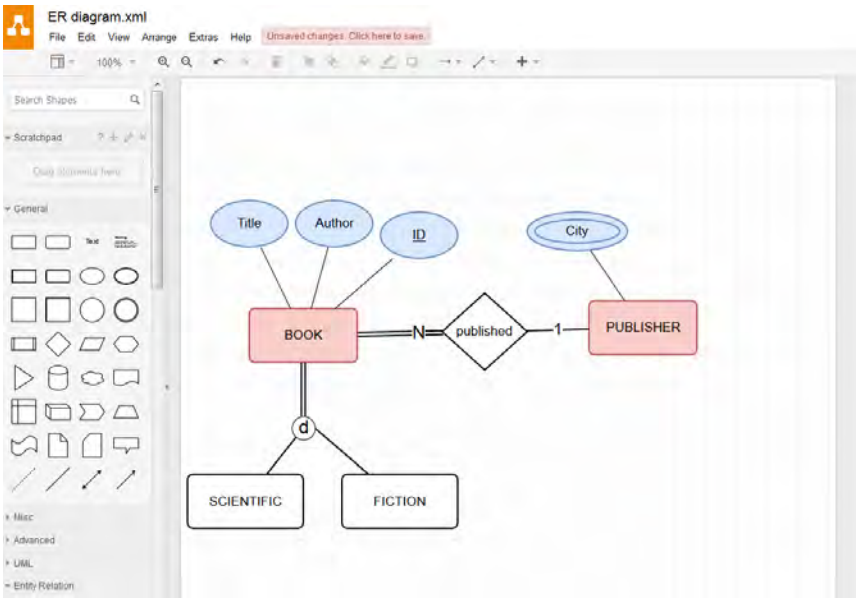
Σχήμα 3.11: SmartDraw

Το SmartDraw [7] είναι ακόμη μια εφαρμογή σχεδίασης διαγραμμάτων, που μπορεί να χρησιμοποιηθεί online ή να εγκατασταθεί σε λειτουργικά συστήματα Windows, εφόσον ο χρήστης το αγοράσει (one-time purchase), ενώ διαθέτει free trial 7 ημερών. Υποστηρίζεται το άνοιγμα αρχείων SmartDraw και Visio, η δε αποθήκευση των αρχείων μπορεί να γίνει σε cloud. Το SmartDraw υποστηρίζει τον σχεδιασμό διαγραμμάτων διαφόρων ειδών: Data Flow, Entity-Relationship, GUI, UML, και άλλα.

Επίσης υπάρχουν templates διαφόρων παραδειγμάτων που μπορούν να χρησιμοποιηθούν άμεσα.

Όσον αφορά την σχεδίαση διαγραμμάτων οντοτήτων - συσχετίσεων, υποστηρίζεται το απλό και εκτεταμένο μοντέλο ER με τους συμβολισμούς αντικειμένων που εξηγήθηκε στην ενότητα 2.3 με μικρές διαφοροποιήσεις, οι συνδέσεις όμως μεταξύ των αντικειμένων δεν έχουν συγκεκριμένους κανόνες. Για παράδειγμα, στις συσχετίσεις δεν υπάρχει συμβολισμός του λόγου πληθικότητας (1:N) δίπλα στις ακμές, αλλά ο χρήστης μπορεί να χρησιμοποιήσει διαφορετικά arrowheads της ακμής σε κάθε πλευρά για να δείξει τον λόγο πληθικότητας. Επιπλέον δεν υποστηρίζεται η δημιουργία διπλής γραμμής (ο χρήστης πάλι θα πρέπει να επιλέξει κατάλληλα arrowheads). Επιπλέον, όταν γίνεται μετακίνηση ενός αντικειμένου, τα ευθύγραμμα τμήματα που το ενώνουν με άλλα αντικείμενα δεν αλλάζουν μήκος με αποτέλεσμα να πρέπει να ξαναδημιουργηθούν. Το SmartDraw δεν υποστηρίζει έλεγχο λαθών ή μετατροπή σε SQL.

3.3.3 draw.io



Σχήμα 3.12: draw.io

Το draw.io είναι μια online δωρεάν εφαρμογή που επιτρέπει την σχεδίαση διαγραμμάτων και την αποθήκευσή τους σε Google Drive, Github, Dropbox, μεταξύ

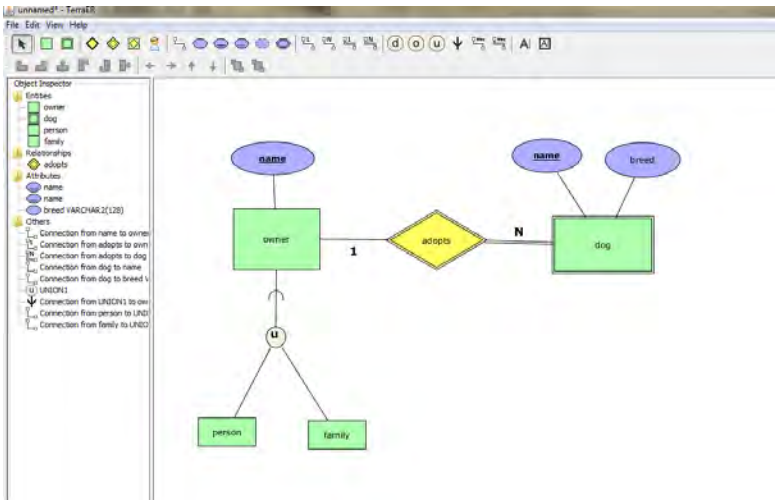
άλλων επιλογών. Υποστηρίζει την δημιουργία διαγραμμάτων ER μέσω ειδικής καρτέλας που περιέχει σύμβολα 'Entity Relation' του απλού μοντέλου, αλλά και σύμβολα από όλες τις κατηγορίες μπορούν να εισαχθούν στο ίδιο διάγραμμα. Είναι δυνατός ο χρωματισμός αντικειμένων όμως πρέπει να γίνεται σε κάθε αντικείμενο ξεχωριστά. Δεν υπάρχει αυστηρός συμβολισμός όσον αφορά την σύνδεση μεταξύ των σχημάτων ή την υποστήριξη του εκτεταμένου μοντέλου και για αυτό τον λόγο απαιτείται ο χρήστης να αυτοσχεδιάσει. Δυο παραδείγματα αυτής της παρατήρησης είναι τα εξής:

- Δεν υπάρχει επιλογή που να ορίζει συγκεκριμένα τον λόγο πληθικότητας στις ακμές μιας συσχέτισης, οπότε ο χρήστης πρέπει να επιλέξει διπλή γραμμή και να εισάγει κείμενο σε αυτήν (1 ή N).

- Δεν υπάρχει επιλογή δημιουργίας υπερκλάσης/εξειδίκευσης/ένωσης αλλά ο χρήστης μπορεί να δημιουργήσει κύκλο με όνομα 'd' και να τον συνδέσει με τις αντίστοιχες οντότητες.

Το draw.io δεν υποστηρίζει την δημιουργία κώδικα SQL.

3.3.4 TerraER



Σχήμα 3.13: TerraER

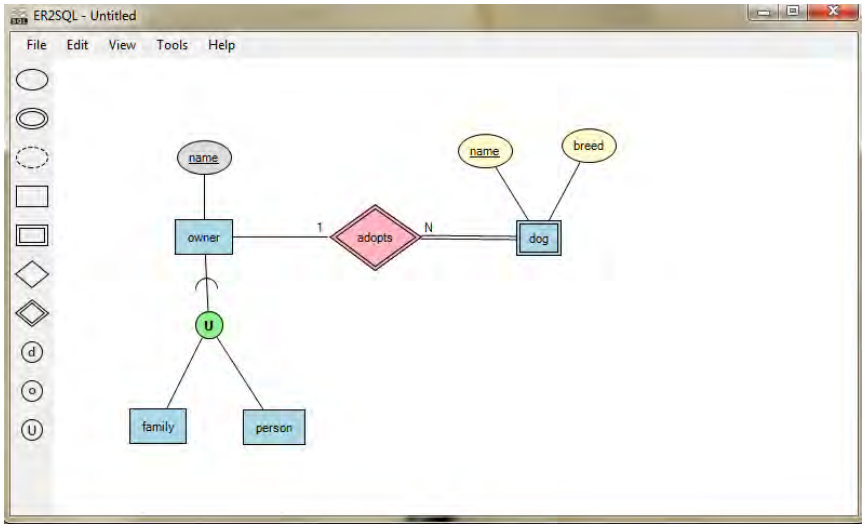
Το TerraER [9] είναι ένα δωρεάν open source πρόγραμμα που σχεδιάστηκε για να βοηθήσει μαθητές να κατανοήσουν τις έννοιες του μοντέλου ER. Είναι συμβατό με

λειτουργικά συστήματα Windows (Windows Vista και μετά) και έχει μέγεθος 2.6 MB. Για την λειτουργία του προγράμματος χρειάζεται να έχει εγκατασταθεί το Java Runtime Environment (JRE), όμως το TerraER δεν χρειάζεται εγκατάσταση (αρκεί μόνο να εκτελεστεί το εκτελέσιμο .jar αρχείο).

Καθώς σχεδιάστηκε αποκλειστικά για την δημιουργία διαγραμμάτων ER, το TerraER περιλαμβάνει έννοιες τόσο του απλού όσο και του εκτεταμένου μοντέλου (υπερκλάση, εξειδίκευση, ένωση). Τα αντικείμενα χρωματίζονται ανάλογα με την κατηγορία στην οποία ανήκουν, και υπάρχει δυνατότητα επιλογής του τύπου δεδομένων ενός γνωρίσματος. Στα αριστερά, υπάρχει λίστα των αντικειμένων που διευκολύνει την διαχείρισή τους.

Ένα μειονέκτημα του συγκεκριμένου προγράμματος είναι ότι μετά την δημιουργία ενός απλού γνωρίσματος, δεν γίνεται να χαρακτηριστεί ως κλειδί, οπότε πρέπει να διαγραφεί και να δημιουργηθεί νέο κλειδί στην θέση του. Επιπλέον, ενώ μπορεί να εισαχθεί ορίζον γνώρισμα σε μια εξειδίκευση, δεν υπάρχει αυτόματος τρόπος για να οριστούν οι τιμές αυτού του γνωρίσματος για την κάθε υποκλάση. Όπως τα εργαλεία που αναφέρθηκαν μέχρι τώρα, το TerraER περιορίζεται μόνο στην σχεδίαση διαγραμμάτων.

3.3.5 ER2SQL

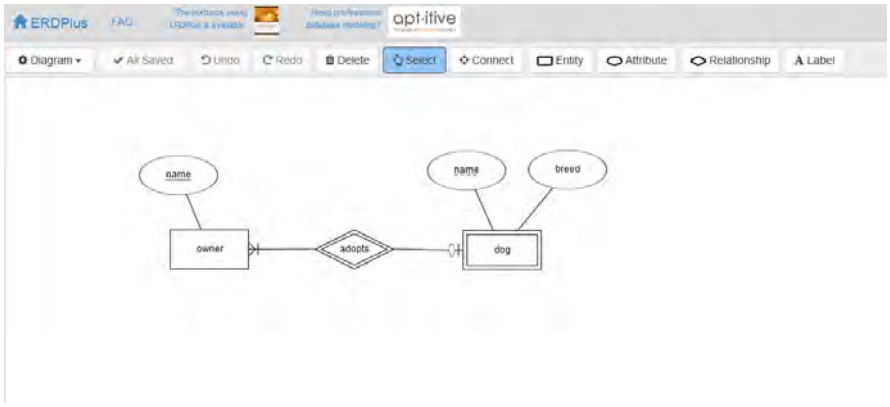


Σχήμα 3.14: ER2SQL

Το ER2SQL είναι αντικείμενο εργασίας [10], και αποτέλεσε κύρια έμπνευση για την ανάπτυξη της παρούσας διπλωματικής εργασίας. Πρόκειται για εφαρμογή που αναπτύχθηκε χρησιμοποιώντας γλώσσα C#, και μπορεί να εγκατασταθεί σε λειτουργικά συστήματα Windows, εφόσον υπάρχει εγκατεστημένο το .NET framework 4.0 ή νεότερες εκδόσεις του.

Το ER2SQL δίνει στον χρήστη την δυνατότητα να σχεδιάσει διαγράμματα οντοτήτων - συσχετίσεων χρησιμοποιώντας συμβολισμό Chen, και υποστηρίζει πλήρως το απλό και το εκτεταμένο μοντέλο ER. Επιπλέον, μπορεί να γίνει ορισμός του τύπου δεδομένων γνωρίσματος, το αν μπορεί να πάρει τιμές NULL ή όχι, και ο ορισμός πρωτεύοντος κλειδιού / υποψήφιου κλειδιού. Σκοπός της εφαρμογής είναι η μετατροπή διαγραμμάτων ER σε σχεσιακά / αντικειμενοσχεσιακά σχήματα (SQL-99 / PostgreSQL). Πρόσθετες λειτουργίες περιλαμβάνουν την εξαγωγή διαγράμματος σε διάφορες μορφές εικόνας καθώς και η αποθήκευση τους ως αρχεία χρησιμοποιώντας το πρότυπο XML.

3.3.6 ERDPlus



Σχήμα 3.15: ERDPlus

Το ERDPlus είναι μια δωρεάν online εφαρμογή σχεδίασης διαγραμμάτων οντοτήτων - συσχετίσεων που δεν χρειάζεται εγκατάσταση. Ο χρήστης μπορεί να δημιουργήσει δωρεάν λογαριασμό και να αποθηκεύσει τα διαγράμματα του στον server, εναλλακτικά μπορεί να μην δημιουργήσει λογαριασμό και να αποθηκεύει τα αρχεία τοπικά. Η σχεδίαση διαγραμμάτων ER βασίζεται μόνο στο απλό μοντέλο, και δεν υποστηρίζει χρωματισμό των αντικειμένων. Στην συνέχεια μπορεί να γίνει μετατροπή του διαγράμματος σε σχεσιακό σχήμα (relational schema) το οποίο είναι ένα δεύτερο διάγραμμα μορφής UML. Τέλος, από το σχεσιακό σχήμα μπορεί να γίνει εξαγωγή κώδικα SQL, αν και αυτό το στάδιο βρίσκεται ακόμα υπό έλεγχο (beta).

Οι εφαρμογές που αναφέρθηκαν και οι ιδιότητές τους συνοψίζονται στον παρακάτω πίνακα:

Εφαρμογή	Υποστήριξη Εκτεταμένου μοντέλου ER	Μετατροπή διαγράμματος σε σχεσιακό σχήμα	Λειτουργία online	Αποθήκευση αρχείων online	Χρήση χωρίς πληρωμή
SQL Draw	✓	✓	✓		✓
Lucidchart	✓		✓	✓	
SmartDraw	✓		✓	✓	
draw.io			✓	✓	✓
TerraER	✓				✓
ER2SQL	✓	✓			✓
ERDPlus		✓	✓	✓	✓

4. Δομή και Εργαλεία Ανάπτυξης

Σε αυτό το κεφάλαιο θα γίνει μια σύντομη ανάλυση των βασικών τεχνολογιών που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής SQL Draw προκειμένου να γίνει καλύτερα κατανοητό το κεφάλαιο 5, στο οποίο αναλύονται με μεγαλύτερη λεπτομέρεια οι συναρτήσεις της εφαρμογής και οι τεχνικές που οδήγησαν στο τελικό αποτέλεσμα. Η ανάλυση ξεκινάει με τις τρεις βασικές γλώσσες του web HTML, CSS και JavaScript και αργότερα αναφέρονται εργαλεία που βοήθησαν στο προγραμματιστικό κομμάτι της εφαρμογής.

4.1 HTML, CSS και JavaScript

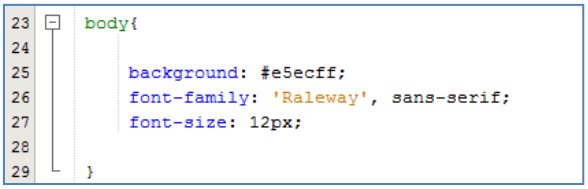
Η βασική ιδέα για το πως λειτουργεί μια web εφαρμογή είναι ότι χωρίζεται σε δυο κομμάτια [12]. Το κομμάτι της εφαρμογής που είναι ορατό στον χρήστη, όπως τα κείμενα, οι εικόνες, τα μενού κ.ά., και το κομμάτι που δεν είναι ορατό αλλά περιλαμβάνει λειτουργίες που επεξεργάζονται δεδομένα που εισάγει ο χρήστης και ενδεχομένως έχουν πρόσβαση σε κάποια βάση δεδομένων στον server. Στην πρώτη περίπτωση χρησιμοποιούνται γλώσσες που έχουν να κάνουν με το design της ιστοσελίδας και αλληλεπίδραση με τον χρήστη, ενώ στην δεύτερη χρησιμοποιούνται server-side γλώσσες (Java, Ruby, κ.ά.) και έχουν την δυνατότητα να επικοινωνούν με μια βάση δεδομένων που βρίσκεται στον server. Κάποιες ιστοσελίδες δεν χρησιμοποιούν καν βάση δεδομένων αλλά βασίζονται μόνο σε γλώσσες που μπορεί να καταλάβει ο περιηγητής διαδικτύου (browser). Οι γλώσσες που θα αναλυθούν (HTML, CSS και JavaScript) ανήκουν στην πρώτη κατηγορία, συνεπώς η εφαρμογή SQL Draw δεν χρησιμοποιεί καθόλου server-side λειτουργίες ούτε χρειάζεται δική της βάση δεδομένων.

4.1.1 HTML

Σχήμα 4.1: Βασικά HTML elements

Η γλώσσα HTML (Hypertext Markup Language) αποτελεί τον "σκελετό" μιας ιστοσελίδας [12], δηλαδή καθορίζει το περιεχόμενό της χρησιμοποιώντας διάφορα στοιχεία (elements). Δεν είναι γλώσσα προγραμματισμού αλλά δόμησης, και επιτρέπει την εισαγωγή κειμένου, συνδέσμων, εικόνων και άλλων αντικειμένων στην ιστοσελίδα. Κάθε στατική σελίδα μιας εφαρμογής είναι ουσιαστικά ένα αρχείο τύπου .html το οποίο μπορεί να διαβάσει ο περιηγητής και στη συνέχεια να εμφανίσει στην οθόνη τα στοιχεία που περιέχει το συγκεκριμένο αρχείο. Κάθε αρχείο .html αποτελείται από κάποια βασικά στοιχεία, τα οποία περιγράφονται από opening tags και closing tags όπως φαίνεται στο σχήμα 4.1. Φυσικά στην παρούσα εργασία έχουν χρησιμοποιηθεί πιο περίπλοκα στοιχεία αλλά προς το παρόν η βασική εικόνα αρκεί για την κατανόηση της δομής.

4.1.2 CSS



Σχήμα 4.2: Κανόνες CSS

Ενώ η HTML καθορίζει το περιεχόμενο μιας σελίδας, η γλώσσα CSS [12] καθορίζει το πως ακριβώς αυτό το περιεχόμενο παρουσιάζεται. Βασίζεται στην δημιουργία κανόνων που περιγράφουν την εμφάνιση συγκεκριμένων στοιχείων και τον τρόπο με τον οποίο διατάσσονται σε σχέση με τα υπόλοιπα. Με άλλα λόγια, η CSS μπορεί να

ορίσει ιδιότητες όπως το χρώμα και του τύπο γραμματοσειράς, το χρώμα φόντου, την απόσταση μεταξύ στοιχείων και πολλές ακόμη δυνατότητες. Μπορεί μια σελίδα να λειτουργεί άψογα και χωρίς την παρουσία κανόνων CSS αλλά αυτοί είναι απαραίτητοι για την αισθητική εικόνα της σελίδας. Το σχήμα 4.2 παρουσιάζει ως παράδειγμα τους κανόνες CSS που χρησιμοποιήθηκαν για το SQL Draw όσον αφορά το HTML στοιχείο 'body'.

4.1.3 JavaScript

Η JavaScript [13] είναι γλώσσα προγραμματισμού που σχεδιάστηκε για να εμπλουτίσει τις σελίδες HTML με επιπλέον δυνατότητες που περιλαμβάνουν αλληλεπίδραση με τον χρήστη, όπως drop-down μενού, animation, popup, εκτέλεση μικρών διαδικασιών ελέγχου και άλλες λειτουργίες. Ενώ συνήθως χρησιμοποιείται για λόγους που αφορούν το design, η JavaScript είναι μια γλώσσα με πολλές δυνατότητες που μπορεί να έχει τις δικές της μεταβλητές, δομές δεδομένων και μπορεί να υποστηρίξει περίπλοκους αλγορίθμους. Σε αντίθεση με server-side γλώσσες δεν μπορεί να επικοινωνήσει με βάσεις δεδομένων αλλά τρέχει αποκλειστικά στον περιηγητή του χρήστη. Η JavaScript συνήθως χρειάζεται να χρησιμοποιηθεί σε συνδυασμό με την HTML για να λειτουργήσει, και μπορεί να ενσωματωθεί ο κώδικάς της είτε απευθείας στην σελίδα είτε σε ξεχωριστό αρχείο.

4.2 Άλλα εργαλεία και βιβλιοθήκες

Bootstrap

Το Bootstrap [14] είναι ένα open source framework για HTML, CSS και JavaScript που διευκολύνει τον σχεδιασμό μιας ιστοσελίδας καθώς έχει έτοιμες κλάσεις στοιχείων που δεν χρειάζεται να γραφούν από την αρχή. Είναι ένα αρκετά διαδεδομένο και χρήσιμο εργαλείο ειδικά για την δημιουργία responsive σελίδων.

JQuery

Η JQuery [15] είναι μια βιβλιοθήκη της JavaScript που χρησιμοποιείται για την αλληλεπίδραση του χρήστη με HTML στοιχεία. Το ίδιο πράγμα μπορεί να γίνει με απλή JavaScript όμως η JQuery απλοποιεί πολύ την διαδικασία καθώς αυτοματοποιεί λειτουργίες όπως animation, event handling και άλλα.

SweetAlert

Το SweetAlert [16] είναι ένα εργαλείο που μπορεί να χρησιμοποιηθεί αντί για το παραδοσιακό alert(); ή prompt(); παράθυρο της JavaScript, είναι εμφανισιακά πολύ πιο όμορφο και μπορεί να προσαρμοστεί εύκολα στο design μιας σελίδας. Παράδειγμα του SweetAlert φαίνεται στο σχήμα 3.6. Χρησιμοποιείται από την συνάρτηση ελέγχου σφαλμάτων για να ειδοποιήσει τον χρήστη σχετικά με το αποτέλεσμα.

Snap.svg

Η Snap.svg [17] είναι μια βιβλιοθήκη JavaScript που επιτρέπει την δημιουργία γραφικών SVG (Scalable Vector Graphics) σε μια ιστοσελίδα. Το πλεονέκτημα ενός SVG σε σχέση με κάποιο κανονικό τύπο εικόνας είναι ότι δεν εξαρτάται από την ανάλυση της οθόνης, και δεν αλλοιώνεται η ποιότητά του όταν αλλάζει σε μέγεθος. Η βιβλιοθήκη αυτή παίζει σημαντικό ρόλο στο SQL Draw καθώς χρησιμοποιείται στην δημιουργία όλων των γραφικών που απεικονίζουν τα αντικείμενα ER και τις συνδέσεις μεταξύ τους. Η Snap.svg υποστηρίζει animations, και αλληλεπίδραση των γραφικών με τον χρήστη.

ContextMenu.js

Το ContextMenu.js [18] είναι ένα plugin που επιτρέπει την δημιουργία ενός μενού κάνοντας κλικ σε κάποιο αντικείμενο της σελίδας. Το μενού μπορεί να προσαρμοστεί στις ανάγκες του σχεδιαστή και επικαλύπτει το default μενού που έχει ο κάθε περιηγητής όταν γίνεται δεξί κλικ πάνω σε κάποιο αντικείμενο. Το συγκεκριμένο εργαλείο χρησιμοποιήθηκε για την ανάπτυξη του μενού κάθε ενός είδους σχημάτων

όπως εξηγήθηκε στο κεφάλαιο 3. Κάθε επιλογή του μενού μπορεί να συνδεθεί με κομμάτι κώδικα που εκτελείται εφόσον γίνει κλικ επάνω της.

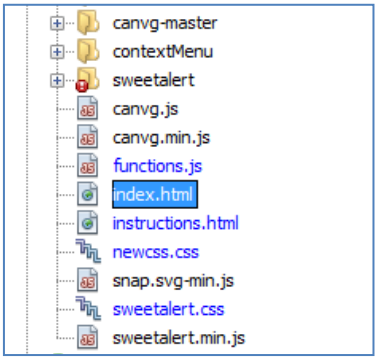
Canvg

Το Canvg [19] είναι ένα εργαλείο ανάλυσης αντικειμένων SVG. Μεταξύ άλλων, μια χρήση του περιλαμβάνει την μετατροπή SVG σε εικόνα PNG χωρίς να χρειαστεί η χρήση server-side γλωσσών. Το συγκεκριμένο εργαλείο χρησιμοποιείται για την μετατροπή ενός διαγράμματος ER (που αποτελείται από SVG στοιχεία) σε εικόνα PNG.

FileSaver.js

Το FileSaver.js [20] δίνει την δυνατότητα αποθήκευσης αρχείων σε μια εφαρμογή web χωρίς να χρειάζεται επικοινωνία της εφαρμογής με κάποιο server. Χρησιμοποιείται για την αποθήκευση διαγραμμάτων σε μορφή αρχείων κειμένου .txt, αλλά και στην αποθήκευση του παραγόμενου κώδικα SQL σε μορφή αρχείων κειμένου.

4.3 Δομή και περιεχόμενο αρχείων



Σχήμα 4.3: Αρχεία εφαρμογής

Με το άνοιγμα του φακέλου με όνομα 'web', ο οποίος περιέχει τα αρχεία της εφαρμογής, βλέπουμε τα αρχεία και τους φακέλους του σχήματος 4.3. Υπάρχουν φάκελοι όπως οι canvg-master, contextMenu, sweetalert και διάφορα .js αρχεία, για την λειτουργία των εργαλείων που αναφέρθηκαν στην ενότητα 4.2. Τα αρχεία και οι

φάκελοι δεν πρέπει να αφαιρεθούν. Στην συνέχεια θα εξηγηθούν τα βασικά αρχεία της εφαρμογής.

index.html:

Πρόκειται για την αρχική σελίδα της εφαρμογής, η οποία περιέχει την περιοχή σχεδίασης και όλες τις λειτουργίες του SQL Draw.

instructions.html:

Αυτή η σελίδα εμφανίζεται όταν ο χρήστης κάνει κλικ στον σύνδεσμο 'HOW TO USE' στο κάτω μέρος της οθόνης.

newcss.css:

Το αρχείο css στο οποίο περιέχονται οι κανόνες που αφορούν το design των σελίδων .html.

functions.js:

Σε αυτό το αρχείο περιέχεται ο κώδικας JavaScript που αφορά τις λειτουργίες της εφαρμογής όπως η δημιουργία γραφικών, αποθήκευση, μετατροπή σε PostgreSQL κ.λπ.

4.4 Ενσωμάτωση του SQL Draw σε web server

Για να λειτουργήσει η εφαρμογή SQL Draw, δηλαδή να γίνει χρήση του κώδικα JavaScript στο αρχείο functions.js, βασική προϋπόθεση είναι να υπάρχει κάποιο αρχείο .html. Αυτό το αρχείο θα αποτελεί μία σελίδα στο web. Υπάρχει κάποιο στοιχείο που πρέπει υποχρεωτικά να βρίσκεται στο αρχείο αυτό, το οποίο αποτελεί την περιοχή σχεδίασης, και το οποίο είναι το εξής:

```
<div  id = "drawing-box" >
    <svg style="width:100%; height: 100%;" id = "svg" ></svg>
    <canvas id="canvas" style = "display:none;"></canvas>
</div>
```

Επιπλέον, πρέπει στο αρχείο .html να υπάρχουν σύνδεσμοι που το συνδέουν με όλα τα εργαλεία που χρειάζεται η εφαρμογή. Αυτό γίνεται με δυο τρόπους: εισαγωγή στοιχείου <script></script> με την τιμή src ίση με κάποιο link (δεν χρειάζονται τα αρχεία της βιβλιοθήκης) ή με την τιμή src να δείχνει προς κάποιο αρχείο στο directory της εφαρμογής. Για παράδειγμα, η χρήση του κώδικα του αρχείου functions.js προϋποθέτει να υπάρχει το αρχείο functions.js στον ίδιο φάκελο με το αρχείο index.html και στο index.html να υπάρχει το στοιχείο:

```
<script src="functions.js"></script>
```

Τα στοιχεία <script></script> πρέπει να τοποθετηθούν είτε στο στοιχείο <head></head> είτε ακριβώς πριν το κλείσιμο του <body></body> ενός .html αρχείου. Για την συνολική λίστα των <script></script> που πρέπει να χρησιμοποιηθούν, συμβουλευτείτε το αρχείο index.html.

Εφόσον όλες οι προϋποθέσεις έχουν πραγματοποιηθεί, για να λειτουργήσουν οι συναρτήσεις της εφαρμογής θα πρέπει με κάποιο τρόπο να τις καλέσουμε. Η JavaScript όπως αναφέρθηκε έχει την δυνατότητα να αλληλεπιδρά με τον χρήστη. Στην περίπτωση του SQL Draw, με το πάτημα κουμπιών (buttons) εκτελούνται οι διάφορε εφαρμογές. Για παράδειγμα, η συνάρτηση ConvertToSQL() που μετατρέπει το διάγραμμα σε PostgreSQL καλείται ως εξής:

```
<button type="button" class="btn btn-primary" onclick="convertToSQL()">Convert to PostgreSQL</button>
```

Χρησιμοποιώντας την ιδιότητα onclick="<function name (arguments)>" μέσα στο button tag, δηλώνουμε ποια συνάρτηση θέλουμε να καλέσουμε όταν γίνει κλικ στο συγκεκριμένο κουμπί. Με ίδιο τρόπο καλούνται και όλες οι υπόλοιπες συναρτήσεις που θα εξηγηθούν στο κεφάλαιο 5.

Όταν το αρχείο .html έχει ολοκληρωθεί, και εφόσον όλα δουλεύουν σωστά, πρέπει να γίνει deploy (μαζί φυσικά με όλα τα αρχεία που το συνοδεύουν) σε κάποιον server. Αυτό μπορεί να γίνει με κάποια υπηρεσία φιλοξενίας ιστοσελίδων που προσφέρει την αγορά domain ή με κάποια δωρεάν υπηρεσία (π.χ. Heroku, GitHub Pages Hosting). Ο τρόπος που θα ανέβουν τα αρχεία αυτά στον server εξαρτάται από τις οδηγίες που προσφέρει η κάθε υπηρεσία.

5. Συναρτήσεις της εφαρμογής SQL Draw

5.1 Δημιουργία και επεξεργασία αντικειμένων

Σε αυτήν την ενότητα θα αναλυθούν οι συναρτήσεις που καλούνται όταν γίνεται κλικ σε κάποια επιλογή του μενού 'New', δηλαδή οι συναρτήσεις που δημιουργούν το αντικείμενο και το τοποθετούν στον χώρο, καθώς και συγκεκριμένες λειτουργίες επεξεργασίας των αντικειμένων. Αρχικά, ας δούμε ξεχωριστά κάθε συνάρτηση δημιουργίας.

createRect ();

```
var r = Snap('#svg');

if(rect_type === "entity"){

    var newRect = r.rect(rx,ry,110,55);
    newRect.attr({
        fill:'#ccffe6',
        stroke:'#000',
        strokeWidth: 2
    });

     newRect.attr({

        id: "entity"

      })

}
```

Σχήμα 5.1: Τμήμα της συνάρτησης createRect (), όπου γίνεται δημιουργία ενός ισχυρού τύπου οντοτήτων.

Η συνάρτηση αυτή χρησιμοποιείται για την δημιουργία ισχυρών και ασθενών οντοτήτων. Ο πλήρης ορισμός της είναι:

createRect (rect_type, rect_name, superPermanentID, specializationList, defCriteriaList, unionSubsList, rx, ry);

Τα ορίσματα rect_type, rect_name καθορίζουν το είδος της οντότητας και το όνομά της, αντίστοιχα. Το όρισμα rect_type μπορεί να πάρει τιμές "entity", "weak_entity" και ανάλογα με την τιμή του καθορίζεται και η μορφή του σχήματος. Το όρισμα superPermanentID παίρνει ως τιμή το μοναδικό ID της υπερκλάσης της οντότητας. Σε περίπτωση που η οντότητα δεν έχει υπερκλάση τότε η τιμή αυτή είναι "none". Το όρισμα specializationList είναι μια λίστα που περιέχει όλες τις πιθανές εξειδικεύσεις στις οποίες η οντότητα αποτελεί υποκλάση. Το όρισμα defCriteriaList περιέχει τυχόν κριτήρια για την κάθε εξειδίκευση. Το όρισμα unionSubsList περιέχει όλες τις ενώσεις για τις οποίες η οντότητα αποτελεί υπερκλάση. Όταν η οντότητα δημιουργείται για πρώτη φορά, η τιμή των ορισμάτων αυτών είναι "none". Τα ορίσματα rx, ry δηλώνουν τις συντεταγμένες του σχήματος στον χώρο.

Η συγκεκριμένη συνάρτηση, δημιουργεί το γραφικό σχήμα χρησιμοποιώντας την βιβλιοθήκη Snap.svg δίνοντάς του όνομα *newRect*. Ένα από τα χαρακτηριστικά πλεονεκτήματα της βιβλιοθήκης είναι ότι τα γραφικά που δημιουργούνται αντιμετωπίζονται από την JavaScript ως κανονικές μεταβλητές, δηλαδή ένα σχήμα είναι ταυτόχρονα και μεταβλητή. Συνεπώς η συνάρτηση δημιουργίας τοποθετεί κάθε νέα μεταβλητή σε μια λίστα με αντικείμενα όμοιου είδους, για την μελλοντική επεξεργασία τους. Επίσης αρχικοποιεί και άλλες παραμέτρους όπως την λίστα των γνωρισμάτων που διαθέτει κάθε οντότητα, ή το μοναδικό ID που την ξεχωρίζει από τις υπόλοιπες της λίστας. Επίσης, η συνάρτηση συνδέει το αντικείμενο με συγκεκριμένα events και τις αντίστοιχες συναρτήσεις όπως μετακίνηση και αυξομείωση μεγέθους. Τέλος, η συνάρτηση φροντίζει και για την διατήρηση του σχήματος στο εσωτερικό της περιοχής σχεδίασης. Όσα αναφέρθηκαν ισχύουν και για όλα τα υπόλοιπα σχήματα-μεταβλητές που θα αναλυθούν παρακάτω.

createCircle();

Η συνάρτηση αυτή χρησιμοποιείται για την δημιουργία εξειδικεύσεων και ενώσεων. Ο πλήρης ορισμός της είναι:

createCircle (c_type, permanentSuperID, defAttr, superDouble, permanentSubID, subDouble, x, y);

Το όρισμα c_type παίρνει τιμές "disjoint", "overlapping", "union" και καθορίζει την μορφή και το μενού του σχήματος. Το όρισμα permanentSuperID δηλώνει την υπερκλάση μιας εξειδίκευσης, το όρισμα defAttr δηλώνει κάποιο ορίζον γνώρισμα εξειδίκευσης εάν υπάρχει, το όρισμα superDouble δηλώνει μερική η ολική συμμετοχή εξειδίκευσης, το όρισμα permanentSubID δηλώνει την υποκλάση μιας ένωσης και το όρισμα subDouble δηλώνει μερική ή ολική συμμετοχή ένωσης. Τέλος, τα ορίσματα x, y δείχνουν την θέση στην οποία πρέπει να τοποθετηθεί το σχήμα. Προσέξτε ότι οι μεταβλητές αυτού του τύπου δεν έχουν όνομα, παρά μόνο το μοναδικό ID τους.

createEllipse();

Η συνάρτηση αυτή χρησιμοποιείται για την δημιουργία γνωρισμάτων (απλά γνωρίσματα, πλειότιμα, παραγόμενα). Ο πλήρης ορισμός της είναι:

createEllipse(ell_type, ell_name, data, n, i, j, primary, unique, notnull, ownerPermanentID, x, y);

Το όρισμα ell_type δείχνει το είδος του γνωρίσματος, και καθορίζει την μορφή του σχήματος και το μενού του. Το όρισμα ell_name δηλώνει το όνομα του αντικειμένου. Τα ορίσματα data, n, i, j σχετίζονται με τον τύπο δεδομένων του ορίσματος όπως αναλύθηκε στο κεφάλαιο 3. Στην συνέχεια ακολουθούν τα ορίσματα primary, unique, notnull που καθορίζουν την ιδιότητα ενός γνωρίσματος να είναι πρωτεύον κλειδί, υποψήφιο κλειδί, και not NULL αντίστοιχα. Αυτά τα ορίσματα παίρνουν τιμές true ή false. Το όρισμα ownerPermanentID δηλώνει τον μοναδικό ιδιοκτήτη του γνωρίσματος. Τέλος, τα ορίσματα x, y δηλώνουν την θέση του σχήματος στον χώρο.

createRhombus();

Η συνάρτηση αυτή χρησιμοποιείται για την δημιουργία συσχετίσεων (απλές συσχετίσεις, προσδιορίζουσες). Ο πλήρης ορισμός της είναι:

createRhombus(rhombus_type, rhombus_name, topID, botID, leftID, rightID, topDouble, botDouble, leftDouble, rightDouble, topRatio, botRatio, leftRatio, rightRatio, x, y);

Το όρισμα rhombus_type δηλώνει το είδος της συσχέτισης, δηλαδή εάν πρόκειται για απλή ή προσδιορίζουσα. Το όρισμα rhombus_name δηλώνει το όνομα του αντικειμένου. Στην συνέχεια ακολουθεί μια σειρά ορισμάτων που αφορά χαρακτηριστικά της κάθε μιας από τις τέσσερις συνδέσεις του ρόμβου. Τα ορίσματα topID, botID, leftID, rightID ορίζουν τον τύπο οντότητας στον οποίο καταλήγει η κάθε σύνδεση. Τα ορίσματα topDouble, botDouble, leftDouble, rightDouble περιγράφουν κατά πόσο η κάθε σύνδεση είναι μερική ή ολική. Τα ορίσματα topRatio, botRatio, leftRatio, rightRatio δηλώνουν τον λόγο πληθικότητας κάθε σύνδεσης (1, N, M, K, L). Τέλος, τα ορίσματα x, y δηλώνουν την θέση του σχήματος στον χώρο.

resizeHandler();

Η συγκεκριμένη συνάρτηση είναι υπεύθυνη για την επεξεργασία του μεγέθους ενός σχήματος. Καλείται χωρίς ορίσματα, αρκεί να υπάρχει η συγκεκριμένη εντολή στις συναρτήσεις δημιουργίας: newRect.dblclick(resizeHandler);. Αυτό σημαίνει ότι για κάθε νέο αντικείμενο τύπου οντότητας, γίνεται εκτέλεση της συνάρτησης resizeHandler με διπλό κλικ στο σχήμα. Η resizeHandler επίσης ελέγχει και το σχήμα του cursor του ποντικού.

start(); , move(dx,dy); , stop();

Αυτές οι συναρτήσεις χρησιμοποιούνται στην συνάρτηση resizeHandler που αναφέρθηκε παραπάνω, και ελέγχουν την λειτουργία resize. Η συνάρτηση move(dx,dy); παίρνει ως όρισμα τις αποστάσεις dx, dy του cursor από την θέση του σχήματος και εκτελεί την επεξεργασία μεγέθους (scaling).

Επιπλέον συναρτήσεις: Context Menu

```
rename: {
  name: "Rename",
  callback: function(key, opt) {

  var str = prompt("Please enter a name.", "Name");


      str = str.replace(/ /g,"_");

      if (str !== null) {

         Snap(this[0]).name = str;

      }

  }
```

Σχήμα 5.2: Μετονομασία αντικειμένου με το ContextMenu.

Κάθε είδος αντικειμένου έχει το δικό του μενού που υποστηρίζεται από το εργαλείο ContextMenu.js. Το συγκεκριμένο εργαλείο δίνει την δυνατότητα εκτέλεσης τμήματος κώδικα κάνοντας κλικ σε κάποια επιλογή του μενού χωρίς να χρειάζεται κλήση κάποιας άλλης συνάρτησης. Επομένως κάποιες απλές σχετικά λειτουργίες εκτελούνται απευθείας, όπως η μετονομασία αντικειμένου ή η διαγραφή αντικειμένων/συνδέσεων. Στο σχήμα 5.2 δίνεται ένα παράδειγμα μετονομασίας τύπου οντοτήτων.

5.2 Συναρτήσεις συνδέσεων

Σε αυτήν την ενότητα θα αναλυθούν οι συναρτήσεις που σχετίζονται με τις συνδέσεις αντικειμένων μεταξύ τους. Παραδείγματα αυτής της κατηγορίας λειτουργιών μπορούν να θεωρηθούν: η σύνδεση γνωρίσματος με τον ιδιοκτήτη του, συνδέσεις τύπων οντοτήτων με συσχετίσεις, συνδέσεις εξειδικεύσεων / ενώσεων.

clickTrigger();

```
"owner": {
        "name": "Owner",
        "items": {
            "nowner-key1": {"name": "Set",
         callback: function(key, opt) {

            if( Snap(this[0]).owner !== 0){

                var indexE = Snap(this[0]).owner.attributes.indexOf(Snap(this[0]).ownerLine);  // remove old owner first
                Snap(this[0]).owner.attributes.splice(indexE,1);
                Snap(this[0]).ownerLine.remove();
                Snap(this[0]).owner = 0;


        }

            // make this shape a start point
              L.start = Snap(this[0]);
             // console.log("START " + L.start.cx + ", " + L.start.cy);
              line_cr = 2;


        },
```

Σχήμα 5.3: Προετοιμασία για σύνδεση γνωρίσματος με τύπο οντοτήτων

Η συνάρτηση clickTrigger(); παίζει σημαντικό ρόλο σε όλες τις λειτουργίες σύνδεσης καθώς επιλέγει ποια συνάρτηση σύνδεσης θα κληθεί. Η clickTrigger καλείται κάθε φορά που γίνεται αριστερό κλικ σε οποιοδήποτε αντικείμενο μέσω της εντολής variable_name.click(this.clickTrigger); που εισάγεται σε όλες τις συναρτήσεις δημιουργίας. Την θέση του variable_name παίρνει το όνομα της κάθε νέας μεταβλητής, δηλαδή newRect, newCircle, κλπ. Όταν πρόκειται να γίνει κάποια σύνδεση, εκτελούνται ορισμένες προετοιμασίες από το σύστημα. Στην συνέχεια η clickTrigger ελέγχει αυτές τις προετοιμασίες και ανάλογα με τα αποτελέσματα του ελέγχου, διαλέγει ποια λειτουργία σύνδεσης θα εκτελεστεί. Για καλύτερη κατανόηση του μηχανισμού αυτού θα γίνει επεξήγηση μέσω παραδείγματος.

Σαν παράδειγμα θα χρησιμοποιηθεί η περίπτωση σύνδεσης γνωρίσματος με τύπο οντοτήτων. Για να εκτελεστεί αυτή η σύνδεση χρειάζονται δυο μεταβλητές: ένα γνώρισμα και ένας τύπος οντότητας. Επίσης, όπως αναφέρθηκε στο κεφάλαιο 3, η διαδικασία για την σύνδεση γνωρίσματος με οντότητα είναι η εξής: δεξί κλικ στο σχήμα γνωρίσματος και από το μενού που προκύπτει επιλογή Owner και μετά Set.

Με την επιλογή Set ορίζεται από το σύστημα η πρώτη μεταβλητή σύνδεσης, δηλαδή το γνώρισμα. Αποθηκεύεται προσωρινά ως 'αρχικό σημείο' μιας global μεταβλητής με όνομα 'L'. Επίσης, αλλάζει η τιμή της global μεταβλητής line_cr από 0 σε 2. Επίσης,

εάν το γνώρισμα έχει ήδη ιδιοκτήτη, η παλιά σύνδεση διαγράφεται για να υλοποιηθεί η καινούρια. Η προετοιμασία αυτή φαίνεται στο σχήμα 5.3, στην επιλογή Set του ContextMenu. Στην συνέχεια, μένει να επιλεγεί με αριστερό κλικ το σχήμα του τύπου οντοτήτων που έχει δημιουργηθεί.

Σε αυτό το σημείο η συνάρτηση clickTrigger εκτελείται και ελέγχει την τιμή της μεταβλητής line_cr. Εφόσον η τιμή της είναι 2 και το αντικείμενο στο οποίο έγινε κλικ επιτρέπεται να γίνει ιδιοκτήτης, τότε επιλέγεται ως τελικό σημείο της μεταβλητής L, γίνεται reset της line_cr και επιλέγεται προς εκτέλεση η συνάρτηση connectAttrToOwner (); η οποία εκτελεί την σύνδεση. Η διαδικασία ελέγχου της clickTrigger φαίνεται στο σχήμα 5.4 (γραμμή ~ 5700 του αρχείου functions.js).

```
else if ((line_cr === 2)  && (L.start !== this) && (this.type === "entity" || this.type === "weak_entity"

  L.end = this;
  // console.log("END " + L.end.cx + ", " + L.end.cy);
  line_cr = 0;

 if(L.end.type === "attribute" || L.end.type === "multi_attribute"){

     if( L.end.owner.type === "attribute" || L.end.owner.type === "multi_attribute"){ return; }

 }

  if(L.start.type === "multi_attribute" && L.end.type === "multi_attribute") { return; }

  connectAttrToOwner ();


}
```

Σχήμα 5.4: Έλεγχος της clickTrigger για σύνδεση γνωρίσματος με τύπο οντοτήτων

connectAttrToOwner ();

Η συνάρτηση αυτή χρησιμοποιεί τις προσωρινές μεταβλητές L.start και L.end για να συνδέσει ένα γνώρισμα με τον ιδιοκτήτη του. Αρχικά, δημιουργείται το σχήμα του ευθύγραμμου τμήματος που απεικονίζει την σύνδεση, χρησιμοποιώντας τις συντεταγμένες του αρχικού και τελικού σημείου ώστε το ευθύγραμμο τμήμα να παραμένει συνδεδεμένο ακόμη και όταν τα αντικείμενα μετακινούνται. Επίσης, η συνάρτηση προσθέτει το γνώρισμα στην λίστα attributes του ιδιοκτήτη, και η μεταβλητή-ιδιοκτήτης ορίζεται σαν owner του γνωρίσματος. Με παρόμοιο τρόπο

γίνονται και όλες οι υπόλοιπες συνδέσεις, οπότε δεν θα εξηγηθούν με πολύ μεγάλη λεπτομέρεια.

connectToSuper();

Η συνάρτηση αυτή χρησιμοποιεί τις προσωρινές μεταβλητές L.start και L.end για να συνδέσει έναν τύπο οντοτήτων ή μια εξειδίκευση με έναν διαφορετικό τύπο οντοτήτων που αποτελεί υπερκλάση του πρώτου.

connectRelToEntityPartial ();

Η συνάρτηση αυτή χρησιμοποιεί τις προσωρινές μεταβλητές L.start και L.end για να συνδέσει μια συσχέτιση με έναν τύπο οντοτήτων, χρησιμοποιώντας μερική συμμετοχή. Γίνεται χρήση ειδικών μεταβλητών που ορίζουν την ύπαρξη διπλής ή μονής γραμμής και λόγου πληθικότητας.

connectRelToEntityTotal ();

Η συνάρτηση αυτή χρησιμοποιεί τις προσωρινές μεταβλητές L.start και L.end για να συνδέσει μια συσχέτιση με έναν τύπο οντοτήτων, χρησιμοποιώντας ολική συμμετοχή. Γίνεται χρήση ειδικών μεταβλητών που ορίζουν την ύπαρξη διπλής ή μονής γραμμής και λόγου πληθικότητας.

connectEntityToSpecialization ();

Η συνάρτηση αυτή χρησιμοποιεί τις προσωρινές μεταβλητές L.start και L.end για να συνδέσει έναν τύπο οντοτήτων με μια εξειδίκευση, ώστε ο τύπος οντοτήτων να θεωρείται υποκλάση.

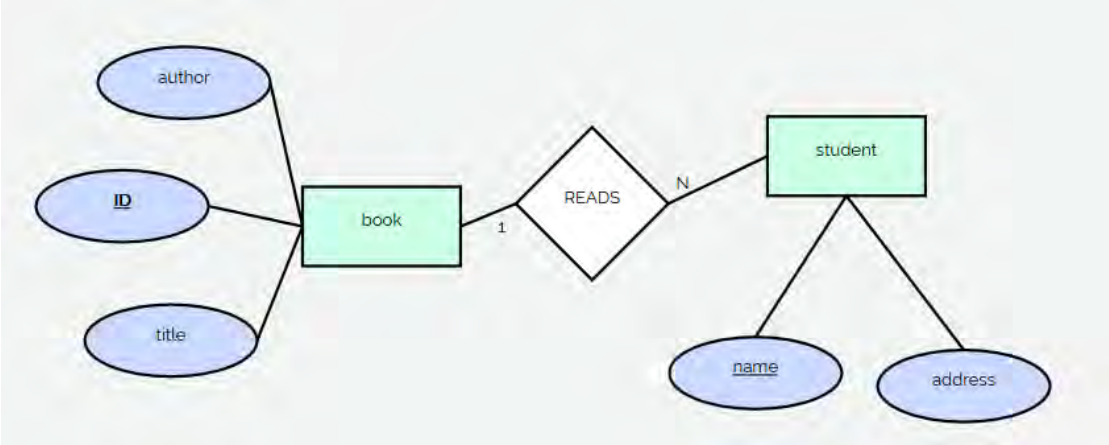
connectUnionToSuper ();

Η συνάρτηση αυτή χρησιμοποιεί τις προσωρινές μεταβλητές L.start και L.end για να προσθέσει σε μια ένωση, μια σύνδεση με τύπο οντότητας που θεωρείται υπερκλάση της ένωσης.

connectEntityToUnion ();

Η συνάρτηση αυτή χρησιμοποιεί τις προσωρινές μεταβλητές L.start και L.end για να συνδέσει έναν τύπο οντότητας με μια ένωση, ώστε ο τύπος οντότητας να αποτελεί μοναδική υποκλάση την ένωσης.

5.3 Αποθήκευση και άνοιγμα αρχείου



Σχήμα 5.5: Διάγραμμα που περιγράφει συσχέτιση μεταξύ φοιτητή και βιβλίου

Η εφαρμογή SQL Draw όπως αναφέρθηκε στο κεφάλαιο 3, υποστηρίζει την αποθήκευση διαγραμμάτων σε μορφή αρχείων κειμένου .txt, με σκοπό το άνοιγμα των αρχείων αυτών σε μελλοντικό χρόνο. Η βασική ιδέα του μηχανισμού είναι η εξής:

Κατά την αποθήκευση, η εφαρμογή δημιουργεί ένα αρχείο κειμένου που βασίζεται στην γλώσσα XML. Η γλώσσα XML είναι μια markup language που επιτρέπει την κωδικοποίηση αρχείων που διαβάζονται από ανθρώπους και υπολογιστικά

συστήματα. Επομένως, κατά την δημιουργία του αρχείου, επιλέγονται διάφορα tags που περιγράφουν όλα τα χαρακτηριστικά του κάθε ενός αντικειμένου στο διάγραμμα και οποιαδήποτε σύνδεση υπάρχει μεταξύ τους. Στο σχήμα 5.5 απεικονίζεται ένα διάγραμμα, και το αρχείο μορφής XML που παράγεται από αυτό το διάγραμμα έχει το εξής περιεχόμενο.

```
<?xml version="1.0" encoding="UTF-8"?>

<diagram>

	<entity>

		<name>book</name>
		<type>entity</type>
		<x>634</x>
		<y>312</y>
		<entsuper>none</entsuper>
		<specsuper>none</specsuper>
		<defcriteria>none</defcriteria>
		<unionsubs>none</unionsubs>

	</entity>

	<entity>

		<name>student</name>
		<type>entity</type>
		<x>987</x>
		<y>254</y>
		<entsuper>none</entsuper>
		<specsuper>none</specsuper>
		<defcriteria>none</defcriteria>
		<unionsubs>none</unionsubs>

	</entity>

	<attribute>

		<a-name>author</a-name>
		<a-type>attribute</a-type>
		<a-data>CHARACTER</a-data>
		<a-n>20</a-n>
		<a-i>0</a-i>
		<a-j>0</a-j>
		<a-primary>false</a-primary>
		<a-unique>false</a-unique>
		<a-notnull>false</a-notnull>
		<a-x>544</a-x>
		<a-y>226</a-y>
		<a-owner>ent0</a-owner>

	</attribute>

	<attribute>

		<a-name>ID</a-name>
		<a-type>attribute</a-type>
```

```
        <a-data>INTEGER</a-data>
        <a-n>1</a-n>
        <a-i>0</a-i>
        <a-j>0</a-j>
        <a-primary>true</a-primary>
        <a-unique>true</a-unique>
        <a-notnull>true</a-notnull>
        <a-x>497</a-x>
        <a-y>328</a-y>
        <a-owner>ent0</a-owner>

    </attribute>

    <attribute>

        <a-name>address</a-name>
        <a-type>attribute</a-type>
        <a-data>CHARACTER VARYING</a-data>
        <a-n>25</a-n>
        <a-i>0</a-i>
        <a-j>0</a-j>
        <a-primary>false</a-primary>
        <a-unique>false</a-unique>
        <a-notnull>false</a-notnull>
        <a-x>1182</a-x>
        <a-y>348</a-y>
        <a-owner>ent1</a-owner>

    </attribute>

    <attribute>

        <a-name>name</a-name>
        <a-type>attribute</a-type>
        <a-data>CHARACTER</a-data>
        <a-n>20</a-n>
        <a-i>0</a-i>
        <a-j>0</a-j>
        <a-primary>false</a-primary>
        <a-unique>true</a-unique>
        <a-notnull>true</a-notnull>
        <a-x>990</a-x>
        <a-y>179</a-y>
        <a-owner>ent1</a-owner>

    </attribute>

    <attribute>
        <a-name>title</a-name>
        <a-type>attribute</a-type>
        <a-data>CHARACTER</a-data>
        <a-n>20</a-n>
        <a-i>0</a-i>
        <a-j>0</a-j>
        <a-primary>false</a-primary>
        <a-unique>false</a-unique>
        <a-notnull>false</a-notnull>
        <a-x>534</a-x>
        <a-y>439</a-y>
        <a-owner>ent0</a-owner>
    </attribute>
```

```
    <relationship>

        <r-name>READS</r-name>
        <r-type>relationship</r-type>
        <r-x>825.4648</r-x>
        <r-y>342.4593999999999</r-y>
        <r-top>none</r-top>
        <r-bot>none</r-bot>
        <r-left>ent0</r-left>
        <r-right>ent1</r-right>
        <r-topd>false</r-topd>
        <r-botd>false</r-botd>
        <r-leftd>false</r-leftd>
        <r-rightd>false</r-rightd>
        <r-topr>none</r-topr>
        <r-botr>none</r-botr>
        <r-leftr>1</r-leftr>
        <r-rightr>N</r-rightr>
    </relationship>

    <client-height>789</client-height>

    <client-width>1711</client-width>

</diagram>
```

Ίσως προσέξατε ότι ενώ στο γνώρισμα με όνομα 'author' υπάρχει το tag <a-owner>ent0</a-owner>, η τιμή ent0 δεν υπάρχει σε κανένα αντικείμενο. Αυτό γίνεται διότι η αποθήκευση της πληροφορίας αυτής είναι περιττή. Όταν δημιουργείται ένα αντικείμενο παίρνει ως μοναδικό ID ένα prefix (εδώ 'ent') ακολουθούμενο από έναν αριθμό που ξεκινάει από 0 και αυξάνεται κατά 1 με κάθε νέο αντικείμενο. Τα αντικείμενα αποθηκεύονται με την ίδια σειρά με την οποία δημιουργούνται, οπότε το ID ent0 αναφέρεται στον τύπο οντότητας με όνομα 'book', ο οποίος θα πάρει το ID ent0 όταν δημιουργηθεί ξανά.

Κατά το άνοιγμα αρχείου, η εφαρμογή διαβάζει το αρχείο κειμένου που περιέχει την περιγραφή όλων των αντικειμένων, και καλεί ξανά τις κατάλληλες συναρτήσεις δημιουργίας για το κάθε αντικείμενο. Αφού όλα τα αντικείμενα έχουν δημιουργηθεί, οι επιπλέον πληροφορίες που περιέχονται στο αρχείο δείχνουν τις συνδέσεις μεταξύ τους και καλούνται οι κατάλληλες συναρτήσεις σύνδεσης όπως εξηγήθηκαν παραπάνω. Με λίγα λόγια, το άνοιγμα αρχείου δημιουργεί από την αρχή το αποθηκευμένο διάγραμμα.

Στην συνέχεια θα αναφερθούν οι σχετικές συναρτήσεις και θα εξηγηθεί ο ρόλος της κάθε μιας συνάρτησης.

generateFile();

Η συνάρτηση αυτή είναι υπεύθυνη για την δημιουργία της περιγραφής ενός διαγράμματος μέσω XML. Διατρέχει όλες τις λίστες που περιέχουν τα αντικείμενα διαγράμματος οι οποίες είναι: globalEntities[], globalAttributes[], globalRelationships[], globalSpecs_Unions[]. Για κάθε στοιχείο που βρίσκεται σε κάποια λίστα, καταγράφονται τα χαρακτηριστικά του σε μορφή XML και αποθηκεύονται σε συμβολοσειρές. Κάθε αντικείμενο έχει κρυφό μοναδικό ID που φυσικά δεν ταυτίζεται με το όνομά του, και χρησιμοποιείται για την περιγραφή των συνδέσεων. Επίσης, η συνάρτηση generateFile() δημιουργεί και tags που περιέχουν το μέγεθος της περιοχής σχεδίασης (client-width, client-height). Καθώς η περιοχή σχεδίασης προσαρμόζεται στην κάθε οθόνη, η αποθήκευση διαστάσεων χρησιμεύει στο να προσαρμόζεται η θέση των σχημάτων σε περίπτωση που αργότερα το αρχείο ανοιχτεί από σύστημα με μικρότερη οθόνη. Τελικά, όλες οι συμβολοσειρές ενώνονται για να δημιουργηθεί το αρχείο, και καλείται η συνάρτηση saveFile() η οποία αναλαμβάνει την δημιουργία και αποθήκευση του αρχείου στο υπολογιστικό σύστημα του χρήστη.

saveFile();

Η συνάρτηση saveFile() βασίζεται στο εργαλείο FileSaver.js και δημιουργεί ένα αρχείο κειμένου, το οποίο ο χρήστης μπορεί να αποθηκεύσει. Ο τρόπος κλήσης της, όπως χρησιμοποιείται από την generateFile() είναι:

return saveFile (this, 'saved-diagram', txt , 'text/plain;charset=utf-8'); Όπου το πρώτο όρισμα λειτουργεί ως reference, το δεύτερο ορίζει το όνομα του αρχείου (ο χρήστης μπορεί να το αλλάξει), το τρίτο όρισμα είναι μια μεταβλητή που περιέχει μια συμβολοσειρά, και τέλος το τελευταίο όρισμα δηλώνει κωδικοποίηση κειμένου.

openFile();

Η συγκεκριμένη συνάρτηση διαβάζει τα αρχεία κειμένου και αποθηκεύει τα χαρακτηριστικά του κάθε αντικειμένου σε προσωρινές μεταβλητές. Όταν όλα τα στοιχεία ενός αντικειμένου έχουν διαβαστεί, καλείται η κατάλληλη συνάρτηση δημιουργίας, χρησιμοποιώντας τις προσωρινές μεταβλητές ως ορίσματα. Για παράδειγμα, στην περίπτωση του σχήματος 5.5 αφού διαβαστούν τα στοιχεία του γνωρίσματος θα γινόταν κλήση της συνάρτησης createEllipse() ως εξής:

createEllipse("attribute", "author", "CHARACTER", 20, 0, 0, "false", "false", "false", "ent0" , 544, 226);

Με παρόμοιο τρόπο δημιουργούνται και τα αντικείμενα των υπόλοιπων κατηγοριών. Όταν τελειώσει η διαδικασία της δημιουργίας, αρχίζουν οι συνδέσεις. Για παράδειγμα, αφού δημιουργηθεί το γνώρισμα author, θα γίνει εύρεση του τύπου οντότητας με ID 'ent0' και θα κληθεί να συνάρτηση connectAttrToOwner().

screenshot();

Η συνάρτηση screenshot(); είναι υπεύθυνη για την παραγωγή εικόνας τύπου PNG που απεικονίζει το διάγραμμα. Αρχικά επιλέγεται ολόκληρη η περιοχή σχεδίασης μέσω της εντολής

```
var svg = document.getElementsByTagName('svg')[0];
```

και στην συνέχεια, με την βοήθεια του εργαλείου canvg που εξηγήθηκε στο κεφάλαιο 4, τα αντικείμενα SVG μετατρέπονται σε εικόνα PNG. Το canvg δεν λειτουργεί σωστά σε όλους τους περιηγητές διαδικτύου, και για αυτόν το λόγο προτείνεται η χρήση πρόσφατης έκδοσης του Firefox.

5.4 Έλεγχος σφαλμάτων

check ();

Η συνάρτηση check (); εκτελείται με το πάτημα του κουμπιού Error Check στην κεντρική οθόνη. Κάνει έναν απλό έλεγχο καλώντας την συνάρτηση errorCheck (); που θα εξηγηθεί παρακάτω. Εφόσον η συνάρτηση errorCheck (); επιστρέψει την τιμή '1' τότε σημαίνει πως δεν υπάρχουν σφάλματα, και εμφανίζεται το μήνυμα "No errors found." στην οθόνη.

errorCheck ();

Η συνάρτηση αυτή αποτελεί το βασικότερο κομμάτι του ελέγχου σφαλμάτων, καθώς εκτελεί βασικούς ελέγχους και καλεί άλλες βοηθητικές συναρτήσεις. Δεν χρειάζεται κανένα όρισμα επειδή οι λίστες όλων των αντικειμένων είναι global και μπορούν να προσπελαστούν από παντού. Βασική μεταβλητή της συνάρτησης είναι η error_text. Κάθε φορά που εντοπίζεται κάποιο λάθος, στην error_text προστίθεται μια συμβολοσειρά με το αντίστοιχο μήνυμα λάθους. Στο τέλος της συνάρτησης, εάν η error_text είναι κενή συμβολοσειρά τότε σημαίνει ότι κανένα λάθος δεν εντοπίστηκε και η συνάρτηση errorCheck (); επιστρέφει την τιμή '1'. Αλλιώς, επιστρέφεται η τιμή '0' και εμφανίζεται στην οθόνη το σύνολο των μηνυμάτων λάθους (χρησιμοποιώντας την μεταβλητή error_text). Όλα τα δυνατά μηνύματα λάθους αναφέρθηκαν στο κεφάλαιο 3. Στην συνέχεια θα εξεταστούν οι βοηθητικές συναρτήσεις της errorCheck ();.

uniqueCheck();

Η συγκεκριμένη συνάρτηση παίρνει ως όρισμα μία λίστα γνωρισμάτων, και ελέγχει για την ύπαρξη υποψήφιων κλειδιών σε αυτήν τη λίστα. Ο τρόπος που χρησιμοποιείται από την συνάρτηση errorCheck (); είναι ο εξής:

Όλα τα γνωρίσματα που ανήκουν στον ίδιο ιδιοκτήτη τοποθετούνται σε μια λίστα, και καλείται η συνάρτηση uniqueCheck();. Επιστρέφεται η τιμή '1' εφόσον υπάρχουν υποψήφια κλειδιά, αλλιώς επιστρέφεται η τιμή '0'. Εάν επιστραφεί η τιμή '0' σημαίνει ότι πρέπει να υπάρξει μήνυμα λάθους που ειδοποιεί τον χρήστη να ορίσει ένα υποψήφιο κλειδί για το συγκεκριμένο αντικείμενο.

Παράδειγμα κλήσης της uniqueCheck(); αποτελεί το παρακάτω κομμάτι κώδικα:

```
if(uniqueCheck(globalEntities[i].attributes) === 0){

error_text = error_text + '\n' + " • Entity named " + '"' + globalEntities[i].name + '"' + "
has no unique attributes.";   }
```

duplicateUniqueCheck();

Με παρόμοιο τρόπο λειτουργεί και η duplicateUniqueCheck(); η οποία ελέγχει την ύπαρξη πολλαπλών υποψήφιων κλειδιών ή πολλαπλών πρωτευόντων κλειδιών. Παίρνει ως όρισμα την λίστα γνωρισμάτων που ανήκουν σε ένα αντικείμενο και επιστρέφει την τιμή '0' όταν υπάρχουν πολλαπλά υποψήφια κλειδιά αλλά κανένα πρωτεύον, την τιμή '1' εάν υπάρχει ακριβώς ένα υποψήφιο / πρωτεύον κλειδί, και την τιμή '2' εάν υπάρχουν πολλαπλά πρωτεύοντα κλειδιά. Στην περίπτωση που επιστραφεί '0' ή '2' τότε ο χρήστης πρέπει να επιλέξει μοναδικό πρωτεύον κλειδί.

attributeNameCheck();

Η συγκεκριμένη συνάρτηση παίρνει ως όρισμα μια λίστα γνωρισμάτων που ανήκουν στον ίδιο ιδιοκτήτη, και ελέγχει την ύπαρξη πολλαπλών γνωρισμάτων με το ίδιο όνομα. Αρχικά η λίστα ταξινομείται ανάλογα με το όνομα των γνωρισμάτων, και εάν δυο διαδοχικά στοιχεία έχουν το ίδιο όνομα τότε επιστέφεται η τιμή '1', αλλιώς επιστρέφεται η τιμή '0'.

5.5 Μετατροπή σε PostgreSQL

5.5.1 Διαδικασία μετατροπής

Όπως αναφέρθηκε στο κεφάλαιο 2, υπάρχουν ομοιότητες ανάμεσα στο μοντέλο ER και στο σχεσιακό μοντέλο. Σύμφωνα με τους R. Elmasri, S. B. Navathe [1], υπάρχει τρόπος ώστε ένα διάγραμμα ER να μετατραπεί σε απεικόνιση στο σχεσιακό μοντέλο, ακολουθώντας την παρακάτω διαδικασία.

Απεικόνιση τύπων οντοτήτων

Ένας ισχυρός τύπος οντοτήτων E μπορεί να περιγραφεί με μία σχέση R (ή πίνακα) που περιέχει όλα τα απλά γνωρίσματά του ως στήλες στον πίνακα, δηλαδή εάν υπάρχει κάποιο σύνθετο γνώρισμα, πρέπει να συμπεριληφθούν μόνο τα απλά γνωρίσματα από τα οποία αποτελείται. Πρέπει κάποιο από τα υποψήφια (unique) κλειδιά του E να επιλεχτεί ως πρωτεύον κλειδί. Εάν επιλεχτεί κάποιο σύνθετο γνώρισμα, τότε τα απλά γνωρίσματα του θα αποτελούν μαζί το πρωτεύον κλειδί. Τα υπόλοιπα υποψήφια κλειδιά θα πρέπει να χαρακτηριστούν ως unique στο σχεσιακό σχήμα.

Για κάποιον ασθενή τύπο οντοτήτων W, δημιουργείται μια σχέση R που περιέχει όλα τα απλά γνωρίσματα του W ως στήλες στον πίνακα. Καθώς οι ασθενείς τύποι οντοτήτων ορίζονται με την βοήθεια ισχυρών τύπων οντοτήτων που αποτελούν ιδιοκτήτες, τα πρωτεύοντα κλειδιά των ιδιοκτητών εισάγονται στην σχέση R ως γνωρίσματα - ξένα κλειδιά. Το πρωτεύον κλειδί της R θα είναι ο συνδυασμός των ξένων κλειδιών και κάποιου μερικού κλειδιού του W.

Εφόσον εισάγεται ξένο κλειδί που αναφέρεται σε άλλον πίνακα E, θα πρέπει να ορίσουμε πως θα αντιμετωπίσει το σύστημα τυχόν αλλαγές στην πλειάδα του E όπου υπάρχει η αναφορά. Προσθέτουμε λοιπόν μια referential triggered action [1], στον περιορισμό ξένου κλειδιού. Οι επιλογές που υπάρχουν στην SQL είναι CASCADE, SET NULL, SET DEFAULT και αναφέρονται στις πράξεις ON DELETE, ON UPDATE.

Επιλέγεται ON UPDATE CASCADE και ON DELETE CASCADE επειδή η ύπαρξη του ασθενούς τύπου εξαρτάται από την ύπαρξη των ιδιοκτητών.

Απεικόνιση Πλειότιμων Γνωρισμάτων

Για κάθε πλειότιμο γνώρισμα A, δημιουργείται μια σχέση R που το αντιπροσωπεύει. Αυτή η σχέση θα περιέχει κάποιο γνώρισμα που αντιστοιχεί στο A, όπως επίσης και απλά γνωρίσματα που μπορεί να υπάρχουν. Περιέχει επίσης, ως ξένο κλειδί, το πρωτεύον κλειδί του ιδιοκτήτη του A. Το πρωτεύον κλειδί της R θα είναι ο συνδυασμός του A και του ξένου κλειδιού. Επίσης όσον αφορά το ξένο κλειδί επιλέγεται ON UPDATE CASCADE και ON DELETE CASCADE.

Απεικόνιση 1:1 Δυαδικών Συσχετίσεων

Υπάρχουν τρεις τρόποι για την συγκεκριμένη απεικόνιση, όμως θα εξηγηθεί μόνο ο τρόπος που χρησιμοποιείται στην εφαρμογή SQL Draw. Εάν θεωρήσουμε ότι οι δυο πίνακες που συμμετέχουν στην δυαδική συσχέτιση είναι οι S και T, διαλέγουμε έναν από τους δυο (προτιμότερο η πλευρά με πλήρη συμμετοχή), έστω S. Στην σχέση S εισάγεται το πρωτεύον κλειδί της T ως ξένο κλειδί, καθώς και όλα τα απλά γνωρίσματα της συσχέτισης.

Απεικόνιση 1:N Δυαδικών Συσχετίσεων

Η απεικόνιση αυτής της περίπτωσης είναι ίδια με παραπάνω, απλά αυτήν τη φορά επιλέγουμε την σχέση S που βρίσκεται στην πλευρά N λόγου πληθικότητας.

Για την απεικόνιση των δυαδικών συσχετίσεων 1:1, 1:N επιλέγουμε ON DELETE SET NULL, ON UPDATE CASCADE.

Απεικόνιση M:N Δυαδικών Συσχετίσεων

Για έναν M:N τύπο συσχέτισης, δημιουργείται ξεχωριστή σχέση (πίνακας) R. Τα πρωτεύοντα κλειδιά των δυο συμμετεχόντων σχέσεων (S , T) εισάγονται ως ξένα κλειδιά στην R, και επίσης εισάγονται όλα τα απλά γνωρίσματα του τύπου συσχέτισης, εάν υπάρχουν. Το πρωτεύον κλειδί της R θα είναι ο συνδυασμός των

ξένων κλειδιών. Επιλέγεται ON UPDATE CASCADE και ON DELETE CASCADE επειδή η ύπαρξη της R εξαρτάται από την ύπαρξη των συμμετεχόντων στην συσχέτιση.

Απεικόνιση Συσχετίσεων Βαθμού > 2

Για έναν τύπο συσχέτισης βαθμού μεγαλύτερου του 2, δημιουργείται ξεχωριστή σχέση (πίνακας) R. Τα πρωτεύοντα κλειδιά των συμμετεχόντων σχέσεων εισάγονται ως ξένα κλειδιά στην R, και επίσης εισάγονται όλα τα απλά γνωρίσματα του τύπου συσχέτισης, εάν υπάρχουν. Το πρωτεύον κλειδί της R θα είναι ο συνδυασμός των ξένων κλειδιών, εξαιρούνται όμως τα ξένα κλειδιά που προέρχονται από πίνακα που αντιστοιχεί σε λόγο πληθικότητας 1. Επιλέγεται ON UPDATE CASCADE και ON DELETE CASCADE επειδή η ύπαρξη της R εξαρτάται από την ύπαρξη των συμμετεχόντων στην συσχέτιση.

Απεικόνιση Εξειδικεύσεων

Σε αυτήν την κατηγορία απεικόνισης υπάρχουν αρκετοί τρόποι για να φτάσουμε σε ένα τελικό αποτέλεσμα, όμως θα αναλυθεί ο τρόπος που χρησιμοποιήθηκε στην εφαρμογή SQL Draw. Ο συγκεκριμένος τρόπος δουλεύει τόσο για disjoint όσο και για overlapping εξειδικεύσεις, και επεκτείνεται και στην περίπτωση ύπαρξης απλής υπερκλάσης (χωρίς εξειδίκευση).

Ας θεωρήσουμε ότι η εξειδίκευση έχει m υποκλάσεις $\{S_1, S_2, ..., S_m\}$ και μια υπερκλάση C με γνωρίσματα $\{k, a_1,...a_n\}$ όπου το k είναι το πρωτεύον κλειδί της C. Δημιουργείται μια σχέση L για την C όπως ακριβώς είδαμε παραπάνω. Δημιουργείται επίσης μια σχέση L_i για κάθε υποκλάση S_i, $1<i<m$. Σε κάθε L_i, περιέχονται τα απλά γνωρίσματα της κάθε S_i και το k. Το πρωτεύον κλειδί σε κάθε L_i αποτελείται από το k.

Απεικόνιση Ενώσεων

Σε μια ένωση, τα κλειδιά των υπερκλάσεων είναι διαφορετικά και δεν γίνεται να χρησιμοποιηθεί μόνο ένα από αυτά. Έτσι, συνήθως χρησιμοποιείται για την απεικόνιση ένα νέο κλειδί, που ονομάζεται αναπληρωματικό (surrogate). Το αναπληρωματικό κλειδί αποτελεί πρωτεύον κλειδί στην σχέση L που αντιστοιχεί στην υποκλάση. Επίσης, στις υπερκλάσεις εισάγονται ξένα κλειδιά παραγόμενα από το

αναπληρωματικό κλειδί. Για τα ξένα κλειδιά σε αυτήν την περίπτωση χρησιμοποιούνται οι επιλογές ON DELETE SET NULL, ON UPDATE CASCADE.

5.5.2 Συναρτήσεις μετατροπής

convertToSQL();

Η convertToSQL(); είναι η βασική συνάρτηση μετατροπής σε PostgreSQL. Αρχικά καλείται η συνάρτηση check(); για να γίνει έλεγχος σφαλμάτων, και εφόσον δεν υπάρχουν ξεκινάει η διαδικασία μετατροπής. Αρχικά ελέγχονται οι συσχετίσεις στις οποίες συμμετέχει η κάθε συνάρτηση, και ανάλογα με το είδος των συσχετίσεων αποφασίζεται εάν χρειάζεται η δημιουργία ξεχωριστού πίνακα για την συσχέτιση, καλώντας την συνάρτηση constructRelTable(); που θα δούμε παρακάτω. Στην συνέχεια, δημιουργείται ο πίνακας για τον κάθε τύπο οντότητας, λαμβάνοντας υπόψη όλα όσα αναφέρθηκαν στην ενότητα 5.5.1, ενώ ιδιαίτερη σημασία δίνεται και στα πλειότιμα γνωρίσματα. Η συνάρτηση convertToSQL(); χρησιμοποιεί συχνά μια βοηθητική συνάρτηση με όνομα findPrimary(); η οποία θα αναφερθεί επίσης παρακάτω.

constructRelTable();

Η παρούσα συνάρτηση καλείται για να δημιουργήσει τον πίνακα που αντιστοιχεί σε M:N τύπο δυαδικών συσχετίσεων ή σε τύπο συσχετίσεων βαθμού μεγαλύτερου του 2. Παίρνει ως όρισμα ένα αντικείμενο συσχέτισης, και επιστρέφει μια συμβολοσειρά που αντιστοιχεί στον συγκεκριμένο πίνακα. Ένα παράδειγμα κλήσης της είναι :

```
var txt5 = constructRelTable(globalRelationships[i]);
```

findPrimary();

Η συνάρτηση findPrimary(); χρησιμοποιείται συχνά στις δυο συναρτήσεις που αναφέρθηκαν, και χρησιμεύει κυρίως στην εύρεση του πρωτεύοντος κλειδιού ενός τύπου οντοτήτων. Παίρνει δυο ορίσματα: έναν τύπο οντοτήτων και μια μεταβλητή boolean. Η μεταβλητή boolean χρησιμοποιείται ως εξής: όταν η τιμή της είναι false σημαίνει ότι δεν θέλουμε να επηρεαστεί το αποτέλεσμα από την ύπαρξη υπερκλάσης, ενώ το αντίθετο ισχύει όταν η τιμή της είναι true. Επιστρέφει ως αποτέλεσμα τρία στοιχεία primary_key_name, unique_names, dataT. Το πρώτο περιέχει το όνομα του πρωτεύοντος κλειδιού του τύπου οντοτήτων, το δεύτερο είναι μια λίστα με τα unique γνωρίσματα, ενώ το τρίτο περιέχει τον τύπο δεδομένων πρωτεύοντος κλειδιού.

6. Συμπεράσματα και μελλοντικά σχέδια

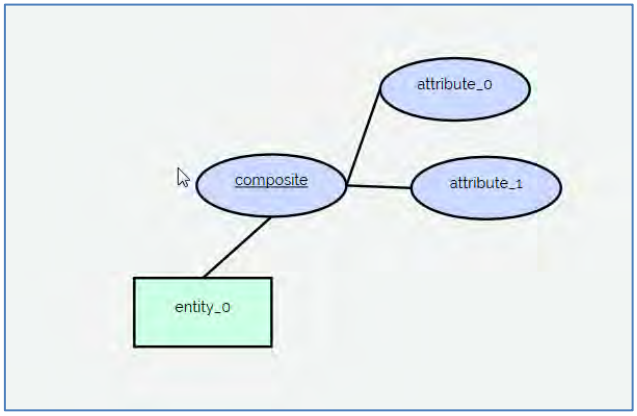
Στην παρούσα διπλωματική εργασία, ασχοληθήκαμε με την ανάλυση της ανάγκης για οργάνωση και διαχείριση μεγάλου όγκου πληροφοριών, καθώς και την εισαγωγή των βάσεων δεδομένων και των εργαλείων για την διαχείρισή τους. Αφού εξετάστηκε το μοντέλο Οντοτήτων - Συσχετίσεων και η χρησιμότητά του στον σχεδιασμό μιας βάσης δεδομένων, αναπτύχθηκε το διαδικτυακό εργαλείο SQL Draw, δηλαδή μια εφαρμογή που λειτουργεί online και χρησιμεύει στην δημιουργία διαγραμμάτων οντοτήτων - συσχετίσεων. Επιπλέον, στο SQL Draw προστέθηκε η δυνατότητα αυτόματης μετατροπής ενός διαγράμματος σε κώδικα PostgreSQL.

Τα συμπεράσματα που προέκυψαν από την ανάπτυξη της εφαρμογής SQL Draw, από την άποψη σχεδιασμού και υλοποίησης μιας εφαρμογής, είναι μεγάλης σημασίας για την προσωπική μου ανάπτυξη στον συγκεκριμένο τομέα. Εκτός από τις γνώσεις που απέκτησα όσον αφορά τις βάσεις δεδομένων, εξοικειώθηκα με την γλώσσα JavaScript και την χρήση όλων των επιπλέον εργαλείων. Απέκτησα εμπειρία στην πολύπλευρη εξέταση του τρόπου λειτουργίας ενός αλγορίθμου και στον εντοπισμό σφαλμάτων που υπάρχουν στον κώδικα.

Στα μελλοντικά σχέδια της εφαρμογής περιλαμβάνονται κυρίως βελτιώσεις ως προς την λειτουργικότητα και την ευκολία χρήσης. Μια αλλαγή που σχετίζεται με την ευκολία χρήσης είναι η υλοποίηση συστήματος που αποθηκεύει τους χρήστες και τα προσωρινά τους διαγράμματα σε server, κάτι που διευκολύνει αρκετά την δημιουργία διαγραμμάτων.

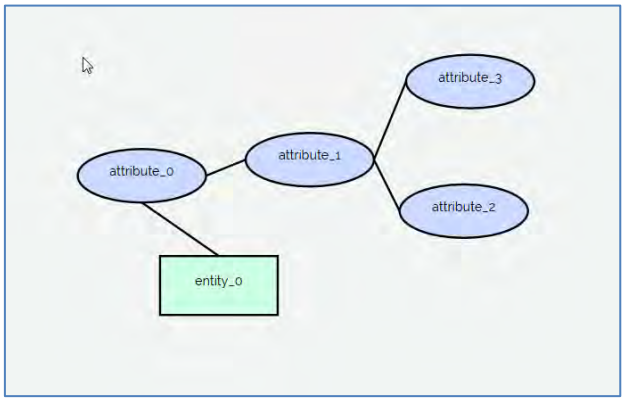
Για την βελτίωση της λειτουργικότητας κύριο μέλημα είναι η αναδιοργάνωση του κώδικα ώστε να γίνει πιο αποτελεσματική και γρήγορη η διαδικασία μετατροπής.

Επίσης, υπάρχουν δυο χαρακτηριστικές περιπτώσεις συνδεσμολογιών τις οποίες η εφαρμογή SQL Draw αυτήν τη στιγμή δεν μπορεί να υποστηρίξει στο στάδιο της μετατροπής σε PostgreSQL.



Σχήμα 6.1: Σύνθετο γνώρισμα ως πρωτεύον κλειδί

Η πρώτη περίπτωση απεικονίζεται στο σχήμα 6.1 και αποτελείται από ένα σύνθετο γνώρισμα που είναι unique (και μοναδικό unique γνώρισμα σε έναν τύπο οντοτήτων) ή primary key. Στην περίπτωση αυτή πρέπει ο τύπος οντότητας στον οποίο ανήκει, να διαθέτει δυο ξεχωριστά πρωτεύοντα κλειδιά (τα γνωρίσματα που αποτελούν το σύνθετο γνώρισμα). Όμως η εφαρμογή SQL Draw υποστηρίζει μοναδικό πρωτεύον κλειδί για κάθε τύπο οντότητας, εκτός εάν πρόκειται για ασθενείς τύπους.



Σχήμα 6.2: Διαδοχικά σύνθετα γνωρίσματα

Η δεύτερη περίπτωση απεικονίζεται στο σχήμα 6.2, και αποτελείται από σύνθετο γνώρισμα που συνδέεται διαδοχικά με ένα άλλο γνώρισμα, δημιουργώντας σύνθετο

γνώρισμα επιπέδου 2. Αυτή η περίπτωση είναι δύσκολη όσον αφορά στον έλεγχο για την μετατροπή του διαγράμματος, οπότε αυτήν τη στιγμή δεν υποστηρίζεται.

Συνεπώς, στα μελλοντικά σχέδια συμπεριλαμβάνεται η υλοποίηση μετατροπής και για αυτές τις περιπτώσεις.

Βιβλιογραφία

[1] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 6th ed. Boston: Addison-Wesley Longman Publishing Co., Inc., 2011.

[2] R. Ramakrishnan and J. Gehrke, *Συστήματα Διαχείρισης Βάσεων Δεδομένων*, μετάφραση Γεώργιος Σίσιας, 3η εκδ. Αθήνα: Εκδόσεις ΤΖΙΟΛΑ, 2012.

[3] I. Sommerville, *Βασικές Αρχές Τεχνολογίας Λογισμικού*, μετάφραση Δημήτρης Τσιλογιάννης, 8η εκδ. Αθήνα: Εκδόσεις Κλειδάριθμος, 2009.

[4] Μανωλόπουλος Ι. και Παπαδόπουλος Α., *Συστήματα Βάσεων Δεδομένων*, Αθήνα: Εκδόσεις Νέων Τεχνολογιών, 2006.

[5] Ιστοσελίδα της The PostgreSQL Global Development Group με τίτλο PostgreSQL: About, 2017.

https://www.postgresql.org/about/ (Ανακτήθηκε 2017-8-26).

[6] Ιστοσελίδα της Lucid Software Inc. με τίτλο Lucidchart, 2017.

https://www.lucidchart.com/ (Ανακτήθηκε 2017-8-31).

[7] Ιστοσελίδα της SmartDraw, LLC με τίτλο SmartDraw, 2017.

https://www.smartdraw.com/ (Ανακτήθηκε 2017-8-31).

[8] Ιστοσελίδα της JGraph Ltd. με τίτλο draw.io, 2017.

https://www.draw.io/ (Ανακτήθηκε 2017-8-31).

[9] Ιστοσελίδα του TerraER με τίτλο TerraER, 2017.

http://www.terraer.com.br/ (Ανακτήθηκε 2017-8-31).

[10] Βουλτσίδης Δ. Μιχαήλ, Ανάπτυξη εργαλείου μετατροπής διαγραμμάτων οντοτήτων - συσχετίσεων σε σχεσιακά και αντικειμενοσχεσιακά σχήματα, Ελληνικό Ανοιχτό Πανεπιστήμιο, 2015.

[11] Ιστοσελίδα του ERDPlus με τίτλο ERDPlus, 2017.

https://erdplus.com/#/ (Ανακτήθηκε 2017-8-31).

[12] J. Duckett, *HTML & CSS Design and Build Websites*, Indianapolis: John Wiley & Sons Inc., 2011.

[13] S. Suehring, *JavaScript Step by Step,* 3rd ed. Microsoft, USA.

[14] Ιστοσελίδα του Bootstrap με τίτλο Bootstrap, 2017.

http://getbootstrap.com/ (Ανακτήθηκε 2017-8-31).

[15] Ιστοσελίδα της The JQuery Foundation με τίτλο JQuery, 2017.

https://jquery.com/ (Ανακτήθηκε 2017-8-31).

[16] Ιστοσελίδα του SweetAlert με τίτλο SweetAlert, 2017.

http://lab.mkblog.cn/sweetalert/ (Ανακτήθηκε 2017-8-31).

[17] Ιστοσελίδα του Snap.svg με τίτλο Snap.svg, 2017.

http://snapsvg.io/ (Ανακτήθηκε 2017-8-31).

[18] Ιστοσελίδα του www.ignitersworld.com με τίτλο ContextMenu.js, 2015.

http://ignitersworld.com/lab/contextMenu.html (Ανακτήθηκε 2017-8-31).

[19] Ιστοσελίδα του GitHub, Inc. με τίτλο GitHub - canvg, 2017.

https://github.com/canvg/canvg (Ανακτήθηκε 2017-8-31).

[20] Ιστοσελίδα του GitHub, Inc. με τίτλο GitHub - eligrey/FileSaver.js, 2017.

httpshttps://github.com/eligrey/FileSaver.js/ (Ανακτήθηκε 2017-8-31).