Ελληνικό Ανοιχτό Πανεπιστήμιο

Σχολή Θετικών Επιστημών και Τεχνολογίας

Μεταπτυχιακή Εξειδίκευση στα Πληροφοριακά Συστήματα

Διπλωματική Εργασία

Ανάπτυξη Ολοκληρωμένου Εργαλείου Μοντελοποίησης Βάσεων Δεδομένων σε Java

Development of an Integrated Database Modelling Tool in Java

Νικολέτα Λαβδαριά

Επιβλέπων καθηγητής: Μιχαήλ Βασιλακόπουλος

Πάτρα, 2023

Ανάπτυξη Ολοκληρωμένου Εργαλείου Μοντελοποίησης Βάσεων Δεδομένων σε Java

Development of an Integrated Database Modelling Tool in Java

Νικολέτα Λαβδαριά

Επιτροπή Επίβλεψης Διπλωματικής Εργασίας

Επιβλέπων Καθηγητής:	Συν-Επιβλέπουσα Καθηγήτρια:
Μιχαήλ Βασιλακόπουλος	Γεωργία Γκαράνη
Καθηγητής	Αναπληρώτρια Καθηγήτρια

Πάτρα, 2023

...στους γονείς μου και τη Ρούλα και το Χρήστο για την υποστήριξή τους

Περίληψη

Το μοντέλο Οντοτήτων-Συσχετίσεων (Entity-Relationship ή ER model) χρησιμοποιείται για να απεικονίσει τις πληροφορίες που θα αποθηκευτούν σε μια βάση δεδομένων, περιγράφοντας τις οντότητες (entities) και τις συσχετίσεις (relationships) μεταξύ τους. Με αυτό τον τρόπο, μπορεί να βοηθήσει μια επιχείρηση να κατανοήσει καλύτερα τις απαιτήσεις των χρηστών της όσον αφορά τη βάση δεδομένων και να διευκολύνει την επικοινωνία μεταξύ των σχεδιαστών και των χρηστών για την επιτυχή ανάπτυξη του συστήματος. Το αποτέλεσμα της σχεδίασης με το μοντέλο ER είναι το διάγραμμα οντοτήτων-συσχετίσεων (ER diagram), το οποίο περιγράφει τις οντότητες και τις συσχετίσεις με γραφικά σχήματα αλλά και τις μεταξύ τους συνδέσεις. Το Εκτεταμένο Μοντέλο Οντοτήτων-Συσχετίσεων(Enhanced Entity-Relationship ή EER model) επεκτείνει το προηγούμενο μοντέλο προσθέτοντας σχέσεις κληρονομικότητας μεταξύ των οντοτήτων όπως η ένωση, η γενίκευση και η εξειδίκευση. Η Λογική Σχεδίαση των Βάσεων Δεδομένων προσπαθεί να μεταφράσει την υψηλού επιπέδου περιγραφή των δεδομένων σε ένα σχήμα που μπορεί να επεξεργαστεί ο υπολογιστής, χρησιμοποιώντας το Σχεσιακό Μοντέλο (Relational model). Το αποτέλεσμα της διαδικασίας αυτής είναι ένα Σχεσιακό Σχήμα (Relational Schema), το οποίο περιγράφει τις σχέσεις μεταξύ των δεδομένων σε μια μορφή που μπορεί να αντιληφθεί ο υπολογιστής και που περιλαμβάνει περιορισμούς ακεραιότητας για τις σχέσεις. Αυτή η διαδικασία ονομάζεται Απεικόνιση του Μοντέλου ER/EER στο Σχεσιακό Μοντέλο.

Στόχος της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη ενός εργαλείου, χρησιμοποιώντας τη γλώσσα προγραμματισμού Java, για το σχεδιασμό Διαγραμμάτων Οντοτήτων Συσχετίσεων μιας Βάσης Δεδομένων. Η εφαρμογή είναι διαθέσιμη σε όλα τα υπολογιστικά συστήματα που υποστηρίζουν το περιβάλλον εκτέλεσης της Java (Java Virtual Machine) και χρησιμοποιεί τη βιβλιοθήκη jGraphX. Το εργαλείο, πέρα από το σχεδιασμό του Εκτεταμένου Διαγράμματος Οντοτήτων-Συσχετίσεων, ελέγχει τη συντακτική ορθότητα του διαγράμματος, εντοπίζοντας πιθανά λάθη που έχουν προκύψει κατά τη σχεδίαση. Στη συνέχεια, εφόσον διορθωθούν τα σφάλματα, δύναται να μετατρέψει το διάγραμμα σε Σχεσιακό Σχήμα. Τέλος, παρέχει τη δυνατότητα δημιουργίας των SQL εντολών που απαιτούνται για τη δημιουργία του αντίστοιχου Σχεσιακού Συστήματος Βάσεων Δεδομένων. Με αυτόν τον

τρόπο, η διαδικασία δημιουργίας του λογικού μοντέλου της βάσης δεδομένων αυτοματοποιείται και γίνεται πιο αποτελεσματική.

Λέξεις – Κλειδιά

Μοντελοποίηση Βάσεων Δεδομένων, Μοντέλο Οντοτήτων-Συσχετίσεων, Εκτεταμένο Μοντέλο Οντοτήτων-Συσχετίσεων, Σχεσιακό Σχήμα, Σχεσιακό Σύστημα Βάσεων Δεδομένων, SQL

Development of an Integrated Database Modelling Tool in Java

Nikoleta Lavdaria

Abstract

The Entity-Relationship or ER model is used to visualize the information to be stored in a database, by describing the entities and the relationships between them. In this way, it can help a business better understand its users' database requirements and facilitate communication between designers and users for successful system development. The result of drawing with the ER model is the entity-relationship diagram (ER diagram), which describes the entities and relationships with graphical shapes and the connections between them. The Enhanced Entity-Relationship (EER) model extends the previous model by adding inheritance relationships between entities such as union, generalization, and specialization. Logical Database Design attempts to translate the high-level description of the data into a schema that can be processed by the computer, using the Relational Model. The result of this process is a Relational Schema, which describes the relationships between data in a form that can be understood by the computer and that includes integrity constraints on the relationships. This process is called Mapping the ER/EER Model to the Relational Model.

This thesis aims to develop a tool, using the Java programming language, for the design of Entity Relationship Diagrams of a Database. The application is available in all computing systems that support the Java runtime environment (Java Virtual Machine). The tool, in addition to the design of the Enhanced Entity-Relationship Diagram, checks the syntactic correctness of the diagram, identifying possible errors that have occurred during the design. Then, once the errors are fixed, it can convert the diagram into a Relational Diagram. Finally, it provides the ability to create the required SQL commands for the corresponding Relational Database System. In this way, the process of creating the logical model of the database is automated and made more efficient.

Keywords

Database Modelling, Entity-Relationship Model, Enhanced Entity-Relationship Model, Relational Schema, Relational Database System, SQL

Περιεχόμενα

Κατάλογος Εικόνων / Σχημάτων

1. Εισαγωγή

1.1 Αντικείμενο

Η ανάπτυξη τεχνολογικών εφαρμογών και υπηρεσιών στη σύγχρονη κοινωνία έχει δώσει τη δυνατότητα στους ανθρώπους να επικοινωνούν, να αλληλοεπιδρούν, να αποθηκεύουν και να διαχειρίζονται πληροφορίες ανεξάρτητα από την απόσταση και το χρόνο. Η ψηφιοποίηση της κοινωνίας έχει επίσης επιτρέψει την ανάπτυξη νέων τομέων, όπως το ηλεκτρονικό εμπόριο, οι υπηρεσίες cloud, η τηλεργασία, η ψηφιακή ψυχαγωγία και η κοινωνική δικτύωση. Όλα αυτά έχουν δημιουργήσει νέες ευκαιρίες, προκλήσεις και αλλαγές στην κοινωνική ζωή.

Η συλλογή, η οργάνωση και η διαχείριση των πληροφοριών έχουν γίνει αναγκαίες διαδικασίες στην ψηφιακή εποχή και οι βάσεις δεδομένων έχουν καθιερωθεί ως ένα απαραίτητο εργαλείο για τη διαχείριση και την αποθήκευση μεγάλου όγκου πληροφοριών. Από τη διαχείριση των προσωπικών δεδομένων στις εφαρμογές κοινωνικής δικτύωσης, τη διαχείριση των δεδομένων των πελατών στις επιχειρήσεις, την αποθήκευση των δεδομένων στις υπηρεσίες cloud και τη διαχείριση των δεδομένων στην έρευνα και την επιστήμη, οι βάσεις δεδομένων έχουν καθιερωθεί ως σημαντικό εργαλείο για την αποθήκευση, οργάνωση και ανάκτηση δεδομένων. Μια βάση δεδομένων είναι ένα σύστημα που αποτελείται από μια συλλογή δεδομένων που διαχειρίζεται και επιτρέπει την αποτελεσματική αποθήκευση, ανάκτηση, ενημέρωση και χρήση των δεδομένων αυτών. Χρησιμοποιείται για την αποθήκευση και οργάνωση μεγάλου όγκου δεδομένων και έχει εφαρμογές σε πολλούς τομείς, όπως οι επιχειρήσεις, οι ιατρικές εγγραφές, οι τράπεζες και η εκπαίδευση. Χρησιμοποιούνται για τη διατήρηση των πληροφοριών πελατών, τη διαχείριση των αποθεμάτων και την παρακολούθηση της απόδοσης ενός επιχειρηματικού συστήματος.

Η δημιουργία και η διαχείριση βάσεων δεδομένων είναι μια επιστήμη που απαιτεί γνώσεις σε πολλούς τομείς, όπως η μαθηματική λογική, η πληροφορική και η τεχνητή νοημοσύνη. Οι επαγγελματίες που ειδικεύονται σε αυτό το πεδίο, όπως οι βασικοί διαχειριστές και οι αναλυτές δεδομένων, είναι απαραίτητοι για τη διαχείριση και αξιοποίηση των δεδομένων σε μια επιχείρηση.

Η ανάγκη για συλλογή, αποθήκευση και διαχείριση πληροφοριών έχει υπάρξει από την εποχή της κατασκευής των πρώτων υπολογιστών. Μια βάση δεδομένων είναι μια συλλογή δεδομένων που είναι οργανωμένα έτσι ώστε να μπορούν να προσπελαστούν, να ενημερωθούν και να διαχειριστούν. Ένα σύστημα διαχείρισης βάσεων δεδομένων (Database Management System ή DBMS) είναι ένα λογισμικό που χρησιμοποιείται για τη δημιουργία, τη διαχείριση και την αλληλεπίδραση με μια βάση δεδομένων και παρέχει ένα σύνολο λειτουργιών και εργαλείων για την οργάνωση, την αποθήκευση, την ανάκτηση, την ενημέρωση και την ανάλυση των δεδομένων που αποθηκεύονται στη βάση δεδομένων.

1.2 Διάρθρωση Εργασίας

Η παρούσα διπλωματική εργασία αποτελείται από 5 κεφάλαια. Στο πρώτο κεφάλαιο γίνεται μια εισαγωγή στο θέμα της εργασίας. Στο δεύτερο κεφάλαιο παρουσιάζονται οι έννοιες του μοντέλου Οντοτήτων-Συσχετίσεων, του Εκτεταμένου μοντέλου Οντοτήτων-Συσχετίσεων, του Σχεσιακού μοντέλου καθώς και του κώδικα SQL. Το τρίτο κεφάλαιο παρουσιάζει μερικές από τις πιο διαδεδομένες υπάρχουσες εφαρμογές σχετικά με τη δημιουργία και επεξεργασία διαγραμμάτων ΜΟΣ και ΕΟΣ, ενώ στο τέταρτο κεφάλαιο αναλύονται οι απαιτήσεις, η ανάπτυξη της εφαρμογής και ο κώδικάς της. Στο πέμπτο κεφάλαιο περιγράφεται η λειτουργία της εφαρμογής. Τέλος, στο έκτο και τελευταίο κεφάλαιο περιγράφονται συνοπτικά οι μελλοντικές επεκτάσεις της εν λόγω εφαρμογής.

2. Θεωρητικό Υπόβαθρο

Το παρόν κεφάλαιο απαρτίζεται από 6 υποενότητες. Στην αρχή του κεφαλαίου, πραγματοποιείται μια εισαγωγή στην έννοια των βάσεων δεδομένων. Αναλύονται οι βασικές έννοιες, όπως οι τύποι βάσεων δεδομένων, οι δομές δεδομένων και οι τρόποι αποθήκευσης και οργάνωσης των δεδομένων. Στη συνέχεια, περιγράφεται η διαδικασία σχεδιασμού μιας βάσης δεδομένων. Αυτό περιλαμβάνει την αναγνώριση των οντοτήτων και των σχέσεων μεταξύ τους, τον καθορισμό των γνωρισμάτων και των περιορισμών τους, καθώς και την απεικόνιση αυτών των στοιχείων σε ένα μοντέλο δεδομένων. Στην τρίτη υποενότητα, αναλύονται τα μοντέλα οντοτήτων-συσχετίσεων. Αυτά τα μοντέλα χρησιμοποιούνται για την αναπαράσταση των οντοτήτων, των σχέσεων και των γνωρισμάτων σε μια βάση δεδομένων. Επίσης, εξηγούνται οι έννοιες του κλειδιού πρωτεύοντος, του κλειδιού ξένου και των συσχετίσεων. Ακόμη, στην τέταρτη υποενότητα, παρουσιάζονται τα εκτεταμένα μοντέλα οντοτήτων-συσχετίσεων, τα οποία επεκτείνουν τα απλά μοντέλα οντοτήτων-συσχετίσεων για να περιγράψουν πιο σύνθετες σχέσεις. Στην πέμπτη υποενότητα, αναλύεται το σχεσιακό μοντέλο βάσεων δεδομένων, που χρησιμοποιεί σχέσεις για την αναπαράσταση των δεδομένων και τη διαχείριση των συσχετίσεων μεταξύ τους. Εξηγούνται οι έννοιες των πινάκων, των γραμμών, των στηλών και των πρωτευόντων κλειδιών. Τέλος, παρουσιάζεται ο κώδικας SQL (Structured Query Language), που χρησιμοποιείται για τη διαχείριση και την επεξεργασία δεδομένων σε μια σχεσιακή βάση δεδομένων. Αναλύονται οι βασικές δομές και εντολές της γλώσσας SQL, όπως η δημιουργία πινάκων, η εισαγωγή, η επεξεργασία και η ανάκτηση δεδομένων.

2.1 Βάσεις Δεδομένων

Μια βάση δεδομένων είναι μια συλλογή δεδομένων που συσχετίζονται μεταξύ τους και αποθηκεύονται σε ένα σύστημα υπολογιστών. Η βάση δεδομένων περιλαμβάνει πληροφορίες που αφορούν μια ομάδα ατόμων ή μια οργάνωση και μπορεί να αναφέρεται σε διάφορα θέματα, όπως πελάτες, προϊόντα, παραγγελίες (Connolly & Begg 2015).

Η βάση δεδομένων έχει ένα σύστημα οργάνωσης, όπου τα δεδομένα αποθηκεύονται σε πίνακες με ονόματα στη βάση δεδομένων. Κάθε πίνακας αποτελείται από σειρές (records) και στήλες

(fields). Κάθε σειρά αναφέρεται σε μια εγγραφή (record) στον πίνακα και κάθε στήλη αναφέρεται σε ένα συγκεκριμένο χαρακτηριστικό (attribute) της εγγραφής (Ramakrisnan & Gehrke 2003).

Με τον όρο DBMS (Database Management System) αναφερόμαστε λογισμικό που επιτρέπει την αποτελεσματική διαχείριση των δεδομένων σε μια βάση δεδομένων. Ο ρόλος του DBMS είναι να επιτρέπει στους χρήστες να αλληλεπιδρούν με τη βάση δεδομένων, να προσθέτουν, να τροποποιούν ή να διαγράφουν δεδομένα, να αναζητούν πληροφορίες και να εκτελούν ερωτήματα (Silberschatz , Korth & Sudarshan 2019). Επίσης, προσφέρει μηχανισμούς για τη διατήρηση της ακεραιότητας των δεδομένων, τον έλεγχο πρόσβασης και τη διαχείριση του συστήματος βάσεων δεδομένων.

Οι πιο συνηθισμένοι τύποι DBMS είναι οι σχεσιακές βάσεις δεδομένων, όπου τα δεδομένα οργανώνονται σε πίνακες με στήλες και γραμμές. Άλλοι τύποι DBMS περιλαμβάνουν τις ιεραρχικές, τις δικτυωτές και τις αντικειμενοστραφείς βάσεις δεδομένων (Watt 2014).

Η επιλογή του κατάλληλου DBMS εξαρτάται από τις ανάγκες του χρήστη και του περιβάλλοντος όπου θα χρησιμοποιηθεί. Κάθε DBMS έχει τα δικά του πλεονεκτήματα και μειονεκτήματα, και οι παράγοντες όπως η απόδοση, η ασφάλεια, η ανθεκτικότητα και η ευκολία χρήσης πρέπει να ληφθούν υπόψη κατά την επιλογή του (Watt 2014).

2.2 Σχεδίαση Βάσεων Δεδομένων

Η διαδικασία σχεδιασμού βάσεων δεδομένων αποτελεί ένα σημαντικό στάδιο στην ανάπτυξη ενός συστήματος βάσεων δεδομένων. Παρέχει μια δομημένη προσέγγιση για τον ορισμό των απαιτήσεων των δεδομένων, τη σχεδίαση του σχήματος της βάσης δεδομένων και την οργάνωση των δεδομένων για αποτελεσματική αποθήκευση και ανάκτηση πληροφοριών (Connolly & Begg 2015). Κάθε στάδιο απαιτεί προσεκτική ανάλυση, σχεδιασμό και επικύρωση για να διασφαλιστεί η αποτελεσματική και αποδοτική λειτουργία της βάσης δεδομένων. Τα κύρια στάδια της διαδικασίας σχεδιασμού βάσεων δεδομένων, όπως περιγράφονται από την Adrienne Watt, είναι τα εξής:

Ανάλυση Απαιτήσεων: Σε αυτό το στάδιο αναλύονται και καταγράφονται οι απαιτήσεις του συστήματος και οι ανάγκες των χρηστών. Έπειτα από συνεντεύξεις με τους πελάτες για να κατανοηθεί το προτεινόμενο σύστημα και να καταγραφούν οι απαιτήσεις δεδομένων και λειτουργικότητας, συντάσσεται ένα έγγραφο με τη συνοπτική περίληψη όλων των απαιτήσεων (Connolly & Begg 2015). Οι απαιτήσεις αυτές πρέπει να περιγράφουν τα δεδομένα, τα χαρακτηριστικά τους, τους περιορισμούς που ισχύουν και τις σχέσεις μεταξύ αυτών. Ο διαχειριστής δεδομένων έχει σημαντικό ρόλο σε αυτήν τη διαδικασία, καθώς εξετάζει τα επιχειρηματικά, νομικά και ηθικά θέματα που επηρεάζουν τις απαιτήσεις δεδομένων.

Σχεδιασμός εννοιολογικής βάσης δεδομένων: Εδώ οι σχεδιαστές χρησιμοποιούν ένα ενοιολογικό μοντέλο, όπως είναι το μοντέλο οντοτήτων-συσχετίσεων (ER) για να περιγράψουν τη βάση δεδομένων και να δημιουργήσουν το αντίστοιχο γραφικό της σχήμα (Ramakrisnan & Gehrke 2003). Το βήμα αυτό είναι ιδιαίτερα σημαντικό καθώς βοηθά στον κατανοητό σχεδιασμό της δομής της βάσης δεδομένων και περιλαμβάνει την αναγνώριση των οντοτήτων, των σχέσεων μεταξύ τους και τον καθορισμό των γνωρισμάτων για κάθε οντότητα.

Σχεδιασμός λογικής βάσης δεδομένων: Κατά το τρίτο βήμα, το εννοιολογικό μοντέλο μετατρέπεται σε ένα λογικό μοντέλο που υλοποιείται με τη χρήση ενός συγκεκριμένου συστήματος διαχείρισης βάσεων δεδομένων (DBMS). Σε αυτό το στάδιο, οι σχεδιαστές δημιουργούν την αναπαράσταση της βάσης δεδομένων χρησιμοποιώντας σχέσεις (πίνακες), όπου αποθηκεύονται τα δεδομένα με συγκεκριμένο τρόπο (Watt 2014). Είναι φανερό ότι ο λογικός σχεδιασμός καθορίζει τον τρόπο αποθήκευσης και χειρισμού των δεδομένων στη βάση δεδομένων. Κατά τη διάρκεια αυτής της διαδικασίας, γίνονται επιλογές σχετικά με τους πίνακες που είναι πιο κατάλληλοι για την αναπαράσταση των δεδομένων στη βάση (Connolly & Begg 2015). Ο σχεδιασμός αυτός πρέπει να λαμβάνει υπόψη διάφορα κριτήρια σχεδίασης, όπως η ευελιξία για αλλαγές, ο έλεγχος της αναπαράστασης των δεδομένων και οι περιορισμοί που απαιτούνται. Στην περίπτωση του μοντέλου οντοτήτων-συσχετίσεων, το διάγραμμα θα μετατραπεί σε ένα σχήμα σχεσιακής βάσης δεδομένων.

Κανονικοποίηση: Στο επόμενο στάδιο, το λογικό σχήμα της βάσης δεδομένων αξιολογείται για ενδεχόμενη κανονικοποίησή του (Connolly & Begg 2015). Αυτό σημαίνει ότι εξετάζονται οι σχέσεις και τα γνωρίσματά τους για την εξάλειψη των πολλαπλών εξαρτήσεων και τη βελτιστοποίηση της δομής της βάσης δεδομένων.

Υλοποίηση της βάσης δεδομένων: Η υλοποίηση αφορά τη δημιουργία μιας βάσης δεδομένων βάσει της προδιαγραφής ενός λογικού σχήματος και επηρεάζεται σημαντικά από την επιλογή διαθέσιμων συστημάτων διαχείρισης βάσεων δεδομένων (DBMS), εργαλείων βάσεων δεδομένων και περιβάλλοντα λειτουργίας (Watt 2014). Υπάρχουν επιπλέον εργασίες πέρα από τη δημιουργία ενός σχήματος βάσης δεδομένων και την υλοποίηση των περιορισμών - τα δεδομένα πρέπει να εισαχθούν στους πίνακες, πρέπει να αντιμετωπιστούν θέματα που σχετίζονται με τους χρήστες και τις διεργασίες τους, και πρέπει να υποστηριχθούν διοικητικές δραστηριότητες που σχετίζονται με ευρύτερες πτυχές της διαχείρισης επιχειρησιακών δεδομένων (Silberschatz , Korth & Sudarshan 2019). Είναι επιθυμητό όσο το δυνατόν περισσότερα από αυτά τα θέματα να αντιμετωπίζονται εντός του DBMS. Η υλοποίηση του λογικού σχήματος σε ένα συγκεκριμένο DBMS απαιτεί λεπτομερείς γνώσεις για τις συγκεκριμένες δυνατότητες και λειτουργίες που αυτό προσφέρει.

Η υλοποίηση με ένα συστημάτων διαχείρισης βάσεων δεδομένων (DBMS) που βασίζεται σε σχέσεις περιλαμβάνει τη χρήση της γλώσσας SQL για τη δημιουργία πινάκων και περιορισμών (Silberschatz , Korth & Sudarshan 2019), που ικανοποιούν την περιγραφή του λογικού σχήματος και την επιλογή κατάλληλου σχήματος αποθήκευσης (εάν το DBMS επιτρέπει αυτό το επίπεδο ελέγχου).

Ένας τρόπος για να επιτευχθεί αυτό είναι να γραφούν οι κατάλληλες εντολές SQL σε ένα αρχείο που μπορεί να εκτελεστεί από ένα DBMS, έτσι ώστε να υπάρχει ένα ανεξάρτητο αρχείο κειμένου με τις εντολές SQL που καθορίζουν τη βάση δεδομένων (Watt 2014). Μια άλλη μέθοδος είναι χρησιμοποιώντας ένα διαδραστικό εργαλείο βάσεων δεδομένων όπως το SQL Server Management Studio ή το Microsoft Access. Οποιοσδήποτε μηχανισμός χρησιμοποιείται για την υλοποίηση του λογικού σχήματος έχει ως αποτέλεσμα την καθορισμένη βάση δεδομένων, με πίνακες και περιορισμούς, αλλά δεν περιέχει δεδομένα για τις διεργασίες χρήστη (Ramakrisnan & Gehrke 2003).

Στο στάδιο της πλήρωσης, προσθέτουμε δεδομένα στη βάση μας. Αυτό μπορεί να γίνει με διάφορους τρόπους, όπως μέσω ενός GUI εργαλείου διαχείρισης βάσεων δεδομένων, με εντολές SQL, ή μέσω κώδικα που έχουμε γράψει σε μια εφαρμογή (Watt 2014). Πρέπει να διασφαλίσουμε ότι τα δεδομένα που προσθέτουμε είναι συνεπή με το σχήμα της βάσης

δεδομένων και ότι δεν παραβιάζουν τους περιορισμούς που έχουν οριστεί (Connolly & Begg 2015).

Μετά την πλήρωση της βάσης με δεδομένα, μπορούμε να πραγματοποιήσουμε αναζητήσεις και επεξεργασία των δεδομένων (Silberschatz , Korth & Sudarshan 2019). Αυτό μπορεί να γίνει μέσω του ίδιου του GUI εργαλείου διαχείρισης βάσεων δεδομένων, μέσω εντολών SQL, ή μέσω κώδικα που έχουμε γράψει σε μια εφαρμογή (Watt 2014).

Τέλος, μετά από μια περίοδο χρήσης της βάσης, μπορεί να απαιτηθεί η διαχείριση των δεδομένων (Silberschatz , Korth & Sudarshan 2019). Αυτό μπορεί να περιλαμβάνει τη διαγραφή ή την τροποποίηση των δεδομένων, τη διαγραφή ή την τροποποίηση του σχήματος της βάσης δεδομένων, ή ακόμα και τη μεταφορά των δεδομένων σε άλλο σύστημα. Επιπλέον, ένα DBMS προσφέρει συνήθως εργαλεία διαχείρισης δεδομένων, όπως παρακολούθηση απόδοσης, επαναφορά δεδομένων και αντίγραφα ασφαλείας (Connolly & Begg 2015).

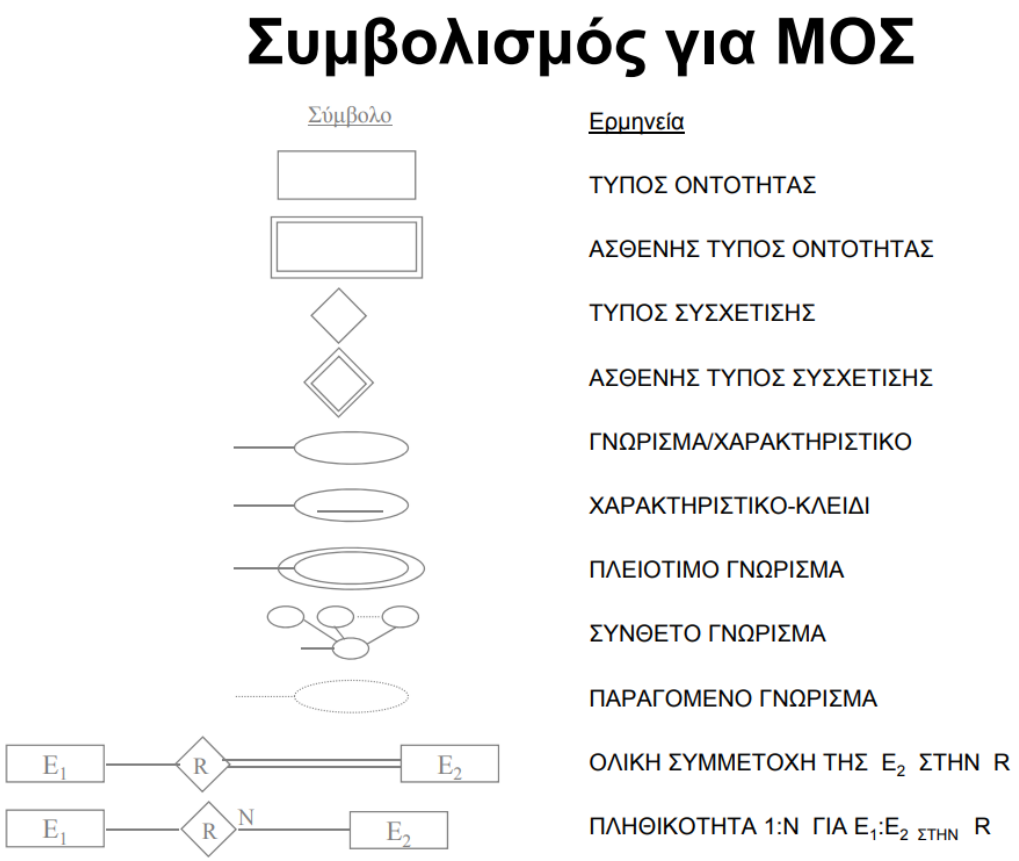
.

2.3 Μοντέλο Οντοτήτων Συσχετίσεων

Το μοντέλο οντοτήτων συσχετίσεων (ΜΟΣ) είναι ένας τρόπος περιγραφής και οργάνωσης δεδομένων σε μια βάση δεδομένων (Connolly & Begg 2015). Στο μοντέλο αυτό, οι οντότητες αναπαριστούν αντικείμενα ή έννοιες που θέλουμε να αποθηκεύσουμε και να διαχειριστούμε στη βάση δεδομένων, ενώ τα γνωρίσματα είναι οι ιδιότητες αυτών των οντοτήτων (Elmasri, & Navathe 2015). Για παράδειγμα, μια οντότητα "Πελάτης" μπορεί να έχει τα γνωρίσματα "Όνομα", "Επώνυμο", "Διεύθυνση". Στο μοντέλο αυτό, οι συσχετίσεις αναπαριστούν τις σχέσεις μεταξύ διαφορετικών οντοτήτων. Υπάρχουν διάφοροι τύποι συσχετίσεων, όπως ένα προς ένα, ένα προς πολλά, και πολλά προς πολλά, ανάλογα με τον τρόπο με τον οποίο οι οντότητες σχετίζονται μεταξύ τους (Silberschatz , Korth & Sudarshan 2019).

Η χρήση του μοντέλου οντοτήτων συσχετίσεων επιτρέπει στους σχεδιαστές βάσεων δεδομένων να αναπαραστήσουν τις οντότητες (π.χ. πελάτες, τοποθεσίες, προϊόντα) και τις συσχετίσεις μεταξύ τους (π.χ. ανήκει, περιέχει) με τη χρήση γραφικών συμβόλων (Βερύκιος, & Βασιλακόπουλος 2022). Αυτή η απεικόνιση καθιστά πιο ευανάγνωστη τη δομή της βάσης δεδομένων και επιτρέπει στους χρήστες να κατανοήσουν πώς συσχετίζονται μεταξύ τους τα δεδομένα (Ramakrisnan & Gehrke 2003). Ακόμη, το μοντέλο οντοτήτων συσχετίσεων

επιτρέπει τη δημιουργία πολλαπλών σχέσεων μεταξύ των οντοτήτων δίνοντας μεγαλύτερη ευελιξία και ακρίβεια στο μοντέλο. Επιπλέον, μπορούν να προστεθούν περιορισμοί για να διασφαλιστεί η συνοχή και η ορθότητα των δεδομένων, όπως ο περιορισμός των τιμών που μπορούν να πάρουν τα γνωρίσματα ενός αντικειμένου (Elmasri, & Navathe 2015). Συνολικά, το μοντέλο οντοτήτων συσχετίσεων είναι ένα πολύ δημοφιλές μοντέλο βάσεων δεδομένων λόγω της ευκολίας στη σχεδίαση, την απλότητα στην κατανόηση και την ευελιξία που παρέχει στους σχεδιαστές βάσεων δεδομένων.



Εικόνα 1: Οι συμβολισμοί του ΜΟΣ

2.3.1 Οντότητα

Οντότητες (Entities): Οι οντότητες αναπαριστούν συγκεκριμένα αντικείμενα ή έννοιες που θέλουμε να αποθηκεύσουμε και να διαχειριστούμε στη βάση δεδομένων (Elmasri, & Navathe 2015). Οι οντότητες αναπαρίστανται συνήθως με ονόματα που περιγράφουν το αντικείμενο ή την έννοια που αντιπροσωπεύουν, όπως ο πελάτης ενός καταστήματος.

Τύπος Οντότητας (Entity Type): Ο τύπος οντότητας αναπαριστά ένα σύνολο οντοτήτων που έχουν τα ίδια γνωρίσματα, για παράδειγμα, ο τύπος οντότητας "Πελάτης" αναπαριστά το σύνολο των πελατών που έχουν κοινά γνωρίσματα όπως το όνομα, το επώνυμο και τη διεύθυνση (Elmasri, & Navathe 2015).

Σύνολο Οντοτήτων (Entity Set): Το σύνολο οντοτήτων αναφέρεται στη συλλογή όλων των τύπων οντοτήτων μιας βάσης δεδομένων. Για παράδειγμα, το σύνολο οντοτήτων “Πελάτες”, “Προιόντα”, “Παραγγελίες” αναφέρεται στους τύπους οντοτήτων που υπάρχουν στην ανάλογη βάση δεδομένων. Στο ER μοντέλο αναπαριστούμε τον τύπο οντοτήτων με ένα ορθογώνιο.

2.3.2 Γνώρισμα

Γνωρίσματα (Attributes): Τα γνωρίσματα αναπαριστούν τις ιδιότητες ή τα χαρακτηριστικά των οντοτήτων. Κάθε οντότητα έχει ένα σύνολο γνωρισμάτων που περιγράφουν τα χαρακτηριστικά της (Ramakrisnan & Gehrke 2003). Συνεχίζοντας το παράδειγμα με τον "Πελάτη", τα γνωρίσματα θα είναι το "Όνομα", το "Επώνυμο", η "Διεύθυνση" κλπ. Τα γνωρίσματα μπορούν να είναι διαφορετικού τύπου δεδομένων, όπως ακέραιοι αριθμοί, αλφαριθμητικά, ημερομηνίες και διακρίνονται στις εξής κατηγορίες:

Απλό Γνώρισμα (Simple Attribute): Ένα απλό γνώρισμα αναφέρεται σε μια μη διαιρούμενη ιδιότητα ενός τύπου οντότητας. Αναπαριστά μια ατομική τιμή που σχετίζεται με την οντότητα. Για παράδειγμα, στον τύπο οντότητας "Πελάτης", το γνώρισμα "Όνομα" είναι ένα απλό γνώρισμα που αναπαριστά το όνομα του πελάτη (Silberschatz , Korth & Sudarshan 2019). Στο ER μοντέλο αναπαριστούμε το απλό γνώρισμα με μία έλλειψη.

Σύνθετο Γνώρισμα (Composite Attribute): Ένα σύνθετο γνώρισμα αναφέρεται σε μια ομάδα συσχετισμένων απλών γνωρισμάτων που συνδυάζονται για να αναπαραστήσουν μια συνολική ιδιότητα της οντότητας (Βερύκιος, & Βασιλακόπουλος 2022). Για παράδειγμα, στον τύπο οντότητας "Πελάτης", το γνώρισμα "Διεύθυνση" μπορεί να αποτελείται από τα υπογνωρίσματα "Οδός", "Πόλη", "Ταχυδρομικός Κώδικας" κλπ.

Μονότιμο Γνώρισμα (Single-valued Attribute): Ένα μονότιμο γνώρισμα αναφέρεται σε ένα γνώρισμα που μπορεί να έχει μία μόνο τιμή για κάθε οντότητα. Αυτό σημαίνει ότι για κάθε εγγραφή της οντότητας, το μονότιμο γνώρισμα έχει μία μοναδική τιμή (Elmasri, & Navathe 2015). Για παράδειγμα, σε έναν τύπο οντότητας "Πελάτης", το γνώρισμα "Αριθμός Τηλεφώνου" μπορεί να θεωρηθεί μονότιμο γνώρισμα, καθώς κάθε πελάτης έχει μόνο έναν αριθμό τηλεφώνου.

Πλειότιμο Γνώρισμα (Multivalued Attribute): Ένα πλειότιμο γνώρισμα αναφέρεται σε ένα γνώρισμα που μπορεί να έχει περισσότερες από μία τιμές για κάθε οντότητα. Αυτό σημαίνει ότι για κάθε εγγραφή της οντότητας, το πλειότιμο γνώρισμα μπορεί να έχει πολλαπλές τιμές (Elmasri, & Navathe 2015). Για παράδειγμα, σε έναν τύπο οντότητας "Προϊόν", το γνώρισμα "Χρώμα" μπορεί να θεωρηθεί πλειότιμο γνώρισμα, καθώς ένα προϊόν μπορεί να έχει πολλά χρώματα. Στο ER μοντέλο αναπαριστούμε το πλειότιμο γνώρισμα με μία έλλειψη η οποία έχει διπλή γραμμή περγράμματος.

Αποθηκευμένο Γνώρισμα (Stored Attribute): Ένα αποθηκευμένο γνώρισμα αναφέρεται σε ένα γνώρισμα που αποθηκεύεται στη βάση δεδομένων και έχει τιμή για κάθε εγγραφή της οντότητας. Τα αποθηκευμένα γνωρίσματα αποτελούν μέρος της δομής της βάσης δεδομένων και αναπαριστούν τα πραγματικά δεδομένα που αποθηκεύονται για κάθε οντότητα (Elmasri, & Navathe 2015). Για παράδειγμα, σε έναν τύπο οντότητας "Πελάτης", το γνώρισμα "Όνομα" μπορεί να θεωρηθεί αποθηκευμένο γνώρισμα, καθώς κάθε πελάτης έχει ένα μοναδικό όνομα που αποθηκεύεται στη βάση δεδομένων.

Παραγόμενο Γνώρισμα (Derived Attribute): Ένα παραγόμενο γνώρισμα αναφέρεται σε ένα γνώρισμα που υπολογίζεται ή παράγεται κάνοντας χρήση ενός αποθηκευμένου γνωρίσματος της οντότητας (Βερύκιος, & Βασιλακόπουλος 2022). Τα παραγόμενα γνωρίσματα δεν αποθηκεύονται φυσικά στη βάση δεδομένων, αλλά υπολογίζονται κατά την ανάκτηση των δεδομένων ή κατά την εκτέλεση ερωτημάτων (Elmasri, & Navathe 2015). Ένα παράδειγμα

παραγόμενου γνωρίσματος είναι ο υπολογισμός της ηλικίας ενός πελάτη βάσει της ημερομηνίας γέννησής του και της τρέχουσας ημερομηνίας. Στο ER μοντέλο αναπαριστούμε το παραγόμενο γνώρισμα με μία έλλειψη η οποία έχει διακεκομμένη γραμμή περιγράμματος.

Null Γνώρισμα: αναφέρονται σε γνωρίσματα που δεν έχουν καμία τιμή ή δεν έχουν καθορισμένη τιμή για μια συγκεκριμένη εγγραφή στη βάση δεδομένων. Ένα null γνώρισμα υποδηλώνει έλλειψη δεδομένων ή μη διαθεσιμότητα τιμής για το συγκεκριμένο γνώρισμα. Ο όρος "null" χρησιμοποιείται συνήθως στα συστήματα διαχείρισης βάσεων δεδομένων (ΣΔΒΔ) για να υποδηλώσει την απουσία τιμής (Elmasri, & Navathe 2015). Το null γνώρισμα μπορεί να συμβολίζεται με διάφορους τρόπους ανάλογα με το ΣΔΒΔ που χρησιμοποιείται, όπως το "NULL", "NaN" (Not a Number) ή άλλα αντίστοιχα σύμβολα. Η χρήση του null γνωρίσματος επιτρέπει την αναπαράσταση και τη διαχείριση απουσιών δεδομένων. Μπορεί να οφείλεται σε πολλούς λόγους, όπως έλλειψη πληροφοριών για ένα συγκεκριμένο γνώρισμα, μη εφαρμογή ή αδυναμία πρόσβασης σε μια τιμή, ή μη ύπαρξη δεδομένων για το συγκεκριμένο γνώρισμα. Κατά την εκτέλεση ερωτημάτων στη βάση δεδομένων, τα null γνωρίσματα μπορούν να αντιμετωπιστούν με τρόπους που επιτρέπουν την αποφυγή σφαλμάτων και την επεξεργασία των υπολοίπων δεδομένων (Elmasri, & Navathe 2015). Είναι σημαντικό να ληφθεί υπόψη η παρουσία null γνωρισμάτων κατά τον σχεδιασμό και την ανάπτυξη της βάσης δεδομένων, καθώς απαιτούν ειδική μεταχείριση για να διασφαλιστεί η σωστή ανάκτηση και επεξεργασία των δεδομένων.

Ο όρος πεδίο ορισμού ενός γνωρίσματος αναφέρεται στο σύνολο των έγκυρων τιμών που μπορεί να πάρει αυτό το γνώρισμα. Το πεδίο ορισμού αφενός καθορίζει το εύρος των επιτρεπόμενων τιμών για ένα γνώρισμα και αφετέρου βοηθά στην επιβολή των περιορισμών ακεραιότητας και συνέπειας των δεδομένων στη βάση δεδομένων (Elmasri, & Navathe 2015). Για παράδειγμα, αν έχουμε ένα γνώρισμα "Ηλικία" σε μια οντότητα "Πελάτες", το πεδίο ορισμού του γνωρίσματος "Ηλικία" είναι το σύνολο των θετικών ακεραίων αριθμών. Αυτό σημαίνει ότι η τιμή του γνωρίσματος "Ηλικία" για κάθε εγγραφή πρέπει να είναι ένας θετικός ακέραιος αριθμός

2.3.3 Κλειδί

Στο ER (Entity-Relationship) μοντέλο, το κλειδί αναφέρεται σε ένα γνώρισμα ενός τύπου οντοτήτων που χρησιμοποιείται για τη μοναδική αναγνώριση της κάθε οντότητας. Το κλειδί διαχωρίζει τις εγγραφές της βάσης δεδομένων και να επιτρέπει την αναζήτηση και την αναγνώριση των εγγραφών (Silberschatz , Korth & Sudarshan 2019). Ένα κλειδί μπορεί να είναι απλό ή σύνθετο. Ένα απλό κλειδί αποτελείται από ένα μόνο γνώρισμα, ενώ ένα σύνθετο κλειδί αποτελείται από περισσότερα του ενός γνωρίσματα (Ramakrisnan & Gehrke 2003). Το κλειδί πρέπει να είναι μοναδικό σε κάθε οντότητα του τύπου οντοτήτων, δηλαδή δεν μπορεί να υπάρχουν δύο οντότητες με τις ίδιες τιμές για το κλειδί, όπως για παράδειγμα ο κωδικός ενός πελάτη στον τύπο οντοτήτων “Πελάτης”. Στο ER μοντέλο αναπαριστούμε το κλειδί με μία έλλειψη η οποία έχει υπογραμμισμένο το όνομά της (Βερύκιος, & Βασιλακόπουλος 2022*)*.

2.3.4 Συσχέτιση

Συσχετίσεις (Relationships): Οι συσχετίσεις αναπαριστούν τις σχέσεις μεταξύ διαφορετικών οντοτήτων. Οι οντότητες συνδέονται μεταξύ τους μέσω συσχετίσεων (Silberschatz , Korth & Sudarshan 2019). Υπάρχουν διάφοροι τύποι συσχετίσεων, όπως μία προς μία, μία προς πολλά και πολλά προς πολλά, ανάλογα με τον τρόπο με τον οποίο οι οντότητες σχετίζονται μεταξύ τους. Για παράδειγμα, μια συσχέτιση "Παραγγελία" μεταξύ των οντοτήτων "Πελάτης" και "Προϊόν" μπορεί να αντιπροσωπεύει το γεγονός ότι ένας πελάτης μπορεί να έχει πολλές παραγγελίες και ένα προϊόν μπορεί να ανήκει σε πολλές παραγγελίες. Στο ER μοντέλο αναπαριστούμε τη συσχέτιση με ένα ρόμβο, ο οποίος συνδέεται μέσω ευθύγραμμων τμημάτων με τις οντότητες που συμμετέχουν στην εν λόγω συσχέτιση (Βερύκιος, & Βασιλακόπουλος 2022*)*.

Ο τύπος συσχέτισης ορίζει τον τρόπο με τον οποίο οι οντότητες συνδέονται και αλληλεπιδρούν μεταξύ τους σε ένα ER μοντέλο. Αυτό επιτρέπει την απεικόνιση πολύπλοκων συσχετίσεων και συνδέσμων μεταξύ των δεδομένων σε μια βάση δεδομένων (Βερύκιος, & Βασιλακόπουλος 2022*)*. Ο τύπος συσχέτισης (relationship type) R ανάμεσα σε n τύπους οντοτήτων E1, E2, ...,

En καθορίζει ένα σύνολο συσχετίσεων μεταξύ των οντοτήτων αυτών των τύπων. Μπορούμε να το διατυπώσουμε με μαθηματικούς όρους ως ένα σύνολο R που αποτελείται από i συσχετίσεις ri, όπου κάθε ri συσχετίζει n ξεχωριστές οντότητες (e1, e2, ..., en). Κάθε οντότητα ej που ανήκει στο ri ανήκει επίσης στον τύπο οντοτήτων Ej, όπου $1 \leq j \leq n$ (Elmasri, & Navathe 2015).

Ο βαθμός ενός τύπου συσχέτισης αναφέρεται στον αριθμό των οντοτήτων που συμμετέχουν στη συσχέτιση. Καθορίζει πόσες οντότητες συνδέονται μεταξύ τους σε μια συσχέτιση (Βερύκιος, & Βασιλακόπουλος 2022*).* Μια συσχέτιση βαθμού 2 ονομάζεται δυαδική, βαθμού 3 τριαδική κτλ. Οι συσχετίσεις μπορούν να συνδέουν οποιονδήποτε αριθμό οντοτήτων μεταξύ τους, ωστόσο, οι πιο συνηθισμένες συσχετίσεις είναι οι δυαδικές, διότι οι πιο πολύπλοκες συσχετίσεις μεγαλύτερου βαθμού (δηλαδή που συνδέουν περισσότερες από δύο οντότητες) τείνουν να είναι πιο δύσκολες στην κατανόηση και στη διαχείριση (Elmasri, & Navathe 2015).

Στο μοντέλο Οντοτήτων-Συσχετίσεων (ER Model), η συσχέτιση ως γνώρισμα αναφέρεται στη χρήση μιας συσχέτισης ως γνωρίσματος σε μια οντότητα. Ας χρησιμοποιήσουμε ένα παράδειγμα για να κατανοήσουμε αυτόν τον όρο. Έστω ότι έχουμε δύο οντότητες, "Πελάτης" και "Παραγγελία", και μια συσχέτιση "Καταχώρηση" μεταξύ τους. Η συσχέτιση "Καταχώρηση" αναπαριστά το γεγονός ότι ένας πελάτης μπορεί να καταχωρήσει πολλές παραγγελίες. Μπορούμε να ορίσουμε την οντότητα "Παραγγελία" με γνωρίσματα όπως "Αριθμός Παραγγελίας", "Ημερομηνία" και ένα γνώρισμα "Πελάτης", που αναπαριστά τη συσχέτιση "Καταχώρηση". Έτσι, στο γνώρισμα "Πελάτης" μπορούμε να αποθηκεύσουμε τον πελάτη που σχετίζεται με κάθε παραγγελία. Η χρήση της συσχέτισης ως γνωρίσματος στο ER Model επιτρέπει την αναπαράσταση συσχετίσεων μεταξύ οντοτήτων ως γνωρίσματα, προσφέροντας περισσότερη ευελιξία στο μοντέλο δεδομένων. Αυτό μας επιτρέπει να αναπαραστήσουμε σύνθετες σχέσεις μεταξύ οντοτήτων και να διαχειριστούμε τις συσχετίσεις αποτελεσματικά.

Ο **λόγος πληθικότητας** (cardinality ratio) σε μια συσχέτιση αναφέρεται στη σχέση μεταξύ του αριθμού των οντοτήτων σε μια πλευρά της συσχέτισης και του αριθμού των οντοτήτων στην άλλη πλευρά (Βερύκιος, & Βασιλακόπουλος 2022*)*. Καθορίζει πόσες οντότητες συνδέονται με μια οντότητα σε μια συσχέτιση. Μπορούμε να έχουμε τους εξής τρεις τύπους λόγους πληθικότητας:

Ένας προς ένα 1:1 (One-to-One): Αυτός ο τύπος λόγου πληθικότητας υποδεικνύει ότι μια οντότητα συσχετίζεται ακριβώς με μία άλλη οντότητα. Σε αυτήν την περίπτωση, οι αριθμοί των οντοτήτων στις δύο πλευρές της συσχέτισης είναι ίσοι (Βερύκιος, & Βασιλακόπουλος 2022*)*.

Ένας προς πολλά 1:N (ή N:1) (One-to-Many): Αυτός ο τύπος λόγου πληθικότητας υποδεικνύει ότι μια οντότητα συσχετίζεται με πολλές οντότητες, ενώ η άλλη πλευρά της συσχέτισης συσχετίζεται με μία και μόνο οντότητα. Σε αυτήν την περίπτωση, ο αριθμός των οντοτήτων στην πλευρά με το "ένα" είναι ένας, ενώ στην πλευρά με το "πολλά" είναι περισσότερος από ένα (Elmasri, & Navathe 2015).

Πολλά προς πολλά N:M (Many-to-Many): Αυτός ο τύπος λόγου πληθικότητας υποδεικνύει ότι πολλές οντότητες συσχετίζονται με πολλές άλλες οντότητες. Σε αυτήν την περίπτωση, ο αριθμός των οντοτήτων στις δύο πλευρές της συσχέτισης μπορεί να είναι οποιοσδήποτε (Βερύκιος, & Βασιλακόπουλος 2022*)*.

Ο λόγος πληθικότητας είναι σημαντικός για τον σχεδιασμό του ER μοντέλου, καθώς καθορίζει τη φύση και την αλληλεπίδραση μεταξύ των οντοτήτων σε μια συσχέτιση. Στο ER μοντέλο αναπαριστούμε το λόγο πληθικότητας με τη χρήση των 1, N, M στα αντίστοιχα ευθύγραμμα τμήματα που συνδέουν τη συσχέτιση με τις οντότητες (Ramakrisnan & Gehrke 2003).

Ο περιορισμός συμμετοχής (participation constraint) σε ένα ER μοντέλο αναφέρεται στον τρόπο με τον οποίο οι οντότητες συμμετέχουν σε μια συσχέτιση (Βερύκιος, & Βασιλακόπουλος 2022*)*. Ο περιορισμός συμμετοχής καθορίζει εάν η συμμετοχή των οντοτήτων είναι ολική (mandatory) ή μερική (partial). Συγκεκριμένα:

Ολική συμμετοχή (Mandatory Participation): Αυτός ο περιορισμός αναφέρει ότι οι οντότητες πρέπει να συμμετέχουν υποχρεωτικά στη συσχέτιση (Βερύκιος, & Βασιλακόπουλος 2022*)*. Αυτό σημαίνει ότι κάθε οντότητα στη συσχέτιση πρέπει να έχει τουλάχιστον μία συσχέτιση με μία άλλη οντότητα στην ίδια συσχέτιση και συμβολίζεται με διπλή γραμμή στο διάγραμμα.

Μερική συμμετοχή (Partial Participation): Αυτός ο περιορισμός αναφέρει ότι οι οντότητες μπορούν να συμμετέχουν προαιρετικά στη συσχέτιση (Βερύκιος, & Βασιλακόπουλος 2022*)*.

Αυτό σημαίνει ότι μια οντότητα μπορεί να έχει μηδενική ή περισσότερες συσχετίσεις με άλλες οντότητες στην ίδια συσχέτιση και συμβολίζεται με απλή γραμμή στο διάγραμμα.

Αναδρομικές συσχετίσεις: Οι αναδρομικές συσχετίσεις αναφέρονται σε περιπτώσεις όπου ένας τύπος οντότητας συσχετίζεται με τον ίδιο τύπο οντότητας (Elmasri, & Navathe 2015). Ένα κλασικό παράδειγμα αναδρομικής συσχέτισης είναι ο "πελάτης" σε μία εταιρεία. Στο πλαίσιο ενός ER μοντέλου, θα είχαμε μία οντότητα "Πελάτης" με γνωρίσματα όπως "Όνομα", "Επίθετο", "Τηλέφωνο" κλπ. Επιπλέον, θα είχαμε μία συσχέτιση "Παραγγελία" μεταξύ των πελατών. Μία παραγγελία μπορεί να έχει πολλούς πελάτες (π.χ., όταν παραγγέλνει μία ομάδα ανθρώπων ή μία εταιρεία), και ταυτόχρονα ένας πελάτης μπορεί να έχει πολλές παραγγελίες. Έτσι, η συσχέτιση "Παραγγελία" είναι μία αναδρομική συσχέτιση, καθώς συσχετίζει τους πελάτες μεταξύ τους. Αυτό σημαίνει ότι ένας πελάτης μπορεί να είναι ο αποστολέας μίας παραγγελίας, αλλά μπορεί επίσης να είναι ένας από τους παραλήπτες της ίδιας παραγγελίας. Αυτό το παράδειγμα δείχνει πώς οι αναδρομικές συσχετίσεις μπορούν να χρησιμοποιηθούν για να μοντελοποιηθούν πολύπλοκες σχέσεις μεταξύ των οντοτήτων και να αντιμετωπιστεί η πολυπλοκότητα του πραγματικού κόσμου (Ramakrisnan & Gehrke 2003).

Σημειώνουμε ότι όπως στην περίπτωση των οντοτήτων, μια συσχέτιση μπορεί να έχει γνωρίσματα. Τα γνωρίσματα συσχέτισης αναφέρονται στις ιδιότητες που αντιστοιχούν σε αυτή και περιγράφουν πληροφορίες σχετικά με τη σχέση μεταξύ των οντοτήτων που συμμετέχουν σε αυτήν (Elmasri, & Navathe 2015).

2.3.5 Ασθενής Οντότητα και Προσδιορίζουσα Συσχέτιση

Ασθενής Οντότητα (Weak Entity): Ένας τύπος οντοτήτων χαρακτηρίζεται ασθενής όταν δεν μπορεί να υπάρξει ανεξάρτητα από μια άλλη ισχυρή οντότητα, γνωστή ως ιδιοκτήτη/προσδιορίζοντα τύπο οντότητας. Η συσχέτιση μεταξύ της ασθενούς οντότητας και του ιδιοκτήτη ονομάζεται **προσδιορίζουσα** (Silberschatz , Korth & Sudarshan 2019). Η ασθενής οντότητα αναπαριστάται με ορθογώνιο με διπλό περίγραμμα, ενώ η προσδιορίζουσα συσχέτιση με ρόμβο που έχει διπλό περίγραμμα. Ένας ασθενής τύπος οντοτήτων στο ER μοντέλο πρέπει να μπορεί να προσδιορίζεται μοναδικά μεταξύ όλων των συσχετιζόμενων οντοτήτων που έχουν τον ίδια ιδιοκτήτη (Elmasri, & Navathe 2015). Για τον σκοπό αυτό, ένας

ασθενής τύπος οντοτήτων συνήθως διαθέτει ένα ή περισσότερα γνωρίσματα που σε συνδυασμό αποδίδουν την μοναδική αυτή ιδιότητα. Το σύνολο αυτών των γνωρισμάτων καλείται μερικό κλειδί και αναπαριστάται όπως το κλειδί αλλά με διακεκομμένη υπογράμμιση (Βερύκιος, & Βασιλακόπουλος 2022*)*.

Στο ER μοντέλο, μια ασθενής οντότητα δεν μπορεί να προσδιοριστεί μόνη της, αλλά πρέπει πάντα να συμμετέχει σε μια προσδιορίζουσα συσχέτιση (Elmasri, & Navathe 2015). Αυτό σημαίνει ότι η ασθενής οντότητα έχει πάντοτε ολική συμμετοχή στην συσχέτιση που συνδέει την ασθενή οντότητα με άλλες οντότητες (Βερύκιος, & Βασιλακόπουλος 2022*)*.

2.4 Εκτεταμένο Μοντέλο Οντοτήτων Συσχετίσεων

Το εκτεταμένο μοντέλο οντοτήτων συσχετίσεων (Extended Entity-Relationship model, EER) είναι μια επέκταση του ER μοντέλου που προσθέτει έννοιες στο ΜΟΣ για την αναπαράσταση πιο πολύπλοκων σεναρίων και δομών δεδομένων (Βερύκιος, & Βασιλακόπουλος 2022*)*.

2.4.1 Υπερκλάση, υποκλάση και κληρονομικότητα

Μια υπερκλάση αναπαριστά μια γενική οντότητα από την οποία προέρχονται άλλες συγκεκριμένες οντότητες, οι οποίες ονομάζονται υποκλάσεις (Silberschatz , Korth & Sudarshan 2019). Η υπερκλάση περιγράφει τα κοινά γνωρίσματα και συσχετίσεις που ισχύουν για όλες τις υποκλάσεις της. Στην ουσία, η υπερκλάση αποτελεί την αφαίρεση των κοινών γνωρισμάτων και συσχετίσεων από τις υποκλάσεις (Elmasri, & Navathe 2015).

Οι υποκλάσεις είναι οντότητες που κληρονομούν τα γνωρίσματα και τις συσχετίσεις από την υπερκλάση. Παρουσιάζουν τα ειδικά γνωρίσματα και συσχετίσεις που τις διαφοροποιούν από τις άλλες υποκλάσεις. Οι υπερκλάσεις και οι υποκλάσεις διαμορφώνουν μια ιεραρχική σχέση μεταξύ των οντοτήτων, παρέχοντας μια μέθοδο οργάνωσης και κληρονομικότητας των γνωρισμάτων και των συσχετίσεων (Elmasri, & Navathe 2015).

2.4.2 Γενίκευση και Εξειδίκευση

Στο εκτεταμένο μοντέλο οντοτήτων-συσχετίσεων (EER), η γενίκευση και η εξειδίκευση χρησιμοποιούνται για να αναπαραστήσουν την ιεραρχική σχέση μεταξύ των οντοτήτων (Silberschatz , Korth & Sudarshan 2019). Ας δούμε τι σημαίνουν αυτοί οι όροι:

Εξειδίκευση(specialization): Η εξειδίκευση αναφέρεται στη διαδικασία όπου μια υπερκλάση διαιρείται σε δύο ή περισσότερες υποκλάσεις που κληρονομούν τα γνωρίσματα και τις συσχετίσεις της υπερκλάσης. Κάθε υποκλάση προσθέτει επιπλέον γνωρίσματα και συσχετίσεις που την διαφοροποιούν από τις άλλες υποκλάσεις (Elmasri, & Navathe 2015). Η εξειδίκευση χρησιμοποιείται για την αναπαράσταση της ειδικότητας και των διαφορών μεταξύ των οντοτήτων.

Στο πλαίσιο του εκτεταμένου μοντέλου οντοτήτων-συσχετίσεων (EER), η μερική εξειδίκευση (partial specialization) και η ολική εξειδίκευση (total specialization) αναφέρονται στις διαδικασίες όπου μια υπερκλάση διαιρείται σε υποκλάσεις για να αναπαραστήσει τις διαφορετικές ειδικές κατηγορίες οντοτήτων (Elmasri, & Navathe 2015).

Η μερική εξειδίκευση συμβαίνει όταν μια υπερκλάση διαιρείται σε υποκλάσεις, αλλά υπάρχουν οντότητες που ανήκουν στην υπερκλάση αλλά δεν ανήκουν σε καμία υποκλάση. Αυτές οι οντότητες δεν έχουν αρκετά γνωρίσματα για να ταυτοποιηθούν σε μια ειδική κατηγορία, οπότε παραμένουν στην υπερκλάση.

Αντίθετα, η ολική εξειδίκευση συμβαίνει όταν μια υπερκλάση διαιρείται σε υποκλάσεις και κάθε οντότητα ανήκει αποκλειστικά σε μια υποκλάση (Ramakrisnan & Gehrke 2003). Κάθε υποκλάση προσδιορίζει επιπλέον γνωρίσματα και σχέσεις που είναι ιδιαίτερα σχετικά με τα χαρακτηριστικά της ειδικής κατηγορίας οντοτήτων που αναπαριστά. Αναπαριστάται με διπλή γραμμή στη σύνδεση του κύκλου με την υπεκλάση (Βερύκιος, & Βασιλακόπουλος 2022).

Η επιλογή μεταξύ μερικής εξειδίκευσης και ολικής εξειδίκευσης εξαρτάται από τις απαιτήσεις του προβλήματος που προσπαθούμε να μοντελοποιήσουμε. Αν θέλουμε να επιτρέψουμε οντότητες να ανήκουν σε πολλαπλές κατηγορίες, τότε χρησιμοποιούμε μερική εξειδίκευση. Αν θέλουμε να απαιτήσουμε από κάθε οντότητα να ανήκει σε μία μόνο κατηγορία, τότε χρησιμοποιούμε ολική εξειδίκευση (Ramakrisnan & Gehrke 2003).

Η έννοια της επικαλυπτόμενης και μη επικαλυπτόμενης εξειδίκευσης αφορά τη σχέση μεταξύ των υποκλάσεων μιας υπερκλάσης σε ένα εκτεταμένο μοντέλο οντοτήτων-συσχετίσεων (EER) (Βερύκιος, & Βασιλακόπουλος 2022*)*.

Η επικαλυπτόμενη εξειδίκευση (overlapping specialization) συμβαίνει όταν οι υποκλάσεις μιας υπερκλάσης καλύπτουν διαφορετικά υποσύνολα της υπερκλάσης. Αυτό σημαίνει ότι μία οντότητα μπορεί να ανήκει σε πολλές υποκλάσεις ταυτόχρονα (Ramakrisnan & Gehrke 2003). Κάθε υποκλάση προσδιορίζει μοναδικά γνωρίσματα και σχέσεις που είναι σχετικά με την ειδική κατηγορία των οντοτήτων που αναπαριστά. Έτσι, μια οντότητα μπορεί να έχει γνωρίσματα που ανήκουν σε πολλές υποκλάσεις. Αναπαριστάται με έναν κύκλο ο οποίος περικλείει το γράμμα ο (Βερύκιος, & Βασιλακόπουλος 2022*)*.

Αντίθετα, η μη επικαλυπτόμενη εξειδίκευση (disjoint specialization) συμβαίνει όταν οι υποκλάσεις μιας υπερκλάσης δεν επικαλύπτονται, δηλαδή κάθε οντότητα ανήκει αποκλειστικά σε μία μόνο υποκλάση (Ramakrisnan & Gehrke 2003). Κάθε υποκλάση προσδιορίζει μοναδικά γνωρίσματα και σχέσεις που είναι σχετικά με την ειδική κατηγορία των οντοτήτων που αναπαριστά. Έτσι, μια οντότητα ανήκει μόνο σε μία υποκλάση και έχει τα γνωρίσματα που σχετίζονται με αυτήν την υποκλάση. Αναπαριστάται με έναν κύκλο ο οποίος περικλείει το γράμμα d (Βερύκιος, & Βασιλακόπουλος 2022*)*.

Η επιλογή ανάμεσα σε επικαλυπτόμενη και μη επικαλυπτόμενη εξειδίκευση εξαρτάται από τις απαιτήσεις και τη λογική του προβλήματος που μοντελοποιούμε. Η επικαλυπτόμενη εξειδίκευση επιτρέπει μεγαλύτερη ευελιξία και πολυπλοκότητα, αλλά μπορεί να οδηγήσει σε πιο περίπλοκα μοντέλα και στην ανάγκη για περαιτέρω αναζήτηση και ανάλυση (Elmasri, & Navathe 2015). Η μη επικαλυπτόμενη εξειδίκευση προσφέρει πιο απλά μοντέλα και πιο σαφείς σχέσεις ανάμεσα στις κατηγορίες, αλλά περιορίζει την ευελιξία και τη δυνατότητα αναπαράστασης πολύπλοκων δομών (Ramakrisnan & Gehrke 2003).

Όταν η διάκριση των υποκλάσεων γίνεται βάσει ενός γνωρίσματος της υπερκλάσης, μπορούμε να προσθέσουμε αυτήν την συνθήκη στο διάγραμμα. Αυτό το γνώρισμα που χρησιμοποιείται για να καθορίσει την υποκλάση μιας οντότητας ονομάζεται ορίζον γνώρισμα (defining attribute), (Βερύκιος, & Βασιλακόπουλος 2022*)*. Στο διάγραμμα, το όνομα του ορίζον γνωρίσματος τοποθετείται κοντά στη γραμμή που συνδέει την υπερκλάση με τον κύκλο. Η προσθήκη αυτής της συνθήκης στο διάγραμμα επισημαίνει ότι η οντότητα ανήκει σε μια

συγκεκριμένη υποκλάση μόνο εάν πληροί τη συνθήκη που ορίζεται από το ορίζον γνώρισμα (Elmasri, & Navathe 2015).

Γενίκευση(generalization): Η γενίκευση αναφέρεται στη διαδικασία όπου πολλαπλές υπάρχουσες οντότητες συνδυάζονται για να δημιουργήσουν μια νέα υπερκλάση, η οποία αντιπροσωπεύει τα κοινά χαρακτηριστικά των αρχικών οντοτήτων (Silberschatz , Korth & Sudarshan 2019). Η γενίκευση βοηθά στην οργάνωση και στην αποφυγή της επανάληψης πληροφορίας. Η νέα υπερκλάση κληρονομεί τα γνωρίσματα και τις συσχετίσεις των υποκλάσεων της. Ουσιαστικά πρόκειται για την αντίστροφη διαδικασίας της εξειδίκευσης και συνεπώς έχει ίδια αναπαράσταση στα διαγράμματα. Στην περίπτωση μίας μόνο υποκλάσης δε δημιουργούμε γενίκευση/εξειδίκευση (Elmasri, & Navathe 2015). Η σχέση υπερκλάσης-υποκλάσης παριστάνεται με τη σύνδεση των δύο τύπων οντοτήτων με μία ευθεία γραμμή, όπου ο άνω τύπος οντότητας είναι η υπερκλάση και ο κάτω τύπος οντότητας είναι η υποκλάση (Βερύκιος, & Βασιλακόπουλος 2022*)*. Αυτή η γραμμή αναπαριστά την ιεραρχική σχέση μεταξύ των δύο τύπων οντοτήτων, όπου ο υποκλάση κληρονομεί τις ιδιότητες και τις συσχετίσεις της υπερκλάσης. Παράλληλα, χρησιμοποιείται το σύμβολο υποσυνόλου που δείχνει από την υπερκλάση προς την υποκλάση και υποδηλώνει ότι η υποκλάση αποτελεί ένα υποσύνολο της υπερκλάσης, δηλαδή οι οντότητες της υποκλάσης είναι μία ειδική κατηγορία των οντοτήτων της υπερκλάσης (Elmasri, & Navathe 2015).

2.4.3 Ένωση

Ένωση ή Κατηγορία (Union): Η ένωση αναφέρεται στην ικανότητα μιας οντότητας να ανήκει σε πολλαπλές υπερκλάσεις. Αυτό σημαίνει ότι μια οντότητα μπορεί να έχει τα χαρακτηριστικά και τη συμπεριφορά που ορίζονται από διάφορες υπερκλάσεις (Silberschatz , Korth & Sudarshan 2019). Η ένωση ουσιαστικά επιτρέπει να αντιμετωπίσουμε μια οντότητα ως μέλος πολλαπλών ομάδων ταυτόχρονα. Αναπαριστάται με έναν κύκλο ο οποίος περικλείει το γράμμα U και συνδέεται με απλές γραμμές με τις υπερκλάσεις (Βερύκιος, & Βασιλακόπουλος 2022*)*. Για την υποκλάση η σύνδεση γίνεται με απλή ή διπλή γραμμή που περιλαμβάνει το σύμβολο του υποσυνόλου για να υποδείξει ότι η υποκλάση αποτελεί ένα υποσύνολο της υπερκλάσης. Η ολική συμμετοχή, που συμβολίζεται με διπλή γραμμή, σημαίνει ότι η υποκλάση περιλαμβάνει όλες τις οντότητες που ανήκουν στην ένωση των υπερκλάσεων.

Αντίθετα, η μερική συμμετοχή σημαίνει ότι η υποκλάση περιλαμβάνει μόνο ένα υποσύνολο των οντοτήτων που ανήκουν στην ένωση των υπερκλάσεων (Ramakrisnan & Gehrke 2003).

2.5 Σχεσιακό Μοντέλο

Το σχεσιακό μοντέλο είναι ένα μοντέλο βάσεων δεδομένων που αναπτύχθηκε από τον Edgar F. Codd στις αρχές της δεκαετίας του 1970. Αποτελεί το πιο διαδεδομένο μοντέλο για την οργάνωση και αποθήκευση δεδομένων σε σύγχρονες εφαρμογές και βασίζεται στην έννοια των σχέσεων, οι οποίες αναπαριστούν συλλογές δεδομένων με συγκεκριμένη δομή (Connolly & Begg 2015). Το σχεσιακό μοντέλο χρησιμοποιεί μια συλλογή πινάκων για να αναπαραστήσει τα δεδομένα και τις σχέσεις μεταξύ τους (Ramakrisnan & Gehrke 2003).

Κάθε πίνακας στο σχεσιακό μοντέλο ονομάζεται σχέση και αντιπροσωπεύει μια οντότητα ή ένα σύνολο οντοτήτων του συστήματος. Κάθε γραμμή του πίνακα(που ονομάζεται πλειάδα) αντιστοιχεί σε μια εγγραφή ή σε μια συγκεκριμένη εμφάνιση της οντότητας, ενώ κάθε στήλη (γνώρισμα) του πίνακα αντιστοιχεί σε μια ιδιότητα ή ένα γνώρισμα της οντότητας (Ramakrisnan & Gehrke 2003).

Οι σχέσεις μεταξύ των οντοτήτων παριστώνται στο σχεσιακό μοντέλο μέσω ειδικών πινάκων που ονομάζονται πίνακες συσχέτισης (Connolly & Begg 2015). Κάθε πίνακας συσχέτισης αντιπροσωπεύει μια σχέση μεταξύ δυο ή περισσότερων οντοτήτων και περιλαμβάνει στήλες που αντιπροσωπεύουν τα γνωρίσματα των σχέσεων (Ramakrisnan & Gehrke 2003).

Με τη βοήθεια του σχεσιακού μοντέλου, μπορούμε να αποθηκεύσουμε, να ανακτήσουμε, να ενημερώσουμε και να διαγράψουμε δεδομένα με έναν αποτελεσματικό τρόπο (Elmasri, & Navathe 2015).Οι σχεσιακές βάσεις δεδομένων χρησιμοποιούνται ευρέως σε πολλούς τομείς, όπως επιχειρηματικές εφαρμογές, ιστοσελίδες, συστήματα διαχείρισης περιεχομένου και πολλά άλλα.

Ας υποθέσουμε ότι έχουμε τον πίνακα "Customers" (Πελάτες) με τα ακόλουθα γνωρίσματα:

- CustomerID (Αναγνωριστικός αριθμός πελάτη)
- FirstName (Όνομα)

- LastName (Επώνυμο)
- Email (Ηλεκτρονική διεύθυνση)
- Address (Διεύθυνση)

Έστω ότι έχουμε 3 εγγεγραμμένους πελάτες στον πίνακα "Customers":

CustomerID	FirstName	LastName	Email	Address
1234444	Μιχάλης	Παπαδόπουλος	michael@example.com	Οδός Αθηνών 123
2567532	Ζωή	Παπαδοπούλου	zoe@example.com	Οδός Πειραιώς 456
3567891	Χρήστος	Παπανικολάου	christos@example.com	Οδός Θεσσαλονίκης 789

Εικόνα 2: Ένα στιγμιότυπο της σχέσης Πελάτης

Ο παραπάνω πίνακας αντιπροσωπεύει ένα στιγμιότυπο της σχέσης Πελάτης. Ο βαθμός της σχέσης αναφέρεται στον αριθμό των γνωρισμάτων ή στηλών που περιέχει ο πίνακας, στην περίπτωσή μας είναι πέντε (CustomerID, FirstName, LastName, Email, Address). Η πληθικότητα της σχέσης αναφέρεται στον αριθμό των εγγραφών ή γραμμών στον πίνακα γνωστές ως πλειάδες, δηλαδή τον αριθμό των πελατών που αντιπροσωπεύονται, όπου στην περίπτωσή μας είναι τρεις.

Για να προσδιορίσουμε το πεδίο ορισμού ενός γνωρίσματος, πρέπει να γνωρίζουμε τον τύπο δεδομένων που μπορεί να πάρει και τη μορφή του. Η μορφή αναφέρεται στον τρόπο παρουσίασης ή μορφοποίησης των τιμών του γνωρίσματος (Connolly & Begg 2015). Για παράδειγμα, ένα γνώρισμα "Ημερομηνία Γέννησης" μπορεί να έχει τον τύπο δεδομένων "Ημερομηνία" και να παρουσιάζεται στη μορφή "HH/MM/EEEE".

Για κάθε σχέση, το πεδίο ορισμού κάθε γνωρίσματος πρέπει να είναι ατομικό, που σημαίνει ότι οι τιμές του γνωρίσματος θεωρούνται αδιαίρετες μονάδες. Αυτό σημαίνει ότι η τιμή του γνωρίσματος δεν μπορεί να διαιρεθεί περαιτέρω σε μικρότερες συνιστώσες (Connolly & Begg 2015). Για παράδειγμα, ένα γνώρισμα "Αριθμός Ταυτότητας" μπορεί να θεωρηθεί ατομικό γνώρισμα, καθώς ο αριθμός δεν μπορεί να χωριστεί σε μικρότερα τμήματα.

Επίσης, είναι σημαντικό να κατανοήσουμε τη διάκριση μεταξύ του σχήματος βάσης δεδομένων και του στιγμιότυπου. Το σχήμα της βάσης δεδομένων αναφέρεται στο λογικό σχεδιασμό της βάσης δεδομένων, δηλαδή τον τρόπο με τον οποίο οργανώνονται και συσχετίζονται οι πίνακες και τα γνωρίσματα (Elmasri, & Navathe 2015). Από την άλλη πλευρά, το στιγμιότυπο αναφέρεται στα δεδομένα που βρίσκονται αποθηκευμένα στη βάση δεδομένων κατά μια συγκεκριμένη στιγμή. Έτσι, μπορούμε να πούμε ότι το σχήμα της βάσης δεδομένων καθορίζει τη δομή και τις συσχετίσεις μεταξύ των διαφόρων πινάκων και γνωρισμάτων, ενώ το στιγμιότυπο αναφέρεται στο πραγματικό περιεχόμενο της βάσης δεδομένων σε μια συγκεκριμένη στιγμή (Connolly & Begg 2015).

Στο σχεσιακό μοντέλο, κάθε πίνακας έχει ένα πρωτεύον κλειδί (primary key) που χρησιμοποιείται για να ξεχωρίζουν με μοναδικό τρόπο οι πλειάδες (Connolly & Begg 2015). Το πρωτεύον κλειδί είναι ένα γνώρισμα του πίνακα που έχει μια μοναδική τιμή για κάθε πλειάδα (Elmasri, & Navathe 2015). Για παράδειγμα, σε έναν πίνακα που περιγράφει καταστήματα μιας αλυσίδας, είναι σημαντικό να υπάρχει μοναδικός αναγνωριστικός κωδικός για κάθε κατάστημα, γνωστός ως πρωτεύον κλειδί. Αποφεύγεται η χρήση του γνωρίσματος Πόλη ως αναγνωριστικό γνώρισμα, επειδή πολλά καταστήματα ενδέχεται να βρίσκονται στην ίδια πόλη. Αν ένας πίνακας έχει περισσότερα από ένα γνωρίσματα που δύνανται να είναι αναγνωριστικά, πρέπει να επιλεχθεί ένα μόνο από αυτά ως πρωτεύον κλειδί, χαρακτηρίζοντας τα υπόλοιπα ως υποψήφια κλειδιά. Σε περίπτωση που ως πρωτεύον κλειδί έχει οριστεί ένα γνώρισμα που αποτελείται από άλλα γνωρίσματα, ονομάζεται σύνθετο κλειδί (composite key). Τα γνωρίσματα που δεν έχουν μοναδικές τιμές μεταξύ των πλειάδων ονομάζονται δευτερεύοντα (secondary) κλειδιά. (Elmasri, & Navathe 2015).

Η ομοιότητα μεταξύ του μοντέλου οντοτήτων συσχετίσεων ER και του σχεσιακού μοντέλου είναι εμφανής με τις οντότητες του μοντέλου ER αντιστοιχούν σε πίνακες στο σχεσιακό μοντέλο και κάθε γνώρισμα του μοντέλου ER να αντιστοιχεί σε μία στήλη του πίνακα (Elmasri, & Navathe 2015).

2.6 Ο κώδικας SQL

Ο κώδικας SQL (Structured Query Language) είναι ένας γλώσσα προγραμματισμού που χρησιμοποιείται για τη διαχείριση και αλληλεπίδραση με σχεσιακές βάσεις δεδομένων (Connolly & Begg 2015). Επιτρέπει τη δημιουργία, την τροποποίηση και την ανάκτηση δεδομένων από μια βάση δεδομένων και αποτελείται από εντολές που εκτελούν διάφορες λειτουργίες. Ενδεικτικά αναφέρουμε τις παρακάτω:

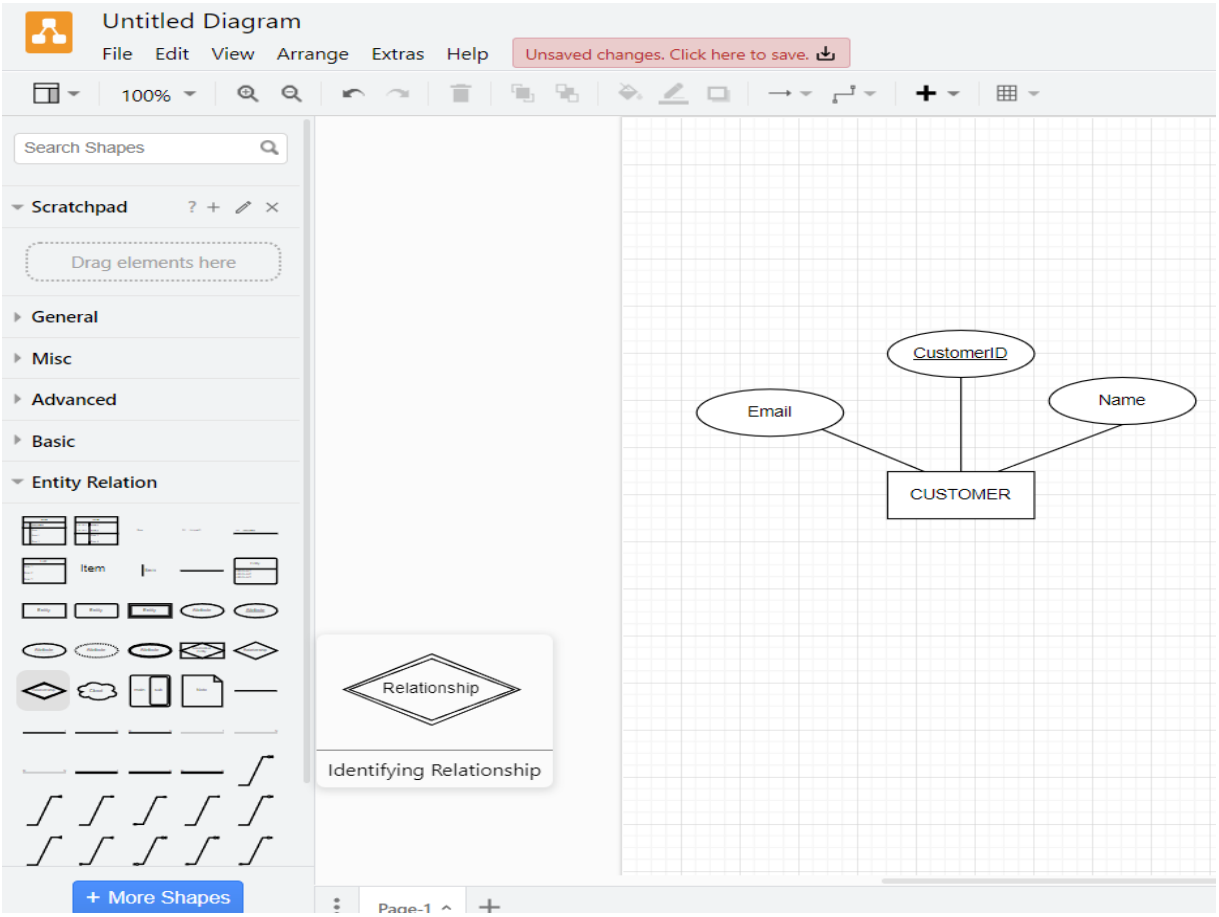
- CREATE: Χρησιμοποιείται για τη δημιουργία νέων αντικειμένων στη βάση δεδομένων, όπως πίνακες, ευρετήρια, διαδικασίες. Για παράδειγμα, η εντολή CREATE TABLE χρησιμοποιείται για τη δημιουργία ενός νέου πίνακα στη βάση δεδομένων (Silberschatz , Korth & Sudarshan 2019). Με αυτήν την εντολή ορίζουμε το όνομα του πίνακα και τις στήλες που περιλαμβάνει, καθεμία με τον τύπο δεδομένων της. Μπορούμε επίσης να ορίσουμε περιορισμούς, όπως περιορισμούς κλειδιού (PRIMARY KEY), περιορισμούς μοναδικότητας (UNIQUE) (Connolly & Begg 2015).
- SELECT: Χρησιμοποιείται για την ανάκτηση δεδομένων από μια βάση δεδομένων. Αποτελεί μία από τις πιο σημαντικές εντολές στην SQL και χρησιμοποιείται για την εκτέλεση ερωτημάτων (queries) που επιστρέφουν αποτελέσματα. Με αυτήν την εντολή ορίζουμε τις στήλες που θέλουμε να επιστραφούν από τον πίνακα. Μπορούμε επίσης να χρησιμοποιήσουμε τον τελεστή "*" για να επιλέξουμε όλες τις στήλες του πίνακα υπερκλάσεις (Silberschatz , Korth & Sudarshan 2019).
- UPDATE: Χρησιμοποιείται για την τροποποίηση υπάρχουσων εγγραφών σε μια βάση δεδομένων. Συγκεκριμένα, επιτρέπει την ενημέρωση τιμών συγκεκριμένων στηλών μιας εγγραφής ή πολλών εγγραφών σε μια βάση δεδομένων υπερκλάσεις (Silberschatz , Korth & Sudarshan 2019). Με αυτήν την εντολή, προσδιορίζουμε τον πίνακα που θέλουμε να ενημερώσουμε και ορίζουμε τις νέες τιμές για τις στήλες που θέλουμε να τροποποιηθούν. Οι νέες τιμές ορίζονται με τη χρήση της λέξης-κλειδί SET.
- ALTER: Χρησιμοποιείται για την τροποποίηση των δομών των αντικειμένων στη βάση δεδομένων, όπως προσθήκη ή διαγραφή στηλών από έναν πίνακα και παρέχει ευελιξία στην τροποποίηση της δομής των αντικειμένων στη βάση δεδομένων σύμφωνα με τις ανάγκες της εφαρμογής υπερκλάσεις (Silberschatz , Korth & Sudarshan 2019).
- CASCADE: Χρησιμοποιείται για τον έλεγχο των ενεργειών που αφορούν τις εξαρτώμενες σχέσεις μεταξύ των πινάκων σε μια βάση δεδομένων. Όταν εκτελείται

μια ενέργεια (όπως η διαγραφή, η τροποποίηση ή η ενημέρωση) σε έναν πίνακα, η CASCADE επεκτείνει την επίδραση αυτής της ενέργειας στους σχετικούς πίνακες που έχουν εξαρτήσεις από τον εν λόγω πίνακα (Connolly & Begg 2015). Για παράδειγμα, αν διαγράφεται μια εγγραφή από έναν πίνακα και έχει οριστεί CASCADE για τη διαγραφή, τότε αυτή η ενέργεια θα επεκταθεί και στους σχετικούς πίνακες, με αποτέλεσμα τη διαγραφή των σχετικών εγγραφών από αυτούς τους πίνακες. Με αυτόν τον τρόπο εξασφαλίζουμε ότι εάν εκτελέσουμε μια ενέργεια σε έναν πίνακα τότε και οι αντίστοιχες ενέργειες θα εφαρμοστούν αυτόματα στους σχετικούς πίνακες, αποφεύγοντας την ανεπιθύμητη ασυνέπεια των δεδομένων υπερκλάσεις (Silberschatz , Korth & Sudarshan 2019).

3. Υπάρχουσες Εφαρμογές

Ο σχεδιασμός διαγραμμάτων οντοτήτων-συσχετίσεων (ER) είναι ένα σημαντικό βήμα στην ανάπτυξη μιας βάσης δεδομένων, όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο. Υπάρχουν πολλά εργαλεία που μπορούν να βοηθήσουν στο σχεδιασμό τέτοιων διαγραμμάτων, παρέχοντας έναν ευέλικτο και εύχρηστο τρόπο για την αναπαράσταση των οντοτήτων, των συσχετίσεων και των γνωρισμάτων τους. Σε αυτό το κεφάλαιο παρουσιάζονται ενδεικτικά μερικά από τα πιο δημοφιλέστερα από αυτά.

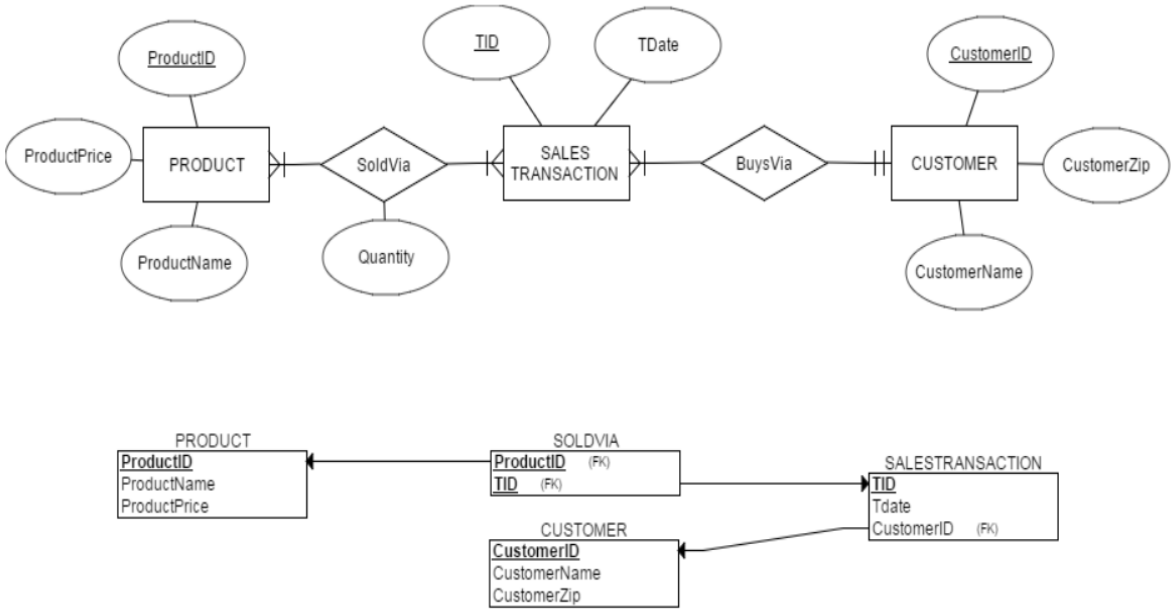
3.1 diagrams.net



Εικόνα 3: Η εφαρμογή diagrams.net

Το diagrams.net είναι ένα διαδικτυακό εργαλείο σχεδίασης διαγραμμάτων, προηγουμένως γνωστό ως draw.io. Παρέχει μια ευέλικτη πλατφόρμα για τη δημιουργία διαφόρων ειδών διαγραμμάτων, συμπεριλαμβανομένων και των διαγραμμάτων οντοτήτων-συσχετίσεων (Entity Relationship Diagrams - ERDs). Παρόλα αυτά, δεν επιβάλλει αυστηρούς ορισμούς όσον αφορά τη σύνδεση των αντικειμένων σε ένα διάγραμμα, ούτε μετατρέπει το διάγραμμα σε Σχεσιακό Σχήμα. Ακόμη, παρόλο που παρέχει μια ευρεία γκάμα σχημάτων και συμβόλων, απουσιάζουν τα σύμβολα των ενώσεων και εξειδικεύσεων τα οποία καλείται να δημιουργήσει ο χρήστης εάν απαιτούνται στο σχεδιασμό του διαγράμματός του.

3.2 ERDPlus

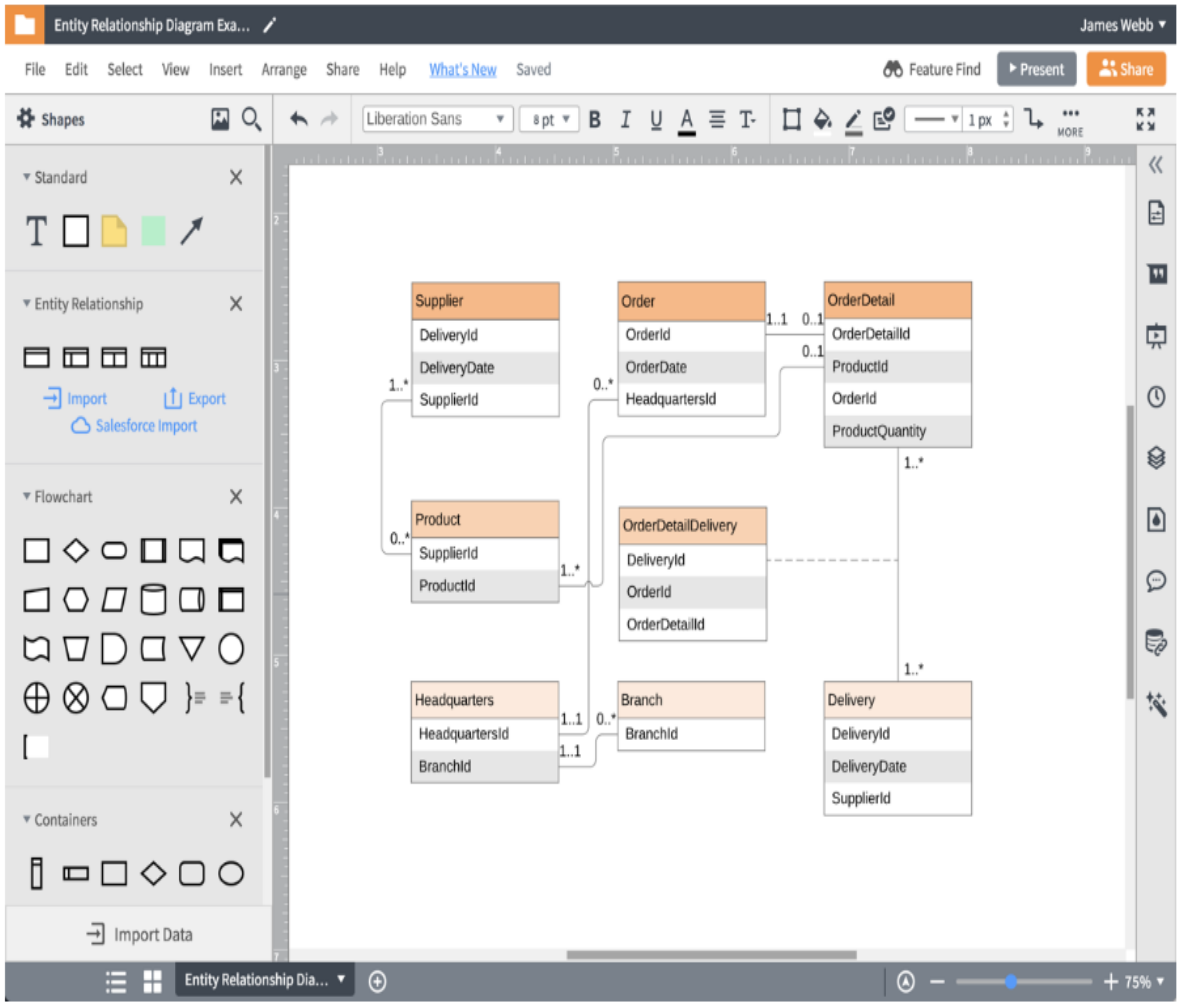


Εικόνα 4: Η μετατροπή σε σχεσιακό σχήμα μέσω της εφαρμογής ERDPlus

Το ERDPlus είναι ένα διαδικτυακό εργαλείο σχεδίασης διαγραμμάτων οντοτήτων-συσχετίσεων (ER). Πρόκειται για μια φιλική προς τον χρήστη διεπαφή χωρίς την απαιτούμενη

εγκατάσταση επιπλέον λογισμικού, καθιστώντας την προσβάσιμη από οποιαδήποτε συσκευή με σύνδεση στο διαδίκτυο. Το ERDPlus παρέχει μια πληθώρα εργαλείων και συμβόλων που επιτρέπουν την προσθήκη οντοτήτων, συσχετίσεων, γνωρισμάτων, κλειδιών και άλλων στοιχείων στο διάγραμμα, αλλά χρησιμοποιείται μόνο για το απλό μοντέλο. Ένα από τα κύρια χαρακτηριστικά του ERDPlus είναι η δυνατότητα αυτόματης μετατροπής του διαγράμματος σε σχεσιακό σχήμα βάσης δεδομένων και η παραγωγή κώδικα SQL. Ακόμη, επιτρέπει την εξαγωγή των διαγραμμάτων ως αρχεία με τη μορφή εικόνας PNG, όμως απουσιάζει έλεγχος της ορθότητας του διαγράμματος με αποτέλεσμα να απαιτεί περαιτέρω προσαρμογές και επεξεργασία. Τέλος, το ERDPlus επιτρέπει τη σχεδίαση και ανάπτυξη διαγραμμάτων ERD μεταξύ πολλαπλών χρηστών.

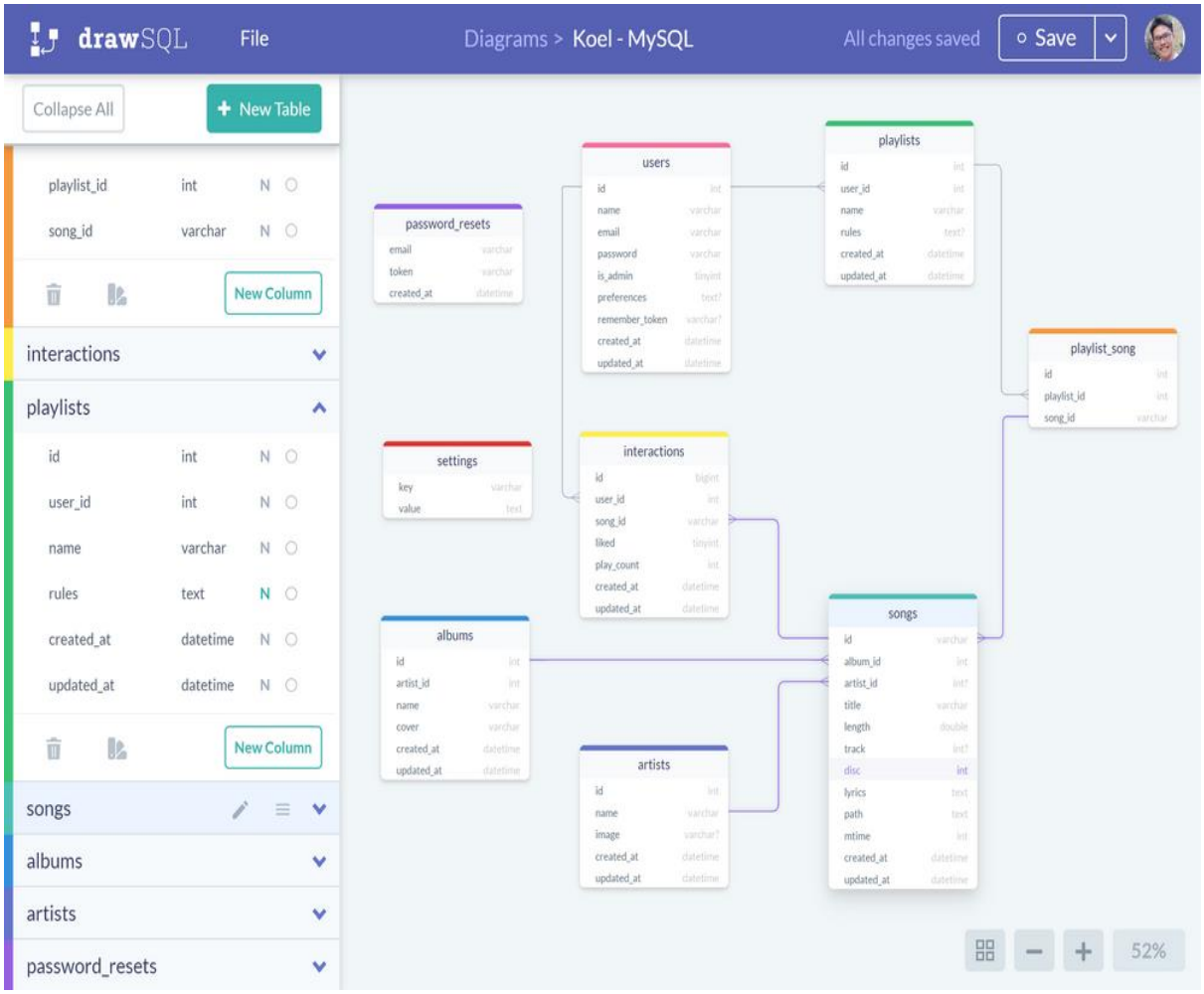
3.3 Lucidchart



Εικόνα 5: Παράδειγμα διαγράμματος στο Lucidchart

Το Lucidchart είναι ένα διαδικτυακό εργαλείο σχεδίασης διαγραμμάτων που παρέχει πλούσια λειτουργικότητα για τη δημιουργία διαφόρων ειδών διαγραμμάτων, συμπεριλαμβανομένων των διαγραμμάτων οντοτήτων-συσχετίσεων (ERDs). Παρέχει εργαλεία και σύμβολα για την αναπαράσταση οντοτήτων, συσχετίσεων, γνωρισμάτων και κλειδιών. Επιπλέον, το Lucidchart επιτρέπει τη συνεργασία των χρηστών σε πραγματικό χρόνο για τον σχεδιασμό και την επεξεργασία των διαγραμμάτων, ενώ παρέχονται δυνατότητες εξαγωγής των διαγραμμάτων σε διάφορες μορφές αρχείων, όπως εικόνες, PDF και αρχεία Visio, προκειμένου να ενσωματωθούν σε άλλα έγγραφα. Σημειώνεται ότι η δωρεάν έκδοση περιορίζεται σε 3 διαγράμματα και 60 αντικείμενα ανά διάγραμμα.

3.4 DrawSQL

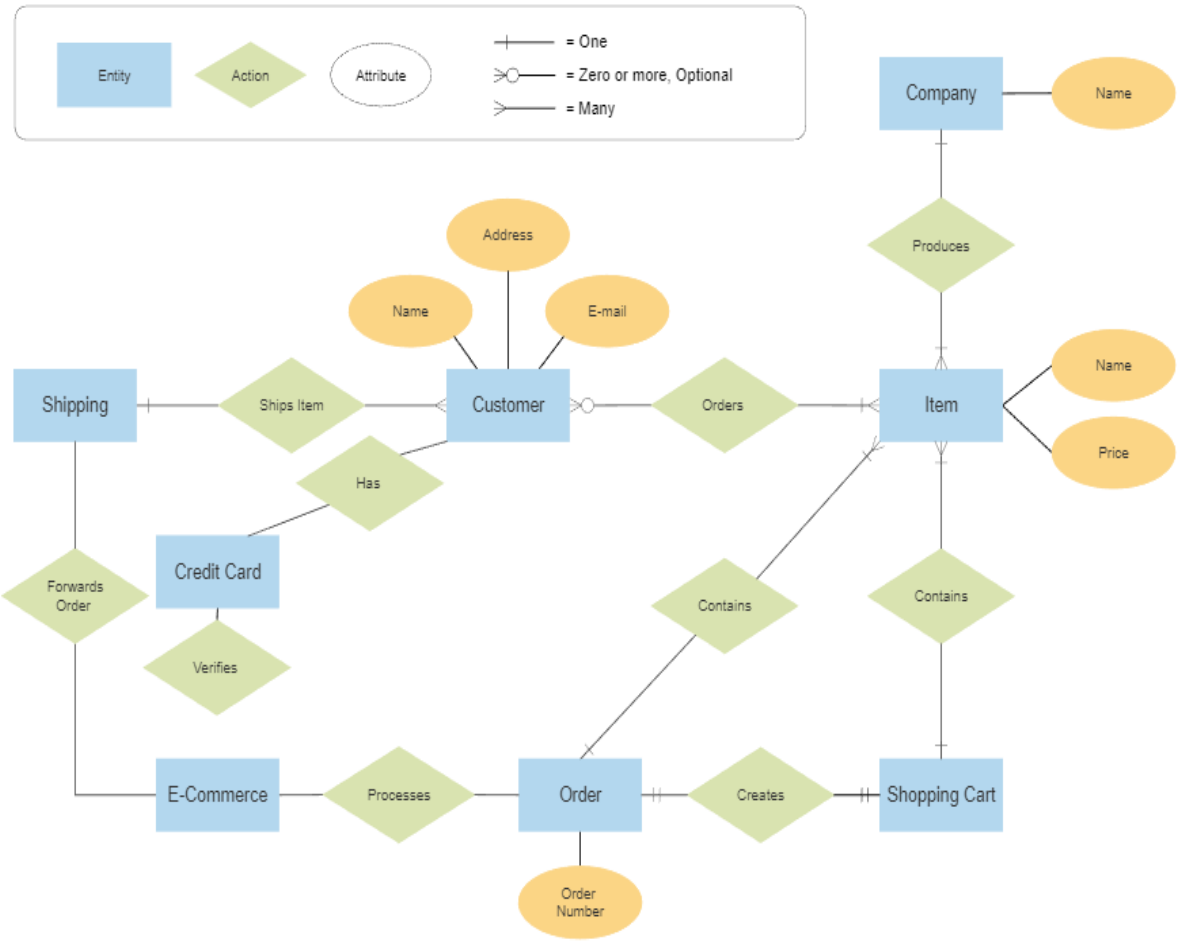


Εικόνα 6: Παράδειγμα διαγράμματος στην εφαρμογή DrawSQL

Το DrawSQL είναι μια διαδικτυακή εφαρμογή που απευθύνεται ειδικά στα διαγράμματα οντοτήτων-συσχετίσεων και διαθέτει μια δωρεάν έκδοση με περιορισμένες δυνατότητες, όπως περιορισμένο αριθμό διαγραμμάτων και αντικειμένων ανά διάγραμμα. Οι χρήστες μπορούν να εισαγάγουν ένα υπάρχον SQL script οποιασδήποτε βάσης δεδομένων θέλουν να οπτικοποιήσουν και να δημιουργήσουν αυτόματα διαγράμματα. Επιπρόσθετα, τα διαγράμματα εξάγονται σε διάφορες μορφές, όπως SQL (DDL) scripts για εκτέλεση στη βάση δεδομένων και σε εικόνες. Όπως και προηγουμένως, επιτρέπεται η συνεργασία μεταξύ των χρηστών στο ίδιο διάγραμμα της βάσης δεδομένων. Υπογραμμίζεται ότι υπάρχουν πάνω από 200 πρότυπα διαγραμμάτων βάσης δεδομένων που οι χρήστες μπορούν να αντλήσουν έμπνευση από και

ακόμη να τα προσαρμόσουν στα διαγράμματά τους. Το DrawSQL υποστηρίζει ορισμένα διαχειριστικά συστήματα βάσεων δεδομένων (DBMS), συμπεριλαμβανομένων των MySQL, PostgreSQL και Microsoft SQL Server.

3.5 SmartDraw

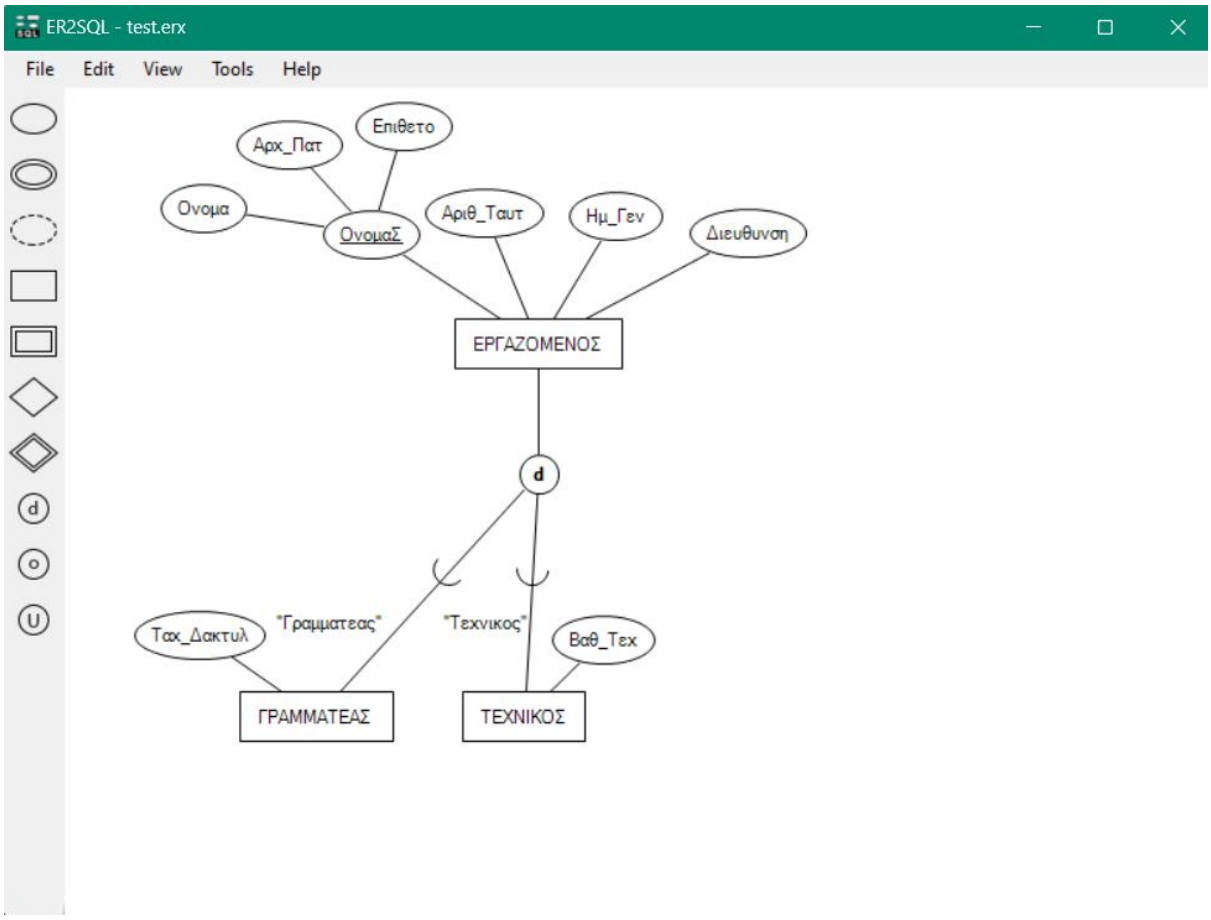


Εικόνα 7: Παράδειγμα διαγράμματος στην SmartDraw

Το SmartDraw είναι ακόμα ένα πρόγραμμα σχεδίασης που παρέχει εργαλεία για τη δημιουργία διαφόρων τύπων διαγραμμάτων, συμπεριλαμβανομένων των διαγραμμάτων οντοτήτων-συσχετίσεων (ER). Το SmartDraw παρέχει στο χρήστη τη δυνατότητα αυτόματης δημιουργίας διαγραμμάτων ER από υπάρχουσες βάσεις δεδομένων ή από πίνακες δεδομένων, ενώ τα

διαγράμματα εξάγονται σε διάφορες μορφές αρχείων, όπως εικόνες, PDF και αρχεία PowerPoint. Το πρόγραμμα προσφέρει μια δοκιμαστική έκδοση για περιορισμένο χρονικό διάστημα χωρίς να απαιτείται εγγραφή.

3.6 ER2SQL



Εικόνα 8: Παράδειγμα διαγράμματος στο ER2SQL

Η εφαρμογή ER2SQL είναι ένα εργαλείο που χρησιμοποιείται για τη δημιουργία και τη μετατροπή διαγραμμάτων οντοτήτων-συσχετίσεων (ER), τόσο απλών όσο και εκτεταμένων, σε κώδικα SQL για τη δημιουργία της αντίστοιχης βάσης δεδομένων. Το εργαλείο επιτρέπει στο χρήστη να ελέγξει την ορθότητα του διαγράμματος που έχει δημιουργήσει και στη συνέχεια να μετατρέψει το εν λόγω διάγραμμα σε σχεσιακό ή αντικειμενοσχεσιακό σχήμα

μέσω SQL κώδικα, παρέχοντας ένα αυτοματοποιημένο μέσο που μεταφράζει τις συνδέσεις και τις οντότητες σε πίνακες, κλειδιά και συναρτήσεις που απαιτούνται για τη δημιουργία της βάσης δεδομένων. Αυτό εξοικονομεί χρόνο και προσπάθεια κατά την ανάπτυξη μιας βάσης δεδομένων. Το εργαλείο διατίθεται δωρεάν και για την εγκατάστασή του απαιτείται λειτουργικό σύστημα Windows με εγκατεστημένο το .NET Framework 4.0 ή νεότερη έκδοση.

3.7 Σύγκριση ERD tools με υπάρχουσες εφαρμογές

Συμπερασματικά, από την παραπάνω παρουσίαση εργαλείων σχεδιασμού διαγραμμάτων οντοτήτων συσχετίσεων, γίνεται φανερή η αναγκαιότητα της δημιουργίας ενός ολοκληρωμένου εργαλείου μοντελοποίησης βάσεων δεδομένων, που θα προσφέρει στο χρήστη μια φιλική και χρήσιμη διεπαφή ώστε να σχεδιάζει και να επεξεργάζεται διαγράμματα ER, να διαχειρίζεται τα αντικείμενα των διαγραμμάτων, να ελέγχει τη συντακτική τους ορθότητα και να εκτελεί μετατροπές σε αντίστοιχα σχεσιακά σχήματα αλλά και κώδικα SQL που να υλοποιεί το σχεσιακό σχήμα σε σχεσιακό σύστημα βάσεων δεδομένων. Σημειώνεται ότι το εργαλείο θα πρέπει να διατίθεται δωρεάν, αξιοποιώντας ανοιχτού κώδικα βιβλιοθήκες και πλατφόρμες που υποστηρίζουν τη δημιουργία τέτοιων εργαλείων. Αυτό επιτρέπει την ανάπτυξή του χωρίς να πρέπει να πληρωθούν άδειες χρήσης ή υπηρεσίες.

Στον επόμενο πίνακα παρουσιάζονται οι ομοιότητες και διαφορές της εφαρμογής ERD tools με τις εφαρμογές που έχουν αναφερθεί σε αυτό το κεφάλαιο.

Εφαρμογή	diagrams	ERDPlus	Lucidchart	DrawSQL	SmartDraw	ER2SQL	**ERD tools**
Εκτεταμένο Μοντέλο					✓	✓	✓
Υποστήριξη κλειδιών	✓					✓	✓
Γραφική υποστήριξη μερικού κλειδιού(με διακεκομμένη υπογράμμιση)	✓						✓
Δυνατότητα μεταβολής μεγέθους γεωμετρικών σχημάτων	✓		✓		✓		✓
Έλεγχος ορθότητας διαγραμμάτων						✓	✓
Μετατροπή σε σχεσιακό σχήμα		✓					✓
Εξαγωγή διαγραμμάτων ως εικόνες png	✓	✓	✓		✓	✓	✓
Εξαγωγή διαγραμμάτων ως svg	✓		✓		✓	✓	✓

Εξαγωγή διαγραμμάτων ως pdf	✓		✓		✓		✓
Μετατροπή σε κώδικα SQL		✓		✓		✓	✓
Δωρεάν χρήση	✓	✓				✓	✓
Συμβατό με διάφορα λειτουργικά συστήματα	✓		✓		✓		✓

4. Ανάπτυξη της εφαρμογής ERD tools

Η ανάπτυξη ενός εργαλείου απαιτεί προσεκτικό σχεδιασμό και υλοποίηση, λαμβάνοντας υπόψη τις συγκεκριμένες απαιτήσεις και προδιαγραφές του έργου. Στο παρόν κεφάλαιο αρχικά παρουσιάζονται οι απαιτήσεις της εφαρμογής, οι οποίες προέκυψαν μέσα από την έρευνα για τα δημοφιλέστερα εργαλεία στο χώρο της δημιουργίας και επεξεργασίας διαγραμμάτων οντοτήτων συσχετίσεων. Στη συνέχεια, ακολουθεί επεξήγηση των κλάσεων του κώδικα που υλοποιεί το εργαλείο, με ιδιαίτερη έμφαση στην κλάση που ελέγχει τη συντακτική ορθότητα ενός διαγράμματος καθώς και στην κλάση που μετατρέπει το διάγραμμα σε σχεσιακό σχήμα ή παράγει τον αντίστοιχο κώδικα SQL που το υλοποιεί.

4.1 Απαιτήσεις της ERD tools

Οι βασικές απαιτήσεις, λειτουργικές και μη, που ενσωματώνονται στην εφαρμογή ERD tools είναι:

Λειτουργικές

1. Σχεδιασμός του Μοντέλου Οντοτήτων Συσχετίσεων (ΜΟΣ): Η εφαρμογή πρέπει να παρέχει γραφικό περιβάλλον όπου οι χρήστες μπορούν να σχεδιάζουν το μοντέλο τους χρησιμοποιώντας τα γραφικά σύμβολα του ΜΟΣ όσο και του Εκτεταμένου ΜΟΣ.
2. Δυνατότητα προσθήκης επιπλέον πληροφοριών για το διάγραμμα: Οι πληροφορίες αυτές αφορούν το σχεσιακό μοντέλο και συνεισφέρουν στην ομαλότερη μετάβαση από το αρχικό διάγραμμα στο σχεσιακό σχήμα.
3. Έλεγχος συντακτικών προβλημάτων: Η εφαρμογή πρέπει να εκτελεί έλεγχο συντακτικής ορθότητας για το μοντέλο που έχει σχεδιαστεί, ελέγχοντας την ορθή χρήση των γραφικών συμβόλων και των συνδέσεων εμφανίζοντας αντίστοιχο μήνυμα λάθους.
4. Μετατροπή σε Σχεσιακό Σχήμα: Η εφαρμογή πρέπει να παρέχει τη δυνατότητα μετατροπής του μοντέλου Οντοτήτων Συσχετίσεων σε Σχεσιακό Σχήμα (ΣΣ), μετατρέποντας τις οντότητες, τα χαρακτηριστικά και τις συσχετίσεις σε πίνακες, στήλες και κλειδιά.

5. Παραγωγή και αποθήκευση κώδικα SQL: Η εφαρμογή πρέπει να μπορεί να παράγει κώδικα SQL (με το πρότυπο SQL - 99) από το Σχεσιακό Σχήμα που έχει δημιουργηθεί, ώστε να είναι δυνατή η υλοποίηση του ΣΣ σε ένα Σχεσιακό Σύστημα Βάσεων Δεδομένων.
6. Αποθήκευση των διαγραμμάτων σε XML μορφή παρέχοντας μια ευέλικτη δομή για την αναπαράσταση δεδομένων με ταυτόχρονη ανεξαρτησία από το περιβάλλον και ευκολία μεταφοράς και ανταλλαγής δεδομένων.
7. Δυνατότητα εξαγωγής σε μορφές εικόνας που απεικονίζει το διάγραμμα που σχεδιάστηκε για χρήση από ποικίλες εφαρμογές γραφείου.
8. Εκτύπωση συμπεριλαμβανομένης και της μετατροπής και αποθήκευσης του αρχείου σε μορφή pdf.

Μη λειτουργικές

1. Πολυπλατφορμική υποστήριξη: Η εφαρμογή πρέπει να είναι συμβατή με διάφορα υπολογιστικά και λειτουργικά συστήματα, παρέχοντας υποστήριξη για την Java Virtual Machine (JVM).
2. Απλό περιβάλλον χρήστη: Η εφαρμογή πρέπει να παρέχει ένα απλό και φιλικό περιβάλλον για τους χρήστες, περιλαμβάνοντας μια ευανάγνωστη διεπαφή.
3. Χρωματισμός των αντικειμένων ενός διαγράμματος: Ο χρωματισμός είναι μια αισθητική πτυχή της εφαρμογής και αφορά την εμφάνιση και τον σχεδιασμό της διεπαφής χρήστη. Βοηθά στην κατανόηση των πληροφοριών, συμβάλλοντας στην ευχρηστία, την προσβασιμότητα και την αισθητική της εφαρμογής.
4. Πληρότητα: Η εφαρμογή πρέπει να υποστηρίζει την αναίρεση και την επανάληψη ενεργειών που έχουν πραγματοποιηθεί από το χρήστη. Αυτό περιλαμβάνει τη δυνατότητα αναίρεσης/επανάληψης πολλαπλών ενεργειών σε σειρά, οι οποίες πρέπει να γίνονται με ακρίβεια, χωρίς να προκαλούν ασυνέπειες ή απώλεια δεδομένων και να εφαρμόζονται σε όλα τα αντικείμενα και τις ιδιότητες που επηρεάζονται από αυτές τις ενέργειες.
5. Προσαρμοσμένες συντομεύσεις πληκτρολογίου: Η εφαρμογή πρέπει να υποστηρίζει τη χρήση των πιο συνηθισμένων συντομεύσεων πληκτρολογίου από τον χρήστη.

6. Απόδοση και απόκριση: Η εφαρμογή πρέπει να είναι αποτελεσματική και αποκρίνεται γρήγορα στις ενέργειες των χρηστών. Αυτό συμπεριλαμβάνει την ελαχιστοποίηση του χρόνου φόρτωσης, την άμεση απόκριση σε εντολές και την αποφυγή καθυστερήσεων ή παρεμβολών.
7. Εκπαιδευτική λειτουργικότητα: Η εφαρμογή πρέπει να μπορεί να χρησιμοποιηθεί ως εργαλείο διδασκαλίας για μαθήματα Βάσεων Δεδομένων, παρέχοντας τη δυνατότητα κατανόησης και εκμάθησης του σχεδιασμού βάσεων δεδομένων με το ΜΟΣ και το ΣΣ.

Έπειτα από μελέτη του απλού, αλλά και του εκτεταμένου μοντέλου οντοτήτων συσχετίσεων ξεκίνησε η ανάπτυξη του κώδικα που υλοποιεί το εργαλείο στη γλώσσα Java. Η Java είναι μια πολύ δημοφιλής γλώσσα προγραμματισμού με πληθώρα πλεονεκτημάτων (Schildt, 2019). Αρχικά, είναι γνωστή για την πλατφόρμα ανεξαρτησίας της, καθώς ο κώδικας Java μπορεί να εκτελεστεί σε οποιοδήποτε σύστημα που έχει εγκατεστημένη την Java Virtual Machine (JVM) (Evans & Warburton, 2014). Αυτό σημαίνει ότι ο κώδικας Java μπορεί να τρέξει σε διάφορα λειτουργικά συστήματα, χωρίς να απαιτείται μεταγλώττιση για κάθε σύστημα ξεχωριστά (Deitel & Deitel, 2011). Επιπλέον, παρέχει ένα ευρύ φάσμα πλούσιων βιβλιοθηκών που καλύπτουν πολλούς τομείς, όπως η γραφική διεπαφή χρήστη, οι βάσεις δεδομένων και πολλά άλλα. Αυτές οι βιβλιοθήκες επιτρέπουν την αποτελεσματική ανάπτυξη προγραμμάτων επεκτείνοντας τις δυνατότητες της γλώσσας. (Evans & Flanagan, 2018). Το εργαλείο που αναπτύχθηκε βασίστηκε στη βιβλιοθήκη JGraph και αξιοποίησε το παράδειγμα που παρέχεται σαν πρότυπο, ως πηγή έμπνευσης για τον σχεδιασμό και την υλοποίησή του. Ακολουθούν μερικές πληροφορίες για τη βιβλιοθήκη.

4.2 Η βιβλιοθήκη jGraphX

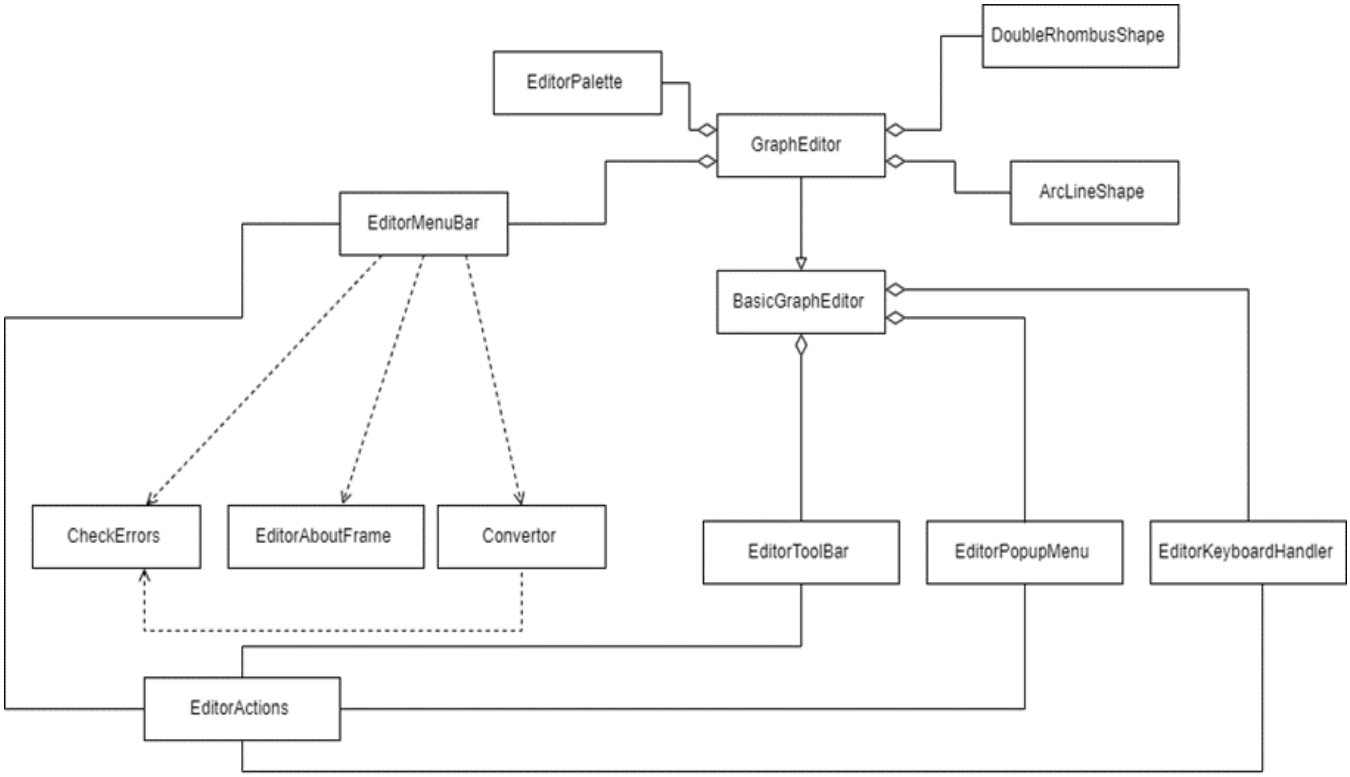
Η jGraphX είναι μια βιβλιοθήκη Java ανοιχτού κώδικα που παρέχει ένα σύνολο κλάσεων και συναρτήσεων για τη δημιουργία, την επεξεργασία και την απόδοση διαδραστικών διαγραμμάτων και γραφημάτων. Είναι βασισμένη στη βιβλιοθήκη Swing και χρησιμοποιεί την αρχιτεκτονική Model-View-Controller (MVC) για να διαχωρίσει την οπτική αναπαράσταση του γραφήματος από το υποκείμενο μοντέλο δεδομένων.

Με τη jGraphX οι προγραμματιστές μπορούν να δημιουργήσουν ένα ευρύ φάσμα εφαρμογών που βασίζονται σε γραφήματα, όπως διαγράμματα ροής, διαγράμματα UML, διαγράμματα δικτύου και άλλα. Η βιβλιοθήκη παρέχει μια σειρά από ενσωματωμένες διατάξεις γραφημάτων, συμπεριλαμβανομένων ιεραρχικών, οργανικών, κυκλικών και ορθογώνιων διατάξεων, και υποστηρίζει επίσης προσαρμοσμένες διατάξεις. Το JGraphX παρέχει λειτουργικότητα για οπτικοποίηση και αλληλεπίδραση με γραφήματα ακμής κόμβου (node-edge). Μερικά από τα βασικά χαρακτηριστικά της jGraphX περιλαμβάνουν:

- Προσαρμόσιμα στυλ και θέματα για κόμβους, άκρες και ετικέτες.
- Δυνατότητα προσθήκης εικόνων, εικονιδίων και συμβουλών εργαλείων σε κόμβους και άκρες.
- Ενσωματωμένη υποστήριξη για λειτουργίες αναίρεσης και επανάληψης.
- Ενσωμάτωση με διάφορες μορφές αρχείων, όπως XML, SVG, PNG και PDF.
- Υποστήριξη για διαδραστικές λειτουργίες όπως μετατόπιση, ζουμ και επιλογή.

Η jGraphX κυκλοφορεί με την άδεια BSD και συντηρείται ενεργά από μια ομάδα προγραμματιστών. Έχει μια μεγάλη κοινότητα χρηστών και εκτεταμένη τεκμηρίωση καθώς και παραδείγματα διαθέσιμα για να βοηθήσουν τους προγραμματιστές να ξεκινήσουν να τη χρησιμοποιούν.

4.3 Διάγραμμα κλάσεων



Εικόνα 9: Διάγραμμα κλάσεων

Κύριες κλάσεις και λειτουργίες στο γραφικό περιβάλλον ERD tools:

•GraphEditor: στοιχείο γραφικού περιβάλλοντος χρήστη (GUI) που επεκτείνει την κλάση BasicGraphEditor και δημιουργεί το μενού με την παλέτα σχημάτων και τα αντίστοιχα tooltips αυτών.

```
// Adds some template cells for dropping into the graph
shapesPalette
        .addTemplate(
            name:"Attribute",
            new ImageIcon(
                    GraphEditor.class
                            .getResource(name:"/com/mxgraph/thesis/swing/images/1.png")),
                            style:"ellipse", width:160, height:60, value:"Attribute");
```

Εικόνα 10: Προσθήκη του αντικειμένου που αναπαριστά ένα γνώρισμα στο μενού των σχημάτων

•ArcLineShape: δημιουργεί το αντικείμενο των συνδέσεων μεταξύ των σχημάτων, λαμβάνοντας υπόψιν τις περιπτώσεις υπερκλάσης/υποκλάσης

•DoubleRhombusShape: δημιουργία σχήματος προσδιορίζουσας συσχέτισης. Η κλάση αυτή επεκτείνει την κλάση mxRhombusShape της βιβλιοθήκης Jgraph και όταν καλείται δημιουργεί ένα ορθογώνιο με διπλό περίγραμμα.

```
public class DoubleRhombusShape extends mxRhombusShape
{
    /**
     *
     */
    public void paintShape(mxGraphics2DCanvas canvas, mxCellState state)
    {
        super.paintShape(canvas, state);

        int inset = (int) Math.round((mxUtils.getFloat(state.getStyle(),
                mxConstants.STYLE_STROKEWIDTH, defaultValue:1) + 4)
                * canvas.getScale());

        Rectangle rect = state.getRectangle();
        int x = rect.x + inset;
        int y = rect.y + inset;
        int w = rect.width - 2 * inset;
        int h = rect.height - 2 * inset;
        int halfWidth = w / 2;
        int halfHeight = h / 2;

        int[] xPoints = {x + halfWidth, x + w,x + halfWidth,x};
        int[] yPoints= {y,y + halfHeight,  y + h,  y + halfHeight};
        canvas.getGraphics().drawPolygon(  xPoints,  yPoints, xPoints.length);

    }
}
```

Εικόνα 11: Δημιουργία σχήματος προσδιορίζουσας συσχέτισης

•BasicGraphEditor: ένα στοιχείο GUI που βασίζεται σε Swing και παρέχει ένα οπτικό πρόγραμμα επεξεργασίας για τη δημιουργία και την επεξεργασία γραφημάτων. Είναι υπεύθυνη για τη δημιουργία και διαχείριση των βασικών στοιχείων που απαιτούνται για την επεξεργασία γραφημάτων, όπως η αρχικοποίηση του γραφήματος, η δημιουργία της γραφικής διεπαφής, η προσθήκη εργαλείων και μενού, καθώς και η διαχείριση των γεγονότων που σχετίζονται με την επεξεργασία του γραφήματος. Στην εικόνα που ακολουθεί, φαίνεται μέρος της μεθόδου ConnectShapes, η οποία δημιουργεί τη σύνδεση μεταξύ 2 αντικειμένων.

```
public void ConnectShapes(mxCell target) {

    final mxGraph graph = graphComponent.getGraph();
    Object parent = graph.getDefaultParent();
    if(connect==1 ||connect==1.1) {
        graph.insertEdge(parent, id:null, value:"", source, target);
    }

    //subclass total
    if(connect==1.2) {
        graph.insertEdge(parent, id:null, value:"", source, target, style:"strokeWidth=5;");
    }

    //superclass partial
    if(connect==1.3) {
        graph.insertEdge(parent, id:null, value:"", source, target, style:"startArrow=open");
    }
```

Εικόνα 12: Η ConnectShapes

•EditorAboutFrame: εμφανίζει νέο παράθυρο με γενικές πληροφορίες σχετικά με το εργαλείο

•EditorActions: συνδέει το κουμπί κάθε μενού του οριζόντιου παραθύρου της διεπαφής με την επιθυμητή λειτουργία, παρέχοντας ένα σύνολο από λειτουργίες που σχετίζονται με την επεξεργασία γραφήματος. Αυτές οι λειτουργίες είναι χρήσιμες για τη διαχείριση των ενεργειών που σχετίζονται με τη δημιουργία, τροποποίηση και απόδοση γραφήματος στο πλαίσιο του επεξεργαστή γραφήματος.

•EditorKeyboardHandler: Αναλαμβάνει τη διαχείριση των συντομεύσεων πληκτρολογίου στο πλαίσιο επεξεργασίας γραφήματος. Ο ρόλος της είναι να επεξεργαστεί τα συμβάντα πληκτρολογίου που λαμβάνονται και να εκτελέσει τις αντίστοιχες ενέργειες στο γράφημα, τα οποία περιλαμβάνουν το άνοιγμα υπάρχοντος αρχείου, τη δημιουργία νέου αρχείου, την αποθήκευσή του, την αναίρεση ή επανάληψη ενός συμβάντος.

```
protected InputMap getInputMap(int condition)
{
    InputMap map = super.getInputMap(condition);

    if (condition == JComponent.WHEN_FOCUSED && map != null)
    {
        map.put(KeyStroke.getKeyStroke(s:"control S"), actionMapKey:"save");
        map.put(KeyStroke.getKeyStroke(s:"control shift S"), actionMapKey:"saveAs");
        map.put(KeyStroke.getKeyStroke(s:"control N"), actionMapKey:"new");
        map.put(KeyStroke.getKeyStroke(s:"control O"), actionMapKey:"open");

        map.put(KeyStroke.getKeyStroke(s:"control Z"), actionMapKey:"undo");
        map.put(KeyStroke.getKeyStroke(s:"control Y"), actionMapKey:"redo");
        map
            .put(KeyStroke.getKeyStroke(s:"control shift V"),
                actionMapKey:"selectVertices");
        map.put(KeyStroke.getKeyStroke(s:"control shift E"), actionMapKey:"selectEdges");
    }

    return map;
}
```

Εικόνα 13: Συντομεύσεις πληκτρολογίου

•EditorMenuBar: δημιουργεί τη μπάρα μενού του πλαισίου επεξεργασίας γραφήματος. Αποτελεί μια γραφική διεπαφή που παρέχει προσβάσιμες ενέργειες που σχετίζονται τόσο με τη διαχείριση όσο και την επεξεργασία ενός γραφήματος. Ορισμένες από τις κύριες λειτουργίες που παρέχονται από την EditorMenuBar περιλαμβάνουν τη δημιουργία και προσθήκη μενού στη μπάρα μενού, τη σύνδεση ενεργειών μενού με τις αντίστοιχες λειτουργίες στο γράφημα, την ενημέρωση της κατάστασης των μενού ανάλογα με την επιλεγμένη κατάσταση του γραφήματος (π.χ. επιλεγμένα αντικείμενα, επεξεργασία κειμένου κλπ.).

```
// Creates the options menu
menu = add(new JMenu(s:"Options"));
item = menu.add(new JMenuItem(text:"Check for errors"));
item.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        new CheckErrors(graphComponent,graph);

    }
});
item = menu.add(new JMenuItem(text:"Convert to Relational schema"));
item.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        new Convertor(graphComponent,graph,convertToSQL:false);

    }
});

item = menu.add(new JMenuItem(text:"Convert to SQL"));
item.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        new Convertor(graphComponent,graph,convertToSQL:true);

    }
});
```

Εικόνα 14: Το μενού Options

•EditorPalette: χρησιμοποιείται για τη δημιουργία μιας συλλογής επαναχρησιμοποιήσιμων γραφικών στοιχείων που εμφανίζονται σε μια παλέτα στη διεπαφή. Οι χρήστες μπορούν να επιλέξουν ένα σχήμα από την παλέτα και να το εισαγάγουν στο γράφημά τους μέσω της λειτουργίας drag and drop.

•EditorPopupMenu: δημιουργεί το αναδυόμενο μενού για κάθε τύπο αντικειμένου. Οταν ο χρήστης κάνει δεξί κλικ σε ένα στοιχείο του γραφήματος, η EditorPopupMenu εμφανίζει ένα μενού με διάφορες επιλογές που αφορούν το επιλεγμένο στοιχείο. Οι επιλογές μπορεί να περιλαμβάνουν ενέργειες όπως τη μετονομασία, την επεξεργασία των ιδιοτήτων, τη σύνδεση με άλλο αντικείμενο του γραφήματος και εξαρτώνται από το είδος του επιλεγμένου στοιχείου.

```
if(cellStyle.contains(s:"ellipse") || cellStyle.contains(s:"doubleEl") ||cellStyle.contains(s:"primary")
   ||cellStyle.contains(s:"unique")||cellStyle.contains(s:"partial")
   ||cellStyle.contains(s:"un_dl")||  cellStyle.contains(s:"doubleEl")){

    addSeparator();
    JMenu dTypes = new JMenu(s:"Data Types");
    JRadioButton  jrb1 = new JRadioButton(text:"BOOLEAN");
    JRadioButton  jrb2 = new JRadioButton(text:"SMALLINT");
    JRadioButton  jrb3 = new JRadioButton(text:"INTEGER");
    JRadioButton  jrb4 = new JRadioButton(text:"BIGINT");
    JRadioButton  jrb5 = new JRadioButton(text:"REAL");
    JRadioButton  jrb6 = new JRadioButton(text:"DOUBLE PRECISION");
    JRadioButton  jrb7 = new JRadioButton(text:"DECIMAL(p,s)");
    JRadioButton  jrb8 = new JRadioButton(text:"CHAR(n)");
    JRadioButton  jrb9 = new JRadioButton(text:"VARCHAR(n)");
    JRadioButton  jrb10 = new JRadioButton(text:"BIT(n)");
    JRadioButton  jrb11 = new JRadioButton(text:"BIT VARYING(n)");
    JRadioButton  jrb12 = new JRadioButton(text:"DATE");
    JRadioButton  jrb13 = new JRadioButton(text:"TIME");
    JRadioButton  jrb14 = new JRadioButton(text:"TIMESTAMP");
    JRadioButton  jrb15 = new JRadioButton(text:"INTERVAL");
    JRadioButton  jrb16 = new JRadioButton(text:"XML");
    JRadioButton  jrb17 = new JRadioButton(text:"OTHER (add type)");
```

Εικόνα 15: Μέρος του popupmenu που αφορά τον τύπο δεδομένων Data Types ενός αντικειμένου

•EditorToolBar: αναπαριστά μία εργαλειοθήκη εικονιδίων για το γράφημα. Χρησιμοποιείται για την προβολή ενός συνόλου εργαλείων και επιλογών που είναι διαθέσιμα για τον χρήστη και μπορεί να τα χρησιμοποιήσει κατά τη δημιουργία ή την επεξεργασία ενός γραφήματος. Οι επιλογές στην εργαλειοθήκη είναι συνδεδεμένες με συγκεκριμένες ενέργειες ή λειτουργίες που μπορούν να εκτελεστούν στο γράφημα. Όταν ο χρήστης επιλέγει ένα εργαλείο από την εργαλειοθήκη, μπορεί να εκτελέσει τη σχετική ενέργεια στο γράφημα, όπως το άνοιγμα υπάρχοντος αρχείου, τη δημιουργία νέου αρχείου, την αποθήκευσή του, την αναίρεση ή επανάληψη ενός συμβάντος αλλά και την αλλαγή του μεγέθους της γραμματοσειράς ενός αντικειμένου, καθιστώντας τα διαγράμματα πλήρως παραμετροποιήσιμα.

•CheckErrors: ελέγχει τη συντακτική ορθότητα του διαγράμματος και εμφανίζει αντίστοιχο παράθυρο όπως περιγράφηκε σε προηγούμενο κεφάλαιο. Θα αναλυθεί λεπτομερώς παρακάτω.

•Convertor: δημιουργεί το Σχεσιακού Σχήμα και παράγει και αποθηκεύει τον κώδικα SQL που υλοποιεί το Σχεσιακό Σχήμα. Και αυτή θα αναλυθεί λεπτομερώς στη συνέχεια.

Μέρος της βιβλιοθήκης jGraphX τροποποιήθηκε για να προστεθούν επιπλέον δυνατότητες, όπως η δυνατότητα δημιουργίας του σχήματος που αναπαριστά το μερικό κλειδί μιας ασθενούς οντότητας και η εξαγωγή των αντικειμένων του διαγράμματος σε μορφή εικόνας svg.

4.3.1 Η κλάση CheckErrors

Όταν ο χρήστης επιλέξει το αντίστοιχο υπομενού για τον έλεγχο σφαλμάτων του διαγράμματος καλείται ο κατασκευαστής της κλάσης CheckErrors όπου δημιουργείται ένας πίνακας με στήλες το όνομα, τον τύπο και την περιγραφή του λάθους που αντιστοιχεί στο αντικείμενο. Πρώτα, διατρέχεται το διάγραμμα και για κάθε αντικείμενο γίνονται οι απαραίτητοι έλεγχοι.

```
public CheckErrors(mxGraphComponent graphComponent, mxGraph graph) {

    frame = new JFrame(title:"List of Errors");
    model = new DefaultTableModel();
    table = new JTable(model);

    // Create a couple of columns
    model.addColumn(columnName:"Type");
    model.addColumn(columnName:"Name");
    model.addColumn(columnName:"Description");

    table.setPreferredScrollableViewportSize(new Dimension(width:600, height:500));
    table.setFillsViewportHeight(fillsViewportHeight:true);
    //Set up column sizes.
    initColumnSizes(table);
```

Εικόνα 16: Ο κατασκευαστής της κλάσης CheckErrors

Σε περίπτωση που εντοπιστεί κάποιο λάθος προστίθεται νέα γραμμή στον πίνακα με τα λάθη, όπως φαίνεται στην εικόνα που ακολουθεί.

```
if (hasWeakS==true){
    AddRow(str1:"Weak entity",ownerCName,str3:"Weak entities cannot be superclasses");
}
```

Εικόνα 17: Η περίπτωση όπου εντοπίστηκε ασθενής οντότητα ως υπερκλάση

Στο τέλος καλείται η μέθοδος countRows(), η οποία ελέγχει εάν ο πίνακας έχει γραμμές χωρίς να προσμετράμε την 1η γραμμή με τους τίτλους των στηλών. Στην περίπτωση που έχουν

προστεθεί γραμμές στον πίνακα η μεταβλητή hasErrors τίθεται false και εμφανίζεται νέο παράθυρο με τα κατάλληλα μηνύματα λάθους που έχουν αναφερθεί στο κεφάλαιο 3.

```
boolean countRows(){
  int rows =  table.getRowCount();
  if (rows>1){
    hasErrors = true;
    frame.setVisible(b:true);
  }
  return hasErrors;
}
```

Εικόνα 18: Η μέθοδος countRows()

Στην αντίθετη περίπτωση, εμφανίζεται το μήνυμα “No errors found” για να ενημερώσει το χρήστη για την ορθότητα τους διαγραμμάτος.

4.3.2 Η κλάση Convertor

Με αντίστοιχη επιλογή του χρήστη από το μενού δημιουργείται το Σχεσιακό Σχήμα ή ο κώδικας SQL που υλοποιεί το Σχεσιακό Σχήμα, αφού πρώτα γίνει αυτοματοποιημένος έλεγχος για σφάλματα του διαγράμματος. Σε περίπτωση μη εύρεσης λαθών ξεκινάει η μετατροπή του ER διαγράμματος σε ΣΣ.

```
public class Convertor extends JFrame {

    ArrayList<Table> tables = new ArrayList<>();

    public Convertor(mxGraphComponent graphComponent, mxGraph graph, boolean convertToSQL ) {

        if (new CheckErrors(graphComponent,graph).hasErrors == false) {
            mxAnalysisGraph aGraph = new mxAnalysisGraph();
            aGraph.setGraph(graph);

            //for every cell
            Object[] cells = graph.getChildVertices(graph.getDefaultParent());
            for (Object c : cells) {
                mxCell cell = (mxCell) c;
                 //get every cell's value
                String getName =  (String) cell.getValue();
                 //get every cell's style
                String cellStyle = cell.getStyle();
```

Εικόνα 19: Η κλάση Convertor

Η διαδικασία που ακολουθήθηκε παρουσιάζει κάποιες διαφορές από τον αλγόριθμο που παρουσίασαν οι Elmasri & Navathe, ώστε να μην δημιουργούνται προβλήματα όπως για παράδειγμα στην περίπτωση απεικόνισης ασθενών τύπων οντοτήτων, καθώς γίνεται αναφορά στο πρωτεύον κλειδί του ιδιοκτήτη, ο οποίος όμως μπορεί να συμμετέχει σε μια εξειδίκευση/ένωση. Ακολουθεί η τροποποιημένη αλληλουχία βημάτων:

1. Απεικόνιση ισχυρών τύπων οντοτήτων: Για κάθε ισχυρό τύπο οντοτήτων δημιουργείται ένας πίνακας με το αντίστοιχο όνομα, που περιλαμβάνει τα απλά του γνωρίσματα. Σε περίπτωση που σύνθετα γνωρίσματα ανήκουν στην οντότητα, συμπεριλαμβάνουμε μόνο τα απλά γνωρίσματα από τα οποία αποτελούνται. Το πρωτεύον κλειδί, το οποίο επίσης προστίθεται στον πίνακα απεικονίζεται υπογραμμισμένο στην περίπτωση του σχεσιακού σχήματος, ενώ για τον κώδικα SQL αφού δημιουργηθούν οι κατάλληλες δηλώσεις για τον πίνακα και τα γνωρίσματά του, ακολουθεί η εντολή PRIMARY KEY () όπως στην εικόνα που ακολουθεί.

2.

```
CREATE TABLE CUSTOMER (
    FirstName   INTEGER,
    LastName    INTEGER,
    email   INTEGER,
    CustomerID  INTEGER NOT NULL,
PRIMARY KEY (CustomerID) );
```

Εικόνα 20: Ο κώδικας SQL για τη δημιουργία του πίνακα Πελάτης

Η αναζήτηση του πρωτεύοντος κλειδιού ενός αντικειμένου γίνεται καλώντας τη μέθοδο PrimaryKey(). Δεχόμενη ως όρισμα το αντικείμενο, δημιουργεί μια νέα λίστα (ArrayList) με το όνομα "pKey" για να αποθηκεύσει τα κελιά που αποτελούν το πρωτεύον κλειδί και αρχικοποιεί μια μεταβλητή με την τιμή false με το όνομα "hasPrimary", η οποία υποδεικνύει εάν βρέθηκε πρωτεύον κλειδί. Στη συνέχεια, ανακτά τις ακμές που εξέρχονται από το αντικείμενο και αν βρεθεί ιδιοκτήτης, τότε αναδρομικά καλείται η ίδια μέθοδος για τον ιδιοκτήτη αυτού του αντικειμένου, προκειμένου να εντοπίσει το πρωτεύον κλειδί του.
Στην αντίθετη περίπτωση, δηλαδή εάν δεν υπάρχει ιδιοκτήτης, ελέγχονται οι ακμές που εξέρχονται από το αντικείμενο για να εντοπιστεί το πρωτεύον κλειδί. Σε περίπτωση που το γνώρισμα που αντιστοιχεί στο κλειδί είναι σύνθετο, προστίθενται μόνο τα απλά γνωρίσματα στη λίστα "pKey". Τέλος, επιστρέφεται η λίστα pKey που περιέχει τα γνωρίσματα που αποτελούν το πρωτεύον κλειδί.

3. Απεικόνιση εξειδικεύσεων/γενικεύσεων: Η υπερκλάση της εξειδίκευσης αντιμετωπίζεται όπως προηγουμένως. Όσον αφορά τις υποκλάσεις που συμμετέχουν σε αυτή, δημιουργείται νέα σχέση-πίνακας για καθεμία από αυτές που περιέχει τα γνωρίσματά τους και ως πρωτεύον κλειδί ορίζεται το κλειδί της υπερκλάσης. Η πληροφορία σχετικά με το κλειδί που κληρονομήθηκε αποθηκεύεται για να αξιοποιηθεί κατάλληλα κατά τη μετατροπή, ενώ δηλώνουμε τον περιορισμό ON DELETE CASCADE ON UPDATE CASCADE για τον κώδικα SQL .

```
Foreign f = new Foreign();
f.List.addAll(table.primaryKey);
//refer to the superclass
f.refersTo = ownerC;
//referential integrity constraint
f.constraint = "ON DELETE CASCADE ON UPDATE CASCADE" ;
table.foreignKey.add(f);
```

Εικόνα 21: Κώδικας για το κλειδί που κληρονομήθηκε

4. Απεικόνιση ενώσεων: Σε κάθε υπερκλάση που συμμετέχει στην ένωση δημιουργούμε μια αναφορά ξένου κλειδιού προς το πρωτεύον κλειδί της οντότητας που αποτελεί την υποκλάση της ένωσης, δηλώνουμε τον περιορισμό ON DELETE SET NULL ON UPDATE CASCADE, και εισάγουμε στους αντίστοιχους πίνακες των υπερκλάσεων τα γνωρίσματα που αποτελούν το ξένο κλειδί. Όσον αφορά την υποκλάση, δηλαδή το αποτέλεσμα της ένωσης, θα πρέπει να έχουμε ορίσει κάποιο κλειδί, γνωστό ως αναπληρωματικό κλειδί, καθώς οι ορίζουσες κλάσεις έχουν διαφορετικά κλειδιά, οπότε δεν είναι εφικτό να χρησιμοποιηθεί ένα από αυτά ως μοναδικό αναγνωριστικό για την ταυτοποίηση των εγγραφών της υποκλάσης.

```
for (mxCell object : PK) {
    mxCell fk = new mxCell();
    fk.setNotNull(boo:false);
    String pkv = (String) object.getValue();
    fk.setValue("FK"+ (table.foreignKey.size() + 1) + "_"+ pkv);
    fk.setDataType(object.getDataType());
    f.List.add(fk);
}
f.refersTo = unionSub;
f.constraint =  "ON DELETE SET NULL ON UPDATE CASCADE" ;
table.attributes.addAll(f.List);
table.foreignKey.add(f);
}
```

Εικόνα 22: Κώδικας για τις υπερκλάσεις της ένωσης

5. Απεικόνιση ασθενών τύπων οντοτήτων: Για κάθε ασθενή τύπο οντοτήτων δημιουργείται ένας πίνακας με το αντίστοιχο όνομα, του οποίου οι στήλες είναι τα απλά

του γνωρίσματα. Επιπλέον, τα πρωτεύοντα κλειδιά των ιδιοκτητών οντοτήτων προστίθενται ως επιπλέον γνωρίσματα - ξένα κλειδιά στον πίνακα. Το πρωτεύον κλειδί ορίζεται ως ο συνδυασμός των ξένων κλειδιών και του μερικού κλειδιού της ασθενούς οντότητας. Εδώ δηλώνεται ο περιορισμός ON DELETE CASCADE ON UPDATE CASCADE.

6. Απεικόνιση δυαδικών 1:1 τύπων συσχετίσεων και δυαδικών 1:N τύπων συσχετίσεων: Χρησιμοποιώντας την προσέγγιση του ξένου κλειδιού, επιλέγεται ένας από τους 2 πίνακες που συμμετέχουν στη συσχέτιση, έχοντας ως προτίμηση την οντότητα με την ολική συμμετοχή. Στην περίπτωση της1:N επιλέγεται ο πίνακας που αντιστοιχεί στην οντότητα που βρίσκεται στην N πλευρά. Σε αυτόν τον πίνακα προστίθενται τα τυχόν γνωρίσματα που ανήκουν στη συσχέτιση καθώς και το πρωτεύον κλειδί της άλλης οντότητας ως ξένο κλειδί.

```
for(mxCell pri:table1.primaryKey) {
  mxCell newpk = new mxCell();
  newpk.setNotNull(totalOrN);
  newpk.setValue("FK"+ counter1 + "_" +pri.getValue());
  newpk.setDataType(pri.getDataType());
  newpk.setStyle(pri.getStyle());
  f.List.add(newpk);
}
f.refersTo = entity1;
f.constraint =  "ON DELETE SET NULL ON UPDATE CASCADE" ;
table.attributes.addAll(f.List);
table.foreignKey.add(f);
}
```

Εικόνα 23: Κώδικας για τους δυαδικούς τύπους συσχετίσεων 1:1 και 1:N

7. Απεικόνιση δυαδικών M:N τύπων συσχετίσεων και Απεικόνιση n-αδικών τύπων συσχετίσεων: Δημιουργείται ένας πίνακας με το όνομα της συσχέτισης με αυτόματο έλεγχο σε περίπτωση που έχει ήδη οριστεί πίνακας με ίδιο όνομα, ώστε να γίνει κατάλληλη μετονομασία. Σε αυτόν τον πίνακα προστίθενται τα τυχόν γνωρίσματα που ανήκουν στη συσχέτιση και ορίζεται ως πρωτεύον κλειδί το σύνολο των πρωτευόντων

κλειδιών των οντοτήτων που συμμετέχουν στη συσχέτιση, τα οποία και προστίθενται ως ξένα κλειδιά. Στην περίπτωση n-αδικού τύπου συσχέτισης, η οντότητα με λόγο πληθικότητας 1 δε συμμετέχει στο πρωτεύον κλειδί αυτής. Ακόμη, έχει ληφθεί υπόψιν η περίπτωση n-αδικού τύπου με όλους τους λόγους πληθικότητας 1, όπου κληρονομείται το κλειδί μίας εκ των οντοτήτων.

```
//if no primary keys found
if ( table.primaryKey.size() == 0) {
  Boolean b = true;
  for (mxCell l : list){
    if(b) {
      table.primaryKey.add(l);
      b=false;
    }
    else {
      table.Unique.add(l);
    }
  }
}
```

Εικόνα 24: Κώδικας για τους δυαδικούς τύπους συσχετίσεων M:N και τους n-αδικούς τύπους

8. Απεικόνιση πλειότιμων γνωρισμάτων: Για κάθε πλειότιμο γνώρισμα δημιουργείται ένας πίνακας με το αντίστοιχο όνομα, στον οποίο προστίθεται ως γνώρισμα το πλειότιμο γνώρισμα και όλα τα απλά γνωρίσματα που ενδεχομένως έχει. Ορίζεται ως ξένο κλειδί το πρωτεύον κλειδί του ιδιοκτήτη του εν λόγω γνωρίσματος, το οποίο σε συνδυασμό με το ίδιο το γνώρισμα συνθέτουν το πρωτεύον κλειδί του νέου πίνακα. Και εδώ δηλώνεται ο περιορισμός ON DELETE CASCADE ON UPDATE CASCADE.

```
public class Table {
    public String name;
    public mxCell cell;
    public String Inherits = "";
    public ArrayList<mxCell> attributes;
    public ArrayList<mxCell> primaryKey;
    public ArrayList<mxCell> Unique;
    public ArrayList<Foreign> foreignKey;

    public Table(String name, mxCell cell) {
        this.name = name;
        this.cell = cell;
        attributes = new ArrayList<mxCell>();
        primaryKey = new ArrayList<mxCell>();
        Unique = new ArrayList<mxCell>();
        foreignKey = new ArrayList<Foreign>();
    }
}

public class Foreign{
        public ArrayList<mxCell> List = new ArrayList<mxCell>();
        public mxCell refersTo = null;
        public String constraint = "";
}

public Table getTableByCell(mxCell cell, ArrayList<Table> tables) {
    for (Table table : tables) {
        if (table.cell == cell) {
            return table;
        }
    }
    return null;
}
```

Εικόνα 25: Πίνακας βάσης δεδομένων και ξένο κλειδί

Κατά την παραγωγή του Κώδικα SQL, δημιουργούνται πρώτα οι πίνακες που έχουν προκύψει από τη μετατροπή του διαγράμματος σύμφωνα με την προαναφερθείσα διαδικασία και στη συνέχεια δηλώνονται τα ξένα κλειδιά που τους αντιστοιχούν. Αυτό γίνεται με την προσθήκη κατάλληλων δηλώσεων ξένων κλειδιών στον κώδικα SQL, οι οποίες αντιπροσωπεύουν τις συσχετίσεις μεταξύ των πινάκων και δηλώνουν ποια

γνωρίσματα αποτελούν ξένα κλειδιά. Η σειρά αυτή είναι απαραίτητη καθώς δεν μπορούμε να αναφερθούμε σε πίνακες που δεν έχουν δημιουργηθεί ακόμα όταν δηλώνουμε ξένα κλειδιά, και έτσι εξασφαλίζεται ότι οι απαραίτητοι πίνακες υπάρχουν πριν δηλωθούν οι συσχετίσεις με αυτά.

```
// makes the sql code
if(convertToSQL==true) {

  String sqlString = "";
  for (Table t : tables){
   // sqlString += table.toString();
    StringBuilder sb = new StringBuilder();
    sb.append("CREATE TABLE " + t.name).append(str:" (\n");

    for (mxCell attribute : t.attributes) {
      sb.append(str:"\t").append((String) attribute.getValue()+ "\t" + attribute.getDataType() +
      (attribute.getNotNull()==true ? ("\t"+"NOT NULL") : "") ).append(str:",\n");
    }
    if(t.primaryKey.size()>0){
      sb.append(str:"PRIMARY KEY (");
      for (int i = 0; i < t.primaryKey.size(); i++) {
        String val = (String) t.primaryKey.get(i).getValue();
        sb.append(val);
        if (i < t.primaryKey.size() - 1) {
          sb.append(str:", ");
        }
      }
     sb.append(str:"),\n");
    }

    for (mxCell uk : t.Unique) {
     sb.append(str:"UNIQUE(").append((String)uk.getValue()).append(str:"),\n");
    }

   sb.delete(sb.length() - 2, sb.length());
   sb.append(" "+ ");\n"+ "\n");
   sqlString += sb.toString();
  }
```

Εικόνα 26: Παραγωγή του κώδικα που υλοποιεί τη σχεσιακή βάση

Αρχικά, υπάρχει ένας έλεγχος if(convertToSQL==true) για να επιβεβαιωθεί ότι πρέπει να γίνει η μετατροπή σε SQL κώδικα. Έπειτα, υπάρχει ένας βρόχος (for) που διατρέχει τους πίνακες (tables) που έχουν δημιουργηθεί από το διάγραμμα. Για κάθε πίνακα, δημιουργείται ένα αντικείμενο StringBuilder (sb), το οποίο χρησιμοποιείται για να κατασκευαστεί ο SQL κώδικας. Αυτός ο κώδικας περιλαμβάνει τη δήλωση CREATE TABLE, το όνομα του πίνακα, τα γνωρίσματά του, το πρωτεύον κλειδί καθώς και τα μοναδικά κλειδιά του. Για κάθε γνώρισμα, προστίθεται μια γραμμή που περιέχει το

όνομα του γνωρίσματος, τον τύπο δεδομένων του και ενδεχομένως τη δήλωση NOT NULL αν το γνώρισμα είναι υποχρεωτικό. Ακόμα, ελέγχεται εάν υπάρχει πρωτεύον κλειδί (primaryKey) για τον πίνακα. Αν υπάρχει, προστίθεται μια γραμμή με τη δήλωση PRIMARY KEY και τα ονόματα των γνωρισμάτων που αποτελούν το πρωτεύον κλειδί. Επιπλέον, υπάρχει ένας βρόχος που εξετάζει τα μοναδικά γνωρίσματα (Unique) του πίνακα και προσθέτει τις κατάλληλες δηλώσεις SQL. Στο τέλος, προστίθεται ο SQL κώδικας για τον πίνακα στο sqlString που αποθηκεύει τον ολοκληρωμένο κώδικα SQL του διαγράμματος.

```
//after declarining the tables we declare their foreign keys
//in order not to generate errors referring to tables that have not been declared
for (Table t : tables){
  StringBuilder s = new StringBuilder();
  if (t.foreignKey.size() > 0){
    for (Foreign frg : t.foreignKey){
        s.append("ALTER TABLE " + t.name + " ADD FOREIGN KEY (") ;
       for(mxCell f : frg.List){
        s.append(f.getValue() + ", ");
       }
        s.delete(s.length() - 2, s.length());
        String ref = (String) frg.refersTo.getValue();
        s.append( ") REFERENCES " + ref +" (");

       for(mxCell f : frg.List){
        String fv = (String) f.getValue() ;
        if (fv.contains(s:"_")) {
          String[] parts = fv.split(regex:"_");
          fv = parts[1];
        }
         s.append( fv  + ", ");
       }
        s.delete(s.length() - 2, s.length());
        s.append(  ") " + frg.constraint + ";" + "\n" + "\n");
    }
  }
  sqlString += s.toString();

}
```

Εικόνα 27: Προσθήκη δηλώσεων ξένων κλειδιών

Μετά από τη δημιουργία των πινάκων, υπάρχει άλλος ένας βρόχος που εξετάζει τους πίνακες ξανά και δημιουργεί τις δηλώσεις για τα ξένα κλειδιά (foreignKey). Αυτές οι δηλώσεις προστίθενται στο sqlString μετά τον κατάλληλο χειρισμό και δημιουργούνται με τη χρήση της δομής ALTER TABLE.

```
//save the sql code
JFileChooser chooser = new JFileChooser();
FileNameExtensionFilter filter = new FileNameExtensionFilter(description:"Text Document (*.txt, *.sql)", ...extensions:"txt", "sql");
chooser.setFileFilter(filter);
chooser.setDialogTitle(dialogTitle:"Save SQL");
int returnVal = chooser.showSaveDialog(parent:null);
if (returnVal == JFileChooser.APPROVE_OPTION) {
  File file = chooser.getSelectedFile();
  String filePath = file.getPath();
  String extension = "";
  if(!filePath.toLowerCase().endsWith(suffix:".txt") && !filePath.toLowerCase().endsWith(suffix:".sql")){
    int option = JOptionPane.showOptionDialog(parentComponent:null, message:"Do you want to save as a .txt or .sql file?",
    title:"Save as", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, icon:null, new Object[] {"txt", "sql"}, initialValue:"txt");
    if(option == JOptionPane.YES_OPTION) {
        extension = ".txt";
    } else if(option == JOptionPane.NO_OPTION) {
        extension = ".sql";
    }
  }
  file = new File(filePath + extension);
  if (file.exists()) {
    int choice = JOptionPane.showConfirmDialog(parentComponent:null,
    message:"The file already exists. Do you want to overwrite it?", title:"Confirm Overwrite", JOptionPane.YES_NO_OPTION);
    if (choice != JOptionPane.YES_OPTION) {
      return;
    }
  }
  try {
      FileWriter writer = new FileWriter(file);
      writer.write(sqlString);
      writer.close();
  } catch (IOException ex) {
      ex.printStackTrace();
  }
}
```

Εικόνα 28: Αποθήκευση του SQL κώδικα

Το επόμενο βήμα είναι η αποθήκευση του παραγόμενου SQL κώδικα σε ένα αρχείο. Χρησιμοποιείται η κλάση `JFileChooser` από τη βιβλιοθήκη Java Swing για να εμφανιστεί ένα παράθυρο διαλόγου επιλογής αρχείου για τον χρήστη και ορίζεται το φίλτρο αρχείων `FileNameExtensionFilter` ώστε να εμφανίζονται μόνο αρχεία με τις επεκτάσεις ".txt" και ".sql". Στη συνέχεια, ορίζεται ο τίτλος του παραθύρου διαλόγου ως "Save SQL".

Μετά την εμφάνιση του παραθύρου διαλόγου, ο χρήστης μπορεί να επιλέξει έναν φάκελο και ένα όνομα αρχείου για να αποθηκευτεί ο SQL κώδικας. Αν ο χρήστης πατήσει το κουμπί "Αποθήκευση" (APPROVE_OPTION), τότε ο κώδικας συνεχίζει την επεξεργασία. Δημιουργείται ένα `File` αντικείμενο με το δοθέν όνομα αρχείου και ελέγχεται η κατάληξη του αρχείου. Αν το αρχείο δεν έχει τις κατάλληλες επεκτάσεις, δηλαδή τις".txt" ή ".sql", τότε εμφανίζεται ένα παράθυρο διαλόγου για να ζητηθεί από το χρήστη να επιλέξει την κατάλληλη επέκταση. Στη συνέχεια, προστίθεται η επιλεγμένη επέκταση στο όνομα του αρχείου.

Σε περίπτωση που το αρχείο που έχει επιλεχθεί για να αποθηκευτεί ο SQL κώδικας υπάρχει ήδη, εμφανίζεται ένα παράθυρο διαλόγου για να επιβεβαιωθεί αν ο χρήστης επιθυμεί να το αντικαταστήσει. Αν ο χρήστης δεν επιλέξει την επιβεβαίωση (JOptionPane.YES_OPTION), τότε η διαδικασία αποθήκευσης διακόπτεται. Αν ο χρήστης συμφωνήσει να αντικαταστήσει το αρχείο (πατώντας το κουμπί "YES"), τότε δημιουργείται ένα νέο αντικείμενο FileWriter για την εγγραφή του κώδικα στο αρχείο. Ο κώδικας SQL αποθηκεύεται στο αρχείο μέσω της μεθόδου write(sqlString) και έπειτα κλείνει ο FileWriter με τη μέθοδο close(). Αν προκύψει κάποιο σφάλμα κατά την εγγραφή του αρχείου (π.χ. αδυναμία πρόσβασης ή μη ύπαρξη του αρχείου), τότε εμφανίζεται το αντίστοιχο μήνυμα σφάλματος στην κονσόλα.

Ο παραπάνω κώδικας επιτρέπει στον χρήστη να αποθηκεύσει τον παραγόμενο SQL κώδικα σε ένα αρχείο και να τον χρησιμοποιήσει αργότερα αν χρειαστεί.

4.4 Αποθήκευση, άνοιγμα και αρχεία εξόδου

Σε αυτή την ενότητα αναφέρονται οι δυνατότητες της εφαρμογής ERD tools σχετικά με το άνοιγμα και την αποθήκευση ενός διαγράμματος αλλά και τα παραγόμενα αρχεία που δύνανται να δημιουργηθούν.

4.4.1 Αποθήκευση

Τα διαγράμματα αποθηκεύονται με την κατάληξη ".erdt", η οποία υποδηλώνει ότι πρόκειται για αρχείο διαγράμματος της εν λόγω εφαρμογής. Η αποθήκευση γίνεται χρησιμοποιώντας τη γλώσσα XML. Η γλώσσα XML (Extensible Markup Language) είναι μια γλώσσα σήμανσης που χρησιμοποιείται για την κωδικοποίηση δεδομένων με δομή. Στην περίπτωση του εργαλείου ERD tools, η XML χρησιμοποιείται για να κωδικοποιήσει κάθε αντικείμενο του διαγράμματος, λαμβάνοντας υπόψη τις ιδιότητες και τις συνδέσεις του.

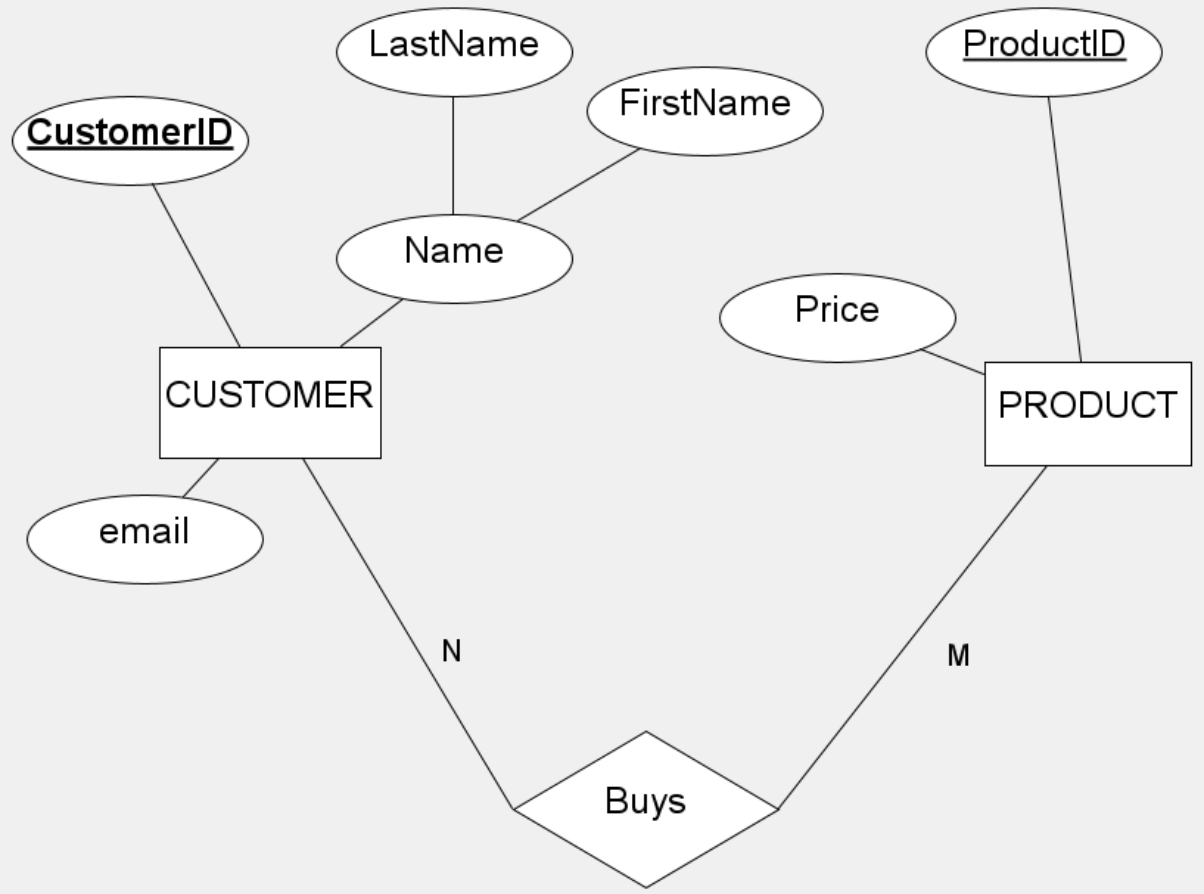
Κατά την αποθήκευση, οι πληροφορίες για κάθε αντικείμενο του διαγράμματος μετατρέπονται σε μια δομή XML. Η XML παρέχει μια διαμορφωμένη δομή για την αποθήκευση των δεδομένων και των σχέσεων μεταξύ τους. Αυτό επιτρέπει την αποτύπωση των λειτουργικών και δομικών χαρακτηριστικών του διαγράμματος σε μια δομή κειμένου. Συνεπώς, η χρήση της κατάληξης ".erdt" εξασφαλίζει την επιτυχημένη αποθήκευση και άνοιγμα των διαγραμμάτων διατηρώντας τις ιδιότητες και τις συνδέσεις τους για περαιτέρω επεξεργασία ή ανάγνωση.

```
// Adds xml format
DefaultFileFilter erdtFilter = new DefaultFileFilter(extension:".erdt",
"ERDT " + mxResources.get(key:"file") + " (.erdt)");

String filename = null;

if (showDialog || editor.getCurrentFile() == null)
{
    String wd;

    if (lastDir != null)
    {
        wd = lastDir;
    }
    else if (editor.getCurrentFile() != null)
    {
        wd = editor.getCurrentFile().getParent();
    }
    else
    {
        wd = System.getProperty(key:"user.dir");
    }


    JFileChooser fc = new JFileChooser(wd);

    fc.setAcceptAllFileFilterUsed(b:false);

    // Adds the default file format
    FileFilter defaultFilter = erdtFilter;
    fc.addChoosableFileFilter(defaultFilter);


    int rc = fc.showDialog(parent:null, mxResources.get(key:"save"));
```

Εικόνα 29: Η μέθοδος actionPerformed της κλάσης SaveAction

Η κλάση SaveAction που βρίσκεται στην EditorActions και επεκτείνει την `AbstractAction`, υλοποιεί τη λειτουργία αποθήκευσης για τον επεξεργαστή γραφήματος. Αναλυτικότερα, η μέθοδος `actionPerformed` υλοποιεί τη δράση που εκτελείται όταν πατηθεί το κουμπί για την αποθήκευση. Αρχικά, ανακτά τον επεξεργαστή γραφήματος `BasicGraphEditor` από το `ActionEvent`. Έπειτα, ελέγχει αν υπάρχει ένα ενεργό γράφημα και εμφανίζει το παράθυρο επιλογής αρχείου. Στη συνέχεια, αναλαμβάνει τη διαχείριση της αποθήκευσης του γραφήματος, με επιλεγμένη μορφή αρχείου την erdt. αποθηκεύει το γράφημα ως XML αρχείο

χρησιμοποιώντας τη μέθοδο `mxUtils.writeFile`. Τέλος, ενημερώνει τον επεξεργαστή γραφήματος για το νέο αρχείο και την κατάσταση τροποποίησης.



Εικόνα 30: Το διάγραμμα ER μεταξύ των τύπων οντοτήτων Πελάτης και Προϊόν όπως και η συσχέτιση-σχέση μεταξύ τους

Παρακάτω παρουσιάζεται ένα δείγμα από ένα αρχείο XML που περιγράφει ένα γράφημα χρησιμοποιώντας τη μορφή mxGraphModel. Το διάγραμμα περιγράφεται μέσω των διάφορων στοιχείων <mxCell>, με κάθε στοιχείο να αντιπροσωπεύει έναν κόμβο ή μια ακμή. Κάθε κόμβος ή ακμή έχει μια μοναδική ταυτότητα (id) και ορισμένες ιδιότητες που καθορίζουν τη μορφή και τις συνδέσεις τους. Οι ιδιότητες αυτές περιλαμβάνουν τον τύπο του στοιχείου (κόμβος ή ακμή), τη μορφή του (π.χ. ορθογώνιο, ρόμβος, έλλειψη), τη γεωμετρία (geometry) που καθορίζει τη θέση, το πλάτος και το ύψος του στην αναπαράσταση του γραφήματος, τις συνδέσεις του με άλλα στοιχεία και το όνομα-τιμή(value) που εμφανίζεται σε αυτό το στοιχείο. Οι ακμές επίσης έχουν μοναδικά αναγνωριστικά και γεωμετρία, καθώς και στοιχεία πηγής (source) και προορισμού (target) που δείχνουν τους κόμβους που συνδέονται με την ακμή.

Το γράφημα της εικόνας 30 αναφέρεται σε μια δομή δεδομένων που αντιπροσωπεύει συσχετίσεις μεταξύ πελατών (CUSTOMER) και προϊόντων (PRODUCT). Ακολουθεί η XML μορφή του αρχείου που προκύπτει από την αποθήκευση του διαγράμματος.

```
<mxGraphModel><root><mxCell id="0"/>
<mxCell id="1" parent="0"/>

<mxCell id="2" parent="1" style="rectangle" value="CUSTOMER" vertex="1">
<mxGeometry as="geometry" height="75.0" width="150.0" x="110.0" y="380.0"/>
</mxCell>

<mxCell id="3" parent="1" style="rectangle" value="PRODUCT" vertex="1">
<mxGeometry as="geometry" height="70.0" width="140.0" x="670.0" y="390.0"/>
</mxCell>

<mxCell id="4" parent="1" style="rhombus" value="Buys" vertex="1"><mxGeometry as="geometry" height="106.0" width="180.0" x="350.0" y="640.0"/>
<mxCell as="leftOwner" id="2" parent="1" style="rectangle" value="CUSTOMER" vertex="1"><mxGeometry as="geometry" height="75.0" width="150.0" x="110.0" y="380.0"/></mxCell>
<mxCell as="rightOwner" id="3" parent="1" style="rectangle" value="PRODUCT" vertex="1"><mxGeometry as="geometry" height="70.0" width="140.0" x="670.0" y="390.0"/></mxCell>
</mxCell>

<mxCell edge="1" id="6" parent="1" source="4" style="spacingLeft=20;textPosition=right;fontSize=20;exitX=1;exitY=0.5;exitPerimeter=1" target="3" value="M">
<mxGeometry as="geometry" relative="1"/></mxCell>

<mxCell id="9" parent="1" style="ellipse" value="Name" vertex="1">
<mxGeometry as="geometry" height="60.0" width="160.0" x="230.0" y="290.0"/>
<mxCell as="owner" id="2" parent="1" style="rectangle" value="CUSTOMER" vertex="1"><mxGeometry as="geometry" height="75.0" width="150.0" x="110.0" y="380.0"/></mxCell>
```

</mxCell>

<mxCell edge="1" id="10" parent="1" source="9" target="2" value=""><mxGeometry as="geometry" relative="1"/></mxCell>

<mxCell id="11" parent="1" style="ellipse" value="email" vertex="1">
<mxGeometry as="geometry" height="60.0" width="160.0" x="20.0" y="480.0"/>
<mxCell as="owner" id="2" parent="1" style="rectangle" value="CUSTOMER" vertex="1">
<mxGeometry as="geometry" height="75.0" width="150.0" x="110.0" y="380.0"/></mxCell></mxCell>

<mxCell edge="1" id="12" parent="1" source="11" target="2" value=""><mxGeometry as="geometry" relative="1"/></mxCell>

<mxCell id="13" parent="1" style="ellipse" value="FirstName" vertex="1"><mxGeometry as="geometry" height="60.0" width="160.0" x="400.0" y="190.0"/><mxCell as="owner" id="9" parent="1" style="ellipse" value="Name" vertex="1"><mxGeometry as="geometry" height="60.0" width="160.0" x="230.0" y="290.0"/>
<mxCell as="owner" id="2" parent="1" style="rectangle" value="CUSTOMER" vertex="1"><mxGeometry as="geometry" height="75.0" width="150.0" x="110.0" y="380.0"/></mxCell></mxCell></mxCell>

<mxCell edge="1" id="14" parent="1" source="13" target="9" value=""><mxGeometry as="geometry" relative="1"/></mxCell>

<mxCell id="15" parent="1" style="ellipse" value="LastName" vertex="1"><mxGeometry as="geometry" height="60.0" width="160.0" x="230.0" y="150.0"/>
<mxCell as="owner" id="9" parent="1" style="ellipse" value="Name" vertex="1"><mxGeometry as="geometry" height="60.0" width="160.0" x="230.0" y="290.0"/>
<mxCell as="owner" id="2" parent="1" style="rectangle" value="CUSTOMER" vertex="1"><mxGeometry as="geometry" height="75.0" width="150.0" x="110.0" y="380.0"/></mxCell></mxCell></mxCell>

<mxCell edge="1" id="16" parent="1" source="15" target="9" value=""><mxGeometry as="geometry" relative="1"/></mxCell>

<mxCell id="19" parent="1" style="ellipse" value="Price" vertex="1"><mxGeometry as="geometry" height="60.0" width="160.0" x="490.0" y="330.0"/>
<mxCell as="owner" id="3" parent="1" style="rectangle" value="PRODUCT" vertex="1"><mxGeometry as="geometry" height="70.0" width="140.0" x="670.0" y="390.0"/></mxCell></mxCell>

<mxCell edge="1" id="20" parent="1" source="19" target="3" value=""><mxGeometry as="geometry" relative="1"/></mxCell>
<mxCell edge="1" id="22" parent="1" source="4" style="spacingLeft=20;textPosition=right;fontSize=20;exitX=0;exitY=0.5;exitPerimeter=1" target="2" value="N"><mxGeometry as="geometry" relative="1"/></mxCell>

<mxCell id="23" parent="1" style="unique" value="ProductID" vertex="1"><mxGeometry as="geometry" height="60.0" width="160.0" x="630.0" y="150.0"/>
<mxCell as="owner" id="3" parent="1" style="rectangle" value="PRODUCT" vertex="1"><mxGeometry as="geometry" height="70.0" width="140.0" x="670.0" y="390.0"/></mxCell></mxCell>
<mxCell edge="1" id="24" parent="1" source="23" target="3" value=""><mxGeometry as="geometry" relative="1"/></mxCell>

<mxCell id="25" parent="1" style="primary" value="CustomerID" vertex="1"><mxGeometry as="geometry" height="60.0" width="160.0" x="10.0" y="210.0"/>
<mxCell as="owner" id="2" parent="1" style="rectangle" value="CUSTOMER" vertex="1"><mxGeometry as="geometry" height="75.0" width="150.0" x="110.0" y="380.0"/></mxCell></mxCell>

<mxCell edge="1" id="26" parent="1" source="25" target="2" value=""><mxGeometry as="geometry" relative="1"/>
</mxCell></root></mxGraphModel>

Η πρώτη γραμμή <mxGraphModel> ανοίγει το στοιχείο που αναπαριστά το μοντέλο του γραφήματος και ακολουθεί ο ριζικός κόμβος, τον οποίο έχουν ως γονέα όλοι οι κόμβοι και ακμές του διαγράμματος. Για τη δήλωση ενός κόμβου χρησιμοποιείται η vertex="1", ενώ για

μία ακμή με η edge="1". Στο εν λόγω παράδειγμα ο κόμβος με την τιμή Buys αναπαριστά τη σχέση αγοράζει μεταξύ πελάτη και προϊόντος. Ας αναλύσουμε την περιγραφή του:

- <mxCell id="4" parent="1" style="rhombus" value="Buys" vertex="1">:

- Το μοναδικό αναγνωριστικό του αντικειμένου είναι id="4".
- Ο γονικός κόμβος του είναι ο κόμβος με id="1".
- Το στυλ του αντικειμένου ορίζεται ως "rhombus", δηλαδή αποτελεί μία συσχέτιση στο γράφημα.
- Το όνομα του αντικειμένου είναι "Buys".
- Το αντικείμενο είναι ένας κόμβος (vertex="1"), δηλαδή δεν αναπαριστά μια ακμή.

- <mxCell as="leftOwner" id="2" parent="1" style="rectangle" value="CUSTOMER" vertex="1">:

- Το αντικείμενο με as="leftOwner" και id="2" συνδέεται στο αριστερό μέρος της συσχέτισης Buys.
- Ο γονικός κόμβος του είναι ο κόμβος με id="1".
- Ο στυλ του αντικειμένου ορίζεται ως "rectangle" δηλαδή πρόκειται για οντότητα.
- Το όνομα του αντικειμένου είναι "CUSTOMER".
- Το αντικείμενο είναι ένας κόμβος (vertex="1").

- <mxCell as="rightOwner" id="3" parent="1" style="rectangle" value="PRODUCT" vertex="1">:

Όμοια με πριν.

- <mxCell edge="1" id="22" parent="1" source="4" style="spacingLeft=20;textPosition=right;fontSize=20;exitX=0;exitY=0.5;exitPerimeter=1" target="2" value="N"><mxGeometry as="geometry" relative="1"/></mxCell>

Εδώ πρόκειται για μία ακμή μεταξύ των αντικειμένων με id=4 και id=2, δηλαδή των CUSTOMER και Buys. Το style αναφέρεται στον τρόπο με τον οποίο συνδέονται αυτά τα

αντικείμενα, που στην περίπτωση αυτή είναι μερική συμμετοχή με λόγο πληθικότητας N, ενώ με τη χρήση του "strokeWidth=5;" ορίζεται το πάχος της γραμμής της ακμής. Η "spacingLeft=20;" ορίζει το περιθώριο αριστερά του κειμένου στην ακμή, η "textPosition=right;" ορίζει τη θέση του κειμένου στην ακμή (δεξιά στην προκείμενη περίπτωση) και η "fontSize=20;" ορίζει το μέγεθος γραμματοσειράς του κειμένου. Ακόμη, με την "exitX=0.5;exitY=1;exitPerimeter=1" ορίζονται οι συντεταγμένες εξόδου της ακμής, δηλαδή το σημείο από το οποίο ξεκινά η ακμή από τον κόμβο πηγής, δηλαδή της συσχέτισης (στην προκείμενη περίπτωση από την αριστερή γωνία του ρόμβου). Η τιμή του relative είναι "1" για τις ακμές, πράγμα που σημαίνει ότι η γεωμετρία της ακμής προσδιορίζεται σε σχέση με τον γονικό κόμβο.

4.4.2 Άνοιγμα αρχείου

Η μέθοδος actionPerformed υλοποιεί τη δράση που εκτελείται όταν πατηθεί το αντίστοιχο κουμπί για το άνοιγμα αρχείων. Ως πρώτο βήμα, ελέγχει αν ο επεξεργαστής γραφήματος δεν έχει τροποποιηθεί ή αν ο χρήστης επιθυμεί να χάσει τις αλλαγές. Στη συνέχεια, δημιουργεί ένα παράθυρο επιλογής αρχείου (file chooser) με την τελευταία διαδρομή που χρησιμοποιήθηκε. Ορίζει ένα φίλτρο αρχείων για τον επιλεγμένο τύπο αρχείου (.erdt) και εμφανίζει το παράθυρο επιλογής αρχείου στο χρήστη. Αν ο χρήστης επιλέξει ένα αρχείο, τότε διαβάζει το περιεχόμενο του αρχείου XML και κωδικοποιεί το XML για να κατασκευάσει το γράφημα στον επεξεργαστή γραφήματος.

```
if (graph != null)
{
    String wd = (lastDir != null) ? lastDir : System
            .getProperty(key:"user.dir");

    JFileChooser fc = new JFileChooser(wd);
    fc.setAcceptAllFileFilterUsed(b:false);

    // Adds file filter for supported file format
    DefaultFileFilter defaultFilter = new DefaultFileFilter(extension:".erdt",
    "ERDT" + mxResources.get(key:"file") + " (.erdt)");
    fc.addChoosableFileFilter(defaultFilter);
    fc.setFileFilter(defaultFilter);

    int rc = fc.showDialog(parent:null,
            mxResources.get(key:"openFile"));

    if (rc == JFileChooser.APPROVE_OPTION)
    {
        lastDir = fc.getSelectedFile().getParent();

        try
        {
            Document document = mxXmlUtils.parseXml(mxUtils.readFile(fc.getSelectedFile().getAbsolutePath()));

            mxCodec codec = new mxCodec(document);
            codec.decode(
                    document.getDocumentElement(),graph.getModel());
                    editor.setCurrentFile(fc.getSelectedFile());

            resetEditor(editor);
        }

        catch (IOException ex)
        {
```

Εικόνα 31: Άνοιγμα αρχείου

Στη συνέχεια, επαναφέρει τον επεξεργαστή γραφήματος. Αν συμβεί κάποιο σφάλμα κατά τη διάρκεια του άνοιγμα του αρχείου, εμφανίζεται ένα παράθυρο μηνύματος λάθους. Συμπερασματικά, ο κώδικας αυτός αναλαμβάνει να επεξεργαστεί το αρχείο XML και να δημιουργήσει το αντίστοιχο γράφημα στην εφαρμογή.

4.4.3 Αρχεία εξόδου

Η κλάση Export είναι υπεύθυνη για την εξαγωγή γραφημάτων σε διάφορα αρχεία εικόνας (PNG και SVG). Ο κώδικας χειρίζεται την αλληλεπίδραση με τον χρήστη, εμφανίζοντας ένα παράθυρο διαλόγου για να επιλέξει τη μορφή αρχείου και την τοποθεσία αποθήκευσης. Ανάλογα με την επιλεγμένη μορφή αρχείου, δημιουργείται η αντίστοιχη εικόνα για το διάγραμμα και αποθηκεύεται στον τοπικό δίσκο. Για παράδειγμα, αν η επιλεγμένη μορφή είναι

PNG, ο κώδικας χρησιμοποιεί την κλάση mxCellRenderer για να δημιουργήσει μια εικόνα PNG από το γράφημα και την αποθηκεύει σε ένα αρχείο.

```
if (showDialog || editor.getCurrentFile() == null)
{
    String wd;
    if (lastDir != null)
    {
        wd = lastDir;
    }
    else if (editor.getCurrentFile() != null)
    {
        wd = editor.getCurrentFile().getParent();
    }
    else
    {
        wd = System.getProperty(key:"user.dir");
    }

    JFileChooser fc = new JFileChooser(wd);
    fc.setAcceptAllFileFilterUsed(b:false);

    // Adds special vector graphics format
    fc.addChoosableFileFilter(new DefaultFileFilter(extension:".svg",
            "SVG " + mxResources.get(key:"file") + " (.svg)"));

    // Adds png format
    fc.addChoosableFileFilter(new DefaultFileFilter(extension:".png",
            "PNG " + mxResources.get(key:"file") + " (.png)"));

    int rc = fc.showDialog(parent:null, mxResources.get(key:"save"));
    dialogShown = true;

    if (rc != JFileChooser.APPROVE_OPTION)
    {
        return;
    }
    else
    {
```

Εικόνα 32: τμήμα της κλάσης Export για τη δημιουργία αρχείων .svg και .png

Η κλάση PrintAction υλοποιεί τη λειτουργικότητα για την εκτύπωση του γραφήματος. Με τη μέθοδο actionPerformed δημιουργείται ένα αντικείμενο της κλάσης PrinterJob για να ρυθμίσει και να διαχειριστεί τη διαδικασία εκτύπωσης. Αφού εμφανίσει το παράθυρο διαλόγου

εκτύπωσης και ο χρήστης επιλέξει την εκτύπωση, η μέθοδος αποκτά την μορφή σελίδας (PageFormat). Στη συνέχεια, δημιουργεί ένα αντικείμενο της κλάσης Paper για να ορίσει την περιοχή εκτύπωσης της σελίδας, με βάση τις προδιαγραφές της μορφής σελίδας. Με αυτό τον τρόπο, το αντικείμενο PrinterJob χρησιμοποιεί την επιλεγμένη μορφή σελίδας και την περιοχή εκτύπωσης για να εκτυπώσει το διάγραμμα μέσω της μεθόδου print().

```
public static class PrintAction extends AbstractAction
{
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() instanceof mxGraphComponent)
        {
            mxGraphComponent graphComponent = (mxGraphComponent) e
                    .getSource();
            PrinterJob pj = PrinterJob.getPrinterJob();

            if (pj.printDialog())
            {
                PageFormat pf = graphComponent.getPageFormat();
                Paper paper = new Paper();
                double margin = 0;
                paper.setImageableArea(margin, margin, pf.getWidth(), pf.getHeight());
                pf.setPaper(paper);
                pj.setPrintable(graphComponent, pf);

                try
                {
                    pj.print();
                }
                catch (PrinterException e2)
                {
                    System.out.println(e2);
                }
            }
        }
    }
}
```

Εικόνα 33: Η κλάση PrintAction

5. Λειτουργία της εφαρμογής ERD tools

Η ανάπτυξη ενός ολοκληρωμένου εργαλείου μοντελοποίησης βάσεων δεδομένων σε Java μπορεί να γίνει με τη χρήση διάφορων τεχνολογιών και πρακτικών. Ως πρώτο βήμα της ανάπτυξης ορίζεται ο καθορισμός των απαιτήσεων, όπου προσδιορίζονται οι λειτουργίες που πρέπει να παρέχει, όπως η δημιουργία και επεξεργασία διαγραμμάτων οντοτήτων συσχετίσεων (ER), ο έλεγχος για πιθανά σφάλματα που προέκυψαν κατά τη σχεδίαση του διαγράμματος και η μετατροπή σε κώδικα SQL. Επόμενο βήμα είναι ο σχεδιασμός της διεπαφής χρήστη και έπειτα η υλοποίηση των λειτουργιών που έχουν προσδιοριστεί. Τέλος, ακολουθούν δοκιμές με στόχο τη βελτιστοποίηση του εργαλείου. Στο παρόν κεφάλαιο, αναλύονται πρώτα οι απαιτήσεις της εφαρμογής ERD tools και γίνεται μια παρουσίαση της εφαρμογής για να περιγραφούν οι δυνατότητές της.

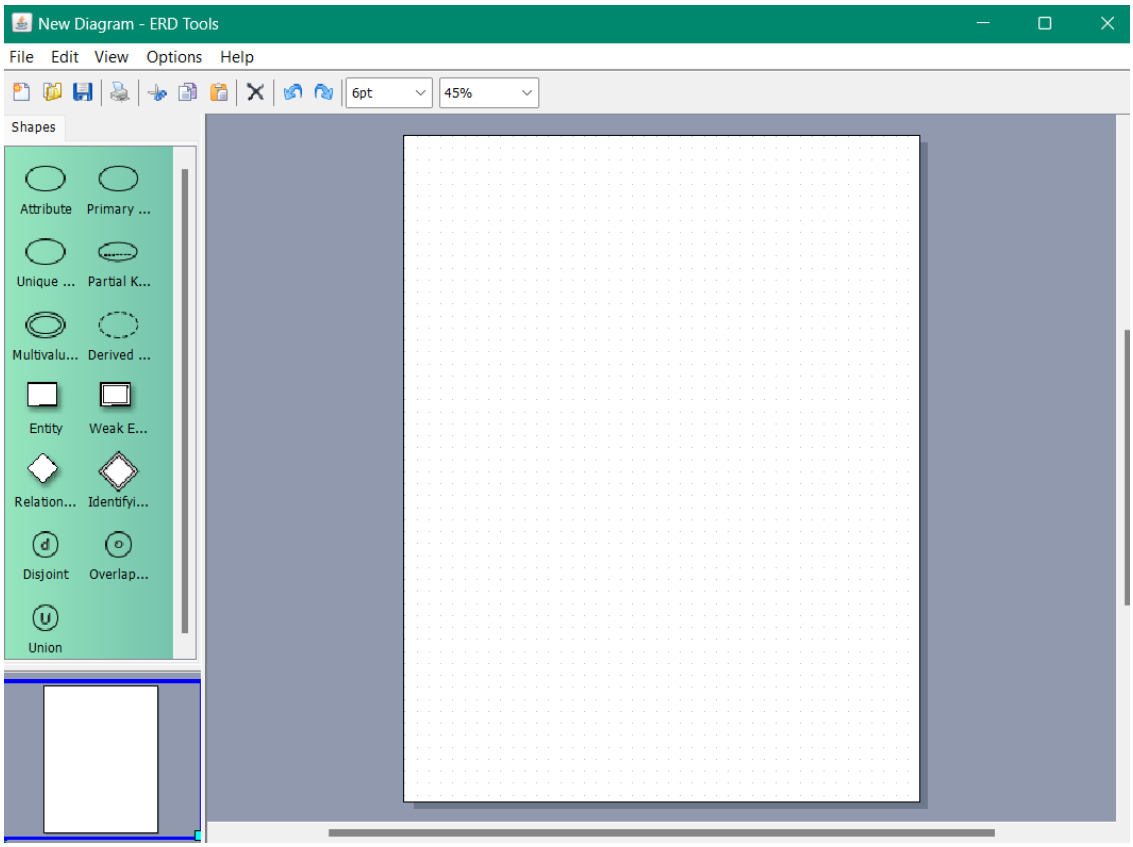
5.1 Περιγραφή της ERD tools

Για την εκτέλεση της εφαρμογής αρκεί ο χρήστης να κάνει διπλό κλικ στο εκτελέσιμο αρχείο **erdt.exe**. Για να διασφαλιστεί η σωστή λειτουργία της εφαρμογής, απαιτείται η εγκατάσταση του Java Runtime Environment (JRE) έκδοσης 18 ή νεότερης. Αυτό σημαίνει ότι ο υπολογιστής του χρήστη πρέπει να έχει εγκατεστημένη την αντίστοιχη έκδοση του JRE πριν μπορέσει να εκτελέσει την εφαρμογή. Για να δημιουργηθεί το εκτελέσιμο αρχείο "erdt.exe", χρησιμοποιήθηκε το εργαλείο Launch4j. Αυτό το εργαλείο είναι υπεύθυνο για τη μετατροπή του αντίστοιχου αρχείου JAR της εφαρμογής, το οποίο δημιουργήθηκε μέσω του Visual Studio, σε ένα εκτελέσιμο αρχείο για το περιβάλλον των Windows (.exe). Το Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού που παρέχει εργαλεία και δυνατότητες για τη δημιουργία, την επεξεργασία και την ανάπτυξη διάφορων εφαρμογών, συμπεριλαμβανομένων των εφαρμογών Java. Έτσι, χρησιμοποιώντας το Visual Studio, δημιουργήθηκε το JAR αρχείο που απαιτείται για την εκτέλεση της εφαρμογής. Η δημιουργία του εκτελέσιμου αρχείου γίνεται για να διευκολυνθεί ο χρήστης στην εκτέλεση της εφαρμογής, χωρίς να χρειάζεται να ανοίξει μια κονσόλα και να εκτελέσει το αρχείο JAR ή το exe απευθείας. Ακολουθεί αναλυτική περιγραφή της εφαρμογής όπου παρουσιάζονται οι βασικές

αλλά και οι προχωρημένες λειτουργίες της, παρέχοντας στους χρήστες ολοκληρωμένες οδηγίες και κατευθυντήριες γραμμές για τη χρήση του εργαλείου.

5.2 Κεντρική Οθόνη

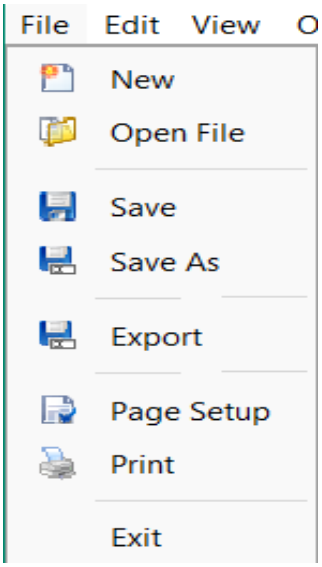
Στην εικόνα που ακολουθεί φαίνεται το κεντρικό παράθυρο της εφαρμογής.



Εικόνα 34: Το παράθυρο της εφαρμογής

Στο οριζόντιο μενού παρέχονται οι παρακάτω δυνατότητες:

Επιλογή File:



•Δημιουργία νέου αρχείου

•Άνοιγμα υπάρχοντος αρχείου

•Αποθήκευση αρχείου

•Εξαγωγή σε μορφή εικόνας png και svg

•Διαμόρφωση μεγέθους και προσανατολισμού σελίδας

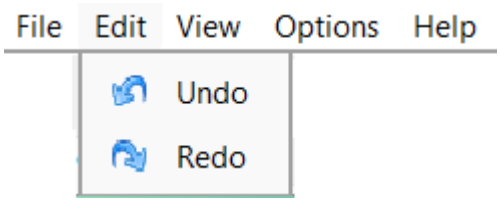
•Εκτύπωση συμπεριλαμβανομένης και της μετατροπής και αποθήκευσης του αρχείου σε pdf

•Έξοδος

Εικόνα 35: Το υπομενού του μενού File

Το μενού File παρέχει στο χρήστη τη δυνατότητα δημιουργίας νέου αρχείου, τη δυνατότητα ανοίγματος ενός υπάρχοντος αρχείου για επεξεργασία καθώς και την αποθήκευσή του σε κάποια επιλεγμένη τοποθεσία. Τα διαγράμματα αποθηκεύονται με την κατάληξη .erdt χρησιμοποιώντας τη γλώσσα XML που κωδικοποιεί κάθε αντικείμενο του διαγράμματος ανάλογα με τις ιδιότητες και τις συνδέσεις του. Επιπλέον, ο χρήστης μπορεί να εξάγει το διάγραμμα ως εικόνα αλλά και ως pdf, ώστε να μπορεί να ενσωματωθεί σε εφαρμογές γραφείου χωρίς περαιτέρω επεξεργασία. Ακόμη, το μέγεθος και ο προσανατολισμός της σελίδας στην οποία σχεδιάζονται τα διαγράμματα μπορούν να διαμορφωθούν ανάλογα με τις προτιμήσεις του χρήστη. Τέλος, η επιλογή Exit κλείνει το παράθυρο της εφαρμογής. Αξίζει να σημειωθεί ότι είναι διαθέσιμες οι αντίστοιχες συντομεύσεις πληκτρολογίου δηλαδή οι control+N για τη δημιουργία νέου αρχείου, control+O για το άνοιγμα υπάρχοντος αρχείου και control+S και control+shift+S για την αποθήκευση ενός αρχείου.

Επιλογή Edit:

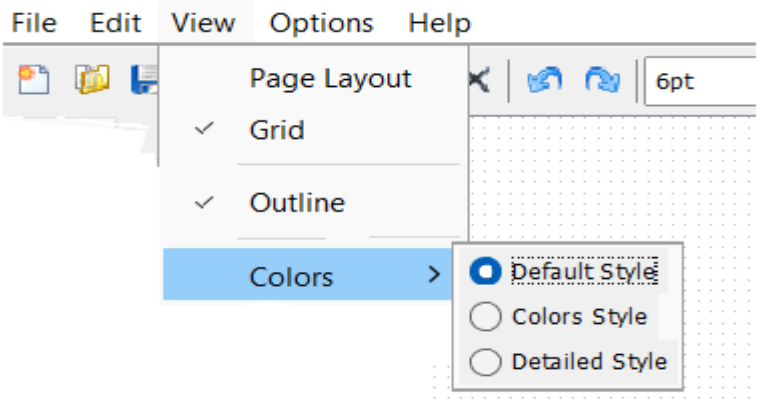


- Αναίρεση
- Επαναφορά

Εικόνα 36: Το υπομενού του μενού Edit

Το μενού Edit επιτρέπει στο χρήστη να αναιρέσει ή να επαναφέρει μια ενέργεια που πραγματοποίησε, όπως η διαγραφή ενός αντικειμένου, μιας σύνδεσης, ή η μετονομασία ενός αντικειμένου, επαναφέροντας το διάγραμμα στην προηγούμενη κατάσταση. Οι συντομεύσεις πληκτρολογίου σε αυτή την περίπτωση είναι control+Z για την αναίρεση και control+Y για την επανάληψη αντίστοιχα.

Επιλογή View:

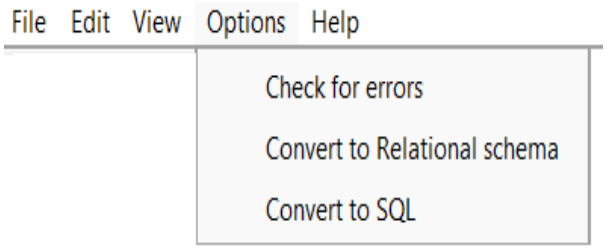


- Οργάνωση προβολής σελίδας
- Εμφάνιση δομής πλέγματος με κουκίδες
- Εμφάνιση μικρογραφίας διαγράμματος
- Χρωματισμός σχημάτων ανάλογα με το είδος τους

Εικόνα 37: Το υπομενού του μενού View

Το μενού View παρέχει δυνατότητες που σχετίζονται με την εμφάνιση των διαγραμμάτων. Πιο συγκεκριμένα, ο χρήστης δύναται να διαμορφώσει το πλαίσιο που καταλαμβάνει η σελίδα του διαγράμματος στην εφαρμογή μετακινώντας την προβολή του γραφήματος στο κέντρο ή επαναφέροντάς την στην αρχική θέση και να εμφανίσει ή να αποκρύψει ένα πλέγμα από κουκκίδες που στοχεύει στην ευκολότερη ευθυγράμμιση των αντικειμένων μέσα σε ένα διάγραμμα. Επίσης, υπάρχει δυνατότητα εμφάνισης της μικρογραφίας ενός διαγράμματος μέσω ενός παραθύρου επισκόπησης. Όταν η λειτουργία είναι επιλεγμένη, εμφανίζεται ένα παράθυρο που παρουσιάζει μια επισκόπηση του γραφήματος και την τρέχουσα θέα του χρήστη. Επιτρέπει στον χρήστη να μετακινείται γρήγορα σε διάφορα μέρη του γραφήματος και να έχει μια γενική εικόνα του γραφήματός του, ακόμα και όταν η προβολή είναι μεγεθυμένη ή μειωμένη.

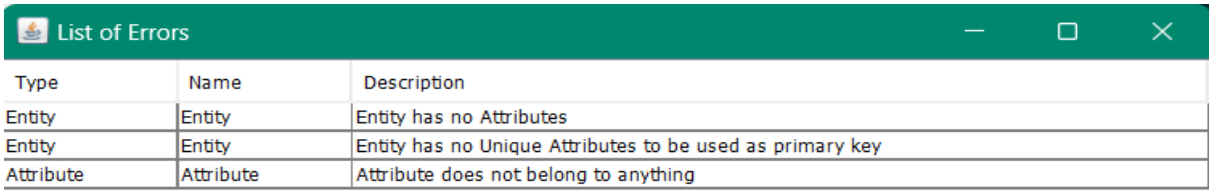
Επιλογή Options:



•Έλεγχος συντακτικής ορθότητας του διαγράμματος
•Δημιουργία Σχεσιακού Σχήματος
•Παραγωγή και αποθήκευση κώδικα SQL που υλοποιεί το Σχεσιακό Σχήμα

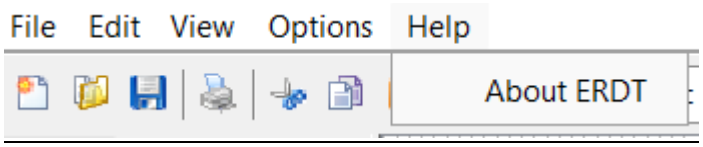
Εικόνα 38: Το υπομενού του μενού Options

Μέσω του μενού Options ο χρήστης είναι σε θέση να πραγματοποιήσει έλεγχο για τυχόν σφάλματα που προέκυψαν κατά το σχεδιασμό του διαγράμματος οντοτήτων συσχετίσεων που δημιούργησε. Σε περίπτωση που δε βρεθούν σφάλματα εμφανίζεται κατάλληλο μήνυμα που ενημερώνει το χρήστη, διαφορετικά εμφανίζεται το παράθυρο της εικόνας που ακολουθεί με την περιγραφή των σφαλμάτων. Αναλυτικότερη περιγραφή του είδους αυτών των σφαλμάτων γίνεται στην επόμενη ενότητα. Ακόμη, το μενού επιτρέπει στο χρήστη να δημιουργήσει το σχεσιακό σχήμα που αντιστοιχεί στο διάγραμμά του, καθώς και να αποθηκεύσει τον κώδικα SQL που υλοποιεί το σχεσιακό σχήμα σε σχεσιακό σύστημα βάσεων δεδομένων. Σημειώνεται ότι εάν έχουν προκύψει σφάλματα, οι δύο παραπάνω επιλογές του μενού επιστρέφουν το παράθυρο σφαλμάτων, χωρίς να προβούν στις αντίστοιχες διαδικασίες μετατροπής.



Εικόνα 39: Ο πίνακας σφαλμάτων

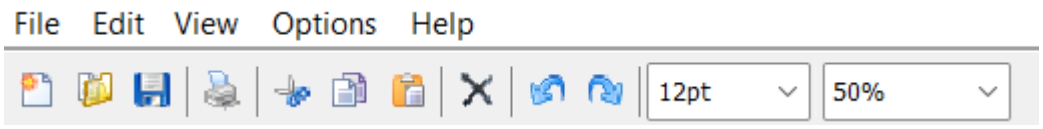
Επιλογή Help:



Εικόνα 40: Το μενού Help

Η επιλογή αυτή αποτελεί το τελευταίο μενού της πρώτης οριζόντιας μπάρας. Εδώ ο χρήστης μπορεί να δει μερικές πληροφορίες για την εφαρμογή.

Η δεύτερη οριζόντια μπάρα έχει, πέρα από τις συντομεύσεις για τις λειτουργίες άνοιγμα, αποθήκευση, αποθήκευση ως, εξαγωγή, αποκοπή, αντιγραφή, επικόλληση, διαγραφή, αναίρεση και επαναφορά, δύο επιπλέον λειτουργίες. Το πρώτο drop down μενού δίνει τη δυνατότητα στο χρήστη, αφού επιλέξει κάποιο αντικείμενο του διαγράμματος, να προσαρμόσει το μέγεθος της γραμματοσειράς του. Το δεύτερο drop down μενού επιτρέπει στο χρήστη να επιλέξει και να αλλάξει την κλίμακα (zoom) του γραφήματος στην εφαρμογή.



Εικόνα 41: Η δεύτερη οριζόντια μπάρα

Μενού αντικειμένων:

Στο αριστερό μέρος του παραθύρου βρίσκεται το μενού με την εργαλειοθήκη των σχημάτων που αντιστοιχούν τόσο στο σχεδιασμό διαγραμμάτων Οντοτήτων-Συσχετίσεων, όσο και Εκτεταμένων διαγραμμάτων Οντοτήτων-Συσχετίσεων. Ο χρήστης έχει τη δυνατότητα να μετακινήσει ένα σχήμα σε οποιαδήποτε θέση επιθυμεί εντός του πλαισίου, χρησιμοποιώντας τη λειτουργία του drag & drop. Απλά κρατώντας το αριστερό πλήκτρο του ποντικιού πατημένο πάνω στο σχήμα και μετακινώντας το ποντίκι, ο χρήστης μεταφέρει το σχήμα στον καμβά του εργαλείου. Τοποθετώντας το ποντίκι σε κάποιο γνώρισμα (hover) εμφανίζεται το αντίστοιχο tooltip με τον τύπο δεδομένων του γνωρίσματος. Οι τύποι αυτοί αξιοποιούνται κατά την παραγωγή του κώδικα SQL.

Μετά τη δημιουργία των σχημάτων, ο χρήστης μπορεί να ρυθμίσει τις ιδιότητες και τις συνδέσεις μεταξύ τους χρησιμοποιώντας ένα αναδυόμενο (popup) μενού που εμφανίζεται όταν κάνει δεξί κλικ πάνω σε κάθε αντικείμενο ξεχωριστά. Αυτό το μενού παρέχει τις επιλογές που αντιστοιχούν στις διαθέσιμες λειτουργίες για το συγκεκριμένο σχήμα, όπως μετονομασία, διαγραφή, ορισμός ιδιοτήτων ή δημιουργία συνδέσεων.

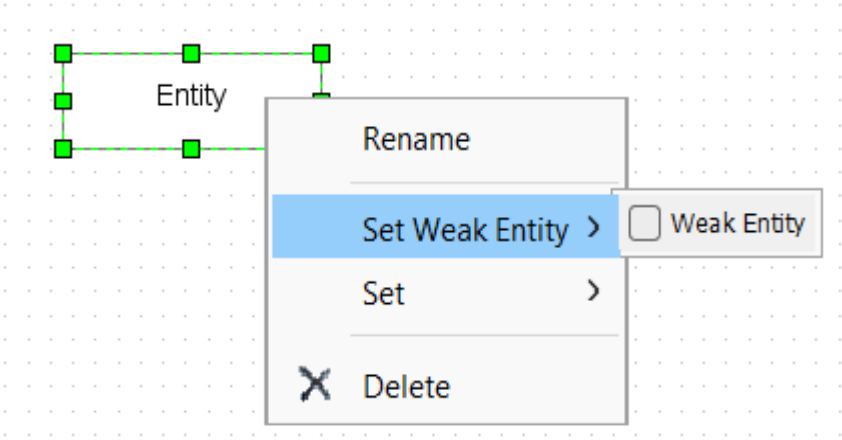
Ακολουθεί αναλυτικότερη επεξήγηση του αναδυόμενου μενού.

- Rename: Τροποποιείται το όνομα του αντικειμένου.
- Delete: Διαγραφή του αντικειμένου και των συνδέσεών του.

Οι δύο παραπάνω επιλογές είναι ίδιες για κάθε αντικείμενο οπότε στην ακόλουθη επεξήγηση του μενού κάθε σχήματος δε θα επαναληφθεί η περιγραφή τους.

Οντότητα (Entity)

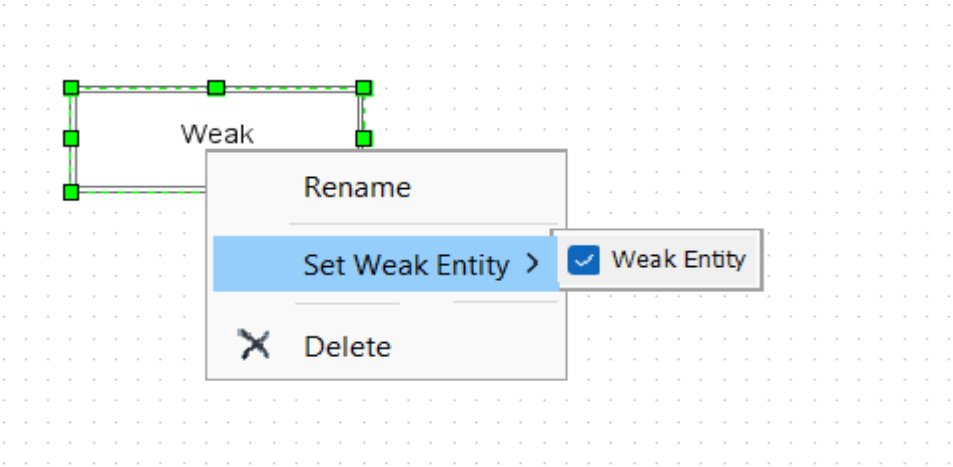
- Set Weak Entity: Καθορισμός ισχυρής ή ασθενούς οντότητας. Ουσιαστικά επιτρέπει στο χρήστη τη μετατροπή οντοτήτων από το ένα είδος στο άλλο, χωρίς διαγραφή συνδέσεων και γνωρισμάτων.
- Set Super: Καθορισμός άμεσης υπερκλάσης της επιλεγμένης οντότητας.



Εικόνα 42: Το αναδυόμενου μενού μίας οντότητας

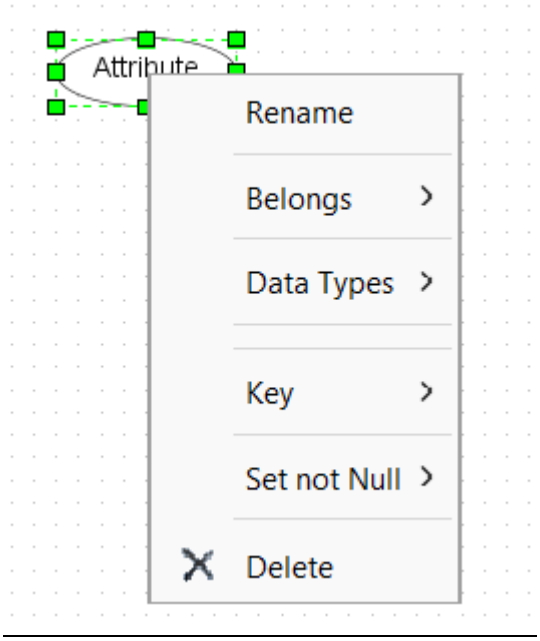
Ασθενής Οντότητα (Weak Entity)

Όμοια με πριν χωρίς την επιλογή για υπερκλάση.



Εικόνα 43: Το αναδυόμενου μενού μίας ασθενούς οντότητας

Γνώρισμα (Attribute)

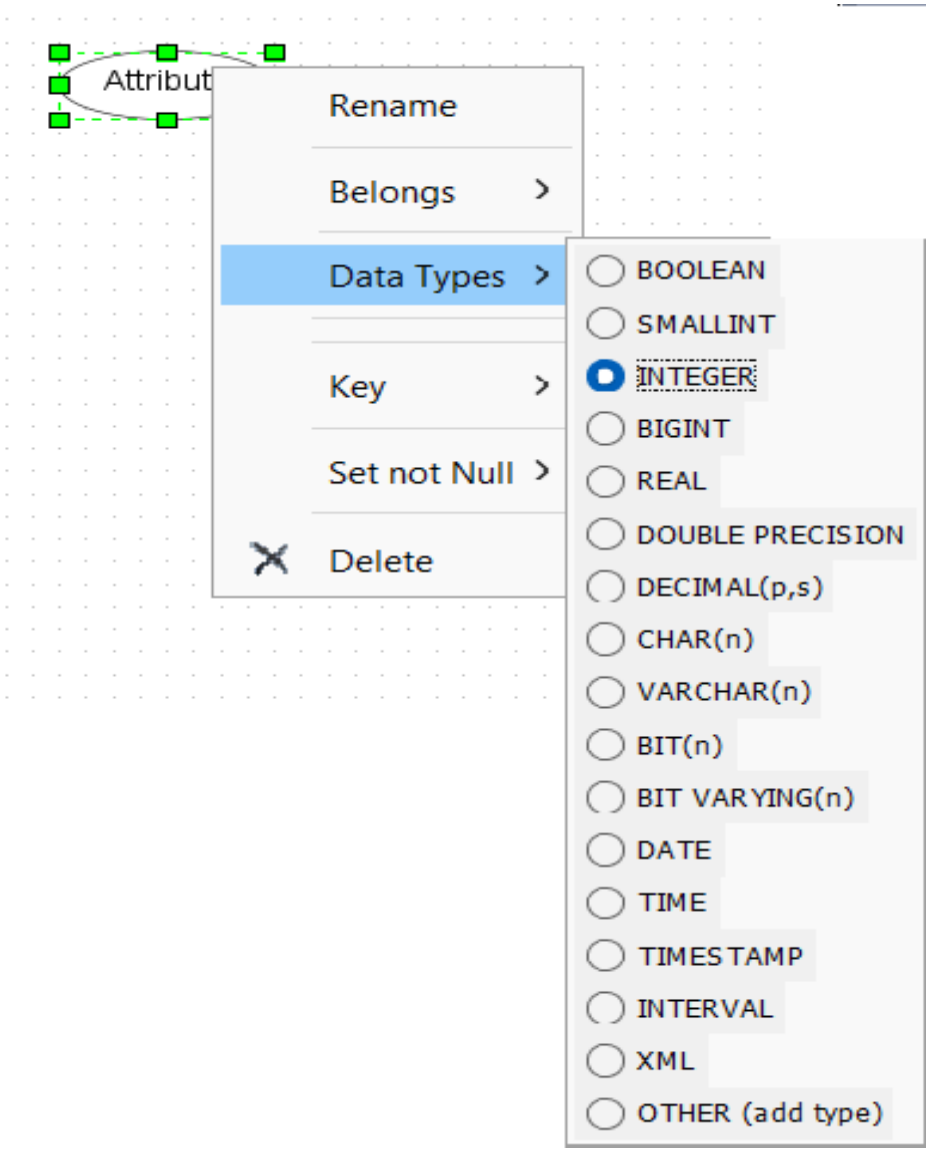


Εικόνα 44: Το αναδυόμενου μενού ενός γνωρίσματος

- Η επιλογή "Belongs" διαιρείται σε δύο επιλογές: "Set Owner" και "Remove Owner". Με την επιλογή "Set", ορίζεται ο τύπο οντότητας/συσχέτισης στον οποίο ανήκει το γνώρισμα. Αφού ο χρήστης κάνει αυτή την επιλογή, πρέπει να κάνει κλικ σε ένα σχήμα οντότητας/συσχέτισης για να ολοκληρωθεί η σύνδεση. Επιπλέον, ένα γνώρισμα μπορεί να έχει ως ιδιοκτήτη ένα άλλο γνώρισμα, όπως στην περίπτωση των σύνθετων γνωρισμάτων. Με την επιλογή "Remove", μπορείτε να αφαιρείται η σύνδεση. Ένα γνώρισμα μπορεί να ανήκει σε ένα μόνο αντικείμενο.

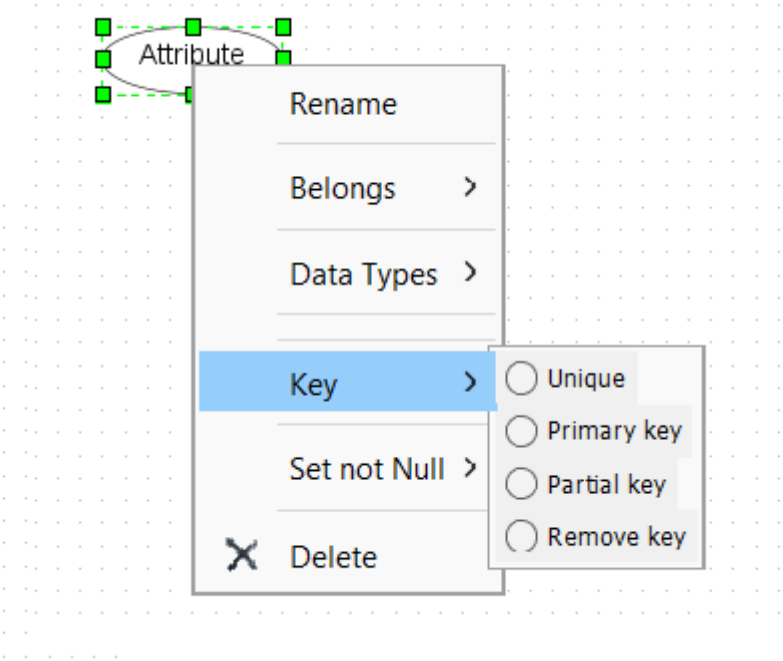
- Data Type: Η επιλογή του τύπου δεδομένων για ένα γνώρισμα σε ένα σχήμα βάσης δεδομένων επιτρέπει στον χρήστη να καθορίσει τον τύπο των δεδομένων που θα αποθηκεύονται στο γνώρισμα. Το εργαλείο έχει ως προκαθορισμένο τύπο την επιλογή INTEGER αλλά ο χρήστης μπορεί να επιλέξει οποιονδήποτε από τους παρακάτω τύπους δεδομένων:
    - INTEGER: Ακέραιοι αριθμοί.
    - BIGINT: Μεγάλοι ακέραιοι αριθμοί.
    - SMALLINT: Μικροί ακέραιοι αριθμοί.

- REAL: Πραγματικοί αριθμοί με μικρή ακρίβεια.
- DOUBLE PRECISION: Διπλής ακρίβειας πραγματικοί αριθμοί.
- DECIMAL (i, j): Δεκαδικοί αριθμοί με αριθμό ψηφίων που καθορίζεται από τα "i" (precision) και "j" (scale).
- CHARACTER (n): Χαρακτήρες με σταθερό μήκος "n".
- CHARACTER VARYING (n): Χαρακτήρες με μεταβλητό μήκος έως "n".
- BIT (n): Bit δεδομένα με σταθερό μήκος "n".
- BIT VARYING (n): Bit δεδομένα με μεταβλητό μήκος έως "n".
- BOOLEAN: Λογική τιμή (True ή False).
- XML: Αποθήκευση και διαχείριση δεδομένων XML.
- TIME: Χρονική στιγμή.
- DATE: Ημερομηνία.
- TIMESTAMP: Ημερομηνία και χρόνος.
- INTERVAL: Χρονικό διάστημα.
- OTHER (Add Type): Ο χρήστης εισάγει έναν προσαρμοσμένο τύπο δεδομένων που δεν περιλαμβάνεται στη λίστα.



Εικόνα 45: Το υπομενού του τύπου δεδομένων

- Η επιλογή Key έχει τις εξής υποκατηγορίες:
    - Primary Key: Χαρακτηρίζει ένα γνώρισμα ως το πρωτεύον κλειδί της οντότητας στην οποία ανήκει.
    - Unique: Χαρακτηρίζει ένα γνώρισμα που δεν μπορεί να πάρει διπλότυπες τιμές.
    - Partial: Το γνώρισμα αποτελεί μερικό κλειδί της ασθενούς οντότητας στην οποία ανήκει.
    - Remove: Αφαιρεί από το γνώρισμα την ιδιότητα του κλειδιού.

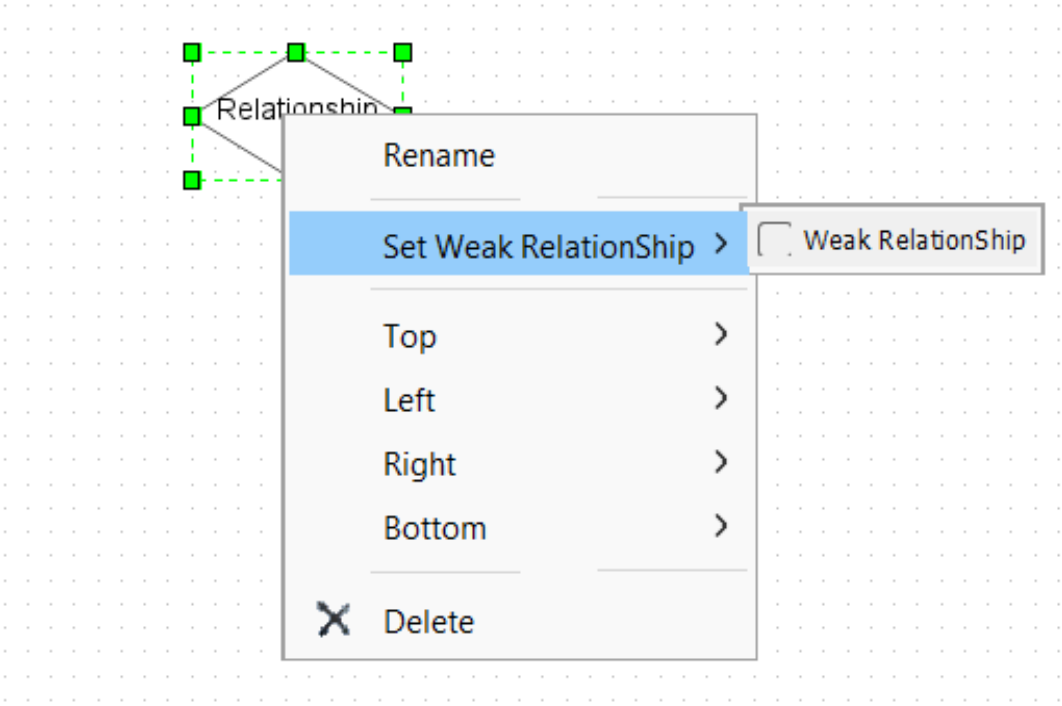


Εικόνα 46 : Το υπομενού του κλειδιού

- Set not NULL: Το γνώρισμα δεν μπορεί να έχει ως τιμή τη NULL.

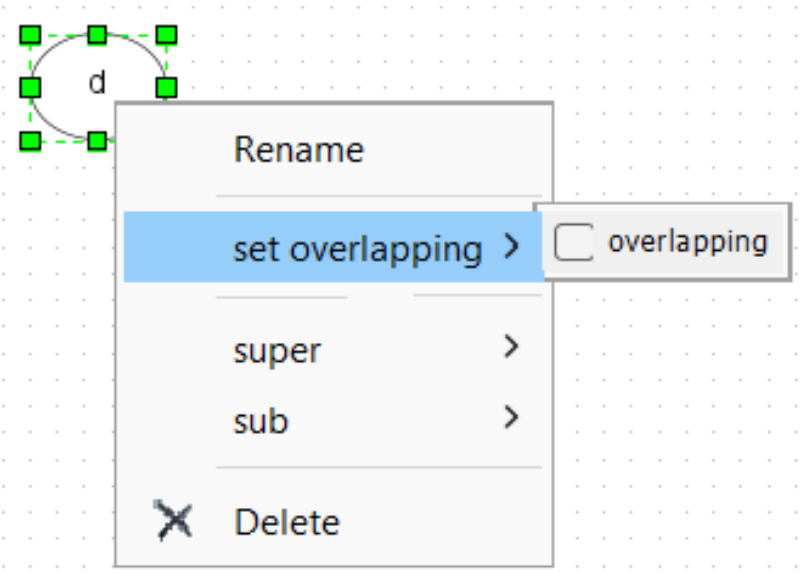
Συσχέτιση (Relationship) και Προσδιορίζουσα Συσχέτιση (Identifying Relationship)

- Top, Left, Right, Bottom: Καθεμία από αυτές τις επιλογές συνδέει μια συσχέτιση με μια οντότητα. Η σύνδεση αντιστοιχεί στην αντίστοιχη γωνία του ρόμβου από την οποία θα δημιουργηθεί το ευθύγραμμο τμήμα για να την απεικονίσει. Επίσης, υπάρχει δυνατότητα εναλλαγής του τύπου συμμετοχής μεταξύ ολικής και μερικής καθώς και του λόγου πληθικότητας, καθώς και εναλλαγής μεταξύ συσχέτισης και προσδιορίζουσας συσχέτισης.



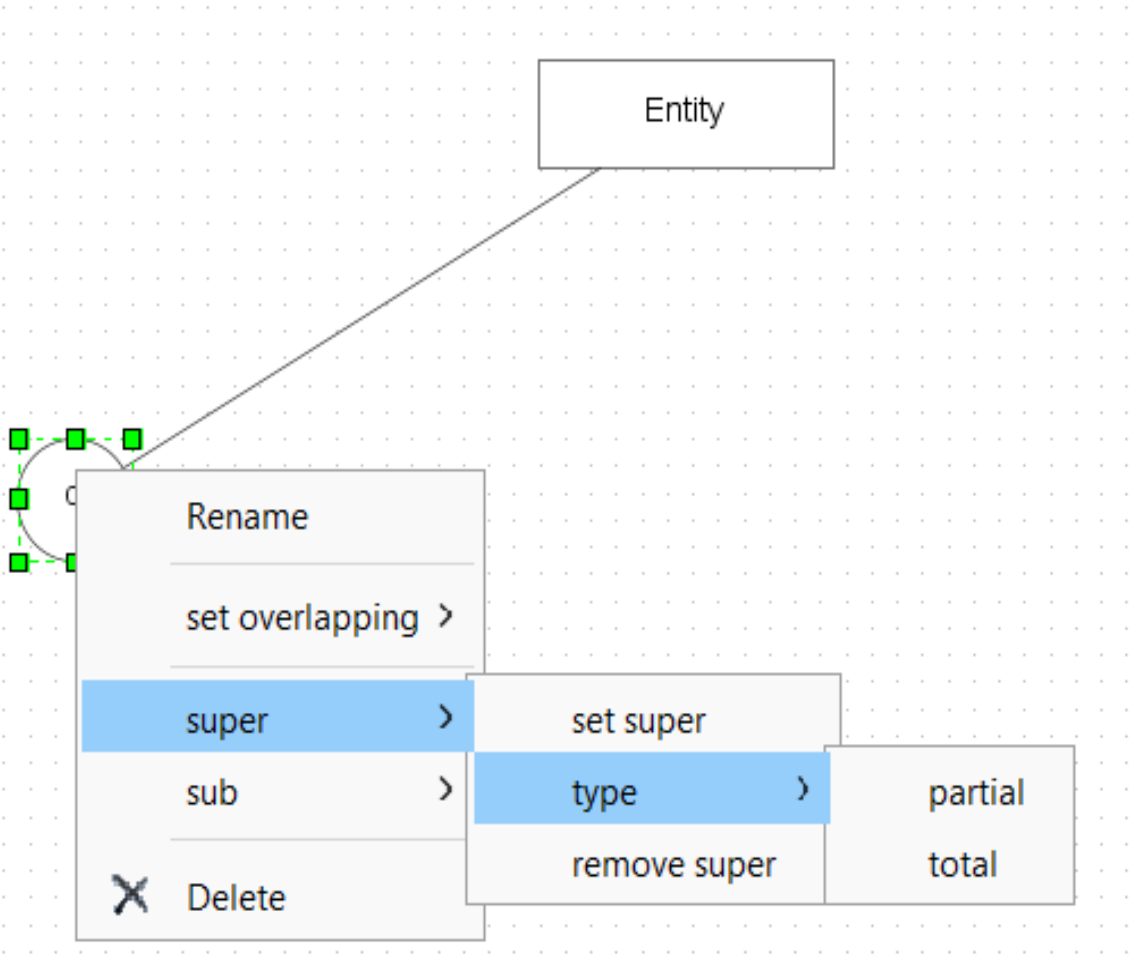
Εικόνα 47: Το αναδυόμενου μενού μίας συσχέτισης

Εξειδίκευση (Specialization), Disjoint & Overlapping



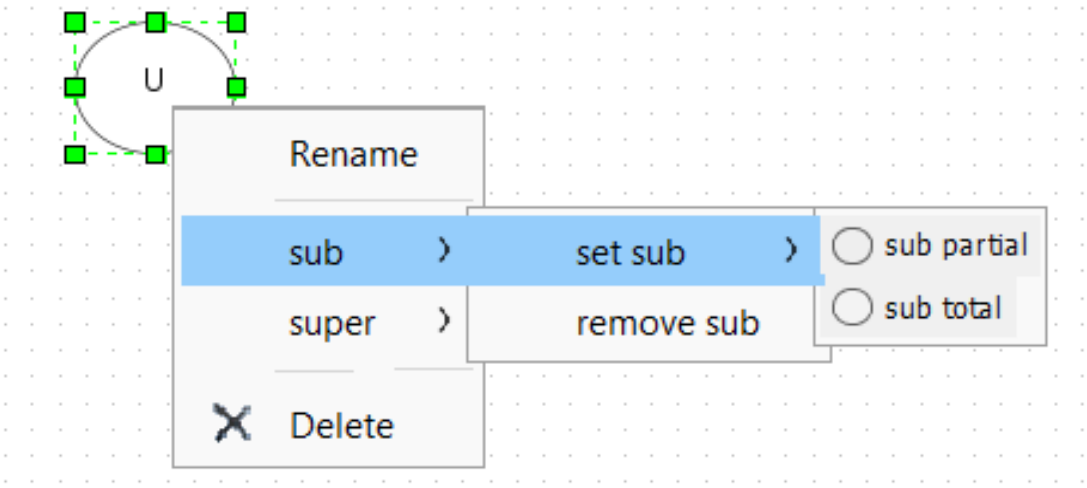
Εικόνα 48: Το αναδυόμενου μενού μια εξειδίκευσης

- Overlapping: Καθορίζει εάν πρόκειται για επικαλυπτόμενη ή όχι εξειδίκευση.
- Super: Καθορίζει την υπερκλάση της εξειδίκευσης και τη συμμετοχή αυτής. Επίσης, μέσω της επιλογής Defining Attribute ο χρήστης καλείται να επιλέξει ένα από τα γνωρίσματα της υπερκλάσης ως το ορίζον γνώρισμα της εξειδίκευσης. Μια εξειδίκευση μπορεί να συνδεθεί με έναν μόνο ισχυρό τύπο οντότητας, ο οποίος θα μετατραπεί σε υπερκλάση της εξειδίκευσης. Ωστόσο, η εξειδίκευση μπορεί να έχει απεριόριστο αριθμό υποκλάσεων. Η επιλογή Remove αφαιρεί τη σύνδεση, αλλά στην περίπτωση της υποκλάσης ο χρήστης κάνει κλικ για να διευκρινίσει την οντότητα.
- Sub: Καθορίζει τις υποκλάσεις της εξειδίκευσης και μέσω της επιλογής Defining Criteria τις προϋποθέσεις ή κανόνες που καθορίζουν ποια στοιχεία θα ανήκουν σε μια συγκεκριμένη οντότητα.



Εικόνα 49: Το αναδυόμενου μενού μιας εξειδίκευσης αναφορικά με τη συμμετοχή

Ένωση (Union)



Εικόνα 50: Το αναδυόμενου μενού μιας ένωσης

- Όπως και στην περίπτωση της εξειδίκευσης, υπάρχουν οι επιλογές Super και Sub, με τη διαφορά ότι μια ένωση μπορεί να έχει πολλές υπερκλάσεις, αλλά μόνο μία υποκλάση που είναι το αποτέλεσμα της ένωσης. Για να προσδιοριστεί ποια σύνδεση πρέπει να αφαιρεθεί στην περίπτωση της υπερκλάσης, ο χρήστης πρέπει να κάνει κλικ για να διευκρινίσει την επιθυμητή οντότητα.

5.3 Έλεγχος Σφαλμάτων

Το εργαλείο ελέγχει τη συντακτική ορθότητα του διαγράμματος μέσω της επιλογής Check for errors που βρίσκεται στο υπομενού της επιλογής Tools στην οριζόντια μπάρα. Σε περίπτωση μη εύρεσης λαθών εμφανίζει νέο παράθυρο με αντίστοιχο μήνυμα, διαφορετικά στο νέο παράθυρο εμφανίζεται η λίστα λαθών, όπου εκτός από την περιγραφή του κάθε λάθους υπάρχει ο τύπος και το όνομα του σχήματος που αφορά, για ευκολότερη και ταχύτερη διόρθωση του διαγράμματος.

Τα μηνύματα που δύνανται να εμφανιστούν είναι τα εξής:

- No errors found: Δεν εντοπίστηκαν λάθη στο διάγραμμα.

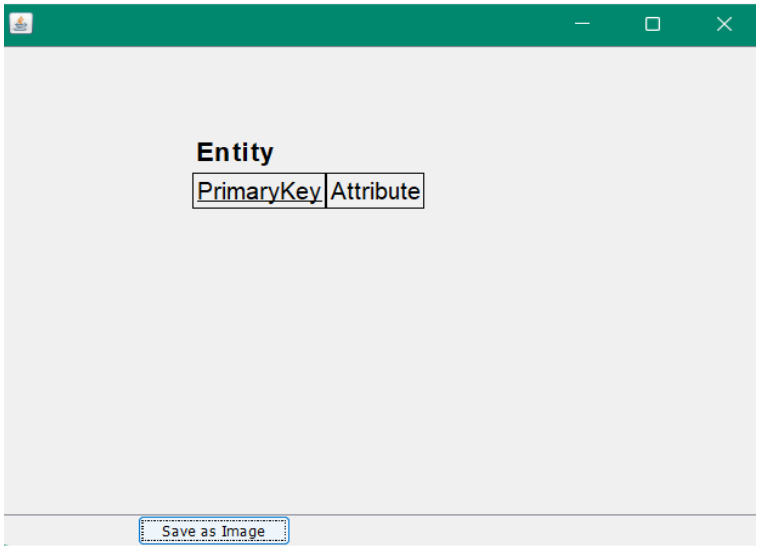
- Entity has duplicate name with another: Σε περίπτωση που εντοπιστούν τύποι οντοτήτων με το ίδιο όνομα.
- Weak entities cannot be superclasses: Σε περίπτωση που ασθενής οντότητα ορίστηκε ως υπερκλάση.
- Weak entities cannot have superclasses: Σε περίπτωση που ασθενής οντότητα ορίστηκε ως υποκλάση.
- Entity has no Attributes: Σε περίπτωση που δεν εντοπίστηκαν συνδεδεμένα γνωρίσματα στον τύπο οντοτήτων.
- Entity/Weak Entity has no Unique Attributes to be used as primary/partial key: Σε περίπτωση που δεν εντοπίστηκε γνώρισμα που έχει χαρακτηριστεί ως πρωτεύον (primary), ή μερικό(partial) κλειδί αν αναφερόμαστε σε ασθενή οντότητα ή ως μοναδικό (unique) γνώρισμα, ούτε υπερκλάση για να κληρονομηθεί από αυτή.
- Entity has multiple unique attributes, please specify the primary key: Σε περίπτωση που εντοπίστηκαν τουλάχιστον 2 μοναδικά γνωρίσματα σε ισχυρό τύπο οντότητας. Εδώ ο χρήστης καλείται να επιλέξει το πρωτεύον κλειδί.
- Weak entity has multiple unique attributes, please specify the partial key: Σε περίπτωση που εντοπίστηκαν τουλάχιστον 2 μοναδικά γνωρίσματα σε ασθενή οντότητα. Εδώ ο χρήστης καλείται να επιλέξει το μερικό κλειδί.
- Entity has multiple primary attributes, please specify only one primary key: Σε περίπτωση που εντοπίστηκαν παραπάνω του ενός κλειδιά ενώ το κλειδί είναι εξ ορισμού μοναδικό.
- Weak entity has multiple partial attributes, please specify only one partial key: Όμοια.
- Entity has partial key: Σε περίπτωση που εντοπίστηκε μερικό κλειδί συνδεδεμένο με ισχυρό τύπο οντότητας, ενώ πρέπει να είναι γνώρισμα ασθενούς οντότητας.
- Weak entity has primary key: Ανάλογα με πριν.
- Weak entity does not have identifying relationship: Σε περίπτωση που εντοπίστηκε ασθενής οντότητα η οποία δε συμμετέχει σε προσδιορίζουσα συσχέτιση.

- Attributes belonging to the same element cannot have the same name: Σε περίπτωση που εντοπίστηκαν γνωρίσματα που ανήκουν στον ίδιο τύπο οντοτήτων και έχουν το ίδιο όνομα.
- Attribute does not belong to anything: Σε περίπτωση που εντοπίστηκε γνώρισμα το οποίο δεν έχει ιδιοκτήτη.
- Attribute must be set to NOT NULL as the one that it belongs to: Σε περίπτωση που εντοπίστηκε γνώρισμα που ανήκει σε σύνθετο NOT NULL γνώρισμα και δεν έχει οριστεί ως NOT NULL.
- Relationship has duplicate name with another: Σε περίπτωση που εντοπιστούν συσχετίσεις με το ίδιο όνομα.
- Relationship does not connect to an entity: Σε περίπτωση που εντοπίστηκε συσχέτιση που δε συνδέεται με οντότητα.
- Relationship must be connected to at least 2 entities: Σε περίπτωση που εντοπίστηκε συσχέτιση που συνδέεται με μία μόνο οντότητα.
- Identifying relationship must be connected to exactly one weak entity: Σε περίπτωση που εντοπίστηκε προσδιορίζουσα συσχέτιση που δε συνδέεται με μία μόνο ασθενή οντότητα.
- Identifying relationship must be connected to at least one strong entity: Σε περίπτωση που εντοπίστηκε προσδιορίζουσα συσχέτιση που δε συνδέεται με καμία ισχυρή οντότητα.
- Identifying relationship must be connected to a weak entity through total participation: Σε περίπτωση που εντοπίστηκε ασθενής οντότητα χωρίς να έχει ολική συμμετοχή στη συσχέτιση.
- Identifying relationship must be connected to all owner entities using '1' cardinality ratio: Σε περίπτωση που εντοπίστηκε προσδιορίζουσα συσχέτιση χωρίς σύνδεση με τις ισχυρές οντότητες με λόγο πληθικότητας 1.

- Identifying relationships cannot have attributes: Σε περίπτωση που εντοπίστηκε προσδιορίζουσα συσχέτιση που έχει γνωρίσματα.
- Specialization has no superclass: Σε περίπτωση που εντοπίστηκε εξειδίκευση στην οποία δεν έχει οριστεί υπερκλάση.
- Specializations must have at least 2 subclasses: Σε περίπτωση που εντοπίστηκε εξειδίκευση που έχει λιγότερες απο 2 υποκλάσεις.
- Union has no subclass: Σε περίπτωση που εντοπίστηκε ένωση στην οποία δεν έχει οριστεί υποκλάση.
- Unions must have at least 2 superclasses: Σε περίπτωση που εντοπίστηκε ένωση που έχει λιγότερες απο 2 υπερκλάσεις.

5.4 Μετατροπή σε Σχεσιακό Σχήμα

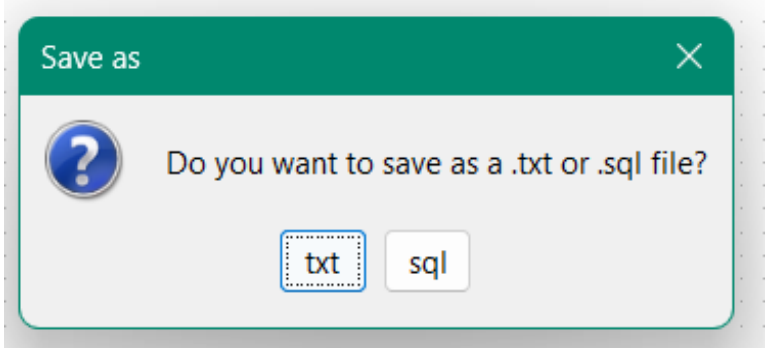
Το εργαλείο μετατρέπει το διάγραμμα σε Σχεσιακό Σχήμα μέσω της επιλογής Convert to Relational schema που βρίσκεται στο υπομενού της επιλογής Tools στην οριζόντια μπάρα. Πραγματοποιείται αυτόματος έλεγχος για τυχόν σφάλματα και στην περίπτωση μη εύρεσης λαθών εμφανίζεται νέο παράθυρο με το Σχεσιακό Σχήμα (ΣΣ). Στο τέλος του παραθύρου βρίσκεται το κουμπί **Save as Image** που δίνει στο χρήστη τη δυνατότητα αποθήκευσης του ΣΣ ως εικόνα.



Εικόνα 51: Το παράθυρο του σχεσιακού σχήματος

5.5 Παραγωγή κώδικα SQL

Το εργαλείο παράγει και αποθηκεύει τον κώδικα SQL (1999) που υλοποιεί το Σχεσιακό Σχήμα μέσω της επιλογής Convert to SQL που βρίσκεται στο υπομενού της επιλογής Tools στην οριζόντια μπάρα. Πραγματοποιείται αυτόματος έλεγχος για τυχόν σφάλματα και στην περίπτωση μη εύρεσης λαθών εμφανίζεται νέο παράθυρο για την αποθήκευση του κώδικα. Αφού ο χρήστης επιλέξει την τοποθεσία και το όνομα του αρχείου που θα αποθηκεύσει εμφανίζεται παράθυρο που του επιτρέπει την αποθήκευση είτε με τη μορφή txt είτε με την sql.



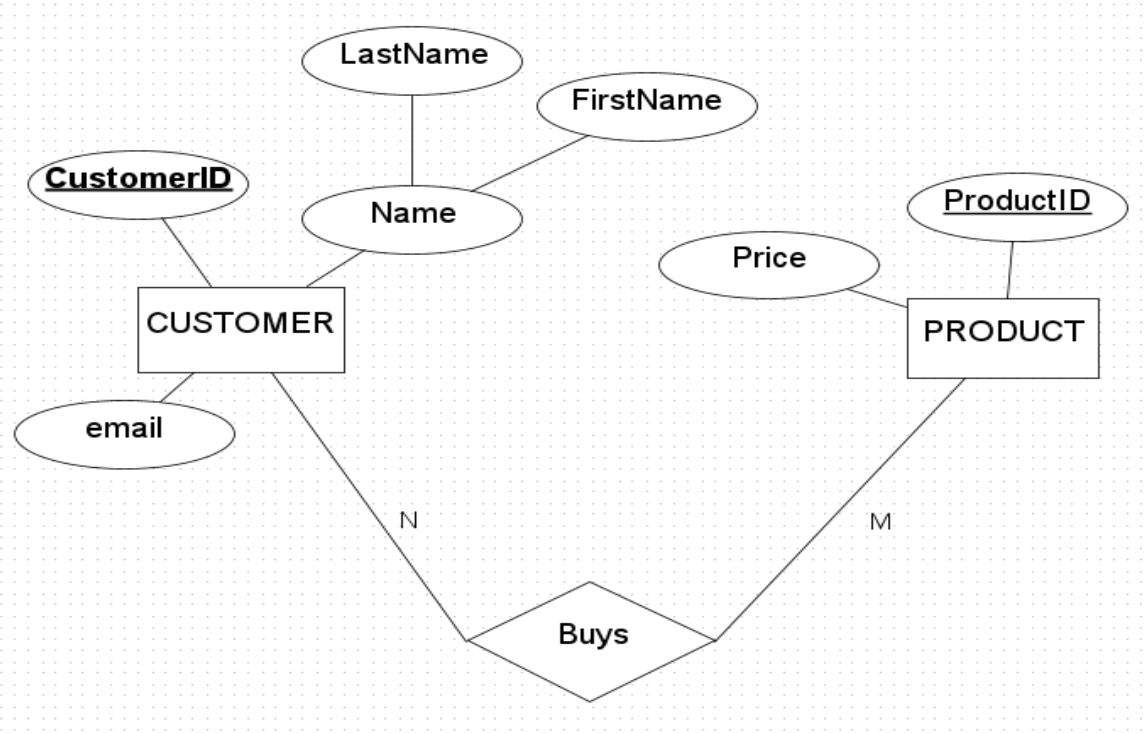
Εικόνα 52: Το παράθυρο επιλογής της μορφής για την αποθήκευση του κώδικα SQL

Στην επόμενη εικόνα βλέπουμε το αποτέλεσμα της Convert to SQL για το σχεσιακό σχήμα που απεικονίστηκε στην εικόνα 51, κάνοντας την επιλογή sql στο προηγούμενο παράθυρο.

```
1  CREATE TABLE Entity (
2      Attribute   INTEGER,
3      PrimaryKey  INTEGER NOT NULL,
4  PRIMARY KEY (PrimaryKey) );
5
```

Εικόνα 53: Ο κώδικας SQL που αντιστοιχεί στο σχεσιακό σχήμα της εικόνας 51

Ακολουθεί ένα παράδειγμα μετατροπής από το ER διάγραμμα στον κώδικα SQL.



Εικόνα 54: Το διάγραμμα ER μεταξύ των τύπων οντοτήτων Πελάτης και Προϊόν όπως και η συσχέτιση-σχέση μεταξύ τους

Το σύνολο οντοτήτων της βάσης δεδομένων σύμφωνα με το διάγραμμα της εικόνας 54 είναι Πελάτης και Προϊόν. Ο τύπος οντοτήτων Πελάτης έχει ως απλό γνώρισμα το email, ως

σύνθετο γνώρισμα το Όνομα που αποτελείται από το μικρό όνομα και το επώνυμο, ενώ ως κλειδί έχει οριστεί το γνώρισμα "CustomerID" που αντιστοιχεί στο μοναδικό αναγνωριστικό κωδικό του Πελάτη. Όμοια, ο τύπος οντοτήτων Προϊόν αποτελείται από το γνώρισμα-κλειδί "ProductID" και την τιμή του προϊόντος. Η συσχέτιση μεταξύ των 2 τύπων οντοτήτων "Αγοράζει/Αγοράζεται" έχει λόγο πληθικότητας πολλά προς πολλά διότι ένας πελάτης είναι σε θέση να αγοράσει πολλά προϊόντα, και κάθε προϊόν μπορεί να αγοραστεί από πολλούς πελάτες. Παρατηρούμε ότι στον αντίστοιχο κώδικα SQL της εικόνας 55, που προκύπτει όταν ο χρήστης επιλέξει από το μενού Options το Convert to SQL, δημιουργούνται τρεις πίνακες για τη δημιουργία της αντίστοιχης βάσης δεδομένων, δύο εκ των οποίων αφορούν τους τύπους οντοτήτων και ένας τη συσχέτιση. Η διαδικασία αυτής της μετατροπής αναλύεται σε επόμενο κεφάλαιο. Στην επόμενη εικόνα απεικονίζεται η επιλογή το αρχείο που θα προκύψει να είναι της μορφής sql και όχι txt.

```sql
CREATE TABLE CUSTOMER (
    FirstName   INTEGER,
    LastName    INTEGER,
    email   INTEGER,
    CustomerID  INTEGER NOT NULL,
PRIMARY KEY (CustomerID) );

CREATE TABLE PRODUCT (
    Price   INTEGER,
    ProductID   INTEGER NOT NULL,
PRIMARY KEY (ProductID) );

CREATE TABLE Buys (
    FK1_CustomerID  INTEGER NOT NULL,
    FK2_ProductID   INTEGER NOT NULL,
PRIMARY KEY (FK1_CustomerID, FK2_ProductID) );

ALTER TABLE Buys ADD FOREIGN KEY (FK1_CustomerID) REFERENCES CUSTOMER (CustomerID) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Buys ADD FOREIGN KEY (FK2_ProductID) REFERENCES PRODUCT (ProductID) ON DELETE CASCADE ON UPDATE CASCADE;
```

Εικόνα 55: Ο κώδικας SQL που αντιστοιχεί στο διάγραμμα της εικόνας 54

6. Συμπεράσματα και επεκτάσεις

Ο σκοπός της παρούσας διπλωματικής ήταν να δημιουργηθεί ένα ολοκληρωμένο εργαλείο τόσο για το σχεδιασμό διαγραμμάτων οντοτήτων συσχετίσεων, όσο και για τον έλεγχο της συντακτικής ορθότητας αυτών, τη μετατροπή των διαγραμμάτων σε Σχεσιακά Σχήματα καθώς και την παραγωγή κώδικα SQL για την υλοποίηση του Σχεσιακού Σχήματος.

Το εργαλείο, αξιοποιεί το συμβολισμό Chen, προσφέροντας δυνατότητες αναπαράστασης διαγραμμάτων σύμφωνα με το απλό και το εκτεταμένο μοντέλο, ενώ οι περαιτέρω λειτουργίες του το καθιστούν ιδανικό για την ανάπτυξη δεξιοτήτων στο σχεδιασμό βάσεων δεδομένων. Μέσω της αναλυτικής περιγραφής των εκάστοτε σχεδιαστικών λαθών που πιθανώς έχουν προκύψει, ο χρήστης είναι σε θέση να αυτοαξιολογήσει τον εαυτό του και να μελετήσει περαιτέρω τα σημεία εκείνα στα οποία παρουσιάζει αδυναμίες.

Αξιοποιώντας την πολυπλατφορμική φύση της Java και την υποστήριξη της JVM, εξασφαλίζεται ότι η εφαρμογή θα είναι προσβάσιμη και λειτουργική σε διάφορες πλατφόρμες και λειτουργικά συστήματα, χωρίς την ανάγκη για επανασχεδίαση ή προσαρμογή του κώδικα για κάθε σύστημα ξεχωριστά.

Ο προτιμώμενος τρόπος υλοποίησης του σχεσιακού σχήματος στη SQL:1999 είναι μέσω των πινάκων, συσχετίσεων (relationships) και περιορισμών (constraints) που ορίζονται στη βάση δεδομένων. Αυτό είναι το βασικό μοντέλο που χρησιμοποιείται σε σχεσιακές βάσεις δεδομένων, ανεξάρτητα από τη συγκεκριμένη υλοποίηση. Η PostgreSQL είναι μια υλοποίηση της SQL που ακολουθεί το πρότυπο SQL:1999. Ως σχεσιακή βάση δεδομένων, η PostgreSQL υλοποιεί το σχεσιακό μοντέλο που περιγράφηκε προηγουμένως. Ωστόσο, επιπλέον προσφέρει επεκτάσεις για αντικειμενοστραφείς δυνατότητες (object-relational) που επιτρέπουν την αποθήκευση και διαχείριση αντικειμένων (Object-Oriented features) στη βάση δεδομένων. Συνεπώς, μπορούμε να πούμε ότι η SQL:1999 υλοποιεί το σχεσιακό μοντέλο δεδομένων, ενώ η PostgreSQL υλοποιεί το αντικειμενοστραφές σχεσιακό μοντέλο δεδομένων.

Ως μελλοντική επέκταση του εργαλείου έχει προγραμματιστεί η προσθήκη της δυνατότητας παραγωγής κώδικα PostgreSQL. Αυτό σημαίνει ότι οι χρήστες του εργαλείου θα μπορούν να δημιουργούν αυτόματα κώδικα που είναι συμβατός με τη σύνταξη και τις δομές της PostgreSQL, πράγμα που καθιστά ευκολότερη τη μετάβαση από το διάγραμμα στην υλοποίηση της βάσης δεδομένων σε ένα περιβάλλον PostgreSQL.

Μία ακόμη επέκταση αφορά τη δυνατότητα επιλογής διαφορετικών συμβολισμών για την αναπαράσταση του μοντέλου οντοτήτων συσχετίσεων όπως οι Min-Max και Crow's Foot. Ο συμβολισμός Min-Max χρησιμοποιείται για να αναπαραστήσει τον βαθμό συμμετοχής μιας οντότητας σε μια συσχέτιση περικλείοντας σε παρανθέσεις τους ελάχιστους (min) και μέγιστους (max) αριθμούς των οντοτήτων που συμμετέχουν σε μια συσχέτιση. Ο συμβολισμός Crow's Foot είναι ένας άλλος γραφικός τρόπος αναπαράστασης ενός μοντέλου δεδομένων και χρησιμοποιεί βέλη για τις συσχετίσεις, ενώ ειδικά σύμβολα αναπαριστούν τους περιορισμούς (όπως "Ένα προς Ένα" ή "Ένα προς Πολλά") και τους βαθμούς συμμετοχής.

Ακόμη, την παρούσα στιγμή, το εργαλείο δεν είναι σε θέση να επεξεργαστεί διαγράμματα στα οποία περιέχονται σύνθετα γνωρίσματα που αποτελούνται από άλλα σύνθετα γνωρίσματα, δηλαδή σύνθετα γνωρίσματα δευτέρου επιπέδου. Συνεπώς, η λειτουργία αυτή προτείνεται ως μελλοντική επέκταση του εργαλείου.

Τέλος, η κανονικοποίηση του Σχεσιακού Σχήματος είναι μία ακόμη δυνατότητα που δεν υλοποιείται στο εργαλείο. Ο στόχος της κανονικοποίησης είναι να μετατρέψει ένα σχήμα σε μια κανονική μορφή, που είναι αποδοτική και αποτελεσματική και παρέχει ομαλή λειτουργία των δεδομένων.

7. Παράρτημα Α: Οδηγίες Χρήσης για την Εφαρμογή ERD tools

Σε αυτό το παράρτημα παρουσιάζονται αναλυτικά οι οδηγίες για τη χρήση της εφαρμογής

1. Εγκατάσταση και Εκκίνηση:

- Βεβαιωθείτε ότι έχετε εγκατεστημένη την Java Runtime Environment (JRE) έκδοση 18 ή νεότερη.
- Κάντε διπλό κλικ στο εκτελέσιμο αρχείο erdt.exe για να ξεκινήσετε την εφαρμογή ERD tools.
- Δεν απαιτείται διαδικασία εγκατάστασης. Το εργαλείο θα λειτουργήσει όταν εκτελεστεί το αρχείο erdt.exe

2. Περιβάλλον Χρήστη:

- Μόλις ξεκινήσει η εφαρμογή, θα εμφανιστεί το περιβάλλον χρήστη της ERD tools.
- Στο περιβάλλον χρήστη, μπορείτε να δημιουργήσετε, να επεξεργαστείτε και να διαχειριστείτε διαγράμματα. Το μενού εργαλείων για τα διαγράμματα Οντοτήτων-Συσχετίσεων και Εκτεταμένων διαγραμμάτων Οντοτήτων-Συσχετίσεων βρίσκεται στο αριστερό μέρος του παραθύρου.

3. Δημιουργία Διαγράμματος:

- Για να δημιουργήσετε ένα νέο διάγραμμα, επιλέξτε "File" > "New" από τη γραμμή μενού.
- Επιλέξτε το αντικείμενο που θέλετε να δημιουργήσετε (π.χ., οντότητα, γνώρισμα, συσχέτιση κλπ.) και σύρετε το αντικείμενο κρατώντας το αριστερό πλήκτρο του ποντικιού πατημένο και μετακινώντας το ποντίκι, ώστε να μεταφέρετε το σχήμα στον καμβά του εργαλείου (drag & drop). Τοποθετώντας το ποντίκι σε κάποιο γνώρισμα (hover) εμφανίζεται το αντίστοιχο tooltip με τον τύπο δεδομένων του γνωρίσματος. Οι τύποι αυτοί αξιοποιούνται κατά την παραγωγή του κώδικα SQL.
- Ξεκινήστε τη σχεδίαση του διαγράμματος σας, προσθέτοντας στοιχεία και συνδέοντάς τα μεταξύ τους. Κάνοντας δεξί κλικ πάνω σε κάποιο αντικείμενο του καμβά εμφανίζεται ένα αναδυόμενο (popup) μενού που περιέχει τις επιλογές που αντιστοιχούν στις διαθέσιμες λειτουργίες για το συγκεκριμένο σχήμα, όπως μετονομασία, διαγραφή του αντικειμένου, ορισμός ιδιοτήτων ή δημιουργία, επεξεργασία και διαγραφή συνδέσεων.

4. Επεξεργασία Διαγράμματος:

- Χρησιμοποιήστε τη γκάμα εργαλείων που παρέχονται στο περιβάλλον χρήστη για να επεξεργαστείτε το διάγραμμα

- Για να μετακινήστε ένα αντικείμενο σε οποιαδήποτε θέση επιθυμείτε εντός του πλαισίου, χρησιμοποιήστε τη λειτουργία του drag & drop. Κρατώντας το αριστερό πλήκτρο του ποντικιού πατημένο πάνω στο εν λόγω αντικείμενο και μετακινώντας το ποντίκι, μεταφέρετε το σχήμα στην επιθυμητή θέση και αφήστε το πλήκτρο του ποντικιού για να αποθηκευτεί το αντικείμενο στην νέα του θέση.

- Για να αλλάξετε το μέγεθος ενός σχήματος στην εφαρμογή επιλέξτε το σχήμα που θέλετε να αλλάξετε το μέγεθός του. Θα εμφανιστεί ένα πλαίσιο με σημεία ελέγχου τα οποία μπορείτε να τραβήξετε ή να σπρώξετε προς τα μέσα ή προς τα έξω, ανάλογα με το επιθυμητό μέγεθος, κρατώντας πατημένο το αριστερό πλήκτρο του ποντικιού. Όταν είστε ικανοποιημένοι με το νέο μέγεθος, απελευθερώστε το πλήκτρο του ποντικιού για να ολοκληρωθεί η αλλαγή. Κάνοντας διπλό κλικ σε κάποιο σχήμα, επαναφέρονται οι αναλογίες του ώστε να αντιπροσωπεύει το αντίστοιχο αντικείμενο του διαγράμματος οντοτήτων-συσχετίσεων.

5. Αποθήκευση και Φόρτωση Διαγράμματος:

- Αποθηκεύστε το διάγραμμά σας επιλέγοντας "File" > "Save" ή "Save as" από τη γραμμή μενού. Θα εμφανιστεί ένα νέο παράθυρο με τίτλο Save από όπου θα επιλέξετε τον τόπο αποθήκευσης. Στο κάτω μέρος του παραθύρου βρίσκεται το πεδίο File name όπου θα πληκτρολογήσετε το όνομα με το οποίο θέλετε να αποθηκευτεί το διάγραμμά σας. Πατώντας το κουμπί Save, το διάγραμμα αποθηκεύεται στην επιλεγμένη τοποθεσία, ενώ στο όνομά του προστίθεται αυτόματα η κατάληξη .erdt για να δηλωθεί ότι πρόκειται για αρχείο της εφαρμογής.

- Για να φορτώσετε ένα αποθηκευμένο διάγραμμα, επιλέξτε "File" > "Open" από το οριζόντιο μενού και θα εμφανιστεί ένα νέο παράθυρο με τίτλο Open File. Σε περίπτωση που έχετε ήδη κάποιο διάγραμμα στον καμβά σχεδίασης, η επιλογή αυτή θα εμφανίσει ένα παράθυρο ελέγχου που σας δίνει την επιλογή να επιβεβαιώσετε αν θέλετε να ανοίξετε ένα νέο αρχείο χωρίς να αποθηκεύσετε τις τρέχουσες αλλαγές σας. Έπειτα μπορείτε να περιηγηθείτε στον φάκελο όπου έχετε αποθηκεύσει αρχεία διαγραμμάτων της μορφής .erdt, να επιλέξετε το αρχείο που θέλετε

να φορτώσετε στην εφαρμογή κάνοντας αριστερό κλικ σε αυτό και τέλος να πατήσετε το κουμπί Open File.

6. Εξαγωγή και Εκτύπωση Διαγράμματος:

- Επιλέξτε "File" > "Export" για να εξαγάγετε το διάγραμμα σε μορφή αρχείου εικόνας png ή svg. Στο κάτω μέρος του παραθύρου που θα εμφανιστεί έχετε τη δυνατότητα να επιλέξετε ανάμεσα στις δύο αυτές μορφές. Αφού πληκτρολογήσετε το επιθυμητό όνομα του αρχείου, μπορείτε να το αποθηκεύσετε κάνοντας κλικ στο κουμπί Save.
- Χρησιμοποιήστε την επιλογή "File" > "Print" για να εκτυπώσετε το διάγραμμα ή να δημιουργήσετε το αντίστοιχο αρχείο pdf μέσω της επιλογής Microsoft Print to PDF.

7. Διαμόρφωση μεγέθους και προσανατολισμού σελίδας:

- Επιλέξτε "File" > "Page Setup" ώστε να διαμορφώσετε το μέγεθος και τον προσανατολισμό (οριζόντιο/κάθετο) της σελίδας στην οποία σχεδιάζονται τα διαγράμματα.

8. Τερματισμός εφαρμογής

- Για να τερματίσετε την εφαρμογή μπορείτε είτε να κάνετε κλικ στο εικονίδιο «Χ» στο πάνω δεξί τμήμα του παραθύρου, είτε μέσω του μενού επιλέγοντας "File" > "Exit". Σε περίπτωση που το διάγραμμά σας δεν έχει αποθηκευτεί θα εμφανιστεί αντίστοιχο παράθυρο ελέγχου.

9. Αναίρεση και επανάληψη

- Επιλέξτε "Edit" > "Undo" ή "Redo" ώστε να αναιρέσετε ή να επαναφέρετε μια ενέργεια που έχει πραγματοποιηθεί, όπως η διαγραφή ενός αντικειμένου, μιας σύνδεσης ή η μετονομασία ενός αντικειμένου, επαναφέροντας το διάγραμμα στην προηγούμενη κατάσταση. Οι συντομεύσεις πληκτρολογίου για αυτές τις λειτουργίες είναι 'Control+Z' για την αναίρεση (Undo) και 'Control+Y' για την επανάληψη (Redo) αντίστοιχα. Αυτός ο μηχανισμός σας επιτρέπει να ανατρέψετε ή να επαναφέρετε βήματα και να διορθώσετε πιθανά λάθη ή ανεπιθύμητες αλλαγές που έχουν γίνει στο διάγραμμα.

10. Εμφάνιση των διαγραμμάτων

- Επιλέξτε View > Page Layout για να διαμορφώσετε το πλαίσιο που καταλαμβάνει η σελίδα του διαγράμματος, μετακινώντας την προβολή του πλαισίου στο κέντρο της σελίδας. Με την

επιλογή View > Grid μπορείτε να εμφανίσετε ή να αποκρύψετε ένα πλέγμα από κουκκίδες που βοηθά στην ευκολότερη ευθυγράμμιση των αντικειμένων μέσα στο διάγραμμα. Με την επιλογή View > Outline μπορείτε να εμφανίσετε μία μικρογραφία του διαγράμματος μέσω ενός παραθύρου επισκόπησης. Το παράθυρο αυτό εμφανίζεται στο κάτω αριστερό μέρος του κεντρικού παραθύρου, όταν επιλεχθεί η λειτουργία. Αυτό επιτρέπει τη γρήγορη και πιο εύκολη μετακίνηση σε διάφορα μέρη του διαγράμματος. Με την επιλογή View > Colors μπορείτε να χρωματίσετε τα αντικείμενα του γραφήματος ανάλογα με το είδος τους.

11. Έλεγχος συντακτικής ορθότητας

- Επιλέξτε Options > Check for errors ώστε να εκτελεστεί αυτόματα ο έλεγχος για τη συντακτική ορθότητα του διαγράμματος που έχετε δημιουργήσει ή φορτώσει στην εφαρμογή. Σε περίπτωση που βρεθεί κάποιο λάθος, θα εμφανιστεί ένα νέο παράθυρο με μία αναλυτική λίστα που αναφέρει το όνομα και το είδος των αντικειμένων για τα οποία έχει εντοπιστεί σφάλμα, καθώς και περιγραφή του σφάλματος.

12. Δημιουργία σχεσιακού σχήματος

- Επιλέξτε Options > Convert to Relational schema ώστε να μετατρέψετε το διάγραμμα σε Σχεσιακό Σχήμα. Αρχικά θα γίνει αυτόματος έλεγχος για σφάλματα και σε περίπτωση που δεν υπάρχουν εμφανίζεται ένα νέο παράθυρο με το Σχεσιακό Σχήμα. Στο κάτω μέρος του παραθύρου που εμφανίζεται υπάρχει και το κουμπί Save as Image που επιτρέπει την αποθήκευση του Σχεσιακού Σχήματος σε αρχείο PNG.

13. Παραγωγή και αποθήκευση κώδικα SQL που υλοποιεί το Σχεσιακό Σχήμα

- Επιλέξτε Options > Convert to SQL ώστε να παράγετε τον κώδικα SQL που υλοποιεί το Σχεσιακό Σχήμα. Αρχικά θα γίνει αυτόματος έλεγχος για σφάλματα και σε περίπτωση που δεν υπάρχουν, παράγεται ο κώδικας SQL. Εμφανίζεται νέο παράθυρο στο οποίο μπορείτε να επιλέξετε την τοποθεσία όπου θα αποθηκεύσετε το αρχείο του κώδικα και στο τέλος έχετε τη δυνατότητα να επιλέξετε αποθήκευση είτε με τη μορφή txt είτε με την sql.

8. Παράρτημα Β: Περιγραφή πηγαίων αρχείων του κώδικα

Στο παράρτημα αυτό παρουσιάζεται μια σύντομη περιγραφή των περιεχομένων των φακέλων του κώδικα της εφαρμογής ERD tools. Ο κώδικας βρίσκεται στη διεύθυνση: https://github.com/nikoletaln/ERDtools.

examples\com\mxgraph\thesis\swing\editor: Περιλαμβάνει τις κλάσεις που έχουν να κάνουν με το σχεδιασμό, την επεξεργασία, την αποθήκευση, τον έλεγχο σφαλμάτων και τη μετατροπή σε σχεσιακό σχήμα και κώδικα SQL, όπως αυτές περιγράφτηκαν στην ενότητα 4.3 του κεφαλαίου 4.

examples\com\mxgraph\thesis\swing\images: Περιλαμβάνει τις εικόνες των σχημάτων που χρησιμοποιούνται στην εφαρμογή.

examples\com\mxgraph\thesis\swing\resources: Περιλαμβάνει τα αρχεία xml που σχετίζονται με την εμφάνιση των αντικειμένων της εφαρμογής.

examples\com\mxgraph\thesis\swing: Περιλαμβάνει την κλάση ArcLineShape η οποία είναι υπεύθυνη για το σχεδιασμό των ενώσεων μεταξύ των σχημάτων. Ακόμη, περιλαμβάνει την κλάση DoubleRhombusShape η οποία σχεδιάζει το αντικείμενο που αντιπροσωπεύει την προσδιορίζουσα συσχέτιση καθώς και την κλάση GraphEditor που αποτελεί στοιχείο γραφικού περιβάλλοντος χρήστη (GUI) και δημιουργεί το μενού με την παλέτα σχημάτων και τα αντίστοιχα tooltips αυτών.

lib: Περιέχει σε μορφή jar τη βιβλιοθήκη JGraphX.

src: Περιέχει τον πηγαίο κώδικα της βιβλιοθήκης JGraphX.

Βιβλιογραφία

Βερύκιος, Β. & Βασιλακόπουλος, Μ. (2022*). Συστήματα Βάσεων Δεδομένων* [Προπτυχιακό εγχειρίδιο]. Κάλλιπος, Ανοικτές Ακαδημαϊκές Εκδόσεις.

Doi: https://dx.doi.org/10.57713/kallipos-36

Βουλτσίδης, Δ. Μ. (2015). *Ανάπτυξη εργαλείου μετατροπής διαγραμμάτων οντοτήτων-συσχετίσεων σε σχεσιακά και αντικειμενοσχεσιακά σχήματα.* Ελληνικό Ανοιχτό Πανεπιστήμιο.

Connolly, T.M., & Begg, C. E. (2015*). Database Systems: a Practical Approach to Design, Implementation and Management* (6th edition). Pearson.

Deitel, P. J., & Deitel, H. M. (2011). *Java: How to Program* (9th Edition). Prentice Hall.

Elmasri, R., & Navathe, B. S. (2007). *Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων* (Μ. Χατζόπουλος, μετάφραση). Αθήνα: Εκδόσεις Δίαυλος.

Elmasri, R., & Navathe, B. S. (2015). *Fundamentals of Database Systems* (7th ed.). Addison-Wesley Longman Publishing Co., Inc.

Evans, S., & Warburton, B. (2014). *Java Performance: The Definitive Guide: Getting the Most Out of Your Code.* O'Reilly Media.

Evans, B. J., & Flanagan, D. (2018). *Java in a Nutshell: A Desktop Quick Reference* (7th ed.). O'Reilly Media.

Gosling, J., Joy, B., & Steele Jr., G. L. (2005). *The Java Language Specification* (3rd ed.). Addison-Wesley.

Schildt, H. (2018). *Java: The Complete Reference,* Eleventh Edition. McGraw-Hill Education.

Schildt, H. (2019). *Java: A Beginner's Guide*, Eighth Edition. McGraw-Hill Education.

Silberschatz A., Korth H.F., & Sudarshan, S. (2019). *Database System Concepts* (7th edition). McGraw-Hill.

Watt, A. (2014). *Database Design* - 2nd Edition. Open Textbook Library. https://opentextbc.ca/dbdesign01/

Ιστοσελίδα της JGraph Ltd. με τίτλο diagrams, 2023. Ανακτήθηκε από https://app.diagrams.net/

Ιστοσελίδα της Lucid Software Inc. με τίτλο Lucidchart, 2023. Ανακτήθηκε από https://www.lucid.app/

Ιστοσελίδα της ERDPlus με τίτλο ERDPlus, 2023. Ανακτήθηκε από https://erdplus.com/

Ιστοσελίδα της DrawSQL - Launch Factor Pty Ltd με τίτλο DrawSQL, 2023. Ανακτήθηκε από https://drawsql.app/

Ιστοσελίδα της Microsoft με τίτλο Visual Studio, 2023. Ανακτήθηκε από https://www.visualstudio.com

Ιστοσελίδα της SmartDraw, LLC με τίτλο SmartDraw, 2023. Ανακτήθηκε από https://www.smartdraw.com/

Βιβλιοθήκη Java για δημιουργία διαγραμμάτων. Ανακτήθηκε από https://github.com/jgraph/jgraphx

Ιστοσελίδα της sourceforge με τίτλο Launch4j Executable Wrapper, 2023. Ανακτήθηκε από https://sourceforge.net/projects/launch4j/

Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν.1599/1986, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης.