

**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**

**Βάσεις Δεδομένων**

Εξαμηνιαία εργασία (project)

Γραπτή αναφορά

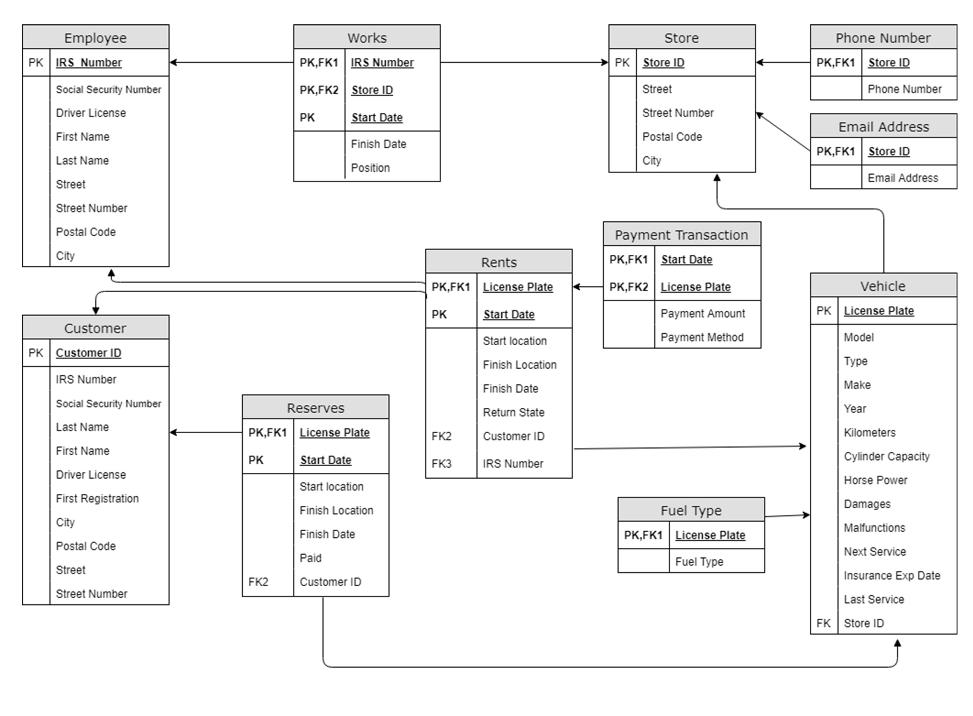
Ομάδα 51

Αριστοτέλης Βοντζαλίδης ΑΜ: 03114927

Αλέξανδρος Κούρτης ΑΜ: 03114148

**Σκοπός της εργασίας**

Η εργασία είχε σκοπό την ανάπτυξη μιας βάσης δεδομένων μιας εταιρίας ενοικίασης αυτοκινήτων. Το σχεσιακό μοντέλο που υλοποιήθηκε είναι αυτό που δόθηκε σαν λύση στη πρώτη σειρά ασκήσεων και είναι το παρακάτω:



Το project υλοποιήθηκε ως εξής: Για τη βάση δεδομένων χρησιμοποιήσαμε MySQL database Management System όπου δημιουργήσαμε τη βάση. Για το interface δημιουργήσαμε ένα Windows Presentation Foundation (WPF) με χρήση XAML για τα γραφικά και C# (C sharp) για τη σύνδεση του interface με τη βάση καθώς και για την υλοποίηση των λειτουργιών του interface. Το WPF είναι ένα desktop app που τρέχει σε συστήματα Windows χωρίς να χρειάζεται εγκατάσταση του προγράμματος.

Πλεονεκτήματα

Με το WPF μπορούμε να φτιάξουμε απλά και εύκολα ένα πολύ εύχρηστο user-interface με απλό και ευανάγνωστο κώδικα σε C#, πράγμα ιδιαίτερα σημαντικό για την συντήρηση του προγράμματος. Ακόμα το application τρέχει τοπικά χωρίς την ανάγκη εγκατάστασης ή την ανάγκη για σύνδεση στο internet. Επιπλέον για μεγάλες βάσεις δεδομένων το πρόγραμμα θα τρέχει ταχύτερα καθώς δεν θα εξαρτάτε από την ταχύτητα της σύνδεσης στο internet. Τέλος το περιβάλλον που χρησιμοποιήσαμε (Visual Studio) έκανε τη σχεδίαση και το debugging πολύ πιο εύκολη διαδικασία.

Μειονεκτήματα

Ένα από τα μειονεκτήματα είναι ότι το WPF είναι εφαρμογή που τρέχει μόνο σε Windows. Ακόμα μια αδυναμία της υλοποίησης είναι ότι η MySQL δεν μπορεί να ελέγξει τους περιορισμούς στα πεδία τιμών ή τους περιορισμούς ακεραιότητας, το οποίο επιλύθηκε σε επίπεδο κώδικα.

**Σχεδιασμός και περιορισμοί της βάσης**

Η βάση υλοποιήθηκε όπως φαίνεται στο σχήμα. Όλες οι τιμές που εισάγονται στη βάση δίνονται από τη χρήστη και ελέγχονται από το app πριν εισαχθούν στη βάση. Τα Primary keys (των πινάκων) IRS\_Number(Employee), StoreID(Store), LicensePlate(Vehicle) και CustomerID(Customer) δίνονται από το χρήστη και ελέγχονται από το πρόγραμμα ώστε οι τιμές τους να είναι ακέραιοι καθώς και να μην υπάρχουν άλλες εγγραφές με ίδιο Primary key (entity integrity). Σε περίπτωση λανθασμένης εισαγωγής το πρόγραμμα ειδοποιεί το χρήστη και δεν εισάγει στη βάση τα δεδομένα.

Για τα σύνολα συσχετίσεων έχουμε:

Για τον πίνακα Works έχουμε ξένα κλειδιά τα Employee.Irs\_number και Store.StoreID και μαζί με το StartDate πεδίο έχουμε το Primary key.

Για τον πίνακα Reserves έχουμε ξένα κλειδιά τα Vehicle.LicensePlate μαζί με το StartDate καθορίζεται το Primary key

Για τον πίνακα Rents έχουμε τα ξένα κλειδιά Vehicle.LicensePlate και μαζί με το StartDate έχουμε το Primary key.

Εκτός από αυτά τα ευρετήρια-Primary keys έχουμε και άλλα ξένα κλειδιά (όπως φαίνεται και στο σχήμα) όπως το ξένο κλειδί Vehicle.StoreID από τον πίνακα Store που συνδέει τα οχήματα με τα καταστήματα ή το Phonenumber.StoreID που συνδέει ένα νούμερο με το κατάστημα στο οποίο ανήκει.

Για όλα τα ξένα κλειδιά γίνεται ο έλεγχος ότι υπάρχει το κλειδί στο πίνακα πατέρα. Τα ευρετήρια (index) είναι όλα τα Primary Keys που δίνονται από το σχήμα και δίνονται από τον χρήστη.

Το πρόγραμμα εξασφαλίζει την ακεραιότητα οντότητας όπως προ αναφέρθηκε ελέγχοντας κάθε φορά αν το υπάρχει καταγραφή με το ίδιο κλειδί στον πίνακα ώστε να μην γίνει η εισαγωγή. Στην περίπτωση όπου το Primary key αποτελείται από παραπάνω από ένα πεδίο (βλ. πίνακας Works) ελέγχεται αντίστοιχα αν υπάρχει καταγραφή με ίδια τα συγκεκριμένα πεδία στον πίνακα.

Για την αναφορική ακεραιότητα η βάση έχει στηθεί με τέτοια constraints ώστε κάθε φορά που διαγράφεται μια εγγραφή σε πίνακα πατέρα (ή ανανεώνεται το Primary Key), να διαγράφεται (ή να αλλάζει αναλόγως) η εγγραφή στον πίνακα παιδί όπου χρησιμοποιεί ως ξένο κλειδί, το κλειδί της πρώτης. Αυτό γίνεται με όπως φαίνεται και παρακάτω αναφέροντας στον ορισμού του ξένου κλειδιού το constraint ‘ON DELETE CASCADE, ON UPDATE CASCADE’. Για παράδειγμα αν διαγράψουμε ένα κατάστημα, τα τηλέφωνα που συνδέονται με αυτό δεν έχουν πλέον λόγο ύπαρξης και διαγράφονται αυτόματα.

**DDL βάσης**

**CREATE** DATABASE Company**;**

**USE** Company

**CREATE** **TABLE** Employee **(**

IRS\_number **int** **UNIQUE** **NOT** **NULL,**

LastName **varchar(**255**)** **NOT** **NULL,**

FirstName **varchar(**255**)** **NOT** **NULL,**

DriverLicense **varchar(**255**),**

SocialSecurityNumber **varchar(**255**),**

Street **varchar(**255**),**

StreetNumber **varchar(**255**),**

PostalCode **varchar(**255**),**

City **varchar(**255**),**

**PRIMARY** **KEY** **(**IRS\_number**)**

**);**

**CREATE** **TABLE** Store **(**

StoreID **int** **UNIQUE** **NOT** **NULL,**

Street **varchar(**255**),**

StreetNumber **varchar(**255**),**

PostalCode **varchar(**255**),**

City **varchar(**255**),**

**PRIMARY** **KEY** **(**StoreID**)**

**);**

**CREATE** **TABLE** PhoneNumber **(**

StoreID **int,**

Num **long,**

**FOREIGN** **KEY** **(**StoreID**)** **REFERENCES** Store**(**StoreID**)** **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE**

**);**

**CREATE** **TABLE** Email **(**

StoreID **int,**

EmailAddress **varchar(**255**),**

**FOREIGN** **KEY** **(**StoreID**)** **REFERENCES** Store**(**StoreID**)** **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE**

**);**

**CREATE** **TABLE** Works **(**

StartDate **date** **NOT** **NULL,**

FinishDate **date,**

Position **varchar(**255**),**

IRS\_number **int,**

StoreID **int,**

**FOREIGN** **KEY** **(**IRS\_number**)** **REFERENCES** Employee**(**IRS\_number**)** **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY** **(**StoreID**)** **REFERENCES** Store**(**StoreID**)** **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE,**

**PRIMARY** **KEY** **(**IRS\_number**,** StoreID**,** StartDate**)**

**);**

**CREATE** **TABLE** Customer **(**

CustomerID **int** **UNIQUE** **NOT** **NULL,**

FirstRegistration **date,**

SocialSecutiryNumber **varchar(**255**),**

DriverLicense **varchar(**255**),**

IRS\_number **varchar(**255**),**

LastName **varchar(**255**)** **NOT** **NULL,**

FirstName **varchar(**255**)** **NOT** **NULL,**

Street **varchar(**255**),**

StreetNumber **varchar(**255**),**

PostalCode **varchar(**255**),**

City **varchar(**255**),**

**PRIMARY** **KEY** **(**CustomerID**)**

**);**

**CREATE** **TABLE** Vehicle **(**

LicensePlate **varchar(**255**)** **UNIQUE** **NOT** **NULL,**

Model **varchar(**255**),**

CarType **varchar(**255**),**

Make **varchar(**255**),**

YearMade **YEAR,**

Kilometers **int,**

CylinderCapacity **int,**

HorsePower **int,**

Damages **varchar(**255**),**

Malfunctions **varchar(**255**),**

NextService **date,**

LastService **date,**

InsuranceExpirationDate **date,**

StoreID **int,**

**PRIMARY** **KEY** **(**LicensePlate**),**

**FOREIGN** **KEY** **(**StoreID**)** **REFERENCES** Store**(**StoreID**)** **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE**

**);**

**CREATE** **TABLE** Fuel **(**

LicensePlate **varchar(**255**),**

FuelType **varchar(**255**),**

**FOREIGN** **KEY** **(**LicensePlate**)** **REFERENCES** Vehicle**(**LicensePlate**)** **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE**

**);**

**CREATE** **TABLE** Reserves **(**

StartLocation **varchar(**255**),**

FinishLocation **varchar(**255**),**

StartDate **date** **UNIQUE** **NOT** **NULL,**

FinishDate **date,**

Paid **BOOLEAN,**

CustomerID **int,**

LicensePlate **varchar(**255**),**

**FOREIGN** **KEY** **(**LicensePlate**)** **REFERENCES** Vehicle**(**LicensePlate**)** **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY** **(**CustomerID**)** **REFERENCES** Customer**(**CustomerID**)** **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE,**

**PRIMARY** **KEY** **(**LicensePlate**,** StartDate**)**

**);**

**CREATE** **TABLE** Rents **(**

StartDate **date** **UNIQUE** **NOT** **NULL,**

StartLocation **varchar(**255**),**

FinishLocation **varchar(**255**),**

FinishDate **date,**

ReturnState **varchar(**255**),**

LicensePlate **varchar(**255**),**

CustomerID **int,**

IRS\_number **int,**

**FOREIGN** **KEY** **(**CustomerID**)** **REFERENCES** Customer**(**CustomerID**)** **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY** **(**LicensePlate**)** **REFERENCES** Vehicle**(**LicensePlate**)** **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY** **(**IRS\_number**)** **REFERENCES** Employee**(**IRS\_number**)** **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE,**

**PRIMARY** **KEY** **(**LicensePlate**,** StartDate**)**

**);**

**CREATE** **TABLE** PaymentTransaction **(**

StartDate **date,**

LicensePlate **varchar(**255**),**

PaymentAmount **int,**

PaymentMethod **varchar(**255**),**

**FOREIGN** **KEY** **(**StartDate**)** **REFERENCES** Rents**(**StartDate**)** **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY** **(**LicensePlate**)** **REFERENCES** Vehicle**(**LicensePlate**)** **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE,**

**PRIMARY** **KEY** **(**LicensePlate**,** StartDate**)**

**);**

Το αρχείο το είναι γραμμένο σε notepad++ και δημιουργήσαμε τη βάση κάνοντας copy paste στο τερματικό MySQL 5.7 Command Line Client. Στο CD βρίσκεται με ονομασία mydata.sql μαζί με τα inserts.sql τα οποία παραλείπονται για συντομία.

**Views**

CD: views.sql

Μη ενημερώσιμα (δεν επιδέχονται αλλαγές λόγο των aggregate functions που περιλαμβάνονται):

Number of cars, χρησιμοποιείται κατά το πάτημα του πλήκτρου Number of Vehicles στη καρτέλα Store:

**CREATE** **VIEW** numberofcars **AS** **(**

**SELECT** store**.**storeid **as** 'Store ID'**,**

store**.**street **as** 'Street'**,**

store**.**City **as** 'City'**,**

**COUNT(**vehicle**.**storeid**)** **as** 'Number of Vehicles'

**FROM** Store

**LEFT** **JOIN** vehicle **ON** store**.**storeid**=**vehicle**.**storeid

**GROUP** **BY** store**.**storeid

**ORDER** **BY** **COUNT(**vehicle**.**storeid**)** **DESC**

**);**

Ενημερώσιμα views (επιδέχονται insert/update/delete και οι αλλαγές εφαρμόζονται στους πίνακες που αναφέρονται):

Horsepower, χρησιμοποιείται μετά το πάτημα του πλήκτρου Order by Horse Power στη καρτέλα Vehicle:

**CREATE** **VIEW** horsepower **AS** **(**

**SELECT**

HorsePower **as** "Horse Power(HP)"**,**

Model **as** "Model"**,**

Make **as** "Maker"**,**

cylindercapacity **as** "Cylinder Capacity (cc)"

**FROM** vehicle

**ORDER** **BY** horsepower **DESC);**

**Triggers**

CD: triggers.sql

**CREATE** **TRIGGER** upd\_employee **BEFORE** **UPDATE** **ON** employee

**FOR** **EACH** **ROW**

**BEGIN**

**IF** **(NEW.**street**=**''**)** **THEN**

**SET** **NEW.**street**=**'Unknown'**;**

**ELSE**

**SET** **NEW.**street **=** **NEW.**street**;**

**END** **IF;**

**IF** **(NEW.**streetnumber**=**""**)** **THEN**

**SET** **NEW.**streetnumber**=**"Unknown"**;**

**ELSE**

**SET** **NEW.**streetnumber **=** **NEW.**streetnumber**;**

**END** **IF;**

**IF** **(NEW.**postalcode**=**""**)** **THEN**

**SET** **NEW.**postalcode**=**"Unknown"**;**

**ELSE**

**SET** **NEW.**postalcode **=** **NEW.**postalcode**;**

**END** **IF;**

**IF** **(NEW.**city**=**''**)** **THEN**

**SET** **NEW.**city**=**'Unknown'**;**

**ELSE**

**SET** **NEW.**city **=** **NEW.**city**;**

**END** **IF;**

**IF** **(NEW.**SocialSecurityNumber**=**''**)** **THEN**

**SET** **NEW.**SocialSecurityNumber**=**'Unknown'**;**

**ELSE**

**SET** **NEW.**SocialSecurityNumber**=New.**SocialSecurityNumber**;**

**END** **IF;**

**END**

**CREATE** **TRIGGER** creat\_employee **BEFORE** **INSERT** **ON** employee

**FOR** **EACH** **ROW**

**BEGIN**

**IF** **(NEW.**street**=**'' **OR** **new.**street**=null)** **THEN**

**SET** **NEW.**street**=**'Unknown'**;**

**ELSE**

**SET** **NEW.**street **=** **NEW.**street**;**

**END** **IF;**

**IF** **(NEW.**streetnumber**=**""**)** **THEN**

**SET** **NEW.**streetnumber**=**"Unknown"**;**

**ELSE**

**SET** **NEW.**streetnumber **=** **NEW.**streetnumber**;**

**END** **IF;**

**IF** **(NEW.**postalcode**=**""**)** **THEN**

**SET** **NEW.**postalcode**=**"Unknown"**;**

**ELSE**

**SET** **NEW.**postalcode **=** **NEW.**postalcode**;**

**END** **IF;**

**IF** **(NEW.**city**=**''**)** **THEN**

**SET** **NEW.**city**=**'Unknown'**;**

**ELSE**

**SET** **NEW.**city **=** **NEW.**city**;**

**END** **IF;**

**IF** **(NEW.**SocialSecurityNumber**=**''**)** **THEN**

**SET** **NEW.**SocialSecurityNumber**=**'Unknown'**;**

**ELSE**

**SET** **NEW.**SocialSecurityNumber**=New.**SocialSecurityNumber**;**

**END** **IF;**

**END**

Τα triggers εκτελούνται αυτόματα από τη βάση κάθε φορά που συμβαίνει κάποιο γεγονός. Τα trigger μπορούν να εξασφαλίζουν για παράδειγμα ότι κάθε φορά που υπάρχει εισαγωγή σε κάποιο πίνακα, το primary key που εισάγεται δεν χρησιμοποιείτε ήδη από κάποια άλλη εγγραφή. Σε εμάς ο έλεγχος αυτός γίνεται από το πρόγραμμα. Τα δικά μας trigger ελέγχουν στον πίνακα employee για κάθε insert (1o trigger) ή update (2o trigger) που γίνεται αν δόθηκε σε κάποιο από τα πεδία της διεύθυνσης ή το πεδίο SocialSecurityNumber κενή συμβολοσειρά. Σε περίπτωση που έχει δοθεί κενή συμβολοσειρά ή ο χαρακτήρας null τότε η τιμή του πεδίου ορίζεται ως Unknown. Η χρήση των triggers εδώ είναι προαιρετική για τη βάση και απλά εξασφαλίζουν μια καλύτερη εμπειρία στο χρήστη.

**SQL queries**

Χρησιμοποιήθηκαν πολλά queries τα οποία δίνονται στη βάση και επιστρέφονται στο πρόγραμμα μέσω της συνάρτησης που δημιουργήσαμε στο αρχείο App, RunCommand(string) η οποία δέχεται από το χρήστη μια συμβολοσειρά (το ερώτημα) και το μεταφέρει στη βάση και εμφανίζει αυτό που επιστρέφει στο DataGrid.

**1o Query:**

Ερώτημα το οποίο απλά επιλέγει και εμφανίζει τα περιεχόμενα του πίνακα employee στην εκκίνηση του προγράμματος:

**SELECT** \* **FROM** Εmployee;

**2o Query:**

Ερώτημα το οποίο περιλαμβάνεται στο view numberofcars και επιστρέφει όλα τα καταστήματα (Left join) μαζί με τον αριθμό των οχημάτων που αντιστοιχεί στο καθένα, αφού τα ομαδοποιεί με βάση το storeID (κλειδί ώστε να φαίνονται όλα) και τα κατατάσσει σε φθίνουσα σειρά ως προς τον αριθμό των οχημάτων τους.

**SELECT** store**.**storeid **as** 'Store ID'**,**

store**.**street **as** 'Street'**,**

store**.**City **as** 'City'**,**

**COUNT(**vehicle**.**storeid**)** **as** 'Number of Vehicles'

**FROM** Store

**LEFT** **JOIN** vehicle **ON** store**.**storeid**=**vehicle**.**storeid

**GROUP** **BY** store**.**storeid

**ORDER** **BY** **COUNT(**vehicle**.**storeid**)** **DESC**

**3o Query:**

Το συγκεκριμένο ερώτημα καλείται μετά το πάτημα του πλήκτρου View Employees στην καρτέλα Store. Μας επιστρέφει όλα τις εγγραφές του πίνακα Works, για τις οποίες το storeID είναι ίδιο με το StoreID του Store που είχαμε επιλέξει από το Grid κατά το πάτημα του πλήκτρου. Επίσης εκτός από τα πεδία του πίνακα Works μας εμφανίζει και το πεδίο lastname από τον πίνακα Employee με βάση την αντιστοιχία του IRS\_number (ξένο κλειδί στον Works και Primary στον Employee).

**SELECT** works**.\*,**

employee**.**lastname **AS** 'Surname'

**FROM** Works

**INNER** **JOIN** employee **ON** employee**.**irs\_number **=** works**.**irs\_number

**WHERE** storeid**=**"+ \_drv["storeid"].ToString() + "**;**

**4o Query:**

Το παρακάτω ερώτημα χρησιμοποιείται κατά το πάτημα του πλήκτρου Citizens per city στην καρτέλα Employee, όπου αφού ο χρήστης εισάγει μια τιμή στο pop up παράθυρο, εμφανίζει τις πόλεις σε φθίνουσα σειρά πληθυσμού, μαζί με τον πληθυσμό της κάθε πόλης και το όνομα της, με ελάχιστη τιμή πληθυσμού την τιμή που εισήγαγε ο χρήστης (μεταβλητή i).

**SELECT**

**COUNT(**irs\_number**)** **AS** 'Population'**,**

city **as** 'City'

**FROM** employee

**GROUP** **BY** city

**HAVING** **(COUNT(**IRS\_NUMBER**)>=**"+ i + "**)**

**ORDER** **BY** **(COUNT(**IRS\_NUMBER**))** **DESC;**

**5o Query:**

Το παρακάτω ερώτημα χρησιμοποιείται για να εμφανίσει τα οχήματα που έχουν χρονολογία κατασκευής μικρότερη (πιο καινούργια) του μέσου όρου χρονολογίας κατασκευής όλων των οχημάτων.

**SELECT** **\***

**FROM** Vehicle **WHERE** yearmade**>(**

**SELECT** **AVG(**yearmade**)** **FROM** Vehicle**);**