ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

ΑΝΑΦΟΡΑ ΕΞΑΜΗΝΙΑΙΑΣ ΕΡΓΑΣΙΑΣ

Εαρινό εξάμηνο 2022-2023

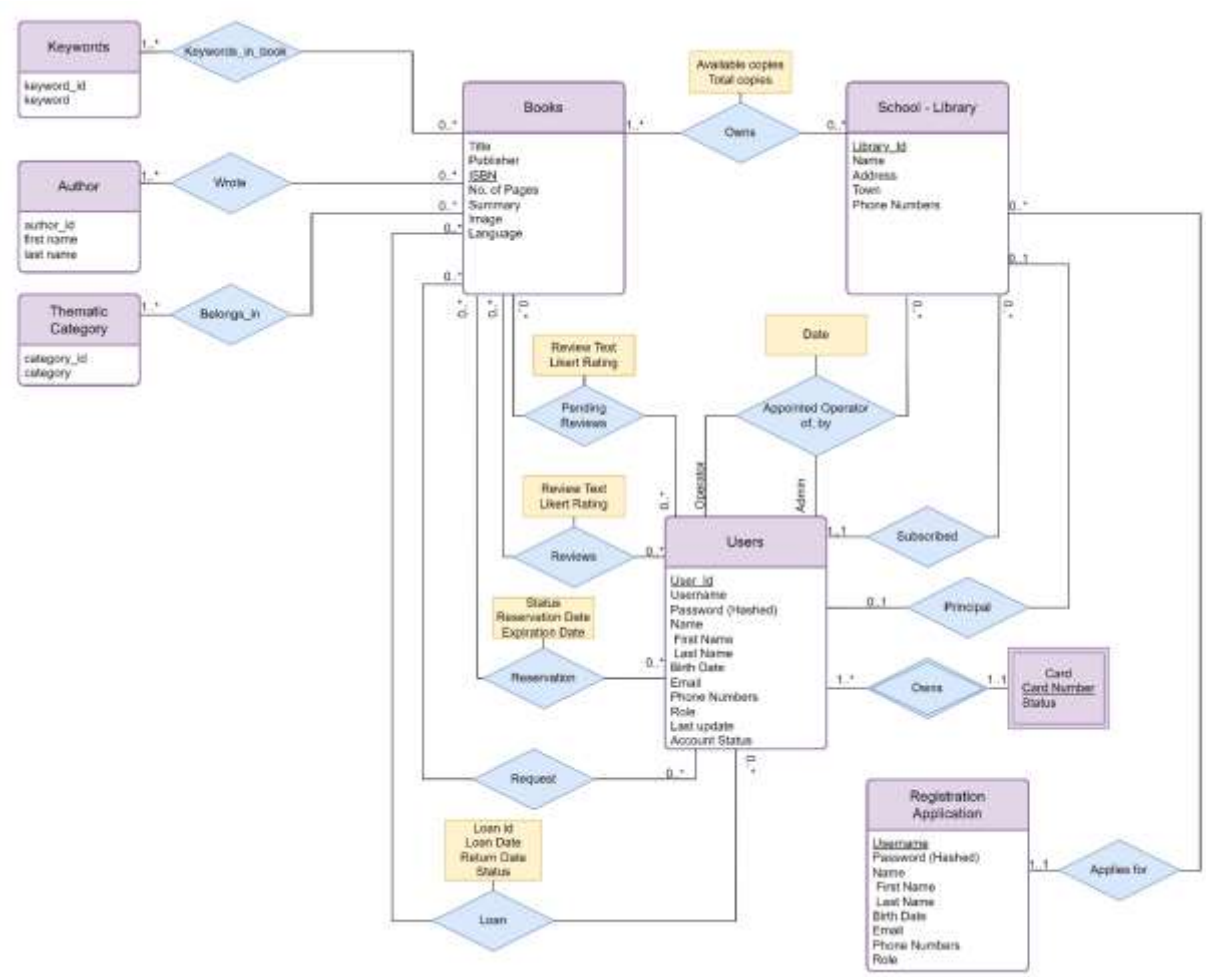
ΟΜΑΔΑ 106

Κοπίτας Χρυσόστομος	03120136
Λουκοβίτης Σπυρίδων	03120120
Σπηλιώτης Αθανάσιος	03120175

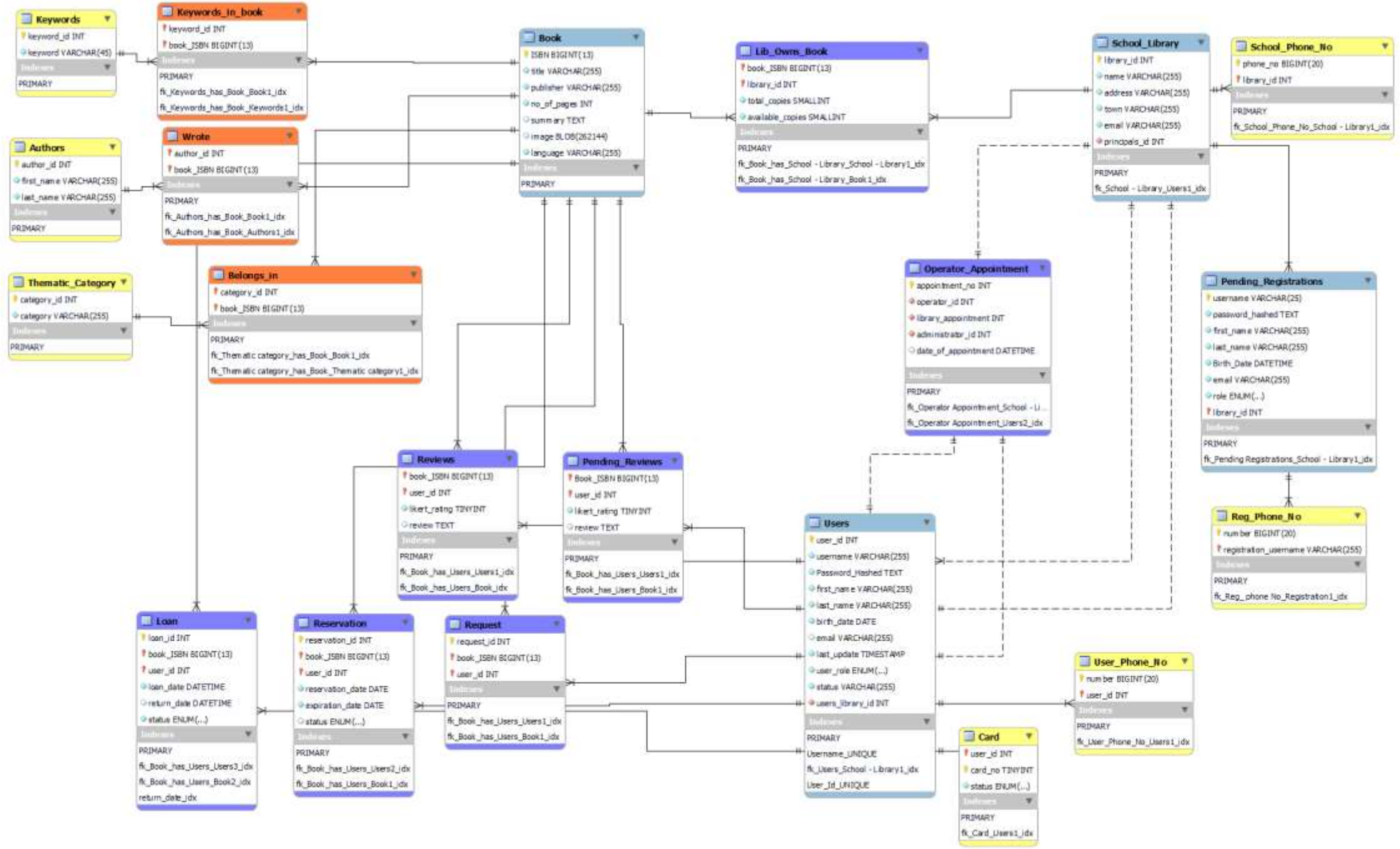
(Τα διαγράμματα μπορεί να μη φαίνονται λόγω compression, αλλά τα αντίστοιχα links στα ονόματα οδηγούν στα ίδια διαγράμματα στο git repo)

1.1 ER & και Σχεσιακό διάγραμμα

ER:



Σχεσιακό Διάγραμμα:


Keywords
keyword_id INT
keyword VARCHAR(45)
Indexes
PRIMARY
Keywords_In_book
keyword_id INT
book_ISBN BIGINT(13)
Indexes
PRIMARY
fk_Keywords_has_Book_Book1_idx
fk_Keywords_has_Book_Keywords1_idx
Book
ISBN BIGINT(13)
title VARCHAR(255)
publisher VARCHAR(255)
no_of_pages INT
summary TEXT
image BLOB(262144)
language VARCHAR(255)
Indexes
PRIMARY
Lib_Owns_Book
book_ISBN BIGINT(13)
library_id INT
total_copies SMALLINT
available_copies SMALLINT
Indexes
PRIMARY
fk_Book_has_School - Library_School - Library1_idx
fk_Book_has_School - Library_Book1_idx
School_Library
library_id INT
name VARCHAR(255)
address VARCHAR(255)
town VARCHAR(255)
email VARCHAR(255)
principals_id INT
Indexes
PRIMARY
fk_School - Library_Users1_idx
School_Phone_No
phone_no BIGINT(20)
library_id INT
Indexes
PRIMARY
fk_School_Phone_No_School - Library1_idx
Authors
author_id INT
first_name VARCHAR(255)
last_name VARCHAR(255)
Indexes
PRIMARY
Wrote
author_id INT
book_ISBN BIGINT(13)
Indexes
PRIMARY
fk_Authors_has_Book_Book1_idx
fk_Authors_has_Book_Authors1_idx
Thematic_Category
category_id INT
category VARCHAR(255)
Indexes
PRIMARY
Belongs_in
category_id INT
book_ISBN BIGINT(13)
Indexes
PRIMARY
fk_Thematic category_has_Book_Book1_idx
fk_Thematic category_has_Book_Thematic category1_idx
Operator_Appointment
appointment_no INT
operator_id INT
library_appointment INT
administrator_id INT
date_of_appointment DATETIME
Indexes
PRIMARY
fk_Operator Appointment_School - Li...
fk_Operator Appointment_Users2_idx
Pending_Registrations
username VARCHAR(25)
password_hashed TEXT
first_name VARCHAR(255)
last_name VARCHAR(255)
Birth_Date DATETIME
email VARCHAR(255)
role ENUM(...)
library_id INT
Indexes
PRIMARY
fk_Pending Registrations_School - Library1_idx
Reg_Phone_No
number BIGINT(20)
registration_username VARCHAR(255)
Indexes
PRIMARY
fk_Reg_phone No_Registration1_idx
Reviews
book_ISBN BIGINT(13)
user_id INT
likert_rating TINYINT
review TEXT
Indexes
PRIMARY
fk_Book_has_Users_Users1_idx
fk_Book_has_Users_Book_idx
Pending_Reviews
Book_ISBN BIGINT(13)
user_id INT
likert_rating TINYINT
review TEXT
Indexes
PRIMARY
fk_Book_has_Users_Users1_idx
fk_Book_has_Users_Book1_idx
Users
user_id INT
username VARCHAR(255)
Password_Hashed TEXT
first_name VARCHAR(255)
last_name VARCHAR(255)
birth_date DATE
email VARCHAR(255)
last_update TIMESTAMP
user_role ENUM(...)
status VARCHAR(255)
users_library_id INT
Indexes
PRIMARY
Username_UNIQUE
fk_Users_School - Library1_idx
User_Id_UNIQUE
User_Phone_No
number BIGINT(20)
user_id INT
Indexes
PRIMARY
fk_User_Phone_No_Users1_idx
Loan
loan_id INT
book_ISBN BIGINT(13)
user_id INT
loan_date DATETIME
return_date DATETIME
status ENUM(...)
Indexes
PRIMARY
fk_Book_has_Users_Users3_idx
fk_Book_has_Users_Book2_idx
return_date_idx
Reservation
reservation_id INT
book_ISBN BIGINT(13)
user_id INT
reservation_date DATE
expiration_date DATE
status ENUM(...)
Indexes
PRIMARY
fk_Book_has_Users_Users2_idx
fk_Book_has_Users_Book1_idx
Request
request_id INT
book_ISBN BIGINT(13)
user_id INT
Indexes
PRIMARY
fk_Book_has_Users_Users1_idx
fk_Book_has_Users_Book1_idx
Card
user_id INT
card_no TINYINT
status ENUM(...)
Indexes
PRIMARY
fk_Card_Users1_idx

Περιγραφή Βάσης

Στη Βάση Δεδομένων περιέχονται 3 βασικές οντότητες. Αυτές είναι:

- Τα βιβλία
- Οι βιβλιοθήκες
- Οι χρήστες

Οι 5 επιπλέον οντότητες που παρουσιάζονται στο ER διάγραμμα είναι:

- Οι αιτήσεις για την εγγραφή χρήστη
- Η/Οι καρτά/ες χρήστη (εξαρτημένη οντότητα)
- Λέξεις κλειδιά που εμφανίζονται στα βιβλία
- Συγγραφείς που γράφουν βιβλία
- Θεματικές κατηγορίες στις οποίες ανήκουν βιβλία

Οι σχέσεις που φαίνονται στο ER διάγραμμα είναι:

- Ο δανεισμός
- Η κράτηση
- Η αίτηση για δανεισμό
- Η αξιολόγηση βιβλίου
- Η ανάθεση καθηκόντων χειριστή βιβλιοθήκης από διαχειριστή
- Η εγγραφή σε μια βιβλιοθήκη
- Η ιδιότητα διευθυντή ενός σχολείου
- Η ιδιοκτησία μιας κάρτας βιβλιοθήκης
- Η συσχέτιση μιας εγγραφής με τη βιβλιοθήκη για την οποία ο (μελλοντικός) χρήστης κάνει αίτηση
- Η συσχέτιση ανάμεσα σε λέξη κλειδί και βιβλίο
- Η συσχέτιση ανάμεσα σε συγγραφέα και βιβλίο
- Η συσχέτιση ανάμεσα σε θεματική κατηγορία και βιβλίο

Υποθέσεις κατά την υλοποίηση:

Υποθέσαμε ότι:

- Κάθε χρήστης μπορεί να ανήκει σε μια και μόνο βιβλιοθήκη
- Κάθε χρήστης μπορεί να δανειστεί βιβλία μόνο από τη βιβλιοθήκη στην οποία είναι εγγεγραμμένος
- Υπάρχει μόνο ένας Operator ανά βιβλιοθήκη
- Ο διευθυντής της βιβλιοθήκης είναι χρήστης και όχι απλά ένα attribute της βιβλιοθήκης
- Ένας χρήστης μπορεί να αξιολογήσει βιβλία ακόμα και αν δεν τα έχει δανειστεί από το σύστημα
- Κατά την αποδοχή ενός registration τυπώνεται αυτόματα η πρώτη κάρτα του χρήστη
- Τα available copies μιας βιβλιοθήκης δεν μπορούν ποτέ να είναι λιγότερα από μηδέν ή περισσότερα από τα total copies. Συνεπώς δεν μπορεί μια βιβλιοθήκη να αφαιρέσει από τα total copies της, τα αντίτυπα τα οποία είναι δανεισμένα αυτή τη στιγμή
- Δεν μπορούν να αλλάξουν τα δεδομένα ενός δανεισμού που έχει ήδη καταχωρηθεί (έτσι δεν μπορεί να αλλάξει μια καθυστερημένη επιστροφή βιβλίου για να μπορεί κάποιος να προσπεράσει τους κανόνες δανεισμού, εκτός αν διαγραφεί ο δανεισμός)
- Μια κράτηση δεν μπορεί να αλλάξει άπαξ και είναι expired για τους ίδιους λόγους
- Κάθε μαθητής πρέπει να είναι νεότερος από 18 χρονών και οποιοσδήποτε άλλος ρόλος πρέπει να είναι μεγαλύτερος

- Ο operator μπορεί να δανειστεί με τα ίδια constraints που περιγράφονται για τον teacher, ενώ ο admin δεν έχει περιορισμό πέραν των φυσικών αντιτύπων της βιβλιοθήκης του.

Indexing Βάσης Δεδομένων

Παρατηρούμε κατ' αρχάς ότι το MySQL Workbench δημιουργεί indices για τα εξής:

- Primary Keys. Αυτό είναι πολύ βοηθητικό, αφού το primary key είναι το βασικό αναγνωριστικό μιας εισόδου σε ένα table και γι αυτό χρησιμοποιείται συχνά.
- Columns with UNIQUE Constraints. Όταν μια στήλη ενός table έχει το UNIQUE constraint και προσπαθούμε να εισάγουμε σε αυτό, πρέπει να ελεγχθεί ότι η τιμή που πάμε να βάλουμε δεν υπάρχει ήδη. Αυτό μπορεί να γίνει πολύ πιο γρήγορα (ανάλογα με την υλοποίηση) αν έχουμε κάποιο index σε αυτή τη στήλη (παράδειγμα είναι το username στους χρήστες).
- Foreign Keys: Τα foreign keys είναι οι στήλες που δημιουργούν, σε μεγάλο βαθμό, τη βασική λογική της βάσης δεδομένων. Είναι αυτά που δείχνουν πώς οι πίνακες συσχετίζονται μεταξύ τους. Γι αυτό το λόγο θέλουμε πολύ συχνά να ελέγξουμε τις τιμές τους. Βασικότερο παράδειγμα είναι το inner join, κατά το οποίο ελέγχουμε την ισότητα ενός foreign key με κάθε γραμμή ενός άλλου πίνακα. Γι αυτό το λόγο, θέλουμε τα searches πάνω στα foreign keys να γίνονται αποδοτικά

Πέραν αυτών των indices προστέθηκε άλλο ένα manually για τη συγκεκριμένη βάση δεδομένων. Αυτό είναι το loan_date_idx στον πίνακα Loan. Σε κάποια από τα ζητούμενα της εκφώνησης χρειάζεται να βρούμε loans τα οποία έχουν περάσει την προθεσμία παράδοσής τους. Ο έλεγχος αυτός γίνεται βάσει το loan_date με μια ανισοτική σχέση, οπότε ένα index μπορεί να βοηθήσει (αν και ο optimizer μπορεί να αποφασίσει να μη το χρησιμοποιήσει) στο να εκτελούνται πιο γρήγορα τα queries αυτά.

1.2 DDL & DML script

DDL scripts

Στο git repo στον φάκελο Setup υπάρχουν το DDL script "ddl_of_semester.sql". Αυτό διαγράφει τη βάση αν υπάρχει ήδη κάποια με το ίδιο όνομα, δημιουργεί τη βάση, όλους τους πίνακες, τα triggers, τα views και τα procedures. Εκεί ορίζονται τα primary keys, foreign keys, constraints. Παρακάτω φαίνεται αποσπασματικά η δημιουργία μερικών tables:

```
-- -----------------------------------------------------
-- Table `semester_project`.`Users`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `semester_project`.`Users` (
  `user_id` INT NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(255) NOT NULL,
  `Password_Hashed` TEXT NOT NULL,
```

```
  `first_name` VARCHAR(255) NOT NULL,
  `last_name` VARCHAR(255) NOT NULL,
  `birth_date` DATE NOT NULL,
  `email` VARCHAR(255) ,
  `last_update` DATE NOT NULL DEFAULT (CURRENT_DATE),
  `user_role` ENUM('Admin', 'Operator', 'Teacher', 'Student') NOT NULL,
  `user_status` VARCHAR(255) NOT NULL,
  `users_library_id` INT NOT NULL,
  PRIMARY KEY (`user_id`),
  UNIQUE INDEX `Username_UNIQUE` (`username` ASC) VISIBLE,
  INDEX `fk_Users_School - Library1_idx` (`users_library_id` ASC) VISIBLE,
  UNIQUE INDEX `User_Id_UNIQUE` (`user_id` ASC) VISIBLE,
  CONSTRAINT `fk_User_Library_id`
    FOREIGN KEY (`users_library_id`)
    REFERENCES `semester_project`.`School_Library` (`library_id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `semester_project`.`Book`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `semester_project`.`Book` (
  `ISBN` BIGINT(13) NOT NULL,
  `title` VARCHAR(255) NOT NULL,
  `publisher` VARCHAR(255) NOT NULL,
  `no_of_pages` INT NOT NULL,
  `summary` TEXT NULL,
  `image` BLOB(262144) NULL,
  `language` VARCHAR(255) NOT NULL DEFAULT 'English',
  PRIMARY KEY (`ISBN`))
ENGINE = InnoDB;
```

DML script

Στο git repo στον φάκελο setup υπάρχει το DML script με όνομα dml_of_semester.sql στο οποίο περιέχονται όλα τα αρχικά δεδομένα που κάνουν populate τη βάση. Τα δεδομένα δημιουργήθηκαν αυτόματα από python scripts που μπορούν να βρεθούν στο git repo. Τα βιβλία και οι πληροφορίες τους έγιναν scrape από τη σελίδα του Ευδόξου με χρήση python script το οποίο βρίσκεται επίσης στο git repo

Σημείωση: Κάθε βιβλίο έχει τη δυνατότητα να έχει και μια φωτογραφία, η οποία είναι το εξώφυλλό του. Για να επιτευχθεί αυτό υπάρχουν 3 συνήθεις τρόποι.

1. Οι φωτογραφίες να μπουν στη βάση με κάποιο encoding ως binary
2. Οι φωτογραφίες να είναι αποθηκευμένες τοπικά, αλλά εκτός της βάσης, ενώ στη βάση να αποθηκεύονται τα paths για να βρεθούν
3. Οι φωτογραφίες να μην αποθηκεύονται, αλλά να αποθηκεύονται links στο ίντερνετ από τα οποία μπορούν να βρεθούν

Ως ομάδα πήραμε την απόφαση να ακολουθήσουμε την πρώτη επιλογή. Αν και δεν είναι η πιο συνήθης επιλογή και μάλλον όχι η πιο αποδοτική από άποψης ταχύτητας (μια μόνο εγγραφή παίρνει πλέον

πολλά pages αντί να έχουμε δεκάδες ανά page), θεωρήσαμε ότι ήταν η επιλογή που θα μας βοηθούσε να μάθουμε περισσότερα πράγματα.
Δυστυχώς για να φορτωθούν οι φωτογραφίες, το dml script δεν είναι αρκετό, αλλά χρειάζεται να τρέξουμε ένα ακόμα python script, με όνομα insert_photos.py, που βρίσκεται στον φάκελο Setup.

Τα queries που ζητούνται από το μέρος 3 της εκφώνησης βρίσκονται στην εφαρμογή app_main.py . Καθώς η εφαρμογή είναι πάνω από 3000 γραμμές κώδικα, τα sql queries παρατίθενται και εδώ. Καθώς τα queries δέχονται και input από τις φόρμες, παρατίθεται όλος ο κώδικας της python για το κάθε ένα, καθώς αλλιώς, μάλλον δεν θα έβγαζαν πολύ νόημα. Σε πολλές περιπτώσεις μας δόθηκε η οδηγία τα queries να τρέχουν και μόνο με κάποια από τα inputs, οπότε στον κώδικα της εφαρμογής υπάρχουν πάνω από ένα cases για κάποια queries. Σε αυτές τις περιπτώσεις, στην αναφορά συμπεριλαμβάνεται το γενικότερο case:

3.1.1

```
if year and month:
                query = """
                    SELECT School_Library.name, library_id, COUNT(*) AS
TotalLoans
                    FROM Loan
                    INNER JOIN Users ON Loan.user_id = Users.user_id
                    INNER JOIN School_Library ON Users.users_library_id =
School_Library.library_id
                    WHERE YEAR(Loan.loan_date) = %s
                        AND MONTHNAME(Loan.loan_date) = %s
                    GROUP BY library_id;
                """
                params = (year, month)
                cursor.execute(query, params)
                result = cursor.fetchall()
```

3.1.2

Συγγραφείς:

```
query = """
                SELECT DISTINCT CONCAT(Authors.first_name, ' ',
Authors.last_name) AS author_name
                FROM Authors
                INNER JOIN Wrote ON Authors.author_id = Wrote.author_id
                INNER JOIN Book ON Wrote.Book_ISBN = Book.ISBN
                INNER JOIN Belongs_in ON Book.ISBN = Belongs_in.Book_ISBN
                INNER JOIN Thematic_Category ON Belongs_in.category_id =
Thematic_Category.category_id
                WHERE Thematic_Category.category = %s;
            """
            cursor.execute(query, (category,))
            result = cursor.fetchall()
```

Δάσκαλοι:

```
query = """
                SELECT DISTINCT CONCAT(Users.first_name,' ',Users.last_name)
AS teacher_name
                FROM Users
                INNER JOIN Loan ON Users.user_id = Loan.user_id
                INNER JOIN Book ON Book.ISBN = Loan.book_ISBN
                INNER JOIN Belongs_in ON Book.ISBN = Belongs_in.book_ISBN
                INNER JOIN Thematic_Category ON Belongs_in.category_id =
Thematic_Category.category_id
                WHERE Thematic_Category.category = %s
                AND Users.user_role = 'Teacher';
            """
            cursor.execute(query, (category,))
            result = cursor.fetchall()
```

3.1.3

```
query = """
                SELECT Users.user_id, Users.first_name, Users.last_name,
COUNT(*) AS num_books_borrowed
                FROM Users
                INNER JOIN Loan ON Users.user_id = Loan.user_id
                WHERE Users.user_role = 'Teacher'
                  AND TIMESTAMPDIFF(YEAR, CURDATE(),Users.birth_date ) < 40
                GROUP BY Users.user_id, Users.first_name, Users.last_name
                ORDER BY num_books_borrowed DESC;
            """
            cursor.execute(query)
            result = cursor.fetchall()
```

3.1.4

```
query = """
               SELECT Authors.first_name, Authors.last_name
                FROM Authors
                WHERE Authors.author_id NOT IN (
                    SELECT DISTINCT Wrote.author_id
                    FROM Wrote
                    INNER JOIN Loan ON Wrote.book_ISBN = Loan.book_ISBN
                );
            """
            cursor.execute(query)
            result = cursor.fetchall()
```

3.1.5

```
query = """

            SELECT u.users_library_id, u.user_id, u.first_name, u.last_name,
SL.loan_count
                FROM Users u
                INNER JOIN (
                    SELECT U.users_library_id, COUNT(*) AS
loan_count
                    FROM Loan L
                    INNER JOIN Users U ON U.user_id =
L.user_id
                    WHERE YEAR(L.loan_date) = %s
                    GROUP BY U.users_library_id
                    HAVING loan_count > 20
                ) AS SL ON u.users_library_id =
SL.users_library_id
                WHERE u.user_role = 'Operator'
               ORDER BY SL.loan_count DESC;
            """



            cursor.execute(query, (year,))
            result = cursor.fetchall()
```

3.1.6

Query:

```
query = '''
            SELECT category1, category2, pair_count
            FROM top_categories
            ORDER BY pair_count DESC
            LIMIT 3;
        '''
        cursor.execute(query)
        results = cursor.fetchall()
```

View:

```
CREATE VIEW top_categories AS
SELECT T1.category AS category1, T2.category AS category2, COUNT(*) AS
pair_count
        FROM (
            SELECT BI1.category_id AS category_id1, BI2.category_id AS
category_id2
            FROM Belongs_in BI1
```

```
            INNER JOIN Belongs_in BI2 ON BI1.book_ISBN = BI2.book_ISBN AND
BI1.category_id < BI2.category_id
            GROUP BY BI1.book_ISBN, BI1.category_id, BI2.category_id
            HAVING COUNT(*) = 1
        ) AS Pairs
        INNER JOIN Thematic_Category T1 ON Pairs.category_id1 =
T1.category_id
        INNER JOIN Thematic_Category T2 ON Pairs.category_id2 =
T2.category_id
        GROUP BY category1, category2
        ORDER BY pair_count DESC;
```

3.1.7

```
query = """
                SELECT A.author_id, A.first_name, A.last_name
                FROM Authors A
                WHERE (SELECT COUNT(*) FROM Wrote W WHERE W.author_id =
A.author_id) <
                      (SELECT COUNT(*) - 5 FROM Wrote GROUP BY author_id
ORDER BY COUNT(*) DESC LIMIT 1);
            """
            cursor.execute(query)
            result = cursor.fetchall()
```

3.2.1

```
query = """
                SELECT Book.title, GROUP_CONCAT(DISTINCT Authors.first_name
SEPARATOR ', ') AS author_first_names,
                GROUP_CONCAT(DISTINCT Authors.last_name SEPARATOR ', ') AS
author_last_names,
                GROUP_CONCAT(DISTINCT Thematic_Category.category SEPARATOR
', ') AS categories, LOB.total_copies
                FROM Book
                INNER JOIN Wrote ON Book.ISBN = Wrote.book_ISBN
                INNER JOIN Authors ON Wrote.author_id = Authors.author_id
                INNER JOIN Belongs_in ON Book.ISBN = Belongs_in.book_ISBN
                INNER JOIN Thematic_Category ON Belongs_in.category_id =
Thematic_Category.category_id
                INNER JOIN Lib_Owns_Book LOB ON Book.ISBN = LOB.book_ISBN
                WHERE LOB.library_id = %s
            """
            params = [library_id]

            if title:
                query += "AND Book.title LIKE %s "
```

```
                params.append(f"%{title}%")

            if category:
                query += "AND Thematic_Category.category LIKE %s "
                params.append(f"%{category}%")

            if name:
                query += "AND (Authors.first_name LIKE %s OR
Authors.last_name LIKE %s) "
                params.extend([f"%{name}%", f"%{name}%"])

            if copies:
                query += "AND LOB.total_copies >= %s "
                params.append(copies)

            query += "GROUP BY Book.ISBN"

            cursor.execute(query, params)
            result = cursor.fetchall()
```

3.2.2

```
query = """
            SELECT Users.first_name, Users.last_name, DATEDIFF(CURDATE(),
Loan.loan_date)-7 AS delay_days
            FROM Users
            INNER JOIN Loan ON Users.user_id = Loan.user_id
            WHERE Loan.return_date IS NULL

            AND DATEDIFF(CURDATE(), Loan.loan_date ) > 7
            AND (Users.first_name LIKE %s)
            AND (Users.last_name LIKE %s)
            AND (DATEDIFF(CURDATE(), Loan.loan_date) > %s + 7 OR %s = '')

            """

            params = (f"%{first_name}%", f"%{last_name}%", delay_days,
delay_days)

            cursor.execute(query, params)
            result = cursor.fetchall()
```

3.2.3

```
if user_id and category:
                query = """
```

```
                    SELECT Users.user_id, Thematic_Category.category,
AVG(Reviews.likert_rating) AS average_rating
                    FROM Users
                    INNER JOIN Reviews ON Users.user_id = Reviews.user_id
                    INNER JOIN Belongs_in ON Reviews.book_ISBN =
Belongs_in.book_ISBN
                    INNER JOIN Thematic_Category ON Belongs_in.category_id =
Thematic_Category.category_id
                    WHERE(Users.user_id = %s OR %s = '')
                    AND (Thematic_Category.category = %s OR %s = ' ')
                    GROUP BY Users.user_id, Thematic_Category.category;
                """
                cursor.execute(query, (user_id, user_id, category,
category))
                result = cursor.fetchall()
```

Τα ερωτήματα 3.3.1 και 3.3.2 είναι υλοποιημένα ως σελίδες και λειτουργίες της εφαρμογής οπότε δεν είναι απλά sql queries
Συγκεκριμένα το 3.3.2 είναι σπασμένο στις σελίδες “past loans” και “active loans” των οποίων τα queries είναι τα εξής:

```
query = "SELECT book_ISBN, return_date, status from Loan where user_id = %s
and (status = 'Returned' or status = 'Late Returned');"
user_id = str(session['user'])
params = (user_id,)
```

```
query = "SELECT book_ISBN, status from Loan where user_id = %s and (status =
'Active' or status = 'Late Active');"
user_id = str(session['user'])
params = (user_id,)
```

1.3 User Manual

Το User Manual, λόγω του μεγέθους του παρατίθεται ξεχωριστά στο git repo με όνομα “User_Manual.pdf”.

Σημείωση:
Για την περιήγηση στην εφαρμογή θα χρειαστείτε τα credentials κάποιων χρηστών. Παρακάτω παρατίθενται τα στοιχεία κάποιων χρηστών που δημιουργήσαμε:

Username	Password	Role	Library id
Student1	password	Student	1
Teacher1	password	Teacher	1

Operator1	password	Operator	1
Admin1	password	Admin	1

Καθώς και μερικών που δημιουργήθηκαν αυτόματα από άλλες βιβλιοθήκες:

Username	Password	Role	Library id
MasonSanchez611	MasonSanchez611	Student	1
JamesWhite847	JamesWhite847	Student	2
IsabellaWalker672	IsabellaWalker672	Student	3
OliverHarris274	OliverHarris274	Student	4
LiamBrown861	LiamBrown861	Student	5
MiaWilliams399	MiaWilliams399	Teacher	1
ElijahBrown315	ElijahBrown315	Teacher	2
MiaLee592	MiaLee592	Teacher	3
ElijahWilliams166	ElijahWilliams166	Teacher	4
ElijahMartin363	ElijahMartin363	Teacher	5
Operator1	Operator1	Operator	1
MiaWilson972	MiaWilson972	Operator	2
SophiaMitchell273	SophiaMitchell273	Operator	3
AvaJackson431	AvaJackson431	Operator	4
NoahGarcia872	NoahGarcia872	Operator	5

1.4 Οδηγίες Εγκατάστασης

Το repository της βάσης δεδομένων στο github:
https://github.com/SpyrosLkv/Database_Project

Οδηγίες εγκατάστασης εφαρμογής:

Βήμα 0

Η εφαρμογή έχει αναπτυχθεί σε περιβάλλον Linux. Πριν ακολουθήσετε τα παρακάτω βήματα για το installation βεβαιωθείτε ότι βρίσκεστε σε περιβάλλον Linux με εγκατεστημένη την python 3 και το pip3. Τα modules που χρειάζονται και δεν είναι native στις τελευταίες εκδόσεις της python 3 αναφέρονται στο requirements.txt και η εγκατάστασή τους περιγράφεται στις παρακάτω οδηγίες.

Βήμα 1

Για την εγκατάσταση της εφαρμογής χρειάζεται να γίνει clone το git repo σε έναν τοπικό φάκελο. Μεταβείτε μέσω terminal στο working directory που επιθυμείτε και τρέξτε την εντολή git clone https://github.com/SpyrosLkv/Database_Project.git

Βήμα 2

Βρείτε και μεταβείτε στον φάκελο Setup. Για να εγκαταστήσουμε τη βάση χρειαζόμαστε έναν sql server (συγκεκριμένα χρησιμοποιήσαμε τη mysql). Τρέξτε στο terminal την εντολή pip3

install -r requirements.txt στο περιβάλλον που θα δουλέψετε. Έτσι θα κατεβούν όλες οι βιβλιοθήκες της python 3 που χρειάζονται και δεν είχαν εγκατασταθεί προηγουμένως.

Βήμα 3

Για την εγκατάσταση της βάσης, συνδεόμαστε στο terminal της mysql και τρέχουμε τα scripts create_semester_project.sql και dml_of_semester.sql με αυτή τη σειρά. Μπορείτε να βγείτε από το terminal της sql τώρα.

Βήμα 4

Καθώς δεν μπορούμε να ξέρουμε με τι username, host, password θα θέλει κάποιος να τρέξει την εφαρμογή, έχουμε δημιουργήσει ένα configuration file. Στον φάκελο που βρίσκεστε θα βρείτε ένα αρχείο config.conf:

```
# config.conf

MYSQL_HOST = localhost
MYSQL_USER = root
MYSQL_PASSWORD = toyot2002
MYSQL_DB = semester_project
```

Σε αυτό το σημείο μπορείτε να αλλάξετε τον HOST, USER, PASSWORD για να τρέξει στον δικό σας σέρβερ (μην αλλάξετε το όνομα της βάσης).

Βήμα 5

Ενώ είστε ακόμα στον φάκελο Setup, τρέξτε την εντολή python3 ./insert_photos.py

Βήμα 6

Βγείτε από τον φάκελο Setup, βρείτε και μεταβείτε στον φάκελο Project και τρέξτε την εντολή python3 ./app_main.py

Βήμα 7

Επισκεφτείτε από κάποιον browser το link http://127.0.0.1:5000 (ή το αντίστοιχο link που θα τυπωθεί στο terminal). Η αρχική σελίδα της εφαρμογής πρέπει να εμφανίζεται.