

**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

**ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ**

ΑΝΑΦΟΡΑ ΕΞΑΜΗΝΙΑΙΑΣ ΕΡΓΑΣΙΑΣ

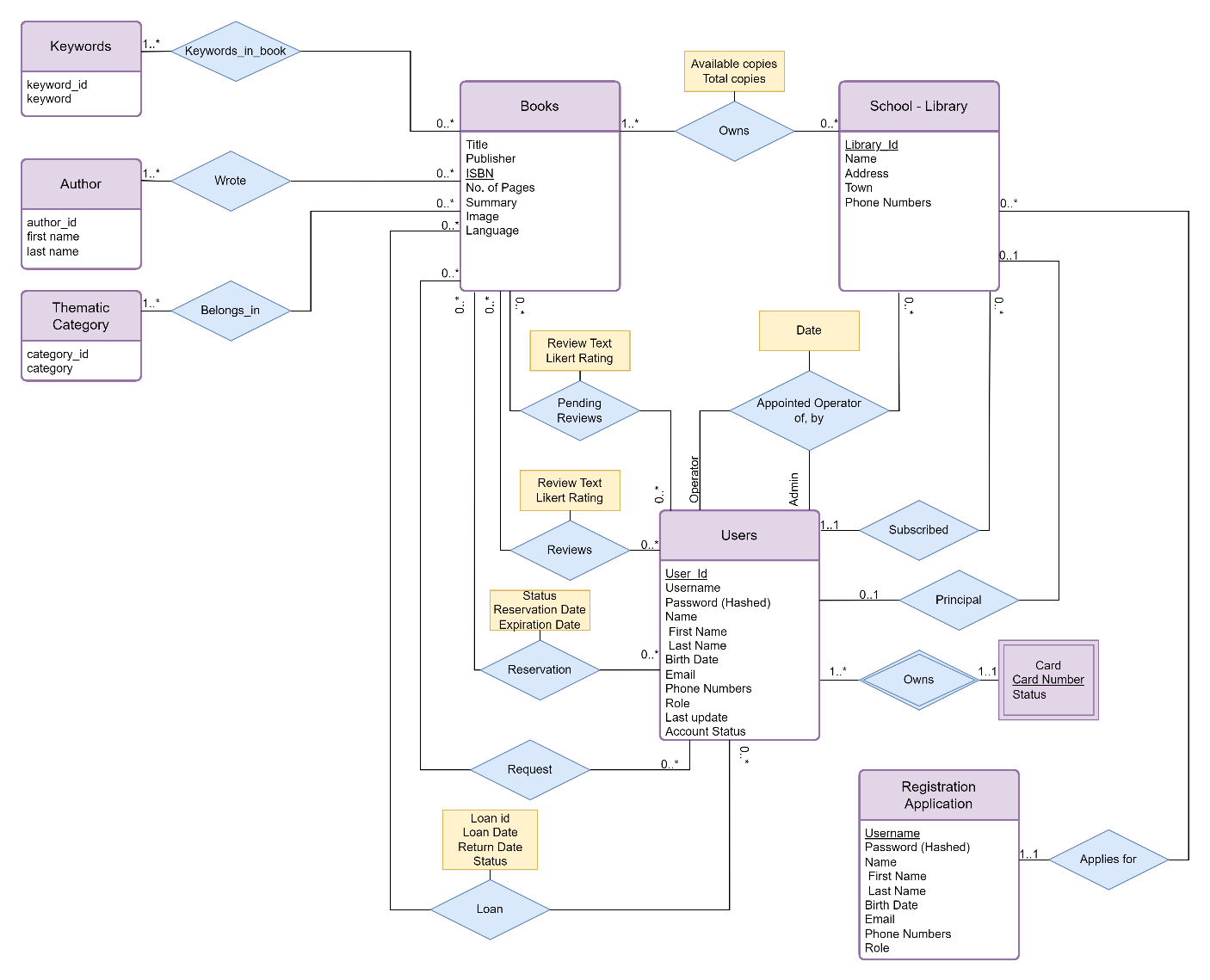
Εαρινό εξάμηνο 2022-2023

ΟΜΑΔΑ 106

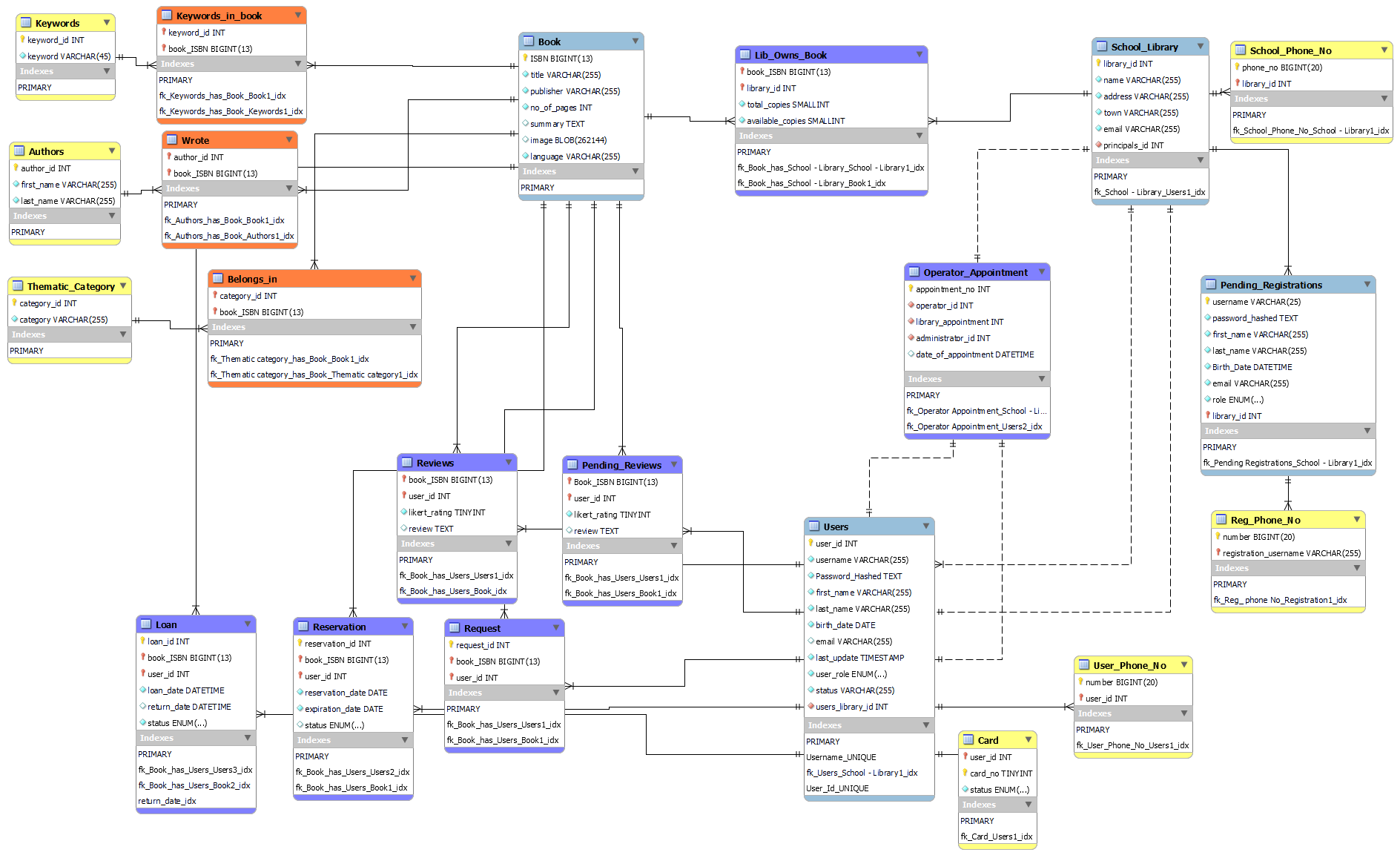
|  |  |
| --- | --- |
| Κοπίτας Χρυσόστομος | 03120136 |
| Λουκοβίτης Σπυρίδων | 03120120 |
| Σπηλιώτης Αθανάσιος | 03120175 |

* 1. ER & και Σχεσιακό διάγραμμα

ER:



Σχεσιακό Διάγραμμα:



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. 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                  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';              """                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. User Manual

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

* 1. 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

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

Βρείτε και μεταβείτε στον φάκελο Setup. Για να εγκαταστήσουμε τη βάση χρειαζόμαστε έναν sql server (συγκερκιμένα χρησιμοποιήσαμε τη mysql). 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.

Βήμα 4

Βγείτε από το terminal της mysql και, ενώ είστε ακόμα στον φάκελο Setup, τρέξτε την εντολή python3 ./insert\_photos.py

Βήμα 5

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

**Σημείωση:** Η εφαρμογή της python αρχικά επιχειρεί να δημιουργήσει μια σύνδεση ανάμεσα στον εαυτό της και τον server της mysql. Ο κώδικας στον οποίο γίνεται αυτό είναι ο εξής:

|  |
| --- |
| *# MySQL configurations*  app.config['MYSQL\_HOST'] = 'localhost'  app.config['MYSQL\_USER'] = 'root'  app.config['MYSQL\_PASSWORD'] = 'toyot2002'  app.config['MYSQL\_DB'] = 'semester\_project'  app.config['MYSQL\_CONNECT\_TIMEOUT'] = 300  app.secret\_key = 'VideoTapes'  mysql = MySQL(app) |

Καθώς ο κωδικός για τον server που θα τρέξει ο κώδικας είναι διαφορετικός, αυτή η εντολή θα πρέπει να αλλάξει.

Βήμα 6

Επισκεφτείτε από κάποιον browser το link <http://127.0.0.1:5000>. Η αρχική σελίδα της εφαρμογής πρέπει να εμφανίζεται