ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Χριστοδουλάκης Σταύρος 03120890
Γεωργιάδη Δάφνη 03120189
Κροντήρα Έλενα 03120440

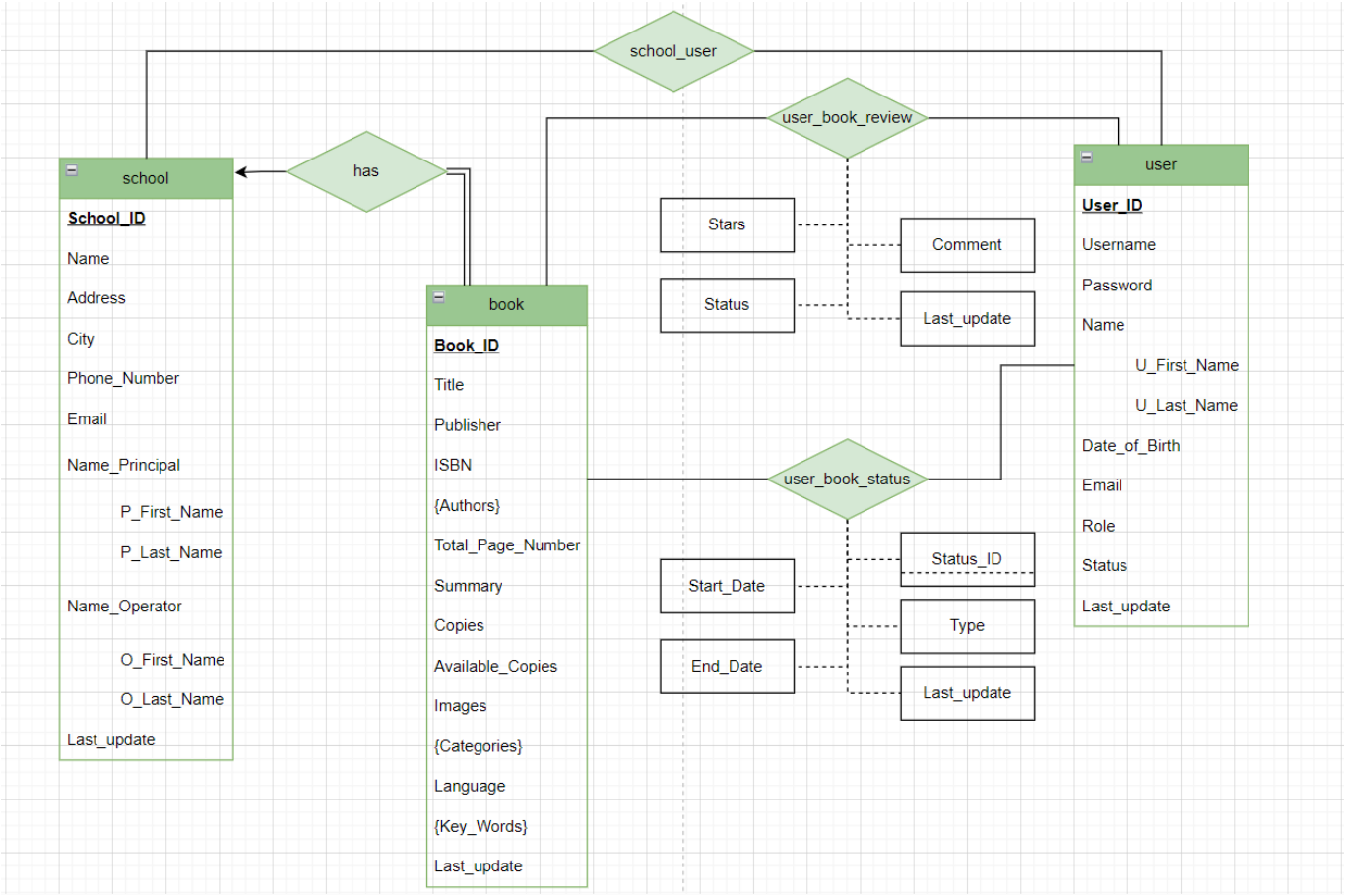
Αναφορά Εργασίας Βάσεων

ΠΕΡΙΕΧΟΜΕΝΑ

LINKS:

GitHub repository link : https://github.com/StavrosXr/Database-Library
Εκεί μπορείτε να βρείτε όλα τα απαραίτητα αρχεία τα οποία αναλύονται παρακάτω

Με βάση τα ερωτήματα της άσκησης παρακάτω φαίνεται το ER diagram.



Στο ER παρατηρούνται τρία entities (school, user, book) τα οποία έχουν τα attributes που ζητήθηκαν στην εκφώνηση καθώς και ορισμένα που κρίναμε οι ίδιοι ότι είναι χρήσιμα για την υλοποίηση της άσκησης, όπως ‘Role’ για τους users και ‘Available_Copies’ για τα books.
Τα attributes που είναι multi values παρουσιάζονται με αυτόν τον τρόπο:{attributes}

Οι σχέσεις μεταξύ των entities είναι οι ακόλουθες:
user – school: παρατηρείται μία N to M σχέση καθώς

- Σε ένα σχολείο μπορούν να ανήκουν πολλοί users
- Ένας user μπορεί:
    - Να ανήκει σε ένα σχολείο (περίπτωση μαθητών)
    - Να ανήκει σε παραπάνω από ένα σχολείο (περίπτωση καθηγητών)
    - Να μην ανήκει σε κάποιο σχολείο (περίπτωση διαχειριστών)

school – book: παρατηρείται μία N to 1 σχέση καθώς:

- Σε ένα σχολείο μπορούν να υπάρχουν πολλά βιβλία
- Ένα βιβλίο (συγκεκριμένο book_id) ανήκει υποχρεωτικά σε ένα σχολείο

Δεν γίνεται κάποιο βιβλίο να μην ανήκει σε κάποιο σχολείο

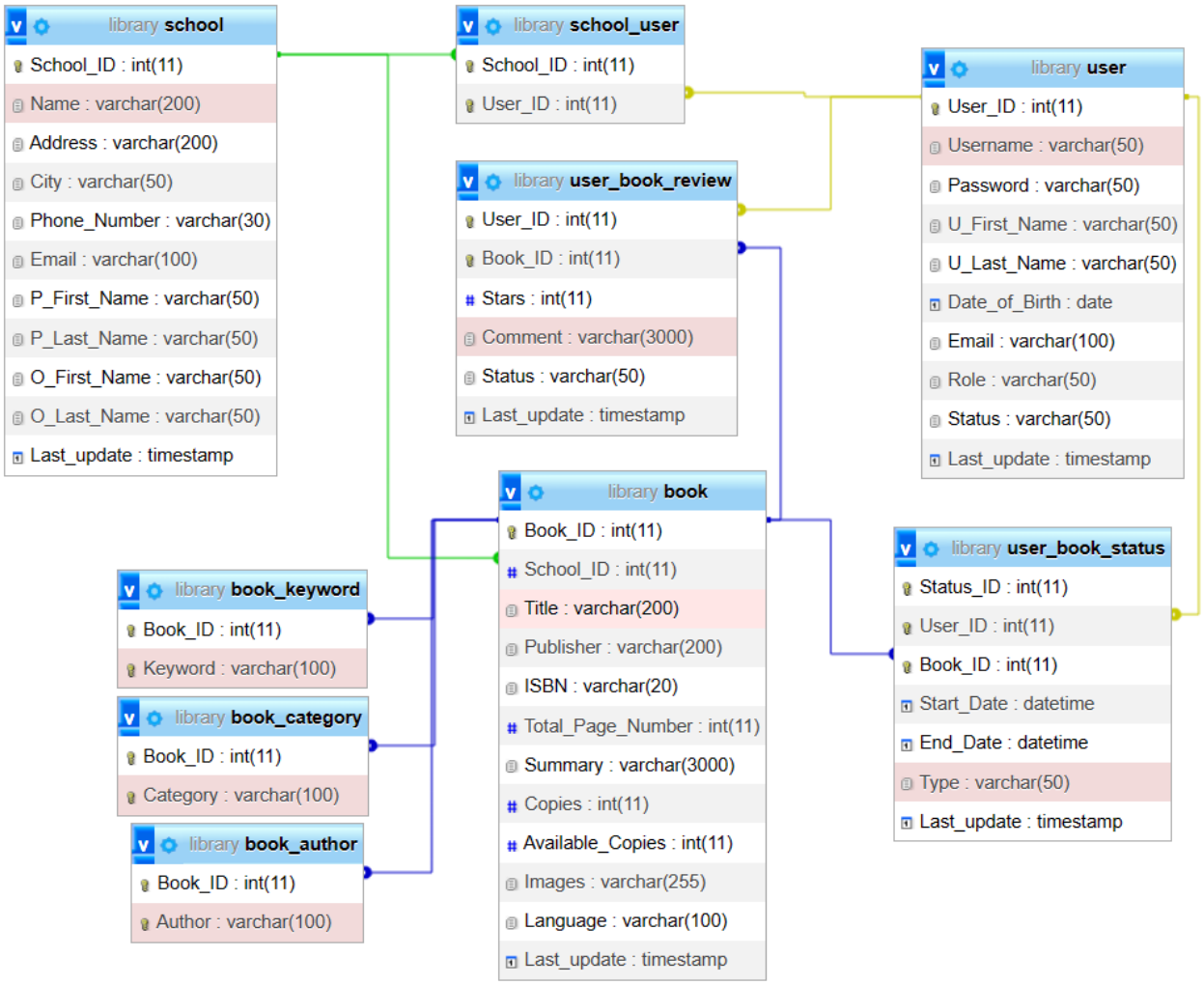
user – book: παρατηρούνται δύο σχέσεις μεταξύ αυτών των δύο entities

- user_book_review: η σχέση είναι N to M
    - Ένα βιβλίο μπορεί να δεχτεί πολλές αξιολογήσεις
    - Ένας χρήσεις κάνει πολλές κάνει πολλές αξιολογήσεις

    - Η σχέση αυτή εμφανίζει ορισμένα χαρακτηριστικά όπως:
        - Stars
        - Comment
        - Status, το οποίο χρησιμοποιείται για να μπορεί ο χειριστής του κάθε σχολείου να δέχεται τα comments των μαθητών προτού ανεβούν στην εφαρμογή.
        - Last_update

- user_book_status: η σχέση είναι N to M
    - Ένα βιβλίο μπορεί να έχει πολλούς δανεισμούς
    - Ένας χρήστης μπορεί να κάνει πολλούς δανεισμούς
    - Η σχέση αυτή εμφανίζει ορισμένα χαρακτηριστικά όπως:
        - Status_ID, χρησιμοποιείται για να κωδικοποιεί τις αιτήσεις των χρηστών για κάποιο αντίτυπο ενός βιβλίου και να μπορεί ο χρήστης να κάνει για το ίδιο βιβλίο και άλλη αίτηση
        - Start_Date, η ημερομηνία έναρξης της κράτησης – δανεισμού του χρήστη
        - End_Date, η ημερομηνία λήξης της κράτησης – δανεισμού του χρήστη
        - Type, χρησιμοποιείται για την διαφοροποίηση των διαφορετικών ειδών αίτησης που μπορεί να υπάρξουν, πιο αναλυτικά:
            - **'Rent'**, όταν ο χρήστης έχει στην κατοχή του ένα βιβλίο εντός προθεσμίας παράδοσης
            - **'Returned'**, όταν ο χρήστης έχει επιστρέψει ένα βιβλίο που είχε στην κατοχή του
            - **'Late'**, όταν ο χρήστης έχει στην κατοχή του ένα βιβλίο, έχει περάσει η διορία για την επιστροφή του και δεν το έχει επιστρέψει
            - **'Application'**, όταν ο χρήστης έχει κάνει μία αίτηση για να δανειστεί ένα βιβλίο το οποίο έχει διαθέσιμα αντίτυπα
            - **'Waiting'**, όταν ο χρήστης έχει κάνει μια αίτηση για να δανειστεί ένα βιβλίο και δεν υπάρχουν αντίτυπα επομένως μπαίνει σε ουρά αναμονής
            - **'Canceled'**, όταν ακυρωθεί μια αίτηση είτε από τον ίδιο τον χρήση είτε από τον χειριστή του σχολείου του
        - Last_update, για δική μας διευκόλυνση

Το Σχεσιακό Διάγραμμα της Βάσης Δεδομένων, φαίνεται παρακάτω:



Παρατηρείται ότι εκτός από τους πίνακες που υπήρχαν στο ER datagram υπάρχουν άλλοι έξι πίνακες, αυτό συμβαίνει για διάφορους λόγους:

- Οι πίνακες book_keyword, book_category και book_author έχουν δημιουργηθεί καθώς οι τιμές των keywords, των categories και των authors είναι multi values και μπορεί να έχουν πολλές τιμές επομένως για την σωστή απεικόνιση τους χρειάζονται έναν επιπλέον πίνακα
- Ο πίνακας school_user δημιουργήθηκε επειδή η σχέση μεταξύ των entities school και user είναι N το M και χρειάζεται ένας επιπλέον πίνακας για την σύνδεση των δύο entities που θα περιέχει τα primary keys τους, δηλαδή το School_ID και το User_ID
- Οι πίνακες user_book_review και user_book_status προέκυψαν από τις αντίστοιχες σχέσεις καθώς είχαν ορισμένα χαρακτηριστικά τα οποία πρέπει να αποτυπωθούν σε πίνακες

Σας δίνουμε τα SQL scripts για την δημιουργία των πινάκων που χρησιμοποιήσαμε στην βάση:

Για την δημιουργία του table **'user'**:

```
CREATE TABLE IF NOT EXISTS user (
   User_ID INT NOT NULL AUTO_INCREMENT,
   Username VARCHAR(50) NOT NULL,
   Password VARCHAR(50) NOT NULL,
   U_First_Name VARCHAR(50) NOT NULL,
   U_Last_Name VARCHAR(50) NOT NULL,
   Date_of_Birth DATE NOT NULL,
   Email VARCHAR(100) NOT NULL,
   Role VARCHAR(50) NOT NULL,
   Status VARCHAR(50) NOT NULL DEFAULT 'Pending',
   Last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
   PRIMARY KEY (User_ID)
);
```

Στο table 'user', το primary key είναι το User_ID, καθώς είναι μοναδικό και ξεχωριστό για κάθε χρήστη. Το User_ID γίνεται auto increment για δική μας ευκολία και για αποφυγή λαθών, επίσης το Status είναι by default 'Pending' ώστε να γίνεται στην πορεία 'Approved' αφού εγκριθεί από τον αρμόδιο.

Για την δημιουργία του table **'school'**:

```
CREATE TABLE IF NOT EXISTS school (
   School_ID INT NOT NULL AUTO_INCREMENT,
   Name VARCHAR(200) NOT NULL,
   Address VARCHAR(200) NOT NULL,
   City VARCHAR(50) NOT NULL,
   Phone_Number VARCHAR(30) NOT NULL,
   Email VARCHAR(100) NOT NULL,
   P_First_Name VARCHAR(50) NOT NULL,
   P_Last_Name VARCHAR(50) NOT NULL,
   O_First_Name VARCHAR(50) NOT NULL,
   O_Last_Name VARCHAR(50) NOT NULL,
   Last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
   PRIMARY KEY (School_ID)
);
```

Ομοίως στο table 'school', το primary key είναι το School_ID καθώς είναι μοναδικό και ξεχωριστό για κάθε σχολείο.

Για την δημιουργία του table **'school_user'**:

```
CREATE TABLE IF NOT EXISTS school_user (
   School_ID INT NOT NULL,
   User_ID INT NOT NULL,
   PRIMARY KEY (School_ID, User_ID),
   FOREIGN KEY (School_ID) REFERENCES school(School_ID) ON UPDATE CASCADE ON DELETE RESTRICT,
   FOREIGN KEY (User_ID) REFERENCES user(User_ID) ON UPDATE CASCADE ON DELETE RESTRICT
);
```

Στον table 'school_user' το primary key αποτελείται από δύο στήλες, το School_ID και το User_ID. Αυτό σημαίνει ότι ο συνδυασμός αυτών των δύο στηλών προσδιορίζει μοναδικά κάθε γραμμή του πίνακα .
Ο πίνακας έχει δύο foreign keys, η στήλη Schoool_ID αναφέρεται στην ομώνυμη στήλη του πίνακα 'school', και η στήλη User_ID αναφέρεται στην ομώνυμη στήλη του πίνακα 'user'.

Για την δημιουργία του table **‘book’**:

```
CREATE TABLE IF NOT EXISTS book (
   Book_ID INT AUTO_INCREMENT PRIMARY KEY,
   School_ID INT NOT NULL,
   Title VARCHAR(200) NOT NULL,
   Publisher VARCHAR(200) NOT NULL,
   ISBN VARCHAR(20) NOT NULL,
   Total_Page_Number INT NOT NULL,
   Summary VARCHAR(3000) NOT NULL,
   Copies INT NOT NULL CHECK (Copies >= 0),
   Available_Copies INT NOT NULL CHECK (Available_Copies >= 0),
   Images VARCHAR(255) NOT NULL,
   Language VARCHAR(100) NOT NULL,
   Last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
   FOREIGN KEY (School_ID) REFERENCES school(School_ID) ON UPDATE CASCADE ON DELETE RESTRICT
);
```

Στο table ‘book’, το primary key είναι το Book_ID καθώς είναι μοναδικό και ξεχωριστό για κάθε βιβλίο. Ο πίνακας έχει ένα foreign key, η στήλη School_ID αναφέρεται στην ομώνυμη στήλη του πίνακα ‘school’, με αυτόν τον τρόπο καθιερώνεται μία σχέση μεταξύ του table school και του table ‘book’. Στα attributes Copies και Available_Copies υπάρχει περιορισμός η τιμή τους να είναι μεγαλύτερη ίση του 0. Κάθε βιβλίο, ως αντικείμενο, έχει πολλά αντίτυπα τα οποία διαμοιράζονται στους χρήστες κατά τις αιτήσεις.

Για την δημιουργία του table **‘book_author’**:

```
CREATE TABLE IF NOT EXISTS book_author (
   Book_ID INT NOT NULL,
   Author VARCHAR(100) NOT NULL,
   PRIMARY KEY (Book_ID, Author),
   FOREIGN KEY (Book_ID) REFERENCES book(Book_ID) ON UPDATE CASCADE ON DELETE RESTRICT
);
```

Το table αυτό αντιπροσωπεύει μια σχέση many-to-many ανάμεσα στο table ‘book’ και στο entity ‘author’. Το primary key αποτελείται από τον συνδυασμό των στηλών Book_ID και Author, αυτό διασφαλίζει ότι κάθε βιβλίο μπορεί να έχει πολλούς συγγραφείς και κάθε συγγραφέας μπορεί να συσχετίζεται με πολλά βιβλία χωρίς διπλότυπες εγγραφές.
Ο πίνακας έχει ένα foreign key, η στήλη Book_ID αναφέρεται στην ομώνυμη στήλη του πίνακα book, με αυτόν τον τρόπο καθιερώνεται μία σχέση μεταξύ του table ‘book’ και του table ‘book_author’.

Για την δημιουργία του table **‘book_category’**:

```
CREATE TABLE IF NOT EXISTS book_category (
   Book_ID INT NOT NULL,
   Category VARCHAR(100) NOT NULL,
   PRIMARY KEY (Book_ID, Category),
   FOREIGN KEY (Book_ID) REFERENCES book(Book_ID) ON UPDATE CASCADE ON DELETE RESTRICT
);
```

Ομοίως με προηγουμένως αυτό το table αντιπροσωπεύει μια σχέση many-to-many ανάμεσα στο table ‘book’ και στο entity ‘category’. Το primary key αποτελείται από τον συνδυασμό των στηλών Book_ID και Category, αυτό διασφαλίζει ότι κάθε βιβλίο μπορεί να ανήκει σε πολλές κατηγορίες και κάθε κατηγορία μπορεί να συσχετίζεται με πολλά βιβλία χωρίς διπλότυπες εγγραφές.
Ο πίνακας έχει ένα foreign key, η στήλη Book_ID αναφέρεται στην ομώνυμη στήλη του πίνακα book, με αυτόν τον τρόπο καθιερώνεται μία σχέση μεταξύ του table ‘book’ και του table ‘book_category’.

Για την δημιουργία του table **‘book_keyword’**:

```
CREATE TABLE IF NOT EXISTS book_keyword (
   Book_ID INT NOT NULL,
   Keyword VARCHAR(100) NOT NULL,
   PRIMARY KEY (Book_ID, Keyword),
   FOREIGN KEY (Book_ID) REFERENCES book(Book_ID) ON UPDATE CASCADE ON DELETE RESTRICT
);
```

Ομοίως με προηγουμένως αυτό το table αντιπροσωπεύει μια σχέση many-to-many ανάμεσα στο table ‘book’ και στο entity ‘keyword’. Το primary key αποτελείται από τον συνδυασμό των στηλών Book_ID και Keyword, αυτό διασφαλίζει ότι κάθε βιβλίο μπορεί έχει πολλές λέξεις κλειδιά και κάθε λέξη κλειδί μπορεί να συσχετίζεται με πολλά βιβλία χωρίς διπλότυπες εγγραφές.
Ο πίνακας έχει ένα foreign key, η στήλη Book_ID αναφέρεται στην ομώνυμη στήλη του πίνακα book, με αυτόν τον τρόπο καθιερώνεται μία σχέση μεταξύ του table’ book’ και του table ‘book_keyword’.

Για την δημιουργία του table **‘user_book_status’**:

```
CREATE TABLE IF NOT EXISTS user_book_status (
   Status_ID INT AUTO_INCREMENT,
   User_ID INT NOT NULL,
   Book_ID INT NOT NULL,
   Start_Date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
   End_Date DATETIME NOT NULL DEFAULT (Start_Date + INTERVAL 7 DAY),
   Type VARCHAR(50) NOT NULL,
   Last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
   PRIMARY KEY (Status_ID, User_ID, Book_ID),
   FOREIGN KEY (User_ID) REFERENCES user(User_ID) ON UPDATE CASCADE ON DELETE RESTRICT,
   FOREIGN KEY (Book_ID) REFERENCES book(Book_ID) ON UPDATE CASCADE ON DELETE RESTRICT
);
```

Το primary key αποτελείται από τον συνδυασμό των στηλών Status_ID, User_ID και Book_ID, αυτό διασφαλίζει ότι κάθε χρήστης μπορεί να έχει πολλαπλές αιτήσεις για το ίδιο βιβλίο και ένα βιβλίο μπορεί να ανήκει σε πολλές αιτήσεις, χωρίς διπλότυπες εγγραφές.
Ο πίνακας έχει δύο foreign keys, η στήλη User_ID αναφέρεται στην ομώνυμη στήλη του πίνακα user, και η στήλη Book_ID αναφέρεται στην ομώνυμη στήλη του πίνακα book.
Το Start_Date ορίζεται αυτόματα ως η τρέχουσα μέρα και ώρα, την στιγμή που γίνεται η αίτηση.
Το End_Date ορίζεται αυτόματα μία βδομάδα αργότερα από την στιγμή που γίνεται η αίτηση διότι η αιτήσεις έχουν περιορισμένη διάρκεια. Η χρήση του Type έχει εξηγηθεί πιο πάνω.

Για την δημιουργία του table **‘user_book_review’**:

```
CREATE TABLE IF NOT EXISTS user_book_review (
   User_ID INT NOT NULL,
   Book_ID INT NOT NULL,
   Stars INT NOT NULL CHECK (Stars >= 1 AND Stars <= 5),
   Comment VARCHAR(3000) NOT NULL,
   Status VARCHAR(50) NOT NULL DEFAULT 'Pending',
   Last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
   PRIMARY KEY (User_ID, Book_ID),
   FOREIGN KEY (User_ID) REFERENCES user(User_ID) ON UPDATE CASCADE ON DELETE RESTRICT,
   FOREIGN KEY (Book_ID) REFERENCES book(Book_ID) ON UPDATE CASCADE ON DELETE RESTRICT
);
```

Το primary key αποτελείται από συνδυασμό των στηλών User_ID και Book_ID, με αυτό τον τρόπο ο κάθε χρήστης μπορεί να έχει το πολύ μια αξιολόγηση σε ένα βιβλίο, για αυτό του δίνεται η δυνατότητα να ανανεώσει την αξιολόγησή του εάν θελήσει.
Ο πίνακας έχει δύο foreign keys, η στήλη User_ID αναφέρεται στην ομώνυμη στήλη του πίνακα user, και η στήλη Book_ID αναφέρεται στην ομώνυμη στήλη του πίνακα book.
Το Stars επιτρέπεται να πάρει τιμές από 1 έως 5 για να είναι σωστή η κλίμακα.
Το Status είναι by default 'Pending' ώστε να γίνεται στην πορεία 'Approved' αφού εγκριθεί από τον αρμόδιο, για τους μαθητές, ενώ για τους υπόλοιπους χρήστες μέσα στην εφαρμογή γίνεται κατευθείαν 'Approved'.

Για όλα τα tables που έχουν foreign keys, για τα foreign keys τους ισχύει ότι:
ON UPDATE CASCADE ON DELETE RESTRICT
Το οποίο καθορίζει τις ενέργειες που πρέπει να γίνουν όταν πραγματοποιούνται είτε ενημερώσεις είτε διαγραφές. Στην περίπτωση του update εάν ενημερωθεί η τιμή του ξένου κλειδιού στο referenced table, πρέπει να ενημερωθεί αυτόματα και η τιμή του ξένου κλειδιού στο παρόν table, αυτό διασφαλίζει την συνέπεια μεταξύ των tables. Στην περίπτωση του delete όταν γίνει προσπάθεια για διαγραφή στο referenced table, η διαγραφή θα μπλοκαριστεί στην περίπτωση που υπάρχουν αντίστοιχες αναφορές σε κάποιον πίνακα που υπάρχει

Τα DDL και DML scripts είναι περιέχονται σε ένα ενιαίο αρχείο που περιέχει όλα τα δεδομένα της βάσης μας και βρίσκεται στο git hub στον φάκελο 'SQL' με το όνομα ΄library.sql΄ LINK

Όσον αφορά τα indexes:
Τα indexes είναι χρήσιμα σε μια Βάση Δεδομένων, τα βασικότερα είναι τα ακόλουθα:

- Βελτιωμένη απόδοση, αφού επιτρέπουν στην βάση να εντοπίζει και να ανακτά δεδομένα πιο γρήγορα.
- Ταχύτερη ανάκτηση δεδομένων, αφού η βάση μπορεί να εντοπίσει συγκεκριμένες σειρές ή περιοχές πιο αποτελεσματικά.
- Αποτελεσματική ταξινόμηση

Στην βάση δημιουργούνται by default indexes για τα primary keys των tables.
Αφού είδαμε τα indexes που είχαν δημιουργηθεί αυτόματα για κάθε table δημιουργήσαμε και τα ακόλουθα:

```
-- For the table 'user'
CREATE INDEX idx_username ON user (Username);
CREATE INDEX idx_role ON user (Role);
-- For the table 'school'
CREATE INDEX idx_name ON school (Name);
-- For the table 'book'
CREATE INDEX idx_title ON book (Title);
-- For the table 'book_keyword'
CREATE INDEX idx_keyword ON book_keyword (Keyword);
-- For the table 'book_author'
CREATE INDEX idx_author ON book_author (Author);
-- For the table 'book_category'
CREATE INDEX idx_school_id ON book_category (Category);
-- For the table 'user_book_status'
CREATE INDEX idx_type ON user_book_status (Type);
```

```
-- Table 'user_book_review'
CREATE INDEX idx_stars ON user_book_review (Stars);
CREATE INDEX idx_status ON user_book_review (Status);
CREATE INDEX idx_user_id ON user_book_review (User_ID);
```

INSTALLATION GUIDE:

Before we begin make sure you have installed the latest version of Python and GIT and you have a program like XAMPP installed to test the code in a local server.

1. Open the terminal and go to the directory you would like to store the app files.
2. Make sure you are running the local host server.
3. In the terminal clone the GitHub repository with this command:
    git clone https://github.com/StavrosXr/Database-Library
4. In the terminal do this command to change the directory to the one we just made:
    "cd Database-Library"
5. In the terminal run the following script to download all required libraries:
    "pip install -r requirements.txt"
    if you get an error when running this try running this instead:
    python -m pip install -r requirements.txt
6. To download the database, we first need to go to the directory that you have the bin of the installed XAMPP (for example: C:\xampp\mysql\bin)
7. Execute this command to make the empty library:
    "mysql -u root -p -e "CREATE DATABASE MyLibrary"
    When prompted to enter a password, hit ENTER.
8. Then in the terminal, execute this command to load the database:
    mysql -u root -p -h localhost MyLibrary < {backup_file_path}
    where {backup_file_path} is the path to the library.sql file.
    for example "..\Database-Library\library.sql
9. Go back to the directory of the project ("..\Database-Library\ )
10. Depending on your python version use the command python3 app.py or python app.py or py app.py and visit http://localhost:5000/ from a browser to run the web app.

USER MANUAL :

το User Manual Μπορει να βρεθει στο repository του GitHub στον φακελο Documents με το ονομα UserManual. Link