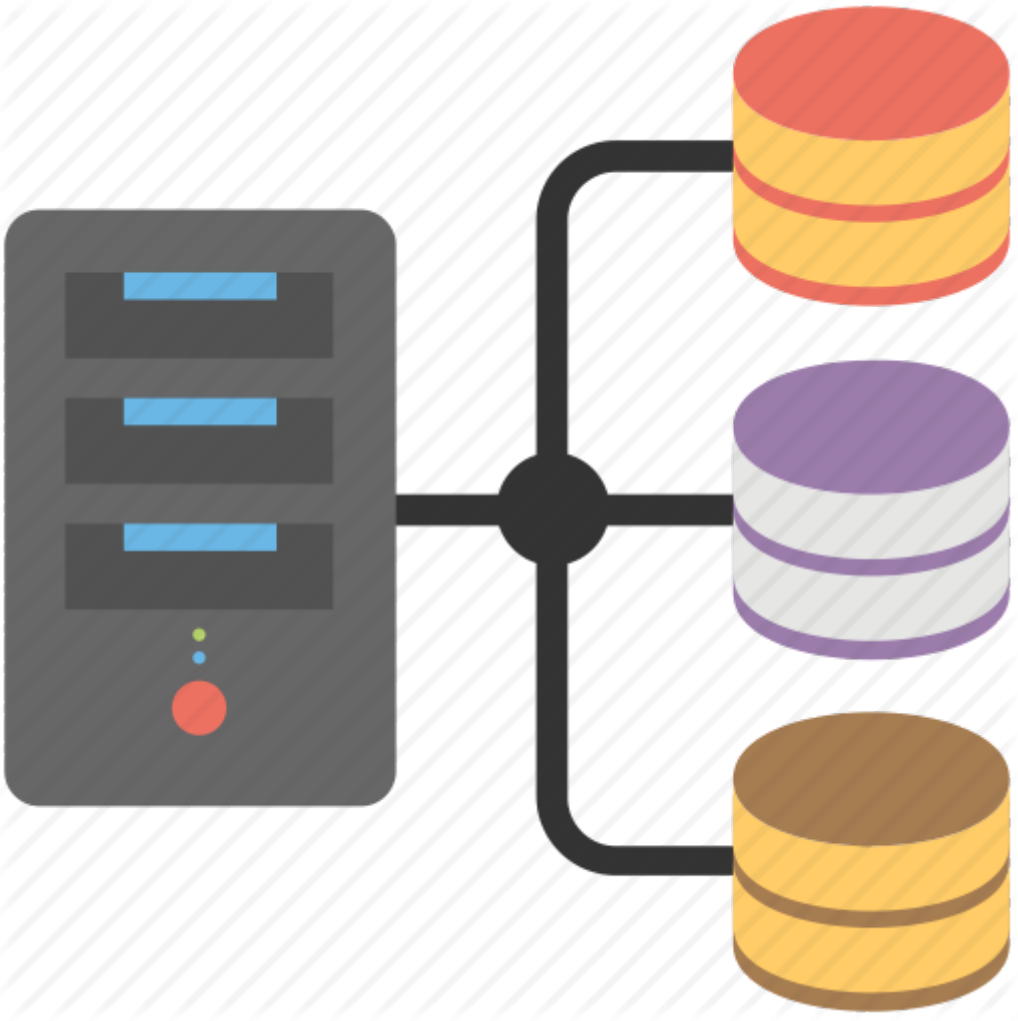
CmpE 321 - 2020/Summer

Project - 3:

Library System with SQL



Prepared by:

Hazar ÇAKIR - 2017400093

I) Introduction:

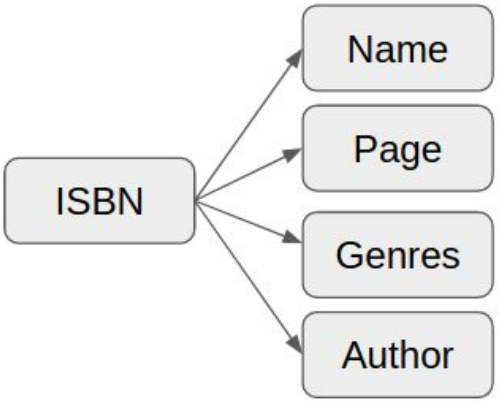
This is a database project which simulates the most basic library system without an authentication mechanism. In the implementation of the project, I use MySQL as the relational database management system and use MySQL servers in my local. I created a database called “project3” and all my project tables are in that one singular database. I use flask environment to handle backend and frontend. Flask is a python based web framework. I use the mysqldb module of flask in order to provide communication between database and backend. Mysqldb allows me to send queries as a string in Python. I use html web pages in cooperation with Bootstrap3 to attain more aesthetic design. Interface is as simple as possible and allows testing all features provided from the library system.

II) Relation definitions :

a) Book :

```
CREATE TABLE Book(
        ISBN CHAR(13) PRIMARY KEY,
        Name VARCHAR(50),
        Page INT,
        Genres VARCHAR(255),
        Author VARCHAR(50) );
```

BOOK				
ISBN (Char 13 Key Value)	**Name (Varchar 80)**	**Page (INT)**	**Genres (Varchar 255)**	**Author (Varchar 50)**
9786055272432	Book_name_1	n	List_of_Genres_1	Author_1
9780747532743	Book_name_2	300	List_of_Genres_2	Author_2
9780345418777	Book_name_3	80	List_of_Genres_3	Author_3
9780345391810	Book_name_4	450	List_of_Genres_4	Author_4
...	...	...	...	..



Key is ISBN.

- Every nonkey attribute is Full Functionally dependent on the key.
- There is no Functional Dependency among nonkey attributes.
    - So this relation is in **3NF.**
- Because there is just one Key, there is no need to explicitly control BCNF Form. Because every 3NF is naturally BCNF when necessary conditions are satisfied. In order to control BCNF, there should be at least 2 keys and those keys should be composite and there should be an intersection between keys.
    - So this relation is automatically in **BCNF**

b) filteredBooks:

```
CREATE TABLE filteredBooks(
    ISBN CHAR(13) PRIMARY KEY,
    Name VARCHAR(50),
    Page INT,
    Genres VARCHAR(255),
    Author VARCHAR(50) );
```

- filteredBooks relation is a copy of the Book relation. Same is valid here.

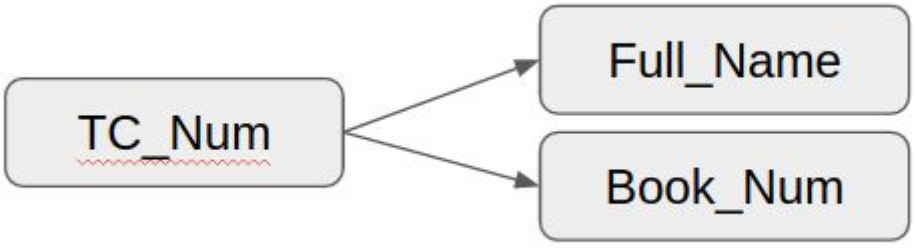
FILTEREDBOOKS				
ISBN (Char 13 Value) Key	**Name (Varchar 80)**	**Page (INT)**	**Genres (Varchar 255)**	**Author (Varchar 50)**
9786055272432	Book_name_1	n	List_of_Genres_1	Author_1
9780747532743	Book_name_2	300	List_of_Genres_2	Author_2

9780345418777	Book_name_3	80	List_of_Genres_3	Author_3
9780345391810	Book_name_4	450	List_of_Genres_4	Author_4
...	...	...	...	..

c) Borrower :

```
CREATE TABLE Borrower(
        TC_Num CHAR(11) PRIMARY KEY,
        Full_Name VARCHAR(50),
        Book_Num INT );
```

BORROWER		
TC_Num (Char 11 Key Value)	**Full_Name (Varchar 50)**	**Book_Num (INT)**
12345678901	Person_name_1	1
98765432109	Person_name_2	0
11223344556	Person_name_3	5
42424242424	Person_name_4	8
...	...	...



Key is TC_Num

- Every nonkey attribute is Full Functionally dependent on the key.
- There is no Functional Dependency among nonkey attributes.
    - So this relation is in **3NF.**

- Because there is just one Key, there is no need to explicitly control BCNF Form. Because every 3NF is naturally BCNF when necessary conditions are satisfied. In order to control BCNF, there should be at least 2 keys and those keys should be composite and there should be an intersection between keys.
    - So this relation is automatically in **BCNF**

d) BorrowBook :

```
CREATE TABLE BorrowBook(
        TC_Num CHAR(11),
        ISBN CHAR(13),
        Duedate DATE,
        PRIMARY KEY (TC_Num, ISBN) );
```

BORROWBOOK		
TC_NUM (Char 11 - Key Value)	**ISBN (Char 13 - Key Value)**	**Duedate (Date)**
12345678901	9786055272432	11.06.2020
98765432109	9780747532743	15.06.2020
11223344556	9780345418777	19.06.2020
42424242424	9780345391810	24.06.2020
...	...	...

Key is TC_Num and ISBN together

- Every nonkey attribute is Full Functionally dependent on the key.
- There is no Functional Dependency among nonkey attributes.

    - So this relation is in **3NF.**
- Because there is just one Key, there is no need to explicitly control BCNF Form. Because every 3NF is naturally BCNF when necessary conditions are satisfied. In order to control BCNF, there should be at least 2 keys and those keys should be composite and there should be an intersection between keys.
    - So this relation is automatically in **BCNF**

III) SQL DML :

a) Create Book:

```
"INSERT INTO Book VALUES('%s', '%s', %s, '%s', '%s');" % (isbn,
    name,page,genres,author)
```

b)Delete Book:

```
SELECT deleteBook('%s'); % (isbn)

CREATE FUNCTION deleteBook(target VARCHAR(13)) RETURNS BOOLEAN
  BEGIN
      IF NOT EXISTS(
        SELECT * FROM Book WHERE ISBN = target)
      THEN RETURN FALSE;
      ELSE
        DELETE FROM Book WHERE ISBN = target;
       UPDATE Borrower SET Book_Num = Book_Num-1 WHERE TC_Num=(SELECT TC_Num FROM
BorrowBook WHERE ISBN = target);
        DELETE FROM BorrowBook WHERE ISBN = target;
        RETURN TRUE;
      END IF;
  END//
```

c) Search Book:

```
"CALL searchBook('%s','%s',%s,%s,'%s');" % (isbn,name,page_Min,page_Max ,author)

for genre in genreList:
```

```
    cur.execute("SELECT filterGenre('%s');" % ('%'+ genre + '%'))
cur.execute("SELECT * FROM filteredBooks;")


CREATE FUNCTION filterGenre(Genrei VARCHAR(255)) RETURNS BOOLEAN
  BEGIN
      IF Genrei <> " "
      THEN DELETE FROM filteredBooks where Genres NOT LIKE Genrei; RETURN TRUE;
      ELSE
          RETURN FALSE;
      END IF;
  END//

CREATE PROCEDURE searchBook (IN ISBNi VARCHAR(255),IN Namei VARCHAR(50),IN
PageMini INT,IN PageMaxi INT,IN Authori VARCHAR(50))
  BEGIN
      DELETE FROM filteredBooks;
      INSERT INTO filteredBooks SELECT * FROM Book;

      IF ISBNi <> " "
      THEN DELETE FROM filteredBooks WHERE ISBN <> ISBNi;
      ELSEIF Namei <> " "
      THEN DELETE FROM filteredBooks WHERE Name NOT LIKE Namei ;
      ELSEIF PageMini <> -1
      THEN DELETE FROM filteredBooks WHERE Page < PageMini;
      ELSEIF PageMaxi <> -1
      THEN DELETE FROM filteredBooks WHERE Page > PageMaxi;
      ELSEIF Authori <> " "
      THEN DELETE FROM filteredBooks WHERE Author NOT LIKE Authori ;

      END IF;
  END//
```

d) Borrow Book:

```
"Select borrowBook(%s, %s);" % (tc_Num,isbn)

CREATE FUNCTION borrowBook(tc_numi VARCHAR(255), ISBNi VARCHAR(255)) RETURNS INT
  BEGIN
```

```
        IF NOT EXISTS(
            SELECT * FROM Borrower WHERE TC_Num = tc_numi)
        THEN RETURN 2;

        ELSEIF NOT EXISTS(
             SELECT * FROM Book WHERE ISBN = ISBNi)
        THEN RETURN 3;

        ELSEIF EXISTS(
             SELECT * FROM BorrowBook WHERE ISBN = ISBNi )
        THEN RETURN 4;
        ELSEIF 8 = (SELECT Book_Num FROM Borrower WHERE TC_Num = tc_numi)
        THEN RETURN 5;
        ELSE
            INSERT INTO BorrowBook(TC_Num,ISBN) VALUES(tc_numi,ISBNi);
            UPDATE Borrower SET Book_Num = Book_Num+1 WHERE TC_Num = tc_numi;
            RETURN 1;
        END IF;
    END//
```

e) Return Book:

```
"Select returnBook(%s);" % (isbn)

 CREATE FUNCTION returnBook(ISBNi VARCHAR(255)) RETURNS INT
    BEGIN
        IF NOT EXISTS(
             SELECT * FROM Book WHERE ISBN = ISBNi)
        THEN RETURN 2;

        ELSEIF NOT EXISTS(
             SELECT * FROM BorrowBook WHERE ISBN = ISBNi )
        THEN RETURN 3;
        ELSE
             UPDATE Borrower SET Book_Num = Book_Num-1 WHERE TC_Num=(SELECT TC_Num
FROM BorrowBook WHERE ISBN = ISBNi);
            DELETE FROM BorrowBook WHERE ISBN = ISBNi;
            RETURN 1;
        END IF;
```

```
  END//
```

f) Control by TC Number:

```
SELECT Book_Num,Full_Name FROM Borrower WHERE TC_Num = %s" % tc_Num
  "SELECT  BorrowBook.ISBN,Book.Name,BorrowBook.Duedate  FROM  BorrowBook,Book,
Borrower  WHERE  BorrowBook.TC_Num  =  '%s'  AND  Borrower.TC_Num  =  '%s'  AND
BorrowBook.ISBN = Book.ISBN;" % (tc_Num,tc_Num)
```

g) Register User:

```
"INSERT INTO Borrower VALUES(%s, \'%s\', 0);" % (tc_Num,name)
```

h) List Users:

```
"SELECT * FROM Borrower;"
```

i) List All Books:

```
"SELECT * FROM Book;"
```

j) List Available Books:

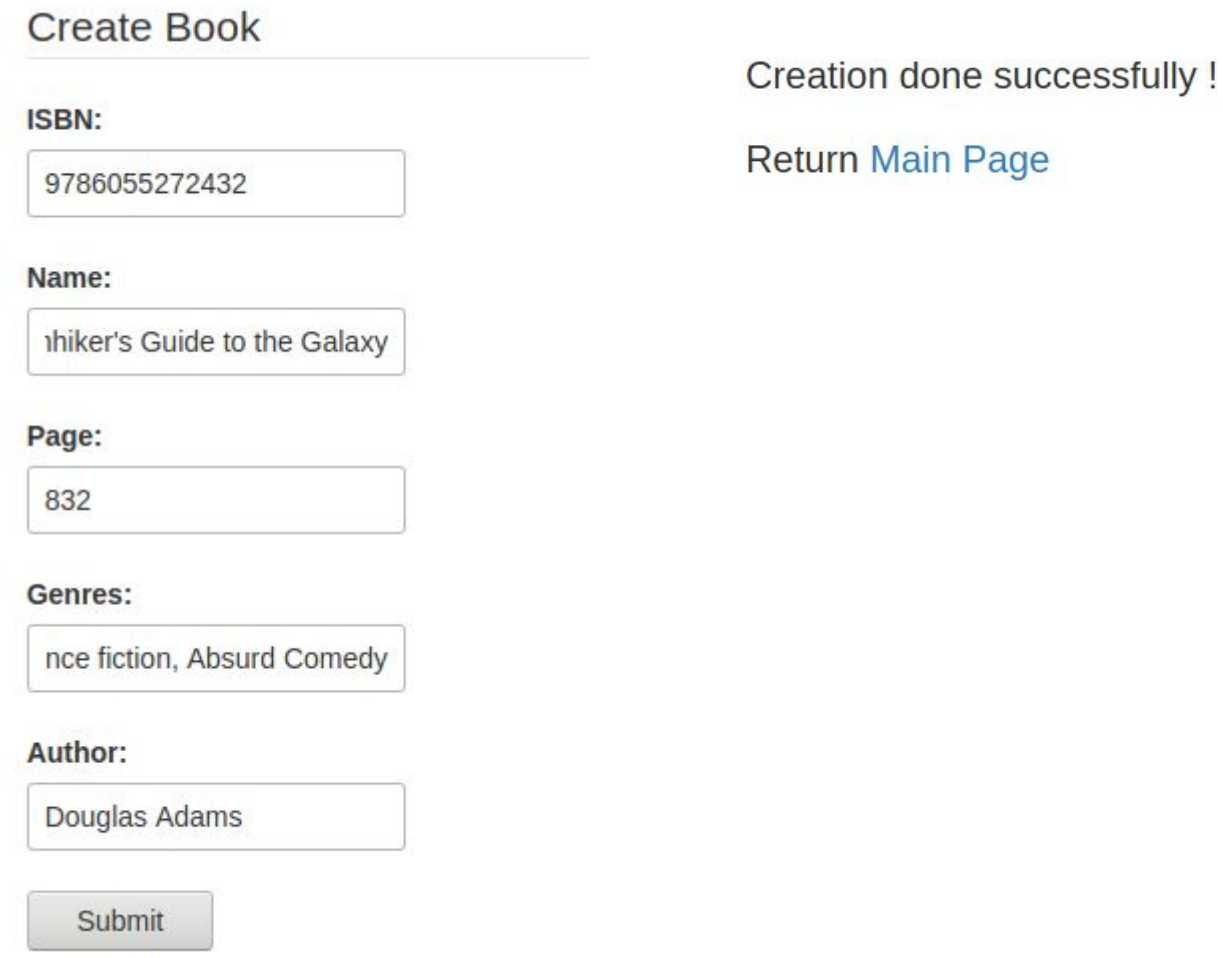
```
"SELECT availableBooks()"
"SELECT * FROM filteredBooks;"

CREATE FUNCTION availableBooks() RETURNS BOOLEAN
  BEGIN
      DELETE FROM filteredBooks;
      INSERT INTO filteredBooks SELECT * FROM Book;
      DELETE FROM filteredBooks WHERE ISBN IN(SELECT ISBN FROM BorrowBook) ;
      RETURN TRUE;
  END//
```

IV) Constraints:

- Book Table has one Primary Key as ISBN
- filteredBooks Table has one Primary Key as ISBN
- Borrower Table has one Primary Key as TC_Num
- BorrowBook Table has two Primary Keys as TC_Num and ISBN
- TC_Num has constant length 11 character
- ISBN has constant length 13 character
- There is one trigger in BorrowBook Table and calculates Duedate rooting from current date via adding 14 days.

V) Screenshots of input and output:

- **Create Book:**



- **Delete Book:**

Delete Book

ISBN:

9786055272432

Delete

Deletion done successfully !

Return Main Page

k) Search Book:

Search Book

ISBN:

Name:

Page Number Min Value:

Page Number Max Value:

Genres:

Fiction

Author:

Search

Your conditions was like:

Genres: Fiction

List of the books that you're looking for:

Book ISBN	Book Name	Page Number	Genres	Author
9780747532743	Harry Potter and the Philosophers Stone	350	Novel, Fiction, Fantastic	J.K.Rowling
9786055272432	The Hitchhikers Guide to the Galaxy	832	Science fiction, Absurd Comedy	Douglas Adams

Return Main Page

l) Borrow Book:

Borrow Book

Tc Number of Borrower:

44444444444

ISBN of Book:

9780060736781

Borrow

Borrow operation done successfully !

Return Main Page

m) Return Book:

n) Control by TC Number:

Control My Info

Tc Number of Borrower:

44444444444

Control

Borrower Tricia McMillan has 2 books:

Book ISBN	Book Name	Due Date
9780060736781	Improbable	2020-09-28
9786055272432	The Hitchhikers Guide to the Galaxy	2020-09-28

Return Main Page

o) Register User:

Registeration done successfully !

Return Main Page

p) List Users:

List of all Users

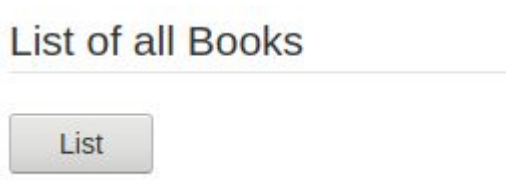
List

List of the Users :

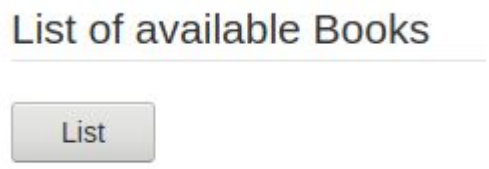
TC_Num	Full Name	Book Number
22211133444	Ford Prefekt	0
44444444444	Tricia McMillan	1
88877766555	Arthur Dent	0

Return Main Page

q)List All Books:



r) List Available Books:



List of the Available Books :

Book ISBN	Book Name	Page Number	Genres	Author
9780747532743	Harry Potter and the Philosophers Stone	350	Novel, Fiction, Fantastic	J.K.Rowling
9786055272432	The Hitchhikers Guide to the Galaxy	832	Science fiction, Absurd Comedy	Douglas Adams

Return Main Page

VI) Readme :

- **Those following codes should run from terminal in the project3 location**

```
--> cd projectFiles/
--> source venv/bin/activate
(venv)--> export FLASK_APP=project3.py
(venv)--> flask run

 * Serving Flask app "project3.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- Backend is working on http://127.0.0.1:5000/ local. You can run in this location. I used a virtual environment to run my backend so no extra installation is necessary.
- Use mysql, I assume mysql-server is installed, if not, please install
- Run Create.sql to configure database in mysql server.

VII) Conclusions & Assessment:

I try to implement this project as realistic and as practical as possible. I applied most of the library operations and tested all of them. Faced with no problem in those tests. It is easy to use and understand.

Hazar Çakır