PROJECT REPORT

LIBRARY MANAGEMENT SYSTEM

Purpose -

To create a database management system (DBMS) that enhances the efficiency with which one can use a library.

Abstract –

For this project, library is a collection of books along with their related information. It is generally accessible to a limited number of people that includes the students and the faculty of a particular institution. It is always a tedious work to keep track of the available books and more often than not the customer wastes his/her time in searching for a book. I believe that this database management system will automate the operations with respect to a book in the library. This will also help the user (library manager) to access the various book details.

Introduction –

The system will contain a database that has all the information about the books. It will also allow the customer to rent a book (virtually) and store his/her details in the database. The customer is either a student or a faculty. I have created two tables for Students and Faculty to store their personal data which will help in signing in/up to access certain facilities provided by my Library Management System (LMS). The admin/owner can add new books, along with removing unwanted books.

I have formulated SQL queries to help customers (student or faculty) rent books that are available in my database. Along with the basic operation of renting a book, my system can answer other questions including number of books available, number of distinct books available, number of books each customer has rented and the cost he/she has to pay before the semester ends.

Tables/Views –

1. Students – Stores the Students’ information
2. Faculty – Stores the Faculty’s information
3. Admin – Stores the owner’s information
4. Track\_stu\_rent – tracks the number of books rented by a particular student
5. Track\_fac\_rent – tracks the number of books rented by a particular faculty
6. V1 – has joined information of books and authors.
7. Books – Has book related information.
8. Authors – Has author related information, for each book present in the Books table.

Delivered –

* I have 51 distinct books available in my database, but in reality, it stores 625 books (book\_available\_count column has the number of books present of each type).
* User can sign in/up.
* Unsigned in user (student or faculty) can display the books available (using view), get a count of number of books and get a count of distinct books.
* Signed in user (student or faculty) can rent book, check dues and checked the number of books rented.

Related Work – (KOHA)

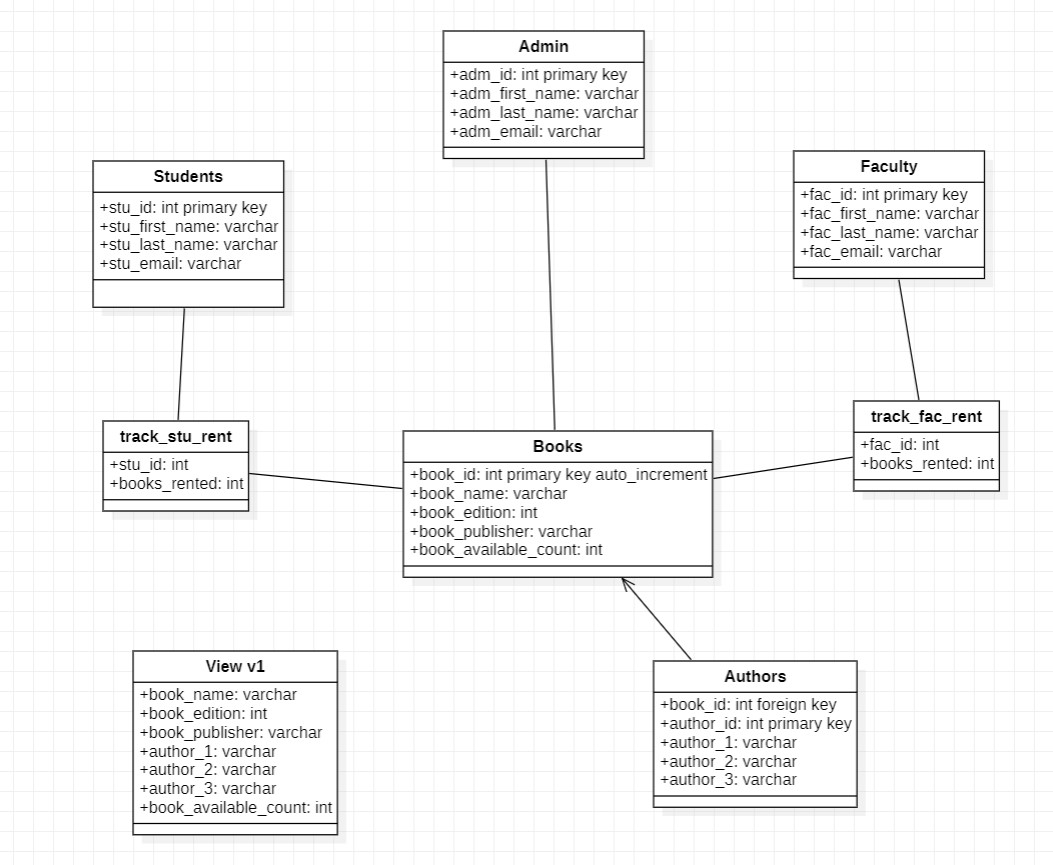
Koha is an Integrated Library System, designed to work on Linux but will work on Windows with the installation of additional modules.

During the semester, I have had a chance to explore the working of KOHA. The most important feature of KOHA is the Online Public Access Catalogue (OPAC), which helps the end users to perform library related operations, most important being the searching of an item (in this case a book). Normally, an end user has to enter a query, in this case it can be a book name, author name etc., OPAC retrieves all the data files (in this case books) that can be closely related with the query.

Before OPAC, Card Catalogue was extensively used for many years. Here, a certain number of cards were maintained that had information regarding the books, and these cards were ordered in alphabetical order. This method was extremely time taking, so now it is completely replaced by OPAC. Some libraries may still have card catalogue but they are only there as a back-up option, OPAC being the main source for searching the books.

My project is inspired by the techniques of OPAC, so the final result should offer efficiency, something close to what a KOHA application offers.

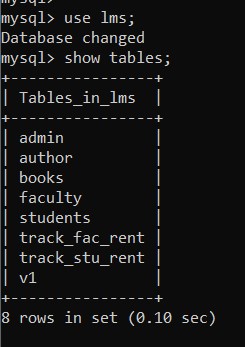
Design –



The ‘Books’ table is where all the data about the books, including name, edition, publisher and number of books available, is stored. The primary key of this table (book\_id) is given the constraint of ‘AUTO\_INCREMENT’, so whenever a new book is inserted in the table, a new unique id need not be mentioned. Initially, I had stored the information regarding authors in this table but that created a lot of redundant data. So, I created another table ‘Authors’ referring to the ‘Books’ table (Foreign Key is present in the ‘Authors’ table that references the parent ‘Books’ table) which stores the author names of every book that is present in the ‘Books’ table. This helped me reduce the redundancy in the ‘Books’ table, thus normalizing the table. The normalization I implemented here is the ‘First Normal Form (1NF)’, which helps in storing similar kind of data (multiple author names in my case) by creating more tables. While creating the foreign key in ‘Authors’ table, I used the ‘on update cascade’ and ‘on delete cascade’ constraints so that when a tuple in the ‘Books’ table gets updated/deleted, same operation should happen for the corresponding tuples in ‘Authors’ table. The ‘Author’ table can store a maximum of three authors, if there is only one or two authors for a certain book then NULL is given to the second and third author respectively.

There are three kinds of users that can access my system – Students, Faculty and Admin/Owner. Three different tables are present that store the personal details of these users. Students and Faculty can sign up by giving their personal details and can login by using their id. After signing in, both the users can rent books by either book name or book author(s). They can also check their dues and the number of books they have rented. I have two different tables ‘track\_stu\_rent’ and ‘track\_fac\_rent’ that keeps track of the number of books each user has rented. Two different tables were necessary because a single table cannot store number of books rented for both users, even if I make that happen there would be way too many NULL values, which is not a good way of designing the database. The admin user can directly delete or insert a book in the ‘Books’ table. The admin may insert a new book when a new book is available for the library to rent, and he/she may delete a book when the number of books available of that type is zero or simply if the library does not wish to rent it anymore. The admin user can only do insert or delete operation after logging in using the required details.

I have also created a view that joins (Natural Join) ‘Books’ and ‘Authors’ table, and displays the attributes including book name, book edition, book publisher, number of books available, author 1, author 2 and author 3.



Evaluation/Analysis –

The main analysis that I can make here is that of reducing the redundancy in ‘Books’ table and avoiding NULL values wherever possible. Redundancy in a table occurs when the table is not normalized, and this may cause insertion/updation/deletion anomalies. NULL values are also not encouraged in a table as the result of any value with a NULL value is ‘unknown’, so it is not included in the output of a SQL SELECT query. The definition of the view is stored and not the output of the query that defines the view. Therefore, whenever the view is called the most recent version of table is queried to create a view.

Conclusion –

I conclude that my project is able to help users login and rent books, along with other facilities mentioned before. The output can be seen by simply running my Python code (on Command Prompt). I have used the SQL CREATE and DROP command to create tables and a view, as part of the Data Definition Language (DDL). Coming to the Data Manipulation Language, I have extensively used SQL INSERT command to fill in the ‘Books’ table, along with DELETE & UPDATE commands to achieve table manipulation. As I have implemented this project using Python, basic knowledge regarding the connectivity of MySQL and Python was vital before I started embedding SQL queries in Python code.

Future Works –

* The ‘Books’ table can further be freed of redundancy by storing the publisher’s name and edition number in different tables.
* A time stamp may be created at the time the user rents a book, so that the library can keep track of the number of days the particular books was rented.
* A web page may be created, using either Django or HTML (along with PHP) to enhance the user interface.

References –

* Database System Concepts (Textbook).
* <https://en.wikipedia.org/wiki/Library_catalog>
* <https://www.eifl.net/resources/koha-worlds-first-free-and-open-source-integrated-library-management-system>

Appendix –

