Library System & Tracking

Santosh Singh Rawat
UBID: srawat3
Engineering Science Data Science
School of Eng. and Applied Sciences
Buffalo, USA
srawat3@buffalo.edu

Susmitha Chowdary Bandaru
UBID: bandaru5
Engineering Science Data Science
School of Eng. and Applied Sciences
Buffalo, USA
bandaru5@buffalo.edu

Preethi Sree Allam
UBID: pallam
Engineering Science Data Science
School of Eng. and Applied Sciences
Buffalo, USA
pallam@buffalo.edu

***Abstract*— This project presents a SQL-based Library Management System designed to streamline essential library functions. It provides efficient record management for students, books, and overdue. This project enhances the library experience by optimizing daily tasks and supporting a seamless borrowing process, reinforcing the library's crucial role in knowledge dissemination and access.**

I. Problem Statement

Inefficient Library Management and Record Keeping with Inaccurate Inventory Tracking

Library systems often fail to provide accurate real-time inventory data, leading to difficulties in tracking the availability of books and managing overdue items. Having these task performed in the Excel can lead to redundancy, inconsistency , difficulty in accessing data, Data isolation, Integrity problems for example it's hard to add new constraints or change existing ones. To resolve and address these issues we have built a database solution which has no redundancies, is easy to maintain, and consistent.

The project aims to address these challenges by implementing a modern and efficient library management system. The system will be designed to provide accurate and real-time information about library resources, streamline borrowing and return processes, improve user engagement, and enhance security and data integrity.

A. Background of the problem

Libraries serve as invaluable repositories of knowledge, providing access to a vast array of books and resources. In the digital age, managing library resources efficiently has become increasingly complex. The traditional paper-based system is no longer adequate to meet the evolving needs of libraries and their patrons. This necessitates the implementation of a modern Library Management System

B. Project's contribution

The introduction of a Library System & Tracking represents a transformative shift aimed at overcoming the limitations inherent in traditional library systems, propelling libraries into an era characterized by heightened efficiency and accessibility. The primary objectives of the project encompass various crucial aspects. Firstly, the implementation of Automated Catalog Management replaces manual cataloging, ensuring the accurate and efficient tracking of all library resources. A pivotal feature is the integration of a User-Friendly Interface, designed to empower both librarians and users with seamless navigation capabilities, thereby enhancing the overall user experience. Additionally, the project adopts a Centralized Database approach, consolidating all library information into a unified repository. This centralized structure facilitates easier management, updates, and retrieval of information. The Streamlined Borrowing Process is another key element, leveraging automation to simplify borrowing procedures, encompassing online requests, automated check-ins and check-outs, and real-time tracking of borrowed items. Notably, the system promotes Accessibility by allowing users to access the library catalog, review their borrowing history, and reserve books from the comfort of their homes. This commitment to digital accessibility not only embraces inclusivity but also contributes to the overall convenience of library services.

II. Target User

The database is designed to be used by the librarian and its associates and even the students(to some extent) in different library branches. They can add new books, students can themself loan a book by entering the book id and their student id, and return them as well, the same way. This job and triggers will automatically take care of managing the due amount for the students. Library Management Team: They oversee the library's strategic goals, policies, and resource allocation.

Library System Administrator: They manage the installation, configuration, and maintenance of the library software, ensuring that it operates smoothly.

Cataloging and Metadata Team: This team is responsible for cataloging new materials, updating records, and ensuring the quality and accuracy of metadata within the database

Acquisitions and Collection Development: Staff members involved in acquisitions and collection development use the database to track and manage the library's collection. They make decisions about purchasing, deaccessioning, and organizing materials.

Circulation Desk Staff: Circulation desk staff use the database to check materials in and out, manage patron accounts, and handle fines and overdue.

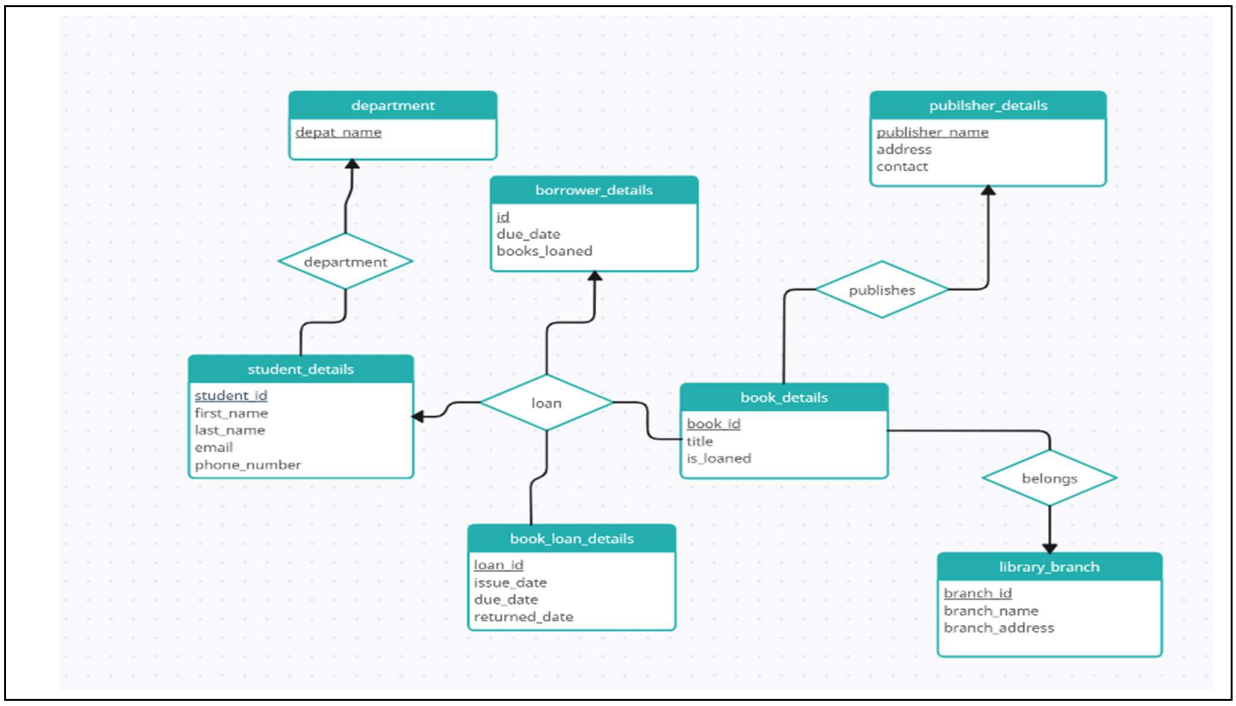
IT or Technical Support Team: The library's IT or technical support team assists with technical issues related to the database and the library management system, such as troubleshooting software problems, ensuring server and network connectivity, and maintaining the security of the database.

Data Analysts: In larger libraries, data analysts may be responsible for extracting and analyzing data from the library management database to make informed decisions about resource allocation, usage patterns, and service improvements.

III. E/R Diagram

The Entity-Relationship Diagram (ERD) outlines the main entities and their relationships within the database.

fig.1. E/R Diagram of "Library System & Tracking" database



IV. Components

Tables:

- **student_details**(student_id, first_name, last_name, email, phone_number, dept_name)

This table maintains student records.

1. student_id: Unique student id assigned by the university. It's the Primary Key for this table. Type: Varchar(10) NOT NULL
2. first_name: It holds the first name of the student. Type: varchar(50)
3. last_name: It holds the last name of the student. Type: varchar(50)
4. email: It holds the university email id of the student. Type: varchar(30)
5. phone_number: This attribute will have the contact of the student. Type:varchar(20)
6. dept_name: It tells, which department the student is in. type:varchar(20)

- **publisher_details**(publisher_name, publisher_address, contact_number)

This table has the details of all the publishers whose books are available in the library.

1. publisher_name: name of the publisher, this is the primary key, as a book will only have one publisher. Type: varchar(100)
2. publisher_address: address details of the publisher. Type: varchar(200)
3. contact_number: contact information for the publisher. Type: varchar(50)

- **library_branch_details**(branch_id, branch_name, branch_address)

This table has the library details of each branch.

1. branch_id :INT PRIMARY KEY NOT NULL IDENTITY, auto generated id, this attribute is unique as to differentiate between each branch.
2. branch_name: VARCHAR(100) NOT NULL, it holds the branch/library name.
3. branch_address: VARCHAR(200) NOT NULL, address of the library branch.

- **book_details**(book_id, title, publisher_name, branch_id, is_loaned)

This table stores the details for all the available books in the library.

1. book_id: INT IDENTITY PRIMARY KEY NOT NULL, it's an auto generated id which is unique to identify each book uniquely.
2. title: VARCHAR(200) NOT NULL, book title
3. publisher_name: VARCHAR(100) NOT NULL, this is the publisher of the book, it's a foreign key referencing the publisher details table. UPDATE/DELETE RISCTRICTED

4. branch_id: INT NOT NULL, it's the branch id, to which this book belongs to. It is a foreign key referencing to branch details table. UPDATE/DELETE RISCTRICTED
5. is_loaned: BOOLEAN DEFAULT FALSE, this field tells if the book is already loaned or available for loan. UPDATE/DELETE RISCTRICTED

- **book_loaned_details**(loan_id, book_id, branch_id, student_id, issue_date, due_date, returned_date)
  This table contains the details of those books which are loaned by the student.

1. loan_id: INT PRIMARY KEY NOT NULL IDENTITY, its auto generated id which is unique for each loan transaction.
2. book_id: INT NOT NULL , it's the unique book id referenced from book details table. UPDATE/DELETE RISCTRICTED
3. branch_id: INT NOT NULL , it's the branch id referenced from library branch details table. UPDATE/DELETE RISCTRICTED
4. student_id: VARCHAR(10) NOT NULL, it's the student id referenced from the student details table. UPDATE/DELETE RISCTRICTED
5. issue_date: DATE NOT NULL, it's the date when the book was loaned by the student.
6. due_date: DATE NOT NULL, it's the date when the book needs to be returned to avoid any late fee.
7. returned_date: DATE, this field will be null initially, and will be populated whenever the book is returned.

- **borrower_details**(id, student_id,due_amount, books_loaned)
  This table is to store the student id who has loaned at least one book in the past.

1. id: INT PRIMARY KEY NOT NULL IDENTITY, its auto generated id which is unique for each borrower.
2. student_id: VARCHAR(10) NOT NULL, stores the student id of the student. UPDATE/DELETE RISCTRICTED
3. due_amount: INT NOT NULL DEFAULT 0, holds the due amount if student fails to return the book before the due date.
4. books_loaned: INT NOT NULL DEFAULT 0, this tells the total number of books loaned by the student.

- **archive_book_loaned_details**(loan_id, book_id, branch_id, student_id, issue_date, due_date, returned_date)
  This table is used for archival data.

1. loan_id: INT PRIMARY KEY NOT NULL
2. book_id: INT NOT NULL , it's the unique book id referenced from book details table. UPDATE/DELETE RISCTRICTED
3. branch_id: INT NOT NULL , it's the branch id referenced from library branch details table. UPDATE/DELETE RISCTRICTED
4. student_id: VARCHAR(10) NOT NULL, it's the student id referenced from the student details table. UPDATE/DELETE RISCTRICTED
5. issue_date: DATE NOT NULL, it's the date when the book was loaned by the student.
6. due_date: DATE NOT NULL, it's the date when the book needs to be returned to avoid any late fee.
7. returned_date: DATE, this field will be null initially, and will be populated whenever the book is returned.

Functions:

- **func_addupdate_borrower_details()**: borrower(student) details will be added/updated to table **borrower_details**

- **func_update_number_of_booked_loaned()**:The book will be removed from the borrower's account and the book will be made available for loan again and the due amount for that book will be lifted.

Triggers:

- **tr_addupdate_borrower_details**: This trigger will be invoked after insert on **book_loaned_details,** and will call function: func_addupdate_borrower_details()
- **tr_update_number_of_booked_loaned**: This trigger will be invoked when updating the returned_date on **book_loaned_details,** and will call function: func_update_number_of_booked_loaned()

Scheduled Jobs:

- **refreshBorrowerDetails**: This job runs daily to update the dues for all the borrowers, it calls sp_updatedues()

- **archive data**: This job runs once a week to archive the data of book_loaned_details by calling sp_archive_book_loaned_details().

Procedures:

- **sp_updatedues()**: The procedure will update the due amount of the borrower whoever hasn't returned the book by the due date mentioned.

- **sp_add_book_loaned_details(IN book_title character varying, IN branch character varying, IN student_id character varying)**: This procedure will be used to loan books to students.

V. BCNF

1. **book_details**(book_id, title, publisher_name, branch_id, is_loaned)
   book_id →publisher_name, branch_id, title, is_loaned
   So, it's in BCNF as book_id is the super key and it can identify all other attributes

2. **student_details**(student_id, first_name, last_name, email, phone_number, dept_name)

   student_id→first_name, last_name, email, phone_number, dept_name
   So, this relation is also in BCNF

3. **publisher_details**(publisher_name, publisher_address, contact_number)
   publisher_details→publisher_address, contact_number
   So, this relation is also in BCNF

4. **library_branch_details**(branch_id, branch_name, branch_address)
   branch_id →branch_name, branch_address
   So, this relation is also in BCNF

5. **book_loaned_details**(loan_id, book_id, branch_id, student_id, issue_date, due_date, returned_date)
   loan_id→book_id, branch_id, student_id, issue_date, due_date, returned_date
   So, this relation is also in BCNF

6. **borrower_details**(id, student_id,due_amount, books_loaned)
   id→student_id,due_amount, books_loaned
   So, this relation is also in BCNF

VI. TASK 5

We incorporated a variety of data sources for information loading, employing Python's Faker library to generate synthetic data for both student and publisher details. The generated data was then stored in CSV format. For book details, information was sourced from the online site openlibrary.org/developers/dumps, and the data was similarly formatted into CSV files. Subsequently, the CSV data was imported into PostgreSQL using the platform's CSV import feature to complete the data loading process.

VII. SQL QUERIES

1. Query to insert student details:

```
INSERT INTO student_details
values('63529189','Jennifer','Glover','cheryl53@buffalo.edu','1393328711587','Biomedical')
```

	student_id [PK] character varying (10)	first_name character varying (50)	last_name character varying (50)	email character varying (30)	phone_number character varying (20)	dept_name character varying (20)
3	63529189	Jennifer	Glover	cheryl53@buffalo.edu	1393328711587	Biomedical
4	83989837	Kelly	Martinez	sandraharris@buffalo.edu	8947196593423	Industrial
5	14425482	Chloe	Douglas	christophertodd@buffalo.edu	0186848339694	Computer Science

2. Query to return a book

```
UPDATE book_loaned_details SET
returned_date=(SELECT CURRENT_DATE) WHERE
student_id='90785928' AND book_id=4892 AND
returned_date is NULL
```

	loan_id [PK] integer	book_id integer	branch_id integer	student_id character varying (10)	issue_date date	due_date date	returned_date date
1	4	5087	2	61706749	2023-10-17	2023-10-17	[null]
2	5	845	2	90785928	2023-10-17	2023-10-17	[null]
3	9	1872	1	90785928	2023-10-17	2023-10-17	[null]
4	8	4892	2	90785928	2023-10-17	2023-10-17	2023-10-18
5	11	5407	3	34826513	2023-10-18	2023-10-25	[null]
6	12	3993	3	92450637	2023-10-18	2023-10-25	[null]
7	13	5421	3	92450637	2023-10-18	2023-10-25	[null]
8	14	4714	3	19192829	2023-10-18	2023-10-25	[null]

3. Query to update student details:

```
UPDATE student_details SET dept_name='Computer
Science' WHERE student_id='63529189'
SELECT * FROM public.student_details WHERE
student_id='63529189'
```

	student_id [PK] character varying (10)	first_name character varying (50)	last_name character varying (50)	email character varying (30)	phone_number character varying (20)	dept_name character varying (20)
1	63529189	Jennifer	Glover	cheryl53@buffalo.edu	1393328711587	Computer Science

4. Query to update the due amount on daily basis
THIS TASK IS DONE BY A PGAGENT JOB 'refreshborrowerdetails'

```
UPDATE borrower_details as b
set due_amount=b.due_amount + s.books_exceded_dd*5
FROM (SELECT student_id,sum(books_exceded_dd) as
books_exceded_dd FROM
(SELECT book_id,student_id,sum(Current_date-
due_date) as books_exceded_dd
FROM book_loaned_details
WHERE due_date<(SELECT CURRENT_DATE) and
returned_date is NULL
GROUP BY book_id,student_id) GROUP BY student_id)
as s
where b.student_id=s.student_id;
```

id [PK] integer	student_id character varying (10)	due_amount integer	books_loaned integer
8	34826513	0	1
9	92450637	0	2
10	19192829	0	1
6	61706749	10	1
7	90785928	20	2

5. Query to loan a book to a student

```
INSERT INTO
book_loaned_details(book_id,branch_id,student_id,
issue_date,due_date)
VALUES (5087,2 ,'61706749'
,CURRENT_DATE,CURRENT_DATE+7)
```

loan_id [PK] integer	book_id integer	branch_id integer	student_id character varying (10)	issue_date date	due_date date	returned_date date
4	5087	2	61706749	2023-10-17	2023-10-17	[null]

6. Query to insert borrower details whenever a book is loaned

THIS TASK IS DONE BY
func_addupdate_borrower_details()

```
INSERT INTO
borrower_details(student_id,due_amount,books_loaned)
values('34826513',0,1)
```

id [PK] integer	student_id character varying (10)	due_amount integer	books_loaned integer
8	34826513	0	1

7. Query to insert new library branch details:

```
INSERT INTO
library_branch_details(branch_name,branch_address)
VALUES('Law Library','OBrian Hall, North
Campus,Buffalo, NY 14260-1110')
```

branch_id [PK] integer	branch_name character varying (100)	branch_address character varying (200)
1	Silverman Library	Oscar A. Silverman Library,University at Buffalo,Capen Hall, North Campus,Buffalo, NY 142...
2	Abbott Library	Abbott Hall, 3435 Main St.,South Campus,Buffalo, NY 14214-3002
3	Lockwood Memorial Library	Lockwood Memorial Library,University at Buffalo,North Campus,Buffalo, NY 14260
4	Law Library	OBrian Hall, North Campus,Buffalo, NY 14260-1110

8. Query to select the student who has not returned the loaned book after the due date, with # of book

```
SELECT student_id, count(student_id) as
books_exceded_dd
FROM book_loaned_details
WHERE due_date<(SELECT CURRENT_DATE) and
returned_date is NULL
GROUP BY student_id
```

	student_id character varying (10)	books_exceded_dd bigint
1	61706749	1
2	90785928	2

9. Query to get the student detail with department name who has loaned at least a book

```
SELECT s.first_name,s.dept_name
FROM borrower_details as b
JOIN student_details as s ON s.student_id=b.student_id
WHERE b.books_loaned>0
```

	first_name character varying (50)	dept_name character varying (20)
1	Jennifer	Electrical
2	Gary	Mechanical
3	Timothy	Computer Science
4	Katherine	Mechanical
5	Rachel	Biomedical

10. Query returns the book with their count in a given library branch

```
SELECT b.title, count(b.title) as bookCount
FROM book_details as b
JOIN library_branch_details as l on
l.branch_id=b.branch_id
WHERE l.branch_name='Law Library'
GROUP BY b.title
```

	title character varying (200)	bookcount bigint
1	A Lawyer's Dilemma	4
2	Alternative Dispute Resolution	7
3	Arbitration and Mediation	5
4	Bankruptcy Law: Your Guide	7
5	Business Law Essentials	6
6	Civil Litigation: A Guide	5
7	Civil Rights in America	5
8	Constitutional Amendments Explained	6
9	Constitutional Law Today	2

VIII. QUERY EXECUTION ANALYSIS

Ensuring consistent and real-time updates of the due amount and the number of books loaned by a student posed a challenge for us. Rather than opting for manual updates every time a student failed to return a book by the due date or for manual removal of a book from a student's account upon return, which would be both time-consuming and costly, we implemented an automated approach. This involved the creation of a scheduled job, "refreshBorrowerDetails," which runs daily to identify students who haven't returned books on time. If a student fails to meet the due date, a corresponding due amount is added to the student's account.

In addition, to maintain consistency in borrower details, we chose to automatically update or add relevant student details in the borrower details table when a book is loaned. Instead of executing each query manually for this task, we implemented an SQL trigger, "tr_addupdate_borrower_details," which activates whenever a new row is inserted into the "book_loaned_details" table.