LMS (Library management System)

- ***The goal of this project is to create an application to manage various things in a library, such as the books, keeping track of users borrowing books and etc.***
- ***A database is essential in such a system because one would need to keep track of various objects and relations between them.***
- ***The database abstracts out various things like concurrency and finding and updating, leaving us to be worried about the high level detail.***

Grp members details

Santhoshi - 112101005

Pawan Kumar - 112101031

Ruthvik - 112101018

Chitresh - 112101032

The data we are dealing with

- Books
- Authors
- Genres
- Vendor
- Publisher
- Payments
- Members
- Branches
- Books available to buy
- Has_book
- Admins
- Employees
- Send_Request

Primary tasks on the data

The library admin

The admin should be able to perform the following actions

- Create new entries of books, authors, genres etc
- Maintain the information of all the branches of the libraries.
- Buy new books for the library
- View information regarding which users currently hold which books
- Send requests to other branches to get books based on demand
- Make purchases based on requests from users (Inventory)
- No.of seats available in the library(different branches)
- Time in and time out of a student,(different branches)
- Delete entries in the database

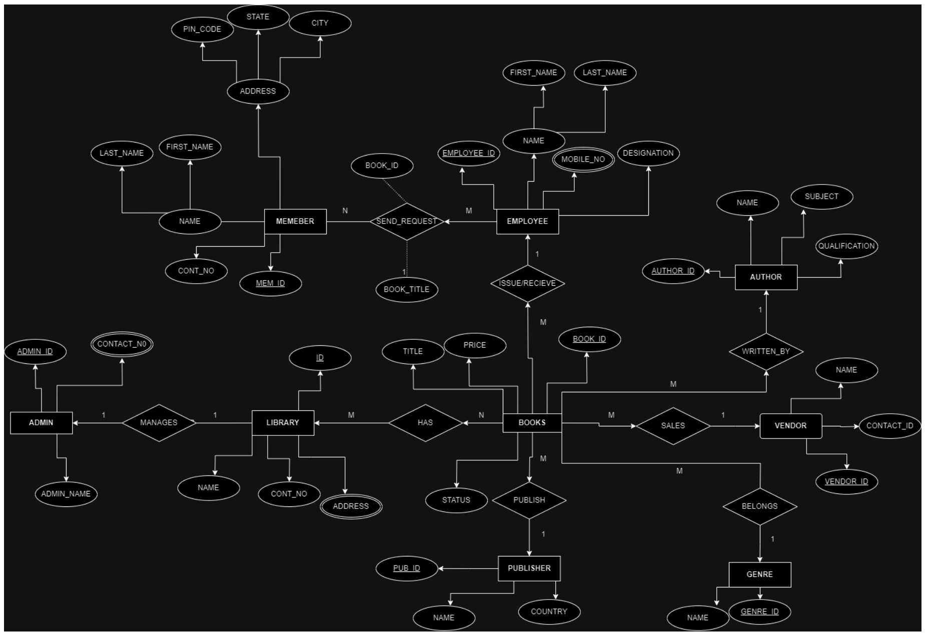


Figure 1: ER Diagram

The users

The users should be able to perform the following actions

- Borrow and return books

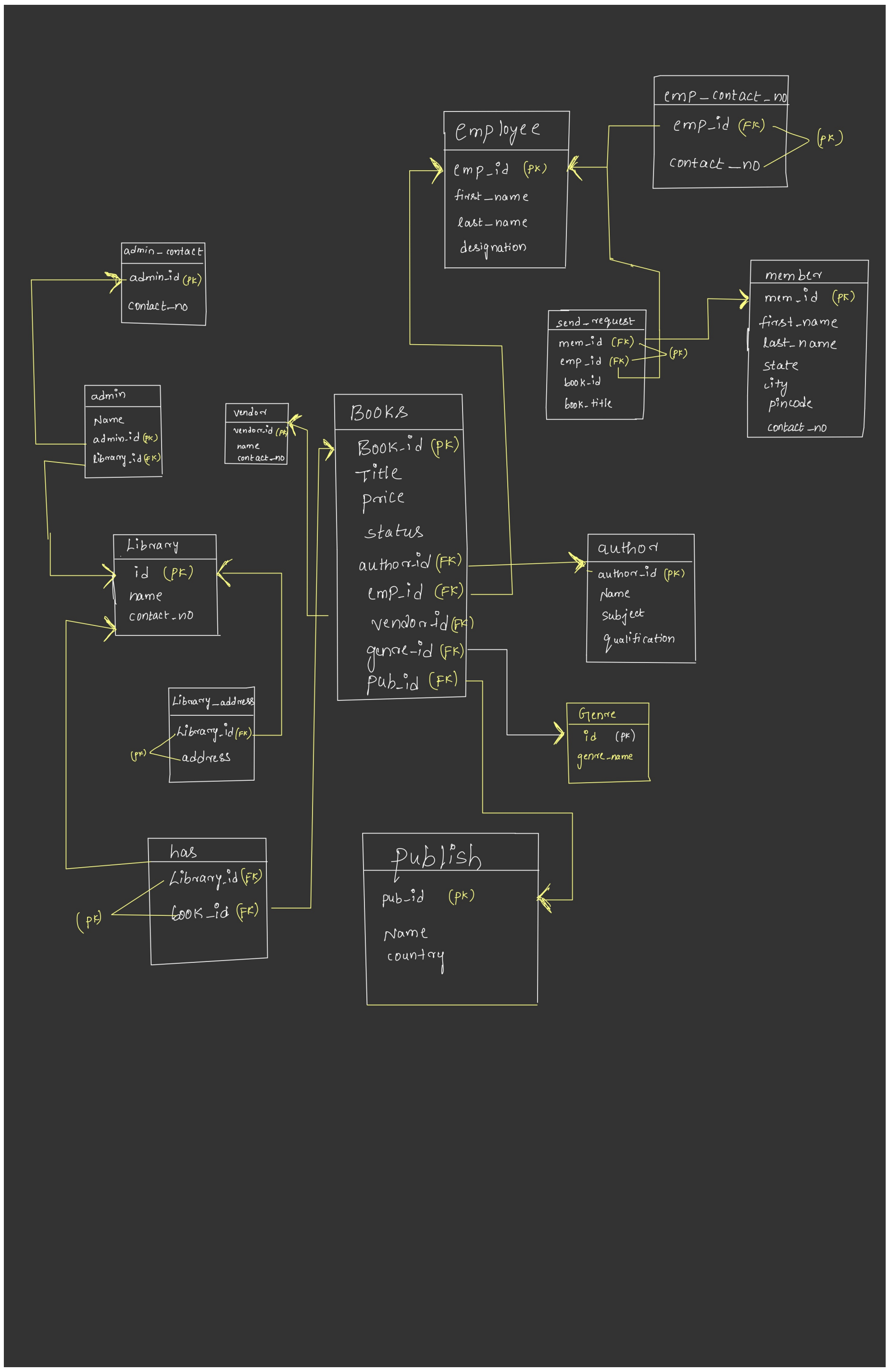


Figure 2: Relational Model

- View available books in the library
- View information regarding what books they currently hold
- Make payments
- Subscribe to membership classes
- Reserve books
- Buy books
- Request for new books to be added to the library
- Book a slot (time) for a seat in a library(particular branch of library)

Table creation

```
create table author(
    author_id int,
    author_name varchar(50),
    author_subject varchar(50),
    qualification varchar(50),
    primary key(author_id)
);


create table genre(
    genre_id int,
    genre_name varchar(30) not null,
    primary key(genre_id)
);


create table vendor(
    vendor_id int,
    v_name varchar(50) not null,
    contact_no int unique not null,
    primary key(vendor_id)
);


create table publisher(
    pub_id int,
    pub_name varchar(50) not null,
    country varchar(20) not null,
    primary key(pub_id)
);


create table members(
    mem_id int,
    first_name varchar(20) not null,
    last_name varchar(20) not null,
    state_name varchar(20) not null,
    city varchar(20) not null,
    pin_code varchar(20) not null,
    contact_no int not null,
    primary key(mem_id)

);


create table lib(
    library_id int,
    library_name varchar(50) not null,
    contact_no int not null,
    primary key(library_id)
);


create table employee(
    emp_id int,
    first_name varchar(20),
    last_name varchar(20),
    designation varchar(20),
    primary key(emp_id)
);


create table books(
    book_id int,
    book_name varchar(50) unique not null,
    book_price int not null,
    status int,
    author_id int,
    genre_id int,
    pub_id int,
    vendor_id int,
    emp_id int,
    primary key(book_id),
    foreign key(author_id) references author(author_id),
    foreign key(genre_id) references genre(genre_id),
    foreign key(pub_id) references publisher(pub_id),
    foreign key(vendor_id) references vendor(vendor_id),
    foreign key(emp_id) references employee(emp_id)
);
```

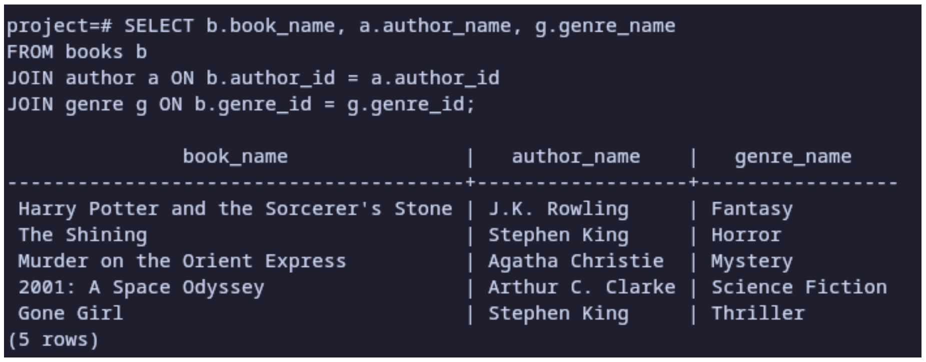
```sql
create table admin_t(
    admin_id int,
    admin_name varchar(30) not null,
    library_id int,
    primary key(admin_id),
    foreign key(library_id) references lib(library_id)
);


create table admin_contact_no(
    admin_id int,
    contact_no int not null,
    foreign key(admin_id) references admin_t(admin_id)
);


create table emp_contact_no(
    emp_id int,
    contact_no int not null,
    primary key(emp_id,contact_no),
    foreign key(emp_id) references employee(emp_id)
);

create table send_request(
    mem_id int,
    emp_id int,
    book_id int not null,
    book_title varchar(100) not null,
    primary key(mem_id,emp_id),
    foreign key(mem_id) references members(mem_id),
    foreign key(emp_id) references employee(emp_id)
);


create table library_address(
    library_id int,
    address varchar(50),
    primary key(library_id,address),
    foreign key(library_id) references lib(library_id)
);


create table has(
    library_id int,
    book_id int,
    primary key(library_id,book_id),
    foreign key(library_id) references lib(library_id),
    foreign key(book_id) references books(book_id)
)
```

Sample Queries

```
-- List all books along with their authors and genres:
SELECT b.book_name, a.author_name, g.genre_name
FROM books b
JOIN author a ON b.author_id = a.author_id
JOIN genre g ON b.genre_id = g.genre_id;

--Find all books published by a specific publisher along with their prices:
SELECT b.book_name, b.book_price
FROM books b
JOIN publisher p ON b.pub_id = p.pub_id
WHERE p.pub_name = 'HarperCollins';

-- List all books available in a specific library along with their authors:
SELECT b.book_name, a.author_name
FROM books b
JOIN has h ON b.book_id = h.book_id
JOIN lib l ON h.library_id = l.library_id
JOIN author a ON b.author_id = a.author_id
WHERE l.library_name = 'City Library';

--Find all books requested by members and the employee handling the request:
SELECT b.book_name, m.first_name AS member_first_name, m.last_name AS member_last_name,
       e.first_name AS employee_first_name, e.last_name AS employee_last_name
FROM send_request sr
JOIN books b ON sr.book_id = b.book_id
JOIN members m ON sr.mem_id = m.mem_id
JOIN employee e ON sr.emp_id = e.emp_id;

--List all books priced above a certain value along with their publishers:
SELECT b.book_name, b.book_price, p.pub_name
FROM books b
JOIN publisher p ON b.pub_id = p.pub_id
WHERE b.book_price > 10;
```

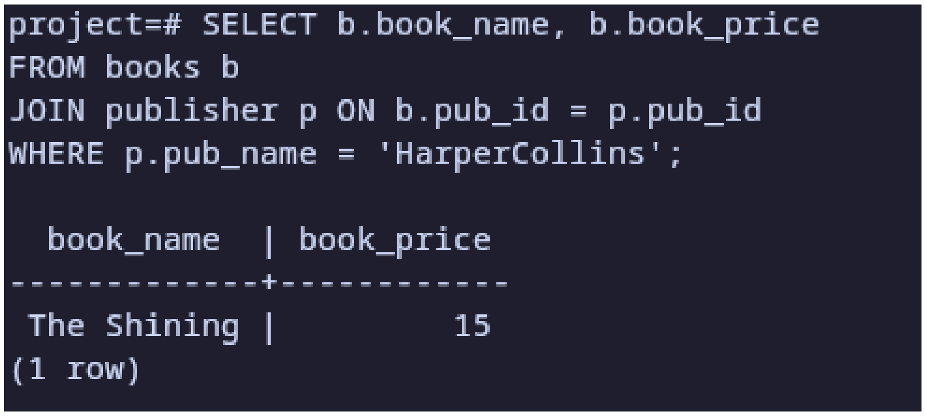
Procedures and functions

```
--Function to calculate the total price of books requested by a member
CREATE OR REPLACE FUNCTION calculate_total_price(mem_id INT) RETURNS INT AS $$
```



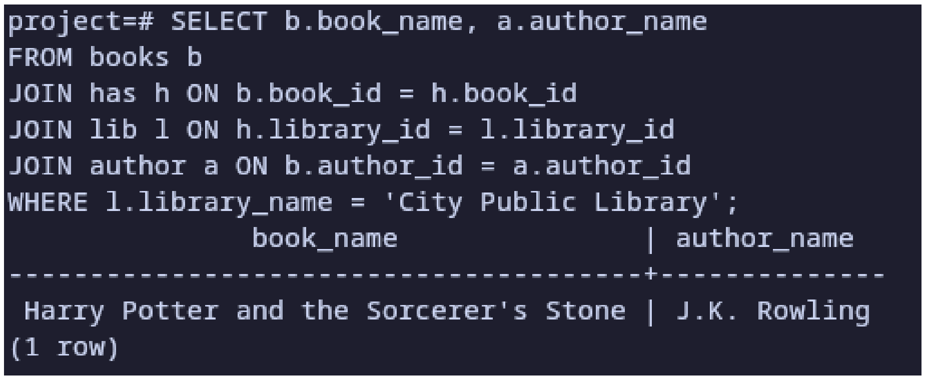
```
project=# SELECT b.book_name, a.author_name, g.genre_name
FROM books b
JOIN author a ON b.author_id = a.author_id
JOIN genre g ON b.genre_id = g.genre_id;

              book_name               |   author_name    |   genre_name
--------------------------------------+------------------+-----------------
 Harry Potter and the Sorcerer's Stone | J.K. Rowling     | Fantasy
 The Shining                          | Stephen King     | Horror
 Murder on the Orient Express         | Agatha Christie  | Mystery
 2001: A Space Odyssey                | Arthur C. Clarke | Science Fiction
 Gone Girl                            | Stephen King     | Thriller
(5 rows)
```

Figure 3: Query 1



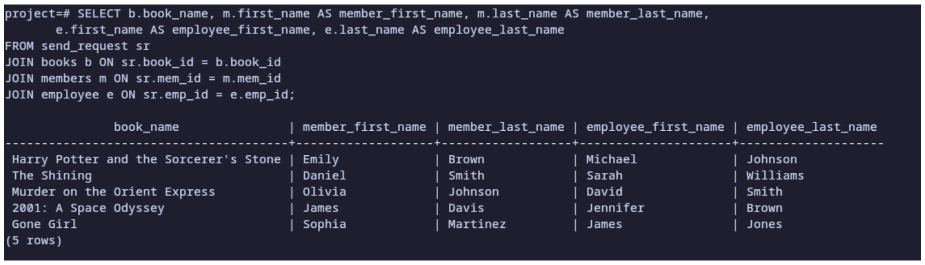
```
project=# SELECT b.book_name, b.book_price
FROM books b
JOIN publisher p ON b.pub_id = p.pub_id
WHERE p.pub_name = 'HarperCollins';

  book_name  | book_price
-------------+------------
 The Shining |         15
(1 row)
```

Figure 4: Query 2



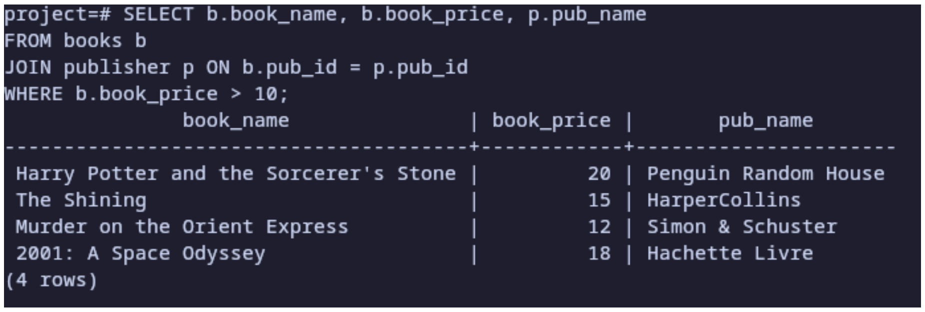
```
project=# SELECT b.book_name, a.author_name
FROM books b
JOIN has h ON b.book_id = h.book_id
JOIN lib l ON h.library_id = l.library_id
JOIN author a ON b.author_id = a.author_id
WHERE l.library_name = 'City Public Library';
               book_name               | author_name
---------------------------------------+--------------
 Harry Potter and the Sorcerer's Stone | J.K. Rowling
(1 row)
```

Figure 5: Query 3



```
project=# SELECT b.book_name, m.first_name AS member_first_name, m.last_name AS member_last_name,
       e.first_name AS employee_first_name, e.last_name AS employee_last_name
FROM send_request sr
JOIN books b ON sr.book_id = b.book_id
JOIN members m ON sr.mem_id = m.mem_id
JOIN employee e ON sr.emp_id = e.emp_id;

               book_name               | member_first_name | member_last_name | employee_first_name | employee_last_name
---------------------------------------+-------------------+------------------+---------------------+--------------------
 Harry Potter and the Sorcerer's Stone | Emily             | Brown            | Michael             | Johnson
 The Shining                           | Daniel            | Smith            | Sarah               | Williams
 Murder on the Orient Express          | Olivia            | Johnson          | David               | Smith
 2001: A Space Odyssey                 | James             | Davis            | Jennifer            | Brown
 Gone Girl                             | Sophia            | Martinez         | James               | Jones
(5 rows)
```

Figure 6: Query 4



```
project=# SELECT b.book_name, b.book_price, p.pub_name
FROM books b
JOIN publisher p ON b.pub_id = p.pub_id
WHERE b.book_price > 10;
               book_name               | book_price |       pub_name
---------------------------------------+------------+----------------------
 Harry Potter and the Sorcerer's Stone |         20 | Penguin Random House
 The Shining                           |         15 | HarperCollins
 Murder on the Orient Express          |         12 | Simon & Schuster
 2001: A Space Odyssey                 |         18 | Hachette Livre
(4 rows)
```

Figure 7: Query 5

```
DECLARE
    total_price INT := 0;
BEGIN
    SELECT SUM(book_price)
    INTO total_price
    FROM send_request sr
    JOIN books b ON sr.book_id = b.book_id
    WHERE sr.mem_id = mem_id;
    RETURN total_price;
END;
$$ LANGUAGE plpgsql;

--Procedure to update the status of a book after it has been borrowed
CREATE OR REPLACE PROCEDURE update_book_status(book_id INT, new_status INT) AS $$
BEGIN
    UPDATE books
    SET status = new_status
    WHERE book_id = book_id;
END;
$$ LANGUAGE plpgsql;
```