Library Management System Database

Introduction:

The library management system is a software application that allows librarians and library staff to manage library resources, including books, borrowers, loans, and library branches. The system helps librarians keep track of the library's inventory and loan history, as well as manage borrower information and loan policies. In this documentation, we will discuss the database schema used to create the library management system and explore the conceptual models used to design and implement the system.

Overview of the Database Schema:

The database schema for the library management system includes five tables: branches, books, library_employees, borrowers, and loans. The branches table contains information about library branches, such as their name, address, and phone number.

The books table contains information about books, including the book title, author, publisher, year of publication, and the branch where the book is located. The library_employees table contains information about library employees, such as their name, email, phone number, address, and the branch where they work.

The borrowers table contains information about library users, such as their name, email, phone number, address, and the branch they are registered with. The loans table keeps track of book loans, including the book ID, borrower ID, loan date, due date, return date, return status, and issue status.

Objective:

The objective of this code is to create a database called "library" to store information about a library's branches, books, library employees, borrowers, loans, and audit information.

The code first creates the database and then creates tables for branches, books, library employees, borrowers, and loans. It also adds a column to the library_employees table using the ALTER TABLE statement.

Next, the code creates a trigger on the library_employees table to insert audit information into a new table called employee_audit whenever a new row is inserted into the library_employees table. The code then inserts data into the library_employees, branches, books, borrowers, and loans table and selects the data from those tables.

Finally, the code creates two views: "employees_view" to display the employees' IDs, names, phone numbers, and addresses, and "branches_view" to display all columns from the branches table. It then deletes a row from the "branches_view" view where the name is "mall masr library."

Entity Relationship Model:

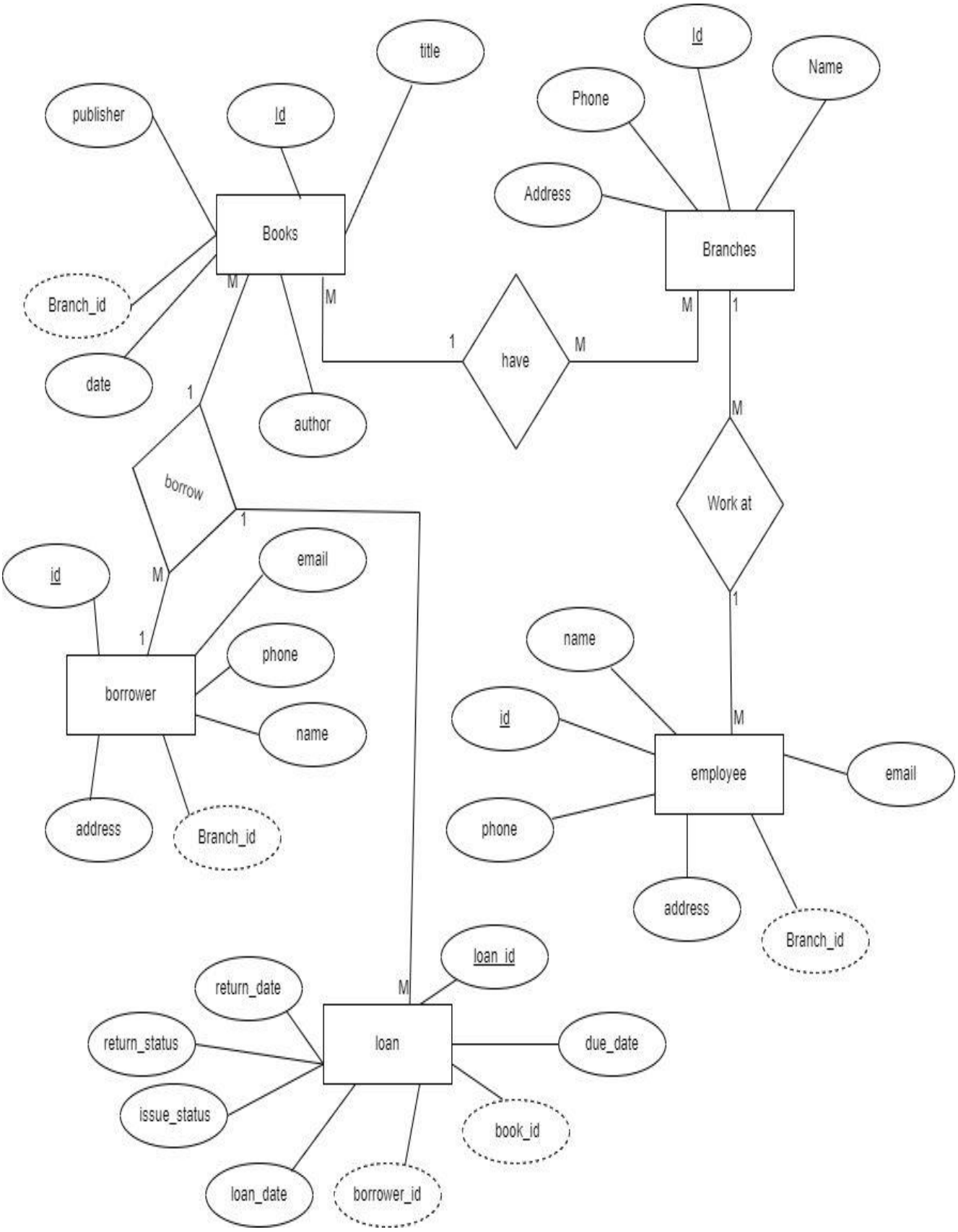
The entity-relationship (ER) model is a type of conceptual model used to describe the relationships between entities in a system. In the library management system, we can use the ER model to represent the relationships between library branches, books, library employees, borrowers, and loans.

The ER model consists of entities, attributes, and relationships. An entity represents a real-world object, such as a book or a borrower. An attribute is a property or characteristic of an entity, such as the title of a book or the name of a borrower. A relationship describes how entities are related to each other, such as the relationship between a book and a borrower when a loan is made.

In the library management system, the branches, books, library_employees, borrowers, and loans tables can be represented as entities in the ER model. The attributes of each entity are the columns in the tables. For example, the attributes of the book's entity include the book title, author, publisher, and year of publication.

The relationships between the entities can be represented using lines or arrows connecting the entities. For example, the book's entity is related to the branch's entity through the branch_id attribute, which indicates the branch where the book is located. Similarly, the loans entity is related to the books and borrower's

entities through their respective IDs, indicating which book is being loaned to which borrower.


Books
publisher
Id
title
Branch_id
date
author
Branches
Id
Name
Phone
Address
have
borrow
Work at
borrower
id
email
phone
name
address
Branch_id
employee
id
name
email
phone
address
Branch_id
loan
loan_id
return_date
return_status
issue_status
loan_date
borrower_id
due_date
book_id
M
1

Queries:

```
-- Create the database
use master
CREATE DATABASE library;


-- Use the database
USE library;

-- Create the branches table
CREATE TABLE branches (
  branch_id INT IDENTITY (1, 1) PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  address VARCHAR(255),
  phone_number VARCHAR(20),
);

-- Create the books table
CREATE TABLE books (
  book_id INT IDENTITY (1, 1) PRIMARY KEY,
  title VARCHAR(255) NOT NULL,
  author VARCHAR(255) NOT NULL,
  publisher VARCHAR(255),
  year_published DATE,-- 00,00,0000
  branch_id INT NOT NULL,
  FOREIGN KEY (branch_id) REFERENCES branches(branch_id)
);


-- Create the library_employees table
CREATE TABLE library_employees (
  employees_id INT IDENTITY (1, 1) PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255),
  phone_number VARCHAR(20),
  address VARCHAR(255),
  branch_id INT NOT NULL,
  FOREIGN KEY (branch_id) REFERENCES branches(branch_id)
);

-- Create the borrowers table
CREATE TABLE borrowers (
  borrower_id INT IDENTITY (1, 1) PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255),
  phone_number VARCHAR(20),
```

```
  address VARCHAR(255),
  branch_id INT NOT NULL,
  FOREIGN KEY (branch_id) REFERENCES branches(branch_id)
);

-- Create the loans table
CREATE TABLE loans (
  loans_id INT IDENTITY (1, 1) PRIMARY KEY,
  book_id INT NOT NULL,
  borrower_id INT NOT NULL,
  loan_date DATE NOT NULL,
  due_date DATE NOT NULL,
  return_date DATE,
  return_status VARCHAR(20),
  issue_status VARCHAR(20),
  FOREIGN KEY (book_id) REFERENCES books(book_id),
  FOREIGN KEY (borrower_id) REFERENCES borrowers(borrower_id)
);

-- add coulumn
ALTER TABLE library_employees
ADD Salary int ;
SELECT * FROM library_employees;



-- Create a new table to store audit information
CREATE TABLE employee_audit (
  audit_id INT IDENTITY (1, 1) PRIMARY KEY, -- The audit ID is an identity
column that auto-increments
  employees_id INT NOT NULL, -- The ID of the employee being audited
  audit_action VARCHAR(20) NOT NULL, -- The type of action being audited
(e.g. INSERT, UPDATE, DELETE)
  audit_timestamp DATETIME NOT NULL, -- The date and time the audit record
was created
  audit_user VARCHAR(255) NOT NULL -- The name of the user who performed the
action being audited
);

-- Create a trigger that fires after an insert operation on the
library_employees table
CREATE TRIGGER library_employees_ATI
ON library_employees
AFTER INSERT
AS
BEGIN
  -- Insert a new row into the employee_audit table for each inserted row in
the library_employees table
  INSERT INTO employee_audit (employees_id, audit_action, audit_timestamp,
audit_user)
```

```
  SELECT employees_id, 'INSERT', GETDATE(), SUSER_SNAME()
  FROM inserted;
END;

-- Select all rows from the employee_audit table to view the audit records
SELECT * FROM employee_audit;


-- insert into library employees
INSERT INTO library_employees (name, email, phone_number, address, branch_id)
VALUES
  ('John Doe', 'johndoe@example.com', '555-1234', '123 Main St', 1),
  ('Jane Smith', 'janesmith@example.com', '555-5678', '456 Elm St', 2),
  ('Bob Johnson', 'bjohnson@example.com', '555-9012', '789 Oak St', 1),
  ('Mary Brown', 'mbrown@example.com', '555-3456', '321 Maple St', 2),
  ('Tom Davis', 'tdavis@example.com', '555-7890', '654 Pine St', 2),
  ('Samantha Lee', 'slee@example.com', '555-2345', '987 Cedar St', 1),
  ('David Kim', 'dkim@example.com', '555-6789', '246 Birch St', 2),
  ('Rachel Chen', 'rchen@example.com', '555-0123', '135 Walnut St', 2),
  ('Michael Park', 'mpark@example.com', '555-4567', '864 Ash St', 1)
  select * from library_employees;



-- insert into branches
INSERT INTO branches (name, phone_number, address, employee_count)
VALUES
  ('Central Library', '555-1234', '123 Main St', 0),
  ('North Branch', '555-5678', '456 Elm St', 0),
  ('South Branch', '555-9012', '789 Oak St', 0),
  ('East Branch', '555-3456', '321 Maple St', 0),
  ('West Branch', '555-7890', '654 Pine St', 0)
  select * from branches;

-- insert into books
INSERT INTO books (title, author, publisher, year_published, branch_id)
VALUES
  ('To Kill a Mockingbird', 'Harper Lee', 'J. B. Lippincott &amp;amp; Co.',
'1960-07-11', 1),
  ('1984', 'George Orwell', 'Secker &amp;amp; Warburg', '1949-06-08', 2),
  ('The Great Gatsby', 'F. Scott Fitzgerald', 'Charles Scribner''s Sons',
'1925-04-10', 2),
  ('Pride and Prejudice', 'Jane Austen', 'T. Egerton, Whitehall', '1813-01-
28', 2),
  ('The Catcher in the Rye', 'J. D. Salinger', 'Little, Brown and Company',
'1951-07-16', 1),
  ('Animal Farm', 'George Orwell', 'Secker &amp;amp; Warburg', '1945-08-17', 1),
  ('Brave New World', 'Aldous Huxley', 'Chatto &amp;amp; Windus', '1932-06-14',
2)
    select * from books;
```

```sql
-- insert into borrowers
INSERT INTO borrowers (name, email, phone_number, address, branch_id)
VALUES
  ('John Doe', 'johndoe@example.com', '123-456-7890', '123 Main St, Anytown
USA', 1),
  ('Jane Smith', 'janesmith@example.com', '555-555-1212', '456 Elm St,
Anytown USA', 2),
  ('Bob Johnson', 'bjohnson@example.com', '555-123-4567', '789 Oak St,
Anytown USA', 1),
  ('Samantha Lee', 'slee@example.com', '555-987-6543', '321 Pine St, Anytown
USA', 2),
  ('David Kim', 'dkim@example.com', '555-555-5555', '555 Maple St, Anytown
USA', 2),
  ('Emily Davis', 'edavis@example.com', '555-555-1234', '777 Cherry St,
Anytown USA', 1),
  ('Mark Wilson', 'mwilson@example.com', '555-555-7890', '999 Walnut St,
Anytown USA', 2),
  ('Sarah Brown', 'sbrown@example.com', '555-456-7890', '888 Cedar St,
Anytown USA', 1)
      select * from borrowers;

-- insert into loans
INSERT INTO loans (book_id, borrower_id, loan_date, due_date, return_date,
return_status, issue_status)
VALUES
  (4, 5, '2022-01-01', '2022-01-15', '2022-01-13', 'Returned', 'Issued'),
  (5, 6, '2022-02-02', '2022-02-16', NULL, NULL, 'Issued'),
  (6, 7, '2022-03-03', '2022-03-17', '2022-03-12', 'Returned', 'Issued'),
  (7, 8, '2022-04-04', '2022-04-18', NULL, NULL, 'Issued'),
  (8, 9, '2022-05-05', '2022-05-19', '2022-05-18', 'Returned', 'Issued'),
  (9, 10, '2022-06-06', '2022-06-20', NULL, NULL, 'Issued'),
  (10, 11, '2022-07-07', '2022-07-21', '2022-07-19', 'Returned', 'Issued')
       select * from loans;



-- Create the book_index index on the books table
CREATE INDEX book_index
ON books (title);

-- View information about the book_index index using the sys.indexes view
SELECT *
FROM sys.indexes
WHERE name = 'book_index';



-- Create view for library_employees with employees_id,name ,
phone_number,address
```

```sql
create view employees_view as
select employees_id,name , phone_number,address from library_employees ;
SELECT * FROM employees_view;


-- Create view for branches table with all columns
create view branches_view
as select * from branches
select * from branches_view

delete from branches_view
where name = 'mall masr library'
SELECT * FROM branches_view;
```

Conclusion:

In this documentation, we have discussed the database schema, conceptual models, and use-case diagram for the library management system. The schema includes five tables: branches, books, library employees, borrowers, and loans, and it allows the system to manage library resources, borrower information, and loan history. The ER model, object model diagram, and use-case diagram provide different perspectives on the system's structure and behavior, and they can be used to design, implement, and test the system. Overall, the library management system is a valuable tool for librarians and library staff to manage and maintain their library's resources and services.

The above code creates a database called `library` and several tables within it, including `branches`, `books`, `library employees`, `borrowers`, and `loans`. It also adds a column to the `library employees` table using `ALTER TABLE`, creates a new table called `employee audit` to store audit information, and creates a trigger on the `library employees` table to insert audit

records into the `employee audit` table when new records are inserted.

The code also inserts data into each of the tables using `INSERT INTO` statements and creates an index on the `books` table using `CREATE INDEX`. It then creates two views, `employees view` and `branches view`, to display specific columns from the `library employees` and `branches` tables, respectively.

Finally, the code deletes a row from the `branches view` view using `DELETE FROM`.

Overall, this code sets up a basic database schema for a library and populates it with some sample data. The views provide a way to display specific columns from the tables without exposing all of the underlying data. The trigger and audit table provide a way to track changes to the `library employees` table over time.

Results:

1. The creation of a new database called "library" using the CREATE DATABASE command.
2. Switching to use the newly created "library" database using the USE command.

3. The creation of four tables: "branches", "books", "library_employees", and "borrowers", which store information about the library's resources, employees, and borrowers.
4. The creation of a fifth table called "loans" that stores information about loans made by the library.
5. The addition of a new column "Salary" to the "library_employees" table using the ALTER TABLE command.
6. The creation of a new table called "employee_audit" to store audit information, including the ID of the employee being audited, the type of action being audited, the date and time the audit record was created, and the name of the user who performed the action being audited.
7. The creation of a trigger that fires after an insert operation on the "library_employees" table and inserts a new row into the "employee_audit" table for each inserted row in the "library_employees" table.
8. Insertion of sample data into the "library_employees", "branches", "books", "borrowers", and "loans" tables to populate the database with sample data.
9. creates an index on the `books` table using `CREATE INDEX`.
10. The creation of two views: "employees_view" and "branches_view", which provide a convenient way to retrieve the data fromthe "library_employees" and "branches" tables respectively. The "employees_view"

only selects the "employees_id", "name", "phone_number", and "address" columns from the "library_employees" table, while the "branches_view" selects all columns from the "branches" table.

11. An attempt to delete a row from the "branches_view" table where the "name" column is equal to "mall masr library", but since the "branches_view" is just a view and not a base table, this operation will not affect the actual "branches" table.