## Step (2): Project Specifications

The library database design includes the following main entities (corresponding to database tables) and their interrelationships:

1. Item = {itemId (Primary Key), type, title, author, isBorrowed}
This table stores detailed information about all items in the library. Attributes include a unique item ID, type, title, author, and borrowing status.

2. Customer={customerId (Primary Key), name, birthDate, address, email}
This table stores detailed information about all registered library members. It includes attributes like member ID, name, birthdate, address, and email.

3. Borrow ={borrowId (Primary Key), customerId (Foreign Key - references Customer), itemId (Foreign Key - references Item), dueDate}
This table handles the management of borrowed items. It associates items with members and includes attributes like borrow ID, member ID, due date, and item ID.

4. Fine={fineId (Primary Key), customerId (Foreign Key - references Customer), amount} This table handles overdue items and associated fines. It includes attributes like fine ID, member ID, and the fine amount.

5. Event={eventId (Primary Key), name, date, roomId (Foreign Key - references Room), type}
This table manages all events held by the library. It includes attributes like event ID, date, room ID, and event type.

6. Room={roomId (Primary Key), type, capacity}
This table stores detailed information about the library's social rooms. It includes attributes like room ID, type, and capacity.

7. RecommendAudience={eventId (Primary Key, Foreign Key - references Event), customerId (Primary Key, Foreign Key - references Customer)}

This table stores information about specific audiences that the library recommends for events. It includes attributes like event ID and member ID.

8. EventAttend= {eventId (Primary Key, Foreign Key - references Event), customerId (Primary Key, Foreign Key - references Customer)}
This table associates members with the events they attend. It includes attributes like event ID and member ID.

9. Employee= {employeeId (Primary Key), name, position, startDate, salary}
This table stores detailed information about library employees. It includes attributes like employee ID, name, position, start date, and salary.

10. PreviousEmployee= {previousEmployeeId (Primary Key), name, position, startDate, endDate, salary}
This table stores detailed information about previous library employees. It includes attributes like previous employee ID, name, position, start date, end date, and salary.

11. FutureItem= {futureItemId (Primary Key), type, addDate, title, author}
This table keeps records of potential future acquisitions for the library. It includes attributes like future item ID, type, add date, title, and author.

Entity Relationships

Item and Customer are associated via the Borrow table: Members can borrow multiple items, and items can be borrowed by multiple members.

Customer and Fine: Members can have multiple fines, each fine is associated with a single member.

Event and Customer are associated via the EventAttend table: Members can attend multiple events, and events can be attended by multiple members.

Event and Customer are associated via the RecommendAudience table: Specific members can be recommended for certain events.

Event and Room: Each event is held in a specific social room.

Employee and PreviousEmployee: Records of the current and past library employees are kept.

This database design enables the library to effectively manage its items, track borrows and overdue fees, manage events, maintain records of employees, as well as record information about potential future additions.

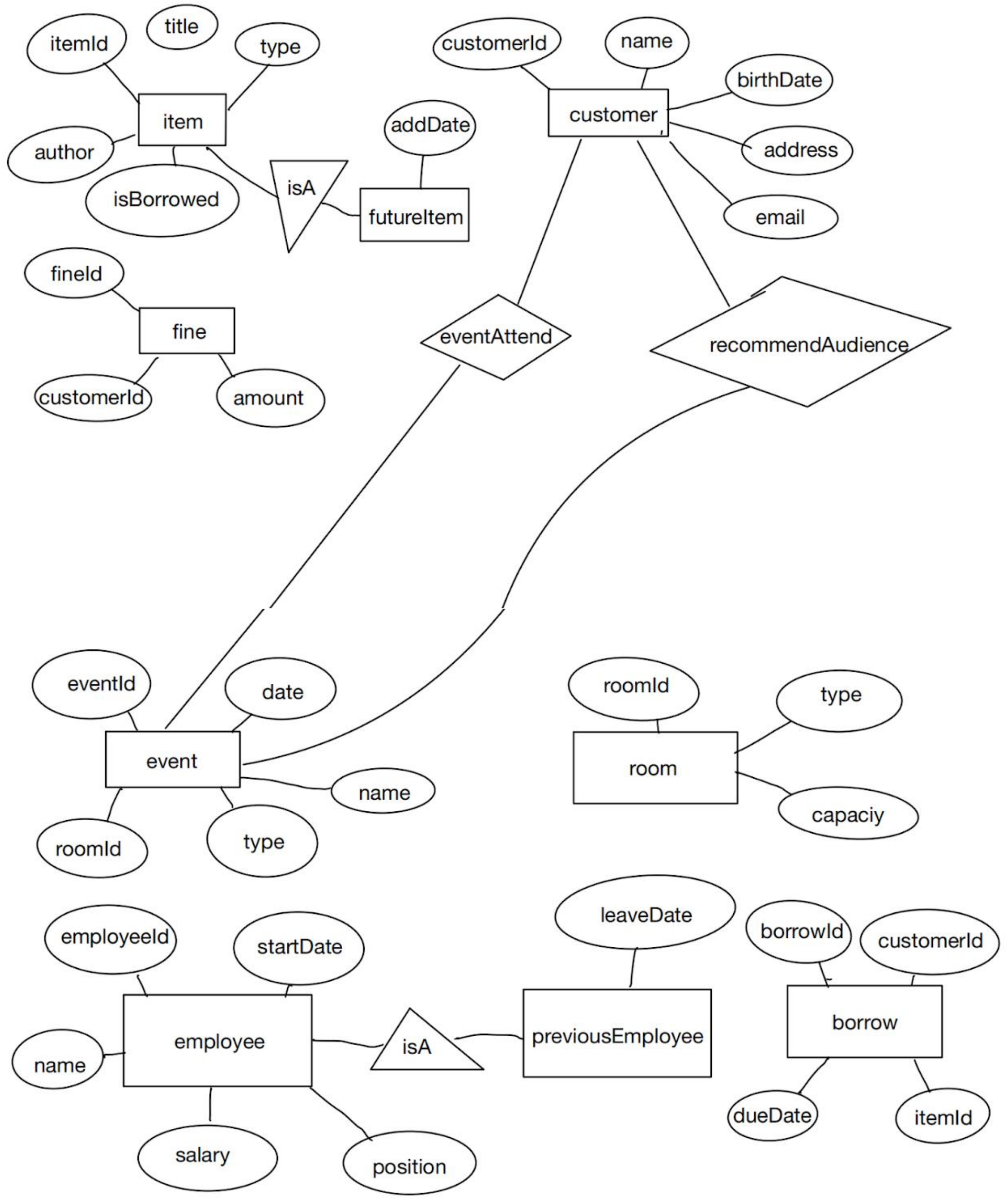
```
%sql sqlite:///Library.db
CREATE TABLE item(
    itemId INTEGER PRIMARY KEY,
    type TEXT NOT NULL,
    title TEXT NOT NULL,
    author TEXT,
    isBorrowed BOOLEAN DEFAULT FALSE
);

CREATE TABLE customer(
    customerId INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    birthDate TEXT,
    address TEXT,
    email TEXT
);
CREATE TABLE borrow (
    borrowId INTEGER PRIMARY KEY,
    customerId INTEGER,
        dueDate TEXT,
        itemId INTEGER,
        FOREIGN KEY (customerId) REFERENCES customer(customerId),
        FOREIGN KEY (itemId) REFERENCES item(itemId)
    );
CREATE TABLE fine(
        fineId INTEGER PRIMARY KEY,
        customerId INTEGER,
        amount REAL,
        FOREIGN KEY (customerId) REFERENCES customer(customerId)
    );
CREATE TABLE event(
        eventId INTEGER PRIMARY KEY,
        name TEXT,
        date TEXT,
        roomId INTEGER,
        type TEXT,
        FOREIGN KEY (roomId) REFERENCES room(roomId)
    );
 CREATE TABLE room(
        roomId INTEGER PRIMARY KEY,
        type TEXT,
        capacity INTEGER
    );
CREATE TABLE recommendAudience(
        eventId INTEGER,
        customerId INTEGER,
        PRIMARY KEY (eventId, customerId),
        FOREIGN KEY (eventId) REFERENCES event(eventId),
        FOREIGN KEY (customerId) REFERENCES customer(customerId)
    );
CREATE TABLE eventAttend(
        eventId INTEGER,
        customerId INTEGER,
        PRIMARY KEY (eventId, customerId),
        FOREIGN KEY (eventId) REFERENCES event(eventId),
        FOREIGN KEY (customerId) REFERENCES customer(customerId)
    );
CREATE TABLE employee(
        employeeId INTEGER PRIMARY KEY,
        name TEXT NOT NULL,
        position TEXT,
        startDate TEXT,
        salary REAL
    );

CREATE TABLE previousEmployee(
        employeeId INTEGER,
        startDate TEXT,
        endDate TEXT,
        salary REAL,
        FOREIGN KEY (employeeId) REFERENCES employee(employeeId)
    )


CREATE TABLE futureItem(
        itemId INTEGER,
        type TEXT,
        addDate TEXT,
        title TEXT,
        author TEXT,
        FOREIGN KEY (itemId) REFERENCES item(itemId)
    )
```

## Step (3): E/R Diagrams (15 points)



## Step (4): Does your design allow anomalies?

To check for anomalies, we need to analyze all non-trivial functional dependencies in our database and ensure that our tables are free of undesirable functional dependencies.

A table is in BCNF (Boyce-Codd Normal Form) if for every non-trivial functional dependency X -> Y, X is a superkey.

Below are the main tables and their functional dependencies:

Item: {itemId} -> {type, title, author, isBorrowed}

For the Item table, the primary key is itemId, and all other fields are functionally dependent on it. Hence, this table is in BCNF.

Customer: {customerId} -> {name, birthDate, address, email}

For the Customer table, the primary key is customerId, and all other fields are functionally dependent on it. Hence, this table is in BCNF.

Borrow: {borrowId} -> {customerId, dueDate, itemId}

For the Borrow table, the primary key is borrowId, and all other fields are functionally dependent on it. Hence, this table is in BCNF.

Fine: {fineId} -> {customerId, amount}

For the Fine table, the primary key is fineId, and all other fields are functionally dependent on it. Hence, this table is in BCNF.

Event: {eventId} -> {name, date, roomId, type}

For the Event table, the primary key is eventId, and all other fields are functionally dependent on it. Hence, this table is in BCNF.

Room: {roomId} -> {type, capacity}

For the Room table, the primary key is roomId, and all other fields are functionally dependent on it. Hence, this table is in BCNF.

RecommendAudience: {eventId, customerId} -> {}

For the RecommendAudience table, the primary key is a combination of eventId and customerId. Hence, this table is in BCNF.

EventAttend: {eventId, customerId} -> {}

For the EventAttend table, the primary key is a combination of eventId and customerId. Hence, this table is in BCNF.

Employee: {employeeId} -> {name, position, startDate, salary}

For the Employee table, the primary key is employeeId, and all other fields are functionally dependent on it. Hence, this table is in BCNF.

PreviousEmployee: {previousEmployeeId} -> {name, position, startDate, endDate, salary}

For the PreviousEmployee table, the primary key is previousEmployeeId, and all other fields are functionally dependent on it. Hence, this table is in BCNF.

FutureItem: {futureItemId} -> {type, addDate, title, author}
For the FutureItem table, the primary key is futureItemId, and all other fields are functionally dependent on it. Hence, this table is in BCNF.

From this analysis, it's clear that all tables are in BCNF, as for every non-trivial functional dependency, the determinant is a superkey. Therefore, the design does not allow anomalies such as insertion, deletion, or update anomalies.