Mini Project: Library Database

CMPT 354

Angus Kan
Yoel Yonata

2. Project Specification

Introduction

The main objective of this mini project is to design and create a database management system for a library. The database application will have to keep records of items that the library has, events that are being held and personnel records. The application will also be able to perform simple tasks such as borrowing and returning items to the library, browsing and registering for future events and requesting for help.

Requirement analysis

- What data is going to be stored?
    - The database will keep a record of all items in the library. Any donated items and new releases will be kept track separately. Customers will have their IDs and their names kept together with all personnel working for the library. Events that are held in the library will be kept together with all its details (location, date, recommended audiences, etc.)
- What are we going to do with the data?
    - The data will be used to keep track of items such as when customers borrow them and return them. The data can be used for keeping track of due dates and fine as well. Customers will be able to access the database of events held in the libraries and register for them.
- Who should access the data?
    - Customers and personnel should be able to access the data. They should only be able to perform actions that are allowable by the application. Such as finding an item, borrowing, returning, finding an event, registering for an event and requesting for help.

E/R Diagram description

- An **item** has attributes title, author’s name, item type, and count, with title and author’s name being the primary key.
    - Each item has a type (print books, online books, magazines, etc.)
    - Customers can borrow only borrow an item if it’s available (count is greater than zero)
- A **Future Item** is a subclass of item, retaining all the attributes and has an additional attribute arrivalDate.
    - There is a constraint on the attribute arrivalDate, such that the arrivalDate cannot be in the past.
    - Items that are donated by customers will be added into the future item table with an arrivalDate to 1 week from the donation day.
    - arrivalDate has a constraint where the arrivalDate cannot be in the past

- **Borrows** is the relation between item and Customers with attributes fine and dueDate.
    - When a customer borrows an item from the library, an entry in borrows will be created.
    - A customer will only be able to borrow one copy of a book at a time.
    - The tuple entry in borrows will be deleted once the book is returned.
    - dueDate has a constraint where the dueDate cannot be in the past
- **Customers** have an ID, firstName, lastName.
    - Customers will be able to perform the following actions:
        - Search, borrow, return and donate an item
        - Search and register for an existing event
        - Volunteer for a library at a certain location
        - Ask for help by giving a request
- A **personnel** has an ID, firstName, lastName, role, salary and location
    - A personnel can also be a customer
    - When a customer volunteers for a location, the customer will have the role of volunteer and a salary of zero.
- An **event** has a name, type, audiences, roomNumber, date and location.
    - Has a constraint where the date of the event cannot be in the past
- **Registers** is a relation between customers and events.
    - When a customer registers for an event, an entry will be added to registers.
- **Request List** has a requestID and the request.
    - When a customer requests for help from a librarian, a random requestID will be assigned to the request and an entry will be created in the request list.

3. E/R Diagram

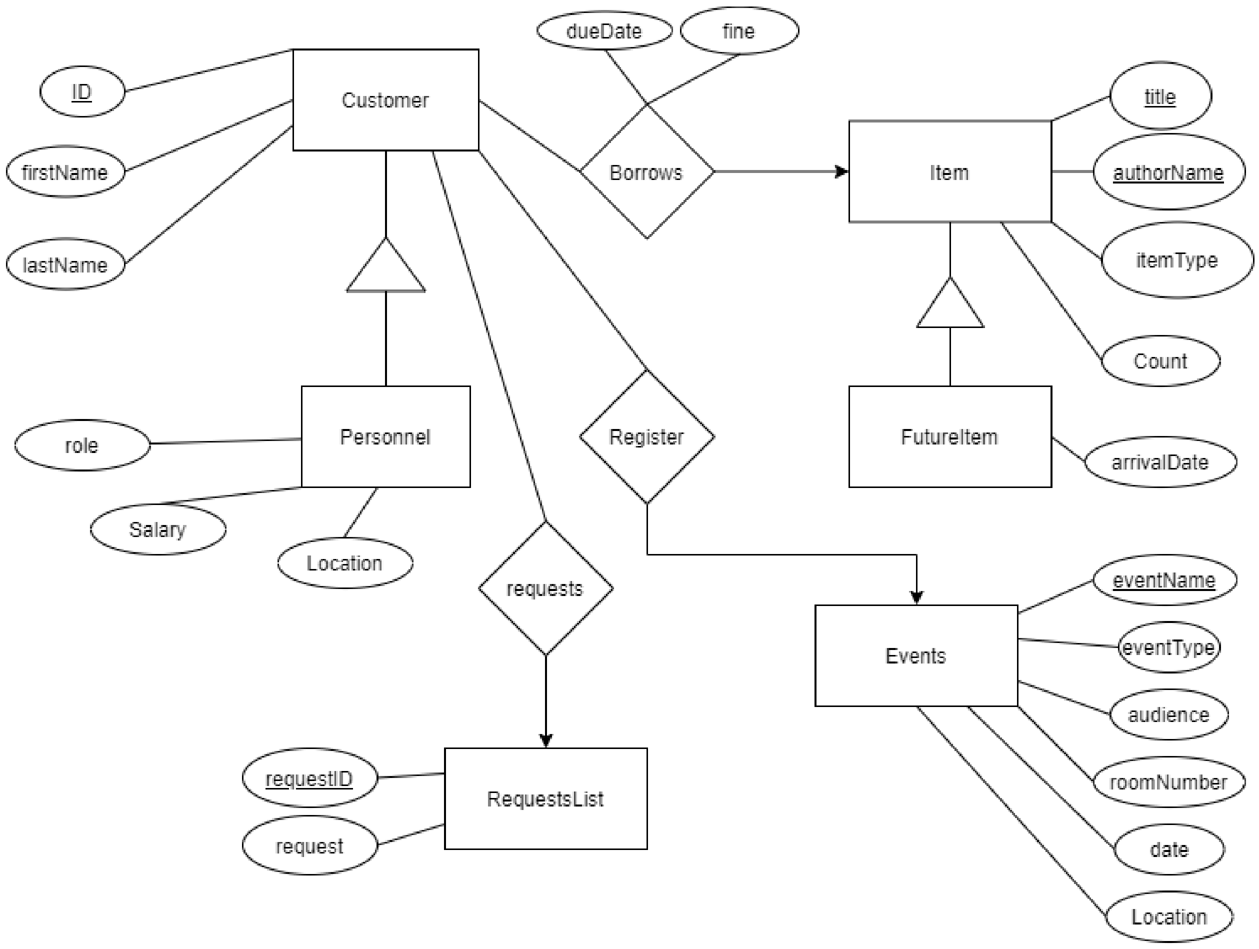

Figure 1. E/R diagram of the library database design

4. Anomalies Analysis

- Customer = {<u>ID</u>, firstName, lastName}
- Personnel = {<u>ID</u>$^{FK\text{-}Customer}$, firstName, lastName, role, salary, location}
- Borrows = {<u>ID</u>$^{FK\text{-}Customer}$,<u>title</u>$^{FK\text{-}Item}$, <u>authorName</u>$^{FK\text{-}Item}$,fine, dueDate}
- Item = {<u>title</u>, <u>authorName</u>, itemType, count}
- FutureItem = {<u>title</u>$^{FK\text{-}Item}$, <u>authorName</u>$^{FK\text{-}Item}$, itemType, count, arrivalDate}
- Events = {<u>eventName</u>, eventType, audience, roomNumber, date, location}
- Register = {<u>ID</u>$^{FK\text{-}Customer}$, <u>eventName</u>$^{FK\text{-}Events}$ }
- Requests = {<u>requestID</u>$^{FK\text{-}RequestList}$,ID$^{FK\text{-}Customer}$}
- RequestList = {<u>requestID</u>, request}

Dependencies
ID → firstName, lastName, role, title, authorName, eventName, requestID
title, authorName → itemType, count, arrivalDate
eventName → eventType, audience, roomNumber, date, location
requestID → request

The database we designed presents no anomaly because it is in a BCNF. We can verify that this is in BCNF from the dependencies mentioned above. As we can see above, the left side of every non trivial functional dependency is a superkey. From the stated functional dependencies, there are no more bad functional dependencies. Therefore our design satisfies the condition to be in BCNF.

5. Schema

There are two triggers within our database, one for adding items into the item table and the other for increasing fines. The first trigger will happen when the arrivalDate reaches the current date, which will create a new tuple in the table item with the values of the item that has just arrived. After it has been added into the item table, the tuple containing the old information will be deleted in the table FutureItem. The second trigger happens when the dueDate reaches the current date, which will increase the fine by 10.