Library Management System

1. Project Specifications

Introduction

This document outlines the specifications for a comprehensive Library Management System database designed to capture the real-world operations of a multi-function library. The primary purpose of this database is to efficiently manage the library's collection of materials, automate the borrowing process, track patron interactions, manage events, and support administrative functions. The following sections detail the entities, relationships, constraints, and functional requirements used to guide the implementation of this database system.

Requirements Description

Item Management

1. The library has different types of items available for patrons to borrow. For example, print books, online books, scientific journals, magazines, CDs, DVDs, and vinyls
    a. Items that do not fall in this category are defined by 'Other'
2. All patrons are able to submit requests for items that they would like to have at the library

Borrowing System

1. Patrons can borrow one item at a time. The borrowed item must be returned by the due date.
2. The loan period for all items is 28 days.
3. If an item is not returned by the due date, patrons are subject to fines.
    a. Fines are calculated based on the replacement cost of an item
    b. If an item is not returned within 14 days of its due date, the item is considered lost
4. Patrons must pay fines prior to borrowing an item
    a. If an item is considered 'lost', the Patron must pay 100% of the item's replacement cost
    b. Once the replacement cost is paid for, the item is available to be borrowed again

Event Management

1. The library hosts events organized by the library staff.
2. Different library events are recommended for specific audiences.
3. Library events are held throughout different library rooms.
4. Only patrons are allowed to attend library events.

Staff Management

1. All staff members must be patrons

2. Managers of different library departments handle personnel and record keeping for personnel.
    a. Each staff record requires a type of
    b. Volunteers do not have access to any of the staff records
3. Managers approve of the acquisition requests from patrons.
4. Library staff have access to all patron information.
5. Staff members can quit at any time. However, method of requesting leave is different depending on staff position
    a. Volunteers are able to quit at any time.
    b. Non-volunteers must speak to upper management

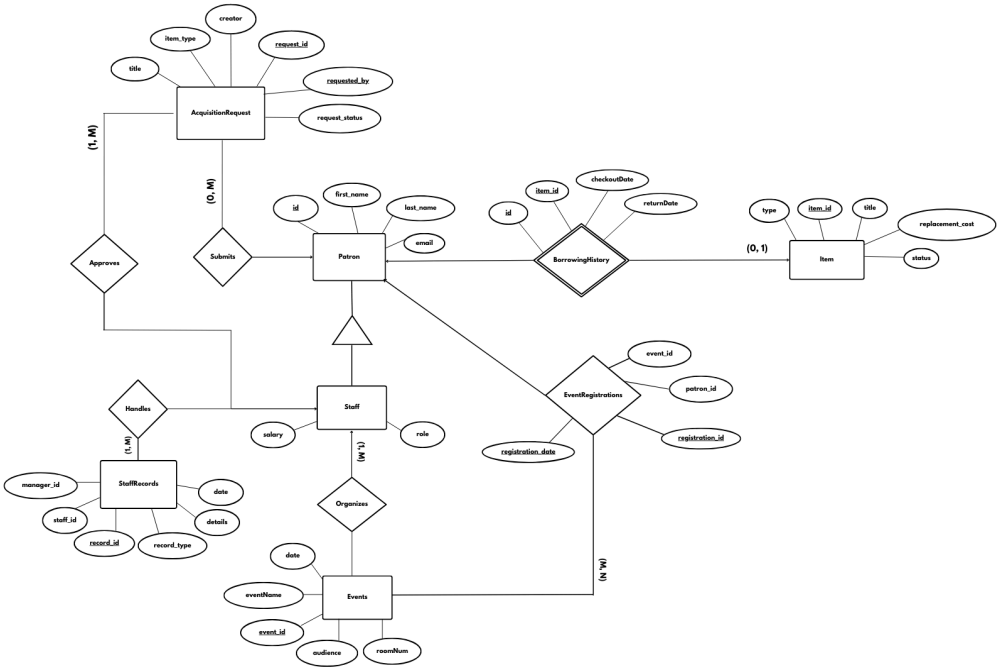
Patron Management

1. The library requires the name and email of a patron and assigns a unique ID to each patron
    a. The registration process randomly generates a unique patron ID that is 4-digits long
        i. In the library database, tables are populated with 1-digit patron IDs for the sake of simplicity
2. Patrons have access to information on the available items at the library, events to attend, can request for help, pay fines, and donate items to the library

Item Donations Management

1. All patrons can donate an item to the library
2. Donated items have a replacement_cost of $0
3. Upon ‘donation’, items are immediately available to be borrowed by patrons

2. E/R Diagram



3. Anomaly Analysis

The library database has the following tables:

- Patron = {id, first_name, last_name, email}
- Staff = {id$^{FK\text{-}Patron}$, position, salary}
- Items = {item_id, title, creator, type, replacement_cost, status}
- BorrowingHistory = {id$^{FK\text{-}Patron}$, item_id$^{FK\text{-}Item}$, checkoutDate, returnDate}
- AcquisitionRequest = {request_id, requested_by$^{FK\text{-}Patron}$, request_status, item_type, creator, title}
- Events = {event_id, organizer$^{FK\text{-}Staff}$, eventName, date, roomNum, audience}
- StaffRecords = {record_id, staff_id$^{FK\text{-}Staff}$, record_type, details, date}
- EventRegistrations = {registration_id, event_id$^{FK\text{-}Events}$, patron_id$^{FK\text{-}Patron}$, registration_date}

Functional Dependencies

Patron Table

- FD: id → first_name, last_name, email
- Primary Key: id
- id^{+} = {id, first_name, last_name, email}
    - The closure of id includes all the attributes of the Patron table → id is a superkey

Staff Table

- id → position, salary
- Primary Key: id
- id^+ = {id, position, salary}
    - The closure of id includes all attributes of the Staff table → id is a superkey

Items Table

- FD: item_id → item_id, title, creator, type, replacement_cost, status
- Primary Key: item_id
- $item\_id^+$ = {item_id, title, type, replacement_cost, status}
    - The closure of item_id includes all attributes of the Items table → item_id is a superkey

BorrowingHistory Table

- FD: id, item_id →id, item_id, checkoutDate, returnDate
- Composite Primary Key: id, item_id
- $(id, item\_id)^+$ = {id, item_id, checkoutDate, returnDate}
    - The closure (id, item_id) contains all attributes of the BorrowingHistory table → (id, item_id) is a superkey

AcquisitionRequest Table

- FD: request_id → request_id, requested_by, request_status, item_type, creator, title
- Primary Key: request_id
- $request\_id^+$ = {request_id, requested_by, request_status, item_type, creator, title}
    - The closure of request_id contains all attributes of the AcquisitionRequest table → request_id is a superkey

Events Table

- FD: event_id → event_id, organizer, eventName, date, roomNum, audience
- Primary Key: event_id
- $event\_id^+$ = {event_id, organizer, eventName, date, roomNum, audience}
    - The closure of event_id includes all attributes of the AcquisitionRequest table → event_id is a superkey

EventRegistrations Table

- FD: registration_id → registration_id, event_id, patron_id, registration_date
- Primary Key: registration_id
- $registration\_id^+$ = {registration_id, event_id, patron_id, registration_date}
    - The closure of registration_id includes all attributes of the EventRegistration table → event_id is a superkey

StaffRecords Table

- FD: record_id → record_id, staff_id, record_type, details, date

- Primary Key: record_id
- record_id$^+$ = {record_id, staff_id, record_type, details, date}
    - The closure of record_id contains all attributes of the AcquisitionRequest table → record_id is a superkey

All tables are in BCNF. The determinant of every functional dependency is a superkey. Additionally, there are no partial or transitive dependencies.

Potential Anomalies

Update anomalies

- The id attribute of Staff table is a foreign key referencing Patron id, every staff member must exist in the Patron first
    - **Constraint**: Requirement for incoming staff members to be patrons first would avoid this anomaly
- Updates to personal information (e.g., email change) must be done in Patron table only
    - **Constraint**: Patron information updates are only made in one place, causing limited flexibility when handling different contact details for staff vs. patrons

Deletion anomalies

- If a Patron is deleted, their Staff record would also be deleted
    - **Deletion constraint**: If a staff member resigns, their staff record would be deleted first. Therefore, that staff member would have the option of remaining as a Patron
        - This results in their Patron record to be intact.
    - Additionally, in a real-world scenario, a the corresponding Patron tuple would only be deleted first if they were not a staff member

By implementing proper constraints and deletion rules, the database maintains data integrity and consistency while avoiding anomalies. However, there is a trade-off between data integrity and flexibility. While the structure prevents anomalies, it requires careful enforcement of constraints to avoid inconvenience in a real-world scenario.

4. Implementation

The database was implemented using SQLite3 in Jupyter Notebook. A lightweight Python GUI was implemented to simulate how a real library system would be used by non-technical users. The GUI made core workflows (searching items, managing patrons, and updating records) easier to test and demonstrate the end-to-end interaction between the user interface and underlying SQLite database.