Interlibrary Loan System Database

Alex Neibart

1. INTRODUCTION

The day-to-day operation of a library involves tracking a large amount of data. Libraries handle information relating to books in their catalogue, patrons, authors, and the status of particular items, including whether those items have been withdrawn or reserved.

The amount of data to be handled is greater in contexts where it is necessary to track the contents and patrons of multiple libraries, as is the case with systems that allow patrons to request and borrow books from participating libraries of which they are not personally members.

An effective database for an interlibrary loan system should ideally make it possible to find the location of a particular book, determine what books are currently in stock, provide the balance of a particular cardholder's account, list the books written by a particular author, and determine whether there are any outstanding reservations on a particular item.

In this paper, I present a database design applicable to interlibrary loan systems. This design consists of a relational database system, standard SQL queries, an API capable of implementing those queries, and a program which generates large data sets.

2. ENTITY-RELATIONSHIP MODEL

Figure 1 shows an entity-relationship diagram outlining the data model used in this database design.

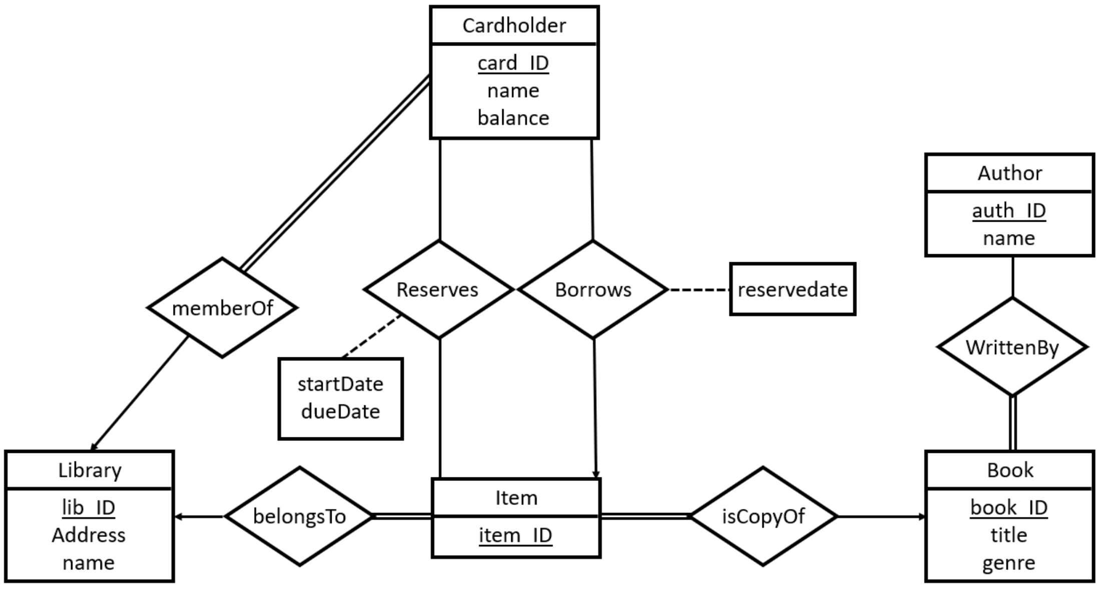

Figure 1: Entity-Relationship Diagram

2.1 Entities

The entities included in the database's entity-relationship model and the attributes of those entities are as follows:

Library: A library that participates in the interlibrary loan system.
lib_ID: The library's primary key.
Address: The library's street address.
Name: The library's name.

Cardholder: A member of one of the libraries.
card_ID: The cardholder's primary key
name: The cardholder's name.
balance: The amount that the cardholder currently owes as a result of late fees, damage, or lost books.

Author: The author of one or more library books.
auth_ID: The author's primary key.
name: The author's name.

Book: An abstract representation of a book that libraries may own one or more copies of.
book_ID: The book's primary key.
title: The book's title.
genre: The genre to which the book belongs.

Item: A physical book belonging to one of the libraries in the system.
item_id: Discriminates the item from other copies of the same book.

2.2 Relationships:

The relationships included in the database's entity-relationship model and the attributes of those relationships are as follows:

member: Cardholders must be members of a single library, but a library can have any number of cardholders.

belongsTo: Items must belong to a single library, whereas libraries can own any number of items.

isCopyOf: Items must be instances of a single book, whereas books can be instantiated as any number of items.

WrittenBy: Books must have been written by one or more authors, whereas authors may have written any number of books.

borrows: Cardholders can borrow any number of items, but each item can be borrowed by at most one cardholder at a time.
startDate: When the borrowed item was withdrawn.
dueDate: When the borrowed item is due back.

reserves: Cardholders can reserve any number of items, and items can be reserved by any number of cardholders (for example, if there is a list of five people who are all waiting for the same book).
reserveDate: The date a reservation was made.

3. RELATIONAL DATABASE SCHEMA

The tables in the relational database schema and the attributes assocated with them are as follows:

Library: Represents libraries participating in the interlibrary loan system.
libID: An integer that acts as the primary key of the Library table.
address: A 50-character varchar.
name: A 50-character varchar.

Cardholder: Represents cardholders belonging to libraries in the interlibrary loan system.
cardID: An integer that acts as the primary key of the Cardholder table.
name: A 30-character varchar.
balance: A float with two decimal places.

Author: Represents authors of books owned by libraries in the interlibrary loan system.
authID: An integer that acts as the primary key of the Author table.
name: A 50-character varchar.

Book: Represents titles that libraries may own copies of.
bookID: An integer that acts as the primary key of the Book table.
title: A 50-character varchar.
genre: A 20-character varchar.

Item: Represents physical books belonging to specific libraries.
itemID: An integer that acts as the primary key of the Item table.
isCopyOf: An integer. It is a foreign key that references bookID from the Book table.
belongsTo: An integer. It is a foreign key that references libID from the Library table.

WrittenBy: Represents relationships between authors and books showing which books were written by which authors.
bookID: An integer. It is a foreign key that references bookID from the Book table.
authID: An integer. It is a foreign key that references authID from the Author table.

Borrows: Represents relationships between cardholders and items showing which items have been withdrawn by which cardholders.
cardID: An integer. It is a foreign key that references cardID from the Cardholder table.
itemID: An integer. It is a foreign key that references itemID from the Item table.
startDate: A date.
dueDate: A date.

Reserves: Represents relationships between cardholders and items showing which cardholders have reserved which items.
cardID: An integer. It is a foreign key that references cardID from the Cardholder table.
itemID: An integer. It is a foreign key that references itemID from the Item table.
reserveDate: A date.

4. QUERIES AND API

The API contains six public functions, each of which implement a single SQL query. These functions are runQ1, runQ2, runQ3, runQ4, runQ5, and runQ6.

4.1 runQ1

runQ1 retrieves the titles of books that belong to a specified genre.

Parameter: The specified genre.
Returns: List containing each row of the query result represented as a string.

SQL:
```
SELECT b.title
FROM Book b
WHERE b.genre = ?;
```

4.2 runQ2
runQ2 retrieves titles of books which were written by the author(s) with a specifed name.

Parameter: The specified author name.
Returns: List containing each row of the query result represented as a string.

SQL:
```
SELECT b.title
FROM Book b, Author a, WrittenBy w
WHERE b.bookID = w.bookID
AND a.authID = w.authID
AND a.name = ?;
```

4.3 runQ3
runQ3 selects cardholders from the library with a specified name who have balances greater than 0.

Parameter: The specified library name.
Returns: List containing each row of the query result represented as a string.

SQL:
```
SELECT c.*
FROM Cardholder c, Library l
WHERE c.memberOf = l.libID
AND l.name = ?
AND c.balance > 0;
```

4.4 runQ4
runQ4 counts the total number of books that have been borrowed by all cardholders belonging to the library with a specified name.

Parameter: The specified library name.
Returns: List containing each row of the query result represented as a string.

SQL:
```
SELECT COUNT(*) as num_books
FROM Library l, Cardholder c, Borrows b
WHERE l.libID = c.memberOf
AND c.cardID = b.cardID
AND l.name = ?;
```

4.5 runQ5
runQ5 displays the number of reservations pertaining to each book in the system, provided that a copy of that book been reserved at least once.

Parameters: None.
Returns: List containing each row of the query result represented as a string.

SQL:
```
SELECT b.bookID, b.title, COUNT(*) as
num_reservations
FROM Reserves r, Book b, Item i
WHERE r.itemID = i.itemID
AND i.isCopyOf = b.bookID
GROUP BY b.bookID, b.title
HAVING num_reservations > 0;
```

4.6 runQ6

runQ6 selects the IDs and titles of all books that are available at the library with a specified name but not elsewhere.

Parameters: The specified library name.
Returns: List containing each row of the query result represented as a string.

SQL:

```
SELECT DISTINCT b.bookID, b.title
FROM Library l, Book b, Item i
WHERE l.libID = i.belongsTo
AND i.isCopyOF = b.BookID
AND l.name = ?
AND NOT EXISTS (
SELECT *
FROM Library l2, Book b2, Item i2
WHERE l2.libID = i2.belongsTo
AND i2.isCopyOF = b2.BookID
AND NOT l2.name = ?
AND b.bookID = b2.bookID
);
```

5. LARGE DATA SET

The large data set was generated with a java program named DataInserter.java. This program generated data by randomly combining words, names, street names, and book genres listed in corresponding text files.

When run, DataInserted.java first clears the current contents of the database, then generates and inserts random data line-by-line. When generating IDs that act as primary keys, it starts from 0 and iterates upwards until it has produced an amount of data specified by the class constant corresponding to a given table. For certain tables it will then insert one row of data that is not randomly generated, so that there are parameters that will predictably work when testing queries.

So as to comply with foreign key constraints, the program started by generating data for the Library table and proceeded to fill Cardholder, Author, Book, Item, WrittenBy, Borrows, and Reserves, in that order.

Although the database itself allows items that have not been borrowed to be reserved, the program follows the intended purpose of the Reserves table by creating reservations only for items that it has already placed on the Borrows table.

6. CONCLUSION

The most straightword way to improve the database design would be to add more queries, particularly queries that determine the library a withdrawn book should next be returned to, which libraries currently have a particular book in stock, which cardholder was the first to reserve a particular book, and which books are currently overdue.

It may also be possible to add update functionality to the API and to add means of accounting for media other than books.

Although most aspects of creating the database design went relatively smoothly, one challenge encountered early in the project was distinguishing between book titles, which came to be represented by the Book table, and specific copies of a book, now represented by the Item table.