CS4416 Project 2016

Library Database:

**Group Members:**

James O’Shaughnessy: 15187497

* Entity relationship Design
* Tables
* Schema.sql
* Trigger.sql

Brian Walsh: 13147811

* Procedures.sql
* Query.sql
* Functional dependencies

**Description:**

The database is used in a library setting where books are taken out on loan for a set period.

When a book is withdrawn the database records:

1. The member who withdraws the book.
2. The date the book must be returned.

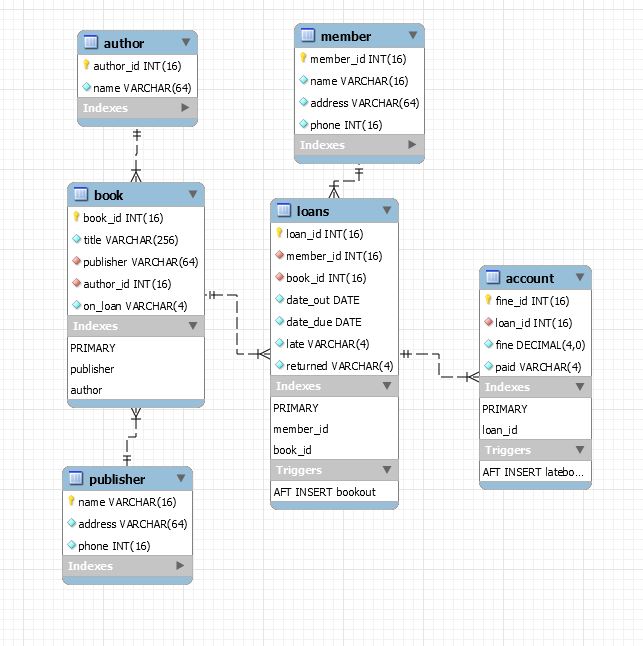
If a member is late on returning a book.

1. A fine is issued to the member account.
2. The book is registered as not returned.

The database has tables that hold records of the books, authors and publishers along with the member details and account information.

1. Queries can be used to access the database for book selection by the member.
2. The library staff can use the database to look up member who have fines outstanding or over due books.
3. The database keeps an inventory of all the books in stock, either withdrawn or not.

**Entity Relationship Diagram:**



**Functional Dependencies:**

**Member**

Member\_id -> name, address, phone

name -> phone

**Loans**

book\_id, member\_id -> date\_out, date\_due

date\_out -> date\_due

**Book**

book\_id -> title, publisher

**Publisher**

Name -> address, phone

**Account**

Member\_id -> overdue\_date, fine, book\_id

**Author**

book\_id -> author

**Tables:**

**Loans**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Loan\_id(pk)** | **Member\_id** | **Book\_id** | **Date\_out** | **Date\_due** | **late** | **returned** |
| INT | Varchar | Int | Date | Date | Varchar | Varchar |
| 1 | 2 | 43 | 12/03/16 | 19/03/16 | no | yes |
| 5 | 7 | 76 | 17/03/16 | 24/03/16 | yes | no |

Loan\_id 🡪 date\_out, date\_due

**Account**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fine\_id(pk)** | **Loan\_id** | **Fine** | **paid** |
| INT | INt | Decimal | Varchar |
| 65 | 43 | 2 | yes |
| 66 | 49 | 2 | no |

**Book**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Book\_id(pk)** | **Title** | **Publiser** | **Author\_id** | **On\_loan** |
| INT | VArchar | Varchar | Int | Varchar |
| 53 | The firm | orion | 456 | yes |
| 37 | The client | orion | 456 | no |

**Publisher**

|  |  |  |
| --- | --- | --- |
| **Name(pk)** | **Adrress** | **phone** |
| Varchar | Varchar | Int |
| Orion | 123 high st. Limerick | 546464646 |
| Oxord | 34 london rd. Liverpool | 4567646464 |

**Member**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name\_id(pk)** | **Name** | **Adrress** | **phone** |
| Int | Varchar | Varchar | Int |
| 64 | Tom Jones |  | 8658585 |
| 45 | Jane Smith |  | 46676757 |

**Author**

|  |  |
| --- | --- |
| **Author\_id(pk)** | **name** |
| Int | Varchar |
| 355 | JK Rowling |
| 322 | Stephen King |

In each table there are no transitive dependencies, which satisfy 3rd Normal Form.

**Justifications:**

Justifications of queries and views:

* For the query to find users with a fine greater then 5, is useful when you have a set charge that goes up 5, every time a certain time period expires after due date is expired. E.g. increases by 5 for every two days after due date is passed. This shows the users who need to be pressed for a return of a book and collection of fine as fine is increasing regularly.
* The query which returns members who have loaned more than one book is useful to keep track of members who may hold a higher fine then a normal customer who borrows one, this customer may receive more reminders of return dates.
* The search query of author who has written the most books would suggest popularity of these books with customers and prompt initiative to invest in more books of this particular author. Can be viewed with the created view [Most Prolific Author].

Justification of procedures:

* The procedure for members with a significant fine could be used in the database to show customers who are serious candidates for legal action or being banned from loaning books again in the future.
* The procedure to show members who have not loaned any books could be used if you want to have a policy of only loaning to members who either never loaned before or have made sure to return all previous loans.

Justification of triggers:

* When a book is late the loan id is registered in the account table, a trigger then changes the status of the book as late in the loans table.
* When a book is withdrawn the loan id and other relevant information is stored in the loans table, this then triggers a change to the on\_loan status from ‘no’ to ‘yes’ in the book table. This would render the book as unavailable in any future searches for available books in the database.