ECM2419

Library Design Report

ALEXANDER COSSINS

UNIVERSITY OF EXETER

User Requirement Specification

Data Requirements

Library Branches

The data requirements on Library Branches include the ability to be able to store more than one library branch. These library branches must also store a name to represent them. Additional storage attributes in this database will include an ID for the branch as well as an address.

Books

The database must hold records of books, required stored data related to these books includes: book title, number of copies and which library branch the book is located in. The database constructed will also contain an ISBN for the book and a category to group the books into genres.

Authors

From the user requirement analysis, it can also be deduced that information on Authors must also be stored. The essential requirements being the author's name, and which book(s) that particular author writes. The database will also include the author's date of birth.

Borrowers

User analysis clearly defines the need for information to be held on the borrowers themselves. The only clear essential requirements are the name and email of the borrower, however I shall assume that more information will be sensible for the database. Since the database deals with fines I shall assume a borrower's credit card number will be stored. The database will store this in plain text for the time being, when moving from local server to a functioning online server, it will be a necessity to encrypt this attribute with hashes of these credit card numbers as well as salt added. This is important to prevent the impact of a data leak or intrusion. I will also assume that a phone number and a borrower ID will be useful for the database.

Loans

It is clear that some state information will need to be held about the loans, specifically the borrower involved in the loan, the date the loan was made, and the date the book was due back according to the loan. The database will also need to deal with calculating fines from loans. Due to this, it will be necessary to record when the book was actually returned, as well as keep track of which fines have been paid. In order to deal with this I shall add an attribute to record when fines linked to a specific loan have been paid.

Publishers

Although no direct data storage necessities, I assumed storing the publisher for books would be a sensible option. I will include their name, phone number and address.

Transaction Requirements

Below is detailed the required interaction a user will need with the database, separated by transaction type. The interactions are specified in a general case here, but the current functionality of the SQL files will be tailored to a certain case. However, the files could easily be converted into well defined SQL procedures to take variable inputs.

Data Queries

1.1	Identify the total number of copies of books each library branch has under a certain author.
1.2	Identify how many copies of a book are available at a certain branch.
1.3	Identify all borrowers who have not borrowed a book since a certain date.
1.4	List the book title, borrower's name and email address linked to each loaned book due today at a specific branch.
1.5	List book Title, Borrowed Date, and Date due for all overdue loaned books by a certain borrower.
1.6	Identify the total outstanding fine for a Borrower.

Data Manipulation

2.1	Update the date a loaned book was returned, allowing a borrower to return a book linked to a loan.
2.2	Update the current number of copies representing a certain book at a particular library branch, used in case of damaged stock.
2.3	Update the date a fine was paid, for when a total fine was paid off by a borrower.

Data Insertion

3.1	Insert a new loan record linking a borrower and book from a particular branch.
3.2	Insert a new book record, with an associated stock count in specified branches.

Conceptual Model Design

Entity Choice and Reasoning

LibraryBranch

I determined that `LibraryBranch` should be a strong entity itself, it will use an `ID` as a primary key. I did not use the name as a primary key here due to the possibility of two branches possibly having the same name.

BookInventory

Since the amount of copies of each book is unique to each branch, I have made a weak entity `BookInventory` to portray this. The cardinality between a branch and book inventory would be one to many. This Is because a library branch can have stock records for many books (or none), but these stock records must be unique to a single branch.

Book

I designed `Book` to be a strong entity, the primary key would be the `ISBN` of the book since this value holds guaranteed uniqueness, whereas a book title does not. `Book` will have a one to many relationship with `BookInventory`, since one book could have multiple stock records in many different library branches.

Author

Author is a strong entity, with the `authorID` being the primary key since there may be many different authors with the same name. Because many authors may contribute work on one book, there is a many to many cardinality relationship between `Book` and `Author`.

Publisher

Publisher being a strong entity has the `name` as a primary key, I assume here that no two publishers will have the same name due to copyright regulations. `Publisher` will have a one to many relationship with `Book` since one publisher can release many different books, but a specific version of a book can only be published by one publisher.

Borrower

Borrower is a strong entity with an `ID` as a primary key, due to many borrowers being able to have the same name. An alternate key is also available for borrowers, being their `creditCardNo`. The reason why I did not choose this as the primary key was because credit card numbers can change, and picking a non static primary key that may change is bad practise, creating possible database difficulties in future. I separated name into a multi-value attribute (first and last name), this allows processing of this information easier avoiding the need to deal with spaces. This will be especially useful if in the future a form is used for data entry into the database.

Loan

I have modelled loaning as a complex ternary relationship, involving `Borrower`, `LibraryBranch` and `Book`. All three of them have a strict cardinality participation of 1, since a particular loan can only be made by one borrower, involving a single book from a certain branch. This relationship is not guaranteed to have already existed however, meaning a branch, book or borrower may not have any loans yet.

Assumptions

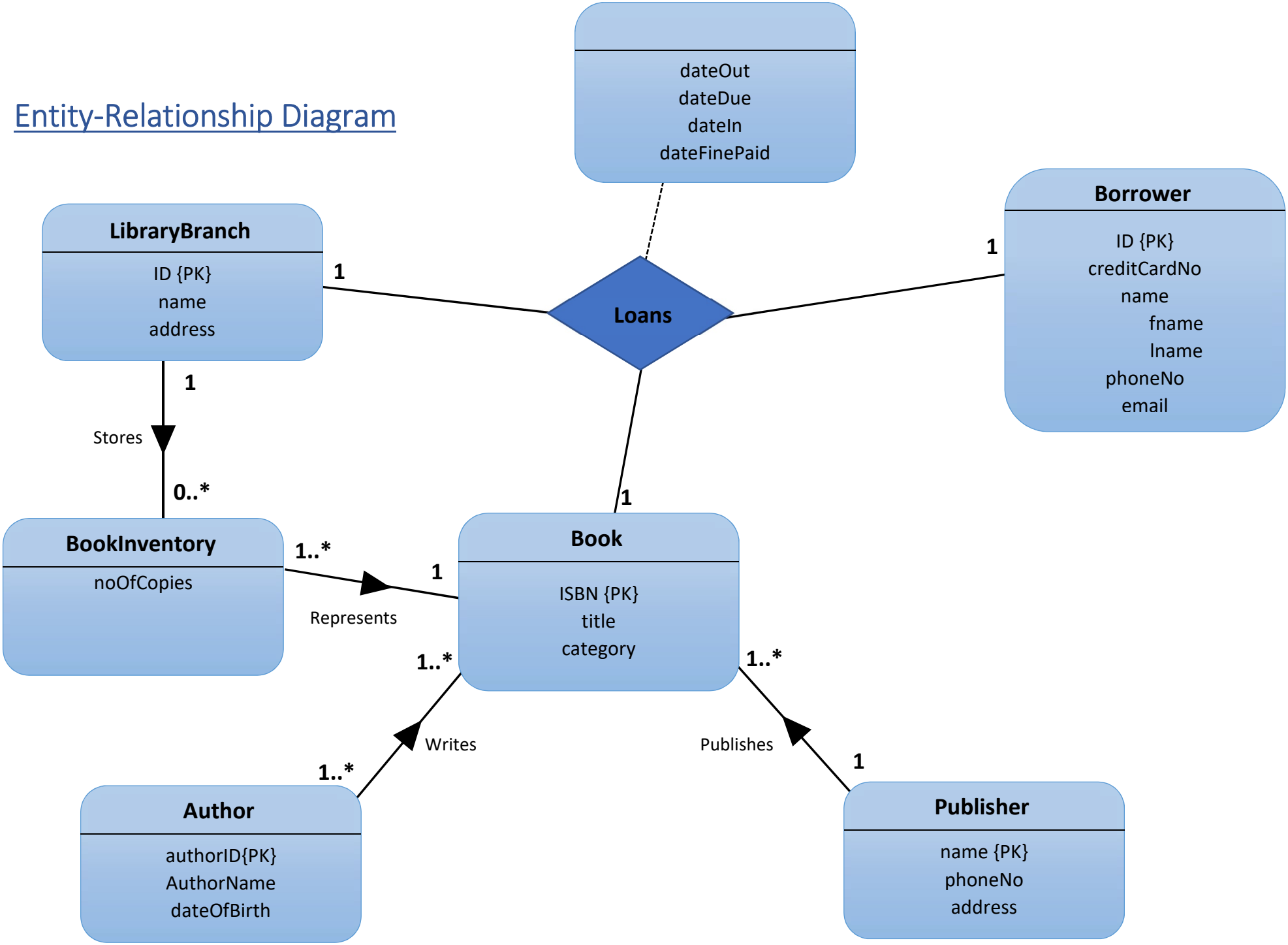
- A borrower that is on the system is a member of all branches, and can borrow a book from any of these branches.
- A borrower can borrower many copies of the same book from the same branch or different branches simultaneously.
- Upon a borrower returning a book of which they have multiple copies currently loaned out, it is assumed they are returning the oldest loan and to the branch they originally took out that loan from.
- Upon a borrower paying their outstanding fine, it is assumed the loans they are paying the fine for have already been returned, or are being returned at the same time as paying. In other words, a fine cannot be paid unless the borrower brings all overdue books with them. The reason for this assumption is because once a borrower pays for a fine, the fine accumulation will stop on that loan.
- It is assumed that for a book record to exist on the system, it will have at least one book inventory record relating to a branch.
- I assume that the fine accumulated per day of a book being overdue is £0.25.

Dynamic Calculation

Instead of hard coding data relating to the fines and currently available copies into the database as derived attributes, I went with the approach of dynamically calculating these upon request via queries. The advantage of this being that updating the database requires very few changes I.E a single `dateIn` update for a book loan return, also it makes the database more flexible. A disadvantage of this however, is that data queries are more convoluted. For example, to calculate available copies of a book I must evaluate number of current active loans and subtract this from the current known stock level.

Entity-Relationship Diagram representing all that has been mentioned above is found on the next page:

Entity-Relationship Diagram



Logical Model Design

Relational Model

Author (authorID, AuthorName, dateOfBirth)

Primary Key: authorID

LibraryBranch (ID, name, address)

Primary Key: ID

Borrower (ID, creditCardNo, fName, lName, phoneNo, email)

Primary Key: ID

Publisher (name, phoneNo, address)

Primary Key: name

Book (ISBN, title, category, publisherName)

Primary Key: ISBN

Foreign Key: publisherName references **Publisher** (name)

BookInventory (ISBN, branchID, noOfCopies)

Primary Key: ISBN, branchID

Foreign Key: branchID references **LibraryBranch** (ID)

Foreign Key: ISBN references **Book** (ISBN)

Contributors (ISBN, authorID)

Primary Key: ISBN, authorID

Foreign Key: ISBN references **Book** (ISBN)

Foreign Key: authorID references **Author** (ID)

Loan (branchID, borrowerID, ISBN, dateOut, dateDue, dateIn, dateFinePaid)

Primary Key: branchID, BorrowerID, ISBN, dateOut

Foreign Key: branchID references **LibraryBranch** (branchID)

Foreign Key: borrowerID references **Borrower** (ID)

Foreign Key: ISBN references **Book** (ISBN)

Relational Model Reasoning

Book

Having a binary one to many relationship with `Publisher` where `Book` has the ‘many’ participation means that `Book` retains the foreign key reference to `Publisher`.

BookInventory

Having a binary one to many relationship with both `LibraryBranch` and `Book` where `BookInventory` has the ‘many’ in both cases participation means that `BookInventory` retains the foreign key reference to both `LibraryBranch` and `Book`.

Contributors

Since `Author` and `Book` had a many to many cardinality relationship as described before, a new relationship entity was needed to be introduced for the logical model. Here I introduce `Contributors` which holds a foreign key reference to both `Book` and `Author`, allowing safe database representation of multiple authors contributing to the same book.

Loan

In order to transform this complex ternary relationship into a logical design, I needed to make the entity Loan. This entity holds a foreign key reference to all three of the relationship participants: `Book`, `Borrower` and `LibraryBranch`. In order to achieve a unique composite primary key, I had to choose: `branchID`, `borrowerID`, `ISBN` and `dateOut`. This allows loans to be distinguished by their borrower, branch and book. However if I did not also add `dateOut` to the composite key, a borrower would not be able to loan multiple copies of the same book at the same branch on the same day. Also `dateOut` must be of datetime datatype to distinguish between multiple loans of the same book on the same day at the same branch.

With the conclusion of this report here, I have enough planning to construct the Physical layer.