User Requirements

Member
A member represents a person who is a member of the library system. An assumption made is that one member is a member of all library branches in the database.
The data stored on each member is their name (forenames and surname), their date of birth and their email. They also have a member ID, which is unique across the Library system, to identify them.

Branch
A branch represents a library within a library system.
The data stored on each branch is the name of the branch, and their location. They also have a branch ID, which is unique across the Library system, to identify each branch.

Author
An author represents a person who has written a book.
The data stored on each author is their name (forenames and surname). They also have an author ID, which is unique across the Library system, to identify them.

Publisher
A publisher represents a company who has published a book.
The data stored on each publisher is their company name. They also have a publisher ID, which is unique across the Library system, to identify them.

Book
A book represents each version of a book available in the library system.
The data stored on each book includes the ISBN, title, genre, cover type (hardback or paperback), the ID of the author of the book and the ID of the publisher of the book.
The book is uniquely referenced by its ISBN, which is unique to each version of the book. This means that a library system can have **different versions** of a book as their ISBN is different.
An assumption made for this database is that a book can only be written by a **single** author and published by a **single** publisher.

Stock
Stock represents the number of books that are stored by a specific branch.
The data stored is the book via its ISBN, the branch of which the stock belongs to via its branch ID, and the number of copies of the book in question. The stock is uniquely identified by the ISBN of the book and the branch ID of the branch that owns the stock.

Loans
A loan represents when a single member borrows a single book from a single branch. An assumption made is that every loan has a duration of 4 weeks. If the book has not been returned within this 4-week period, then the member has incurred a fine.

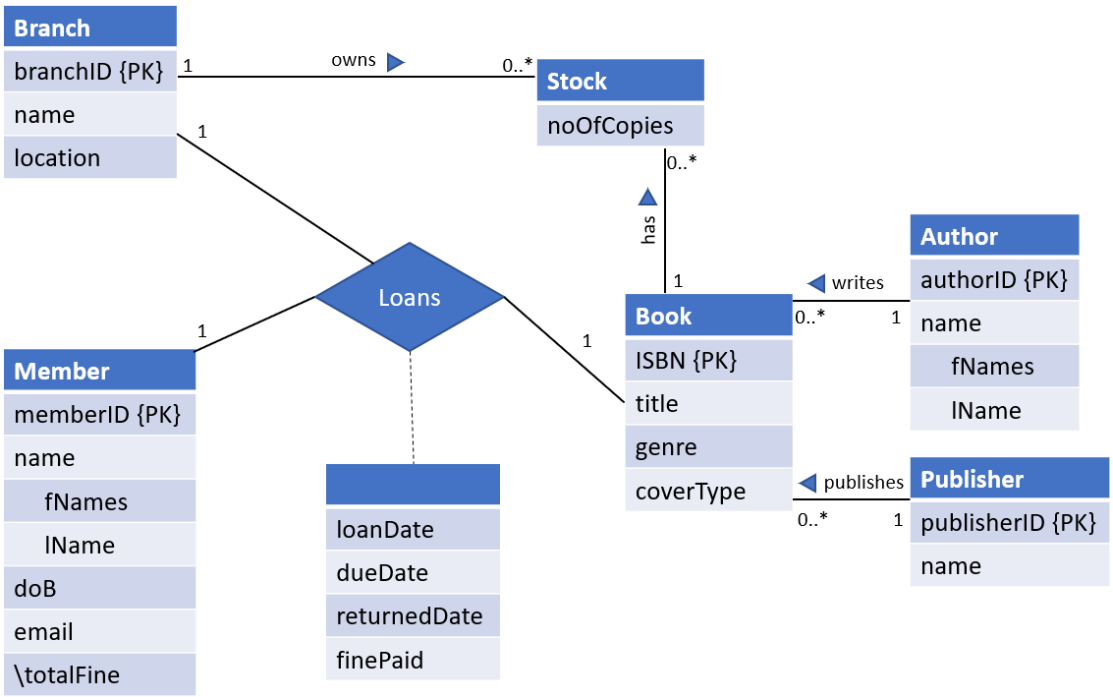
One assumption made about the fine is that it is £0.20 per day the book is overdue. Another assumption made is that, once the book has ben returned, the fine is **fixed** and will **not** continue to

increase until the fine is paid. A final assumption is that the member **can** pay a fine on an overdue book before they return it, however the fine will continue to increase until it is returned.

The data stored about the loan is the member who loaned the book via their member ID, the book being loaned via its ISBN, the branch that the book is being loaned from via its branch ID, the date the loan occurs, the date it is due back (4 weeks' time), the actual date the book is returned, and, if a fine has been accumulated, the amount that the member has repaid.

When a member returns a book, if they currently have more than one copy of that book out on loan, an assumption made is that the book returned is the copy that was loaned out first.

Entity Relationship Diagram



In this design I have acted upon the assumptions made in the user requirements as well as implementing newer assumptions to ensure that the designed database is in the third normal form.

The member ID was chosen as an identifying key as no other attribute would be unique. The name of a member would not be unique, even when combined with their DoB. The members email address could be a primary key; however, it was not chosen as a user's email could change at some point which could potentially require many updates to the database.
The member entity contains an additional derived attribute **totalFine** which is an attribute that is only calculated through queries and not stored in the database. To update this, the relevant finePaid attribute(s) in the **loans** entity would have to be updated.

As previously mentioned, a book is written by a single author and published by a single publisher. However, an author can write multiple books and a publisher publish many books. This resulted in

authors and publishers being stored as separate entities, having one-to-many relationships with the book entity.
The ISBN was chosen to uniquely identify a book because it is already used in many library systems to do just that. Furthermore, the title of the book is not a uniquely identifying field as books can and do have the same title. A combination of the author ID and book title could have been used as a unique identifier, however this becomes a more complicated method compared to using the ISBN of the book.
The publisher and author both have optional participation in the **publishes** and **writes** relationships respectively as an author and publisher does not need to be linked in the database to a book for it to exist. An example of this situation is if a book was deleted, leaving the author and publisher records in their respective tables.
Their primary keys are generated IDs for similar reasons that their other fields cannot be used to guarantee a unique identification.

A decision was made to not store a table containing each individual copy of a book in a branch and instead store the number of copies of a book at a branch. This was made with the cardinality of the table in mind. If each copy of a book was its own tuple, then the cardinality of that table would increase significantly with each new copy added to a branch. In the stock table, each set of copies is a single tuple, allowing for the cardinality of this table to be reduced greatly.
For example, if 4 branches each contained 4 copies of a book, in the chosen design the stock table would contain 4 tuples. However, in the rejected design, the table would have 16 tuples.

The stock entity manages the relation between the book entity and the branch entity as the initial relationship between those two would have been many-to-many. This design allows for a single branch to own many stock of a single book.
The **owns** relationship between a branch and stock is one-to-many with branch having optional participation and stock having mandatory participation because stock must belong to a single branch, whereas a branch can have many stock or none.
This is a similar situation for the **has** relationship between a book and stock as stock must belong to a single book whereas a book can have many stock or none.

The **loans** relationship states that a single member can loan a single book from a single branch. If a member wishes to loan out multiple books, then they must loan each book out individually.
The primary key of this table is a composition of the member ID, book ISBN and the date that the book is loaned out on. This means that, as a further assumption, a member cannot loan out two copies of the same book on the same day to ensure that those three fields can uniquely identify a loan.

Relational Diagram

Member (memberID, fNames, lName, doB, email)
Primary Key: memberID
Candidate Key: email

Branch (branchID, name, location)
Primary Key: branchID

Author (authorID, fNames, lName)
Primary Key: authorID

Publisher (publisherID, name)
Primary Key: publisherID

Book (ISBN, title, genre, coverType, author, publisher)
Primary Key: ISBN
Foreign Key: author references Author(authorID)
Foreign Key: publisher references Publisher(publisherID)

Stock (ISBN, branchID, noOfCopies)
Primary Key: ISBN, branchID
Foreign Key: ISBN references Book(ISBN)
Foreign Key: branchID references Branch(branchID)

Loans (memberID, ISBN, loanDate, dueDate, returnedDate, finePaid, branchID)
Primary Key: memberID, ISBN, loanDate
Foreign Key: memberID references Member(memberID)
Foreign Key: ISBN references Book(ISBN)
Foreign Key: branchID references Branch(branchID)