CHRIS HUI
DAVID NITU
EVELINA SARGSYAN
GEORGE SMAU

DATABASE SYSTEMS – ECS740P
COURSEWORK 1

GROUP 5

12th NOVEMBER 2021

Table of Contents

1. INTRODUCTION

The aim of this coursework is to design and implement a college library database system application. This report will include the design of the conceptual model for the database by describing and depicting the entity and attribute types of the system. To achieve this, the following steps will be outlined:

- Listing the assumptions made before the conceptual database design.
- Creating a conceptual model of the database in the form of an EER diagram.
- Deriving a relational schema from the aforementioned EER diagram.
- Normalising of the data up to 3rd Normal Form.

The full specification of the college library database is located in the appendix.

2. ASSUMPTIONS

Before designing this database system, there are some important aspects that must be taken into consideration. Outlined below is a list of assumptions that will help contextualise our design.

2.1 Member:

1. An individual is considered a member of the library system only if they have a library card. Members can have only one library card.
2. Each member can be specialised into either a student, staff or both wich is generalized into a teaching assistant subclass.
3. A member's account privileges are determined by the account type, which can initially be: student, staff or teaching assistant. The account state will be used to determine how many resources a member can borrow at any given time.
4. Students may borrow up to 5 resources at any given time whilst staff members may borrow up to 10. Members who are both students and staff members (teaching assistants) have the same privileges as staff members, meaning they can to borrow 10 resources at any given time.
5. Each member may have one or more contact addresses of one or more types (home, work, term-time, temporary).
6. Members who are no longer part of the library will have an account state of 'closed'., with data associated with the account still stored in the database.
7. Each member may have only one email address.

8. Each member may have zero or more phone numbers (Personal phone number, landline number, work phone number, etc.)

2.2. Loans:

9. If a loan is not returned in time, it will be specialized into an overdue loan. A trigger will be used to automatically classify loans as overdue based on their start date and the loan period of a resource. The LoanStartDate attribute stored both the date and the time a loan is made.
10. Charges are not stored as they can change daily. A total charge is stored once an overdue loan is resolved to keep track of payments made by the members.
11. If overall charges go over $10 the member is suspended and cannot longer loan new resources. In this situation, the account type will be changed to ‘suspended’.
12. A member can return resources without paying their fines, but they cannot pay their fines unless all their overdue resources are returned.
13. A member must pay all fines in full when attempting to make a payment.
14. In this initial database design, members cannot reserve resources.
15. Members cannot extend loan duration. To book the same resource again, the member needs to return it and then loan it again.

2.3 Resources:

16. There are three types of loan duration for each resource, this is determined by the system administrator when adding a new resource to the system. The three possible loan duration values are, 2 weeks, 2 days and 0 days (resource is for library use only and therefore, not loanable)
17. Members of the library that have any overdue loans can return the resources, but can’t loan any other resource until all their fines have been settled.
18. Resources are organised in the library by a combination of Floor number, Room number, Class number, Row number and shelf.
19. There are two types of resources: books and media. Media contains CDs, DVDs and videos.
20. Resource locations within the library are specified in the ShelfLocation table such that every unique resouce is located using a single genre specific class number.
21. Subjects are a more specific descriptor within a genre, e.g The genre of a book titled ‘An introduction to microbiology’ would be ‘Biology’, whilst the subject it belongs to would be ‘Microbiology’
22. The library has one or many copies of books / DVDs / CDs, these are physical copies stored in the shelves within the library.

23. Each book/media has a unique ISBN or ISSN, a title, edition, publisher, and publication year. Each book/media can have multiple authors.
24. Each copy has a unique copy number, and a shelf number to identify its location in the library shelves.

3. EER DIAGRAM

With these assumptions and the coursework specification in mind, the enhanced entity relationship (EER) model was used as the conceptual data model. It is a high-level model that is DBMS / implementation agnostic and describes the key entity types, attribute types and relationship types.

Considering the specification requirements and the list of assumptions in the preceding section, the EER diagram could be divided into five main categories:

3.1 Member

The Member entity type represents the library account of each valid library member of the college library system, with unique attribute types such as unique MemberId and MemberName. The MemberName is a composite entity type, consisting of FirstName and LastName. Both Staff and Student inherit attribute types from Member. Staff members and student members contain attribute types that are specific to their respective subclasses (e.g. JobRole/YearOfStudy) As a library member can be both Staff and Student (e.g. for the case of a teaching assistant), a TeachingAssitant is a subclass that is union of both Student and Staff. There is an aggregation relationship between Member and Address, a member can have multiple associated addresses (e.g. term-time address, home address, work address)

3.2 Loan

The Loan entity type keeps a record of the details concerning resource loans by library members, including the loan start date and return date. The OverdueLoan is a subclass of Loan and maintains a record of all overdue loans including accumulating fines.

3.3 Resource

The library possesses books, videos, DVDs, and CDs that can be borrowed by its members, or to be used within the library. The DVDs / CDs / videos can be combined into a single entity type (Media) as there are not enough distinctly separate attribute types between them in the context of a college library database, to justify having separate entity types.

The Resource entity type is a superclass disjoint specialised in Book and Media subclasses. The Resource superclass stores attribute types that are present in both Book and Media entity types (e.g. Title, Edition, Language), while Book and Media stores attribute types specific to their entity types (e.g. NumberOfPages for Book, Runtime for Media). Attribute type SerialNumber is stored in Resource, while ISBN / ISSN is stored in Book and Media respectively. These are differences in naming conventions and represent the same attribute. As library resources can have multiple copies, the UniqueCopy entity type keeps a record of each unique copy of each resource, and stores ISBN/ISSN number, ShelfId, and duration of the loan.

3.4 Library

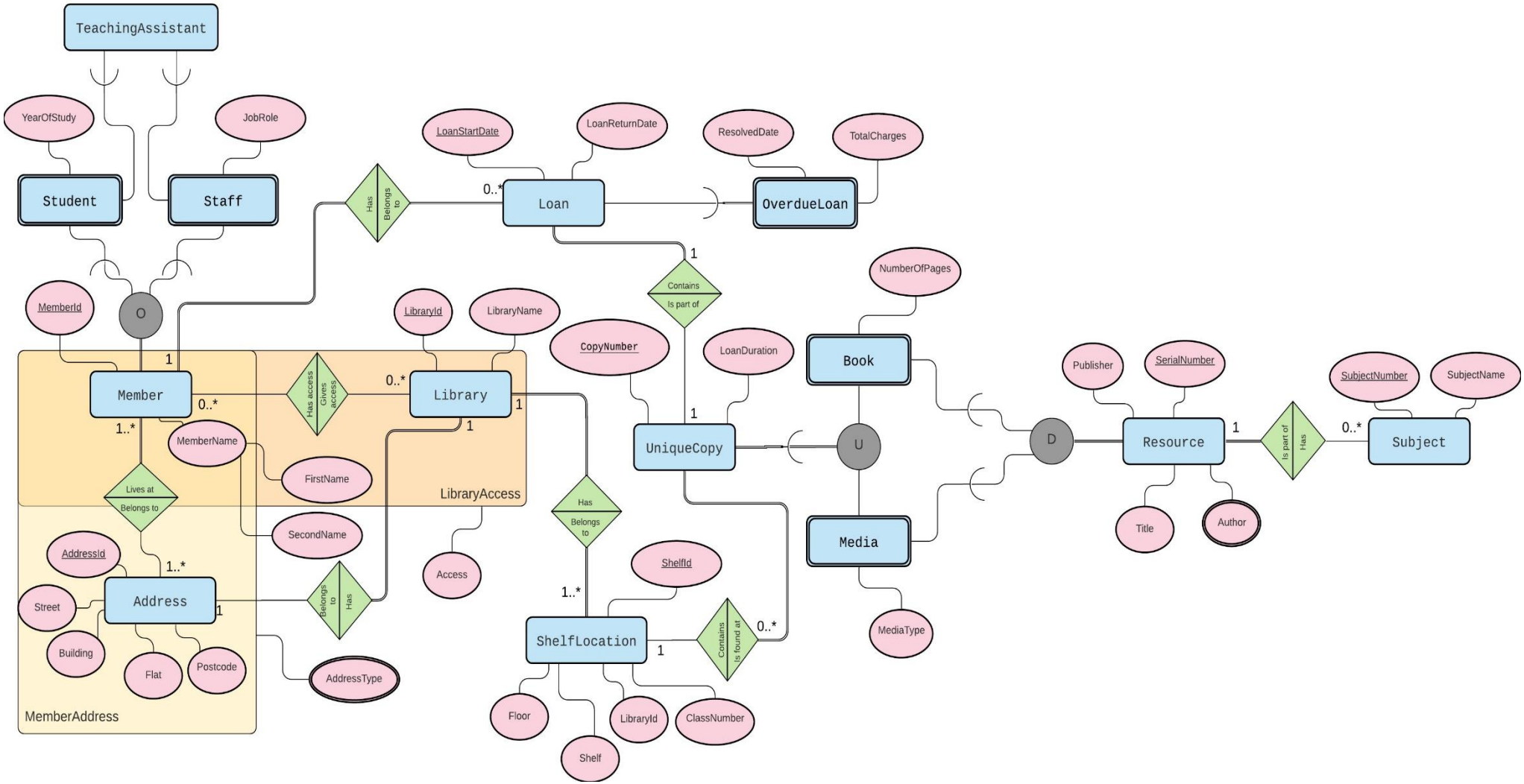
The ShelfLocation entity type stores specific shelf location as well as classification number. A classification number is used to classify library resources by subject area, it also shows the place of the item on the shelves in relation to other subjects. It is therefore appropriate to record this in the shelf location entity type.

3.5 Address

The Address entity type stores address records for both library members and library building(s). The attribute types Street, Building, Flat, Postcode is stored within this entity type.

3.6 Aggregates

Within this database model there are two relationships that are represented via aggregation, the MemberAddress and LibraryAccess. Aggregation is used as an abstraction to represent these relationships as higher level entity sets.


TeachingAssistant
YearOfStudy
JobRole
Student
Staff
MemberId
O
LoanStartDate
LoanReturnDate
ResolvedDate
TotalCharges
Has
Belongs to
0..*
Loan
OverdueLoan
1
Contains
Is part of
NumberOfPages
LibraryId
LibraryName
CopyNumber
LoanDuration
Book
Publisher
SerialNumber
SubjectNumber
SubjectName
1
Member
0..*
Has access
Gives access
0..*
Library
1
1..*
MemberName
FirstName
LibraryAccess
Lives at
Belongs to
SecondName
Access
Has
Belongs to
UniqueCopy
1
U
D
Resource
1
Is part of
Has
0..*
Subject
AddressId
1..*
Address
Street
Building
Flat
Postcode
1
Belongs to
Has
AddressType
MemberAddress
ShelfId
1..*
ShelfLocation
1
Contains
Is found at
0..*
Floor
Shelf
LibraryId
ClassNumber
Media
MediaType
Title
Author

4. RELATIONAL DATABASE SCHEMA

The following section covers the relational model derived from the EER diagram. The primary keys are under-lined and foreign keys are in italics.

4.1 Library Member

Member (MemberId, FirstName, LastName, Dob, Email, *AccountStateId, AccountTypeId*)

The Member entity type has a primary key of MemberId. As previously mentioned in assumptions, each valid member of the library will hold library cards, and will be given a unique MemberId identifying them as valid members of the library. Therefore, MemberId will serve as an unique identifier for each member account of the library. The remaining attributes serve as contact information for each member.

In addition, as per the specification, it is required to keep track of the member type (i.e. student / staff / teaching assistant) of each account, as well as the status of each account (e.g. member is suspended as fines owed by a member amount to more than $10) AccountState and AccountType were intially stored as attribute types of the Member entity type. However, if these are stored directly as string fields, there would be a possibility of inconsistencies in data entry. In an effort to improve data quality, separate tables (look-up tables) were implemented. These contain a list of valid values, and are related to the Member entity type. The look-up tables are as follows.

AccountType (AccountTypeId, TypeValue)
AccountState (AccountStateID, StateValue)

The columns can be set to one of a range of valid values. For AccountType it would be StudentAccount, StaffAccount, or TeachingAssistantAccount. For AccountState it would be Active, Suspended, or Closed.

It is stated in the specification that students can loan a maximum of 5 resource items, and staff a maximum of 10 resource items. This is not directly stored in the table as an attribute type, as it can be derived from the AccountType lookup table. In addition, the AccountType lookup table satisfies the requirement to keep track of ‘student and staff members of the library’.

A trigger would be in place to change the AccountState from ‘Active’ to ‘Suspended’, in the case that the amount owed in fines by a member is more than 10 dollars. Similarly, the AccountState would be changed from ‘Suspended’ to ‘Active', when the member returns loaned resources and pays fines in full. This satisfies the requirement to maintain a record of ‘a list of library members who have been suspended due to overdue loans or unpaid fines.’

Student (*PersonId*, Programme, EducationLevel, YearOfStudy)
Staff (*PersonId*, Department, JobRole, Salary, TA)

The Staff and Student entity types are subclasses to the Member superclass, and do not have a key attribute type of their own. Teaching assistants can be considered both students and staff, thus we have the attribute type TA as a flag for teaching assistants. This is stored in Staff as there are typically fewer staff than students.

PhoneNumber (*MemberId*, CountryCode, Number, PhoneType)

In addition to the email attribute type stored in the Member entity type, the PhoneNumber entity type stores phone number information and has a one to zero-or-many relationship with the Member entity type; as each library member could have zero or multiple contact phone numbers. The MemberId attribute type is a foreign key, with all four of MemberId, CoutnryCode, Number, and PhoneType forming a composite primary key.

LibraryAccess (*AccountId, LibraryId*, Access)

LibraryAccess is an aggregation added to future-proof the database in the case that there will be changes in the library, or more than 1 library in the future. A member could potentially have access to 0 or more libraries. If a library is removed, any member that had access to the library will remain in the database. AccountId and LibraryId are foreign keys linking to Member and Library entity types, and together forming the primary key for the LibraryAccess entity type.

MemberAddress (*PersonId, AddressId, AddressType*)

Similar to LibraryAccess, MemberAddress is an aggregation that links the LibraryMember entity types with the Address entity type. The foreign keys PersonId and AddressId form a composite primary key. The AddressType attribute type stores the type of address of each member e.g. term-time address, home address, work address.

4.2 Loans

Loan (*MemberId*, *CopyNumber*, LoanStartDate, LoanReturnDate)

The Loan entity type has MemberId, CopyNumber, and LoanStartDate as the composite primary key. The foreign key MemberId provides a linkage to the library member making a library loan. While the foreign key CopyNumber provides a linkage to the unique library resource that is being loaned. The LoanStartDate primary key stores information on the start date of the loan.

It should be noted that due date and overdue date are not directly stored in this entity type, as the due date could be derived from the LoanStartDate + LoanDuration (stored in UniqueCopy entity type), and Overdue could be triggered once the current date is past the derived due date. The LoanReturnDate attribute type is set to null by default, indicating that the loan has not been returned. Once the member returns the resource, LoanReturnDate will be populated thus indicating that the loan has been returned. Popular library resources could be derived from the occurrences of a specific SerialNumber, linking to the ISBN or ISSN primary keys of the Book or Media entity, via the UniqueCopy entity type.

This entity type satisfies the requirement to maintain a record of 'all current loans, including whether they are overdue', and 'previous loans to help in identifying popular resources'.

OverdueLoan (*MemberId*, *CopyNumber*, *SerialNumber*, *LoanStartDate*, FineResolvedDate, TotalFine)

The OverdueLoan entity type has been added to avoid null values in FineResolvedDate and TotalFine, if these attribute types were stored directly in the Loan entity type. The entries populated in this table will be strictly library loans that are past due. A trigger would be in place to monitor the current date against the derived due date of a particular resource. Once a loan is past due, a trigger is used to populate the TotalFine attribute, and is incremented by 1 dollar per day. FineResolvedDate is set to null by default indicating that the fine has not been paid, and stores the date that the library member pays the fine.

4.3 Library, location and address

Library (LibraryId, LibraryName, *AddressId*)

The Library entity type is added to future-proof the database in the case that there will be changes to the library address, or more than 1 library in the future. The Primary key LibraryId is generated incrementally when/if a library is added. The AddressId foreign key links the library to a specific address.

Address (AddressId, Street, Building, Flat, Postcode)

The Address entity type has a primary key of AddressId, which is incrementally generated for each address added to the database. No other attribute types uniquely identify the attribute types of this tuple. In the case of Postcode, it is possible that multiple addresses occupy the same postcode (e.g. apartments, student dormitory), thus not appropriate as a candidate key for this entity type.

ShelfLocation (ShelfId, *LibraryId*, Floor, Room, ClassNumber, Row, Shelf)

The ShelfLocation entity type has a composite primary key of ShelfId and LibraryId. The ShelfId attribute type is an integer incrementally generated for each shelf location in the library. The LibraryId foreign key links the location of the shelf to a specific library. As previously mentioned in the Entity-Relationship Diagram section, the ClassNumber attribute type classify library resources by subject area, and shows the place of the item on the shelves in relation to other subjects

4.4 Resource

UniqueCopy (CopyNumber, LoanDuration, *SerialNumber, ShelfId*)

The UniqueCopy entity type has CopyNumber and SerialNumber as a composite primary key that uniquely identifies each copy of a resource in the library. CopyNumber is an integer generated incrementally for each unique book / media item in the library, as there could be multiple copies of the same book / media within the library. The number of copies of a particular resource could be derived by grouping the SerialNumber attribute type.

It should be noted that the availability of a particular copy of a resource is not stored as an attribute type in the UniqueCopy entity type, as this could be derived from the Loan table. If the UniqueCopy primary key is present in the Loan table, with a null ReturnDate attribute, this would indicate that the item has been loaned by another library member, and is not available for loan for other library members.

As stated in the specification, each resource could have different loan periods, e.g. 2 weeks, 2 days, or can only be used within the library. The loan duration associated with each item is stored in the LoanDuration attribute type. The foreign key ShelfId links to a specific shelf location in the library. This satisfies the requirement to keep track of ‘what each resource is, its class number, how many copies of it are held by the library, and where these are located in the library.’

Book (ISBN, BTitle, BEdition, Publisher, BLanguage, PublicationYear, NumberOfPages, *SubjectNumber*)
Media (ISSN, MTitle, MEdition, PublisherId, MLanguage, MediaType, Runtime, ReleaseDate, *SubjectNumber*)

The Book and Media entity types have the primary key of ISBN and ISSN respectively. No other attributes uniquely identify these entity types. SubjectNumber is a foreign key that links these entity types to a specific Subject.

CreatedBy (*SerialNumber, AuthorId*)
Author (AuthorId, AuthorFirstName, AuthorLastName)

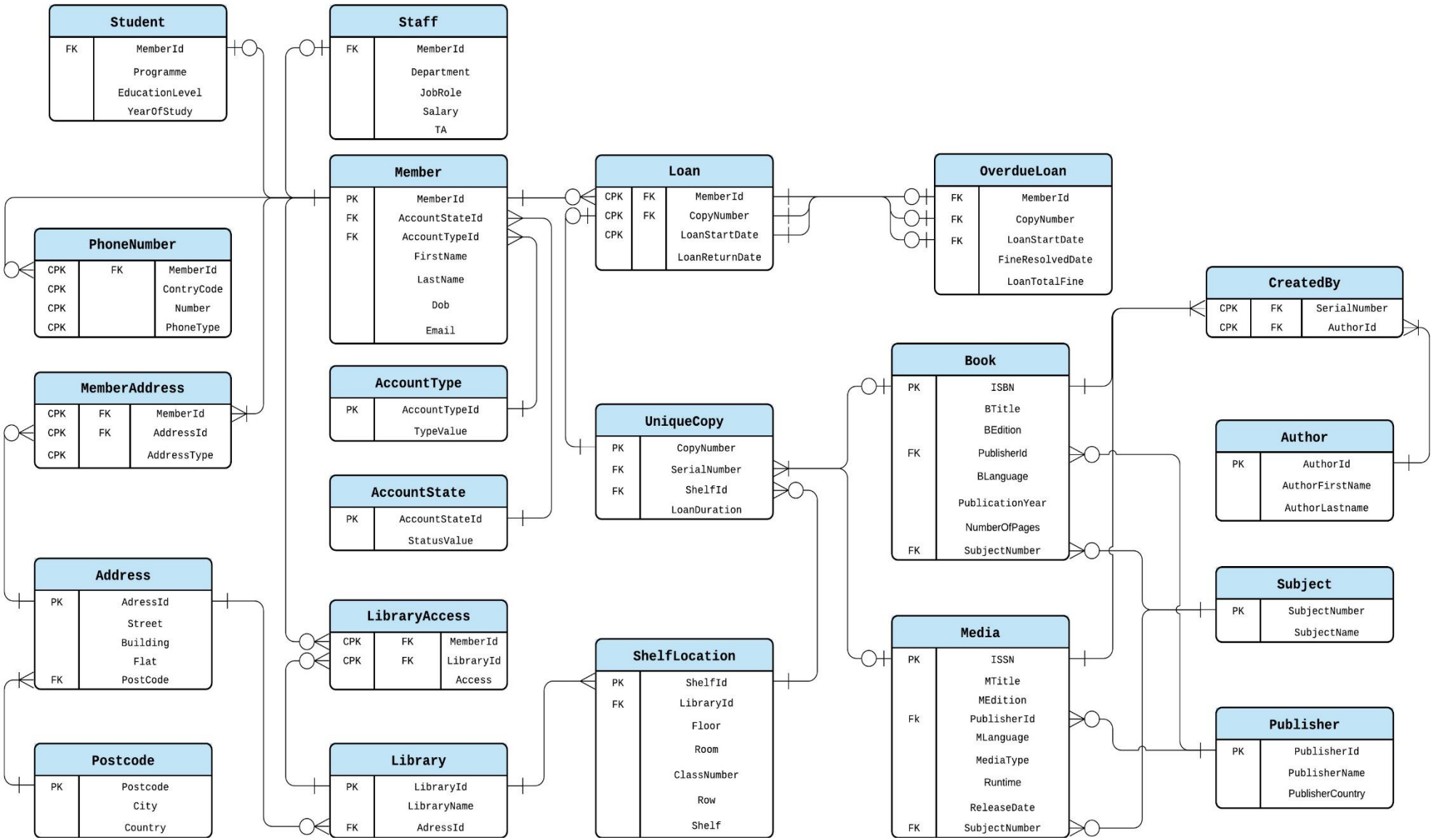
The Author entity type has a primary key of AuthorId, which is incrementally generated for each author in the database. ResourceAuthor is an associative entity type for BookDetail / MediaDetail with the Author entity type. We implemented this relation due to the consideration that a resource (i.e. book or media) could have multiple authors, and an author could have authored multiple resources. Therefore BookDetail and MediDetail have a one-to-many relationship with ResourceAuthor, and ResourceAuthor have a many-to-one relationship with Author.

Publisher (PublisherId, PublisherName, Country)

Publisher entity type has a primary key PublisherId which is also foreign key in Book and Media. The relationship of this table with Book/Media is one-to-many.

Subject (SubjectNumber,SubjectNumber)

The primary key for the Subject entity type is SubjectNumber. SubjectNumber can be the same for many resources as it reflects the subject of a particular resource. Therefore, it has a one to zero-or-many relationship to Book and Media entity types.


Student
FK MemberId
Programme
EducationLevel
YearOfStudy
Staff
FK MemberId
Department
JobRole
Salary
TA
Member
PK MemberId
FK AccountStateId
FK AccountTypeId
FirstName
LastName
Dob
Email
Loan
CPK FK MemberId
CPK FK CopyNumber
CPK LoanStartDate
LoanReturnDate
OverdueLoan
FK MemberId
FK CopyNumber
FK LoanStartDate
FineResolvedDate
LoanTotalFine
PhoneNumber
CPK FK MemberId
CPK ContryCode
CPK Number
CPK PhoneType
MemberAddress
CPK FK MemberId
CPK FK AddressId
CPK AddressType
AccountType
PK AccountTypeId
TypeValue
AccountState
PK AccountStateId
StatusValue
UniqueCopy
PK CopyNumber
FK SerialNumber
FK ShelfId
LoanDuration
Book
PK ISBN
BTitle
BEdition
FK PublisherId
BLanguage
PublicationYear
NumberOfPages
FK SubjectNumber
CreatedBy
CPK FK SerialNumber
CPK FK AuthorId
Author
PK AuthorId
AuthorFirstName
AuthorLastname
Subject
PK SubjectNumber
SubjectName
Address
PK AdressId
Street
Building
Flat
FK PostCode
LibraryAccess
CPK FK MemberId
CPK FK LibraryId
Access
ShelfLocation
PK ShelfId
FK LibraryId
Floor
Room
ClassNumber
Row
Shelf
Media
PK ISSN
MTitle
MEdition
Fk PublisherId
MLanguage
MediaType
Runtime
ReleaseDate
FK SubjectNumber
Publisher
PK PublisherId
PublisherName
PublisherCountry
Postcode
PK Postcode
City
Country
Library
PK LibraryId
LibraryName
FK AdressId

5. NORMALISATION

5.1 The normalisation process

The universal relation assumption states that one can store all the elements of a database within a single table. This design is inherently flawed because it can easily lead to data redundancy in the form of insertion and deletion errors. However, it is a good place to start normalising the system, as all elements are displayed in relation to each other.

5.2 Universal Relation

The universal relation assumption states that one can store all the elements of a database within a single table. This design is inherently flawed because it can easily lead to data redundancy in the form of insertion and deletion errors. However, it is a good place to start normalising the system as you can see all elements in relation to each other.

Using the selected attributes, the following universal relation for the system was built.

U(MemberId, MemberName, Dob, PhoneNumber, Email, MemberAddress, AccountType, AccountState, Department, JobRole, Salary, TA, Programme, EducationLevel, YearOf Study, ISBN, BTitle, BAuthor, BEdition, BLanguage, BPublisher, BPublicationYear, NumberOfPages, BPublisher, ISSN, MTitle, MAuthor, MEdition, MLanguage, MPublisher, MediaType, RunTime, ReleaseDate, CopyNumber, LoanDuration, ClassNumber, ClassName, ShelfLocation, LibraryId, LibraryName, LibraryAddress, Access)

5.3 Functional dependencies

An additional step starting the normalisation process involves identifying the functional dependencies within the universal relation. A functional dependency between two attribute types, a prime (primary key) and a non-prime attribute type, implies that for every given instance of the prime attribute type, the value of the prime attribute type uniquely determines the value of the non-prime attribute type and can be written as follows: **Prime-attribute → Non-prime-attribute**.

In the universal relation presented above, the following functional dependencies can be identified:

1. The **MemberId** attribute type can determine the attribute types describing a member instance of the database.

   MemberId → {MemberName, Dob, Email, PhoneNumber, MemberAddress, AccountType, AccountState, Department, JobRole, Salary, TA, Programme, EducationLevel, YearOfStudy}

2. The **ISBN** and the **ISSN** attribute types each determine the attribute types for the Book and Media entity types respectively.

   ISBN → {BTitle, BAuthor, BEdition, BLanguage, BPublisher, BPublicationYear, NumberOfPages, Bpublisher}

   ISSN → {MTitle, MAuthor, MEdition, Language, MPublisher, MediaType, RunTime, ReleaseDate}

3. The **CopyNumber** can determine the **LoanDuration** attribute type. For this relationship the **SerialNumber** attribute type is also used as a foreign key. This is just a naming convention, and it refers to either **ISBN** or **ISSN**.

CopyNumber → {LoanDuration, SerialNumber}

4. The **LibraryId** can determine the **LibraryName, LibraryAddress** and **ShelfLocation** attribute type.

LibraryId → {LibraryName, LibraryAddress, ShelfLocation}

5. The **LibraryId** and the **MemberId** can determine the **LibraryAccess** attribute type.

LibraryId, MemberId → {Access}

6. The **MemberId**, **CopyNumber** and **LoanStartDate** attribute types can determine the attribute types describing a loan instance of the database.

MemberId, CopyNumber, LoanStartDate → {LoanReturnDate, LoanTotalFine, FineResolvedDate}

5.4 First form normalisation (1NF)

For a table to be in first normal form it must satisfy the following conditions:

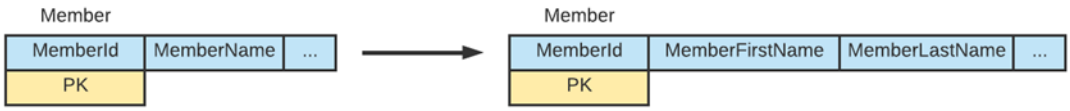
1. The table columns and rows must not require being inserted in a specific order for the data to make sense.
2. There are no duplicate column names or row entries.
3. Each row-column intersection must contain atomic values. In other words, the table must not contain any multivalued or composite attribute values.

Within the identified functional dependency relationships, all columns store unique names, but not all store atomic values. The following attribute types have been identified as being either multivalued or composite:

- **MemberName** (composite attribute)
- **PhoneNumber** (multivalued attribute)
- **MemberAddress** (multivalued attribute)
- **LibraryAddress** (composite attribute)
- **ShelfLocation** (multivalued attribute)
- **BAuthor** and **MAuthor** (multivalued attribute)
- **BPublisher** and **MPublisher** (composite attribute)

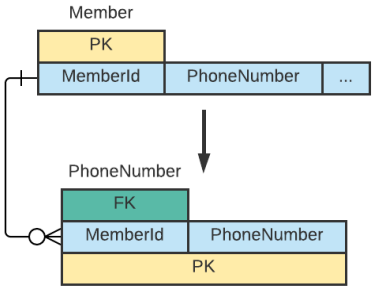
5.4.1 MemberName

The MemberName attribute type is a composition between MemberFirstName and MemberLastName. A composite attribute type can be normalised by adding a new column for each element it contains. Therefore, the table will be undergoing the following changes: (MemberId, MemberName, …) transforms into (MemberId, MemberFirstName, MemberLastName, …).

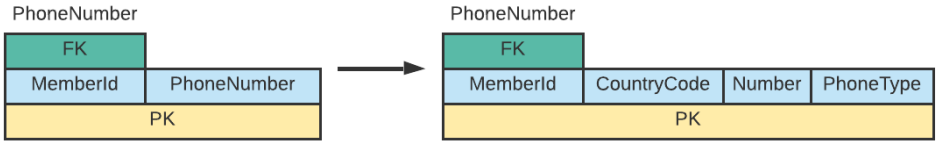


5.4.2 PhoneNumber

The PhoneNumber is a multivalued attribute type as it can store multiple phone numbers for each member. There are two ways to deal with multivalued attributes in the normalisation process. Either multiple tuples are stored within the same table, or a new table is created that uses a composite primary key formed by the primary key of the owner entity and one or more attribute type belonging to the newly formed table. The latter is recommended to avoid data redundancy, so for the normalization of this database design the second approach was used (unless otherwise stated). Therefore, a new table (PhoneNumber) is created.



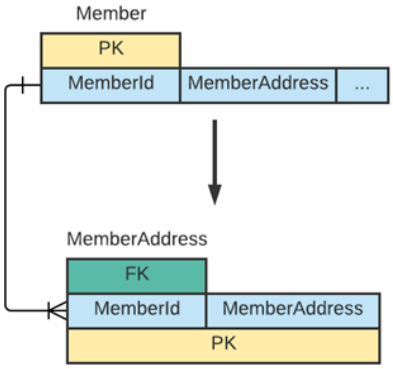
In the PhoneNumber table the MemberId functions as a foreign key but at the same time a primary key in composition with the PhoneNumber attribute type. However, this table still doesn't satisfy 1NF because PhoneNumber is a composite attribute type. The table is therefore transformed once more as follows: (MemberId, PhoneNumber) transforms into (MemberId, CountryCode, Number, PhoneType). This results in the table having a composite primary key formed by the attributes MemberId, CountryCode, Number and PhoneType.



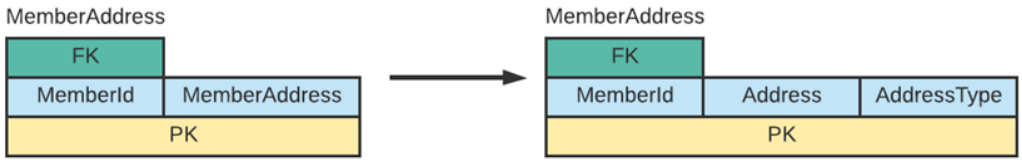
5.4.3 MemberAddress

MemberAddress is a multivalued attribute type as it can store multiple addresses for each member. The design is normalized by creating a new table (MemberAddress) to store the Member Address attribute type.

In the MemberAddress table the MemberId functions as both a foreign key and a primary key in composition with the MemberAddress attribute type. However, this table is still not in 1NF because MemberAddress is a composite attribute type. The table is therefore transformed once more as follows: (MemberId, MemberAddresses) transforms into (MemberId, Address, AddressType). This results in the table having a composite primary key formed by the attributes MemberId, Address and AddressType.



The MemberAddress table is still not in 1NF because as previously mentioned, a composite attribute type is still present (Address) The composite attribute type will be split into different columns for each of its elements. For the purpouse of simplicity, the Addres attribute type is assume to be split, bit it will be used under a single attribute type to make the MemberAddress table simpler to work with. This issue will be addressed in the second normal form because it involves partial dependency between address and its subcategories.



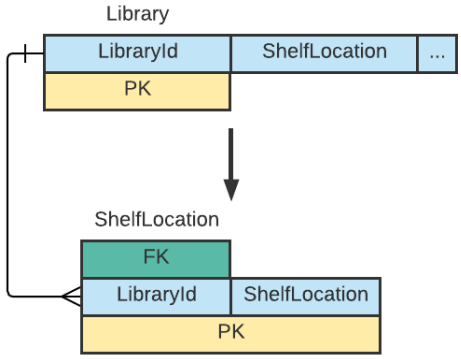
5.4.4 LibraryAddress

The LibraryAddress attribute is a composite attribute like the Address attribute within the MemberAddress table and it is composed of the same fields. Similarly, the LibraryAddress attribute type is assumned to be split into single

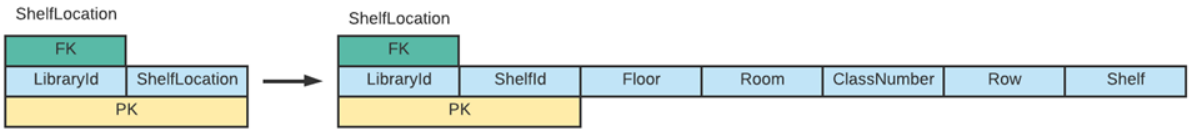
elements, but will be used under a single attribute type for simplicity. This issue will also be addressed in the third normal form as because it involves transitive dependency between address and its subcategories.

5.4.5 ShelfLocation

The ShelfLocation attribute type is multivalued because it describes the location where a member can find resources within the library. The library can have multiple locations for storing resources. Therefore, a new table (ShelfLocation) is created. The LibraryId attribute type and the ShelfLocation attribute types will serve as composite primary keys.

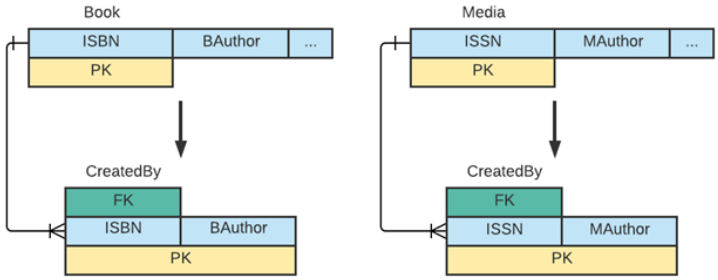


The ShelfLocation attribute type within the ShelfLocation table is composite and can split into the following attributes: ShelfId, Floor, Room, ClassNumber, Row, Shelf. The table will be transformed as follows: (LibraryName, ShelfLocation) transforms into (LibraryName, ShelfId, Floor, Room, ClassNumber, Row, Shelf). ShelfId is used in conjunction with LibraryId to avoid having a very big composite primary key. As different libraries might have different naming systems for the rooms and shelves, the table doesn't need to be simplified anymore since all rows will most likely be unique.

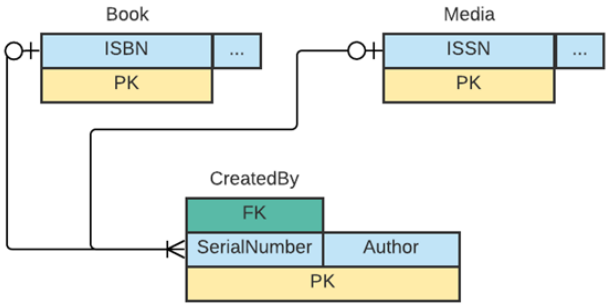


5.4.6 BAuthor and MAuthor

The BAuthor and MAuthor attribute type are multivalued, as it allows the storage of multiple Authors for each book or media. Therefore, a new table Created By was created with the ISBN/ISSN and BAuthor/MAuthor attribute types forming a composite primary key.



Since both books and media have authors and authors might even write books and create media at the same time, the CreatedBy table can store both book-author and media-author relationships as shown below. The SerialNumber is used conventionally to represent an option between either ISBN or ISSN. The specific BAuthor and MAuthor are also simplified to just Author as a naming convention.



Within the CreatedBy table, the Author attribute type is composite because it is formed by the author's first name and last name. The composite attribute type will be split into different columns for each of its elements. For the time being the Author attribute type is considered split but represented by the Author attribute type for simplicity. This issue will be addressed in the third normal form because it involves transitive dependency between author and its subcategories.

5.4.7 BPublisher and MPublisher

The BPublisher and MPublisher attribute types are composite and can be split into PubliserName and PublisherCountry. Similarly, to the Author attribute type, the BPublisher and MPublisher attribute types will be assumed to be split and will be dealt with during third normal form because it involves transitive dependency between publisher and its subcategories.

Following first form normalisation of the multivalued and composite attributes the following tables have been obtained:

- **Members table**: (MemberId, MemberFirstName, MemberLastName, Dob, Email, AccountType, AccountState, Department, JobRole, Salary, TA, Programme, EducationLevel, YearOfStudy)
- **PhoneNumber table**: (MemberId, CountryCode, Number, PhoneType)
- **MemberAddress table**: (MemberId, AddressId, AddressType)
- **Library table**: (LibraryId, LibraryName, LibraryAddress)
- **Book table**: (ISBN, BTitle, BEdition, BLanguage, BPublisher, BPublicationYear, NumberOfPages, Bpublisher)
- **Media table**: (ISSN, MTitle, MEdition, Language, MPublisher, MediaType, RunTime, ReleaseDate)
- **CreatedBy table**: (SerialNumber, Author)
- **ShelfLocation table**: (LibraryId, ShelfId, Floor, Room, ClassNumber, Row, Shelf)
- **UniqueCopy table**: (CopyNumber, SerialNumber, LoanDuration, ShelfId) – *the ShelfId and SerialNumber are also used in the UniqueCopy table as a foreign keys. ShelfId is used to identify the specific location of each unique copy within the library. This attribute type was not mentioned in the initial functional dependencies because it was only obtained after 1NF*.
- **LibraryAccess table**: (LibraryId, MemberId, Access)
- **Loan table**: (MemberId, CopyNumber, LoanStartDate, LoanReturnDate, LoanTotalFine, FineResolvedDate)

5.5 Second form normalisation (2NF)

For a table to be in second normal form it must satisfy the following conditions:

1. The table must satisfy first normal form
2. Every non-prime attribute type must be fully dependent on the primary key. In the situations where a composite primary key is present, the non-prime attribute type must be dependent on every attribute type forming the primary key.

To satisfy 2NF the functional dependency must be checked for every table. If there is no partial dependency within the table, the table will be in 2NF. These things considered, only tables with composite primary key can break 2NF.

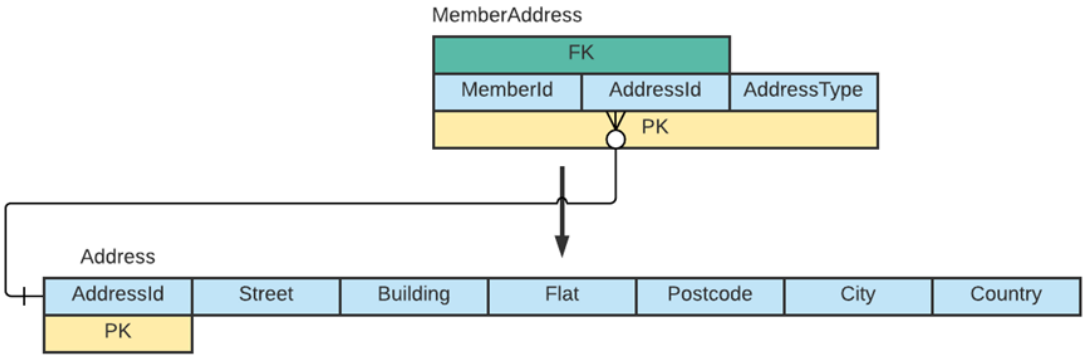
- Tables *Member, Library, Book, Media, UniqueCopy and ShelfLocation* all have single-valued primary keys, and therefore satisfy 2NF.
- Tables *PhoneNumber, MemberAddress, CreatedBym LibraryAccess* and *Loan* all have composite primary keys so they will be checked for partial dependency to determine if the satisfy 2NF.

5.5.1 PhoneNumber

The PhoneNumber table contains four attributes, all being part of the composite primary key. Therefore, there cannot be any partial dependency between any attribute types, so the table is in 2NF.

5.5.2 MemberAddress

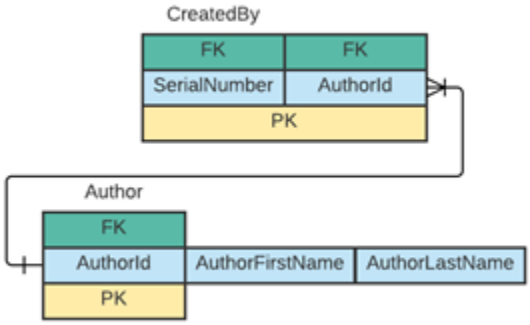
The MemberAddress table contains 3 attribute types: MemberId, Address and AddressType. The Address attribute type is composite and will be split in the following attribute types: AddressId, Street, Building, Flat, Postcode, City, Country. The Street, Building, Flat, Postcode, City and Country attribute types are dependent on the AddressId only, which replaces the Address attribute type in the primary key. Therefore, partial dependency can be identified, and a new table Address is created that will use the AddressId as a primary key and store the other attribute types dependent on AddressId.



5.5.3 CreatedBy

The CreatedBy table contains two attribute types, both being part of the composite primary key. The Author attribute type is composite and can be split into AuthorId, AuthorFirstName and AuthorLastName. AuthorId will replace the Author attribute type as part of the composite primary key. In this situation partial dependency can be identified since the AuthorFirstName and AuthorLastName are only dependent on the AuthorId. Therefore a new table

(Author) is created that uses the AuthorId as primary key and stores the AuthorFirstName and AuthorLastName attribute types.



5.5.4 LibraryAccess

The LibraryAccess table contains two prime attributes (MemberId, LibraryId) and a non-prime attribute (Access). The Access attribute type is functionally dependent on both prime attributes, therefore, there cannot be any partial dependency, so the table is in 2NF.

5.5.5 Loan

The Loan table contains a composite primary key formed by the MemberId, the CopyNumber and LoanStartDate attribute types. All the non-prime attribute types are functionally dependent on all three prime attribute types, therefore there is no partial dependency, so the table is in 2NF.

Following normalisation to the second normal form, the library database system is now composed of the following tables:

- **Member table**: (MemberId, MemberFirstName, MemberLastName, Dob, Email, AccountType, AccountState, Department, JobRole, Salary, TA, Programme, EducationLevel, YearOfStudy)
- **PhoneNumber table**: (MemberId, CountryCode, Number, PhoneType)
- **MemberAddress table**: (MemberId, AddressId, AddressType)
- **Address table**: (AddressId, Street, Building, Flat, Postcode, City, Country)
- **Library table**: (LibraryId, LibraryName, LibraryAddress)
- **Book table**: (ISBN, BTitle, BEdition, BLanguage, BPublisher, PublicationYear, NumberOfPages, SubjectId, SubjectName)
- **Media table**: (ISSN, MTitle, MEdition, Language, MPublisher, MediaType, RunTime, ReleaseDate, SubjectId, SubjectName)
- **CreatedBy table**: (SerialNumber, AuthorId)

- **Author table**: (AuthorId, AuthorFirstName, AuthorLastName)
- ShelfLocation table: (LibraryId, ShelfId, Floor, Room, ClassNumber, Row, Shelf)
- **UniqueCopy table**: (CopyNumber, SerialNumber, LoanDuration, ShelfId) – *the ShelfId and SerialNumber are also used in the UniqueCopy table as a foreign keys. ShelfId is used to identify the specific location of each unique copy within the library. This attribute type was not mentioned in the initial functional dependencies because it was only obtained after 1NF.*
- **LibraryAccess table**: (LibraryId, MemberId, Access)
- **Loan table**: (MemberId, CopyNumber, LoanStartDate, LoanReturnDate, LoanTotalFine, FineResolvedDate)

5.6 Third form normalisation

For a table to be in third normal form it must satisfy the following conditions:

1. The table must satisfy second normal form
2. There is no transitive dependency for non-prime attribute types, meaning that all non-prime attribute types must be functionally dependent only on the primary key and not on another non-prime attribute.

To determine whether a table satisfies 3NF, it must be checked for transitive dependency.

5.6.1 Member

The Member table contains multiple attribute types. All attribute types are functionally dependent on the primary key MemberId. Therefore, there is no transitive dependency between non-prime attribute types, so the table is in 3NF.

5.6.2 PhoneNumber

The PhoneNumber table contains 4 attributes, all composing the primary key. There are none non-prime attribute types in this table, so there can't be any transitive dependency. Therefore, the table is in 3NF.

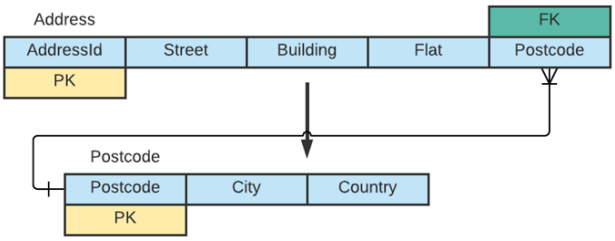
5.6.3 MemberAddress

The MemberAddress table contains 3 attributes, all composing the primary key. There are none non-prime attribute types in this table, so there can't be any transitive dependency. Therefore, the table is in 3NF.

5.6.4 Address

The Address table has the AddressId as the primary key. At the same time, it also has a City and Country attribute types that depend on the Postcode, a non-prime attribute type. City and Country is transitively dependent on AddressId via Postcode. Therefore, a new table (Postcode) is created using the Postcode attribute type as the primary

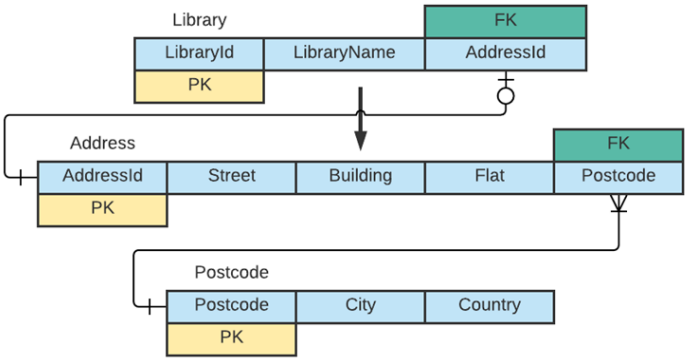
key. The City and Country attribute types are also stored in the Postcode table thus removing the transitive dependency from the Address table.



At this stage, there is no transitive dependency left in both the Address and the Postcode table, so both tables are in 3NF.

5.6.5 Library

The Library table contains the LibraryId, LibraryName and LibraryAddress attribute types. The LibraryAddress attribute type is composite and can be split into the primary key AddressId, and the following non-prime attribute types: Street, Building, Flat, Postcode, City, Country. All these non-prime attribute types are dependent on the prime attribute type AddressId. Therefore, they are transitively dependent on LibraryId via the AddressId attribute type. The Street, Building, Flat, Postcode, City and Country attribute types must be moved into a new table having AddressId as their primary key. Since such a table was already created (i.e. the Address table), the LibraryAddress elements can be stored in there, thus getting rid of the transitive dependency within the Library table. Furthermore, the Postcode attribute will be dealt with in the same way as for the Address table. The transformation is shown below.

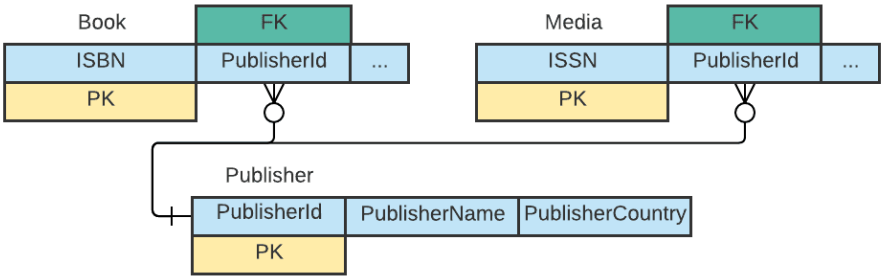


Following these transformations, there is no transitive dependency in the Library table anymore, 3NF is satisfied.

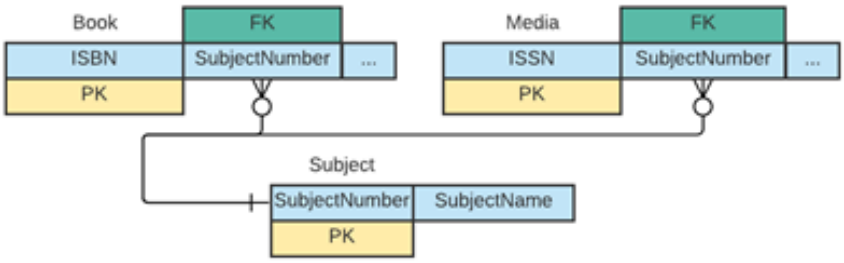
5.6.6 Book and Media

The Book and Media table contains a single transitive dependency each. The Publisher attribute type, being composite, can be split into PublisherName and PublisherCountry attribute types, which are not directly dependent

on primary keys of the Book or Media tables. To remove transitive dependency, a new table is created (Publisher), that can store both the PublisherName and the PublisherCountry attribute types for both tables since the attributes are the same. The new table uses a PublisherId primary key to link it to the Book and Media tables. The transformation is displayed below.



The same applies to the SubjectNumber and SubjectName attribute types. Since SubjectName is functionally dependent on SubjectNumber, and SubjectNumber is a non-prime attribute type, this is identified as transitive dependency on ISBN/ISSN via the SubjectNumber attribute type. A new table has to be created (Subject) that will use SubjectNumber as a primary key and store the SubjectName.



5.6.7 CreatedBy

The CreatedBy table contains 2 attributes, both composing the primary key. There are none non-prime attribute types in this table, so there can't be any transitive dependency. Therefore, the table is in 3NF.

5.6.8 Author

The Author table contains 3 attribute types, AuthorId, AuthorFirstName and AuthorLastName. Both AuthorFirstName and AuthorLastName are functionally dependent on AuthorId, the primary key. Therefore there is no transitive dependency so the table satisfies 3NF.

5.6.9 ShelfLocation

The ShelfLocation table has a single valued primary key, ShelfId. All the other non-prime attributes have been identified as functionally dependent on ShelfId, so there is no transitive dependency in this table. Therefore, the table must satisfy 3NF.

5.6.10 UniqueCopy

The UniqueCopy table also has a single valued primary key, CopyNumber and all the other non-prime attributes have been identified as functionally dependent on it. Therefore, there is no transitive dependency so the table must satisfy 3NF.

5.6.11 LibraryAccess

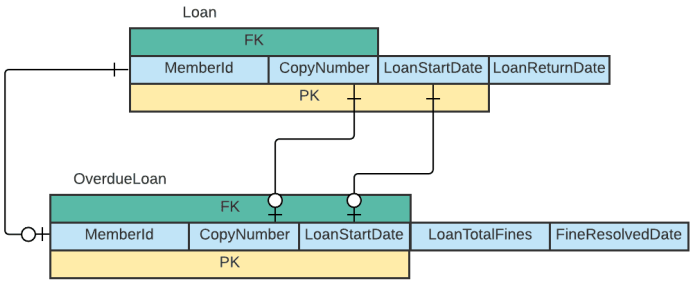
The Library access table has a composite primary key formed by the MemberId and LibraryId attribute types. The only other attribute type in this table, Access, is functionally dependent on both primary keys, so there is no transitive dependency. Therefore, the table must satisfy 3NF.

5.6.12 Loan

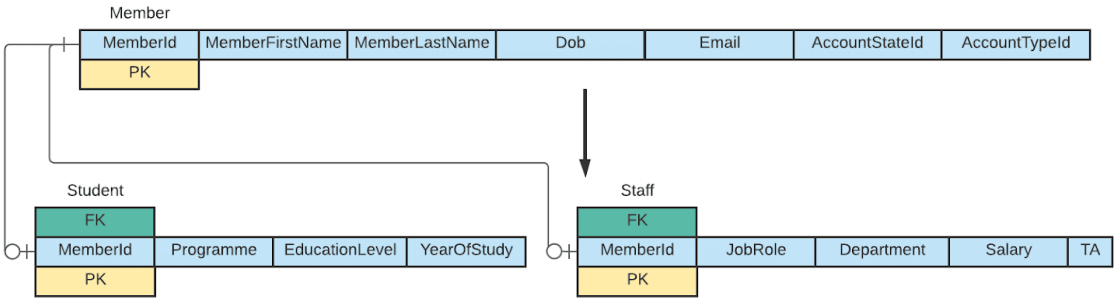
The Loan table has a composite primary key formed by the MemberId, CopyNumber and LoanStartDate prime attribute types. All the non-prime attribute types are fully functionally dependent on the composite primary key and therefore don't show any transitive dependency. This means that the table must be in 3NF.

5.6.13 Further improvements

As a data quality improvement exercise, the FineResolvedDate and LoanTotalFine attribute types (associated with past-due loans) are stored in a separate OverdueLoan table. This reduces the amount of 'null' values in Loan entities.



Similarly, attribute types associated with Student and Staff members will be stored in separate tables. This reduces the amount of 'null' values in the Member entities.



6. CONCLUSION

This library database design is rather simplistic allowing basic functionality for tracking Members, Loans and Resources within a library. Further functionality could be added such as reserving resources and extending current loans.

In reviewing the normalised entity types presented in section 5, it can be observed that the entity types in the relational database schema are normalised to 3NF. As such, the database is ready for implementation.

7. APPENDIX

Database Application requirements - College Library System.

A college library provides various resources for students and staff, including books, videos, DVDs and CDs. It is common that several copies are kept of some resources, for example recommended books for courses. The usual loan period of a resource is 2 weeks, but some resources are available for short loan only (2 days) and some other resources can only be used within the library.

The library consists of 3 floors. Resources are stored in the library on shelves. To locate a specific item in the library a combination of floor number and shelf number are used. In addition to this, a class number system is used to identify in which subject area a particular item belongs, for example all resources concerned with Database Systems will have the same class number.

Students hold library cards which identify them as valid members of the Library. Students can lone a number of different resources at one time, but the total number of resources they may borrow at a given time must never exceed 5. Staff members at the College also hold library cards, and are allowed to lone up to 10 different items at one time.

The library charges fines for resources that are loaned for longer than the time allowed for that resource. For each day a resource is overdue the member is fined one dollar. When the amount owed in fines by a member is more than 10 dollars, that member is suspended until all resources have been returned and all fines paid in full.

The system is required to maintain a record of all of the above details. Specifically it needs to keep track of:

- What each resource is, its class number, how many copies of it are held by the library and where these are located in the library
- Student and staff members of the library
- All current loans, including whether they are overdue
- A record of previous loans to help in identifying popular resources
- The details of any fines owed by members
- A list of library members who have been suspended due to overdue loans or unpaid fines