# ECS740P - DATABASE SYSTEMS

## Coursework 1

## College Library System

Group 3

Introduction 2

Assumptions 3

E-R Diagram 5

Relational Database Schema 7

Normalization 9

Conclusions 15

Bibliography 16

**Introduction**

This report aims to outline the conceptual design of a database for a college library. The relationships between entities and attributes within the database system would be explored and discussed by detailing the following:

1. Assumptions which the conceptual designs are based on
2. An entity-relationship diagram of the database
3. A relational database schema
4. A normalised design for the application up to the 3rd normal form
5. A review on whether all specified functions have been met

Requirements

The requirements of the database application are summarised as follow:

* The college library provides various resources including books, videos, DVDs and CDs.
* Some of the resources include multiple copies.
* The loan period of a resource can be 2 weeks, 2 days or can only be used within the library.
* The library has three floors and resources are stored in the library on shelves. A resource is located by the floor number and a shelf number.
* Resources are also assigned a class number, to identify which subject area the item belongs.
* Students and staff members are the main users of the library, both user groups hold library cards which identify them as valid members of the library.
* Students have a borrowing limit of 5 resources at a time, while staff members have a borrowing limit of 10 resources at a time.
* The library charges fine for overdue resources, at a rate of 1 dollar per day.
* When the amount owed in fines by a member is more than 10 dollars. The member is suspended until all resources have been returned and all fines are paid in full.
* The database application is to maintain a record of all the above details and keep track of the following:
  + What each resource is, the class number, how many copies of it are held by the library and where are they 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.

Assumptions

Based on the above requirements certain assumptions have been added to the design of the database application. This is where the requirements have not explicitly specified the functionality of the database and so any ambiguities are cleared.

The assumptions made are classified into three areas and are listed as follow:

Assumptions regarding the policies of the library:

1. Only students or staff members of the college can become a valid member of the library in order to borrow resources. All valid members possess a single valid personal library card with a unique library card ID which cannot be shared.
2. In case of a member bearing both the identity of a student and a staff member, they are classified as a staff member in the system and have a borrowing limit of 10 resources at a time.
3. Resources having the same resource ID are treated as multiple copies of the same resource, this is independent of the edition, publishing year, or any other factor.
4. A member can borrow multiple copies of the same resource at a time, and it would be counted as loan of multiple resources.
5. A member is suspended when the amount owed in fines (resulting from when an overdue resource has breached the given borrow limit) exceeds 10 dollars, instead of when the aggregate amount of fine resulting from all borrowed resources exceeds 10 dollars.
6. A member can borrow the same resource multiple times; however, the member must not borrow the same resource for more than once in a calendar day.
7. A copy of a resource can be returned within the same day of borrowing, such that the copy would be available to be borrowed by another user, but not by the same user (see assumption 6)
8. A copy of a resource can only be borrowed once it has been returned by the previous borrower.
9. Only resources with valid resource ID and copy number can be borrowed.
10. Different copies of the same resource can have different loan periods.
11. Different copies of the same resource should belong to the same category, hence having the same resource class.
12. Every copy of a resource has its own copy number. The copy number is unique within the scope of a resource; however, it is not unique across different resources. For example, if a copy of “Resource A” has a copy number of “01”, no other copy of the same “Resource A” should have the same copy number “01. Nevertheless, there could be another copy with copy number “01” that belongs to a separate resource, say “Resource B”.
13. For each resource returned late, the fine payment for that particular copy of resource must be paid in full, and the copy should be returned immediately after making the fine payment.

Assumptions regarding the scope of the database application:

1. All resources within the database application are tangible and have a finite number of copies. Digital resources such as e-books are not included in this database application.
2. The database application is only designed for one single library within a single building.
3. The librarians and other library staff play no role in this application unless they are a valid member of the library.

Assumptions regarding the physical design of the library:

1. No shelf should span across different floors of the library.
2. Each of the three floors in the library shall contain no more than 999 shelves.
3. The shelf number is consisted of 4 digits, with the first digit being the floor number it is located at. For instance, shelf 1323 is located on 1/F and shelf 2008 is located on 2/F.
4. All resources of the same resource class are stored in their respective designated shelves. For some resource classes with many copies of resources, the copies are stored across multiple shelves. As for some resources classes with few copies, multiple resource classes might share one shelf.

Entity-Relationship Diagram

Figure 1 below is the entity-relationship diagram drawn using Chen ERD notation.

It is worth noting that not all attributes of each entity and relationship are exhausted in the ER diagram. Instead, only the primary keys and several more attributes are included to give clarity of the nature of the entities and the relationship in between.

There are four main entities in the concept designs, namely the user, the resource, the copy, and the fine payment. Copy and fine payment are weak entities, their existences are depended on resource and user respectively. The relationship between the resource and a copy of resource is a one-to-many relationship, i.e., one resource can hold many copies, but a copy can only belong to one resource. The relationship between a user and a fine payment is also a one-to-many relationship since a user can make multiple fine payments, and a fine payment can only be made by one user. The relationship between a user and a copy is many-to-many, it implies that a user can borrow multiple copies, and a copy can be borrowed by multiple users over time.

Also, for enhanced clarity, some derived attributes such as “Resource status” of the “Copy” entity, “outstanding fine” and “due date” of the “borrows” relationship, are included in the diagram with dotted line circles, although they are not intended to be stored in the database application and could be derived on query. More on the method of derivation for these critical data will be explained in the conclusion part.

For the “Prints” and “Multimedia” subclasses of the “Resource” entity, they are considered to have a partial distinct relationship with their parent class. Though it is not specified in the requirements, these subclasses are created for a potential resource searching function, which exists in most common library systems, and that the attributes used when searching for printed resources (including books and journals) and multimedia resources (including CDs, and DVDs) are essentially different. Furthermore, the main reason of having a partial relationship is to avoid the situation of having a new type of resource that could not be classified into either “Print” or “Multimedia” in the future.

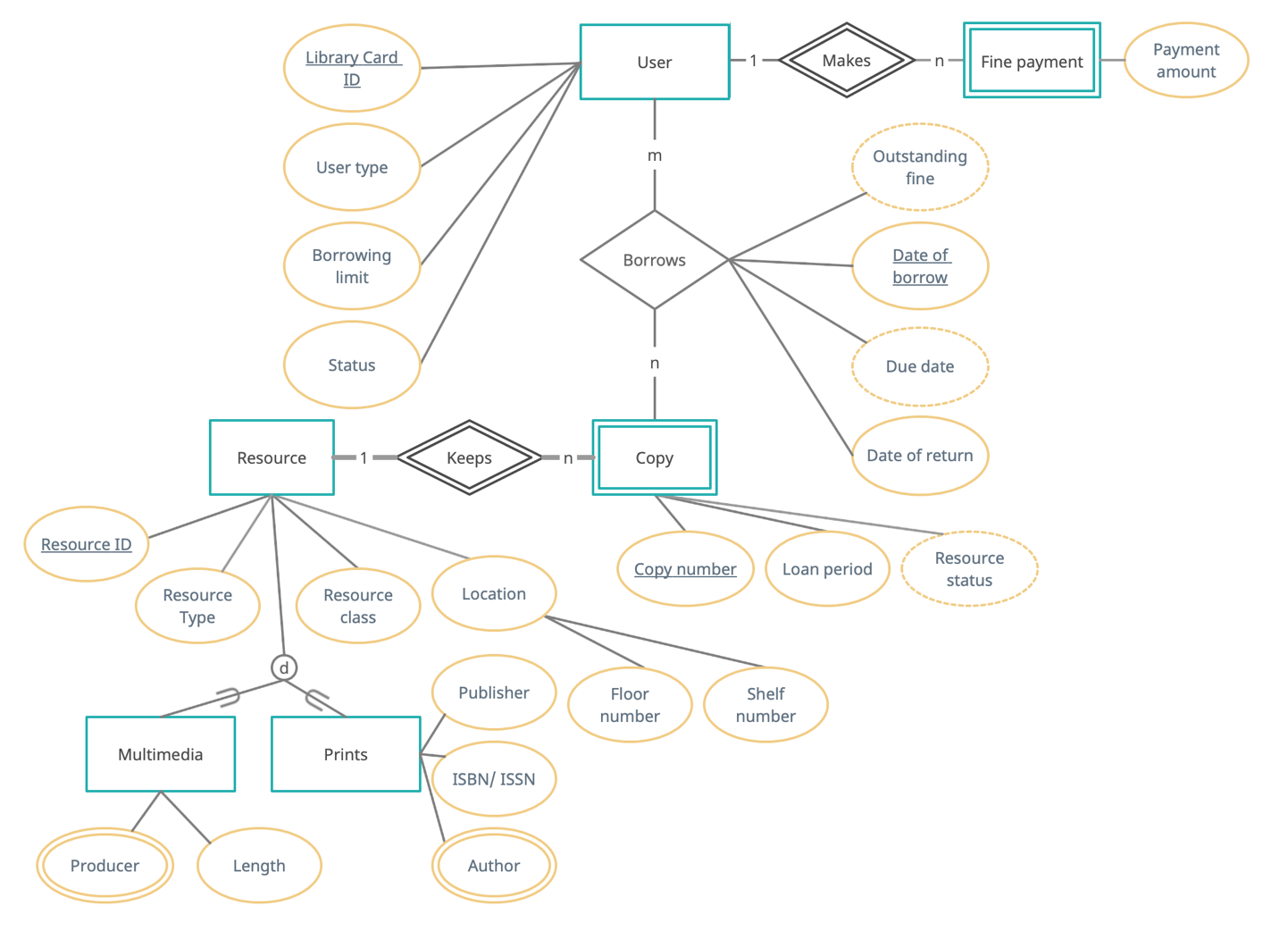


Figure 1. Entity-Relationship Diagram (Chen notation)

**Relationship Database Schema**

From the ER diagram we can map out the relationship database schema below. The primary keys for each entity will be identified with an underscore, while the foreign keys will be identified in italics.

**User** (Library Card ID, Name, Mobile Number, User Type, Borrowing Limit, Status)

As stated in the assumption, each valid user of the library owns a library card with a unique Library Card ID. Mobile number and name of user could be the candidate keys, but some users might not have a mobile number and hence is not suitable to be a primary key. Name, on the other hand, might not be unique. Thus, the Library Card ID is considered the best choice of primary key to identify every user in the User table. User Type, Borrowing Limit and Status are attributes specified in the requirements of the database application. While the user type and borrowing limit are specified in the library policy, the status attribute can be used to indicate whether the user is active, suspended, or deactivated. Name and mobile number, though not selected as a primary key, we have added them in the table as non-key attributes for potential additional functions, such as to print the user’s name on the library card, or to contact the user for giving item returning reminders.

**Resource** (Resource ID, Title, Resource type, Resource class, Location)

**Prints** (Resource ID, Publisher, ISBN\_ISSN, Author)

**Multimedia** (Resource ID, Producer, Length)

With “Prints” and “Multimedia” being the subclasses of “Resource”, we have the option of placing the attributes of the subclasses as columns of the main “Resource” table. However, as they would involve the unnecessarily extensive use of null values, we have chosen to isolate the exclusive attributes of both subclasses and build separate tables.

For the main Resource table, as the database application is required to track the resource type, resource class and location of each resource, they are included as the main attributes with the unique Resource ID as the primary key. Finally, a Title attribute is also included in the Resource table for conventional purpose. The Title of the resource could be a candidate key, but again there might be resources with different content having the same title.

**Copy** (*Resource ID*, Copy number, Loan period)

The Copy entity is considered to be a child of the Resource entity, in other words, a weak entity, as the existence of a copy depends on the parent entity. With the assumption that the loan period could be different for each copy of the same resource, the loan period attribute is included in the Copy table instead of in the Resource table. Also, having considered the weak relationship between the two entities, the foreign key of Resource ID is used together with the copy number to form the primary keys of the table, to uniquely identify the tuples.

**Loan** (*Library Card ID*, *Resource ID*, *Copy number*, Date of borrow, Date of return)

The next step is to map the many-to-many relationship between the library users and the copies they borrow. Since it is a many-to-many relationship, we have separately created a loan table to store all the details of each loan. The primary keys of the User table and the Copy table, namely the Library Card ID, Resource ID and the Copy number are joined by the Date of Borrow to form the primary keys of the Loan table, since a user cannot borrow the same copy of the same resource multiple times within a single day. The only non-key attribute in the table is the date of return, which is intended to have a default null value, indicating that the borrowed item has not yet been returned.

**Payment** (*Library Card ID*, *Resource ID*, *Copy number*, *Date of borrow*, *Date of return*, Payment amount)

To keep track of the received fine payment, a new Payment table is formed. Although the primary keys of this table are identical to the above Loan table, hence having all four of the primary keys foreign, there are some intrinsic differences between the two tables.

As the main purpose of this Payment table is to track fine payment history and to spot frequent violators, a tuple is added to the table once a fine payment has been made, versus a tuple is added to the Loan table once an item is being borrowed.

One might argue that the payment amount attribute could be added to the Loan table as an additional attribute such that the fine payments could be tracked without using an extra table, and that could be true. However, as we anticipate that most of the items would be returned to the library in a timely manner, most records in the Loan table would not involve a fine. Therefore, having a specific table to record the fine payments can avoid a massive number of null values, and let the library staff have a focused view of the fine payments.

It is worth noting that a foreign key of “Date of return” is also imported to the table, this is also to keep track of the payment date, as the payment date would be the same day the item is returned to the library.

Below the full relational database schema is found, including a diagram:

**User** (libcard\_id, name, mobile\_number, user\_type, borrowing\_limit, status)

**Resource** (resource\_id, title, resource\_type, resource\_class, location)

**Multimedia** (*resource\_id*, producer, length)

**Prints** (*resource\_id*, publisher, author, isbn\_issn)

**Copy** (*resource\_id*, copy\_number, loan\_period)

**Loan** (*libcard\_id, resource\_id, copy\_number,* date\_borrow, date\_return)

**Payment** (*libcard\_id, resource\_id, copy\_number, date\_borrow*, *date\_return*, payment\_amount)

Graphical user interface, application, Teams

Description automatically generated

Figure 2. Graphical representation of Relational Database Schema

**Normalization**

**First Normal Form**

With the primary keys finalised from the above session, it is now possible to move on to the normalization. This includes finalizing the grouping of the attributes in the entities.

Initially, first form normalization is defined as the following:

‘*A relation in which the intersection of each row and column contains one and only one value*’ (Connelly & Begg, 2015: 573)

A relation is in first normal form if it does not contain any composite or multi-valued attribute.

Now applying the above definition to the entities created from the specification, it is possible to deduce that:

**User** (libcard\_id, last\_name, first\_name, country\_code, mobile\_number, user\_type, borrowing\_limit)

To break down all the columns into atomic value and make the content more readable, the attribute of Name has been separated as Last\_Name and First\_Name; Meanwhile, the attribute of Mobile\_Number has been separated as Country\_Code and Mobile\_Number.

**Resource** (resource\_id, title, resource\_type, resource\_class, floor\_number, shelf\_number)

To break down all the columns into atomic value and make the content more readable, the attribute of Location has been separated as Floor\_number and Shelf\_number.

**Copy** (resource id, copy\_number, loan\_period)

This class is already in first normal form as there are no instances of attributes that contain multiple values.

**Prints** (resource\_id, publisher, isbn\_issn, author)

It is possible that a printed resource, e.g., a book or a journal, could have more than one author with the same publisher and ISBN/ISSN. To remove the multi-valued attributes, we should have a separate relation for (Resource ID, Author) pairs:

**Print\_info** (resource\_id, publisher, isbn\_issn)

**Print\_author** (resource\_id, author)

**Multimedia** (resource id, producer, length)

Like the Prints entity, it is possible that a multimedia could have more than one producer. To remove the multi-valued attributes, we should have a separate relation for (Resource ID, Producer) pairs:

**Multimedia\_Producer** (resource\_id, producer)

**Multimedia\_Length** (resource\_id, length)

**Loan** (libcard\_id, resource\_id, copy\_number, date\_borrow, date\_return)

This loan table is already in first normal form as there are no instances of attributes that contain multiple values.

**Payment** (libcard\_id, resource\_id, copy\_number, date\_borrow, payment\_amount)

Again, this loan table is already in its first normal form as there are no instances of attributes that contain multiple values.

**Second Normal Form**

Second form normalization (2NF) is defined as the following:

*‘A relation that is in first normal form and every non-primary-key attribute is fully functionally dependent on the primary key’* (Connelly & Begg, 2015: 578)

To normalise the relationships into second normal form, the following rules should be applied.

* The relationships are already in their first normal forms.
* All non-key attributes should be fully functionally dependent on all of the primary keys.
* Any relation must not contain any partial dependencies.

Therefore, it can be developed as:

**User** (libcard\_id, last\_name, first\_name, country\_code, mobile\_number, user\_type, borrowing\_limit, status)

**Resource** (resource\_id, title, resource\_type, resource\_class, floor\_number, shelf\_number)

The two relationships above are in second normal form as there are no composite primary keys and no partial dependencies on any non-key attribute. In other words, all non-key attributes are fully dependent on all parts of their primary key.

**Copy** (resource\_id, copy\_number, loan\_period)

This primary key is a composite value with Resource\_ID and Copy\_number. However, this relationship is already in second normal form as there is no partial dependencies on any non-key attribute. All non-key attributes are fully dependent on all parts of their primary key.

**Print\_info** (resource id, publisher, isbn\_issn)

**Print\_author** (resource id, author)

**Multimedia\_Producer** (resource id, producer)

**Multimedia\_Length** (resource id, length)

The above four tables are in second normal form as there are no composite primary keys and no partial dependencies on any non-key attribute. Therefore, all non-key attributes are fully dependent on all parts of their primary key.

**Loan** (library card id, resource id, copy number, date\_borrow, date\_return)

We are going to compare each of the non-key attributes in relation to the four primary keys and check whether they are dependent on all four of them.

For the date of return, it cannot be dependent of the library card ID alone, as the same user can borrow multiple resources at one time, it is necessary to specify which resource is being returned. Hence the resource ID is added in, but it is still not sufficient, since a user can borrow multiple copies of the same resource in one go, therefore the copy number as well must be depended on. The date of borrow is also depended on, as the same user can borrow the same copy over multiple days, the date of borrow is there to specify which loan record is the copy being returned to. Furthermore, as there might be situations where the same copy is borrowed and returned by one user, and being borrowed by another user again, all within the same day, the library card ID is also depended on. Therefore, the date of return attribute is functionally dependant on all four primary keys. In other word, this entity is in second normal form already.

**Payment** (libcard\_id, resource\_id, copy\_number, date\_borrow, date\_return, payment\_amount)

Regarding the payment amount we need to perform the following process; we need to map the fine payment with the library card ID to specify which user’s borrowing record the payment is directed to; map the resource ID and copy number to specify the copy which the fine is for; and map the date of borrow to specify, over time, precisely which borrowing record it is. Therefore, the payment amount attribute is functionally dependant on all four primary keys. In other word, this entity is in second normal form already.

This has proved that all non-key attributes should be fully functionally dependent on all the primary keys in the table.

**Third Normal Form**

Third form normalization is defined as the following:

*‘A relation that is in first and second normal form and in which no non-primary-key attribute is transitively dependent on the primary key.’* (Connelly & Begg, 2015: 580)

To normalise the relationships into third normal form, the following rules should be applied.

* The relationships are already in their second normal forms.
* All transitive dependencies within a table are eliminated.
* No non-key attribute is needed to be changed resulting from an update of another non-key attribute.

**User** (libcard\_id, last\_name, first\_name, country\_code, mobile\_number, user\_type, borrowing\_limit)

Inspecting for transitive dependencies, we know that Borrowing\_Limit is determined by User\_Type alone, so there is a transitive dependency between Borrowing\_Limit and (Resource\_Id, Copy\_number) via User\_Type. So, we take Borrowing\_Limit out of the 2NF relation to get:

**User** (libcard\_id, last\_name, first\_name, country\_code, mobile\_number, user\_type)

**User\_Limit** (user\_type, borrowing\_limit)

**Resource** (resource\_id, title, resource\_type, resource\_class, floor\_number, shelf\_number)

Inspecting for transitive dependencies, we know that Floor\_number is determined by Shelf\_number alone, so there is a transitive dependency between Floor\_number and (Resource\_Id, Copy\_number) via Shelf\_number. So, the above table could be split into the below:

**Resource** (resource\_id, resource\_type, resource\_class, shelf\_number)

**Location** (shelf\_number, floor\_number)

In the meantime, we also assumed that the same class of resource will be stored in their designated shelves, we can hence take the involved attributes out of the above tables to get:

**Resource** (resource\_id, resource\_type, resource\_class)

**Resource\_shelf** (resource\_class, shelf\_number)

**Location** (shelf\_number, floor\_number)

**Copy** (resource\_id, copy\_number, loan\_period)

**Print\_info** (resource id publisher, isbn\_issn)

**Print\_author** (resource id, author)

**Multimedia\_Producer** (resource id, producer)

**Multimedia\_Length** (resource id, length)

No transitive dependencies can be found in the above five tables, hence they are in 3NF.

**Loan** (library card id, resource id, copy number, date\_borrow, date\_return)

Although the primary key is a composite key comprised of Library Card ID, Resource ID, Copy Number and Date of borrow, there is only one single non-key attribute, which is the Date\_return. Therefore, there is neither no transitive dependency between non-key attribute and primary key nor any relation between each non-key attribute. The attribute Date\_return is fully dependent on the whole set of primary keys.

**Payment** (libcard\_id, resource\_id, copy\_number, date\_borrow, date\_return, payment\_amount)

There are only two non-key attributes in the table, which are the Date\_return and the Payment\_amount. While there are no dependencies between the two attributes, no transitive dependencies exist in the Payment table.

**Conclusions**

After reaching third form normalisation, we can now review whether the conceptual design is a sufficient basis for building a database application that meets all the requirements.

On the resource side, for the type, loan period and location of the resources, they are stored as attributes either in the resource table or in the copy table. This is after considering their respective functional dependencies.

On the user side, the identity of the user (staff or student) and borrowing limit are also stored as attributes in the User and Location tables.

Other requirements without an associated stored attribute could be derived using queries, with simple explanation as follow:

* The number of copies of each resource can be derived by finding the number of tuples with the given Resource ID.
* All current loans can be filtered from the Loan table, where the date of return is a null value.
* From the list of current loans, it is also possible to filter the number of current loans initiated by each user. By comparing the number with the borrowing limit of the user, the system can ensure that a user cannot borrow past his/her limit.
* Similarly, a record of previous loans can be filtered from the Loan table, where the date of return is not a null value. This can help identifying popular resources.
* All overdue current loans can be filtered from the Loan table, joining with the Copy table to find the loan period of the copy. Then the loan period is added to the date of borrow in the loan table, to get the due date of the loan. If the due date is past and the return date remains a null value, the tuple is an overdue current loan.
* The details of any current fine can be queried with a similar flow, joining the Loan table with the Copy table to find out the loan period of the copy, hence deriving the due date of the loan record. If the return date is a null value, the amount of current fine can be calculated by .
* The suspension of a user can be triggered by the accumulation of fines, with the calculations shown above. Moreover, as the status of a user is set as a stored attribute, end user of the database application can also arbitrarily suspend or deactivate a user account on other reasons, such as a student graduating or a staff member leaving service.

Besides the specified requirements, the design concept also enables the potential implementation of some extra functions, such as searching for a resource by the title, author, ISBN for a printed resource, or by the producer for a multimedia resource. Moreover, the library can keep a detailed record of past fine payments.

**Bibliography**

*Thomas Connelly & Carolyn Begg, Database Systems: A Practical Approach to Design Implementation and Management - 6th Edition, Pearson Education 2019*