2020

Group Coursework 1

COLLEGE LIBRARY SYSTEM

GROUP 8

STUDENT ID:
200773869
200149271
200873723

Contents

Introduction

The following report will create an outline for a college library system database. As the aim of the report is to map out the conceptual design of the database, it will include:

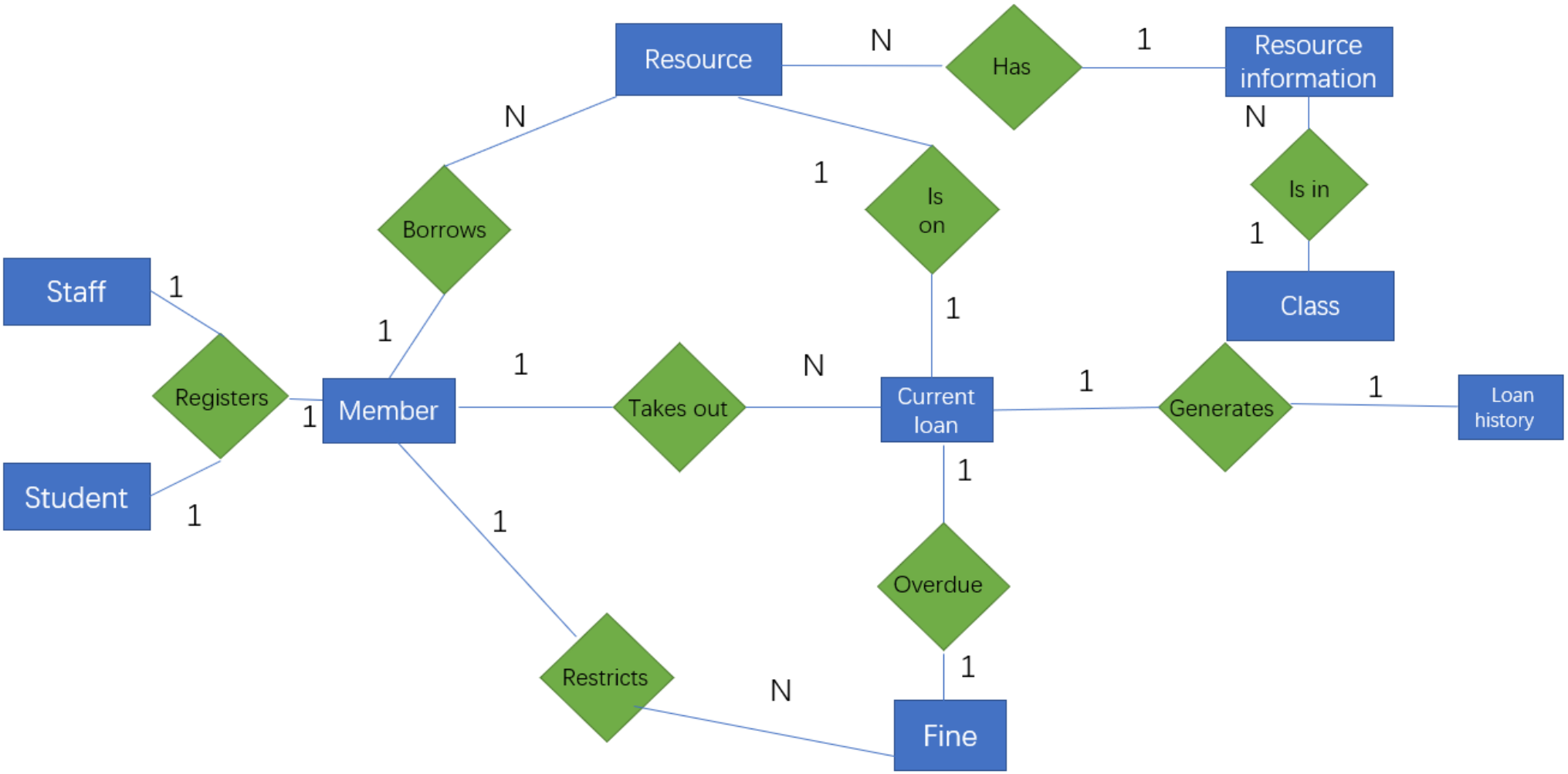
- All assumption made when designing the database
- An ER diagram of the database
- The relational database schema
- The normalisation processes up to Third Normal Form

Assumptions

These are assumptions derived from logical assumptions regarding how the library operates and to keep the system from going out of scope:

1. The college has another database that stores more detailed data on the students and staffs, which we can access when needed. We are assuming that there is already a pre-existing, separate database from which library staff can extract fundamental information about students and staff.
2. We assume that all staff and student are automatically registered as library members by the library staff upon enrolment to the college.
3. Students and staff receive a unique ID number and email upon enrolment as a member of the college.
4. All staff members (including library staff) are treated the same by the system.
5. The maximum number of shelves on each floor is 100.
6. 'Loan History' is only used to check popularity during a particular period.
7. Only college emails will be accepted when people are added to the database, and students and staff cannot change their email.
8. If a resource has no ISBN or EAN, library staff will assign a unique 13-digit code to distinguish it from others.

Entity-Relationship Diagram



Firstly, 'Staff' and 'Student' keep a clear, separate records of students' and staff's personal information. The entity 'Member' keeps track of members' accessibility to the library. 'Staff' and 'Student' must enrol as members to gain library membership rights, borrow books. Each student or member of staff can only hold one membership with the library. We wondered whether the 'Student', 'Staff' and 'Member' entities could be combined within the single entity 'Member' but decided that having three separate entities makes it simpler to identify which members have or have not registered with the library.

Secondly, a unique 'Resource ID' for each copy of a single resource is designed to monitor where is it. In order to avoid data redundancy, which is caused by showing the same identical information multiple times for books with several copies, we separated the resource table into two entities. The first one is 'Resource', and it holds the availability of each copy of resources. The other is 'Resource information', and it shows details of the resources (e.g. title, author, publisher, etc.)

Thirdly, a class number system is used to identify the subject area for a particular resource. The 'Class' entity records which class number is associated with which subject. A resource referenced in the entity 'Resource information' can only have one subject but a subject can have numerous resources.

Furthermore, we introduced 'Current Loan' to store data about resource copies currently on loan. When a member takes out a resource(/resources), related loan data is generated for each copy taken out. A member can borrow multiple resources or copies, but each must be returned within a set period. Details are recorded in the entity 'Current Loan' for each copy borrowed of a resource, and each piece of information is only belonging to one member. As each copy in 'Resource' can only be borrowed by one member, each item from 'Resource' takes up just one row in 'Current Loan'.

Since members can only hold items for a limited period, if resources are not returned before the due date (they are overdue), a member may be fined. The fine total is calculated from the number of days that a resource is overdue, each fine only corresponds to one row in 'Current Loan'.

We designed 'Loan History' to record loan information about a resource loan from the point when an item is returned. Therefore, the necessary information about that loan in 'Current Loan' is then transferred to 'Loan History'. Once an item is returned, either library staff or the system can transfer a loan's details to the loan history entity. Each row in 'Current Loan' can only generate one row in 'Loan History'.

'Fine' can be used to see whether a fine has been paid off. Each fine is associated with one member but a member might owe multiple fines.

We separated 'Current Loan' from 'Loan History' to make it easier to find information about loans since we want to use information about current loans for separate purposes than for past loans. 'Loan History', as the specification requires, is only needed to check the popularity of a resource but 'Current Loan' is used to track member responsibility for present loans and calculate total fines.

The first reason having the separate attributes ‘Student_ID’, ‘Staff_ID’ and ‘Email’ was that students and staff have different loan quotas which need to be recorded. The second is that students typically study in a college for a few years. Therefore, their membership end date is set for the whole year group. However, a member of staff’s membership end date depends on their individual plans. As 'Student_ID’ is a separate attribute, it should be distinguishable from ‘Staff_ID’- this enables library staff to tell whether the user is a student or member of staff.

Relational Database Schema

This section of our report maps out and explains the relational database schema that we decided on based on our ER diagram. For each entity, we have underlined the primary key and italicised any foreign keys. We have omitted derived attributes from our ER diagram and relational schema since these can easily be calculated from present attributes (we will explain how below).

We use foreign keys to enforce referential integrity by securing an association between pairs of entities.

Student (Student_ID, S_Name, *Email*, S_Start_Date, S_Finish_Date)

- We chose 'Student_ID' as the primary key here since this makes it possible to uniquely identify a particular row in the 'Student' table. The college creates student IDs so that each student has an individual student id, already obtained at the time of enrolling on to the college. Additionally, 'Student_ID' is reliable as it does not change. 'S_Name' is a less desirable candidate since whilst it might work for a small department where two people would be unlikely to have the same name, it is inadequate for a college holding many people.
- The name and email attributes are there so the library system can quickly send emails such as reminders and invoices to students without having to access a college database from outside of the library. The start and finish dates of a student at the university are needed to check the validity of the student membership. 'Email' is used as a foreign key to connect the 'Student' and 'Member' entities. We decided to make 'S_Start_Date' and 'S_Finish_Date' simple attributes rather than composite attributes (split up into day, month and year) since SQL typically treats dates as single values.

Staff (Staff_ID, ST_Name, ST_Start_Date, ST_Finish_Date, *Email*)

- We chose 'Staff_ID' as the primary key since it uniquely identifies records within the 'Staff' table and staff are given one as soon as they join the college. The 'Name', 'Email', 'ST_Start_Date', 'ST_Finish_Date' and 'Email' attributes are added for the reasons mentioned in the 'Student' entity explanation.
- We decided to have 'ST_Start_Date' and 'ST_Finish_Date' both as single attributes. Whilst 'ST_State_Date' and 'ST_Finish_Date' could each be considered to be a composite attribute (composed of ST_Start_Day, ST_Start_Month and ST_Start_Year), SQL typically treats dates as single values.

Member (*Email*, Resource_Cap)

- We chose 'Email' as the primary key since each student and staff member is given just one email address when they start with the college (which does not change)- Therefore, it is a reliable primary key. Additionally, 'Email' is able to reference the student and staff entities.

- Additionally, ‘Resource_Cap’ is the maximum number of resources that a member is permitted to borrow- it is based on the separate numbers of books that students and staff can borrow at one time.
- A member’s suspension status (showing which members are permitted to take out a current loan) can be derived from the ‘Fine’ and ‘Current_Loan’ tables (as shown below).

Resource (Resource ID, *ISBN/EAN*, Location, #th_copy)

- The primary key for resource availability is ‘Resource_ID’, which is a unique code for a particular copy of a resource. Using ‘Resource_ID’ allows library staff to check which resources are on loan.
- The foreign key is ISBN, linking ‘Resource’ with the entity ‘Resource Information’ by providing separate information about a resource. ‘#th copy’ is used to help library staff monitor resource availability, showing which copies are on loan or remain in stock.
- The attribute ‘Location’ helps to locate a particular resource in the library- a composite attribute combining ‘Floor’ and ‘Shelf_Number’. The Location attribute is a combination of floor number and shelf number with the format ‘FSSS’. The ‘F’ stands for floor number and ‘SSS’ stands for shelf number.

Resource Information (*ISBN/EAN, Class_Number*, Resource_Type, Title, Author, Publisher, Edition, Publication_Year, Loan_Period)

- ISBN/EAN: ISBN stands for “International Standard Book Number”. It contains 13 digits, which can be used as a product identifier by publisher, retailers, libraries and suppliers for multiple purposes including ordering, listing, sales and stock controlling (ISBN, 2020). An ISBN is exclusively assigned to a publication with a specific title, edition and format, and can be identified as long as the book made publicly available. For some items which don not have an ISBN, the system will use its EAN (International Article Number or also called European Article Number), a 13-digits unique identification number. Also, ISBN and EAN are compatible because the EAN for items which already has an ISBN is the ISBN (GS1, 2015). Therefore, it makes sense to store ISBN/EAN in the same column.
- Here, the primary key is ‘ISBN/EAN’. An ‘ISBN/EAN’ can uniquely identify a resource but cannot be used to identify a copy of that resource. The primary key ‘ISBN/EAN’ enables the system to list every resource without repeating them in this entity.
- ‘Loan_Period’ is needed to identify the length of time a resource can be taken out for. If the item is only available for use inside the library, the loan period will be 0. For resources which can only be used inside the library, the libary will deny issuing the book to members and showing "this resource is only available to use inside the library" when it detects the loan period is 0 day.
- ‘Class_Number’ (a foreign key) makes it possible to identify the subject of each resource (e.g. books on Software Engineering have the same class number).

- The total number of copies can be derived from the “Resource” table, and derived by calculating the total number of rows with a particular ‘ISBN/EAN’.

Class (*Class_Number*, Class_Name)

- The Primary key for this entity is ‘Class_Number’- which is used to identify the subject of a resource.

Current Loan (Loan_Ref#, *Email*, *Resource_ID*, Issue_Date)

- ‘Loan_Ref#’ is the primary key here, and it is generated when a user borrows a resource from the library. This uniquely identifies a row in the Current Loan table. We have the foreign keys ‘Loan_Ref#’, ‘Email’ and ‘Resource_ID’ to link the ‘Current Loan’ table to the ‘Resource’ and ‘Member’ tables.
- Before a member can take out a resource, the resource’s availability needs to be checked- found by checking if there is a row with it’s ‘Resource_ID’ in the ‘Current Loan’ table.
- When an item is issued by the library, ‘Issue_Date’ will be recorded as that day.
- ‘Email’ is needed as a foreign key to be able to identify who is responsible for returning a resource on time. ‘Resource_ID’ is key to identifying which particular copy has been borrowed.
- We considered ‘Due Date’ as an attribute but decided against having it since it can be derived from the ‘Issue_Date’ and ‘Loan_Period’ attributes in the entity ‘Resource Information’.

Loan History (*Loan_Ref#*, *Resource_ID*,Return_Date, *Issue_Date*)

- The primary key here is ‘Loan_Ref#’.
- We omitted email from this entity since the specification simply asks for a record of ‘Loan History’ to make resource popularity easily discoverable, and email is not necessary for doing this.
- The popularity of a resource can be found in ‘Loan History’ by totalling the number of rows that have the same ‘ISBN/EAN’ number across a particular timeframe or by checking how recently a copy of a resource was taken out. The ‘ISBN/EAN’ associated with a ‘Resource_ID’ can be found through a join of the ‘Resource’ and ‘Resource Information’ tables.
- Information under the same ‘Loan_Ref#’ will be automatically deleted from ‘Current Loan’ and added to ‘Loan History’ when a resource is returned.

Fine (*Loan_Ref#*, Date_Paid, *Email*)

- The ‘Fine’ table is here to help library staff determine which fines have been paid off.

- The primary key is ‘Loan_Ref’ as each loan can only be related to one fine. ‘Loan_Ref’ links ‘Current Loan’ to the entity ‘Fine’, which can identify which item on loan caused a fine.
- The ‘Fine’ table’s attribute ‘Date_Paid’ can be used to check who has paid off a loan and who needs to be invoiced. Suspended members have a fine total of 10 or more dollars and ‘Resource_Cap’ will be changed to 0 for that member in the ‘Member’ table. Our research suggests to us that a 'Fine_Total’ attribute would be useful in real-life application but can still be derived from the ‘Current Loan’ entity, and is not strictly necessary. ‘Email’ is a foreign key that connects the ‘Fine’ entity to the ‘Member’ entity.
- ‘Email’ can be derived from the ‘Current Loan’ entity but is still in this entity as there could be a situation where the resource has been returned but the fine for that resource has not been paid.
- ‘Due Date’ can be derived from ‘Loan History’ and ‘Resource Information’.
- The total fine amount can be derived by comparing ‘Due Date’ against Current Date (this is easily calculated with SQL) and multiplying the difference by $1
- The suspension status of a member can be found by calculating the fine total of each row with a particular ‘Email’ where ‘Date Paid’ has null as its value.
  This total is compared against 10 to see if the member has been suspended. This process can be repeated to get a list of suspended members.

Normalisation

In order to check the functionality of the system, simplify and reduce redundancy in the system, we will now remodel the system, apply normalisation and compare the result with our design. To do this, we will re-examine the original specification and note the attributes it calls for without being influenced by preconceptions of how the database should be. This approach suggests having the following attributes:

- ISBN/EAN (needed to Identify the Resource)
- Class_Number (needed to identify the resource subject and implement the class number system)
- Class_Name
- Location (needed to locate a resource)
- Resource_Type
- Title
- Author
- Publisher
- Edition
- Publication_Year
- Loan_Period
- #th_copy (Needed to identify a particular copy of a resource)
- Loan_Ref#
- Issue_Date
- Return_Date
- #Resource_Cap
- Student_ID
- Staff_ID
- Name
- Email (Needed to identify the resource borrower)
- Start_Date
- Finish_Date
- Date_Paid

We will now organise the basic attributes into groups according to the information that the attributes would contain. We will also work out what the suitable primary keys would be.

The initial step in doing this is to note the entire set of universal relations for the attributes listed above:

U(ISBN/EAN ,Class_Number ,Class_Name ,Location ,Resource_
Type ,Title ,Author ,Publisher ,Edition ,Publication_Year ,Loan_Period ,#th_copy, Loan_Ref# ,Issue_Date ,Return_Date ,Resource_Cap ,Student_ID ,Staff_ID, Name, Email, Start_Date, Finish_Date, Date_Paid)

Next, we will separate out the set of universal relations by identifying their functional dependencies- this will help us to group the entities.

Member => (Email, Staff_ID, Student_ID, Name Resource_Cap, Start_Date, Finish_Date)

Resource => (ISBN/EAN, #th_copy, Class_Number, Class_Name, Location, Resource_Type, Title, Author, Publisher, Edition, Publication_Year, Loan_Period)

Loan => (Loan_Ref#, ISBN/EAN, #th_copy, Email, Issue_Date, Return_Date, Date_Paid)

Finally, we will note the grouped entities that we have been led towards having, and examine candidate keys in order to work out the most appropriate primary keys.

Member (Email, Staff_ID, Student_ID, Name, Resource_Cap, Start_Date, Finish_Date)

- A composite primary key of Staff_ID and Student_ID could uniquely identify the tuples of this table but considering that each student and staff member have their own unique college email, and 'Email' has been chosen as the primary key for simplicity.

Resource (ISBN/EAN, #th_Copy, Class_Number, Class_Name, Location, Resource_Type, Title, Author, Publisher, Edition, Publication_Year, Loan_Period)

- Having 'ISBN/EAN' and '#th_copy' as a composite primary key is the only way to identify a particular copy of a resource since 'ISBN/EAN' by itself is insufficient in identifying particular copies and '#th_copy' is insufficient in identifying a particular resource.

Loan (Loan_Ref#, *ISBN/EAN*, #th_Copy, *Email*, Issue_Date, Return_Date, Date_Paid)

- Having 'Loan_Ref#', 'ISBN/EAN' and '#th_copy' together as a composite primary key is the simplest way to identify a loan uniquely. Whilst 'Loan_Ref#' is a unique reference generated when a book is borrowed from the library, to identify a particular loan instance, we must know what resource and which copy a member is taking out.

Now that we have sorted the attributes into appropriate groups, we can proceed to normalise the design.

First Normal Form Normalisation

Applying First Normal Form to the entities in this schema, we find that:

Member (Email, Staff_ID, Student_ID, Given_Name, Surname, Resource_Cap, Start_Date, Finish_Date)

Here is an example:

Email	Student_ID	Staff_ID	Name	Start_Date	Resource_Cap	Finish_Date
mc081@college.ac.uk	10914489	-	Matilda Clarke	11/09/2020	5	11/06/2024
si.cheng@college.ac.uk	-	00003231	Si Cheng	11/09/2020	10	-

We will now split Name into two separate columns, ‘GIven_Name’ and ‘Surname’. This entity is now in First Normal Form since there are no sets of values in any of the column’s rows.

Here is an example:

Email	Student_ID	Staff_ID	Given_Name	Surname	Start_Date	Resource_Cap	Finish_Date
mc081@college.ac.uk	10914489	-	Matilda	Clarke	11/09/2020	5	11/06/2024
si.cheng@college.ac.uk	-	00003231	Si	Cheng	11/09/2020	10	-

Resource (ISBN/EAN, #th_copy, Class_Number, Class_Name, Location, Resource_Type, Title, Edition, Publication_Year, Loan_Period)

Here is an example of the 'Resource' table before normalisation- this is not yet in First Normal Form since there are instances of attributes with multiple values in a row of the author and publisher columns:

ISBN/EAN	#th_copy	Class_Number	Class_Name	Location	Resource_Type	Title	Author	Publisher	Edition	Publication_Year	Loan_Period
9781292061184	1	0001	Computer Science	0012	Book	Database Systems	Thomas Connolly, Carolyn Begg	Pearson	6th	2010	1 week
9781292061184	2	0001	Computer Science	0012	Book	Database Systems	Thomas Connolly, Carolyn Begg	Pearson	6th	2010	1 week
9781292061184	3	0001	Computer Science	0012	Book	Database Systems	Thomas Connolly, Carolyn Begg	Pearson	6th	2010	1 week
9780198228589	1	0015	History	1120	Book	British history, 1815-1906	Norman McCord	Oxford University Press	1st	1991	1 week
9781907838101	1	0023	Foreign Language Study	0013	CD	Chinese in steps. Student book 1	George Zhang, Linda Li, Lik Suen	Sinolingua London Ltd.	1st	2011	2 days
9787560076195	1	0023	Foreign Language Study	0013	Book	Little Oxford English-Chinese Dictionary	Oxford University Press	Foreign Language Teaching and Research Press, Oxford University Press	9th	2015	1 week

This table contains anomalies since we cannot list authors or publishers without there being a resource/book (e.g. we might want to just add information about an author) since ISBN/EAN is the primary key and so cannot contain null values (this is called an insertion anomaly). We also cannot delete a resource without losing information on the author or publisher (this is a deletion anomaly). In addition, when updating information such as the author's name or publisher we would have to modify data in every row, which risks distorting or corrupting the data- called an update anomaly (Connolly and Begg, 2015). This table is also not storage-efficient since if we had a large number of resources to store in the database, all the information would need to be repeated across a huge number of rows.

Data inconsistency is an additional problem. If Norman McCord had written multiple books which needed to be stored in the system, library staff would have to type his name into the database multiple times and it seems likely that McCord's name would be spelt incorrectly at some point. This would render the data inconsistent and make it difficult for library staff to search for a book with just the author's name since certain results will be missing.

We currently have sets of values in the author column. The authors' names are also not atomic- the author's given name and surname are actually different values. Without separating these values, it is hard to sort on either surname or given name. To fix this, we will need to break up this table's entities into separate tables. We will therefore need a separate Author table and separate Publisher table.

For Example:

Author_ID	A_Given_Name	A_Surname
1	Norman	McCord
2	Thomas	Connolly
3	Carolyn	Begg

Publisher_ID	Publisher
1	Pearson
2	Foreign Language Teaching and Research Press
3	Oxford University Press

The Author and Publisher tables will need to use a surrogate primary key- which is artificially made to deal with an unserviceable natural primary key. We cannot use the combination of the author's given name and surnames as a primary key since the author's name might not be unique in a large database. Placing the data into separate tables based on the entities represented by that data, we overcome previously spotted anomalies. We can now add authors without needing to add a resource, we can delete resources without simultaneously having to remove author or publisher details.

Yet, since the relationship between 'Resource' and 'Author' is many-to-many (i.e. one author can write multiple books and one book can be written by multiple authors), we must have another table to link the two- a 'Book-Author' table (here we also exemplify how one book having 2 authors can be represented). The relationship between 'Resource' and 'Publisher' is also many-to-many (e.g. we sometimes find books with more than one publisher) so we also need a 'Resource-Publisher' table:

For Example:

ISBN/EAN	Author_ID
9780198228589	1
9781292061184	2
9781292061184	3

ISBN/EAN	Publisher_ID
9780198228589	1
9787560076195	2
9787560076195	3

In summary, this suggests the need to create the following new tables:
Author (Author ID, A_Given_Name, A_Surname)
Resource–Author (*ISBN/EAN*, *Author ID*)
Publisher (Publisher ID, Publisher)
Resource-Publisher (*ISBN/EAN, Publisher*)
Along with the following slimmed-down Resource table:

Resource (ISBN/EAN, #th_copy, Class_Number, Class_Name, Location, Resource_Type, Title, Edition, Publication_Year, Loan_Period)

For Example:

ISBN/EAN	#th_copy	Class_Number	Class_Name	Location	Resource_Type	Title	Edition	Publication_Year	Loan_Period
9781292061184	1	0001	Computer Science	0012	Book	Database Systems	6th	2010	1 week
9781292061184	2	0001	Computer Science	0012	Book	Database Systems	6th	2010	1 week
9781292061184	3	0001	Computer Science	0012	Book	Database Systems	6th	2010	1 week
9780198228589	1	0015	History	1120	Book	British history, 1815-1906	1st	1991	1 week
9781907838101	1	0023	Foreign Language Study	0013	CD	Chinese in steps. Student book 1	1st	2011	2 days
9787560076195	1	0023	Foreign Language Study	0013	Book	Little Oxford English-Chinese Dictionary	9th	2015	1 week

Loan (Loan_Ref#, *ISBN/EAN*, *#th_copy*, *Email*, Issue_Date, Return_Date, Date_Paid)

This entity is in First Normal Form already as there are no instances of attributes that contain multiple values.

Example:

Loan_Ref#	ISBN/EAN	#th_copy	Email	Issue_Date	Return_Date	Date_Paid
000120482	9781292061184	3	si.cheng@college.ac.uk	01/10/2020	03/10/2020	04/11/2020
000120483	9780198228589	1	mc081@college.ac.uk	04/11/2020	-	-
000120484	9781907838101	1	mc081@college.ac.uk	04/11/2020	-	-

Second Normal Form Normalisation

Second Normal Form requires non-key columns to all be in First Normal Form and to be dependent on the entire primary key (not just part of a composite key) (Connolly and Begg, 2015). By applying Second Normal Form to the entities that we decided on based on the specification requirements, we find that:

Member (Email, Staff_ ID, Student_ ID, Given_Name, Surname, Resource_ Cap, Start_ Date, Finish_ Date)

This entity is in Second Normal Form already as all the attributes are in First Normal form already and are functionally dependent on Email.

Resource (Resource ID, #th_copy, Location, ISBN/EAN)

'Resource_Type', 'Title', 'Edition', 'Publication_Year', 'Loan_Period' are all dependent on 'ISBN/EAN' but not on '#th_copy' (the attributes are only dependent on half of the composite primary key). To avoid partial dependency, we will split the table into two tables-

‘Resource’ and ‘Resource Information’. Because ‘ISBN/EAN’ by itself does not uniquely identify a copy of a resource, ‘Resource ID’ (which can uniquely identify a copy of a particular resource) has been added to be the primary key instead.

Example:

Resource_ID	#th_copy	Location	ISBN/EAN
0023873451	1	0012	9781292061184
0023873452	2	0012	*9781292061184*
0023873453	3	0012	*9781292061184*
0023487238	1	1120	*9780198228589*
1023487682	1	0013	*9781907838101*

Resource Information (*ISBN/EAN*, Class_Number, Class_Name, Resource_Type, Title, Edition, Publication_Year, Loan_Period)

Example:

ISBN/EAN	Class_Number	Class_Name	Resource_Type	Title	Edition	Publication_Year	Loan_Period
9781292061184	0001	Computer Science	Book	Database Systems	6th	2010	1 week
9780198228589	0015	History	Book	British history, 1815-1906	1st	1991	1 week
9781907838101	0023	Foreign Language Study	CD	Chinese in steps. Student book 1	1st	2011	2 days
9787560076195	0023	Foreign Language Study	Book	Little Oxford English-Chinese Dictionary	9th	2015	1 week

Author (Author_ID, A_Given_Name, A_Surname)

This entity is in Second Normal Form already as all the attributes are functionally dependent on the primary key ‘Author ID’.

Resource–Author (*ISBN/EAN*, *Author_ID*)
This entity is in Second Normal Form already as all the attributes are functionally dependent on the primary key ‘ISBN/EAN’.

Publisher (Publisher_ID, Publisher)

This entity is in Second Normal Form already as all the attributes are functionally dependent on the primary key ‘Publisher ID’.

Resource-Publisher (*ISBN/EAN, Publisher_ID*)

This entity is in Second Normal Form already as all the attributes are functionally dependent on the primary key ‘ISBN/EAN’.

Loan (Loan_Ref#, *Resource_ID*, Email, Issue_Date, Return_Date, Date_Paid)
Example:

Loan_Ref#	ISBN/EAN	Resource_ ID	Email	Issue_Date	Return_Date	Date_Paid
000120482	9781292061184	0023873453	si.cheng@college.ac.uk	01/10/2020	03/11/2020	04/11/2020
000120483	9780198228589	0023487238	mc081@college.ac.uk	04/11/2020	-	-
000120484	9781907838101	1023487682	mc081@college.ac.uk	04/11/2020	-	-

‘Resource_ID’ has replaced ‘ISBN/EAN’ and ‘#th_copy’ as the foreign key in the ‘Loan’ table as this links the ‘Loan’ table to the ‘Resource’ table.
This entity is in Second Normal Form already as all the attributes are functionally dependent on ‘Loan_Ref#’.

Third Normal Form Normalisation

Third Normal form requires that all columns depend directly on the primary key (Connolly and Begg, 2015). A table violates Third Normal Form when a column depends on another column, which in turn depends on the primary key (this is a transitive dependency) (Connolly and Begg, 2015). By applying Third Normal Form to the entities that we decided on based on the specification requirements, we find that:

Member (Email, Staff_ID, Student_ID, Given_Name, Surname, Resource_Cap, Start_Date, Finish_Date)

This entity is in Third Normal Form already as all the attributes are non-transitively dependent on ‘Email’.

Resource (Resource ID, #th copy, Location, ISBN/EAN)

This entity is in third normal form already as all the attributes are non-transitively dependent on Email.

Resource Information (*ISBN/EAN*, Class_Number, Resource_Type, Title, Edition, Publication_Year, Loan_Period)
This entity is in third normal form already as all the attributes are non-transitively dependent on Email.

Here is an example demonstrating that each column only has atomic values:

ISBN/EAN	Class_Number	Resource_ Type	Title	Edition	Publication_Yea r	Loan_Period
9781292061184	0001	Book	Database Systems	6th	2010	1 week
9780198228589	0015	Book	British history, 1815-1906	1st	1991	1 week
9781907838101	0023	CD	Chinese in steps. Student book 1	1st	2011	2 days
9787560076195	0023	Book	Little Oxford English-Chinese Dictionary	9th	2015	1 week

Class (*Class Number*, Class_Name)

Class_Number	Class_Name
0001	Computer Science
0015	History
0023	Foreign Language Study

'Class_Name' was dependent on 'Class_Number' and 'Class_Number' dependent on 'ISBN/EAN'. Therefore, it was necessary to create a new entity for 'Class_Name' and 'Class_Number' (as we show here), with 'Class_Number' as a foreign key to link the Resource table to the newly created table.

Author (Author_ID, A_Given_Name, A_Surname)

This entity is in Third Normal Form already as all the attributes depend directly (non-transitively) on the primary key 'Author ID'.

Resource–Author (*ISBN/EAN*, *Author_ID*)

This entity is in Third Normal Form already as all the attributes depend directly on the primary key 'ISBN/EAN'.

Publisher (Publisher_ID, Publisher)

This entity is in Third Normal Form already as all the attributes depend directly on the primary key 'Publisher ID'.

Resource-Publisher (*ISBN/EAN*, *Publisher_ID*)

This entity is in Third Normal Form already as all the attributes depend directly on the primary key 'ISBN/EAN'.

Loan (Loan_Ref#, *Resource ID*, *Email*, Issue_Date, Return_Date, Date_Paid)

This entity is in Third Normal Form already as all the attributes are non-transitively dependent on the primary key ‘Loan_ref#’.

Conclusion

We now compare the result of normalisation with our relational schema.

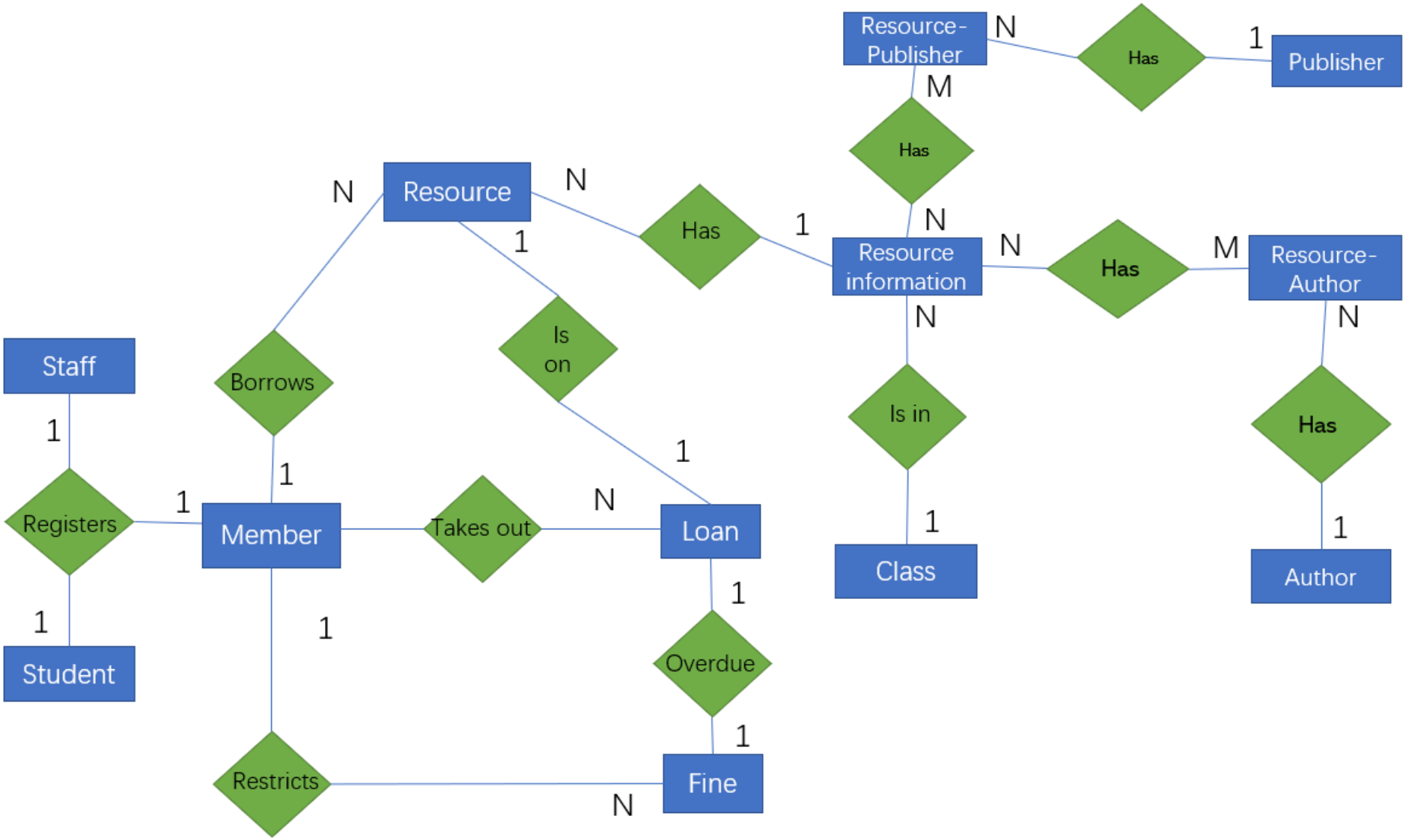
Member (Email, Staff_ID, Student_ID, Given_Name, Surname, Resource_Cap, Start_Date, Finish_Date)
Resource (Resource ID, #th copy, Location, ISBN/EAN)
Resource Information (*ISBN/EAN*, Class_Number, ResourceType, Title, Edition, Publication_Year, Loan_Period)
Class (*Class Number*, Class_Name)
Loan (Loan_Ref#, *Resource_ID*, *Email*, Issue_Date, Return_Date, Date_Paid)
Author (Author_ID, A_Given_Name, A_Surname)
Resource–Author (*ISBN/EAN, Author_ID*)
Publisher (Publisher_ID, Publisher)
Resource-Publisher (*ISBN/EAN, Publisher_ID*)

This schema would suggest that the entities 'Student', 'Staff', 'Fine', 'Loan History' are unnecessary to meet the system's basic requirements. 'Loan History' and 'Fine' could be combined into the entity 'Current Loan' (renamed 'Loan'). The information within the entities 'Student', 'Staff' and 'Member' could also be combined into one 'Member' entity.

Taking this into account, we decided to combine the 'Current Loan' and 'Loan History' entities as the advantage of having different tables would not be worth the possible data redundancy problems. The schema for our system would include:

Student (Student_ID, S_Surname, S_Given_Name, *Email*, S_Start_Date, S_Finish_Date)
Staff (Staff_ID, ST_Surname, ST_Given_Name, ST_Start_Date, ST_Finish_Date, *Email*)
Member (*Email*, Resource_Cap)
Resource (Resource ID, *ISBN/EAN*, Location, #th_copy)
Resource Information (ISBN/EAN, *Class_Number*, Resource_Type, Title, Edition, Publication_Year, Loan_Period)
Author (Author_ID, A_Given_Name, A_Surname)
Resource–Author (ISBN/EAN, Author_ID)
Publisher (Publisher_ID, Publisher)
Resource-Publisher (ISBN/EAN, *Publisher_ID*)
Class (*Class_Number*, Class_Name)
Loan (Loan_Ref#, *Email*, *Resource_ID*, Issue_Date, Return_Date)
Fine (*Loan_Ref#*, Date_Paid, *Email*)

Library staff would now not transfer the necessary information about a loan from Current Loan into Loan History at the point when a resource is returned. Our revised ER diagram would be:


Resource-Publisher
N
Has
1
Publisher
M
Has
N
Resource
N
Has
1
Resource information
N
Has
M
Resource-Author
1
N
N
N
Is on
Borrows
Is in
Has
Staff
1
1
1
1
Registers
1
Member
Takes out
N
Loan
1
Class
1
Author
1
1
Student
1
Overdue
1
Restricts
Fine
N

References

Harrington, J. (2016) *Relational Database Design and Implementation, Fourth Edition*. 4th edn. Burlington: Morgan Kaufmann.

Connolly, T. and Begg, C. (2015) *Database systems*. Boston: Pearson Education.

ISBN (2020) *What is an ISBN?* | *International ISBN Agency*, *Isbn-international.org*. Available at: https://www.isbn-international.org/content/what-isbn (Accessed: 5 November 2020).

GS1 (2015) *EAN/UPC - Barcodes* | *GS1*, *Gs1.org*. Available at: https://www.gs1.org/standards/barcodes/ean-upc (Accessed: 5 November 2020).