ECM2419 Database Theory and Design

Student number: 660050748

Continuous Assessment - Report

This report will highlight a library database design and justify the choices made in modelling and implementing that design. There will be three sections, each concerned with a different aspect of the database modelling process. In the first section, the requirement analysis is presented with detailed descriptions of the conceived requirements. The next section will have its attention aimed at the conceptual model of the database with thorough explanations of the included entity relationship diagram. Finally, the last section shows the conversion process from the conceptual model design to a logical model design.

Section A – User Requirement Specification

In this section, the report will describe the data requirements gathered through the requirement analysis process. The library database consists out of eight entities, branch, staff, librarian, manager, member, author, book, and bookcopy. Each entity will be covered in its respective section where its functions and position within the system are explained.

Branch

The library system includes multiple branches which are located throughout the UK. Each branch should have a catalogue of books of which it ought to have physical copies stored. The same book can be in the catalogue of multiple branches, but they can all have a different number of copies stored for that book. The book *1984* by George Orwell can be in the catalogue of both the Exeter and St Thomas branch but while there are ten copies in St Thomas, Exeter only has two. Each branch can only loan each individual copy to a single member because each copy is unique. If a user does not return the copy they borrowed, they will own a fine to that branch. As members can borrow multiple copies from multiple branches they can own fines to multiple branches as well. Additionally, each branch should be managed by one manager and have at least one librarian.

Staff

Branches need personnel to work for them. As there are different duties within a branch, staff ought to be a superclass with numerous subclasses of which a staff member only can assume one. Because things are to be kept relatively simple, only two subclasses – manager and librarian – exist. Staff members have to be either of those and can only work at one branch at a time.

Librarian

A librarian is a staff member who is in charge of administrative tasks as well as providing customer assistance. Librarians can only work at one branch at a time and are paid a fixed hourly rate.

Manager

Branches need to be managed by a member of staff which is the manager. A manager is paid a fixed salary and can only work at one branch. Moreover, branches can only have one manager.

Member

Any person who wishes to borrow a book from any of the library branches must be a member of the library. Members need to be uniquely identifiable and are able to borrow anything from zero to many copies from each branch. Although they may borrow copies of the same book from multiple branches, each individual copy can only be borrowed once and only from one branch. The borrowing process is recorded as a loan with the individual branch. Should a member not return a copy before the due date at the time of the loan, they will have to pay a fine. Additionally, as members can borrow multiple different copies from multiple branches they can owe fines to multiple branches.

Author

As books don't appear out of thin air but are written by someone, the database should also contain authors to make tracking down copies of books written by an author possible. Though this might be an odd requirement, authors don't need to have written any books as a library might want to add an author to the system of which they intend to add books in the future.

Book

Books are without doubt essential to any library system. In this case however, they should not be the actual copies which members can borrow but instead be part of the library catalogue. This is similar to the food menu one might find in a restaurant where different dishes are listed, all of which are abstract from the actual dish you will be served. Similarly, a library has a catalogue with books from which one can pick books which ought to be stored as copies in the library. As a result of this, a book might have zero or many copies associated with it at a given point in time. Finally, Books can have one or multiple authors.

BookCopy

The book copy represents the actual copy which will be borrowed by a member. Using the restaurant analogy from the previous section, this would be the actual dish served to a customer in the restaurant. Thus, each copy only exists at one point in space, i.e. it can only belong to one branch at a time and only be borrowed by one member at a time.

A.1 – Transaction Requirements

Data Entry

- Enter details of book, author, staff, bookcopy, branch, member.

Data update/deletion

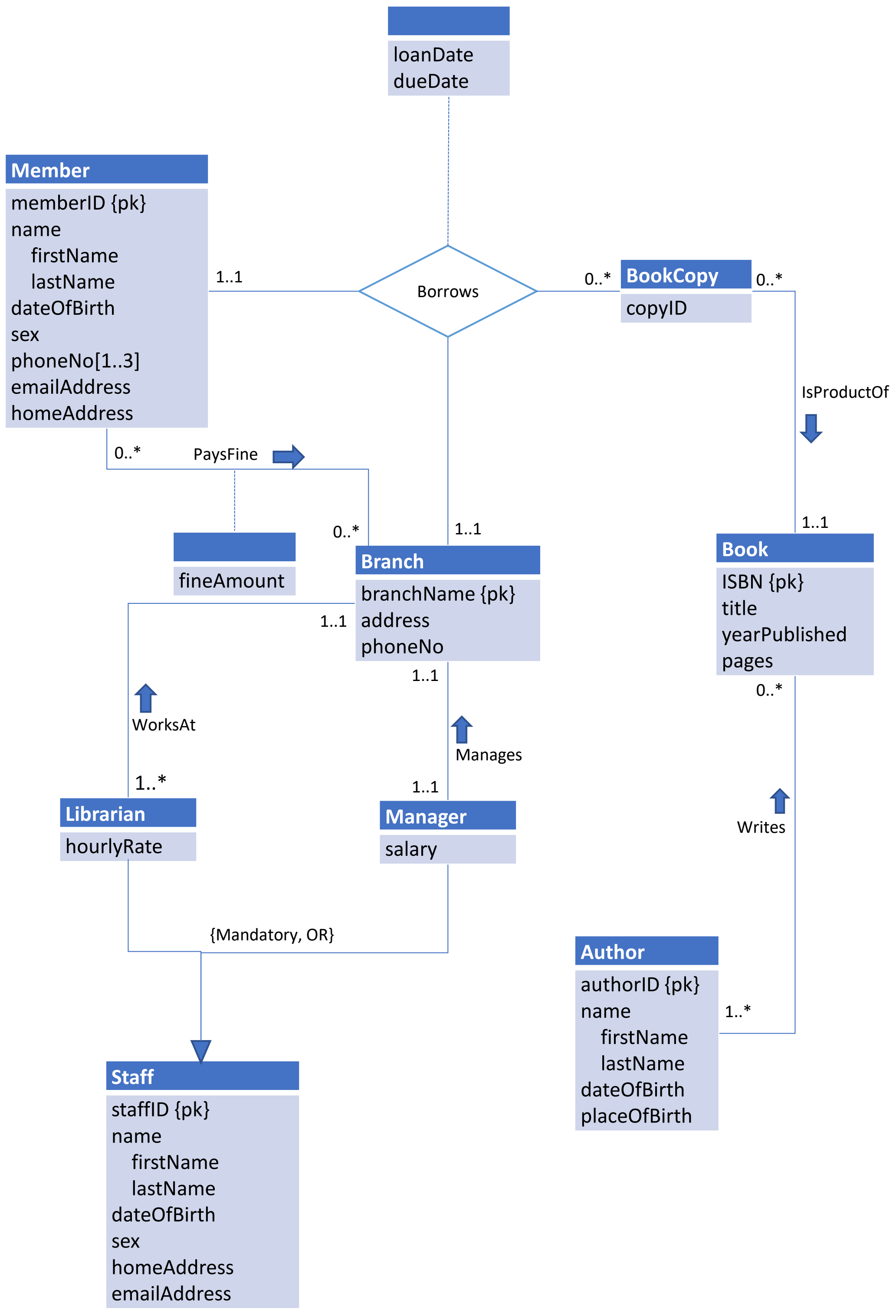
- Update/delete details of book, author, staff, bookcopy, branch, member.

Data queries

- List how many copies of books by George Orwell are held by each branch.
- List how many copies of the book *The Lost Tribe* are available at the St Thomas branch.
- List the names of all the borrower who haven't borrowed a book since Jan 1, 2017.
- For each book loaned out from St Thomas Library with the due date today, retrieve the book title, the borrower's name and the borrower's email address.
- List the book title, due date and borrowed date of overdue books from Ian Cooper.
- List the total fine of Ian Cooper.

Section B – Conceptual Model

The following section will explain how the user requirement specifications were transformed into an entity relationship model and justify the choice of attributes, multiplicity constraints, and relationships between the different entities.



Branch
Attributes: branchName (primary key), address, phoneNumber.

Branch only holds three attributes; the branch name which acts as the primary key, the address, and the phone number. It is assumed that each branch has a unique name by which it can be identified, e.g. St Thomas Library or Exeter Library, which is why it will act as a primary key. Branches also need an address so that they can be found on a map, and a phone number so customers can ask inquiries directly to a member of staff. The branch does not have a direct relationship with books because it only has the actual copies of them and can get the book's information through that.

Branch – Manager And Branch – Librarian Relationship
Entities involved: Branch, Manager, Librarian

Branches need staff to work for them, which is why it needs to have at least one staff member of each type work for them. However, while a branch can have multiple librarians working for them, it can only have one manager.

Branch – Borrower PaysFine Relationship
Entities involved: Branch, Borrower

Every now and then, a borrower will not return their book on time and receive a fine. Because a borrower can have multiple fines with multiple branches, e.g. one with St Thomas and three with Exeter, the relationship is many to many. Furthermore, to keep track of how much a fine was, a relationship attribute was added.

Borrows Relationship
Entities involved: Branch, Member, BookCopy

There are always three physical units involved in a book borrowing process. The book, the borrower, and the branch where it is borrowed from. As a consequence of the structure of this interaction, the decision to model this as a ternary relationship was made. The cardinality is so that each copy of a book can only be borrowed by one borrower from one branch. By fixing the branch and book copy entity first, i.e. a branch can have zero to many copies of the same book with each copy belonging to only that one branch, the cardinality of the borrower in this relationship becomes one to one. As this relationship inevitably creates new data, i.e. information about the loan itself so that it may be tracked, a relationship attribute was added containing the loan date and the date the book is to be returned.

Member
Attributes: memberID (primary key), name (composite: firstName, lastName), dateOfBirth, sex, homeAddress, emailAddress, phoneNumber (min: 1, max: 3).

The member entity holds multiple attributes. It is identified by its unique member ID which acts a primary key. This is to circumvent any issues which might arise from using a composite primary key of attribute values which could be found in other entities as well, e.g. name and address. Additional personal information about the member can be found in all the other attributes. I.e. the composite attribute of name, sex, date of birth, phone number, email address, and home address. Note that the phone number is a multivalued attribute which must contain at least one and at most three values.

Author
Attributes: authorID (primary key), name (composite: firstName, lastName), dateOfBirth, placeOfBirth.

Authors are uniquely identified by an author ID given to them by the library. The reasoning for this is similar to providing an individual member ID. Because there might be authors with the same first and last name, giving each of them an ID bypasses the issue of not being able to tell apart two entity occurrences. Aside from their composite name attribute, authors also have their date of birth, sex, and place of birth recorded. Including the place of birth allows to track down authors from the same country and categorise them.

Book
Attributes: ISBN (primary key), title, pageCount, yearPublished.

Although books are created by an author, they are not a weak entity because they should still be identifiable through a primary key. This decision is mostly due to the fact that books in the real world have a unique identifier called ISBN (International Standard Book Number) by which they can be distinguished from other books. Again, this is to avoid confusion which would arise from using any of the other attributes as primary keys as they can be found in other entity occurrences as well.

Author – Book Relationship
Entities involved: Author, Book

Authors write books, so the existence of this relationship is fairly self-explanatory. A book needs to have at least on author because someone has to write it, i.e. it has a mandatory participation. Nevertheless, as authors collaborate, the cardinality for book is many. The other way around, authors have optional participation because not all authors might have books in the database. The cardinality remains many however.

BookCopy
Attributes: copyID

The book copy is a weak entity with only one attribute because it cannot exist separately from the book entity. Nonetheless, it has a copy ID to be able to identify it once it comes into existence.

BookCopy – Book Relationship
Entities involved: BookCopy, Book

In this binary relationship, the book copy has a mandatory participation with a one cardinality. This is as a consequence of each copy only belonging to one single book. Thinking about how this would work physically makes it more straightforward. It simply is not possible for a physical copy to be a copy of two books simultaneously. The book on the other hand, has an optional participation and a cardinality of many as no copies might exist at a point in time.

Staff
Attributes: staffID (primary key), name (composite: firstName, lastName), dateOfBirth, sex, homeAddress, emailAddress.

The staff entity is a superclass to both the librarian and the manager entity. Making it a superclass was deemed a reasonable decision as both librarian and manager effectively are staff members of a branch and thus share a great number of attributes. The participation constraint imposed is a mandatory one because those should be the only two roles a staff can assume in this model. It is also disjoint so that a manager cannot be a librarian and a librarian cannot be a manager. Staff are identified by their unique staff ID but also have similar attributes as a member, e.g. date of birth, name etc.

Librarian
Attributes: hourlyRate

A librarian has only one attribute besides those inherited from the staff entity, namely the hourly rate which they are being paid.

Manager
Attributes: salary

The manager also only has one additional attribute which is the salary they are being paid.

Section C – Logical Model

The last and final section of this report will shed light onto the transformation from entity relationship to relational model by showcasing the different normalising procedural stages which were taken.

Branch

Due to the one to one relationship between branch and manager, the two entities will be merged into one with branch receiving the additional attributes of managerID and managerSalary. The managerID will reference the staff relation as the manager entity was a subclass of staff. This is not the only foreign key however, as the branch needs an address so that people can find out where it is. Having a separate relation for address helps with update and deletion anomalies as multiple people can live at the same address and it will not have to be reinserted if a tuple is deleted.

Branch(branchName, addressID, phoneNo, managerID, managerSalary)
Primary Key: branchName
Foreign Key: addressID references Address(addressID)
Foreign Key: managerID references Staff(staffID)

branchName	addressID	phoneNo	managerID	managerSalary
Exeter Library	1000	01392 723867	00000000001	70000

Author

The author entity can directly be translated into the relational model by transferring all of the attributes present.

Author(authorID, firstName, lastName, dateOfBirth, placeOfBirth)
Primary Key: authorID

authorID	firstName	lastName	dateOfBirth	placeOfBirth
749195	George	Orwell	1903-06-25	India

Book

Book is just as easily translated as author, all of its attributes are directly placed in the relation.

Book(ISBN, title, yearPublished, pages)
Primary Key: ISBN

ISBN	title	yearPublished	pages
978-0-14139-304-9	1984	1950	336

Book – Author Relationship

Due to the many to many relationship between book and author, it is necessary for a third relation to link these two together. This relation will only contain the primary keys of book and author as foreign keys.

BookInvolvement(ISBN, authorID)
Primary Key: ISBN, authorID
Foreign Key: ISBN references Book(ISBN),
Foreign Key: authorID referneces Author(authorID)

ISBN	authorID
978-0-14139-304-9	749195

Bookcopy

A result of the zero to many relationship between BookCopy and Book, the child entity (BookCopy) will receive a foreign key of Book. It also has a zero to many relationship with branch and will therefore receive the primary key of branch as another foreign key. However, as a branch could have multiple copies of the same book, these two attributes wouldn't be sufficient to act as primary keys. Henceforth, each copy will be uniquely identified by its own id.

BookCopy(copyID, ISBN, branchName, available)
Primary Key: copyID
Foreign Key: ISBN references Book(ISBN)
Foreign Key: branchName references Branch(branchName)

copyID	ISBN	branchName	available
8	978-0-14139-304-9	Exeter Library	N

Fine

Although one could include a fine as an attribute in the loan relation, keeping them separate is of interest for the library as it probably wants to track when a fine was issued and when it is to be paid. This would mean additional fields in the loan table which might not even be populated (a fine isn't issued until the due date has passed and the item has not been returned). To reduce any overhead caused by this, a separate table is the better alternative. This can obviously be deduced from the ER diagram and the many to many relationship between member and branch, but further elaborating it here is sensible. Each fine is identified by a unique id and references the branch and member table.d

Fine(fineID, memberID, branchName, copyID, fineAmount, finedOnDate)
Primary Key: fineID
Foreign Key: memberID references member(memberID)
Foreign Key: branchName references branch(branchName)

fineID	memberID	branchName	copyID	fineAmount	finedOnDate
012345	000000001	Exeter Library	8	20.00	2016-05-31

Loan

The ternary relationship between member, branch and bookcopy requires an additional relation to properly function. This relation will have three foreign keys referencing each of the pariticpants in that relationship. Though it would be possible to use a composite primary key made of loanDate, dueDate, and copyID, this was not implemented so that the table could be in a third normal form. As the library mighr want to build a history of loans, a binary returned field was added.

Loan(loanID, loanDate, dueDate, copyID, memberID, branchName, returned)
Primary Key: loanDate, dueDate, copyID
Foreign Key: copyID, memberID, branchName

loanID	loanDate	dueDate	copyID	memberID	branchName	returned
4	2016-03-01 00:00:00	2016-05-01	8	0000000001	Exeter Library	N

Member

For the member entity, all of its attributes can directly be transferred to the relational model. Much like the Branch relation, it will have a foreign key referencing an address in the address table.

Member(memberID, firstName, lastName, dateOfBirth, sex, emailAddress, addressID)
Primary Key: memberID
Foreign key: addressID references Address(addressID)

memberID	firstName	lastName	dateOfBirth	sex	emailAddress	addressID
0000000001	Ian	Cooper	1993-10-09	M	iancooper@gmail.com	1002

MemberPhone

Unlike the staff or branch entity, members can have up to three phones registered in the database. These need to be placed in a separate relation to avoid data redundancies and unused space in the member entity. Because this is essentially a one to many relationship between MemberPhoneNo and Member, the child entity (MemberPhoneNo) will receive the member primary key as a foreign key.

MemberPhoneNo(phoneNo, memberID)
Primary Key: memberID, phoneNo
Foreign Key: memberID

phoneNo	memberID
0756233485	000000001

Address

The reason for having an address relation was already mentioned in the Branch description. It prevents update, insertion, and deletion anomalies. If two people live at the same address because they are flatmates or members of the same family, the member and staff relations would have a redundant data causing the aforementioned anomalies. The separate address relation allows multiple tuples to reference the same address.

Address(addressID, streetName, houseNo, city, zipCode, county, country)
Primary Key: addressID

addressID	streetName	houseNo	city	zipCode	county	country
1001	Magdalen Drive	23	Derby	DE2 3SU	DerbyShire	UK

Staff

The staff entity will keep all of its attributes and receive the addressID as a foreign key.

Staff(staffID, firstName, lastName, dateOfBirth, sex, addressID, emailAddress)
Primary Key: staffID
Foreign Key: addressID (address)

staffID	firstName	lastName	dateOfBirth	sex	addressID	emailAddress
0000000001	Lisa	Turner	1980-07-10	F	1006	lisaTurner@gmail.com

Librarian

Just like Manager, Librarian receives the primary key of staff as its primary key and foreign key.

Librarian(staffID, hourlyRate)
Primary Key: staffID
Foreign Key: staffID (staff)

staffID	hourlyRate
0000000003	12.50

Appendix

SQL code only submitted online as requested in email received on 21/11/2017 15:08.