Homework 1 By Sara Rydell and Filip Northman

Required tasks (P)

Task 1

Motivation

We have included all data under Database Data Requirements from the study case and we have also included attributes expected to have multiple values as instructed. We have designed the schemas below using the example format given in the Database Data Requirements since we think that logic is suitable as a basis for the database which is why we have created the separate schemas Books, Loans, Students, Admins, Fines, and Transactions. We also created schemas for the relations Borrows, Lends, Makes, Pays and Creates that we see as logical in our diagram design. We have then underlined the primary keys of each schema.
All attributes that usually are represented as numbers in all of the following schemas, we choose to give the domain type integer. The attributes usually in the format of text or text combined with numbers gets the domain type string. The attributes containing the word date are of the domain type date. Some of the attributes have the properties of being true or not, and therefore we choose to give them the domain type boolean.
To better get a better overview of our work we have chosen to temporarily color code the domain types noted in the schemas: integer-red, string-green, date-blue, and boolean-purple.
For now we assume all ID attributes for the schemas to be of domain type integers only, although we do consider that ID's sometimes consist of characters and numbers. Therefore we take into consideration that we might need to change the domain types of some of the ID attributes later on.
In the Important notes part of the study case the Physical ID, Borrowing ID, User ID and Transaction ID were also pointed out as attributes only referring to one single physical book, borrowing, user or transaction respectively. Thus they will be the primary keys for the schemas. They will therefore also be foreign keys of other schemas where they are used but aren't the main identifying attribute.

Schemas for tables:

Schema for required Information about books:
Books(Physical ID:integer, Title:string, ISBN:integer, Author:string, Edition:integer, Genre:string, Language:string, Publisher:string, Date of publication:date, Prequel:string, Damage:boolean)

For now we assume the attribute Prequel to be a string so that the prequel of a book will be saved as the title of the previous books in its book series, if it's a sequel. We could also have alternative solutions, for instance that it could be a boolean value or use an ID for the prequel book, although using the book title seems for now to be the most logical solution since there aren't any general IDs for each book (not being physical copies).

The attribute Damage can be a boolean value which we choose for now, although it could also be a text description of the damage or maybe consist of a scale of how damaged the book is. For now we don't know how this information will be used so it's a first suggestion.

Schema for required Information about borrowing and returning a book:
Loans(Borrowing ID:integer**,** Physical ID:integer, User ID:integer, Date of Borrowing:date, Due Date:date, Date of Return:date)

We choose to call this entity Loans since it pretty much sums up the action of borrowing and returning books.

Schema for required information about users:
User(User ID:integer, Full name:string, Email:string, Address:string)
Schema for required information about students:
Students(User ID:integer, Programme:string,)
Schema for required information about admins:
Admins(User ID:integer, Department:string, Phone Number:integer)

We decided to make the schemas Students and Admins into two subset schemas of the "mother"-subset schema User as a solution. We did this since the two schemas Students and Admins have many of their attributes in common and only had a few additions each. Subset schemas are also supposed to be a good solution to use when we want to give the subsets different authority and roles in the system, as mentioned during our lectures, and because of this we think it's a good solution for future use.
We here use the format where the subset schemas include the primary key to connect it to the super class as well as the attributes specific for each subset schema.

Schema for required information about fines:
Fines(Borrowing ID:integer, Amount:integer)

This is a weak entity since this entity set doesn't have a primary key. In this case Transactions is the identifying entity set and the discriminator (partial keys) are the attributes Borrowing ID and Amount.

Schema for required information about Transactions:
Transactions(Transaction ID:integer, Borrowing ID:integer, Amount:integer, Date of payment:date, Payment method:string)

Schemas for relations:
Pays(Transaction ID:integer, Amount:integer)
Creates(Borrowing ID:integer, Due date:date)
We chose to include Borrowing ID and Due date in the relation that creates the fine since those parameters should be used to know if a user should pay a fine. The rest of the relations come more natural since it is the primary keys of respective entity sets that make up the relations.

Task 2

a) **For the entity sets:**
The schema for Books has no foreign keys
The schema for Loans has the foreign keys Physical ID and User ID,
The schema for Users has no foreign keys
The schema for Students has the foreign key User ID
The schema for Admins has no foreign keys has the foreign key User ID
The schema for Fines has the foreign key Borrowing ID
The schema for Transactions has the foreign key Borrowing ID

For the relations:
The schema for Pays has the foreign key Transaction ID
The schema for Creates has the foreign key Borrowing ID

What schema the foreign keys references:
Physical ID references the schema Books.
Borrowing ID references the Loans.
User ID references the schemas Student and Admins.
Transaction ID references the schema Transactions.

b) For all the entity sets above we think that they could not have referred to other primary keys since the actions between them are very specific.
For Borrows we must keep the User ID to identify the person who borrows the book, although we probably could have referred to the book that is borrowed in different ways. For instance by using the Physical ID instead of the Borrowing ID, although in this solution we wouldn't connect this action to a loan which would be problematic.
For Lends we must keep the Physical ID to identify which book has been lended, although we can connect the loan to the users in different ways. For instance the User ID although then we haven't connected it to the loan itself.
For Makes we must keep the User ID but could probably make the connection to the transaction in another way. For instance connect directly to the weak entity set Fine and then connect it to the transaction. This wouldn't in our opinion have been as logical as our current solution.
For Pays we could have followed the planning as in the previous example, or maybe even connect it directly to the user with the User ID, although in these solutions not having a clear logic with how fines will be calculated and transactions should be made.
For Creates we must keep the Borrowing ID since it is from the loan details that we create a fine (if the user hasn't for instance, returned a book in time) and then it should be connected to how the user will pay and what amount which can be solved in different ways. For instance by connecting it directly to the Transaction ID although in this solution it would be difficult to differentiate the transaction from the fine.

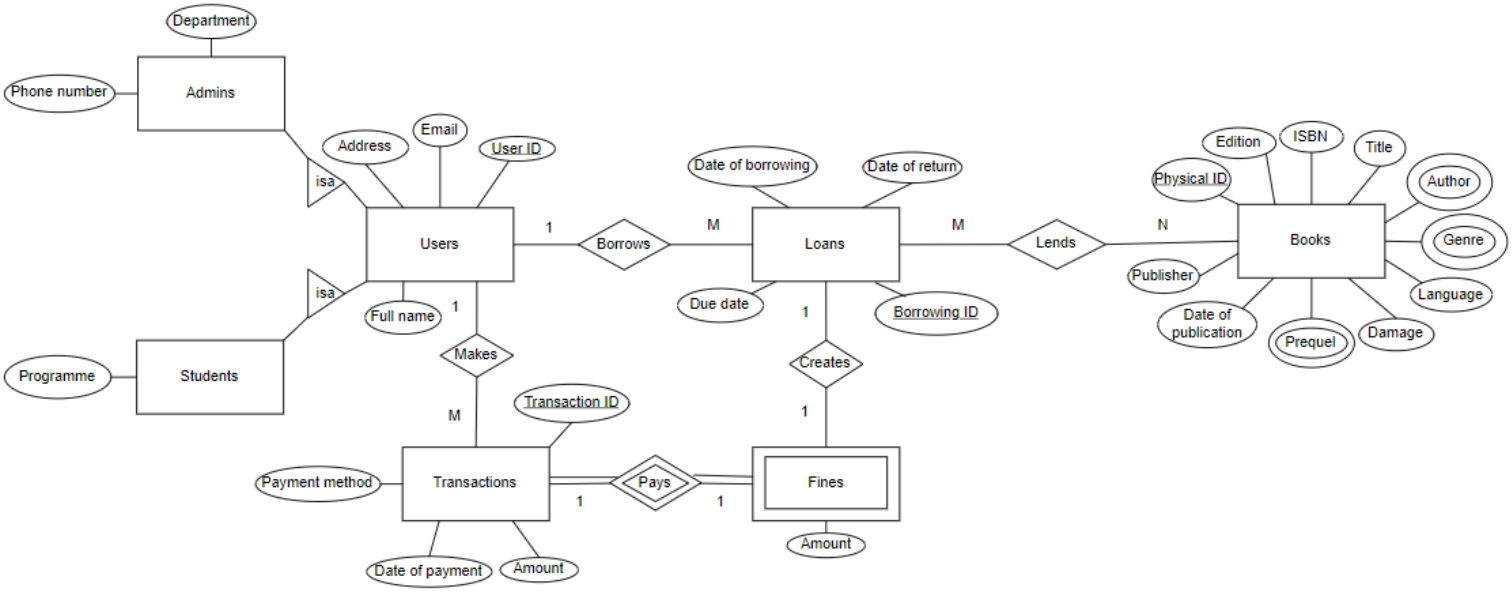
c) All attributes can't be primary keys since all of them aren't a specific enough identifier for the entities. For instance the attributes genre isn't enough information to identify a single

book in the library as the Physical ID does. We also want to have as few primary keys as possible for clarity in the design.

Task 3

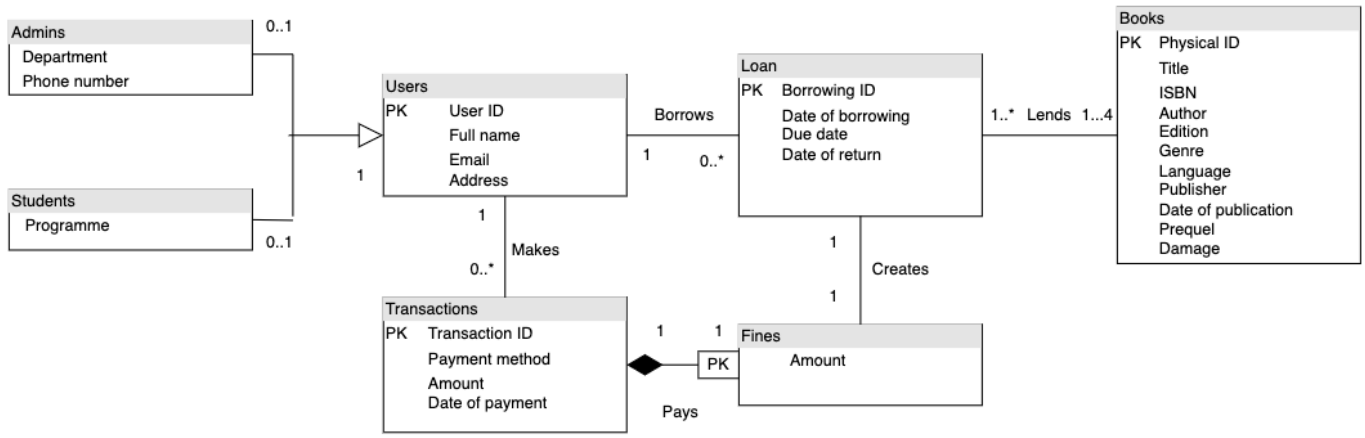
a) **Cardinality/multiplicity**
   Users-Loans is one-to-many, since one person can borrow at the library multiple times.
   Loans-Books is many-to-many, since you can make many loans while borrowing many books.
   Users-Transactions is one-to-many, since one person could have to pay multiple fines.
   Transactions-Fines is one-to-one, since one transaction pays off each fine.
   Loans-Fines is one-to-one, since one fine is created from each loan.
b) **Weak entities**
   Fines is a weak entity set since it doesn't have a primary key.
c) **Relations**
   Users and Loans are related through borrows by their primary keys since users loans from the library.
   Loans and Books are related through Lends by their primary keys since books are lended by the library.
   Users and Transactions are related through Makes by their primary keys since users might have to pay to the library.
   Transactions and Fines are related through Pays transactions pay for the fines.
   Fines and Loans are related through Create since fines are created if the user doesn‘t return their books in time.

A physical book may have multiple authors, genres and prequels that's why these attributes are marked as multivalued in the diagram. For Users we designed the Students and Admins subclasses to Users by using isa relations. Weak entity sets are visualized using the double lined rectangle.


Department
Admins
Phone number
isa
Address
Email
User ID
Users
Full name
isa
Students
Programme
1
Borrows
M
Date of borrowing
Date of return
Loans
Due date
Borrowing ID
M
Lends
N
Edition
ISBN
Title
Physical ID
Author
Books
Genre
Publisher
Language
Date of publication
Prequel
Damage
1
Makes
M
Transaction ID
Payment method
Transactions
Date of payment
Amount
1
Pays
1
Creates
1
Fines
Amount

Tasks (P+)

Task 1



This is our equivalent design of the ER-diagram in UML.

Task 2

Multiplicity:
Between admins and users we chose 0..1 to 1 because the user doesn't have to be an admin and therefore null, but could be 1 as the user is an admin. Same thing goes for students.

Between users and transactions we chose 1 to 0..* because there is always 1 user, then if the book is returned in time there will be no transaction, although if it is late there will be a transaction added and a user can be given many fines to pay.

Between transactions and fines we chose 1 to 1 because there will only be one transaction per fine, we thought about the possible relation many-to-one since Klarna is a payment option too, but down payments are usually not used in these contexts so it's not logical to have this multiplicity.

Between loan and fines we chose 1 to 1 as one loan will only create one fine. There could be the case that if you loan several books on one day they could have different loan times but that seems weird.

Between loan and books we chose 1..* to 1..4. We chose 1..* is because a single user can have several loans, as in they borrow books on multiple occasions while they already have borrowed

books. We chose 1..4 as you have to borrow at least 1 book in a loan but the maximum number of books you can borrow is 4 as the detailed requirements in the case study says.

Relationship:
Between users and admins/students, users inherit admin/students as they are subgroups of users.

Between user and transaction there is an association as the user makes a transaction.
It is an association as the relationship is connected with primary keys by an action. This is the case for all associations.

Between transactions and fines there is a composition as transactions is dependent on fines and cannot exist without it. This is because what is there to transact without a fine? That is if we assume having a library card is free, which it should be.

Between user and loans there is the association, Borrows as the user borrows/loans.

Between loan and fine there is the association creates as the loan creates a fine if the book isn't returned in time.

Between loan and books there is the association lends as books get lended in a loan.