Q1- Data modelling

Provide an UML diagram for the data described hereafter. Use the notation presented in class. If some aspects of the application seem unclear to you, make a choice and explain it in your answer.

Let's consider information about artists and their music. Artists can be either musicians or singers. Artists are identified by their name and we know his/her hometown and homepage URL. A song is sung by one single singer and played by one or several musicians. Albums have a unique title and are produced by a single record company. We know the year when the album has been recorded and the songs it contains. A song is identified by its name and has a length. Finally, a record company is identified by its name, we know its address, phone number and homepage. All companies have produced at least one album.

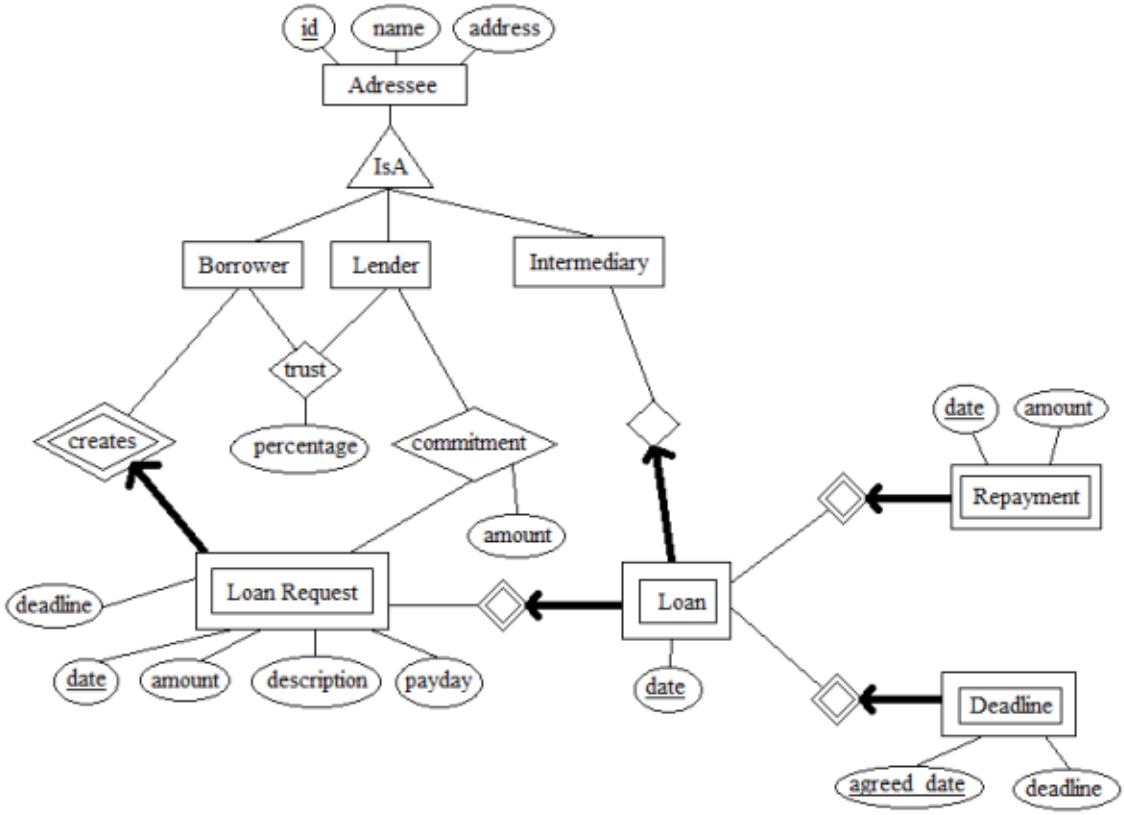
1 Data modeling

Micro loans are small loans, which is beginning to gain popularity especially among borrowers in developing countries. The idea is to bring venture lenders together using information technology. Typically, the loans will be used to finance startup or development of the borrower's company, so that there is a realistic chance for repayment. The money in a loan can, unlike traditional loans, come from many lenders. In this problem, you must create an E-R model that describes the information necessary to manage micro loans. The following information form the basis for creating the model:

- Each borrower and lender must be registered with information about name and address.
- A loan starts with a loan request, which contains information about when the loan should at latest be granted, The total amount being discussed (US-dollars), and how long the payback period is. Also, a description is included of how the money will be used. The rent on the payment is calculated in the loan amount, which is to say, the full amount is not paid .
- Lenders can commit to an optional portion of the total amount of a loan request.
- When the commitments for the loan request covers the requested amount, the request is converted to a loan. If not enough commitments can be reached, the loan request is cancelled. A borrower can have more than one request, and more than one loan at a time, but can at most make one request per day.
- The loan is paid through an "intermediary", typically a local department of a charity, who has a name and an address.
- The borrower chooses when he or she will make a payment. Every payment must be registered in the database with an amount and a date (at most one payment per loan per day). The lenders share the repayment based on how large a part of the loan they are responsible for.
- If the loan is not repaid before the agreed upon deadline, a new date is agreed. The database must not delete the old deadline, but save the history (the deadline can be overridden multiple times).

a) Make an E-R model for the data described above. If you make any assumptions about data that doesn't show from the problem, they must be described. Use the E-R notation from KBL. Put an emphasis on having the model express as many properties about the data as possible, for instance participation constraints.

Example answer:



b) Make a relational data model for micro loans:

- Describe at least **two** of the relations using SQL DDL (make reasonable assumptions about data types), and
- state the relation schemas for the other relations.

The emphasis is if there is a correlation between the relational model and the E-R diagram from a), along with primary key and foreign key constrations being stated for all relation. It is *not* necessary to state CHECK constraints and the like.

Example answer: It is assumed, that borrowers, lenders, and intermediaries are disjoint entities. Below MySQLs ENUM type is used, which is not part of the syllabus.

```
CREATE TABLE Adressee (
```

```
  id INT PRIMARY KEY,
  type ENUM('borrower', 'lender', 'intermediary'),
  name VARCHAR(50),
  address VARCHAR(50)
);

CREATE TABLE Trust (
  borrower INT REFERENCES Adressee(id),
  lender INT REFERENCES Adressee(id),
  percentage INT,
  PRIMARY KEY (borrower,lender)
);

CREATE TABLE LoanRequest (
  id INT REFERENCES Adressee(id),
  date DATE,
  amount INT,
  description VARCHAR(1000),
  payday DATE,
  deadline DATE,
  PRIMARY KEY (id,date)
);

CREATE TABLE Commitment (
  lender INT REFERENCES Adressee(id),
  borrower INT,
  loanrequestDate DATE,
  FOREIGN KEY (borrower,loanrequestDate) REFERENCES LoanRequest(id,dato),
  amount INT,
  PRIMARY KEY (lender, borrower, loanrequestDate,amount)
);

CREATE TABLE Loan (
  id INT,
  RequestDate DATE,
  date DATE,
  intermediary REFERENCES Adressee(id),
  FOREIGN KEY (id,RequestDate) REFERENCES LoanRequest(id,date),
  PRIMARY KEY(date,id,RequestDate)
);
 CREATE TABLE Repayment (
   id INT,
   date DATE,
   RequestDate DATE,
   amount INT,
   FOREIGN KEY (id,RequestDate) REFERENCES LoanRequest(id,date),
   PRIMARY KEY (date,id,RequestDate)
 );
```

```
CREATE TABLE Deadline (
  id INT,
  agreedDate DATE,
  RequestDate DATE,
  deadline DATE,
  FOREIGN KEY (id,RequestDate) REFERENCES LoanRequest(id,date),
  PRIMARY KEY (agreedDate,id,RequestDate)
);
```

Q2- Querying a relational database

Let's consider two relations about Vehicles and their manucfacturers :

Vehicle(VehicleID, YearOfManufacture, Model, ManufacturerCode)
Manufacturer(ManufacturerCode, ManufacturerName)

Express the following queries in relational algebra and in SQL.
Answer in the space provided after each question, and hand this sheet in.

1) Give the model of the vehicles manufactured by Renault.

2) Give the code and the name of the manufacturer of the vehicles manufactured in 2012.

3) Give the code of the manufacturers without vehicles manufactured in 2012.

4) Give the code of the manufacturers having at least two different vehicle models.

Q3- Normalization

1. **Find the highest normal form in R.**

   R(A,B,C,D,E) FD={A->BCDE, BC->ACE, D->E}

2. **A car rental firm wants to use a relational DBMS to manage its vehicle fleet and rentals. It includes the following relation:**

   RESERVATION (CarNbr, CustomerName, Addr, DepartureDate, Length)

The known functional dependencies are:
F= {CustomerName --> Addr,
CarNbr, DepartureDate -> CustomerName, Length}

A reservation states the date DepartureDate on which a car CarNbr is reserved for a customer CustomerName for a time lapse Length. Each customer has one single address Addr. A vehicle can be reserved only once for a given departure date.

1. *What is (are?) the key of the relation RESERVATION?*

2. *What is the normal form of the relation RESERVATION?*

Q4- SQL

`Repayment(borrower_id,name,address,loanamount,requestdate,repayment_date,repayment_amount)`

To solve the problem, the information from the description in problem 3 must be used.

a) Write an SQL request that returns all the tuples with information on repayments from the borrower with `id` equal to 42, and where the lent amount exceeds 1000 USD.

Example answer:

```
SELECT *
FROM Repayment
WHERE borrower_id=42 AND loanamount>1000;
```

b) Write an SQL request that for each address finds the total repaid amount for the address.

Example answer:

```
SELECT address, SUM(repayment_amount)
FROM Repayment
GROUP BY address;
```

c) Write an SQL request that finds all names which has a unique address, which to say is where there does not exist a tuple with a different name and same address.

Example answer:

```
SELECT name
FROM Repayment A
WHERE 1=
   (SELECT COUNT(DISTINCT name)
    FROM Repayment B
    WHERE A.address=B.address);
```

d) Write an SQL command, which deletes all information on ended loans, which is to say loans where the total repaid amount equals the lend amount.

Example answer:

```
DELETE FROM Repayment A
WHERE loanamount=
   (SELECT SUM(repayment_amount)
    FROM Repayment B
    WHERE B.borrower_id=A.borrower_id AND B.requestdate=A.requestdate);
```

Q5 – Transactions

Two sqlplus sessions are launched: S1 and S2. The following commands are executed on session S1:
S1: set autocommit off;
S1: create table R1 (a int, b int);
S1: insert into R1 values (1,2);
S1: insert into R1 values (1,3);
S1: insert into R1 values (2,3);
S1: commit;
S1: rollback;
*S1: select * from R1;*
- What is the result of the last command?