Database Systems I
CS3431
C-2018

Solution of Homework 1

Problem 1
Design Assumptions

- In this diagram, the “Branch” entity set is weak because the branch ID is unique only within the bank.

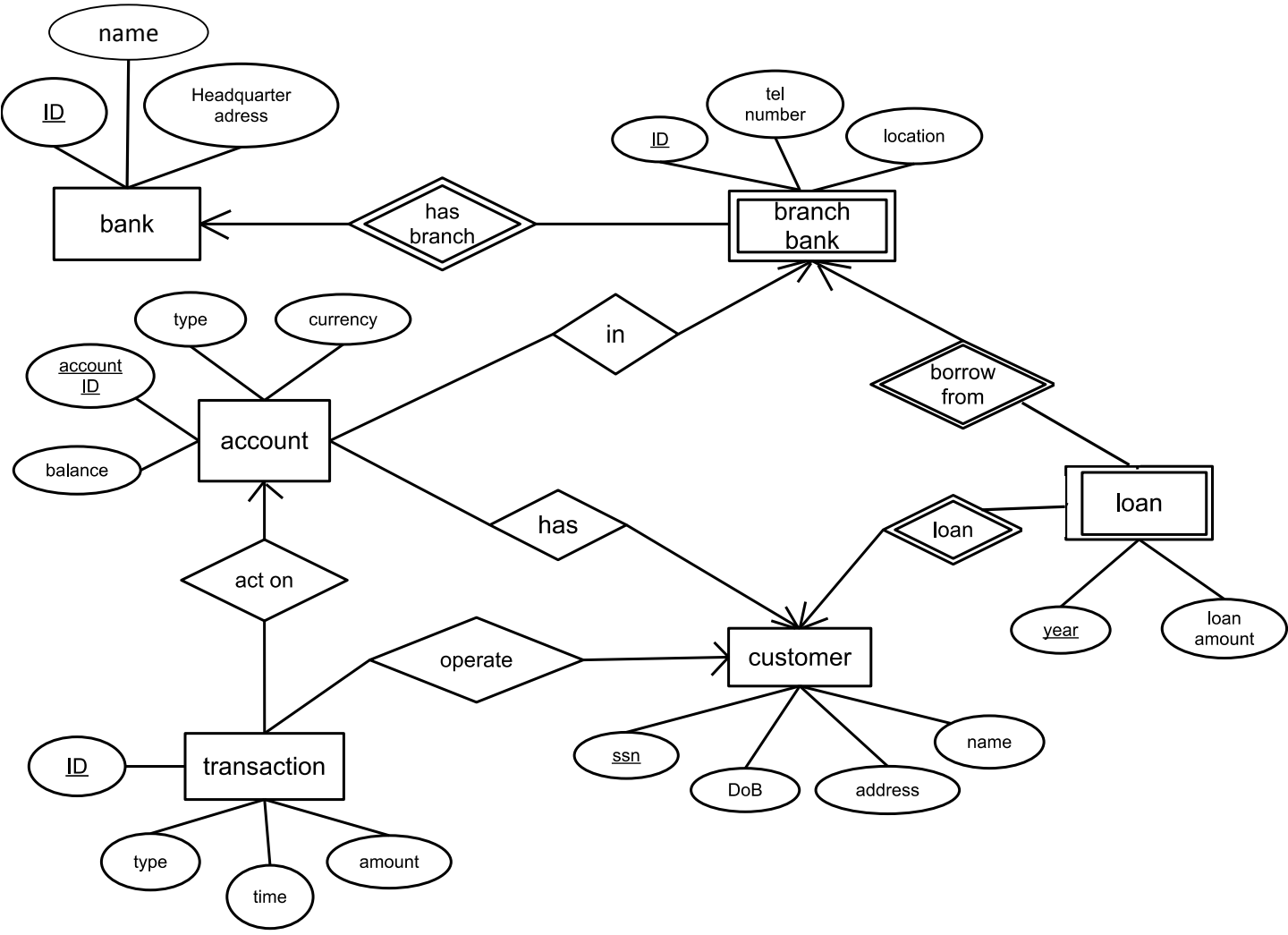
- For a transaction, we introduced an new attribute “ID” which we assume to be unique, and in this case the “Transaction” entity set is a strong one. If you model the transaction as a many-to-many relationship that is also correct, but the transaction’s time should be part of the key.

- In The loan weak entity set, we consider the “year” as part of the key. This means that the primary key for this weak entity set will be (Branch ID, Bank ID, customer SSN, and Year). In this case, a customer cannot take two loans from the same branch at the same year.

- It is also valid that you model “Loan” as a many-to-many relationship between “Customer” & “Branch”. In this case it will be called something like “Lend”. This relationship will have two attributes “year” & “amount”. And again “year” must be marked as part of the key.

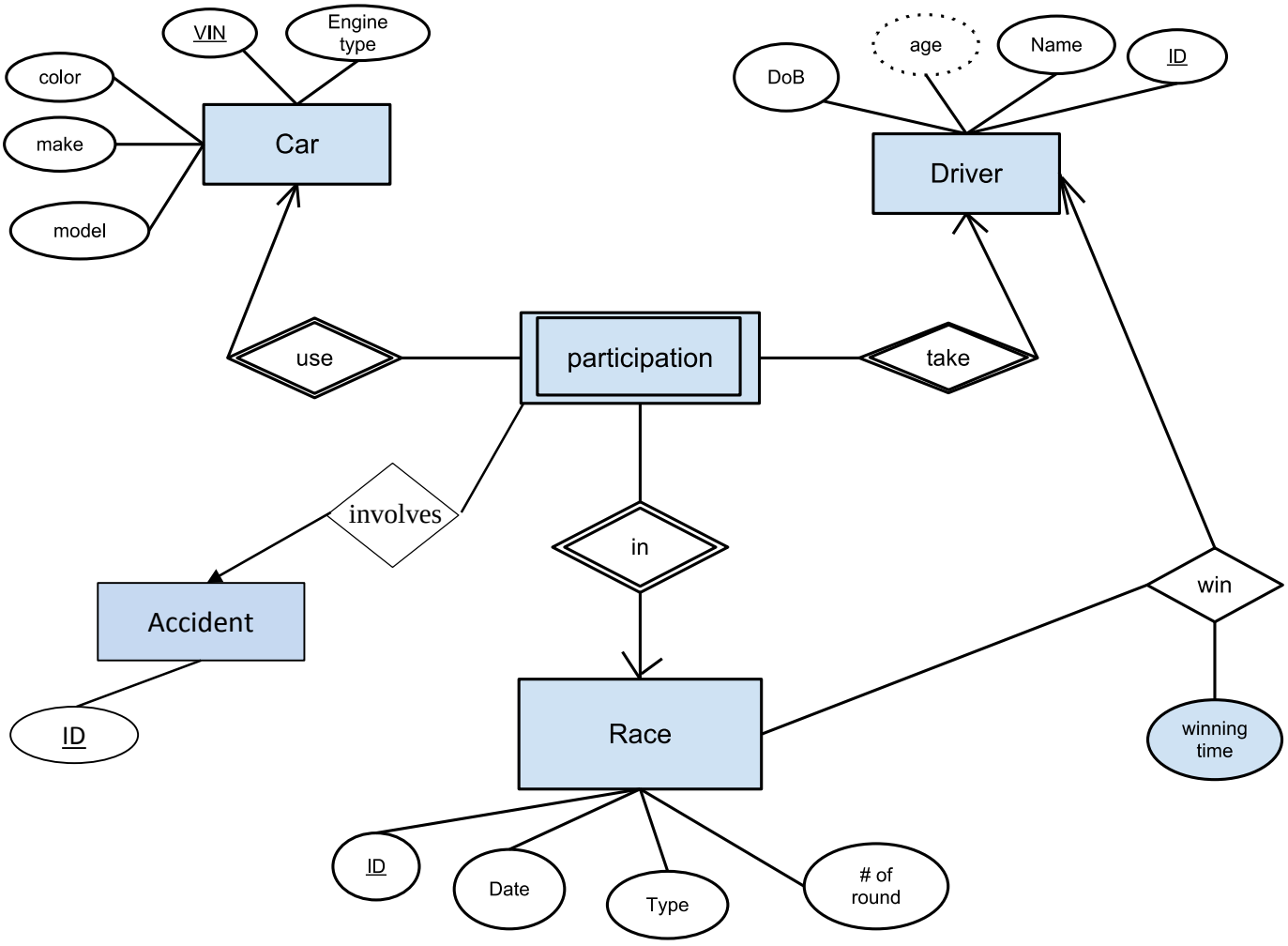
- For transactions, In the given design we assume that a customer may make a transaction on accounts not belonging to that customer, e.g., Customer X will deposit money in Customer’s Y account. In this case, the “operate” relationship is needed.
  - If in your design, you assume only the customer that owns an account can do a traction on it, then in this case, no need for “Operate” relationship. This is because the customer is already known from the “Has” relationship. Now the “operate” becomes a redundant one.

Problem 1
ERD



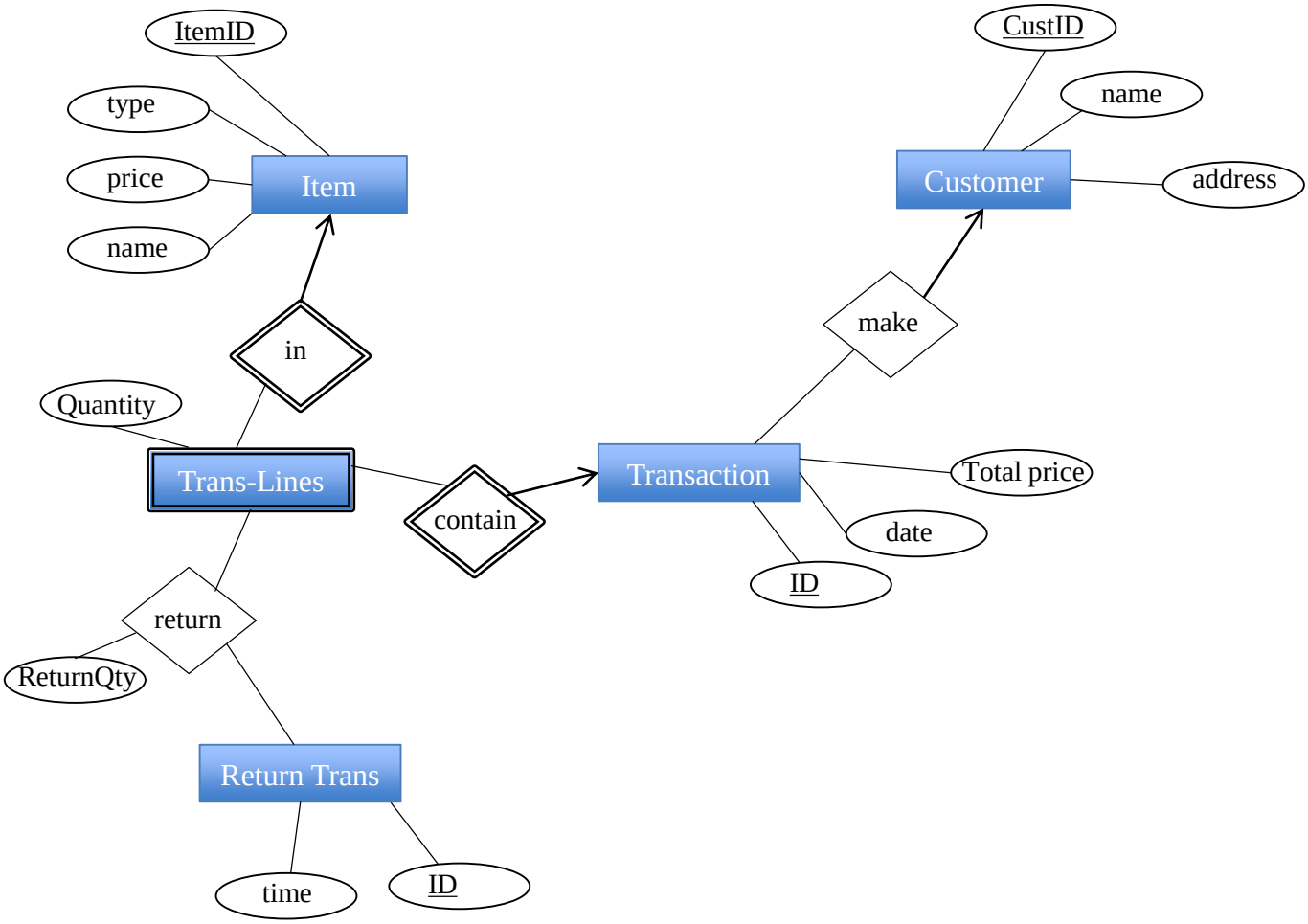
Problem 2

- A new entity set called “participation” is needed to capture which car is used by which driver in a specific race.
- Another solution is to make “participation” a three-way relationship between “Car”, “Race”, and “Driver”. But, as discussed in class, it is better to break multi-way relationships into multiple binary ones.
- Without the “Participation” entity set (or 3-way relationship), the connection between “Car”, “Driver”, and “Race” becomes lose (cannot tell which driver used which car in a given race).
- You may introduce a unique primary key for “Participation” and model it as a Strong entity set.
- The best way to model ‘Accidents’ is to link them to a specific participations. In this way you capture which car, driver, and race in an easy way. In this design we assume once a participation is involved in accident, it will exit the race (so it is at most one accident).



Problem 3

- The design is straightforward except for the “Return Transactions”
- We assumed each transaction has a unique ID (Primary key)
- The relationship between Transaction and Items, i.e., “Trans-Lines” could have been modeled as many-to-many relationship. Yet, this will make it hard to model the “Return Transaction”.
- In the Return Transaction, we need to capture the original transaction, the customer, and the item to be returned. All this information are in “Trans-Lines” entity set. Therefore, the best design is to link “Return Trans” to “Trans-Lines”. In this case, we avoid any redundant relationships.
- *Again: Other designs are possible, but hopefully they do not contain redundancy...*



Problem 4

```
Create Table Student(
  StudentId    number  Primary Key,
  name          varchar2(50),
  address       varchar2(100),
  gender        varchar2(6),
 gpa            number Default 0,
 major          varchar2(50)   not null,
 minor           varchar2(50),
Constraint genderValue check (gender in 'Male' or 'Female'));
```

```
Create Table Course(
  CourseId     number  Primary Key,
  title          varchar2(50),
  credits       number);
```

Two possible design options for the registration table

```
Create Table Register(
  CourseId        number Foreign Key References Course(CourseId),
  StudentId       number Foreign Key References Student(StudentId),
  semester        varchar2(20),
   grade          number,
Constraint pk Primary Key (CourseId, StudentId, semester ));
```

```
Create Table Register(
  RegId           number Primary Key,
  CourseId        number Foreign Key References Course(CourseId),
  StudentId       number Foreign Key References Student(StudentId),
  semester        varchar2(20),
   grade          number,
Constraint pk  Unique(CourseId, StudentId, semester ));
```