Exercises

6.1 Complete the SQL DDL definition of the bank database of Figure 6.2 to include the relations loan and borrower.

Answer:

```sql
create table loan
    (loan-number char(10),
     branch-name char(15),
     amount integer,
     primary key (loan-number),
     foreign key (branch-name) references branch)

create table borrower
    (customer-name char(20),
     loan-number char(10),
     primary key (customer-name, loan-number),
     foreign key (customer-name) references customer,
     foreign key (loan-number) references loan)
```

Declaring the pair customer-name, loan-number of relation borrower as primary key ensures that the relation does not contain duplicates.

6.2 Consider the following relational database:

```sql
employee (employee-name, street, city)
works (employee-name, company-name, salary)
company (company-name, city)
manages (employee-name, manager-name)
```

Give an SQL DDL definition of this database. Identify referential-integrity constraints that should hold, and include them in the DDL definition.

Answer:

```sql
create table employee
    (person-name char(20),
     street char(30),
     city char(30),
     primary key (person-name) )
```
create table works
(person-name char(20),
company-name char(15),
salary integer,
primary key (person-name),
foreign key (person-name) references employee,
foreign key (company-name) references company)

create table company
(company-name char(15),
city char(30),
primary key (company-name))

create table manages
(person-name char(20),
manager-name char(20),
primary key (person-name),
foreign key (person-name) references employee,
foreign key (manager-name) references employee)

Note that alternative datatypes are possible. Other choices for not null attributes may be acceptable.

6.3 Referential-integrity constraints as defined in this chapter involve exactly two relations. Consider a database that includes the following relations:

salaried-worker (name, office, phone, salary)
hourly-worker (name, hourly-wage)
address (name, street, city)

Suppose that we wish to require that every name that appears in address appear in either salaried-worker or hourly-worker, but not necessarily in both.

a. Propose a syntax for expressing such constraints.
b. Discuss the actions that the system must take to enforce a constraint of this form.

Answer:
a. For simplicity, we present a variant of the SQL syntax. As part of the create table expression for address we include

   foreign key (name) references salaried-worker or hourly-worker

b. To enforce this constraint, whenever a tuple is inserted into the address relation, a lookup on the name value must be made on the salaried-worker relation and (if that lookup failed) on the hourly-worker relation (or vice-versa).
6.4 SQL allows a foreign-key dependency to refer to the same relation, as in the following example:

```sql
create table manager
(  employee-name   char(20),
  manager-name    char(20),
  primary key employee-name,
  foreign key (manager-name) references manager
  on delete cascade )
```

Here, `employee-name` is a key to the table `manager`, meaning that each employee has at most one manager. The foreign-key clause requires that every manager also be an employee. Explain exactly what happens when a tuple in the relation `manager` is deleted.

**Answer:** The tuples of all employees of the manager, at all levels, get deleted as well! This happens in a series of steps. The initial deletion will trigger deletion of all the tuples corresponding to direct employees of the manager. These deletions will in turn cause deletions of second level employee tuples, and so on, till all direct and indirect employee tuples are deleted.

6.5 Suppose there are two relations `r` and `s`, such that the foreign key `B` of `r` references the primary key `A` of `s`. Describe how the trigger mechanism can be used to implement the `on delete cascade` option, when a tuple is deleted from `s`.

**Answer:** We define triggers for each relation whose primary-key is referred to by the foreign-key of some other relation. The trigger would be activated whenever a tuple is deleted from the referred-to relation. The action performed by the trigger would be to visit all the referring relations, and delete all the tuples in them whose foreign-key attribute value is the same as the primary-key attribute value of the deleted tuple in the referred-to relation. These set of triggers will take care of the `on delete cascade` operation.

6.6 Write an assertion for the bank database to ensure that the assets value for the Perryridge branch is equal to the sum of all the amounts lent by the Perryridge branch.

**Answer:** The assertion-name is arbitrary. We have chosen the name `perry`. Note that since the assertion applies only to the Perryridge branch we must restrict attention to only the Perryridge tuple of the `branch` relation rather than writing a constraint on the entire relation.

```sql
create assertion perry check
  (not exists (select *
    from branch
    where branch-name = 'Perryridge' and
    assets ≠ (select sum (amount)
      from loan
      where branch-name = 'Perryridge')))
```