DB - HW4

4.7

Consider the employee database of Figure 4.12. Give an SQL DDL definition of this database. Identify referential-integrity constraints that should hold, and include them in the DDL definition.

Answer:

```
CREATE TABLE employee
            (ID         CHAR(20),
             person_name VARCHAR(20),
             street      CHAR(30),
             city        CHAR(30),
             PRIMARY KEY (ID));

CREATE TABLE works
            (ID           CHAR(20),
             company_name CHAR(15),
             salary       INTEGER,
             PRIMARY KEY (ID),
             FOREIGN KEY (ID)           REFERENCES employee,
             FOREIGN KEY (company_name) REFERENCES company);

CREATE TABLE company
            (company_name CHAR(15),
             city         CHAR(20),
             PRIMARY KEY (company_name));

CREATE TABLE manages
            (ID           CHAR(20),
             manager_id   CHAR(20),
             PRIMARY KEY (ID),
             FOREIGN KEY (ID) REFERENCES employee);
```

4.18

For the database of Figure 4.12, write a query to find the ID of each employee with no manager. Note that an employee may simply have no manager listed or may have a null manager. Write your query using an outer join and then write it again using no outer join at all.

Answer:

```
SELECT ID
FROM   employee NATURAL LEFT OUTER JOIN manages
WHERE  manager_name IS NULL;
```

```
SELECT ID
FROM   employee AS E
WHERE  NOT EXISTS
      (SELECT ID
       FROM   manages AS M
       WHERE  E.ID = M.ID
       AND    M.manager_id IS NOT NULL);
```

5.4

Describe the circumstances in which you would choose to use embedded SQL rather than SQL alone or only a general-purpose programming language.

Answer:

- Why not write SQL alone?

  The SQL can only do declarative actions. If you want to present the query result on GUI, it would be impossible.

- Why not write only a general-purpose programming language?

  To realize a query in a general-purpose programming language is much more inconvenient than doing the same thing in SQL.

5.15

Consider an employee database with two relations

> **employee (employee_name, street, city**
> **works (employee_name, company_name, salary)**

where the primary keys are underlined. Write a function *avg_salary* that takes a company name as an argument and finds the average salary of employees at that cmpany. Then write an SQL statement, using that function, to find companies whose employees earn a higher salary, on average, than the average salary at “First Bank”.

Answer:

```
CREATE FUNCTION avg_salary (com_name VARCHAR(25))
  RETURN  INTEGER
  DECLARE res INTEGER;
    SELECT avg(salary) INTO res
    FROM   works
    WHERE  works.company_name = com_name;
  RETURN res;
END
```

```
SELECT company_name
FROM   works
WHERE  avg_salary(company_name) > avg_salary('First Bank');
```

5.19

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 casade option when a tuple is deleted from s.

Answer:

In this example, we should define a trigger for relation *s*, and whenever a tuple is deleted from *s*, it should be activated.

The trigger is supposed to visit relation *r*, and delete all the tuples whose foreign key value are the same as the deleted tuple’s primary key value in relation *s*.

By doing so, the trigger machanism can be used to implement the on delete casade option when a tuple is deleted from *s*.