Name

CWID

Homework Assignment 2

Due Date: October 21th, 2014 12:30pm (noon)

CS425 - Database Organization

Results

Please leave this empty!

2.1 2.2

2.3 2.4 2.5 2.6 2.7 2.8 2.9 2.10

2.11 2.12

2.15 2.16 2.17 2.18 2.19 Sum

Instructions

- Try to answer all the questions using what you have learned in class
- **When writing a query, write the query in a way that it would work over all possible database instances and not just for the given example instance!**
- Some questions are marked as bonus. You do not have to answer these questions to get full points for the assignment. However, you can get bonus points for these questions!
- **Please submit the homework as a hard copy in class or electronically using blackboard**

Consider the following database schema and example instance storing information about organ donors:

organ

donor	organ	available
Alice	Heart	2014
Bob	Lung	2015
Bob	Bladder	2015
Peter	Foot	2011
Gert	Lung	2014

donor

dName	age	bloodType
Alice	53	A+
Peter	34	AB+
Bob	44	AB-
Gert	23	A-

patient

pName	insurance	age	bloodType
Hilde	HMO	13	A-
Fritz	PPO	87	AB+

takeCare

patient	organ	doctor
Hilde	Lung	Wilhelm
Fritz	Heart	Wilhelm

doctor

docName	insurance	rate
Wilhelm	HMO	15,000
Wilhelm	PPO	20,000
Heinz	HMO	12,000
Pferd	PPO	14,000

Hints:

- Attributes with black background form the primary key of an relation
- The attribute *donor* of relation *organ* is a foreign key to relation *donor*.
- The attribute *patient* of relation *takeCare* is a foreign key to relations *patient*.
- The attribute *doctor* of relation *takeCare* stores doctors. However, it is not a foreign key to relation *doctor*, because the primary key of that relation also includes insurance information.

Part 2.1 SQL DDL (Total: 14 Points)

Question 2.1.1 (7 Points)

Write an SQL statement that creates a new table *assignedTo* that stores the *donor* and the donor's *organ* assigned to a *patient*. Furthermore, we want to store a *treatment price* for each such assignment. The combination of donor, organ, and patient uniquely identifies an assignment. Each assignment has a treatment price that is bigger than 0 and smaller than 1,000,000 dollar.

Solution

```
CREATE TABLE assignedTo (
    donor VARCHAR(256),
    organ VARCHAR(256),
    patient VARCHAR(256),
    price NUMERIC(8,2) NOT NULL CHECK(price BETWEEN 1 AND 999999),
    PRIMARY KEY (donor, organ, patient),
    FOREIGN KEY (donor, organ) REFERENCES organ (donor,organ),
    FOREIGN KEY (patient) REFERENCES patient (pName)
);
```

Question 2.1.2 (7 Points)

Write an SQL statement that creates a table *worksFor* that records which doctor works for which hospital. For each such relationship between doctors and hospitals we record a salary for the doctor. A doctor may work for several hospitals (and obviously a hospital can employ multiple doctors).

Solution

```
CREATE TABLE worksFor (
   doc VARCHAR(256),
   hospital VARCHAR(256),
   salary NUMERIC(8),
   PRIMARY KEY (doc, hospital)
);
```

Different data types or length for *donorName* and *amount* are fine too.

Part 2.2 SQL Queries (Total: 56 + 10 BONUS Points)

Question 2.2.1 (5 Points)

Write an SQL query that returns the names of all donors that donate hearts or lungs. Make sure that every donor is only returned once.

Solution

```
SELECT DISTINCT donor
FROM organ
WHERE organ = 'Heart' OR organ = 'Lung';
```

Question 2.2.2 (5 Points)

Write an SQL query that returns organs and their availability.

Solution

```
SELECT organ, available
FROM organ;
```

Question 2.2.3 (7 Points)

Return the name and rate for all doctors that support HMO combined with each patient they are taking care of.

Solution

```
SELECT docName, rate
FROM doctor d, patient p, takeCare t
WHERE d.docName = t.doctor
      AND t.patient = p.pName
      AND d.insurance = 'HMO';
```

Question 2.2.4 (7 Points)

Write an SQL query that returns the average age for all donors.

Solution

```
SELECT avg(age) AS avAge
FROM donor;
```

Question 2.2.5 (7 Points)

Write an SQL query that returns the number of compatible donor-patient pairs. A donor is considere compatible to a patient if they have the same bloodType.

Solution

```
SELECT count(*) AS compPairs
FROM patient JOIN donor USING (bloodType);
```

Question 2.2.6 (8 Points)

Write an SQL query that returns blood types for which the average age of donors for that blood type is below 35.

Solution

```
SELECT bloodType
FROM donor
GROUP BY bloodType
HAVING avg(age) < 35
```

Alternatively, do the selection in an outer query's `WHERE` clause.

Question 2.2.7 (9 Points)

Write an SQL query that returns the names of potential donors for each patient. A donor is a potential donor for a patient if he/she has the same blood type and is donating an organ that the patient needs. Return each such pair only once.

Solution

```
SELECT DISTINCT pName, dName
FROM donor d, organ o, patient p, takeCare t
WHERE d.dName = o.donor
      AND d.bloodType = p.bloodType
      AND o.organ = t.organ
      AND p.pName = t.patient;
```

Question 2.2.8 (8 Points)

Write an SQL query that returns the name(s) of the doctors which have the lowest rate.

Solution

```
SELECT d.docName
FROM doctor
WHERE rate = (SELECT min(rate) FROM doctor);
```

Question 2.2.9 BONUS (5 Points)

Write an SQL query that returns patients (name) for which there exist no donor that is of the right blood type and donates the correct organ.

Solution

```
SELECT pName
FROM patient p JOIN takeCare t ON (p.pName = t.patient)
WHERE NOT EXISTS (SELECT *
                  FROM organ o, donor d
                  WHERE o.donor = d.dName
                        AND d.bloodType = p.bloodType
                        AND o.organ = t.organ);
```

Question 2.2.10 BONUS (5 Points)

Write an SQL query that returns the average number of organs donated by donors per blood type.

Solution

```
SELECT avg(numO) AS avgNum, bloodType
FROM
    (SELECT count(*) AS numO, dName, bloodType
    FROM organ o JOIN  donor d ON (o.donor = d.dName)
    GROUP BY dName, bloodType) AS n
GROUP BY bloodType;
```

Part 2.3 SQL Updates (Total: 30 + 5 BONUS Points)

Question 2.3.1 (7 Points)

Write an SQL operation that deletes all organs that were available before 2014.

Solution

```
DELETE FROM organ
WHERE available < 2014;
```

Question 2.3.2 (8 Points)

Increase the rate of all doctors for HMO insurances by 1,000.

Solution

```
UPDATE doctor
SET rate = rate + 1000
WHERE insurance = 'HMO'
```

Question 2.3.3 (6 Points)

Insert a new organ *Foot* for donor *Alice* available in 2014.

Solution

```
INSERT INTO organ VALUES ('Alice', 'Foot', 2014);
```

Question 2.3.4 (9 Points)

Update the availability of all hearts to 2016 if their current availability is 2015.

Solution

```
UPDATE organ
SET available = 2016
WHERE organ = 'heart' AND available = 2015;
```

Question 2.3.5 BONUS (5 Points)

Change the rates for all doctors. The HMO rate should be set to the average rate for that doctor minus 500. The PPO rate should be set to the average rate for that doctor plus 500.

Solution

```
UPDATE doctor d
SET rate = (SELECT avg(rate) FROM doctor r WHERE r.docName = d.docName)
          + CASE WHEN insurance = 'PPO' THEN 500 ELSE -500 END;
```