Name

CWID

Homework Assignment 2

Due Date: October 15th, 2013 12:30pm

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 animals in a zoo:

cage

cageId	section	sqrFeet	allowedWeight
B1	bird house	1000	200
A1	africa	50000	8000
A2	africa	3000	4000
P1	prairie	500	500

keeper

kName	salary
Heinz	5400
Gertrud	2300
Hilde	50

keptIn

species	aName	cageId
rabbit	Pete	P1
rabbit	Micky	P1
elephant	Jason	A1
camel	Pete	A2
hawk	Alice	B1

attendsTo

kName	cageId
Heinz	A1
Heinz	B1
Heinz	P1
Gertrud	A2

animal

species	aName	weight
rabbit	Pete	10.0
rabbit	Micky	0.2
elephant	Jason	6000.0
camel	Pete	500.4
hawk	Alice	24.2

Hints:

- Underlined attributes form the primary key of an relation
- The attribute *kName* of relation *attendsTo* is a foreign key to *kName* in relation *keeper*. The attribute *cageId* of relation *attendsTo* is a foreign key to *cageId* in relation *cage*.
- The attributes *aName* and *species* of relation *keptIn* form a foreign key to *aName* and *species* in relation *animal*. The attribute *cageId* of relation *keptIn* is a foreign key to *cageId* in relation *cage*.

Part 2.1 SQL DDL (Total: 14 Points)

Question 2.1.1 (7 Points)

Write an SQL statement that creates a new table *section* that stores the *name*, *size* (in square feet), and *budget* of each section in the zoo. The name of a section uniquely identifies this section. Every section has a budget.

Solution

```
    CREATE TABLE section (
        name VARCHAR(256) PRIMARY KEY,
        size INT,
        budget DOUBLE NOT NULL
    );
```

Question 2.1.2 (7 Points)

Write an SQL statement that creates a table *donation* that records donations made to the zoo. For each donation we record the *name of the donor*, the *dollar amount*, and whether the donation is tax free (boolean value). Donations higher than $10,000 can not be tax free.

Solution

```
CREATE TABLE donation (
   donorName VARCHAR(256),
   amount NUMERIC(12,2),
   taxFree BOOLEAN,
   CHECK (NOT(taxFree) OR (amount <= 10000))
);
```

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 species and name for each animal in the zoo.

Solution

```
SELECT species, aName
FROM animal;
```

Question 2.2.2 (5 Points)

Write an SQL query that returns the names of all animals that are rabbits.

Solution

```
SELECT aName
FROM animal
WHERE species = 'rabbit';
```

Question 2.2.3 (7 Points)

Write an SQL query that returns the name of each zoo keeper combined with each animal's name for all animals that live in cages the keeper is attending to.

Solution

```
SELECT kName, aName
FROM keeper NATURAL JOIN attendsTo NATURAL JOIN keptIn NATURAL JOIN animal;
```

Question 2.2.4 (7 Points)

Write an SQL query that returns the combined weight of all animals living in the zoo.

Solution

```
SELECT sum(weight) AS totalweight
FROM animal;
```

Question 2.2.5 (7 Points)

Write an SQL query that returns the number of animals in each section of the zoo (that are kept in cages within a certain section).

Solution

```
SELECT section, count(*) AS numAnimals
FROM keptIn NATURAL JOIN cage
GROUP BY section;
```

Question 2.2.6 (8 Points)

Write an SQL query that returns the number of cages in the africa and the number of cages in the bird house sections of the zoo. Return both the section name and the number of cages for each such section.

Solution

```
SELECT section, count(*) AS totCage
FROM cage
GROUP BY section
HAVING section = 'africa' OR section = 'bird␣house';
```

Alternatively, do the selection in the `WHERE` clause.

Question 2.2.7 (9 Points)

Write an SQL query that returns each zoo keeper's name and the number of animal he/she takes care of (animals that live in a cage the keeper attends to).

Solution

```
SELECT kName, sum(CASE WHEN aName IS NULL THEN 0 ELSE 1 END) AS numAnimal
FROM keeper NATURAL LEFT OUTER JOIN (attendsTO NATURAL JOIN keptIn)
GROUP BY kName;
```

or less tricky

```
SELECT k.kName, COALESCE(c.numAnimal , 0) AS numAnimal
FROM keeper k
    LEFT OUTER JOIN
    (SELECT kName, count(*) AS numAnimal
    FROM keeper NATURAL JOIN attendsTO NATURAL JOIN keptIn
    GROUP BY kName) c
    ON (k.kName = c.kName);
```

even possible without COALESCE or CASE:

```
SELECT kName, sum(numAnimal) AS numAnimal
FROM ((SELECT kName, 0 AS numAnimal FROM keeper k)
     UNION ALL
     (SELECT kName, count(*) AS numAnimal
     FROM keeper NATURAL JOIN attendsTO NATURAL JOIN keptIn
     GROUP BY kName)) u2
GROUP BY kName;
```

Question 2.2.8 (8 Points)

Write an SQL query that returns the cageId of all cages that are over-occupied, i.e., where the total weight of all animals living is the cage is higher than the allowedWeight.

Solution

The trick is to recognize that *allowedWeight* can be added to the group by clause without actually changing the groups.

```
SELECT cageId
FROM cage NATURAL JOIN keptIn NATURAL JOIN animal
GROUP BY cageId, allowedWeight
HAVING sum(weight) > allowedWeight;
```

Alternatively,

```
SELECT c.cageId
FROM cage c
     JOIN
     (SELECT cageId, sum(weight) AS totalW
     FROM keptIn NATURAL JOIN animal
     GROUP BY cageId) w
     ON (c.cageId = w.cageId AND w.totalW > c.allowedWeight);
```

Question 2.2.9 BONUS (5 Points)

Write an SQL query that returns the salary of zoo keepers that attend to at least one cage in each section of the zoo.

Solution

```
SELECT salary
FROM cage NATURAL JOIN attendsTo NATURAL JOIN keeper
GROUP BY kName, salary
HAVING count(DISTINCT section) = (SELECT count(DISTINCT section) FROM cage);
```

Alternatively,

```
SELECT salary, kName
FROM keeper k
WHERE NOT EXISTS (SELECT *
                  FROM (SELECT DISTINCT section, cageid FROM cage) c1
                  WHERE NOT EXISTS (SELECT *
                                    FROM attendsTo a NATURAL JOIN cage c2
                                    WHERE a.kName = k.kName AND c2.section = c1.section));
```

Question 2.2.10 BONUS (5 Points)

Write an SQL query that returns animals (their name, weight, and species) that weight less then the average weight for their species.

Solution

```
SELECT aName, weight, species
FROM animal a
WHERE a.weight < (SELECT avg(weight) FROM animal b WHERE a.species = b.species);
```

Alternatively, have a subquery that computes average weight per species and join that subquery with the animal relation.

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

Question 2.3.1 (7 Points)

Write an SQL operation that deletes all cages in the africa section.

Solution

```
DELETE FROM cage
WHERE section = 'africa';
```

Question 2.3.2 (6 Points)

Insert a new rabbit named Fluffy weighting 3.5 pounds into the animal table.

Solution

```
INSERT INTO animal VALUES ('rabbit', 'Fluffy', 3.5);
```

Question 2.3.3 (8 Points)

Increase the salary of all zoo keepers by 15%.

Solution

```
UPDATE keeper SET salary = 1.15 * salary;
```

Question 2.3.4 (9 Points)

For all cages that are larger than 5000 sqrFeet set the allowedWeight to 10,000.

Solution

```
UPDATE cage SET allowedWeight = 10000 WHERE sqlFeet > 5000;
```

Question 2.3.5 BONUS (5 Points)

Increase the salary of all zoo keepers by 100 per animal they are taking care of. Only apply this update to zoo keepers that take care of more than 2 animals.

Solution

```
UPDATE keeper k
SET salary = salary + (SELECT 100 * count(*)
                      FROM attendsTo a NATURAL JOIN keptIn i
                      WHERE a.kName = k.kName)
WHERE 2 < (SELECT count(*) FROM attendsTo a WHERE k.kName = a.kName);
```