Name

CWID

Midterm Exam

October 30th, 2017
3:15-4:30

CS425 - Database Organization

Results

Please leave this empty!

1.1 1.2 1.3 1.4 Sum

Instructions

- Try to answer all the questions using what you have learned in class. Keep hard questions until the end.
- **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!**
- The exam is closed book and closed notes!
- **For relational algebra questions assume set semantics!**

Consider the following database schema and example instance for a car database:

person

name	age	state
Peter	32	IL
Alice	57	CA
Bob	17	NY

car

license	color	productionYear	model
A54-FGY	green	2011	Porsche 510
G55-HES	blue	2017	Porsche 510
F55-JRK	red	2010	Impala
A33-IKF	red	1980	Golf

owns

owner	car	ownsSince
Peter	A54-FGY	2015
Peter	F55-JRK	2010
Alice	A33-IKF	1990

model

mId	brand	weight
Porsche 510	Porsche	1300
Porsche 310	Porsche	1400
Impala	GM	3000
Golf	Volkswagen	2500
S500	Mercedes	3000

Hints:

- Attributes with black background form the primary key of a relation (.e.g, `name` for relation `person`)
- The attribute `car` of relation `owns` is a foreign key to `license` of relation `car`.
- The attribute `owner` of relation `owns` is a foreign key to `name` of relation `person`.
- The attribute `model` of relation `car` is a foreign key to `mId` of relation `model`.
- All foreign keys have been created with the **CASCADE** option.

Part 1.1 Relational Algebra (Total: 28 Points)

Question 1.1.1 (8 Points)

Write a **relational algebra** expression that returns the *name* and *age* of all persons that are 50 years or older and live in Illinois (`IL`).

Solution

$$\pi_{name,age}(\sigma_{age \geq 50 \wedge state=IL}(\text{ person }))$$

Question 1.1.2 (10 Points)

Write a **relational algebra** expression that returns the license plate number (*license*), color and model of cars that are owned by persons younger than 21 which live in New York (`NY`) or California (`CA`) and which weight more than `1000` (pounds).

Solution

$$\pi_{license,color,model}(\sigma_{age<21 \wedge (state=NY \vee state=CA)}(\text{ person }) \bowtie_{name=owner} \text{ owns } \bowtie_{car=license} \text{ car } \bowtie \sigma_{weight>1000}(\text{ model }))$$

Question 1.1.3 (10 Points)

Write a **relational algebra** expression that returns for each brand the heaviest car model produced by that brand. Return the *mId* and *brand* for each such car.

Solution

$$\pi_{mId,brand}({}_{brand}\mathcal{G}_{max(weight)\rightarrow weight}(\text{ model }))\bowtie \text{model }$$

Part 1.2 SQL - DDL (Total: 16 Points)

Question 1.2.1 (16 Points)

Write an **SQL DDL statement** that creates a new relation `parkingSpot` that records information about about parking lots. This relations should have attributes *streetName*, *streetNumber*, *lotNumber*, *price*, and *car*. A parking spot is uniquely identified by the combination of *streetName*, *streetNumber*, and *lotNumber*. Attribute *car* is a foreign key to relation *car*. When a car that is currently assigned a parking spot is deleted from the database, then the *car* attribute of parking spot should be set to `NULL`. If a license plate number of a car is updated, then the *car* attribute of all parking spots the car is assigned to should be updated too. Note that both *prices* and *lot numbers* cannot be negative. *Prices* cannot be `NULL`.

Solution

```
CREATE TABLE parkingSpot (
    streenName VARCHAR(30),
    streetNumber INT,
    lotNumber INT,
    price NUMERIC(8,2) NOT NULL,
    car VARCHAR(12)
    PRIMARY KEY (streetName, streetNumber, logNumber),
    FOREIGN KEY (car) REFERENCES car ON DELETE SET NULL ON UPDATE CASCADE,
    CHECK (price >= 0.0 AND lotNumber >= 0)
);
```

Part 1.3 SQL - Queries (Total: 36 Points)

Question 1.3.1 (10 Points)

Write an **SQL query** that returns the *names* of persons that own Porsche cars (cars with *brand* `Porsche`).

Solution

```
SELECT name
FROM car c, owns o, person p, model m
WHERE c.license = o.car
      AND o.owner = p.name
      AND m.mId = c.model
      AND model = 'Porsche';
```

alternatively explicit joins of course.

Question 1.3.2 (12 Points)

Write an **SQL query** that returns total weight of cars per brand. Note that here you should consider cars (relation *car*) not car models. For instance, in the example database there are two Porsche 510 cars each weighting 1300. Thus, the total weight of Porsche cars is 2600.

Solution

```
SELECT sum(weight) AS totWe, brand
FROM car c, model m
WHERE c.model = m.mId
GROUP BY brand;
```

Question 1.3.3 (14 Points)

Write an **SQL query** that returns car `models` for which no red cars exists.

Solution

```
SELECT DISTINCT model
FROM car
WHERE model NOT IN (SELECT mId
                    FROM model
                    WHERE color = 'red');
```

alternatives include aggregation (count is 0) and set difference

Part 1.4 SQL - Updates (Total: 20 Points)

Question 1.4.1 (13 Points)

Write an **SQL statement** that deletes the all models for which there is no car of this model. In the provided example database this would delete the model *S500*.

Solution

```
DELETE FROM model
WHERE mId NOT IN (SELECT model FROM car);
```

Question 1.4.2 (7 Points)

Write an **SQL statement** that increase the *age* of all persons by 1 year.

Solution

```
UPDATE person SET age = age + 1;
```