Databases
TDA357/DIT621

Exercise 6

Relational Algebra and Transactions

Relational Algebra

1. Assume the following schema:

   Artists (<u>name</u>)
   Persons (<u>name</u>, birthDate)
   name → Artists.name
   Bands (<u>name</u>)
   name → Artists.name
   MembersOf (<u>person</u>, <u>band</u>)
   person → Persons.name
   band → Bands.name
   Albums (<u>title</u>, year, artist)
   artist → Artists.name
   Songs (<u>title</u>, length, artist)
   artist → Artists.name
   Tracks (<u>song</u>, <u>album</u>)
   song → Songs.title
   album → Albums.title

   (a) Write an SQL query and a relational algebra expression to find all solo albums, i.e., albums whose artist is not a band but a person. The format of the output is:

   ```
     title
   -------------------------
     Under Stars
     Woman of the world
   ```

   (b) Write an SQL query and a relational algebra expression to find all albums by Amy MacDonald from year 2016 or later, with their years of appearance. The format of the output is:

```
  title              | year
---------------------+------
  Under Stars        | 2017
  Woman of the world | 2018
```

(c) Write a query, in SQL and in relational algebra, to find the total length of Amy MacDonald's Album "This is the life". The total length means the total length of all songs on that album, assuming, for simplicity, that lengths are expressed in seconds as an integer. The query output is of the following format:

```
 sum
-----
 3528
```

2. Consider the following schema:

Airports (<u>code</u>, city)
Flights (<u>code</u>, airlineName)
FlightLegs (<u>code</u>, <u>from</u>, to, departureTime, arrivalTime)
code → Flights.code
from → Airports.code
to → Airports.code
(code, to) UNIQUE

Each flight-leg represents one take-off and landing, and a flight might have multiple flight-legs.

(a) Below is an SQL query that finds all airports that have departures or arrivals (or both) of flights operated by Lufthansa or SAS (or both).

Express this query as a relational algebra expression:

```
SELECT DISTINCT served
FROM ((SELECT to AS served, airlineName
       FROM Flights JOIN FlightLegs ON FlightLegs.code = Flights.code)
      UNION
      (SELECT from AS served, airlineName
       FROM Flights JOIN FlightLegs ON FlightLegs.code = Flights.code))
     AS D
WHERE D.airlineName = 'Lufthansa' OR D.airlineName = 'SAS';
```

(b) Translate the following relational algebra expression to an SQL query:

$$\pi_{\texttt{First.departureTime,Second.arrivalTime}} \left((\rho_{\texttt{First}}\, \texttt{FlightLegs}) \bowtie_{\texttt{First.to=Second.from}} (\rho_{\texttt{Second}}\, \texttt{FlightLegs}) \right)$$

(c) Write a relational algebra expression for finding the code of all flights that have more than two flight-legs.

(d) Write a relational algebra expression for finding the code of all flights that departs from Gothenburg and land in London (you do not have to list the flight-legs).

3. (Previous exam question.)

Recall the vet clinic problem from exercises 1, extended with the table that was added for the bookings.

Clients (<u>name</u>, email, phonenr)
Pets (<u>chipnr</u>, name, owner, type, born)
owner → Client.name
type ∈ {dog, cat, rabbit}
Vets (<u>id</u>, name, phonenr, specialisation)
specialisation ∈ {dog, cat, rabbit}
Bookings (<u>emp</u>, chipnr, <u>bdate</u>, <u>time</u>, length)
emp → Vets.id
chipnr → Pets.chipnr
time ∈ {9,10,11,13,14,15}
length ∈ {30,60}
UNIQUE (chipnr, bdate, time)

Recall that chipnr and id are numbers that identify the pets and the employees respectively, and born is the year on which the pet was born. Assume name is enough to identify a client. The clinic only takes care of dogs, cats or rabbits. Consultations can only be booked at 9, 10, 11, 13, 14 or 15 hours. Length of consultations can be 30 or 60 minutes.

Note: For this question, it is fine to use the same expression in relational algebra as in SQL to work with dates, so for example use CURRENT_DATE even in relational algebra to get today's date.

(a) Write a relational algebra expression that gives all the pets' names and their owners that had a booking during the last year (the last 365 days). The result cannot contain repetition.

Note: the result should only contain the bookings that took place the last year, not future bookings!

Hint: CURRENT_DATE - 1 gives yesterday's date.

(b) Write a relational algebra expression that gives the ids of all cat specialists that have no bookings today at 13.

(c) Write a relational algebra expression that gives the name and phone number of all clients with at least 2 pets in the clinic.

Transactions

4. Consider an existing database with the following database definition in a PostgreSQL DBMS:

```
CREATE TABLE Users (
  id INT PRIMARY KEY,
  name TEXT,
  password TEXT);

CREATE TABLE UserStatus (
  id INT PRIMARY KEY REFERENCES Users,
  loggedin BOOLEAN NOT NULL);

CREATE TABLE Logbook (
  id INT REFERENCES Users,
  timestamp INT,
  name TEXT,
  PRIMARY KEY (id, timestamp));
```

   (a) Users of a web application are allowed to query this database for a certain user id. This function implemented in JDBC using the following code fragment:

```
...
String query = "SELECT * FROM UserStatus WHERE id = '" + userinput + "'";
PreparedStatement stmt = conn.prepareStatement (query);
ResultSet rs = stmt.executeQuery ();
...
```

   Does this code contain an SQL injection vulnerability? If it does not, why not? If it does, how would you correct the code?

   (b) The JDBC code below is supposed to run a batch job in the following way: loop through an array of users, set each users password hash to a certain value and add a logbook message for that user; if any of the users do not exist, the whole batch job should be rolled back.

   The queries seem to work fine, and if there is an invalid id in the array the error message is printed, but all the other ids in the array are still changed! What goes wrong and how can it be fixed

   Hint: Two separate things need to be fixed, think about what happens both before and after the error.

```
int[] blockList = {11,42,55}; // Ids to be locked
PreparedStatement update = conn.prepareStatement (
  "UPDATE Users SET password = 'qiyh4XPJGsOZ2MEAy' WHERE id = ?");
```

```
PreparedStatement insert = conn.prepareStatement (
  "INSERT INTO Logbook VALUES (?, now(), 'Account locked')");
conn.setAutoCommit (false); // Enable transactions

for (int user : blockList){ // For each user in blocklist ...
    update.setInt (1, user);
    insert.setInt (1, user);
    int res = update.executeUpdate ();
    if (res == 0){ // 0 rows affected - user does not exist!
      System.err.println ("Error, missing id: " + user);
      conn.rollback ();
    } else {
      insert.executeUpdate ();
      conn.commit ();
    }
}
```