Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Datenbanksysteme · Prof. Dr. Grust

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Datenbanksysteme I

WS 2021/22
Torsten Grust, Christian Duta

Assignment #12

Submission Deadline: February 02, 2022 - 10:00

aircraft(*aid*, *name*, *manufacturer*, *cruisingrange*)
pilots(*pid*, *name*, *salary*)
certified(*pid, aid*)
flights(*flno*, *from*, *to*, *distance*, *departs*, *arrives*)
crew(*flno, pid*)

Figure 1: Airline database schema

So

This assignment revolves around the airline database schema in Figure 1. With this assignment we provided CSV-files (archived in `airline.zip`) which contain instances of the airline database. Feel free to extend these CSV-files with more rows **and hand hin the modified CSV-files as well**.

Column *cruiserange* describes the maximum distance an aircraft is able to travel airborne before having to land. Column *distance* is the total distance required to complete a flight from start to finish. Column *departs* and *arrives* are timestamps. To use these timestamps in PyQL, import the `datetime` library (`from datetime import *`). Use function `datetime.fromisoformat(string)` to convert timestamp strings to a timestamp of type `datetime`. Timestamps can be compared with the usual operators ($<$, $=$, $>$, $\leq$, $\ldots$).

Exercise 1: Relational Algebra (12 Points)

Consider the airline database schema in Figure 1. Formulate the following queries using the PyQL implementations of the relational algebra operators (like in the lecture).

1. Find the *flno* and the origin of all flights *to* 'Berlin'.

   Schema of the result: `flno | origin`

2. Find the *aid* and *name* of all aircraft which can be used on non-stop flights *from* 'Bonn' *to* 'Madras'.

   Schema of the result: `aid`

3. Find the *pid* and *name* of all pilots certified for some 'Boeing' aircraft.

   Schema of the result: `pid | name`

4. Find the name of all cities that lie on a round trip flight route with exactly three flights.
   Note: Departure and arrival times do *not* have to be considered here.

   Schema of the result: `city`

Exercise 2: SQL and Relational Algebra (9 Points)

Consider the airline database schema in Figure 1.

1. The following query returns the *flno* of flights directly from Basel to Amsterdam and all flights from Basel to Berlin for which a connecting flight to Amsterdam exists.

   ```
   SELECT f.flno
   FROM   flights AS f
   WHERE  (f.from = 'Basel' AND f.to = 'Amsterdam') OR
          (f.from = 'Basel' AND f.to = 'Berlin' AND
          EXISTS (SELECT 1
                  FROM   flights AS f1
                  WHERE  f1.from = f.to AND
                         f1.to = 'Amsterdam' AND f1.departs > f.arrives));
   ```

   Translate this SQL query into an equivalent relational algebra expression using the PyQL implementations of the relational algebra operators (like in the lecture).

2. The other way round, the following RA query computes pilots that are only certified for aircraft with a cruising range of less than 800 km (or for no aircraft at all). Write an equivalent SQL query.

$$\text{pilots} \setminus (\pi[\text{pid}, \text{name}, \text{salary}](\text{pilots} \bowtie (\text{certified} \bowtie \pi[\text{aid}](\sigma[\text{range} \geq 800](\text{aircraft})))))$$

Exercise 3: Non-monotonic Queries in the Relational Algebra (9 Points)

Consider the airline database schema in Figure 1. You are given the following non-monotonic problems:

i. Find the *pid* and *name* of all pilots which can be deployed for the flight with *flno* 'LH 970'.
A pilot is deployable for a flight, if they are certified for an aircraft which can pass the flight distance and they are not yet deployed on another flight (see relation crew in Figure 1) during the scheduled flight time. The scheduled flight time is defined by *departs* and *arrives*.

   Schema of the result: `pid | name`

ii. Find *flno*, *from* and *to* of the flights with the shortest traveling distance.

   Schema of the result: `flno | from | to`

For both problems i and ii formulate the relational algebra using the PyQL implementations of the relational algebra operators (like in the lecture) which produces the expected result. Hand in the query **in the form of a query operator tree (plan)** written in plain text (`.txt`) or as a PDF[1].

[1] Consider using web-based drawing tools like `https://app.diagrams.net/`.