Examination in Databases (TDA357/DIT620)

20 March 2015 at 8:30-12:30, Hörsalsvägen 5

CHALMERS UNIVERSITY OF TECHNOLOGY AND UNIVERSITY OF GOTHENBURG
Department of Computer Science and Engineering

Examiner: Aarne Ranta tel. 772 10 82. The examiner will visit the exam room at 9:30 and 11:30.

Results: Will be published 10 April at the latest.

Exam review: In April after the Easter break. To be announced at course web page and in the Google group.

Grades: Chalmers: 24 for 3, 36 for 4, 48 for 5. GU: 24 for G, 42 for VG.

Help material: One "cheat sheet", which is a A4 sheet with hand-written notes. You may write on both sides of that sheet. If you bring a sheet, it must be handed in with your answers to the exam questions. One English language dictionary is also allowed.

Specific instructions:

- Answer in English where possible. You may clarify your answers in Swedish if you are not confident you have expressed yourself correctly in English.
- Begin the answer to each question (numbers 1 to 6) on a new page (the a,b,c,... parts with the same number can be on the same page)
- Write clearly: unreadable = wrong!
- Fewer points are given for unnecessarily complicated solutions.
- Indicate clearly if you make any assumptions that are not given in the question.
- In particular: in SQL questions, use standard SQL or Oracle. If you use any other variant (such as MySQL), say this; but full points are not guaranteed.
- Write the question number on every page. If you need many pages for one question, number the pages as well, for instance, "Question 3 p. 2".

Most questions in this exam will be about one and the same domain: flights. This domain makes heavy use of databases, familiar from web applications. The exam will go through different aspects having to do with airlines, flight connections, and bookings.

Table 1. We start with a table listing some flights. It is an example of a relation that links together flight codes, airlines, airports, and aircraft. The airlines and airports are real ones, whereas the codes are fake.

flight code	airline	prime flight	operating airline	departure city	departure airport	destination city	destination airport	aircraft type	seats
SK111	SAS	SK111	SAS	Gothenburg	GOT	Frankfurt	FRA	B737	140
LH555	Lufthansa	SK111	SAS	Gothenburg	GOT	Frankfurt	FRA	B737	140
AF111	Air France	AF111	Air France	Gothenburg	GOT	Paris	CDG	A320	170
LH111	Lufthansa	LH111	Lufthansa	Frankfurt	FRA	Paris	CDG	A321	200
LH222	Lufthansa	LH222	Lufthansa	Frankfurt	FRA	Malta	MLA	A320	170
AF222	Air France	AF222	Air France	Paris	ORY	Malta	MLA	A320	170
AB222	Air Berlin	AB222	Air Berlin	Frankfurt	FRA	Munich	MUC	A320	170
KM111	Air Malta	KM111	Air Malta	Munich	MUC	Malta	MLA	A319	140
LH333	Lufthansa	KM111	Air Malta	Munich	MUC	Malta	MLA	A319	140
SK222	SAS	KM111	Air Malta	Munich	MUC	Malta	MLA	A319	140
AF333	Air France	AF333	Air France	Paris	CDG	Frankfurt	FRA	A320	170

We assume the following (slightly simplified) conventions for this domain:

- the “flight code” attribute determines all other attributes on a row
- the “prime flight” is the flight code used by the airline operating the flight; the “flight code” in the first column can thus belong to another airline that has a code sharing agreement with the operating airline
- the “prime flight” appears in the table as a “flight code” as well, having itself as prime flight
- each airport has a unique code
- every aircraft of the same type has the same number of seats

(It is a common practice that one and the same flight can be booked using different airlines. Each airline uses a different "flight code", but the passengers end up in the same plane. The code used by the actual operating airline is called the "prime flight" code. For example, whether you book flight LH333 with Lufthansa or flight SK222 with SAS, you end up in the plane of Air Malta flight KM111.)

Question 1. Modelling and Design, basic (11p)

Question 1a (2p)
Find at least four redundancies in Table 1.

Question 1b (5p)
Draw an Entity-Relationship diagram that models the data in Table 1 in a meaningful way. The diagram must have some separate entities and relationships. Mark the keys by underlining them.

Question 1c (4p)
Convert your Entity-Relationship (E-R) diagram to a database schema. Mark all keys and references.

Question 2. Modelling and Design, advanced (9p)

This question refers to the same domain as Question 1.

Question 2a (3p)
Identify the functional dependencies and keys in the domain as described in Question 1. You must have some functional dependencies that are not superkeys. Consider the entire Table 1 as one relation. For functional dependencies, it is enough to list a base (a minimal set that implies all the others).

Question 2b (4p)
Starting with Table 1 and the functional dependencies and keys in (2a), decompose the relation into BCNF (Boyce-Codd Normal Form). Show all intermediate steps. **Notice**: if you find out that the relation is already in BCNF, then you have done something wrong in (2a).

Question 2c (2p)
Suppose you know the attributes of a relation and that it has no functional dependencies.

- Do you have enough information to bring it to BCNF. If yes, how? If no, why?
- Do you have enough information to bring it to the Fourth Normal Form (4NF). If yes, how? If no, why?

Question 3. Construction and Querying, basic (17p)

Question 3a (4p)
Convert the schema you designed in Question 1c to SQL. Make sure you use reasonable datatypes and express all referential constraints properly.

Questions 3b-3d

In the questions that follow, we need information about departure and arrival times. We assume the following relations have been implemented as tables in SQL:

Airports(<u>code</u>,city)

FlightCodes(<u>code</u>, airlineName)

Flights(departureAirport, destinationAirport, departureTime, arrivalTime, <u>code</u>)
departureAirport -> Airports.code
destinationAirport -> Airports.code
code -> FlightCodes.code

The listed flight code is the prime flight (i.e. the one used by the operating airline). For simplicity, we assume that departureTime and arrivalTime are integers denoting full hours, all in the same time zone, and that

$0 \leq \text{departureTime} < \text{arrivalTime} < 24$

Question 3b (3p)
Using this schema (not the one you defined in 1c), write an SQL query that finds all airports that have departures or arrivals (or both) of flights operated by Lufthansa or SAS (or both).

Question 3c (3p)
Using the schema from (3b), write an SQL query that shows the names of all cities together with the number of flights that depart from them, and sorts them by the number of flights in descending order (i.e. the city with the largest number of departures first).

Question 3d (7p)
Using the above schema (same as in 3b and 3c), write a view that lists all connections from any city X to any other city Y involving 1 or 2 legs (i.e. separate flights between two cities: if you fly from Gothenburg to Paris with a change in Frankfurt, the connection has two legs).

The query must return a table with the following information (and nothing else):

- the departure city X and destination city Y
- the departure time from X and the arrival time in Y
- the number of legs
- the total time from departure in X to arrival in Y
- the total time spent in the air

A change is possible if and only if (1) it happens at the same airport (2) the changing time is at least 1 hour, and (3) the connecting flight is on the same day.

Question 4. Construction and Querying, advanced (6p)

Note: In these questions, we don't care about the set/multiset distinction.

Question 4a (3p)
Express the query of question in (3b) by a relational algebra expression.

Question 4b (3p)
Translate the following relational algebra expression to an SQL query:

$$\pi_{\text{First.depatureTime,Second.arrivalTime}} ((\rho_{\text{First}}(\text{Flights})) \bowtie_{\text{First.destinationAirport = Second.departureAirport}} (\rho_{\text{Second}}(\text{Flights})))$$

Question 5. Triggers, basic (9p)

Assume the following two tables are added to the schema in (3b): a table listing for each flight the number of available seats and the price per ticket,

```
  AvailableFlights(flight, numberOfFreeSeats, price)
    flight -> Flights.code
```

another one listing the passengers that have been booked for each flight, with the price they have paid and a unique booking reference number (an integer, to keep things simple),

```
  Bookings(reference, flight, passenger, price),
    flight -> Flights.code
```

Question 5a (2p)
Create a view that lists booking references, passengers, flight codes, and departure and destination cities.

Question 5b (7p)
Create a trigger on the view defined in (5a). This trigger takes care of booking a new passenger to a flight. It is fired by an insertion of a passenger and a flight code to the view, for instance, "book Annie Adams for AF666". Its effect should be the following:

- if the number of free seats on AF666 is positive, decrement it by one; the booking is successful
- if there are no free seats, the booking fails
- if the booking is successful, add Annie Adams and AF666 to Bookings, with the price given in AvailableFlights when booking her; also add a booking reference which is the maximum of the previous references (for all flights) plus one
- if the booking is successful, increment the price by 50 SEK for the next passenger (thus the fuller the flight, the more you pay)

Question 6. Miscellaneous, advanced (8p)

Question 6a (4p)
Bookings are typically made concurrently by many users. Assuming that the booking steps in (5b) are *not* an atomic transaction, show sequences of events (i.e. queries and updates from different customers) where

- a customer is suggested a flight but it turns out to be full
- a customer is told that a flight is full although it has seats
- a customer has to pay overprice i.e. a price that applies to customers booking later

For each sequence, also indicate which isolation level would prevent the undesired outcome from happening.

Question 6b (4p)
Write an XML document that contains a DTD corresponding to the database schema in (3b) together with data used on the first and the last row in the table of Table 1 (where you can invent the departure and arrival times yourself).