Exercise 3 Theory topics: relations and dependencies

1. Functional dependencies and normal forms (Lecture 6).

Exercise 1 (exam 2015 + exercise session 2 2018)

First we will look at the flights table, that you saw in the previous exercise session. As you already know the table, most of you will not need to study it again. Nevertheless, I attach its whole description.

<table>
<thead>
<tr>
<th>flight code</th>
<th>airline</th>
<th>prime flight</th>
<th>operating airline</th>
<th>departure city</th>
<th>destination</th>
<th>aircraft</th>
<th>type</th>
<th>seats</th>
</tr>
</thead>
<tbody>
<tr>
<td>SK111</td>
<td>SAS</td>
<td>SK111</td>
<td>Gotenborg</td>
<td>GOT</td>
<td>Frankfurt</td>
<td>FRA</td>
<td>3735</td>
<td>140</td>
</tr>
<tr>
<td>LH333</td>
<td>Lufthansa</td>
<td>LH333</td>
<td>Gotenborg</td>
<td>GOT</td>
<td>Frankfurt</td>
<td>FRA</td>
<td>3735</td>
<td>140</td>
</tr>
<tr>
<td>AF111</td>
<td>Air France</td>
<td>AF111</td>
<td>Gotenborg</td>
<td>GOT</td>
<td>Paris</td>
<td>CDG</td>
<td>A320</td>
<td>170</td>
</tr>
<tr>
<td>LH111</td>
<td>Lufthansa</td>
<td>LH111</td>
<td>Lufthansa</td>
<td>FRA</td>
<td>Paris</td>
<td>CDG</td>
<td>A321</td>
<td>200</td>
</tr>
<tr>
<td>LH222</td>
<td>Lufthansa</td>
<td>LH222</td>
<td>Lufthansa</td>
<td>FRA</td>
<td>Malta</td>
<td>MLA</td>
<td>A320</td>
<td>170</td>
</tr>
<tr>
<td>AF222</td>
<td>Air France</td>
<td>AF222</td>
<td>Air France</td>
<td>Paris</td>
<td>ORY</td>
<td>Malta</td>
<td>MLA</td>
<td>170</td>
</tr>
<tr>
<td>AB222</td>
<td>Air Berlin</td>
<td>AB222</td>
<td>Air Berlin</td>
<td>FRA</td>
<td>Munich</td>
<td>MUC</td>
<td>A320</td>
<td>170</td>
</tr>
<tr>
<td>KM111</td>
<td>Air Malta</td>
<td>KM111</td>
<td>Air Malta</td>
<td>MUC</td>
<td>Malta</td>
<td>MLA</td>
<td>A319</td>
<td>140</td>
</tr>
<tr>
<td>LH333</td>
<td>Lufthansa</td>
<td>LH333</td>
<td>Air Malta</td>
<td>MUC</td>
<td>Malta</td>
<td>MLA</td>
<td>A319</td>
<td>140</td>
</tr>
<tr>
<td>SK222</td>
<td>SAS</td>
<td>SK222</td>
<td>Air Malta</td>
<td>MUC</td>
<td>Malta</td>
<td>MLA</td>
<td>A319</td>
<td>140</td>
</tr>
<tr>
<td>AF333</td>
<td>Air France</td>
<td>AF333</td>
<td>Air France</td>
<td>CDG</td>
<td>Frankfurt</td>
<td>FRA</td>
<td>A320</td>
<td>170</td>
</tr>
</tbody>
</table>

Table 1 Flights table

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 1a. Identify the functional dependencies and keys in the domain as described above. 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). (3p.)

Functional dependencies:

- flightCode -> (all attributes) (enough to say airline and primeFlight)
- departureAirport -> departureCity
destinationAirport -> destinationCity
aircraft -> seats
primeFlight -> all attributes except flightcode and airline

optionally also:
primeFlight airline -> (all attributes) (enough to write flightCode)
primeFlight -> operatingAirline

keys:
flightCode

optionally also (assuming another company uses just one code for sharing a flight)
primeFlight, airline

The optional dependencies and keys are not required in the answer, but compensate for possibly missing other ones.

**Question 1b.** Starting with Table 1 and the functional dependencies and keys in (1a), 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 (1a). (4p.)

R1(_aircraftType, seats)
R2(_destinationAirport, destinationCity)
R3(_departureAirport, departureCity)
R4(_flightCode, airline, primeFlight, operatingAirline, departureAirport, destinationAirport, aircraftType)
  departureAirport -> R3.departureAirport
  destinationAirport -> R2.destinationAirport
  aircraftType -> R1.aircraftType

primeFlight -> operatingAirline brings another relation if considered

**Question 1c.** 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? (2p.)

A relation that has no functional dependencies.

Enough information to bring it to BCNF? Yes. It is already in BCNF, so we need not do anything.

Enough information to bring it to 4NF? No. There can be multivalued dependencies that are not functional dependencies.

**Exercise 2** (from Exercise session 16 November 2017)

Suppose we have relation \( R(A, B, C, D, E, F, G) \) and functional dependencies

- \( AC \rightarrow E \)
- \( EG \rightarrow D \)
• \( AC \rightarrow B \)
• \( DF \rightarrow A \)

**Question 2a.** This relation \( R \) has three keys. Indicate which ones with some simple reasoning. Moreover, tell why the other two are not keys of \( R \). (4p)

- \( \{ A, C, E, F \} \) - this one does not identify all attributes
- \( \{ A, C, F, G \} \) - ok!
- \( \{ C, D, F, G \} \) - ok!
- \( \{ C, E, F, G \} \) - ok!
- \( \{ B, C, E, F, G \} \) - this one is a superkey but not a key, since attribute \( G \) can be removed and the resulting set of attributes is a key

**Question 2b.** Decompose this relation \( R \) to Boyce-Codd normal form. Remember to show each step in the normalisation process, and at each step write down which functional dependency is being used. (4p)

Decompose on \( AC \rightarrow E \)
\[ \{ AC \} + = \{ ACBE \} \]
\( R1(A, C, B, E) \)
\( R2(A, C, D, F, G) \)
\( A, C \rightarrow R1. (A, C) \)

Decompose \( R2 \) on \( DF \rightarrow A \)
\( \{ D, F \} + = \{ DFA \} \)
\( R21(D, F, A) \)
\( R22(D, F, C, G) \)
\( D, F \rightarrow R21. (D, F) \)

Key of \( R22 \) is \( DFCG \)

**Question 2c.** Indicate and justify which functional dependency (-ies) of \( R \) violate Third Normal Form. (2p)

\( AC \rightarrow B \)
Left side is not a superkey of \( R \), and \( B \) is not prime in \( R \).

**Question 2d.** Decompose relation \( R \) to Third Normal Form (3p).

\( R1(A, B, C, E) \)
\( R2(D, E, G) \)
\( R3(A, D, F) \)
\( R4(C, D, F, G) \)

**Exercise 3 (from Re-exam August 2015)**

**Question 3a.** Give an example of a relation that is in BCNF (BoyceCodd Normal Form) but not in 4NF (Fourth Normal Form). Show all the information that is needed: attributes, dependencies, keys, etc, clearly stating what the 4NF violations are, as well as an instance (a set of tuples). (2p)

<table>
<thead>
<tr>
<th>Company</th>
<th>Produces</th>
<th>SellsTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company A</td>
<td>Milk</td>
<td>ICA</td>
</tr>
<tr>
<td>Company A</td>
<td>Milk</td>
<td>Netto</td>
</tr>
<tr>
<td>Company A</td>
<td>Yogurt</td>
<td>ICA</td>
</tr>
<tr>
<td>Company A</td>
<td>Yogurt</td>
<td>Netto</td>
</tr>
</tbody>
</table>
Key (company, produces, sellsTo)

MVD company ->>> produces

Question 3b. Transform your relation in (2a) to 4NF. (2p)

Not in 4NF. Decomposition:

<table>
<thead>
<tr>
<th>Company</th>
<th>Produces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company A</td>
<td>Milk</td>
</tr>
<tr>
<td>Company A</td>
<td>Yogurt</td>
</tr>
</tbody>
</table>

Key (company, produces)

<table>
<thead>
<tr>
<th>Company</th>
<th>SellsTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company A</td>
<td>ICA</td>
</tr>
<tr>
<td>Company A</td>
<td>Netto</td>
</tr>
</tbody>
</table>

Key (company, sellsTo)

2. Relational Algebra and query compilation (Lecture 7)

Exercise 4. (Database Tutorial 3: Relational Algebra and SQL, 24 Nov 2017, changed values)

Given the relation:

Country(continent, country_name, capital, cities_above_milion, area, population, sea_access)

Question 4a. write a relational algebra query that returns, for each continent with more than 30 countries, the total combined cities with population above one million with an access to the sea (ocean). The query should return tuples of the form(continent, totalCities). (3p.)

\[ \pi_{\text{continent, totalCities}} (\sigma_{\text{seaAccess} \& \text{countryCount} > 30} (\gamma_{\text{continent}, \text{Count}(*) \rightarrow \text{countryCount}} (\text{SUM(citiesAboveMilion} \rightarrow \text{totalCities} (\text{Country})))) \]

In SQL:

```
SELECT Continent, SUM(CitiesAboveMilion) as TotalCities
FROM Country
WHERE Sea_access
GROUP BY Continent
HAVING COUNT(*) > 30;
```

This solution has several problems.

- In relational algebra is the result of the aggregation because you first sum and afterwards try to filter.
- In the SQL that the condition in the WHERE and the HAVING are evaluated in the wrong order.
\[ \pi_{\text{continent, totalCities}} \left( \sigma_{\text{countryCount}>30} \left( \gamma_{\text{continent, COUNT(*)}} \ni \gamma_{\text{continent, SUM(CitiesAboveMilion)}-\text{totalCities}} \right) \right) \] 

In SQL:

SELECT Continent, TotalCities FROM -- \( \pi_{\text{continent, totalCities}} \) --

\( \gamma_{\text{continent, COUNT(*)}} \ni \gamma_{\text{continent, countryCount>30}} \)

GROUP BY Continent HAVING COUNT(*) > 30 AS C

NATURAL JOIN -- \( \bowtie \) --

\( \gamma_{\text{continent, SUM(CitiesAboveMilion)}-\text{totalCities}} \)

(SELECT Continent, SUM(CitiesAboveMilion) as totalCities FROM

(SELECT * FROM Country WHERE SeaAccess) AS D

-- \( \sigma_{\text{seaAccess}} \) --

GROUP BY Continent)

**Exercise 5** (exam 2015 + exercise session 2 2018)

Now, going back to flights table from Exercise 1. Now, let’s assume a the following schema.

Airports(code ,city)

FlightCodes(code, airlineName)

Flights(departureAirport, destinationAirport, departureTime, arrivalTime, code )

departureAirport \( \rightarrow \) Airports.code

destinationAirport \( \rightarrow \) Airports.code

code \( \rightarrow \) FlightCodes.code

**Question 5a** : Below you can find an SQL query that finds all airports that have departures or arrivals (or both) of flights operated by Lufthansa or SAS (or both), based on the second schema in the description of exercise 5. Express this query by a relational algebra expression. (5p)

```sql
SELECT DISTINCT served 
FROM (( SELECT destinationAirport AS served, airlineName 
FROM FlightCodes 
JOIN Flights 
ON Flights.code = FlightCodes.code) 
UNION 
(SELECT departureAirport AS served, airlineName 
FROM FlightCodes 
JOIN Flights 
ON Flights.code = FlightCodes.code)) AS D 
WHERE D.airlineName = 'Lufthansa' OR D.airlineName = 'SAS';
```
\[ \pi_{\text{served}}(\sigma_{\text{D.\_airlineName}='lufthansa' \text{ OR } \text{D.\_airlineName}='SAS'}(\rho_{\text{D}}(\pi_{\text{served.\_airlineName}}(\rho_{(\text{served/destination\_Airport})(\text{FlightCodes} \bowtie_{\text{Flights.\_code=FlightCodes.\_code}} \text{ Flights)})) \cup \pi_{\text{served.\_airlineName}}(\rho_{(\text{served/departure\_Airport})(\text{FlightCodes} \bowtie_{\text{Flights.\_code=FlightCodes.\_code}} \text{ Flights)})))))\]

**Question 5b**: Translate the following relational algebra expression to an SQL query: (3p).

\[ \pi_{\text{First.\_depTime, Second.\_arrivalTime}}(((\sigma_{\text{First}}(\text{Flights})) \bowtie_{\text{First.\_destination\_Airport} = \text{Second.\_departure\_Airport}}(\sigma_{\text{Second}}(\text{Flights})))\]

**Answer:**

```
15     SELECT First.\_depTime, Second.\_arrivalTime
16     FROM Flights AS First
17     JOIN Flights AS Second
18     ON First.\_dest\_Airport = Second.\_dep\_Airport ;
```