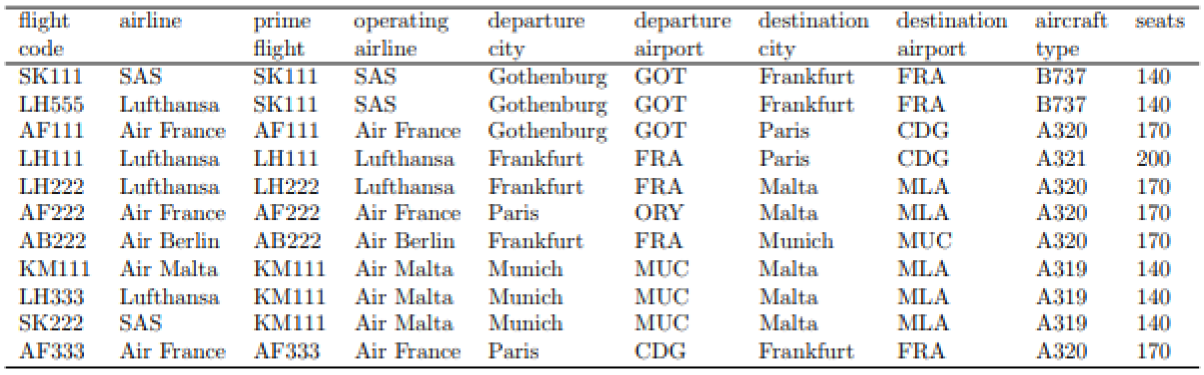
Databases (TDA357, DIT620)

Exercise 3 Theory topics: relations and dependencies 14/2 Wed

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.



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

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)
primeFlight -> operatingAirline brings another relation if considered

R4(_flightCode, airline, primeFlight, operatingAirline, departureAirport,
destinationAirport, aircraftType)
departureAirport -> R3.departureAirport
destinationAirport -> R2.destinationAirport
aircraftType -> R1.aircraftType

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 -> E

$\{AC\}+ = \{ACBE\}$

$R1(A, C, B, E)$

$R2(\underline{A, C}, D, F, G)$

$A, C -> R1.(A, C)$

Decompose R2 on FA -> B

$\{D, F\}+ = \{DFA\}$

$R21(D, F, A)$

$R22(\underline{D, F}, C, G)$

$F, A -> R21.(D, F)$

Key of R22 is DFCG

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

AC -> 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)

Company	Produces	SellsTo
Company A	Milk	ICA
Company A	Milk	Netto
Company A	Yogurt	ICA
Company A	Yogurt	Netto

Key (company, produced, sellsTo)

MVD company ->> produced

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

Not in 4NF. Decomposition:

Company	Produces
Company A	Milk
Company A	Yogurt

Key (company, produced)

Company	Produces
Company A	ICA
Company A	Netto

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.)

$$\pi_{continent,\ \ totalCities}(\sigma_{seaAccess \& countryCount>30}$$

$$(\gamma_{continent, Count(*)\rightarrow countryCount, SUM(citiesAboveMilion\rightarrow totalCities}(Country)))$$

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.

$$\pi_{continent,\ \ totalCities}\left(\left(\sigma_{countryCount>30}\left(\gamma_{continent,\ \ COUNT(*)\rightarrow continentCount}(Country)\right)\right)\right.$$

$$\left.\bowtie \left(\gamma_{continent, SUM(CitiesAboveMilioin)\rightarrow totalCities}\left(\sigma_{seaAccess}(Country)\right)\right)\right)$$

In SQL:

SELECT Continent, TotalCities FROM --$\pi_{continent,\ \ totalCities}$

-- $\gamma_{continent,\ \ COUNT(*)\rightarrow continentCount}(Country)$--

(SELECT Continent AS ContinentCount FROM Country

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

NATURAL JOIN -- ⋈

--$\gamma_{continent,SUM(CitiesAboveMilioin)\rightarrow totalCities}$ –

(SELECT Continent, SUM(CitiesAboveMilion) as totalCities FROM

(SELECT * FROM Country WHERE SeaAccess) AS D --$\sigma_{seaAccess}(Country)$

GROUP BY Continent)

<u>Exercise 5</u> (exam 2015 + exercise session 2 2018)

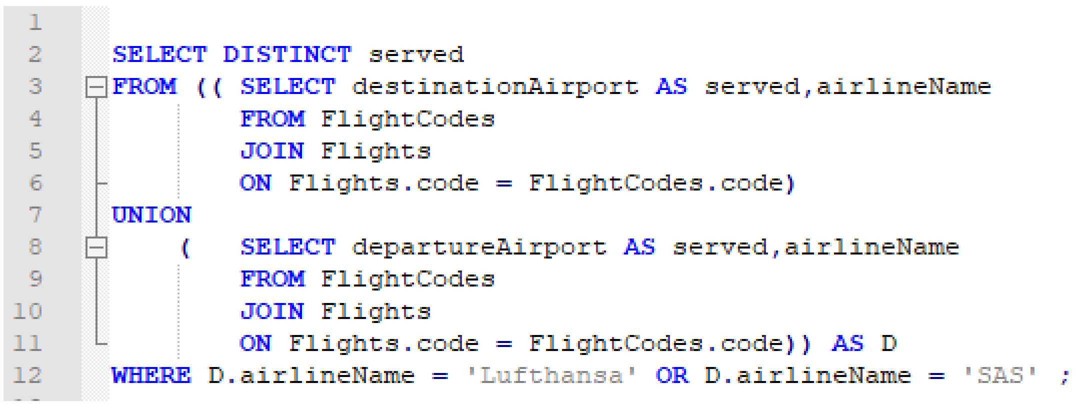
Now, going back to flights table from Exercise 1. Now, let's assume a the following schema.

Airports(code ,city)

FlightCodes(code , airlineName)

Flights(departureAirport, destinationAirport, departureTime, arrivalTime, code)
departureAirport → Airports.code
destinationAirport → Airports.code
code → FlightCodes.code

<u>Question 5a</u> : 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)



```

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' ;
```

$\pi_{served}(\sigma_{D.airlineName='Lufthansa'\,OR\,D.arilineName='SAS'}(\rho_D($

$\pi_{served,airlineName}\left(\rho_{(served\ /destinationAirport)}\left(FlighCodes\ \bowtie_{Flights.code=FlightCodes.code} Flights\right)\right)$

$\cup$

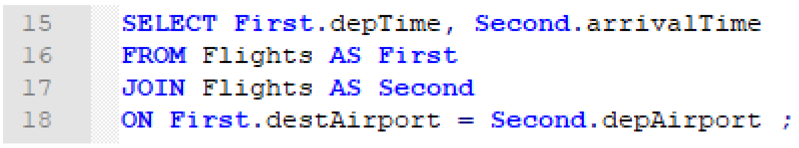
$\pi_{served,airlineName}\left(\rho_{(served\ /\ depeartureAirport)}\left(FlightCodes\ \bowtie_{Flights.code=FlightCodes.code} Flights\right)\right)$

))

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

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

Answer:



```
SELECT First.depTime, Second.arrivalTime
FROM Flights AS First
JOIN Flights AS Second
ON First.destAirport = Second.depAirport ;
```