TDA357/DIT621 – Databases

Lecture ?? – Super-off-the-books exam revision session

If you can read this, you are sitting close enough the board

Please have all three 2019-mm-dd exams available
Start looking at 2019-01-16 question 1

Jonas Duregård

Exam prep!

- Remember: You are allowed to bring one A4 of hand-written notes to the exam
- You will also be given a standard reference that you can find on the course page
  - The reference is printed with the exam, you are not allowed to bring it

Exam structure

- Fairly predictable pattern that will continue this year
- Six questions
  - ER
  - FDs+MVDs+NFs
  - SQL
  - RA
  - Triggers, constraints etc. in SQL DDL
  - Misc. topics including JSON

Curveball questions

- Last years questions didn’t have every possible topic
- For instance MVD/4NF and transactions where not on the exams
- Maybe that means you can safely ignore them this year?
- Having one or both of those this year would be a bit of a curveball
- Is Jonas the kind of person to throw those kinds of curveballs?

ER (From 2019-01-16)

a) Your task is to make an ER-diagram for the database of a gym company, managing their facilities and their customer records.

The database should contain a set of current and past customers. Each customer has a name and an email-address. Not every customer is a current member, and the database should keep track of which customers are currently members, and when their membership expires.

Each gym facility has a city, an address and a name. Two facilities can have the same name, but only if they are in different cities. Gym facilities can be established in any city across the world (but you can assume cities have unique names).

The database should also keep a record of times when each customer has accessed any gym facility. This may include multiple accesses from the same customer to the same facility at different times.

And now a few words from ours sponsors

At this point you're probably thinking "*Wow, it sure is nice of Jonas to organize this extra exam prep session, if only there was something we could do for him*"

Well – there is! You can answer the course survey (after it opens that is)

Your opinions (positive and negative) help me improve this course!

So please answer the course survey!

FDs (2019-08-29)

Consider a relation R(A, B, C, D, E) with the following Functional Dependencies

A -> B

B -> A

B, D -> E

B, C -> E

E -> A

c) Decompose the relation into BCNF. You only have to provide the final schema, not all the steps taken to compute it. If you do include all the steps, be sure to clearly indicate which relations are actually in the final schema (as opposed to intermediate relations).

MVDs

- Give an example of a domain+relation with MVDs but no FDs
- Normalize it to 4NF

SQL

- You know this stuff

Relational Algebra (From SQL of 2019-06)

Below is the schema for a database of users sending messages to each other. Readtime is either a time when the message was read by the receiver, or null if it has not been read yet (no other attributes can be null). Logintime is the last time the user was logged in.

Users(<u>username</u>, email, logintime)

Messages(<u>sender</u>, <u>receiver</u>, content, <u>sendtime</u>, readtime)
sender -> Users.username
receiver -> Users.username

a) Write an S RA Exp y that finds sender username, sender email, and content of each unread message sent to the user 'admin', with the oldest message first in the result.

b) Write an SQL query that calculates the average time between reading and sending messages (for messages that have been read). Assume sendtime and readtime are timestamps or similar, so the expression `readtime-sendtime` gives the time between sending and reading (if the message has been read).

Constraints - quickround

- What is the best way to ensure that a column is e.g. the sum or the average of something else?
  - Answer: Use a view! If a column value is completely decided by other values, there is no point in allowing the value to be modified at all, so placing it in a view and computing it from other values makes sense.

Constraints – Bonus question!

- Suppose we have:
  Teams(teamName)
  Players(team, number, name, age)
- What is the best way to ensure the following: "If a team name is modified, the team value of all players in that team should be modified as well"
  A view, constraint or trigger?
  - Answer: A constraint, specifically a reference constraint in Players with ON UPDATE CASCADE

Your task: The database contains dots in a two-dimensional plane and connections between those dots. The external interface for queries should look like this (constraints not included):

```
Dots(x_pos, y_pos, idnr, radix)

Connections(from_idnr, to_idnr)
```

This interface can include views and tables, and additional views and tables may be created as needed. Each row in `Dots` is a dot, with a position (`x_pos, y_pos` are integer coordinates), an identification number and a radix (see below). Each row in `Connections` represents a line from one dot to another, identified by their id-numbers.

Implement the following constraints in your design. Put letters in the margin of your code indicating where each constraint is implemented (possibly the same letter in several places):

a) There is at most one dot on any position (x,y-pair) and each dot has a unique idnr.
b) Dots can only connect to valid dots (those that are in `Dots`), and dots cannot be connected to themselves. Also there cannot be multiple lines to/from the same dots.
c) The radix of a dot should be equal to the number of dots it is connected to.
d) The radix cannot exceed eight. Attempting to add additional connections should fail.
e) All connections are bi-directional, meaning if there is a connection from point A to point B, there must also be a connection from point B to point A.
f) If a dot is deleted, all its connections should be automatically deleted as well.

JSON

- Make sure to learn the JSON syntax
  - Relatively easy points for things like translating a table to JSON
- Hopefully improved this year since JSON was in the labs

- Also: Understand JSON path. It consists of a small set of operators, but understanding the concept is perhaps not trivial.

”Other topics”

- Security (SQL injection)
  - You have seen this in the labs
- Transactions
  - Phenomena (Dirty reads, non-repeatable read, phantoms)
  - Isolation levels (Read uncommitted/committed, repeatable read, serializable)
  - Things like “show how programs that do this can fail horribly”
    - Involves showing interleavings of queries in two or more transactions