# Databases Re-exam

TDA357 (Chalmers), DIT620 (University of Gothenburg)

25 August 2016 at 14:00 in M

Department of Computer Science and Engineering

Examiner: Aarne Ranta tel. 031 772 10 82.

Results: Will be published by 15 September.

Exam review: upon agreement with the teacher.

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 an 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: You can answer in English, Danish, Dutch, Finnish, French, German, Italian, Norwegian, Spanish, or Swedish (in this exam; next time it can be another set of languages ;-) 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. If you need many pages for one question, number the pages as well, for instance, Question 3 p. 2. Write the question number on every 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 PostgreSQL. If you use any other variant (such as Oracle or MySQL), say this; but full points are not guaranteed since this may change the nature of the question.

## 1 Modelling (12p)

The domain to model is football teams and matches between them:

- A team has a name, which identifies it uniquely.
- A team has several players, and each player has a number, a name, and an age.
- A player is uniquely identified by his/her team and number (but players' names need not be unique).
- A match is between two different teams, one of which is “home” and the other “guest” team. Both teams have some number of goals scored.
- Every match has a set of players from both teams participating in it.

**a**. Draw an Entity-Relationship diagram for this domain. **Do not use multi-valued attributes**. (7p).

**c**. Show the corresponding database schema in SQL (CREATE TABLE statements), including all constraints and other features of the domain model that your E-R model does not express. (4p)

**c**. Give a list of those constraints that the E-R model and SQL cannot express. If there are no such constraints, just say that there aren't. (1p)

## 2 Functional dependencies and normal forms (8p)

Consider a relation $R(A, B, C, D)$ with the functional dependencies $A \to B$ and $B \to C$.

**a**. Give a real example of attributes $A, B, C, D$ with just these dependencies. Justify your answer! (1p)

**b**. Show all derived functional dependencies, as well as keys. (3p)

**c**. List the BCNF violations of the relation. Show all possible decompositions leading to BCNF and the sets of relations resulting from them. Will the resulting sets of relations be always the same? (4p)

## 3 SQL queries (12p)

Consider the relation `Players(team,number,name,age)`.

**a**. Write an SQL query that returns the name and age of every player whose age is under 25. (3p)

**b**. Write an SQL query that returns the name and age of every player whose age is higher than the age of player number 11 in the team named “IFC”. (4p)

**c**. Write an SQL query that returns the name and age of every player whose age is higher than the average age of all players in the team where he/she plays. (5p)

## 4 Relations (8p)

**a**. Write the query of Question 3b in relational algebra. (4p)

**b**. Consider the following tables:

```
PlayerAges:                PlayerTeams:

  name   | age               name   | team | number
--------+-----             --------+------+--------
 Arnold |  11               Ben    | IFC  |     15
 Ben    |  12               Cicero | GAS  |     16
                            Donald | GAS  |     12
```

Show the results of each of the following queries:

```
SELECT * FROM PlayerAges NATURAL JOIN PlayerTeams ;

SELECT * FROM PlayerAges FULL OUTER JOIN PlayerTeams USING (name) ;

SELECT * FROM PlayerAges INNER JOIN PlayerTeams ON (age = number) ;
```

(4p)

## 5 Constraints, triggers, and views (12p)

We continue with the football domain, now looking at tournament records. In a round-robin tournament (such as the inital stage of FIFA world cups), every team plays with every other team. The results of the matches can be gathered in the table

```
Matches(team1, team2, goals1, goals2)
```

The outcome of the tournament can be gathered in the table

```
Results(team, matches, goals_scored, goals_conceded, points)
```

where the winner of each match gets 3 points and the loser 0 points; in case of a draw, both teams get 1 point each. ("Goals conceded" means goals scored by the opposing team.) To give an example, we may have three matches with the outcomes

```
GAS-HACK  1-0  (giving 3 to GAS,  0 to HACK)
IFC-GAS   2-2  (       1 to IFC,  1 to GAS)
HACK-IFC  5-1  (       3 to HACK, 0 to IFC)
```

This leads to the following Results table:

```
 team | matches | goals_scored | goals_conceded | points
------+---------+--------------+----------------+--------
 GAS  |       2 |            3 |              2 |      4
 HACK |       2 |            5 |              2 |      3
 IFC  |       2 |            3 |              7 |      1
```

**a**. From the above description, it is clear that the Results table is redundant, because it can be computed from the Matches table. There are two ways to do this:

1. By writing a trigger that updates the Results table after each insert in the Matches table.
2. By implementing Results as a view on Matches.

Present *one* of these solutions, whichever you find better and/or easier for you. You need not care about the order of the rows in the Results table yet: it will be fixed in the (b) part of this question. (9p)

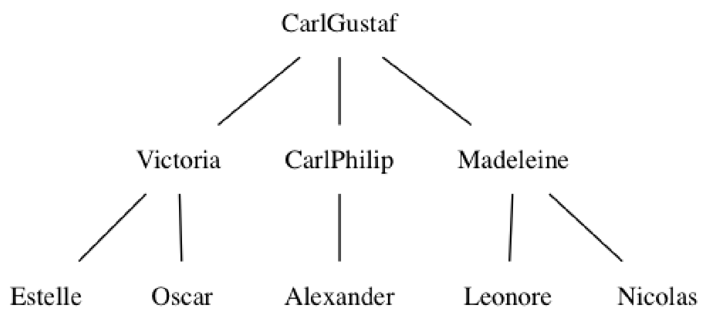
**b**. Assuming a Results table as above, write a view that orders the teams by the following rules:

1. The total number of points (larger ones first).
2. If the points are the same, the goal difference (i.e. goals scored minus goals conceded).
3. If still the same, the goals scored.

(More rules exist for breaking remaining ties, but we don't require them here.) (3p)

## 6 XML (8p)

A **rose tree** is a tree whose every node has a label and branches to zero or more rose trees. Rose trees can be used for many purposes, for instance, syntax trees and family trees (showing descendants but ignoring spouses). Here is an example of a family tree, starting with a node for the present king of Sweden and showing his children and grandchildren:



**a**. Design a DTD for representing rose trees and nothing but rose trees. The labels of the branches nodes should carry a value that can be any string (`#PCDATA`). (3p)

**b**. Show an XML element representing the above example tree, and which is valid according to your DTD. (3p)

**c**. Write an XPath query that returns all labels of a rose tree. For the above example, it should return just the names of all persons included. (2p)