UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of science and technology

EXAMINATION QUESTION PAPER

Exam in:	INF-2700 Database Systems
Date:	Thursday 30.11.2017
Time:	09:00 - 13:00
Place:	Adm.bygget, B.154
Approved aids:	None
Type of sheets (squares/lines):	Digital exam
Number of pages incl. cover page:	7
Contact person during the exam:	Weihai Yu
Phone:	41429077

NB! It is not allowed to submit scratch paper along with the answer sheets. If you do submit scratch paper, it will not be evaluated.

Question 1 (40%)

Below are some database tables with example data for a football application.

- Teams

tid	name
t1	team1
t2	team2
t3	team3

- Players

  The player `pid` with `name` plays for team `tid`.

pid	name	tid
p11	ola	t1
p12	odin	t1
p21	per	t2
p22	peter	t2
p31	martin	t3
p32	markus	t3

- Matches

  The match `mid` between two teams `tid1` and `tid2` was played on `date` at `start` time.

mid	tid2	tid2	date	start
m1	t1	t2	2017-10-01	18:00
m2	t3	t2	2017-11-01	18:00
m3	t3	t1	2017-11-20	17:00

- Goals

  The goal `gid` in match `mid` was scored by player `pid`.

  A `true` value of attribute `own` indicates an own-goal. An *own-goal* occurs when a player (mistakenly) sets the ball into the goal of the player's own team, resulting in a goal being scored for the opposition.

gid	mid	pid	own
g1	m1	p11	false
g2	m1	p21	true
g3	m1	p22	false
g4	m1	p11	false
g5	m3	p11	false
g6	m3	p12	false
g7	m3	p11	false
g8	m3	p12	false

The *primary keys* of the tables are in **bold** text.

Foreign key in Players:

- `tid`: references `tid` of Teams

Foreign keys in Matches:

- `tid1`: references `tid` of Teams
- `tid2`: references `tid` of Teams

Foreign keys in Goals:

- `mid`: references `mid` of Matches
- `pid`: references `pid` of Players

Write queries to find the required information.

Queries 1–5 must be formulated in *both relational algebra and SQL*.

Queries 6–10 need only be formulated in *SQL*.

Note: In the result tables of your SQL queries, there should be *no* identical (duplicate) rows.

Relational algebra *and* SQL (1–5):

1. List of different names of all players.
   The result for the example database is:

name
ola
odin
per
peter
martin
markus

2. Dates and start time of all `t1`'s matches.
   The result for the example database is:

date	start
2017-10-01	18:00
2017-11-20	17:00

3. Names of all players of `team1`.
   The result for the example database is:

name
ola
odin

4. Matches without goals. For each match, show the names of the teams, date and start time.
   The result for the example database is:

name	name	date	start
team3	team2	2017-11-01	18:00

5. Names of players who *only* made own-goals.
   The result for the example database is:

name
per

6. Number of teams.

   The result for the example database is:

numberOfTeams
3

7. List of team names and the numbers of players of the teams,

   The result for the example database is:

name	numberOfPlayers
team1	2
team2	2
team3	2

8. List of players who scored at least two goals (own-goals are *not* counted).

   The list should show the names of the players and the numbers of goals the players scored. The players are listed in the descending order of the numbers of goals they scored.

   The result for the example database is:

name	numberOfGoals
ola	4
odin	2

9. The matches and scores of `team1`.

   The scores include the goals scored by the team and the own-goals made by the opponent team.

   The result for the example database is:

mid	scores
m1	3
m3	4

10. Names of players who scored in all matches of the team (own-goals are *not* included).

    The result for the example database is:

name
ola

Question 2 (20%)

Now consider the physical data organization of the database in Question 1.

In the questions below, we will focus on queries like this one:

```
SELECT t1.name, t2.name
FROM   Matches m, Teams t1, Teams t2
WHERE  date = '2017-11-30' AND t1.tid = m.tid1 AND t2.tid = m.tid2;
```

The tables involved in the queries are organized as below:

- Table `Teams` is organized as a B^+-tree on attribute `tid`.
- Table `Matches` is organized as a B^+-tree on attribute `date`.

Answer the following questions.

1. What is the primary performance overhead of database systems in general?
2. Describe the file structure of the `Teams` table.
3. Sketch an *execution plan* of the above query.
4. What is the performance overhead of your execution plan?

Question 3 (20%)

Answer the following questions. Please explain the relevant concepts while answering the questions.

1. What is *functional dependency* $X \rightarrow Y$ of a relation instance r?

   For the relation instance below, check if the following functional dependencies are satisfied. If your answer is "no", explain why.

A	B	C
x	1	t
x	2	t
y	3	u
z	4	u

   a) $A \rightarrow B$

   b) $A \rightarrow C$

   c) $AB \rightarrow C$

   d) $AC \rightarrow B$

2. What is a *superkey* of a relation schema?

   Can you define a superkey using functional dependencies?

3. What is a relation schema in *Boyce-Codd Normal Form* (BCNF)?

4. We have a relation schema $Addresses\ (stname, stnr, postcode, city)$, where $stname$ stands for "street name" and $stnr$ for "street number".

   The $Addresses$ schema has the following functional dependencies:

   - $\{stname, stnr, city\} \rightarrow postcode$
   - $postcode \rightarrow city$

   The $Addresses$ schema is *not* in BCNF. Why? What is the problem with not being in BCNF?

5. How would you solve the problem?

6. Does your solution introduce any new problem?

Question 4 (20%)

1. What is an *ACID transaction*?

2. Is the following transaction schedule *serializable*? Explain why.

$$read_1(x), read_2(y), write_1(x), read_2(x), write_2(x), commit_2, commit_1$$

3. Is the above transaction schedule *strict*? Explain why.

4. Describe a concurrency control mechanism that enforces serializable and strict transaction schedules.

–END–