3. Experiment: ER-model, integrity, nested tables, recursion

Notes

We would like to remind you that you can always ask us if there are ambiguities in the exercises.

Exercise 3.1 (ER-Modelle; 6+8+9+17 P.)
You will find three ER-diagrams in your repository. The scenario is a simplified version of the scenario of exercise 1.1. The main aspects follow:

- The Bundesliga (first and second) is a double round-robin tournament.
- In the first and second Bundesliga participate 18 teams.
- There are teams being promoted or relegated directly and teams that have to play relegation matches. A relegation round consists of first and second leg and has always a winner.
- The DFB-Cup is a single-eliminatin tournament.
- Clubs cannot play against theirself although they might have more than one team.

a) Does each model express the given scenario properly (in the sense that no valid instance is excluded)? Give a short answer.
b) Describe the main differences between the diagrams 1 and 2 and between 2 and 3 with a few sentences.
c) Describe an instance for each model that is not valid and that is also not valid in the football scenario. Then describe for each model a valid instance that is not valid in the football scenario, however.
d) Transform the ER-models into relations and create the corresponding tables. Use integrity constraints that are part of the table definition but no triggers.

Note: Because of the deadline extension for exercise 1.1 you will get the diagrams after the deadline or on handing in your solution.

Exercise 3.2 (Definition von referentiellen Integritätsbedingungen; 30 P.)
The MONDIAL schema does not contain referential integrity constraints. Write a script to create the database with referential integrity constraints (extend `create.sql`). Therefore write a script `add-constraints.sql` where you add the integrity constraints with `ALTER TABLE`. You may also change the table definitions instead. Additionally, create a script `drop-constraints.sql` that deletes these constraints.

Consider *all* constraints that are useful for the MONDIAL-scenario. Some examples follow:

- If a country or province is deleted, everything which belongs to that country/province should also be deleted. The same holds for all data that is only relevant in ccorrelation with some continent/city etc.
- A city cannot be deleted if it is a capital of some country which is not deleted, or if it is the seat of some organization which is not deleted.
- An organization can only be deleted if it has no members.
- Neighborhood relationships are deleted if one of the participating objects is deleted.
- When deleting some mountain (or similar objects), also their geogaphic location is deleted.
- There must not be any cascading deletion of information which is independent from the delete issued by the user.

Hints: Consider the fact that during the database generation, referential integrity constraints can be temporarily violated. The database schema is removed by the script `drop-mondial.sql` (removes all MONDIAL-tables). The MONDIAL-schema *without* integrity constraints is generated by `create-mondial.sql`.

Exercise 3.3 (Nested Tables; 20 P.)
a) Create a table *Country_n* which contains all columns of *Country* and additionally contains the information about spoken languages and religions for each country as nested tables. Fill the new tables with the respective data. **Hint:** Only one insert statement is required. Use `cast` and `multiset`.
b) Write a query over this table that returns all countries where German is spoken. The part of the population that speaks German should be given in percent beside the country name.

c) Retrieve the execution plan for the query in part b). Which conclusions can you draw from it about the actual organization of the table in a)? Does the application of nested tables make sense here? Are there use cases where nested tables fit better for?

Hint: The indications about illegal or wrong queries w. r. t. nested tables in the lecture notes are related to Oracle 8 and do not apply in that form to Oracle 11.

Exercise 3.4 (Transitive Hülle, hierarchische Anfragen; 20 P.)

a) Compute all tributary rivers of the river *Zaire*, i.e., all rivers that flow either directly or indirectly into the Zaire. Use an auxiliary relation that collects rivers. Do *not* use ORACLE's `CONNECT BY PRIOR`. Compute the length of the whole river system.

b) Now compute the river system by using `CONNECT BY PRIOR` (see "Hierarchical Queries" of ORACLE's "SQL Reference").

c) Discuss the possibilities to retrieve the seas connected with the North Sea (so to say reachable by ship) by using `CONNECT BY PRIOR`.

d) Now use `WITH ... SELECT ...` (subquery factoring clause) to compute the river system.

Exercise 3.5 (Komplexe Attributtypen; 5 P.)

Specify the nearest city for every mountain in Russia. Use the coordinates for the distance calculation (simplification: use the Pythagorean theorem).

Hint: The city may be in a neighboring country.

Exercise 3.6 (Optimierung, 10 P.))

Count towns, cities and megacities. Places with less than 100 000 inhabitants count as towns, places with 100 000 or more but less than 1 000 000 inhabitants as cities. The remaining bigger places fall into the third category. Accessing the secondary storage is expensive. Therfore write a query that reads the *City*-table only once to answer the query. **Hint:** You can see how often a table is read in the execution plan. Do not use PL/SQL.

Deadline: 9.6.2010, 11h