5. Experiment: View Updates, integrity, indexes, PL/SQL

Exercise 5.1 (View Updates; 20 P.)
Use a database without referential integrity constraints to ensure that possible updates do not fail due to the constraints.

a) Define a view that contains for all countries the important attributes from the tables *Politics*, *Economy* and *Population*, and the (derived) population density. Formulate a query over the view that computes the GDP/inhabitant, the infant mortality, and the population growth, ordered by GDP/inhabitant.
   Find out for which columns updates are allowed and justify this. Update the database so that it contains the estimated population for the next year later.

b) Define a second view that additionally contains the continent where most of the country belongs to.
   Again, find out for which columns updates are allowed and justify this. Try to change the database to make Russia an african country.

Exercise 5.2 (Integrität; 15 P.)
You'll find an ER-diagram (`diagramm_4.pdf`) and an a script (`diagramm_4.sql`) to create an according database schema in your repository. Extend the schema by the following constraints (always as simple as possible, e.g., rather with CHECK-constraits than with triggers). When possible, change only the given table definition.

- Only participants of the same season (Wettbewerbszeitraum) can hold matches.
- The round of the game must also belong to the season of the participants.
- Clubs cannot play against theirselves.
- Every half-time score must be less than the final result.

Hinweis: Change the original script in this and the following exercises. Pay attention to its executablility. If necessary, adapt the insert statements, too.

Exercise 5.3 (Indexe; 5 P.)
The indexes in the schema of exercise 2 are not always well chosen (there is always an index for primary keys and UNIQUE constraints). Improve the indexes in order to make joins between parent and child tables more efficient. If necessary, you can also add new indexes.

Exercise 5.4 (Universelle Relation für Obertyp; 15 P.)
Extend the table *Spiel* into a universal relation for the supertype (cf. *Spielrunde*). Remove the table for the subtypes of *Spiel* and adjust the views. Make sure that tuples in *Verlängerung* can still reference only matches of the type *Entscheidungsspiel* although the foreign key references *Spiel* directly now. **Hint:** You have to change only the table definitions of *Verlängerung* and *Spiel*.

Exercise 5.5 (Disjunktheit Untertypen; 15 P.)
The table *Wettbewerb* cannot be extended into a universal relation due to the different attributes of the subtypes without permitting null values for the inappropriate attributes of the different subtypes. Therefore keep the given table structure but extend it in a way such that it will be guaranteed that the extensions of the subtypes are pairwise disjoint.

Exercise 5.6 (Funktionsbasierte Indexe; 10 P.)
The constraints of the schema are such that there can be a first leg and a second leg between two participants in each season. However, the DFB-Pokal allows only one game. Write a function-based index that assures this uniqueness.

Exercise 5.7 (Cursor; 10 P.)
Write a PL/SQL procedure to compute the minimal amount of the world's population that is necessary to produce 50% of the world's GDP. For each country, give the number of people who contribute to the result. Assume that the GDP of each country is equally distributed over all its inhabitants. The solution should use a cursor on a suitable `SELECT`-Statement. You can insert the result into a table or write it on the screen with DBMS_OUTPUT.PUT_LINE().

Deadline: 7.7.2010, 11h