# 2. Experiment: SQL queries, Views, Updates

## Notes

If not stated otherwise the exercises should be solved with only one query without further aids (like views). Don't forget to *format* and to *comment* your solutions meaningfully.

**Exercise 2.1 (Update; 15 P.)**

a) Exchange the countries of the continents Europe and Asia by using *only one* update-statement. Do not use auxiliary constructs, PL/SQL, `DECODE`, `CASE`, etc. **Hint:** To some extent, the new value of the attribute *continent* has to be computed as function of the old one. Undo the changes afterwards.

b) Now use `CASE` to move European countries to Asia, Asian countries to America, and American countries to Europe also just with one update. Undo these changes, too.

**Exercise 2.2 (sym_borders; 10 P.)**

a) Write a query to check whether the relation *borders* is asymetrical. The output should be "yes" or "no".

b) Define a view *sym_ borders* that contains the symmetric closure of *borders*. This view can be used in all subsequent exercises.

**Exercise 2.3 (Sichten für Zwischenergebnisse; 15 P.)**

Calculate the population density of the region containing the countries Algeria, Libya and all of their neighbors. Compare the result with the population density of the region excluding the desert areas. Therefore you need to examine if there are deserts in mondial that are both inside and outside the above region. Use views to structure your solution reasonably.

**Exercise 2.4 (Redundanz; 10 P.)**

The information about habitants is stored redundantly: Both in *Country.population* and in *Province.population*. Test the data for inconsistencies:

a) Compute all countries having the sum over the population of their provinces being different by at least five percent from the number of habintants. Show the name, the absolute values, and the percental difference.

b) Compute all countries that exhibit inconsistencies among their population and the population of their cities. Here, all differences should be considered as faulty.

c) Find another possible contradiction regarding the population numbers which is still not considered and test the data for it.

**Exercise 2.5 (Urbanisierung; 20 P.)**

a) Find out whether there is a correlation between the degree of urbanization and the per capita income of a country. For that use the Pearson correlation coefficent $r$. Only the standard aggregation functions (`MIN`, `MAX`, `SUM`, `COUNT`, `AVG`) are allowed. The formula is:

$$r = \frac{\sum_{i=1}^{n}(x_i - m_x) \cdot (y_i - m_y)}{n \cdot s_x \cdot s_y}$$

Therein $x_i$ and $y_i$ are the values of the variables (here: degree of urbanization and per capita income), $s_x$ and $s_y$ their standard deviations plus $m_x$ and $m_y$ being their arithmetic means. There are positive (the more the more) and negative (the more the less) correlations. Regard an absolute value of $r$ greater than 0.1 as weak, greater than 0.3 as medium and greater than 0.5 as strong correlation.

**Notes:** The formula should be written as given in SQL computing $r$. Exclude countries that have a contradictory population value regarding their cities. For simplicity, assume that places with more than 10000 habintants are classified as urbanized.

b) Try to substantiate your findings with a few sentences.

**Exercise 2.6 (Durchschnittskoordinaten; 10 P.)**

Find out the average coordinates of the geographical distribution of the world population.

a) Calculate the average value of the coordinates of all cities at first.

b) Now try to compensate the uneven distribution of cities in the database: Compute the average value of the average coordinates of all cities per country weighted by the country's population. Hence you compute the avergage coordinates per country, weight them according to the country population and eventually compute the overall average.

**Exercise 2.7 (Mengenvergleich; 10 P.)**
Compute all pairs of european countries that are adjacent to the same set of seas. **Note:** Sets are equal if they contain the same elements. Because set comparision cannot be expressed directly in SQL the equality has to be determined using the set operators `UNION`, `INTERSECT`, and `MINUS` (in Oracle for *except*).

Deadline: 19.5.2010, 11h