# Exercise Sheet 11: Application Programming I (until Thursday, 18.01.2018) (40 Points)

**Please note:** you need **50%** of all exercise points to receive the *Studienleistung* for this lecture. In order to pass the RDB I Module, you need both the Studienleistung **and** you need to pass the exam. Exercises have to be turned in until **Thursday before the lecture** either in the lecture hall or into our mailbox at the IFIS floor (Mühlenpfordtstraße 23, 2nd floor). Please do not forget your **Matrikelnummer** and your **tutorial group number** on your solutions. **If you forget** to write your Matrikelnummer and/or your tutorial group number, you get **automatically 0 points**. Your solutions may be in German or English. Unless otherwise specified: **Always use your own words!**

## Exercise 11.1 (12 points)

Answer the following questions

1) What problems may arise with updatable views? (2 points)
2) Briefly explain each property of the ACID principle. (4 points)
3) Why is the ACID principle needed in databases? (2 points)
4) When a View is updateable? (2 points)
5) Why do we need indexes in databases? (2 points)

## Exercise 11.2 (10 points)

Consider the following schema:

**Location**(id, name, year, country)
**Contest**(name)
**Event**(contest → Contest, name, location→Location,date, sport_type)

The database schema corresponds to sport events that belong to a particular competition and take place at a specific location and date.

1. - Create the View UpcomingContests that contains all contests for which no events have yet been entered. (4 Points)

2. - Create a Materialized View that would be manually updated. The view must find the name and the competition of all events in the sport Bobsleigh that start in the same country at the same time. The View must have the name, location and sport type of these events. Write the Materialized View using the syntax of DB2 that was shown in the lecture and also write the same View using PostgreSQL syntax (6 Points).

## Exercise 11.3 (6 points)

Consider the following SQL statements:

```
CREATE TABLE cities(
        id INT NOT NULL PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        population INT NOT NULL
)

CREATE TABLE countries(
         id INT NOT NULL PRIMARY KEY,
         name VARCHAR(255) NOT NULL,
         continent VARCHAR(255) NOT NULL,
         surface_area REAL NOT NULL)
)
```

1) Create a View called "small_cities" that contains all the names and ids of the cities that have a population between 2500 and 3000. (2 points)

2) Create a View called "americas" that contains only tuples of the table countries that are located in America. (2 points)

3) Create an index on the attribute name of the cities table. (2 points)

## Exercise 11.4 (12 points)

Consider the following relation schema:

**Movie**(id, title, year)
**Person**(id, name, gender, birthday)
**Genre**(name, description)
**actor**(person → Person, movie → Movie, role)
**director**(person → Person, movie → Movie)
**reviewer**(person → Person, movie → Movie, stars)
**hasGenre**(movie → Movie, genre → Genre)

a) Write a SQL statement to find the name of all reviewers who have reviewed the movie "The Last Jedi". (2 points)
b) Provide SQL statements for creating all necessary indexes to speed up the SQL query from a). (4 points) **HINT**: remember that some attributes already have an index defined.
c) For each person (id, name) in the database create a view that can give the number of different roles the person has played. The result should also include those people who have never acted in any movie. (3 points)
d) Create a View that returns the titles of all movies directed by more than one person (3 points)