Practical work 2: "Design and implementation of an object-relational database for "domain""*

***(replace the word "domain" with the domain you were given, e.g.: "Design and implementation of an object-relational database for airport").**

This document is the definition of the first homework in the course "Technology of large database". **NOTE THAT YOU MUST FILL OUT GROUP APPLICATION TO GET YOUR VARIATION.**

Working with your domain description

You are given a domain description of 300-350 words. It contains the main entity sets (or object types) that must be stored as well as main functions that the system must do.

Please note that the domain description is not intended to be precise user requirements – instead consider it as a general description of the database to which you can add freely, create formulas for vague parts etc. You will be graded based on the minimal constructs created as well as the appropriateness of the constructs used and their complexity. Please note that the works that will not contain object types and will do simple relational design and implementation, will not get passing mark!

The purpose of the vagueness of the domain description is twofold: (1) you may adapt the complexity of the domain according to your study goals, and (2) to prevent the situations when two very similar solutions are submitted. With these domain descriptions, it is almost impossible to create two identical works unless some illegal communication has taken place (such as 2 groups fully working together).

Note that you do not have to implement every feature listed in the domain description.

Submissions

The deadline for submissions is December 4 with option to correct the work and submit it once more until December 15, or simply: December 15. Please note that if you wish to use emergency extension, you must send e-mail to mara.pudane@rtu.lv **before** the deadline. **You may only use the deadline if at least 2 of 3 group members have not used it.**

You will have to submit 3 things:

- report (with a reference to student's profile where the system is implemented and link to demonstration video – as mentioned before, this is to ensure everything runs) – must submit on ORTUS;
- commented *.sql file – must submit on ORTUS;
- group presentation video – all members must participate and explain at least the part that the student has worked on specifically.

For this work, you must work on the department server.

Report

You must submit a report in *.doc or *.docx format (*.pdf also may be added but only if you are working in other document editor than MS Word) that contains:

0. Your **student ID, name and surname as well as the title of the laboratory work**

1. The **ER diagram of the design.**

2. The **reference to student's Oracle server profile** where the system implemented (just write: the system is integrated in the X's profile).

3. The **link to the video demonstration** (either as a private video on, e.g., YouTube or via some file sharing service). The video demonstration (3-5 minutes) can be a recorded group call, e.g., in Zoom, where you show your created structures and explain more interesting parts of the design.

4. All the queries, their **semantic and technical explanations**.

5. **Group work report**: group work - how did you split the work? What was productive, what was not? What problems if any arouse and why?

6. **Conclusions** (this is extremely important part) that **must** include:

- as explained in the presentation after the previous practical work.

*Commented *.sql file*

A code with your comment for each query, with more complex/interesting parts pointed out.

The tasks in the practical work

1. Develop ER diagram of structures (it is expected you have 3-5 tables and correspondingly: 3-5 object types).

The *minimal requirements*:

- ERD: table with row type objects.
- ERD: table with column type objects.
- ERD: table with object collection.

The list of object types, their attributes and methods.

The *maximal requirements* (the requirements that will give you higher evaluations in this task – you do not need to do all of this to get maximum points, but here are some ideas to increase complexity):

- Object hierarchies designed.
- More tables with abovementioned constructs designed.
- Multilevel collections designed.

Tables must be related both, technically in a scheme as well as semantically (i.e., logically, by meaning). CASE tools or chart drawing tools must be used to create diagram.

2. Definition of object types for objects which are included in the tables, including object attributes and methods.

Minimal/maximal requirements correspond to the Task 1 – you must implement your design.

3. Definition of REF type attributes and subtypes to create links among objects.

Overall correspond to the Task 1 – you must implement your design.

Minimal requirement: REF attributes, *maximal requirement*: more complex structures, subtypes.

4. Creation of data storage structures.

Minimal/maximal requirements correspond to the Task 1 – you must implement your design.

5. Data input with INSERT command and optionally: with SQL*Loader

Minimal requirement: 5-7 rows per table or 3-5 object created. Note that complexity of INSERT statements will depend on the complexity of your table structures.

6. Queries with the Value(), Table(), DEREF() functions to demonstrate how information can be obtained

Minimal requirements: At least 2 meaningful queries with each function, so in total at least 6.
Maximal requirements: Please note that here logical complexity is evaluated. All queries do not have to be complex to get maximum grade. Complexity may also depend on your previously defined structures (e.g., if you use multi-level collections).

7. Create methods for object types.

The minimal requirements: use map/order methods (1 method) (method+ demonstration), use MEMBER methods (2 methods) (methods + demonstration).

The maximal requirements: non-trivial methods, that use more complex comparisons or do more complex calculations – i.e., give a couple more constructs in code.

8. Preparing the submission:

- writing a report (see above);
- documenting code (.*sql file) (see above);
- making sure all is integrated on at least one account;
- preparing video demonstration.