Prof. Dr. Stefan Brass
Institut für Informatik
MLU Halle-Wittenberg

October 31, 2019

Databases IIB: DBMS-Implementation — Exercise Sheet 2 —

Part a) and c) will be discussed in class, you only have to submit Part d). Please upload your solution into the StudIP file folder called “Hausaufgabe_2” in the StudIP entry of the lecture (with course ID `06fb9210013971fbdbb78ff1746b744e`). The deadline is November 5 (the day before the next lecture).

It is permitted to form groups of up two members, but please make sure that both members can fully explain all homeworks submitted by the group. The intention is certainly not that each member solves only half of the exercises.

Please upload only one file per group. I can unpack archives in `zip`, `tar` and `rar` formats. Please make sure that unpacking the archive yields a directory with your name(s).

Not all submitted homeworks will be corrected, but all homework exercises will be discussed in class. If you should have questions about your homework, please ask! A precondition for getting credit for this course is that you submit solutions to two thirds of the homeworks. Obviously wrong or very incomplete submissions do not count.

In-Class Exercises

a) What would you answer to the following questions in an oral exam?

- Please explain the concept of physical data independence. Name some reasons why it might be necessary that the physical design of a database is modified.
- Please explain the concept of transactions. What are the main properties the DBMS must ensure, and what are typical techniques for doing this?
- What are the modules/components/layers in a typical DBMS architecture? What are the steps in which an SQL query is processed?
- What are the advantages that the meta-data (e.g. the database schema) is stored as data in the system catalog (data dictionary)?
- What are typical tasks of a database administrator (DBA)?
- Name and explain at least one table/view of the Oracle data dictionary.
- Where can you get more information about tables/views of the Oracle data dictionary? Suppose you are looking for data dictionary tables containing information about privileges (access rights). How would you find the right table?
- What is the difference between `user_*`, `all_*`, `dba_*`, and `V$*` tables/views in the oracle data dictionary?

- How are tables/views inside an Oracle database identified (the table name alone is not sufficient if one looks at all tables stored inside an Oracle database)?
- How can I refer to a table of a different user in an SQL query in Oracle?
- What is a "synonym" in Oracle? What is the purpose of "public synonyms"?
- What are some differences between the languages C++ and Java? (If you did not learn Java before, you can replace some other programming language or concentrate on the specific features of the C++ language.) Discuss some positive and negative aspects of both languages if one wants to implement a DBMS.

b) Please read some information about "Adminer", a web-interface to different database systems.

[https://www.adminer.org/]

The web address of our lab database and account information will be sent to the participants via EMail. At the moment, the user account has not yet DBA rights (it is a normal Oracle user account). You will get DBA rights later.

Adminer has the menu item "SQL Kommando" on the left side near the top. If you click on this link, you get a text area where you can enter SQL commands. Click on "Ausführen" to execute the command.

In addition to the personal account, I created one test account to which all participants have access. In this way, you can experiment with access rights. Please do not change tables in this account. Send me an EMail if something does not work.

Note that the menu for databases in Adminer actually shows tablespaces. Adminer was originally made for other systems such as MySQL and somehow they mapped Oracle concepts to the existing menus.

I already installed the EMP-DEPT database in the test account and gave read access to `PUBLIC`. I also created the example tables from the course "Databases I". Your own account is empty, but Adminer permits you to "Import" SQL-Scripts such as

[http://users.informatik.uni-halle.de/~brass/dbi19/sql/empdept.sql]

Note that you must uncheck "Bei Fehler anhalten" ("Stopp on Error"), because the script contains `DROP TABLE` commands at the beginning for all tables that it later creates. This permits to execute the script repeatedly, e.g. when one has deleted some data. But on the first run it will give error messages.

You can get a list of all tables with

```
SELECT * FROM CAT;
```

Try this. Then do the following exercise.

c) This is not an official homework, but it will help you to repeat contents of the lecture.

Please write an SQL query that shows all foreign keys of tables owned by the test account. More precisely, the query should list foreign key columns (table, position in foreign key, column name) and the referenced key columns (table and column name).

Homework Exercises

d) Consider homework results as in the following table:

HOMEWORKS			
FIRST_NAME	LAST_NAME	EXERCISE_NO	POINTS
Ann	Smith	1	10
Ann	Smith	2	8
Michael	Jones	1	9
Michael	Jones	2	9
Richard	Turner	1	5

Please define a C++ class for objects that correspond to a single row in the table, i.e. it has an attribute for each of the four columns. The requirements are as follows:

- The column values are set in the constructor.
- The attributes should be private, so that they cannot be changed from outside the class.
- There should be four methods (without parameters) for read access to the attributes (`get`-methods). You can choose the names of class and methods yourself.
- You should not use the `string` class, but classic "C-like" strings (`char`-arrays). An important requirement is that the constructor copies the strings. It gets a `const char *` for the string-valued attributes, but that might point to an input buffer which is overwritten for the next line read from a file (we do reading from a file next week). Therefore, it does not suffice to store only the pointer in the object! You can assume that the names are not longer than 20 characters. If the constructor should be called with strings longer than this limit, it can simply cut off the remaining part (you may print a warning to `cout` when this happens, error handling will be discussed later). In no case array boundaries may be violated! If you prefer to allocate memory dynamically (with `new`), you can allocate as much as necessary, and copy also strings of length > 20. Do not forget to free dynamically allocated memory again (with `delete`) in the destructor!
- Write the class definition into a header file (`.h`), and the implementation of the constructor into the corresponding `.cpp`-file. You can choose yourself where you want to specify the implementation of the `get`-methods.
- Write a small `main`-Program to test your class. It should construct two objects and print the result of some calls to the `get`-Methods. The `main`-Program should be contained in the third source file.