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

October 19, 2017

Databases IIB: DBMS-Implementation — Exercise Sheet 2 —

Part a) to d) will be discussed in class, you only have to submit Part e) and f). Please send your solutions to the instructor via EMail: `brass@informatik.uni-halle.de` (with "`dbi17`" in the subject line). The official deadline is October 25, 12:00.

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) In the exercise group the access data for an "Oracle 11g" database will be announced.

Machine (IP):	`141.48.` _____ `.` _____
Oracle Password (SYS):	____________________________________
Web Admin:	`https://141.48.` _____ `.` _____ `:1158/em/` (if `https` should not work, try `http`)

The following environment variables must be set:

```
export ORACLE_HOME=/app/oracle/product/11.2.0/dbhome_1
export ORACLE_SID=orcl
export ORACLE_BASE=/app/oracle
```

Furthermore, the search path for commands (environment variable `PATH`) must contain the directory `$ORACLE_HOME/bin`. This can be done as follows:

```
export PATH=$PATH:$HOME/bin:$ORACLE_HOME/bin
```

You can enter these settings for instance in the `.bashrc` file which is executed for each interactive non-login shell started. (Usually the `.bash_profile` executed by login shells contains a command to read the `.bashrc`, too).

If you do not want to change your startup scripts, you can also write these settings into a file `setora` and execute that file with

```
. setora
```

("`.`" is a short notation for the "`source`" command built into the `bash` shell: It executes the commands in the file in the current shell, whereas normally a shell script is executed in a new shell.) A file with the above commands can also be downloaded from

[http://www.informatik.uni-halle.de/˜brass/dbi17/setora]

The command `printenv` shows the settings of the environment variables. Since there might be quite a lot of environment variables, you can list only the lines containing the substring "`ORA`" with

```
printenv | fgrep ORA
```

When you managed to log in and set the environment variables, you should be able to log into Oracle with the following command:

```
sqlplus SYSTEM
```

This account has DBA rights, so be careful! For some administrative commands, the following is needed:

```
sqlplus SYS as SYSDBA
```

Please check whether you can log in, this will be needed for a homework on the next exercise sheet.

If something does not work, you can check whether the Oracle server runs at all by looking for the background processes with the following command:

```
ps -ef | fgrep ora
```

When you are logged into SQL*Plus, you can get the list of all user accounts with

```
SELECT * FROM ALL_USERS;
```

Note that the end of SQL-commands (queries, updates, etc.) must be marked with a “;”. You can see a list of SQL*Plus commands (which are available in addition the the standard SQL commands) with “`? index`”. You can get more information on a specific command *c* with “`?` *c*”. SQL*Plus commands do not need a semicolon, they end at the end of the line. An important SQL*Plus command is “`quit`” to leave the program.

c) The server machine runs Linux (CentOS). Please name ten important Unix/Linux commands. Explain some differences between Windows and Linux.

d) Compile the prime number program from your first homework with `g++` (the GNU C++ compiler) on this Linux machine. A requirement for future programming exercises is that the source code can be compiled with `g++` under Linux, so you should check for portability early.

   You can copy a file between computers with

```
scp file user@machine:path
```

   If you use a Windows computer, you should get `PuTTY` for remote login. This contains a command `pscp` that can be used in the same way as `scp`.

Homework Exercises

e) Look at the Oracle Documentation webpage:

[http://www.oracle.com/pls/db112/homepage]

Name the titles of at least three manuals (that seem interesting to you). The purpose of this task is to get some overview on the available documentation.

f) 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, you can allocate as much as necessary, and copy also strings of length > 20.
- 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.