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

November 14, 2019

# Databases IIB: DBMS-Implementation
# — Exercise Sheet 4 —

Please read Part a) and b) and mark questions which you want to discuss in class. You only have to submit Part c) and d). Please upload your solution (a pure ASCII text file with the two SQL statements) into the StudIP file folder called "Hausaufgabe_4" in the StudIP entry of the lecture. The deadline is November 19 (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. Please upload only one file per group. Name the file such that it contains the names of all group members.

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.

## Some Feedback on Homework 2

- Please make sure that all submitted homeworks can be compiled. If you need help, clearly mark in the subject line of your email that you have a question (and include the error message of your compiler). I am quite generous with late homeworks, but I do not want to get homework submissions with compilation errors.
- Of course, the difference between returning a value (as the result of a method) and printing a value must be clear. The accessor methods for the attributes must return the value and print nothing.

  Probably most classes in a complex system should not print anything, since this reduces possible usages of the class. Unexpected output might e.g. destroy the format of a message returned from the server to the client. Furthermore, today most programs have a graphical user interface and it is not clear whether output sent to `cout` will be seen by anybody.
- Some students declared the accessor method for `firstname`/`lastname` as returning "`char *`" instead of "`const char *`". Since the method returns only the address of the array in the object, it would be simple for the caller to assign something and destroy the contents of the array. The whole point of having accessor methods and not making the attribute `public` was to protect the array and its contents. Note that if the method is declared as `const` (i.e. not changing the object state), one cannot give non-`const` pointers to object components away.



- Make sure that you understand that it is possible to write the method body already in the class declaration in the `.h`-file. This has the advantage that the compiler can replace the call to the method directly by its body (if it is not recursive and not too long). It has the disadvantage that the class declaration might become less readable because the human reader (in contrast to the compiler) needs only the interface of the method, not its implementation. Since the accessor methods contain only a single statement in their body (the `return` for the attribute value), this disadvantage is very small in this case.

  Note that you are still free to change the implementation of the method later. However, it would require recompilation of all code that used the class. If you only change a method body in a separate `.cpp`-file, it would suffice to recompile that file and link all the object files. But adding an attribute to the class would in the same way require recompilation of all source files that used the class, because more memory is now needed for the objects of the class. Therefore, there are anyway changes of the implementation of the class that need this recompilation.
- Part of the specification was to copy all the characters of the string in the constructor, not only the pointer. E.g. if you declare the the attribute in the object as "`const char *first_name;`" and the constructor parameter with the same name and type, then the assignment

  ```
  this->first_name = first_name;
  ```

  copies only the address of the array that contains the characters of the name, but not the contents. If this array is later reused, for instance when you read the next input line, then the attribute still points to the same array, but a different name is stored in it.
- One option is to declare an array of sufficient size as an attribute of the object:

  ```
  char first_name[21];
  ```

  Note that you need the array length 21 if you want to store names up to length 20, because an additional byte is needed for the null-termination of the string (end-of-string marker `'\0'`).
- The other option is to find out how long the name is and to allocate an array of the right size.

  ```
  int length = strlen(first_name);
  this->first_name = new char[length+1];
  ```

  If you did this, you should free the array in the destructor. Note that this has to be done with

  ```
  delete[] this->first_name;
  ```

  The simple `delete` (without `[]`) cannot be used for arrays.

- Note that the function `strcpy` from the standard C library is inherently unsafe: It copies all characters of the string, even if the array is too small, which means that other variables stored in memory after the array may be overwritten. Whereas in Java, arrays are objects and contain an attribute for their length, arrays in C++ are simply $n$ variables of the base type allocated in consecutive memory locations. E.g. a `char`-array of size 20 will be exactly 20 bytes long (`char` is one byte long, except possible on extremely strange machine architectures). Since `strcpy` gets only the start address of the array, and no information about its length, it is clear that it cannot check the boundaries. There is another function `strncpy` which has an additional parameter for the array length $n$, but unfortunately it will simply stop after copying $n$ characters, thus the null-termination of the string might be missing. Then future usages of the string, e.g. when printing the name, will violate the array boundaries because they print bytes until they find a null byte. However, if you first computed the length of the input string with `strlen`, and then allocated an array of the right size, `strcpy` and `strncpy` will be safe (until the contents of the input array changes).
- Note that good testing code in the `main` program would include a name that is longer than 20 charaters. Very few students did that.
- Many students repeated the loop for copying the contents of the input array for `firstname` and `lastname`. Because of the limit of 20 characters, this loop is not extremely simple and short. Therefore, this is a case of Copy&Paste programming which should be avoided. One should write an auxiliary `private` method for copying the string. The method must be declared in the class declaraction in the `.h`-file, but because it is private, it does not change the interface of the class.

## Repetition Questions

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

- What different types of files does Oracle use?
- What is the purpose of a control file?
- Explain why the insertion of a single row into a table can change multiple blocks in the database files.
- What is the purpose of the log (or redo log in Oracle)?
- What has to be written to disk when the DBMS executes a `COMMIT` statement? What does not have to be written to disk?
- Suppose there was a power failure and the contents of main memory was lost. Why are the log files essential even if we do not worry about ongoing transactions at the moment of the power failure?
- What does ARCHIVELOG mode mean in Oracle? Against what failure is one not protected if the database runs in NOARCHIVELOG mode?



- What has to happen before Oracle can overwrite a redo log file?

b) Check your knowledge of C++ by answering these questions:

- Which syntax is used for calling the constructor of a superclass in the constructor of its subclass?
- Consider the following example:

```
#include <iostream>
using namespace std;
class C {
    public: void p() { cout << "C\n"; }
};

class S : public C {
    public: void p() { cout << "S\n"; }
};

int main() {
    S obj;
    C* x = &obj;
    x->p();
}
```

  What does this program print? If you do not like this, what can you change?

## Homework Exercise 4

c) Write an SQL query that shows the original names of all deleted tables in the data dictionary together with the date when they were deleted. Read the Section “Using Flashback Drop and Managing the Recycle Bin” from Part III/Chapter 20 “Managing Tables” from the Oracle “Database Administrator’s Guide”:

[https://docs.oracle.com/en/database/oracle/oracle-database/18/admin/]

Please answer also the following questions:

- Which command can be used to really delete a table from the recycle bin?
- Which command can be used to recover a table from the recycle bin?

d) Use your account ⟨`Lastname`⟩`_dba` for the following task. This account got the right “`SELECT ANY DICTIONARY`” and also “`SELECT ANY TABLE`”.

Write an SQL query that lists at least three different types of files that Oracle uses. The output should have two columns: A short string indicating the type of the file (choose something sensible), and the file name (with full path).