CmpE321 - Project 2

Gökçe Uludoğan (gokce.uludogan@boun.edu.tr)

Deadline: July 29, 2019, Monday, 09:00

1 Project Description

In this project, you are expected to implement the storage manager design for the first project. It is allowed to make changes in the design. The implementation must support the following operations:

DDL Operations

- Create a type
- Delete a type
- List all types

DML Operations:

- Create a record
- Delete a record
- Update a record
- Search for a record
- List all records of a type

2 Input/Output

Submissions will be graded automatically in a Linux system (specifically Ubuntu 16.04) for various test cases. The allowed languages are C, C++, Java, Python3. The input and output file names will be given as arguments to your executable program. For each language, the commands that will be executed are as follows:

Java

```
javac 2016400XXX/src/*.java
java -classpath 2015400XXX/src/ storageManager inputFile outputFile
```

C/C++

The compilers will be used are gcc/g++ 5.4.0. You are allowed to use any C++ version as long as it is supported by this compiler and you write Makefile accordingly. [1]

```
make -C 2016400XXX/src
./2016400XXX/src/storageManager inputFile outputFile
```

Python3

```
python3 2016400XXX/src/storageManager.py inputFile outputFile
```

Your directory must have the structure in Figure 1.

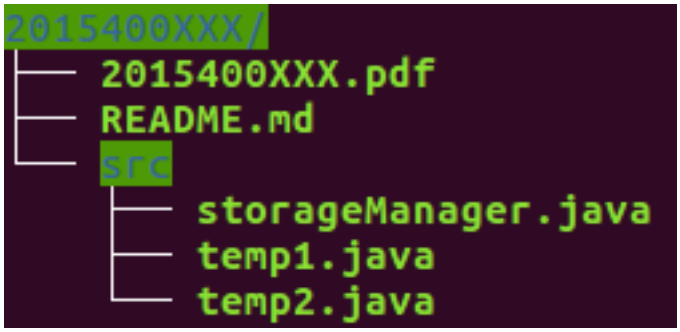

Figure 1: Directory structure

Input file contains a list of operations. Each line corresponds to an operation. Format for each operation is as follows:

DDL operations

Operation	Input Format	Output Format
Create	create type <type-name><number-of-fields><field1-name><field2-name>...	None
Delete	delete type <type-name>	None
List	list type	<type1-name> <type2-name> ...

DML operations

Operation	Input Format	Output Format
Create	create record <type-name><field1-value><field2-value>...	None
Delete	delete record <type-name><primary-key>	None
Update	update record <type-name><primary-key><field2-value><field3-value>...	None
Search	search record <type-name><primary-key>	<field1-value><field2-value>...
List	list record <type-name>	<record1-field1-value><record1-field2-value>... <record2-field1-value><record2-field2-value>... ...

[1] Make sure you include C++ version with *-std=c++XX* in Makefile

Sample Inputs & Outputs

Sample Input File	Sample Output File	Explanation
create type cat 3 name age species	5 2 3	list record cat
create type Human 2 name age	8 1 0	
create record cat 5 2 3	5 2 3	list record cat
create record cat 8 1 0	8 5 1	
list record cat	Human	list type
update record cat 8 5 1	cat	
list record cat	2 15	search record Human 2
create record Human 1 25	1 25	list record Human
create record Human 2 15	Human	list type
list type		
search record Human 2		
delete record Human 2		
list record Human		
delete type cat		
list type		

Table 1: Test case 1

Sample Input File	Sample Output File	Explanation
list record Human	1 25	list record Human
create type cat 3 name age species		
list record cat		
search record cat 5		
delete type cat		
delete type Human		
search record Human 1		

Table 2: Test case 2

3 Implementation Notes

- You must read data files page by page, similar to the previous project.
- Primary key of a record shall be value of first field of that record.
- Search, update and delete of records shall always be done by primary key.
- When a type is deleted, all records of that type must be deleted.
- While types must be listed by ascending type names, records must be listed by ascending primary key. Note that the built-in sort functions do sorting case sensitively by default. This means that the elements starting with uppercase letters come first in the order like in test case 1.

- Test cases are not necessarily independent from each other. So, type Human is created in test case 1 and it is still accessible in test case 2.
- You can assume that type names, field names and values cannot be longer than 10 characters.
- Consider possible leading and trailing spaces in input lines. Write robust codes that handle such situations as well.
- Note that field values' can be negative, they don't necessarily be positive.
- Make sure `delete type` command works as expected, it will be used between independent test cases.

4 Report & Grading

You are expected to submit a report *written in LaTeX* that contains the sections below. State clearly what changes from the design for the first project.

1. **Title Page:** Write course name, semester, assignment title, your name and student number.
2. **Introduction:** Briefly describe the project in your own words.
3. **Assumptions** & **Constraints:** Clearly specify your assumptions and constraints of the system in an itemized or tabular format.
4. **Storage Structures:** Explain your system catalogue, page design, page header, record header etc. with tables/diagrams/figures.
5. **Operations:** Write your DDL and DML operations and refer to corresponding storage structures when needed in pseudocode.
6. **Conclusions** & **Assessment:** Evaluate your design, considering its ups and downs. Discuss possible improvements.

Tentative grading weights are as follows:

- Test cases: 70%
- Code & Readme & Auto-Runnable Submission: 20%
- Report: 10%

5 Submission

The submissions will be through Moodle. Submit a zipped file named with your student number (e.g. 2016400XXX.zip). This zip must contain the report, source code and a Readme. If your file size is over Moodle upload limit, submit a link (drive, dropbox etc.) for downloading your report. Note that your reports will be inspected for plagiarism with previous years' submissions as well as this year's. Any sign of plagiarism will be penalized.