CS 405G: Introduction to database systems
Assignment 5

Assigned: Nov. 18th, 2013
Due: Dec. 9th, 2013

Reading assignment: Chapter 17 and Chapter 18.

Objective:

1. Understand the relationships between DBMS, storage and indexing
2. Implement a B+ tree based DBMS

Given the scope of the homework, it is a team project with you and your partner. Each team only needs to send in one copy of the code and report.

Program Description

(1) Memory Management class:

Both B+-tree and files are disk-based data structures and all the data are stored in disk files. To simplify the design, we have an in-memory version of the B+-tree and file, detailed below.

In your memory management for B+-tree, you need to allocate a large chunk of memory (~10M byes) at the beginning of running your code. The memory is then divided into "pages" (e.g. 1K byte in size). All the memory related operations are page based. You need to have the following functionalities to perform basic memory management:

a) Allocate: giving a page id that may be used by the B+-tree/file
b) Deallocate(p id): the page return to the memory management class and may be used for subsequent operations.

(2) You need to have the following functions related to the B+-tree class.

a) Insert(X): add an element X in the B+-tree.
b) Delete(X): delete an element X from the tree.
c) Size(): return the number of elements in the tree
d) isEmpty(): return true if the vector contains no element and false, otherwise.
e) Search(X): return the element of X, if it is in the tree. Otherwise, indicate that it is empty
f) print(): print out the elements of the vector

Each none-leaf node in the B+-tree is a single memory page, containing search keys and pointers (page id).
Each leaf node in the B+-tree is a single memory page, containing search keys and data.

(3) Implementing a DBMS supporting the following operations. **We always build the index (using B+-tree) for the primary key variable.**

a) Create a new table
b) Display all records in a table
c) Display records based on equality search on primary keys
d) Add records in a table
e) Delete records in a table

The syntax for the create table is exactly the SQL statement:

```
CREATE TABLE (variable list option);
```

- Variable list option := variable: type option, Variable list option
- To simplify the design, we only support two types: integer and fixed length string with length 10, i.e. “INTEGER” and “CHAR(10)”. The two allowed options are the string “PRIMARY KEY” and “” (empty string). Exactly one variable may have the option “PRIMARY KEY”.
- We may create multiple tables.

For displaying records based on equality search on primary keys, the format is

```
SELECT * FROM <table name> WHERE primary_key = <value>
```

The syntax for the rest of the operations are exactly the SQL statement.

How to test your code

You should name your executables as “myDBMS” for DBMS class, “myMEM” for the memory class, and “myBP” for the B+-tree class.

For the DBMS class executable myDBMS, it should accept an in-line parameter, which is the name of the testing file. For example, to run myDBMS, you should type:

```
$myDBMS testfile
```

where testfile is a testing file. You do not necessary name your test file as testfile.
The testing file for the myDBMS class is a set of SQL commands, separated by ";".
For example: the following is a valid testing file:

```
CREATE TABLE student (sid: CHAR(10) PRIMARY KEY, sage:
INTEGER, sdep:CHAR(10));
INSERT INTO student VALUES ('111', 20, 'cs');
INSERT INTO student VALUES ('112', 20, 'math');
INSERT INTO student VALUES ('113', 21, 'physics');
SELECT * FROM student
```

If you process the test file, you should see

```
111, 20, cs,
112, 20, math
113, 21, physics
```

What to turn in:

Send in a single zip file to liuj@netlab.uky.edu with the title "CS405-Homework 5". Including the following two pieces of information:

(1) Source codes with a makefile or instructions to compile your code if you use JAVA or C++. If you do not use makefile before, the following is a good tutorial:

http://capone.mtsu.edu/csdept/FacilitiesAndResources/make.htm

Source code needs to be properly commented.

(2) A report with the following components:

a. A brief discussion of the data structure that you use to implement the first two classes. Justify your answer.

b. Show your test files and your testing results.

c. Briefly discuss error handlings that you have implemented

Grading:

I. Correctness of the code (whether it can compile, whether it produces the correct output when correct input files are provided). 50%

II. Style of the coding including comments. 10%

III. Robust of the code with error handling. 15%

IV. Report (style, completeness, clearness). 25%