Assignment 4: Mini RDBMS Project

Due time: 30th Nov 2017 (11:55 pm)

Assignment Rules:

1) This is an ***individual*** assignment; you are required to work on your own.
2) The assignment solution you submit must be solely your own work; copying or letting others to copy are both considered cheating.
3) No late submission will be accepted!

Submission:

1) You are required to submit a .zip file of the source code files listed belowto the CASS submission system: https://course.cse.ust.hk/cass/student/#/submit
   Please use your student ID as the name of the .zip file.
   e.g. If your student ID is 20190001, your .zip file should be named as "20190001.zip".
   There are 4 source code files that you need to submit:
   - API.cc
   - BufferManager.cc
   - RecordManager.cc
   - CatalogManager.cc

   Please do remember to submit all these required files. Please do not submit any other file(s).
2) We will compile and test your codes by using Visual Studio 2015 on the machines in Lab 4210.

1 Project overview

The main purpose of this project is to deepen the understanding of the principles of the database system. You are required to implement SOME FUNCTIONS in a mini SQL engine (RDBMS) called Mini RDBMS. By entering SQL statements through a character interface, users can create/drop tables; insert/delete/select records from tables; create/delete index on tables.
Below are the details of this project.

1.1 Overview of the requirements

Data type

Mini RDBMS should support 3 data types: int, char(n), float.

Note: char(n) should satisfy 1 <= n <= 255.

Table definition

A table can have at most 32 attributes. Each attribute can be marked as 'unique' or not. Users can specify key attribute.

Creation and deletion of the index

It should create index for key attribute automatically. And for the attribute that are marked 'unique', user can create index on that by SQL statement. Therefore, all index is created on single attribute and single value.

Search entities

It should support interval query (e.g., ‘LESS THAN’), equal value query, multiple conditions query (connected using ‘AND’).

Insertion and deletion

It should support one or multiple deletion each time, and one insertion each time.

1.2 System Architecture

Here is the system architecture of Mini RDBMS:

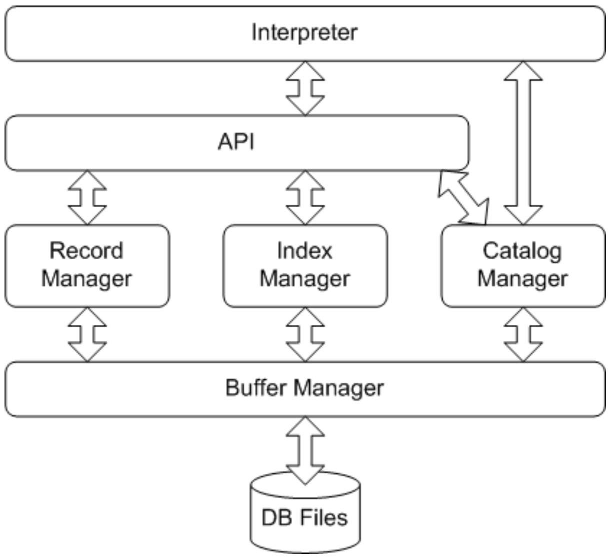

Figure 1: System Architecture

1.3 Model Overview

1.3.1 Interpreter

The Interpreter module directly interacts with the user. Its main functions include:

1) **Program flow control**, including:

- Start and initialize the system to receive commands,
- Process the commands,
- Show command results,
- Exit process.

2) **User command interpretation**, including:

- Receive and interpret the command entered by the user,
- Generate the internal data structure of the command,
- Check the syntax correctness and semantic correctness of the command,
- Call the API function provided by the correct command execution and display the execution result,
- Shows the error message when the command is incorrect.

1.3.2 API

API module is the core of the whole system, its main function is to provide an interface to execute SQL statements. Then the Interpreter layer can call the API module directly.

The input of API module is the command generated by the Interpreter layer. API module determines the execution rules based on the information provided by the Catalog Manager, and calls the corresponding interfaces provided by the Record Manager, Index Manager and Catalog Manager to conduct the execution, and finally returns the execution results to the Interpreter module.

1.3.3 Catalog Manager

The Catalog Manager manages all schema information for the database, including:

1) The definition of all the tables in the database, including the names of the tables, the number of fields (columns) in the tables, the primary keys, the indexes defined in the tables.
2) The definition of each field in the table information, including the type of field and whether it is unique.
3) The definition of all the indexes in the database, including the tables the indexes are built on, and the columns the indexes are built on.

The Catalog Manager must also provide interfaces to access and manipulate the information mentioned above for Interpreter and API modules.

1.3.4 Record Manager

Record Manager should manage the data files in the record table. The main function is to create and delete data files (defined by creation and deletion of the table), as well as insertion, deletion and select operations. It also needs to provide the corresponding interface for other parts in the system.

Specifically, the select operation requires the ability to support non-conditional search and search with one condition (as listed in Section 1.2).

The data file consists of one or more data blocks, the size of blocks should be the same as the size of buffer block. A block contains one or more records.

For simplicity, It is only required to support for fixed-length records of storage, instead of cross-block storage.

1.3.5 Buffer Manager

Buffer Manager is responsible for buffer management, including:

1) Read the specified data to the system buffer or buffer data to the file;
2) To achieve the replacement algorithm of the buffer, when the buffer is full, select the appropriate page to be replaced;
3) Record the status of each page in the buffer, e.g., whether the page has been modified or not;
4) Provide the pin page buffer function, lock the buffer page and do not allow replacement;
5) To improve the efficiency of disk I / O operations, the buffer and file system interaction unit should be blocks. The size of a block should be an integral multiple of the file system. Generally, the disk interaction unit can be set to 4KB or 8KB.

1.3.6 DB Files

DB Files refers to all the data files that make up a database, and consists of Record data files, Index data files, and Catalog data files.

2 Project requirements

2.1 Your jobs

Read the code carefully, and fill in the code to following the function. There are 5 TODOs in the project.

1) In file ‘API.cc’, finish the code of dropping one table in API level.
2) In file ‘BufferManager.cc’, finish the code of reading a block from disk to main memory, and when it’s necessary replace the block using LRU schema.
3) In file ‘RecordManager.cc’, you need to insert a record into the database file.
4) In file ‘RecordManager.cc’, you need to delete a recode from database.
5) In file ‘CatalogManager.cc’, you need to load table information and return a table structure.

For other files, you don’t need to worry about but you **DO NEED TO** read and understand how it works. ‘filter’ is in charge for query conditions, ‘global’ is for data structures, ‘indexManager’ is for manage index.

2.2 Test cases and marking schema

No partial marks!
We will test you work based on the test cases, 10 test cases and each for 10 marks.
The detailed requirements are listed below.

- The DBMS should be based on SQL92 standard.
- Every SQL statement should end with a ‘;’, and can be written in one row or multiple rows.
- Every keyword (i.e., create, drop, insert, delete, char, int, …) in SQL statement should be in lower case.

1. Create table statement

The CREATE TABLE statement is used to create a new table in a database.

Syntax:

```
create table table_name (
        column_1 datatype,
        column_2 datatype,
        ...
        primary key ( column_name )
);
```

The column parameters specify the names of the columns of the table.
The datatype parameter specifies the type of data the column can hold (e.g. int, char, etc.).
If this statement is executed successfully, please output a message indicating the successful execution, otherwise there should be a warning message indicating the reason for the failure.

Sample test case:

```
create table student (
        sno char(8),
        sname char(16) unique,
        sage int,
        sgender char (1),
        primary key ( sno )
);
```

2. Drop table statement

Syntax:

```
drop table table_name;
```

If this statement is executed successfully, please output a message indicating the successful execution, otherwise there should be a warning message indicating the reason for the failure.

Sample test case:

```
drop table student;
```

3. Select statement

Syntax:

```
select * from table_name;
```

OR

```
select * from table_name
where column_1 op value_1 and conlumn_2 op value_2 ... and column_n op value_n;
```

Note: op is the arithmetic operators, i.e., =, <>, <, >, <=, >=

If this statement is executed successfully, please output a message indicating the successful execution, otherwise there should be a warning message indicating the reason for the failure.

Sample test case:

```
select * from student;
select * from student where sno = '12345';
select * from student where sage > 18 and sgender = 'F'
```

4. Insert statement

Syntax:

```
insert into table_name values(value_1, value_2, ..., value_n);
```

If this statement is executed successfully, please output a message indicating the successful execution, otherwise there should be a warning message indicating the reason for the failure.

Sample test case:

```
insert into student values ('12345', 'jason',22, 'M');
```

5. Delete statement

Syntax:

```
delete from table_name;
```

OR

```
delete from table_name
where column_1 op value_1 and conlumn_2 op value_2 ... and column_n op value_n;
```

If this statement is executed successfully, please output a message indicating the successful execution, otherwise there should be a warning message indicating the reason for the failure.

Sample test case:

```
delete from student;
delete from student where sno = '12345';
```

6. Quit database statement

Syntax:

```
quit;
```

7. Create index statement

Syntax:

```
create index index_name
on table_name ( column_1, column_2, ...);
```

If this statement is executed successfully, please output a message indicating the successful execution, otherwise there should be a warning message indicating the reason for the failure.

Sample test case:

```
create index stunameidx on student ( sname );
```

8. Delete index statement

Syntax:

```
delete index index_name;
```

If this statement is executed successfully, please output a message indicating the successful execution, otherwise there should be a warning message indicating the reason for the failure.

Sample test case:

```
drop index stunameidx;
```

9. Execute the SQL script file

Syntax:

```
execfile file_name;
```

SQL script file can contain any statement of the 8 SQL statements mentioned above. Your DBMS system should execute the SQL statements in the SQL script file sequentially.

Note: In COMP3311_Assignment4.zip, we have also given you the sample test cases in 'testcases.txt', and the expected output in 'ans_testcases.txt'. Please refer to these two files as your sample input and sample output.