**COP6726 Report of Project1**

**Abstract:**

DBFile is used to store and retrieve record from disk, for project1, DBFile only support simple heap structure, which is an unordered file of records, where new records simply go at the end of the file. My implementation support smoothly switch between reading and writing on file, user can stop reading at any time and start to write immediate without further operations, and vice versa. Gtest is used for testing Create, Close and Open operations, and there are some other test cases stored in README file that used for testing correctness of reading and writing.

**Compiling and running project:**

Please make sure you are using linux system, this makefile don’t work for mac system.

Steps of compiling and running project(after entering directory of project):

1. make clean
2. make
3. make test.out
4. ./test.out

The directory for storing tpch data is hardcoded in test.cc file. Please make change in test.cc if you want to specify your own directory for storing tpch data.

For compiling and running gtest, refer to the end of this report.

**Explanation for functions:**

**void MoveFirst ();**

This method will move pointer that point to current record to very first record in the very first page of file. This “pointer” is not actually existed, instead we have a buffer page that indicated current page of file, and that buffer page hold the current record, which is “pointer”. So we can reload buffer page to represent very first page of file, to move “pointer” to first record.

This method also cannot be used without open file, otherwise the method would just return.

**void Add (Record &addMe);**

This method is used to add new record into file. Specific speaking, record will be added into buffer page in the main memory. Once buffer page is full, then buffer page will be written into disk based file, so that record will be actually added into disk, and buffer page would be emptied.

One important thing is: This method support switch between reading and writing, which means that if file were reading in previous, then this method would first emptify buffer page, and reload lastest page stored in file into buffer page, so that file can continue to do writing.

Additional, this method also can only can be used when file was already opened.

**int GetNext (Record &fetchMe);**

This method is used to do reading for file. Specific speaking, it will retrieve next record in the buffer page. Once buffer page is full, then buffer page will be reloaded with next page stored in the disk based file.

For case of switching from writing to reading, this method would first do MoveFirst method, to move pointer to default position and update buffer page to very first page stored in file, and then do actual reading.

Also, method would not be used when file is not open.

Return 1 if there is record to read, return 0 if there are no records to read or file is not open.

**int GetNext (Record &fetchMe, CNF &applyMe, Record &literal);**

This is an advanced version of GetNext, it will get next record that accept by the CNF predicate. Specific implementation is that it will keep using basic version of GetNext to get record, and check if record can be accepted by CNF until retrieving record accepted by CNF or there is no more record stored in the file.

**int Create (char \*name, fType myType, void \*startup);**

This method simply create or erase and recreate the new heap file(.bin file that store binary data for records) that used to store records. This method would also act like normal open after real file creation.

Return 1 if creation succeed, or return 0 if file cannot be created(Such as file is opening right now, so file cannot be erased to recreate).

**int Open (char \*name);**

This method simply open the file and set some state variables to default value.

Return 1 if Opening succeed, or return 0 is opening failed(Such as file was opened and not closed before).

**int Close ();**

This method simply close file and update some states variables. If buffer page is not empty and file was writing before closing, then it will first writ records in buffer page into disk based file.

Return 1 if closing works, or return 0 if closing failed(Such as file not opened before, so there is no sense to close it).

**void Load (Schema &mySchema, char \*loadMe);**

This method will read all records in selected(indicated of loadMe) .tbl file, and loaded them into current bin file.

I added extra condition here to check whether open, if file is open then loading would continue to execute, but if file is not open then just return and interrupt the following loading.

**Test.out results:**

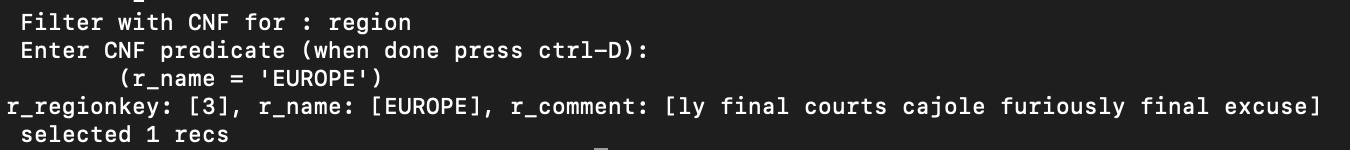
Below are results of q1, q2, q3, q11, q12 under 1GB data and 10MB data.

I changed some syntax errors of queries listed in original README file. For example: In q2 and q3, string like 'middle east' and 'japan' should be capitalized in order to get expected results listed in output.log.

But it seems like q11 are totally incorrect, and I don’t know specific reason for that.

**Q1: (r\_name = 'EUROPE')**

Result of 1G data:



Result of 10MB data:

文本

描述已自动生成

**Q2: (r\_name <'MIDDLE EAST') AND (r\_regionkey > 1)**

Result of 1G data:

**文本

描述已自动生成**

Result of 10MB data:文本

描述已自动生成

**Q3: (n\_regionkey = 3) AND (n\_nationkey > 10) AND (n\_name > 'JAPAN')**

Result of 1G data:

**文本

描述已自动生成**

Result of 10MB data:

**文本

描述已自动生成**

**Q11: (l\_shipdate > '1994-01-01') AND**

**(l\_shipdate < '1994-01-07') AND**

**(l\_discount > 0.05) AND**

**(l\_discount < 0.06) AND**

**(l\_quantity = 4.0)**

Result of 1G data:

文本

描述已自动生成

Result of 10MB data:

文本

描述已自动生成

**Q12: (l\_orderkey > 100) AND**

**(l\_orderkey < 1000) AND**

**(l\_partkey > 100) AND**

**(l\_partkey < 5000) AND**

**(l\_shipmode = 'AIR') AND**

**(l\_linestatus = 'F') AND**

**(l\_tax < 0.07)**

Result of 1G data:

**文本

描述已自动生成**

Result of 10MB data:

**砖墙上

低可信度描述已自动生成**

**Gtest results:**

To running the gtest, first make sure you enter under root directory of project, where should contain gtest.cpp file Makefile.

You must have gtest installed on your computer to running my gtest.

<https://github.com/google/googletest/blob/master/googletest/README.md#standalone-cmake-project>

Steps for running gtest:

1. make clean

2. make gtest

3. ./gtest.out

There are totally 12 tests that testing Create, Close and Open methods of DBFile. Some of them just test function as simple caller, but some of them tested under different circumstance, such as close a file after it’s already closed etc.

CreateTest is testing for simple create file, it should succeed;

CreateAfterOpenTest is testing for create file after open it without close, it should fail;

CreateAfterCloseTest is testing for create file after close it, should succeed;

CreateAfterCreateTest is testing for create file after created but not closed, should fail;

OpenTest is testing for simple open file, should succeed;

OpenAfterOpenTest is testing for open file after it's already opened, should fail;

OpenAfterCloseTest is testing for open file after it's closed, should succeed;

OpenAfterCreateTest is testing for open file after it's created but not closed, should fail;

CloseTest is testing for direct close file without create or open, should fail;

CloseAfterOpenTest is testing for close file after open, should succeed;

CloseAfterCloseTest is testing for close file after it's already closed, should fail;

CloseAfterCreateTest is testing for close file after create, should succeed;

Below are results of gtest, there are some error messages generated by DBFile when user did some wrong operations.

文本

描述已自动生成