Project Report

Group Info

UFID: 8115 5459 Shaileshbhai Revabhai Gothi

UFID: 1451 4662 Siddardha Majety

Getting Started

A test driver has been created to test the program. This test driver gives a menu-based interface to three options that allows you to test your code:

1. load (read a tpch file and write it out a heap DBFile)
2. scan (read records from an existing heap DBFile)
3. scan & filter (read records and filter using a CNF predicate)

Note that the driver only works with the tpch files (generated using the dbgen program).

Using the driver:

1. SETTINGS: The following variables control the various file locations and they are declared in test.cc (just after the #include header declarations):

- **dbfile_dir**: this is where the created heap db files will be stored. By default, this is set to "" (thus all the heap dbfiles will be created in the current directory).
- **tpch_dir:** this stores the directory path where the tpch-files can be found. By default, tpch_dir is set to "/cise/tmp/dbi_sp11/DATA/"(Please *change this path according to location of your tpch_dir)*
- **catalog_path**: this stores the catalog file path. By default this is set to "" (*Please change this path if you want to change storage location of your catalog*.)

2. Once load of a file has been selected, you can select option 2 or 3 to scan/filter all the records from the heap DBfile. If option 3 is selected, a CNF should be supplied.

How to run

1. Unzip the contents/pull from bitbucket
2. Update the path for tpch_dir in test.cc file.
3. Run the below command.

```
# Compile test driver
make -B
# Run test driver and follow the instructions on console.
./test.out
```

```
# Compile main
make -B main
# Run main file and provide CNF.
./main
```

Example CNF's

Use the table below to identify the tpch file associated with each CNF. table | CNF

region	q1 q2
nation	q3
supplier	q4 q5
customer	q6
part	q7
partsupp	q8 q9
orders	q10
lineitem	q11 q12

The expected output for these CNF's can be found in the file "output.log". Example CNF below:

q1 (r_name = 'EUROPE')

q2 (r_name < 'middle east') AND (r_regionkey > 1)

q3 (n_regionkey = 3) AND (n_nationkey > 10) AND (n_name > 'japan')

q4 (s_suppkey < 10)

q5 (s_nationkey = 18) AND (s_acctbal > 1000) AND (s_suppkey < 400)

q6 (c_nationkey = 23) AND (c_mktsegment = 'FURNITURE') AND (c_acctbal > 7023.99) AND (c_acctbal < 7110.83)

q7 (p_brand = 'Brand#13') AND (p_retailprice > 500) AND (p_retailprice < 930) AND (p_size > 28) AND (p_size < 1000000)

q8 (ps_supplycost > 999.98)

q9 (ps_availqty < 10) (ps_supplycost > 100) AND (ps_suppkey < 300) AND

q10 (o_orderpriority = '1-URGENT') AND (o_orderstatus = '0') AND (o_shippriority = 0) AND (o_totalprice > 1015.68) AND (o_totalprice < 1051.89)

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)

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)

Implementation Details:

DBFile Class Reference

Acts as a factory class to create DBFile based on type heap, sorted and tree provided. *. More...

`#include <DBFile.h>`

Public Member Functions

int	**Create** (const char *fpath, fType file_type, void *startup)
int	**Open** (const char *fpath)
int	**Close** ()
void	**Load** (Schema &myschema, const char *loadpath)
void	**MoveFirst** ()
void	**Add** (Record &addme)
int	**GetNext** (Record &fetchme)
int	**GetNext** (Record &fetchme, CNF &cnf, Record &literal)

Detailed Description

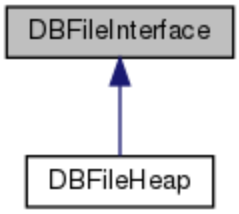
Acts as a factory class to create DBFile based on type heap, sorted and tree provided. *.

DBFileInterface Class Reference abstract

A Interface of DBFIle. Provides signature of methods that represents a **DBFile**. Three types of **DBFile** heap, sorted and tree are implemented based on this interface. Requires 3 variables,. More...

`#include <DBFileInterface.h>`

Inheritance diagram for DBFileInterface:



[legend]

Public Member Functions

virtual int	**Create** (const char *fpath, fType file_type, void *startup)=0
virtual int	**Open** (const char *fpath)=0
virtual int	**Close** ()=0
virtual void	**Load** (**Schema** &myschema, const char *loadpath)=0
virtual void	**MoveFirst** ()=0
virtual void	**Add** (**Record** &addme)=0
virtual int	**GetNext** (**Record** &fetchme)=0
virtual int	**GetNext** (**Record** &fetchme, **CNF** &cnf, **Record** &literal)=0

Detailed Description

A Interface of DBFIle. Provides signature of methods that represents a DBFile. Three types of DBFile heap, sorted and tree are implemented based on this interface. Requires 3 variables,.

1. oFile, A file to store pages of records.
2. oPage, Pages which stores limited number of records. When Page is full its stored to file and a new page is created.
3. pageNumber, An offest representing current page Number in file.

Member Function Documentation

◆ Add()

virtual void DBFileInterface::Add (Record & addme) pure virtual

In order to add records to the file, the function Add is used. In the case of the unordered heap file that you are implementing in this assignment, this function simply adds the new record to the end of the file.

Implemented in DBFileHeap.

◆ Close()

virtual int DBFileInterface::Close ( ) pure virtual

Close simply closes the file. The return value is a 1 on success and a zero on failure.

Implemented in DBFileHeap.

◆ Create()

virtual int DBFileInterface::Create (const char * fpath,
fType file_type,
void * startup
) pure virtual

Used to create a file. The first parameter to this function is a text string that tells you where the binary data is physically to be located – you should store the actual database data using the File class from File.h. The second parameter to the Create function tells you the type of the file. Finally, the last parameter to Create is a dummy parameter that you won't use for this assignment, but you will use for assignment two. The return value from Create is a 1 on success and a zero on failure.

Implemented in DBFileHeap.

◆ GetNext() [1/2]

virtual int DBFileInterface::GetNext (Record & fetchme) pure virtual

GetNext simply gets the next record from the file and returns it to the user, where "next" is defined to be relative to the current l ocation of the pointer. After the function call returns, the pointer into the file is incremented, so a subsequent call to GetNext won't return the same record twice. The return value is an integer whose value is zero if and only if there is not a valid record returned from the function call (which will be the case, for example, if the last record in the file has already been returned).

Implemented in DBFileHeap.

◆ GetNext() [2/2]

virtual int DBFileInterface::GetNext (Record & fetchme,

CNF & cnf,

Record & literal

) pure virtual

GetNext also accepts a selection predicate (this is a conjunctive normal form expression). It returns the next record in the file that is accepted by the selection predicate. The literal record is used to check the selection predicate, and is created when the parse tree for the CNF is processed.

Implemented in DBFileHeap.

◆ Load()

virtual void DBFileInterface::Load (Schema & myschema,

const char * loadpath

) pure virtual

Load function bulk loads the DBFile instance from a text file, appending new data to it using the SuckNextRecord function from Record.h. The character string passed to Load is the name of the data file to bulk load.

Implemented in DBFileHeap.

◆ MoveFirst()

virtual void DBFileInterface::MoveFirst ( ) pure virtual

Each DBFile instance has a “pointer” to the current record in the file. By default, this pointer is at the first record in the file, but it can move in response to record retrievals. The following function forces the pointer to correspond to the first record in the file.

Implemented in DBFileHeap.

◆ Open()

virtual int DBFileInterface::Open (const char * fpath) pure virtual

This function assumes that the DBFile already exists and has previously been created and then closed. The one parameter to this function is simply the physical location of the file. If your DBFile needs to know anything else about itself, it should have written this to an auxiliary text file that it will also open at startup. The return value is a 1 on success and a zero on failure.

Implemented in DBFileHeap.

The documentation for this class was generated from the following files:

- DBFileInterface.h

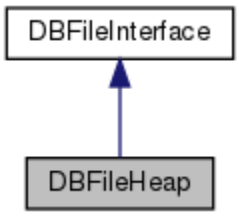
- DBFileInterface.cc

DBFileHeap Class Reference

A Heap Implmentation of DBFIle. Creates an unordered file of records, where new records simply go to the end of file. More...

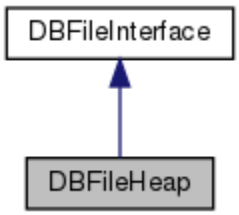
`#include <DBFileHeap.h>`

Inheritance diagram for DBFileHeap:



[legend]

Collaboration diagram for DBFileHeap:



[legend]

Public Member Functions

int	**Create** (const char *fpath, fType file_type, void *startup)
	Erases an existing file and creates a new **File** with Heap type record storage. More...
int	**Open** (const char *fpath)
	Open the existing heap file. More...
int	**Close** ()

	Close the open heap file. More...
void	**Load** (**Schema** &myschema, const char *loadpath)
	Loads all record from .tbl file to Heap file. More...
void	**MoveFirst** ()
	Move to the first record in file.
void	**Add** (**Record** &addme)
	Adds a new record to the end of file. More...
int	**GetNext** (**Record** &fetchme)
	Gets next record from the file and returns to user, where "next" is defined relative to the current pointer. After return pointer in file is increment to next record. More...
int	**GetNext** (**Record** &fetchme, **CNF** &cnf, **Record** &literal)
	Gets next record from the file based on selection predicate (a conjuctive normal form expression), where "next" is defined relative to the current pointer in selection predicate. After return pointer in file is increment to next record. More...

Detailed Description

A Heap Implmentation of DBFIle. Creates an unordered file of records, where new records simply go to the end of file.

Member Function Documentation

◆ Add()

void DBFileHeap::Add (**Record** & rec) virtual

Adds a new record to the end of file.

Parameters

rec record that needs to be inserted.

Implements **DBFileInterface**.

◆ Close()

int DBFileHeap::Close ( ) virtual

Close the open heap file.

Returns

1 on success and 0 on failure.

Implements **DBFileInterface**.

◆ Create()

int DBFileHeap::Create (const char * f_path,
fType f_type,
void * startup
) virtual

Erases an existing file and creates a new **File** with Heap type record storage.

Parameters

f_path path of file that needs to be created.

Returns

1 on success and 0 on failure.

Implements **DBFileInterface**.

◆ GetNext() [1/2]

int DBFileHeap::GetNext (Record & fetchme) virtual

Gets next record from the file and returns to user, where "next" is defined relative to the current pointer. After return pointer in file is increment to next record.

Parameters

fetchme Pointer to which record needs to returned.

Returns

1 on success and 0 on failure.

Implements DBFileInterface.

◆ GetNext() [2/2]

int DBFileHeap::GetNext (Record & fetchme,
CNF & cnf,
Record & literal
) virtual

Gets next record from the file based on selection predicate (a conjuctive normal form expression), where "next" is defined relative to the current pointer in selection predicate. After return pointer in file is increment to next record.

Parameters

fetchme Pointer to which record needs to returned.

cnf Conjuctive normal form expression to filter the records.

literal

Returns

1 on success and 0 on failure.

Implements DBFileInterface.

◆ Load()

```
void DBFileHeap::Load ( Schema &     f_schema,
                        const char *  loadpath
                      )                                    virtual
```

Loads all record from .tbl file to Heap file.

Parameters

f_schema schema of file to be loaded.

loadpath path of file to be loaded.

Implements DBFileInterface.

◆ Open()

```
int DBFileHeap::Open ( const char *  f_path )              virtual
```

Open the existing heap file.

Parameters

f_path Path of file to open.

Returns

1 on success and 0 on failure.

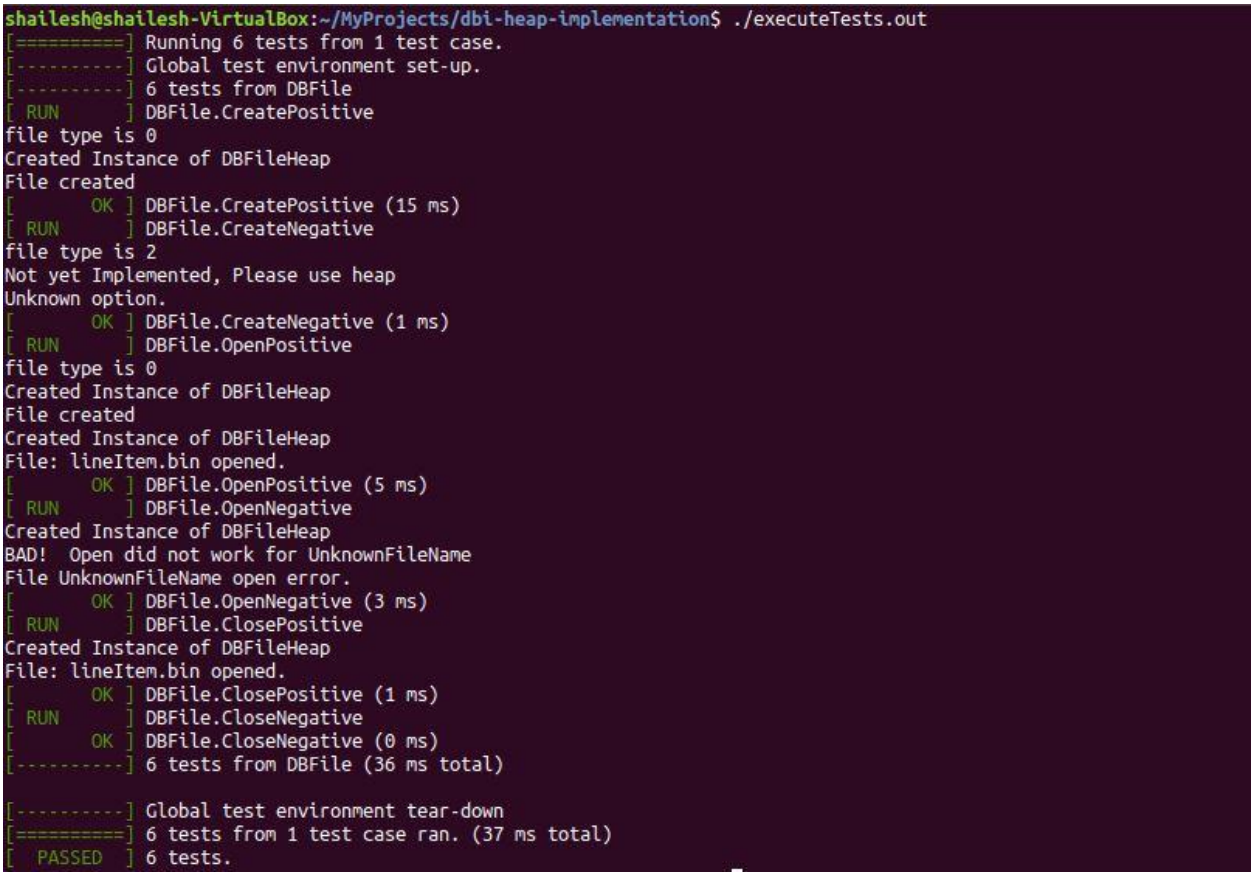
Implements DBFileInterface.

Run Testcases:

Use below command to compile and run testcases.

```
# Compile gTests
make -B executeTests.out
# Run All test cases.
./executeTests.out
```

Test cases run results:



Screenshots of the CNF generated from running our code on 1GB files.

Query 1 : (r_name = 'EUROPE')

```
 catalog location:      catalog
 tpch files dir:        DATA/10MB/
 heap files dir:


 select test:
         1. load file
         2. scan
         3. scan & filter
         3

 select table:
         1. nation
         2. region
         3. customer
         4. part
         5. partsupp
         6. orders
         7. lineitem
         2
 Filter with CNF for : region
 Enter CNF predicate (when done press ctrl-D):
        (r_name = 'EUROPE')Created Instance of DBFileHeap
File: region.bin opened.
Moved to first Page
r_regionkey: [3], r_name: [EUROPE], r_comment: [ly final courts cajole furiously final excuse]
 selected 1 recs
```

Query 2 : (r_name < 'MIDDLE EAST') AND (r_regionkey > 1)

```
 Filter with CNF for : region
 Enter CNF predicate (when done press ctrl-D):
        (r_regionkey > 1) AND (r_name < 'MIDDLE EAST')
Created Instance of DBFileHeap
File: region.bin opened.
Moved to first Page
r_regionkey: [2], r_name: [ASIA], r_comment: [ges. thinly even pinto beans ca]
r_regionkey: [3], r_name: [EUROPE], r_comment: [ly final courts cajole furiously final excuse]
 selected 2 recs
```

Query 3 : (n_regionkey = 3) AND (n_nationkey > 10) AND (n_name > JAPAN)

```
 select table:
        1. nation
        2. region
        3. customer
        4. part
        5. partsupp
        6. orders
        7. lineitem
        1
 Filter with CNF for : nation
 Enter CNF predicate (when done press ctrl-D):
        (n_regionkey = 3) AND (n_nationkey > 10) AND (n_name > 'JAPAN')
Created Instance of DBFileHeap
File: nation.bin opened.
Moved to first Page
n_nationkey: [19], n_name: [ROMANIA], n_regionkey: [3], n_comment: [ular asymptotes are about the furious multipliers. express dependencies nag above the ironically ironic account]
n_nationkey: [22], n_name: [RUSSIA], n_regionkey: [3], n_comment: [ requests against the platelets use never according to the quickly regular pint]
n_nationkey: [23], n_name: [UNITED KINGDOM], n_regionkey: [3], n_comment: [eans boost carefully special requests. accounts are. carefull]
 selected 3 recs
```

Query 11 : (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)

```
 select table:
        1. nation
        2. region
        3. customer
        4. part
        5. partsupp
        6. orders
        7. lineitem
        7
 Filter with CNF for : lineitem
 Enter CNF predicate (when done press ctrl-D):
       (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)
Created Instance of DBFileHeap
File: lineitem.bin opened.
Moved to first Page
 selected 0 recs
```

Query 12 : (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)

```
select table:
       1. nation
       2. region
       3. customer
       4. part
       5. partsupp
       6. orders
       7. lineitem
       7
Filter with CNF for : lineitem
Enter CNF predicate (when done press ctrl-D):
      (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)
reated Instance of DBFileHeap
ile: lineitem.bin opened.
oved to first Page
_orderkey: [130], l_partkey: [1739], l_suppkey: [4240], l_linenumber: [2], l_quantity: [48], l_extendedprice: [78755], l_discount: [0.03], l_tax: [0.02], l_returnflag: [R], l_linestatu
: [F], l_shipdate: [1992-07-01], l_commitdate: [1992-07-12], l_receiptdate: [1992-07-24], l_shipinstruct: [NONE], l_shipmode: [AIR], l_comment: [lithely alongside of the regu]
_orderkey: [194], l_partkey: [2594], l_suppkey: [5095], l_linenumber: [1], l_quantity: [17], l_extendedprice: [25442], l_discount: [0.05], l_tax: [0.04], l_returnflag: [R], l_linestatu
: [F], l_shipdate: [1992-05-24], l_commitdate: [1992-05-22], l_receiptdate: [1992-05-30], l_shipinstruct: [COLLECT COD], l_shipmode: [AIR], l_comment: [ regular deposi]
selected 2 recs
```