CENG351- DATA MANAGEMENT AND FILE STRUCTURES
PROGRAMMING HOMEWORK 1

1. Title : DATABASE IMPLEMENTATION USING PILE FILE

2. Due Date : 30.11.2011 23:55

3. Submission / Evaluation

You will submit your homeworks via course page on cow. Please put all related files (including Makefile) into a tar file named “hw1.tar” and submit it. Everything you submit should be your own work.

Evaluation will be done using blackbox testing techniques. So please do not forget to add necessary spaces and new lines. Grading will be done using following chart:

Insertion	%20
Deletion	%20
Update	%15
Select	%15
Join	%30

As mentioned in syllabbus you have a late submission right of 3 days in total for programming homeworks. After that your homeworks will be graded over 100 – 10*day^2.

Note: You are strongly adviced to start implementing the homework as soon as possible. It may be much more time consuming than you think.

4. Tasks

You will implement a basic database management system which stores data on a pile file containing variable-length records which consist of variable-length fields. Your system will get user commands from standard input and do some stuff according to it. Possible commands are "insert","delete", "update", "select", "join" and "quit". Each one is described in detail later in this document.

Two different file structures will be parsed in the scope of this homework . One of them is

Main Database File		
ID	Name	Department
509	Larry Page	Founder
120	Bill Gates	Founder
203	Eric Schmidt	Chairman
135	Sergey Brin	Founder

Join File		
Size	Count	ID
A4	5	509
A3	1	203
A4	18	509
A3	5	120
A5	2	203

your "main database file" which stores your records, the other one is "join file" which is used with "join" command. Both file structures are also described in detail later in this document.

Figure 1: Example records of main database and join file to be used later.

4.1. Compilation

Your submission should include a makefile which includes all necessary include paths and libraries to be linked. In other words, "make" command should be enough to compile.

<u>Example Makefile:</u>

```
dtbase: file1.o file2.o file3.o
	g++ -o dtbase file1.o file2.o file3.o -lm

file1.o: file1.cpp
	g++ -c file1.cpp

file2.o: file2.cpp
	g++ -c file2.cpp

file3.o: file3.cpp
	g++ -c file3.cpp
```

Note: Do not forget to add a tab before each line starting with "g++" command.

4.2. Execution

- Your system will be executed using command "./dtbase <file_name>". <file_name> is the path to the main database file.
- After execution, the system should wait for user commands from standard input until "quit" command is entered.
- Main database file will not be empty. It always includes header part; but, it may not contain any records. For example, assume your system is executed with a main database file which does not have any record initially, some records are added to it and the system quits. If the system executed again with the same main database file, previously added

records should be available to the user.
- During execution changes to the main database file should be reflected immediately. Do not forget to use flush() function after each operation.

4.3. Commands

Commands should be read line by line. They are not valid until Enter/Return key is pressed. “insert”, “delete”, “update” commands do not print anything to standard output while “select” and “join” commands do.

4.3.1. Insert

Insert command will be given as follows:
insert <record>

<record> is a structure like:
<record>: <field1>|<field2>|...|<fieldn>

- Number and order of the fields are the same as main database file (first field is the record id).
- Your system should add necessary information(eg. size of the record, empty flag, "#") to the record structure and write it to the main database file.
- If there is an empty slot large enough for the new record it will be used for insertion offset. Otherwise new record will be written to the end of the file.
- While inserting to an empty slot worst-fit strategy will be applied. Avail list is sorted in descending order according to size of the slots. The largest slot will be used always. If slot used is longer than the new record unused space will be added to avail list and it should be updated..

4.3.2. Delete

Delete command will be given as follows:
delete <record_id>

- <record_id> is the id of the record to be deleted. As mentioned above first field of each record is the id of it.
- Delete operation is like soft deletion. Empty flag of the record will marked to be empty (set to 1) and the slot will be added to avail list. Content of the record will not be erased.
- After adding new slot to the avail list, it should be updated.

4.3.3. Update

Update command will be given as follows:
update <record>

<record> is a structure like:
<record>: <field1>|<field2>|...|<fieldN>

- Record to be updated is found using record id (first field of <record> structure).
- If size of the new record is less than or equal to size of the old record, new record is written to the offset of the old record in the main database file. In this case unused bytes, if exist, should be added to avail list and it should be updated.
- If size of the new record is greater than size of the old record simply delete old one and insert new one.

4.3.4. Select

Select command will be given as follows:
select <record_id>

- Your system should find the record with id equals to <record_id> and write it to standard output in the following format:

```
<field_name1> = <field1>
<field_name2> = <field2>
            .
            .
            .
<field_nameN> = <fieldN>
```

<field_nameX> is the name of the Xth field given in the header of the main database file.
<fieldX> is the value of the Xth field of the related record.

- If record is not found write "Record not found" to the standard output.

Example output of command “select 509” with files in Figure 1 is as follows:

```
ID = 509
Name = Larry Page
Department = Founder
```

4.3.5. Join

Join command will be given as follows:

```
join <file_name>
```

<file_name> is the path to the join file.

- Records are located in the join file according to hash value of record id. Hash function returns the block number in which the record resides.
- Record id will not be necessarily first field of in the join file. You will find the field corresponding to record id by its name in the join file. Record id field in the join table has the same name as record id field (first field) in the main database file.
- Your system should find number of the records in the join file corresponding to each record in the main database file.

Example output of join command with the files in Figure 1 as follows:

```
509 2
120 1
203 2
135 0
```

Note: Order is not important

4.4. Hash Function

Your hash function will be the following function:

```
int hash(unsigned int record_id)
{
        return (record_id % 20);
}
```

As mentioned record_id is the first field of each record in main database file. It is stored as array of ASCII characters. Therefore, you should convert it to an unsigned integer before giving it to the hash function as a parameter.

4.5. Updating Avail List

Because worst-fit strategy is used to insert records into empty places, avail list should always be sorted in descending order according to their size variables. To keep avail list sorted, after every change in it you should sort it again.

In the avail list adjacent empty places should not exist. In other words, if two adjacent empty places are available you should merge them into one and sort the avail list again. After delete and update commands the emptied place should be checked if it has adjacent empty place. If there is they should be merged and avail list is updated and sorted accordingly.

Note: Be careful about update command. If a record is updated to be shorter, add unused space at the end of the record into avail list in the same manner.

5. File Structures

As mentioned earlier, you are to operate on two different files having different structures. One of them is "main database file" and the other one is "join file".

- Both files are binary. So, you should open these files with binary flag.
- Your system will read and write to the main database file while the join file will be used just for reading. It is better to use mode flags accordingly while opening the files.

5.1. Main Database File

- Main database file is a pile file containing variable-length records with variable-length fields.
- First field of each record is the id of that record.
- Records are not sorted on any field.
- There is not limitation on the number of records. You should parse until you end up with the end of file (EOF).

Main database file has following structure:
<header><records>

<header> is a structure like:
<number_of_fields><field_name1>|<field_name2>|...|<field_nameN>|<avail_list>

<records> is a structure like:
<record><record>....<record>

<number_of_fields> is a 4-byte unsigned integer

<field_nameX> is a variable length string of ASCI characters. Field names are seperated from each other by '|' character.

<avail_list> has constant size of 164 bytes. It can contain 20 empty slots at most. If it contains less than 20 slots unused space in it should be filled with 0s (null). It has following structure:

<number_of_slots><slot_offset1><slot_size1>....<slot_offsetN><slot_sizeN>

<number_of_slots> is a 4-byte unsigned integer storing number of empty slots.

<slot_offsetX> is a 4-byte unsigned integer holding starting offset of the Xth empty slot

<slot_sizeX> is a 4-byte unsigned integer holding size of the Xth empty slot.

<record> is a variable-size structure like:
<record_size><empty_flag><field1>|<field2>|...|<fieldN>#

<record_size> is a 4-byte unsigned integer holding size of the record. Size includes size of the <record_size>, <empty_flag>, '#' character and '|' characters.

<empty_flag> is a 1-byte unsigned char. Value is 1 (not ASCII) if record is empty, 0 (null) otherwise.

<fieldX> is the string of ASCI characters holding value of the Xth field of the record. Fields are seperated from each other by '|' character.

5.2. Join File

Join file is a hashed sequential file. Records are hashed on the field which has the same name with record id field in the main database file. Hash function returns number of block in which the record resides.

Each block can contain 24 records at most. If there are more than 24 records hashed to a block, an overflow block is added at the end of the join file and number of the block is written at the end of previous block. Do not forget that a block containing less than 24 records does not mean there is no overflow block. Always check next block parameter to be sure whether there is an overflow block or not.

Join file has following structure:
<header><blocks>

<header> is a constant-size structure of 100 bytes. Unused bytes at the end of the header is filled with 0s (null). It has following structure:
<number_of fields><field_name1>|<field_name2>|...|<field_nameN>|

<blocks> has following structure:
<block><block>...<block>

<number_of_fields> is a 4-byte unsigned integer holding the number of fields in a record.
<field_nameX> is a variable length string of ASCI characters. Field names are seperated from each other by '|' character.

<block> is a structure of constant size of 2408 bytes having following schema:
<number_of_records><record><record>...<record><next_block>

<number_of_records> is a 4-byte unsigned integer holding the number of records in current block.

<next_block> is a 4-byte unsigned integer holding number of next block. It is 0 (null) if there is no more overflow block.

<record> is a constant size structure of 100 bytes. Unused bytes at the end of the structure is filled with 0s (null). All bytes of empty records are 0(null). It is a structure like following:
<field1>|<field2>|...|<fieldN>

<fieldX> is a variable length string of ASCI characters holding value of Xth field of the record.

6. Contact

Please follow course page on the newsgroup for updates and clarifications. You can also ask your homework related questions on the newsgroup. If you need help you can find me in my office (A-402). Please send an e-mail to isikligil(at)ceng.metu.edu.tr before coming.

Office Hours: Wednesday, Thursday, Friday: 10:00 – 12:00 and anytime you find me in the office.

Kolay gelsin.
Emre