Project 2 – Developing Key-Value NoSQL database with Indexing

Due: Wednesday, March 27, 2024. 11:59:59 PM (PT)

Objective:

1. Practical experience with NoSQL database with indexing structure.
2. Experience with operating systems including file systems.
3. Keep the definition simple. Don't read things into the problem that aren't there.

This will be a group project by 2 members, so you can continue to use the group for Project 1. For Project 2, you write the program in any language that is supported under any Integrated Development Environment (IDE) and runs on your selected OS either Mac or Windows. Keep in mind that more help may be available to you in some languages than in others. Furthermore, available controls, etc. may make some of these tasks easier in one language than in another.

1 byte per char in this proj

Problem Statement:

each record: 40 bytes

Implement Key-Value based NoSQL database including following requirements:

utf-8 file, txt file

- **NoSQL database is based on Portable File System (PFS)**, i.e., each database has its own file on your operating system (more information will be given during the class)
- **PFS (i.e., database file)**:
  - File name: [database_name].db0
    - E.g.: test_group1.db0, test_group1.db1 . . .
  - Initial allocated size is 1,024 Kbytes (i.e., 1 Mbytes). Then, automatically increased by 1,024 Kbytes if needed (i.e., in case of full).
  - Block based: block size = 256 bytes
  - Block allocation method: select one of methods below:
    - Linked allocation
    - Indexed allocation
  - Head block should have the following info
    - List of File control block (FCB), i.e., file info, for example:
      - File name
      - File type: data file or index file
      - Block # (either starting block or indexed block)
      - File size (i.e., # of blocks used)
      - Index File info
      - Other information
    - Free block list
      - bit map (vector) or linked free space management

1 - data
0 - no data

- **Data File** should be Key-Value format.
  - Key is integer. If there is no integer value as a key in a data, you can generate a sequence value when each data is inserted
  - Sample data will be provided
    - movie.csv file from MovieLenz database
    - https://grouplens.org/datasets/movielens/
    - https://files.grouplens.org/datasets/movielens/ml-latest-small.zip
- **Index File** should be block-based index file
  - Indexing structure in Project 1 should be converted into block-based indexing file structure (No Memory based Indexing)
  - For example, 'node' is changed into 'block', then blocking factor (i.e., node size) will be determined by each record size + block size
  - Assuming key is unique. However, if your NoSQL database can handle duplicated keys, it is credited as extra points.

extra credit
- handle duplicate data
- handle by string

When your program for NoSQL database is running, it should somehow accept the following commands:

open PFSfile	Allocate a new 1 MByte "PFS" file if it does not already exist. If it does exist, begin using it for further commands.
put myfile	Insert data from your OS file, i.e., "myfile" into your NoSQL database, i.e., PFS file.
get myfile	Download data file "myfile" from your NoSQL database, i.e., PFS file, and save it to the current OS directory.
rm myfile	Delete "myfile" from your NoSQL database, PFS file.
dir	List all data files in your NoSQL database, i.e., PFS file.
find [myfile.key]	Find 'value' using a given 'key', i.e., return a record which is associate with a key from 'myfile' in your NoSQL database. In addition, you need to show how many blocks are accessed during 'find'
~~putr myfile "Remarks"~~	~~Append remarks to the FCB for myfile in your NoSQL database, i.e., PFS file.~~
kill PFSfile	Delete the NoSQL database, i.e., remove PFSfile from OS file system.
quit	Exit NoSQL database.

read File Control Block (FCB)

1 block keep 6 records, 40 bytes

You can provide those commands through command line interface ONLY (No GUI will be accepted).

Limits:

1. NoSQL database is NOT a memory based file system, but based on the existing file system on your OS, such as Windows, Mac, or Linux/Unix.

2. Command-Line Interface (CLI) Only, i.e., cmd.exe (in Windows) or shell interface (in Mac). When PFS is executed, it will show the prompt as:

```
C:\> nosql.exe
NoSQL> open test_group1
```

3. Data filenames are a maximum of 20 bytes. And, file extension is db0, db1, db2 ....

read FCB

4. The directory need handle only *Name*, *Size*, *Time* and *Date*. For example,

```
NoSQL> dir
movie          128 bytes      12:30 PM  September 2
customer       512 bytes      11:00 AM  November 11
. . .
```

5. Find should return both a record value and # of blocks accessed. For example,

```
NoSQL> find weblogs.69827
```

table.key

```
69827, [15/Sep/2013:23:58:36 +0100] "GET /KBDOC-
00033.html HTTP/1.0" 200 14417
```

entire data

```
# of Blocks = 5
```

6. If the original NoSQL database file, i.e., PFS file, fills up then you should create a new database "volume" with the same name but a different suffix - e.g., pfs.db1, pfs.db2, etc., each the same size as the first "volume".

7. In your NoSQL database, each data and index file has one File Control Block (FCB) that includes *file name, file size, create time, create date, starting block ID, ending block ID* and more (if needed).

8. Your NoSQL database system should consist of two main parts, i) Directory Structure (i.e., header info) and ii) Data Blocks. Therefore, you need to define the directory data structure that includes File Control Block (FCB) for each file.

9. In addition, your data files system should be able to manage free blocks. You can use any techniques that you learned in the classroom, such as bit map (vector) or linked free space management. Free block management should be part of directory structure.

If you need to make assumptions, do so. Make a “reasonable” choice & include it in the write-up. Reasonable means that you can explain the logic behind your choice. These problems change each semester, and it is difficult to imagine every question that might come up. When in doubt, ask to Professor Lee or the TA.

Write-up

You should submit a write-up as well as your program. Your write-up should include any known bugs, limitations, and assumptions in your program. This write-up should be in text-format and titled as ‘README’. It should be submitted along with your code. TA will use the ‘README’ file to compile (or install) and run your program. If the TA has trouble with your program, then TA will contact you to makeup it.

Submission

You will submit Github link for your project repository to Canvas. Your submission should include at least (1) your sources, and (2) README.md.

Make sure your name and NEU ID are listed in both your README.md and source codes. You may resubmit your program at any time. The submission time of the latest submission will be used for determining whether the assignment is on time or not. Late submissions will be accepted at a penalty of 15 % per day (up to 2 days). In other words, it may pay you to do this project early on the off chance that something prohibits your submitting it in a timely way. If your program is not working by the deadline, submit it anyway and review it together with TA for partial credit. Do not take a zero on any project just because the program isn’t working yet. If you have trouble getting started, ask the professor or the TA.

Grading

points element

- 10 Defined Data Structure for FCB, Directory, File Block
  (should be presented during a presentation and writing-up)
- 10 Allocate new NoSQL database file
- 10 Insert data file into NoSQL database
- 5 Download data file from NoSQL database
- 5 Handle second data file Extent when full
- 5 Kill NoSQL database file
- 5 Delete data file from NoSQL database file
- 20 Find a record using a key from NoSQL database file
- 10 Use an index properly for find
- 10 DIRectory listing
- 05 Writeup
- 05 Comments in code

To receive full credit for comments in the code you should have headers at the start of every module, subroutine, or function explaining the inputs, outputs and function of the module. You should have a comment on every data item explaining what it is about. (Almost) every line of code should have a comment explaining what is going on. A comment such as /* add 1 to counter */ will not be sufficient. The comment should explain what is being counted.