DB Project

Overview

These project stages will enable you to learn how to build a database system. You will build a working single-user DBMS that can execute certain simple SQL queries. The objective is to learn how a DBMS is organized and what goes on inside it when queries are executed.
At the end of each stage we will evaluate your work. At the end of it all, you will have implemented a miniature DBMS which would take any query and output of the query would be printed to the screen.

Other Details

- Each team must consist of two members. The document to update your team details has been shared already. This will be your final team for whole course duration.
- You can use **C++/Java** for this project.
- Make your code as reusable as possible. Reuse code of each stage and avoid changing previous stage codes.
- Test your code thoroughly by using appropriate test cases.
- Make sure you know about what you (and your partner) did for the project.
- The project implementations of each group must be completely distinct from each other. You will not share any code across groups. Nor will you attempt to use any code from previous offerings of this course.

Deliverable 1

You need to implement the following **public** methods: (Refer to sample code files uploaded on courses portal. You are supposed to fill-in code segments for below mentioned methods)

void readConfig(String configFilePath)

{

/* You need to read the configuration file and extract the page size and number of pages (these two parameter together define the maximum main memory you can use). Values are in number of bytes.

You should read the names of the tables from the configuration file. You can assume that the table data exists in a file named <table_name>.csv at the path pointed by the config parameter PATH_FOR_DATA.

You will need other metadata information given in config file for future deliverables. */

}

void populateDBInfo()

{

/* The data present in each table needs to be represented in pages. Read the file corresponding to each table line by line (for now assume 1 line = 1 record)

Maintain a mapping from PageNumber to (StartingRecordId, EndingRecordId) in memory.

You can assume unspanned file organisation and record length will not be greater than page size. */

}

String getRecord(String tableName, int recordId)

{

/* Get the corresponding record of the specified table.
DO NOT perform I/O every time. Each time a request is received, if the page containing the record is already in memory, return that record else bring corresponding page in memory. You are supposed to implement LRU page replacement algorithm for the same. Print **HIT** if the page is in memory, else print **MISS <pageNumber>** where <pageNumber> is the page number of memory page which is to be replaced. (You can assume page numbers starting from 0. So, you have total 0 to <NUM_PAGES - 1> pages.) */

}

void insertRecord(String tableName, String record)

{

/* Get the last page for the corresponding Table in main memory, if not already present.
If the page has enough free space, then append the record to the page else, get new free page and add record to the new page. Flush modified page to the disk immediately. */

}

/*All the above mentioned methods should be present in a public class named **DBSystem.** */

Basic Format of Configuration File:

PAGESIZE xxxx
NUM_PAGES xxxx
PATH_FOR_DATA <path>

BEGIN
<TABLE1_NAME>
<ATTRIBUTE1_NAME>,<ATTRIBUTE1_TYPE>
<ATTRIBUTE2_NAME>,<ATTRIBUTE2_TYPE>
….
END

BEGIN
<TABLE2_NAME>
<ATTRIBUTE1_NAME>,<ATTRIBUTE1_TYPE>
<ATTRIBUTE2_NAME>,<ATTRIBUTE2_TYPE>
<ATTRIBUTE3_NAME>,<ATTRIBUTE3_TYPE>
….
END
….

/*ATTRIBUTE_TYPE will be either of **int, string, fixed_char(<size>), float.** */