German University in Cairo
Faculty of Media Engineering and Technology
Spring 2025
Dr. Mohamed Karam

CSEN604 – Data Bases II

Project MS1 Description
(Deadline 10/4 at 11:59 pm)

Introduction:

The main objective of this project is to dive deeper into a subset of the components and functionality provided by a DBMS. Moreover, further focus and attention are to be directed toward the assessment and analysis of the behavior and performance of these components.

Taking milestones one in perspective, the functionalities to be **implemented** are:
1. Create table.
2. Insert into table.
3. Select from table:
   3.1. All data.
   3.2. Based on a condition (as values per columns).
   3.3. Using pointer (as direct record access).
4. Retrieve full (or partial) trace of operations performed on a table.

These requirements, *theoretically speaking,* are implemented as per the following historical structure discussed in class:
1. DBMS level.
2. Table level.
3. Page level.

For the rest of this document, further elaborations regarding requirements, grading, starter code, and simplification assumptions are provided.

1. Simplification Assumptions:

Towards achieving the main objective of this project, some assumptions are made to simplify some tasks, opening more focus room for others.
These assumptions are **as follows:**

1. **Data types normalization:** For this milestone, all data are assumed to be of type **STRING**, such that no data types transformation is required, nor null/empty cells are to be taken into account.

2. **All selects are star selects (SELECT *):** There will be no selection of specific columns to be returned, conditions will apply to the selection of the records **(as the WHERE part).**

3. **Arraylist of string arrays as select output:** As there is no reordering for columns, and for

uniform abstraction, the output of select functions is **Arraylist of string arrays**, which has no restriction on the data structure used for storage and operations done on the data before the output, but the output must be an **Arraylist**.

4. **String array as insertion input:** As per the data types simplification performed, a record to be inserted into a table is an array of strings.

5. **Primary key of the table:** No primary key of the table is required. Thereby, inserting a new record always goes to the end of the table.

2. Grading:

Grading for this milestone will be performed using a set of unit tests for the DBMS level, each with equal weight. In other words, the ratio of the correct (green) tests with respect to the total number of test cases is the assigned grade.
There will a test file provided with the starter code, you can use this for the initial implementation steps, the rest of the tests will be published one week after the publishing of the project description.

3. Starter Code:

There will be three essential Java files provided:

1. `DBApp.java`
2. `FileManager.java`
3. `DBAppTests.java`

You can add any optional files, like:

1. `Table.java`
2. `Page.java`

The optional files, with their provided test files, are for guidance to start coding from different difficulty levels. For detailed guidance, start from the optional ones, and for advanced coding, start from the DBMS level. **Only the DBApp code and test files will be graded.**

4. Implementation details:

The project consists of 4 main components:

1. `DBAppTests.java:`
   This file contains the test units for the advanced level, you are not allowed to change anything in this file.
   This will be the same file used for grading after submission.

2. `FileManager.java:`
   This file is responsible for serializing and deserializing the table with its pages, you are not allowed to change anything in this file for it to work correctly. It starts by getting the directory of the project file to create a folder to store all the tables in it, called **"Tables"**, this folder contains new folders for each new table created. This folder contains the serialized table file, with pages of the table.
   The methods that are in the file are:

   a. `public static boolean storeTable(String tableName, Table t)`
      This function is responsible for storing the table in its specific folder as a **'.db'** file that is considered as the **METADATA FILE** for the table, the METADATA file for the table contains all the information\attributes of the table **WITHOUT** the actual data of it. It takes the table name and table object as input and returns True if the table is stored normally, and False if there is an error storing the table.

   b. `public static Table loadTable(String tableName)`
      This function is responsible for retrieving the saved table file from the **'Tables'** directory. It takes the table name as input and returns the loaded table.

   c. `public static boolean storeTablePage(String tableName, int pageNumber, Page p)`
      This function is responsible for storing the pages of the table. It stores it as a number in the table folder, and acts just like the `storeTable` function, but it takes the table name, the page number, and the page object.

   d. `public static Page loadTablePage(String tableName, int pageNumber)`
      This function is responsible for loading a specific page from a table by its index from the table folder, it acts just like the `loadTable` function, but takes the table name and the page number as input and returns the loaded page.

   e. `public static void reset()`
      This function is responsible for emptying the **'Tables'** directory by deleting all the folders and files inside of it.

   f. `public static String trace()`
      This function is responsible for returning the **'Trace'** file of the directory, the trace is a string that contains the table names with all their page names too.

      Example for the file manager trace:
      `Tables{ student{ 0.db 1.db student.db } }`, this means that there is only one table created, that contains 2 pages.

3. DBApp.java:
   This folder contains the main project implementation, the DBApp class will contain functions that need to be implemented based on the assumptions above. You are **not allowed** to change any attribute name or function signature at all.
   The following is the details of the DBApp class:

   a. The `dataPageSize` is the maximum number of rows/tuples allowed within a single page, it is set only once with the value at the start of the class to be used later in the class.

   b. `public static void createTable(String tableName, String[] columnsNames)`
      This function is responsible for creating a new table with its column names, the table should be stored on the hard disk in the 'Tables' directory, without creating new pages yet.

   c. `public static void insert(String tableName, String[] record)`
      This function is responsible for inserting a new record into the table, the record should be inserted into the pages of the specific table, and then the table and its pages should be updated on the hard disk.

   d. `public static ArrayList<String []> select(//Different Parameters)`
      This function is responsible for selecting from the table, depending on the parameters the function takes, the output is determined:
      - For the input of only the table name, the output is equivalent to selecting everything from the table.
      - For the input of the table name, the column name(s), and the value(s), this is equivalent to selecting with some conditions in the **WHERE**.
      - For the input of the table name, a page number, and a record number, this returns the specific record from that page only, without passing through the whole table.

   e. The **'Trace'** of the table is a set of strings that contain the operations and changes done to the table, with the time of the operation execution. There are two types of trace getters:
      - `public static String getFullTrace(String tableName):` This gets the full trace string of the table, this trace string contains all the operations done on the table.
      - `public static String getLastTrace(String tableName):` This gets only one string that contains the last operation done on the table.

   f. `public static void main(String []args)`
      This contains the main code that will be used for testing the code implementation and outputs before trying the test files.

4. The optional classes are used for deep-level implementation and guidance, alongside the test files for these classes.

5. Main method example:

This is an example of the main method:

```
	public static void main(String []args) throws IOException
	{

		String[] cols = {"id","name","major","semester","gpa"};
		createTable("student", cols);
		String[] r1 = {"1", "stud1", "CS", "5", "0.9"};
		insert("student", r1);

		String[] r2 = {"2", "stud2", "BI", "7", "1.2"};
		insert("student", r2);

		String[] r3 = {"3", "stud3", "CS", "2", "2.4"};
		insert("student", r3);

		String[] r4 = {"4", "stud4", "DMET", "9", "1.2"};
		insert("student", r4);

		String[] r5 = {"5", "stud5", "BI", "4", "3.5"};
		insert("student", r5);


	  System.out.println("Output of selecting the whole table content:");
		ArrayList<String[]> result1 = select("student");
	  for (String[] array : result1) {
	      for (String str : array) {
	          System.out.print(str + " ");
	      }
	      System.out.println();
	  }

	  System.out.println("--------------------------------");
	  System.out.println("Output of selecting the output by position:");
		ArrayList<String[]> result2 = select("student", 1, 1);
	  for (String[] array : result2) {
	      for (String str : array) {
	          System.out.print(str + " ");
	      }
	      System.out.println();
	  }

	  System.out.println("--------------------------------");
	  System.out.println("Output of selecting the output by column condition:");
		ArrayList<String[]> result3 = select("student", new String[]{"gpa"}, new
String[]{"1.2"});
	  for (String[] array : result3) {
```

```
            for (String str : array) {
                System.out.print(str + " ");
            }
            System.out.println();
        }
        System.out.println("--------------------------------");
             System.out.println("Full Trace of the table:");
             System.out.println(getFullTrace("student"));
             System.out.println("--------------------------------");
             System.out.println("Last Trace of the table:");
             System.out.println(getLastTrace("student"));
             System.out.println("--------------------------------");
             System.out.println("The trace of the Tables Folder:");
             System.out.println(FileManager.trace());
             FileManager.reset();
             System.out.println("--------------------------------");
             System.out.println("The trace of the Tables Folder after resetting:");
             System.out.println(FileManager.trace());


    }
```

This is the output of this code snippet:

```
Output of selecting the whole table content:
1 stud1 CS 5 0.9
2 stud2 BI 7 1.2
3 stud3 CS 2 2.4
4 stud4 DMET 9 1.2
5 stud5 BI 4 3.5
--------------------------------
Output of selecting the output by position:
4 stud4 DMET 9 1.2
--------------------------------
Output of selecting the output by column condition:
2 stud2 BI 7 1.2
4 stud4 DMET 9 1.2
--------------------------------
Full Trace of the table:
Table created name:student, columnsNames:[id, name, major, semester, gpa]
Inserted:[1, stud1, CS, 5, 0.9], at page number:0, execution time (mil):5
Inserted:[2, stud2, BI, 7, 1.2], at page number:0, execution time (mil):6
Inserted:[3, stud3, CS, 2, 2.4], at page number:1, execution time (mil):5
Inserted:[4, stud4, DMET, 9, 1.2], at page number:1, execution time (mil):5
Inserted:[5, stud5, BI, 4, 3.5], at page number:2, execution time (mil):6
Select all pages:3, records:5, execution time (mil):8
Select pointer page:1, record:1, total output count:1, execution time (mil):2
Select condition:[gpa]->[1.2], Records per page:[[0, 1], [1, 1]], records:2,
execution time (mil):6
Pages Count: 3, Records Count: 5
--------------------------------
Last Trace of the table:
```

```
Select condition:[gpa]->[1.2], Records per page:[[0, 1], [1, 1]], records:2,
execution time (mil):6
-------------------------------
The trace of the Tables Folder:
Tables{ student{ 0.db 1.db 2.db student.db } }
-------------------------------
The trace of the Tables Folder after resetting:
Tables{ }
```

6. Team and submission rules:

Please note the following rules for the project:

1. The project team is up to 4 members.
2. No random assignments of teams will be formed, it's your responsibility to find a team, or work alone on the project.
3. The submission link will be available on the CMS one week before the deadline.
4. You are required to submit a zip file containing the following:
   a. A text file containing the names, IDs, Tutorials, and majors of the team members.
   b. The whole code folder, exported as a zip file.
5. You ARE NOT allowed to change any method name given from the starter code.