Group Project - Building a simple and portable distributed database, DDBMS, and analytics systems

Product Title: D2_DB
Team Size: 5 to 6
Required Programming Language: Core Java

Project Objective:

By completing this project, a student will be able to describe concepts in modelling notation (e.g., Entity-Relation Diagrams or UML) and how they would be used. A student will be able to describe the most common designs for core database system components including the query optimizer, query executor, storage manager, access methods, and transaction processor. By completing this task, a student will be able to explain the techniques used for data replication, allocation during the database design process, and data analytics

In this project, each group is required to create a light-weight distributed database management system (e.g. metadata management, data structure design, data storing, retrieval, building database and logs, analysis, and security), with custom database structure for in memory operations using various data structures such as Arrays, Tree, Linked Lists etc., and custom file design for persistent storage.

Plagiarism Policy:

- This project is a group task. Collaboration of any type (outside the assigned group) amounts to a violation of the academic integrity policy and will be reported to the AIO.
- Content cannot be copied verbatim from any source(s). Please understand the concept and write in your own words. In addition, cite the actual source. Failing to do so will be considered as plagiarism and/or cheating.
- The Dalhousie Academic Integrity policy applies to all material submitted as part of this course. Please understand the policy, which is available at: https://www.dal.ca/dept/university_secretariat/academic-integrity.html

Project Rubric:

Based on the discussion board rubric (McKinney, 2018)

	Excellent (20%)	Proficient (15%)	Marginal (10%)	Unacceptable (0%)
Completeness including Citation	All required tasks are completed	Submission highlights tasks completion. However, missed some tasks in between, which created a disconnection	Some tasks are completed, which are disjoint in nature.	Incorrect and irrelevant
Correctness	All parts of the given tasks are correct	Most of the given tasks are correct However, some	Most of the given tasks are incorrect. The submission	Incorrect and unacceptable

		portions need minor modifications	requires major modifications.	
Novelty	The submission contains novel contribution in key segments, which is a clear indication of application knowledge	The submission lacks novel contributions. There are some evidences of novelty, however, it is not significant	The submission does not contain novel contributions. However, there is an evidence of some effort	There is no novelty
Clarity	The written or graphical materials, and developed applications provide a clear picture of the concept, and highlights the clarity	The written or graphical materials, and developed applications do not show clear picture of the concept. There is room for improvement	The written or graphical materials, and developed applications fail to prove the clarity. Background knowledge is needed	Failed to prove the clarity. Need proper background knowledge to perform the tasks
Group Work	Evidence of group work, meeting logs, Coordination	Evidence of group work. However, missed meeting logs, room for improvement in Coordination	Missed meeting logs, failed to display group coordination	No group work done. Project is unacceptable

Project Requirements

In this project, you need to build a simple distributed database management system (D2_DB) that will operate in **2 Linux virtual instances** in the Google Cloud Platform. Your team should explore and implement data structure concepts for creating the databases, its management system, and security. In addition, your team should build an analytics engine to perform basic data analytics. Your database should handle requests handle from 2 users (one user for each VM instance). The database management system layer should provide a command-line interface (Graphical User Interface is not required), and perform various functionalities of database management system listed below.

Functional Requirements:

Module 1: DB Design

1. You need to identify one or two **linear data structure**(s), which can be used for query, and data processing.
2. Once the data is processed it can be stored in a customized file format, which will be considered as your persistent storage
   Customized File Format cannot be in any of these formats - JSON/XML/CSV/Serialized or binary
3. You need to maintain data dictionary or meta data file for each local DBMS, and a global metadata file; You can use same customized file format to maintain the data dictionary.

Module 2: Query Implementation

1. Your application should validate and execute the following operations. (Standard SQL command and queries only)
    a. Create database
    b. Use database
    c. Create table
    d. Insert into table
    e. Select from table with single where condition (AND || OR || NOT are not required)
    f. Update one column with single where condition (AND not required)
    g. Delete a row with single where condition

Module 3: Transaction Processing

1. Your system should identify what is a transaction and what is a normal query.
2. In case of transaction, you cannot write the data immediately to persistent storage, you need to perform the operation in the linear data structure.
3. Transaction follows ACID properties, so you need to consider that. [Hint: It can be achieved by scanning logs]

{**You do need to work on concurrency control. Single distributed transaction is sufficient**}

Module 4: Log Management

You need to create 3 Logs – **JSON format is allowed for Logs**

a. General Logs: query execution time, state of the database (e.g. how many tables are there with number of records at a given time)
b. Event Logs: changes in database, concurrent transactions, crash reports, etc.
c. Query Logs: capture user queries and timestamp of query submission

{Do not use any in-build package or libraries; You need to perform normal file read write operations to capture the query, timestamp, login details, data change etc.}

Module 5: Data Modelling – Reverse Engineering

1. Your application should provide option for generating an ERD based on **current database state**.
2. User will provide the database name, and the application will create ERD based on the metadata, and data files.

{**You do not have to create any graphical ERD, a text file containing tables, columns, relationships, cardinality will be sufficient**}

Module 6: Export Structure & Value

1. Your application should provide an option for exporting structure and values (in standard SQL format) for each database, which is selected by the end user.
2. This is like SQL dump (structure+value) created by any standard DBMS Export option.
3. You cannot use any external packages, and as mentioned, your export format must be in SQL
4. Your data dump must capture the current state of the database (E.g. if there is any update performed on a data, it must be reflected in the data dump. Do not write ***create*** or ***insert*** statements from console input to a file as the SQL export. You must dynamically generate it)

Module 7: Analytics

1. The D2_DB application should provide some basic analytics – The results must be written in files, and displayed on screen
    a. How many queries (valid + invalid) are submitted by Database. (E.g. ran in VM1 instance)
    E.g.: >> count queries;
    "user SDEY submitted 10 queries for DB1 running on Virtual Machine 1"
    "user Alex submitted 3 queries for DB2 running on Virtual Machine 2"
    b. How many Update operations are successfully executed by Tables
    E.g. >> count update DB1;
    "Total 9 Update operations are performed on Employee"
    "Total 3 Update operations are performed on Department"

Module 8: User Interface & Login Security

1. Your user interface design should be basic console based
2. It must provide access to the valid user only.
3. User interface can be menu driven, and provide options for registration or login
4. For registration, it should accept the userID, password, and security questions/answers - and store all information in the **User_Profile** text file
    a. Use some hashing like md5, Sha1 etc. and store the hashed UserID, hashed password in the file. (you can use Java libraries for hashing)
    b. You do not have to encrypt security question/answer.
5. If a registered user wants to access the system he/she/they needs to provide valid UserID, and password, which will be hashed, and checked with the entry available in the User_Profile text file. The security answer will also be asked at login. Since you are building a distributed database management system, **User_Profile** text file needs to be present in both vistual machine instances.
6. Once the users gain access, they get 5 options.
    E.g.
        1. Write Queries
        2. Export
        3. Data Model
        4. Analytics
7. The "Write Queries" option should work for both normal queries and transaction.

Deliverables:

Phase 1 (Feasibility Study and Design): (2%)

- Each group will work on the problem and identify a solution.
- Each group will perform a feasibility study and submit a short-recorded group presentation with their meeting Logs. (only 1-member can present for 10 min. as group representative)
- Each group will submit their tentative design, and implementation plan (maximum 5 pages. Include citations if you use any source for background study, coding skills, or concepts.)

Phase 2 (Go-Live and Project Closure): (20%)

- Each group will complete the task incorporating the changes suggested in Phase 1

- Each group will complete the project implementation and record the details in the form of a final report
- Each group will submit a report (40 to 65 pages) and a recorded group presentation explaining major code blocks, and demonstration of the product. (every member needs to present)
    - Final Report should contain architecture details, design details, implementation details of each module, use case, test case, meeting logs etc.

- There will be a synchronous Q&A session with each group at the end of the course

Visual Representation for your understanding