CSCI 5408

Data Management and Warehousing

Group-04

Project: Tiny DB

SPRINT 2 REPORT

Group Members:

Ramya Kommalapati (B00982851)

Parth Madhvani (B00981268)

Shifa Mirza(B00981843)

GitLab:

https://git.cs.dal.ca/mirza/csci_5408_s24_4

Contents

Table of Figures:

PSEUDO CODE:

CLASS Logger :

CONSTANTS:

```
    LOG_DIRECTORY_GENERAL = "logs/general/"
    LOG_DIRECTORY_QUERY = "logs/query/"
    LOG_DIRECTORY_EVENT = "logs/event/"
    LOG_FILE = "_TinyDBGroup04.log"
    DATE_FORMAT = "MM-dd-yyyy"
    TIMESTAMP_FORMAT = "MM-dd-yyyy HH:mm:ss"
```

METHOD log(String directoryPath, LogCategory category, String message)

```
    // Get  today's date formatted as DATE_FORMAT
    today = current date formatted as DATE_FORMAT
```

TRY:

```
      // Open a PrintWriter with append mode for the log file
      open PrintWriter using FileWriter(directoryPath + today + LOG_FILE, append mode)

      // Get current timestamp formatted as TIMESTAMP_FORMAT
      timestamp = current timestamp formatted as TIMESTAMP_FORMAT

      // Write log entry in the format: [timestamp] ### [category] message
      write "[timestamp] ### [category] message" to the file
```

CATCH IOException:

```
            print error stack trace
```

METHOD createLogDirectory(String directoryPath)

```
    // Create a File object for the specified directory path
    logDir = create File object using directoryPath
    IF logDir does not exist THEN
      // If directory doesn't exist, attempt to create it
      IF create directories(logDir) THEN
        // Print success message if directory creation was successful
        print "Log directory created successfully: directoryPath"
      ELSE
        // Print error message if directory creation failed
        print "Failed to create log directory: directoryPath"
```

METHOD logGeneral(String message)
// Ensure the general log directory exists
createLogDirectory(LOG_DIRECTORY_GENERAL)
// Log the message under the GENERAL category
log(LOG_DIRECTORY_GENERAL, LogCategory.GENERAL, message)

METHOD logQuery(String message)
// Ensure the query log directory exists
createLogDirectory(LOG_DIRECTORY_QUERY)
// Log the message under the QUERY category
log(LOG_DIRECTORY_QUERY, LogCategory.QUERY, message)

METHOD logEvent(String message)
// Ensure the event log directory exists
createLogDirectory(LOG_DIRECTORY_EVENT)
// Log the message under the EVENT category
log(LOG_DIRECTORY_EVENT, LogCategory.EVENT, message)

ENUM LogCategory
// enum constants for different log categories
GENERAL
QUERY
EVENT

CLASS DBMS:

METHOD loadTable(tableName):
// Construct the file path for the table data

filePathForTable = current_directory + "/databases/test/" + tableName + "/data.txt"
// Initialize a new Table object with the given table name
table = new Table(tableName)
Try:
// Open the file for reading
bufferedReader = new BufferedReader(new FileReader(filePathForTable))
i = 0 // Line counter to distinguish between header and data lines
While (line = bufferedReader.readLine()) is not null:
// Split the line into columns using the delimiter " ### "
columns = line.split(" ### ")
For j = 0 to length of columns:

```
            If i == 0:
                // If it's the first line, treat it as the header and add columns to the table
                column = new Column(columns[j])
                table.addColumn(column)
            Else:
                // For subsequent lines, add data to the corresponding columns
                temp = table.getColumns().get(j)
                temp.data.add(columns[j])
        i++  // Increment the line counter
    Catch Exception as e:
        // Print the error message if an exception occurs
        Print(e.getMessage())
    // Return the populated table
    Return table
```

METHOD checkCardinality(tableName1 , tableName2):

```
Function checkCardinality(tableName1, tableName2):
    // Load the second table
    table2 = loadTable(tableName2)
    // Initialize variables for referenced columns
    referencedColumnIn1 = null
    columnWhichRefersIn2 = null
    // Construct the file path for the metadata of the second table
    filePathForTable1 = current_directory + "/databases/test/" + tableName2 + "/metadata.txt"
    Try:
        // Open the metadata file for reading
        bufferedReader = new BufferedReader(new FileReader(filePathForTable1))
        While (line = bufferedReader.readLine()) is not null:
            // Split the line into content using space as delimiter
            content = line.split(" ")
            // Check if the content array has more than 2 elements
            If length of content > 2:
                // Assign the first and fourth elements to the respective variables
                referencedColumnIn1 = content[0]
                columnWhichRefersIn2 = content[3].split(".")[1]
    Catch Exception as e:
        // Print the error message if an exception occurs
        Print(e.getMessage())
    // Check if the referenced column in table2 contains duplicates
    isColumnWhichRefersDuplicate = containsDuplicates(table2.getColumn(referencedColumnIn1).data)
    If isColumnWhichRefersDuplicate:
```

```
        // Return "1-to-N" if duplicates are found
        Return "1-to-N"
    Else:
        // Return "1-to-1" if no duplicates are found
        Return "1-to-1"
```

METHOD containsDuplicates(list)

```
Function containsDuplicates(list):
    // Initialize a HashSet to store unique items
    set = new HashSet()
    // Iterate over each item in the list
    For each item in list:
        // Try to add the item to the set
        If not set.add(item):
            // If add returns false, a duplicate was found
            Return true  // Duplicate found
    // If the loop completes without finding duplicates
    Return false  // No duplicates found
```

METHOD createERD(databaseName):

```
Function createERD(databaseName):
    // Check if the database exists
    If not isDataBasePresent(databaseName):
        Print "Database does not exist"
        Return false

    // Construct the path to the database
    databasePath = current_directory + "/databases/" + databaseName
    databases = new File(databasePath)

    // Get a list of files in the database directory
    files = databases.listFiles()
    Print(files)

    // Construct the path for the ERD file
    ERDpath = current_directory + "/databases/" + databaseName + "/erd.txt"

    // Iterate over each file in the database directory
    For each file in files:
        If file.isDirectory():
```

```
        Try:
            // Open the metadata file for reading
            bufferedReader = new BufferedReader(new FileReader(file.getPath() + "/metadata.txt"))

            tableStructure = new List()
            tableName = file.getName()
            referencedTable = ""
            tableStructure.add(tableName)
            line = ""
            hasReference = false
            cardinality = ""
            // Read the metadata file line by line
            While (line = bufferedReader.readLine()) is not null:
                If line contains "references":
                    hasReference = true
                    referencedTable = line.split(" ")[3].split(".")[0]
                    cardinality = checkCardinality(referencedTable, tableName)
                tableStructure.add(line)
            // Open the ERD file for writing in append mode
            bufferedWriter = new BufferedWriter(new FileWriter(ERDpath, true))
            For i = 0 to length of tableStructure:
                If i == 0:
                    bufferedWriter.write("Table: " + tableName)
                    bufferedWriter.newLine()
                    If hasReference:
                        bufferedWriter.write("Cardinality: " + referencedTable + " " + cardinality + " " +
tableName)
                        bufferedWriter.newLine()
                Else:
                    bufferedWriter.write("\t" + tableStructure.get(i))
                    bufferedWriter.newLine()
            bufferedWriter.write("**************************")
            bufferedWriter.newLine()
            bufferedWriter.close()
        Catch Exception as e:
            Print(e.getMessage())
    Return true
```

exportDataStructure Method

1. Initialize a Scanner object for user input.
2. Define databaseFolder as a new File object pointing to the "./databases" directory.
3. List all files in the databaseFolder and store them in dbs.
4. Print "Choose option from below".
5. Loop through dbs and print the available database options for the user to choose from.
6. Prompt the user to enter their choice.
7. Adjust the user's input to match the array index.
8. If the user's choice is invalid (out of range), print "Invalid choice".
9. Otherwise, call generateSqlDump with the selected database's absolute path.

generateSqlDump Method

10. Define generateSqlDump to take databaseFullPath as a parameter.
11. Call the export method of SQLDumpUtility with databaseFullPath.

SQLDumpUtility Class

export Method

12. Define export to take databasePath as a parameter.
13. Create a File object dbFolder for the databasePath.
14. Extract the database name from dbFolder.
15. Define dumpFilePath as the path for the SQL dump file.
16. Check if the dump file already exists.
    - If it exists, delete it and print "Delete existing dump".
17. Create a FileWriter object writer for the dumpFilePath.
18. Write SQL commands to create and use the database.
19. Find all data.txt files in the database folder using findDataFiles.
20. Loop through each data.txt file.
    - Determine the table name from the parent folder.
    - Read the metadata from metadata.txt.
    - Generate the SQL dump for the table using generateTableSQLDump.
    - Write the SQL dump to the file.
21. Print a success message with the SQL dump file path.

findDataFiles Method

22. Define findDataFiles to take databaseFolderPath as a parameter.
23. Create a list dataFiles to store found data files.
24. Create a File object folder for the databaseFolderPath.
25. Call findDataFilesRecursive with folder and dataFiles.
26. Return dataFiles.

findDataFilesRecursive Method

27. Define findDataFilesRecursive to take folder and dataFiles as parameters.
28. List all files in the folder.
29. Loop through each file.

- If the file is a directory, call findDataFilesRecursive recursively.
- If the file is a data.txt file, add it to dataFiles.

readMetadata Method

30. Define readMetadata to take metadataFilePath as a parameter.
31. Create a StringBuilder object metadataBuilder.
32. Open a BufferedReader for metadataFilePath.
33. Loop through each line in the file.
    - Append the line to metadataBuilder.
34. Return metadataBuilder.toString().

generateTableSQLDump Method

35. Define generateTableSQLDump to take tableName, dataFilePath, and metadata as parameters.
36. Create a StringBuilder object sqlDumpBuilder.
37. Append the CREATE TABLE statement to sqlDumpBuilder.
38. Split metadata into lines.
39. Loop through each line in metadata.
    - Split the line into columnName, dataType, and optional constraints.
    - Append the column definition to sqlDumpBuilder.
40. Close the CREATE TABLE statement.
41. Open a BufferedReader for dataFilePath.
42. Loop through each line in the file.
    - Skip the first line (header).
    - Skip empty lines.
    - Split the line into values.
    - Remove single quotes from values.
    - Format values as strings or integers.
    - Append INSERT INTO statements to sqlDumpBuilder.
43. Return sqlDumpBuilder.toString().

isInteger Method

44. Define isInteger to take value as a parameter.
45. Try to parse value as an integer.
46. Return true if parsing is successful, otherwise return false.

TEST CASES AND EVIDENCE OF TESTING:

Description: The project folder structure before the creation of log directory

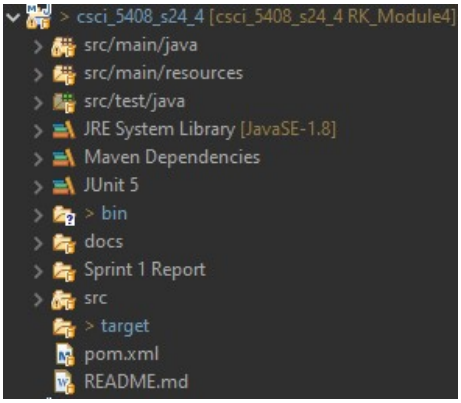

Figure 1 : Before the creation of logs directory

Description : The application is started so the general logs is created because the start and exit of program comes under general log and since general log doesn't exists before it was created

```
Log directory created successfully: logs/general/
Main Menu
1. Register
2. Login
3. Exit
Enter your choice:
```

Figure 2 : general log created successfully

Description: As the application started, and user registration was chosen the event directory was created as it does not exist previously

```
Log directory created successfully: logs/general/
Main Menu
1. Register
2. Login
3. Exit
Enter your choice: 1
Log directory created successfully: logs/event/

--- User Registration ---
Enter UserID (email):
```

Figure 3 : Event log is created successfully

Description: Once the user login is successful, he can write and execute the DB queries so option to write queries was chosen as event logs exists before the log was created

```
Main Menu
1. Register
2. Login
3. Exit
Enter your choice: 2

--- User Login ---
Enter UserID: test1@example.com
Enter Password: 123456789
What is your pet's name?: test
What city were you born in?: test
What is your favorite movie?: test
Login successful.

User Menu
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Log directory created successfully: logs/query/
```

Figure 4 : Query log was created successfully

Description: The folder structure of the project after the logs directory was created.

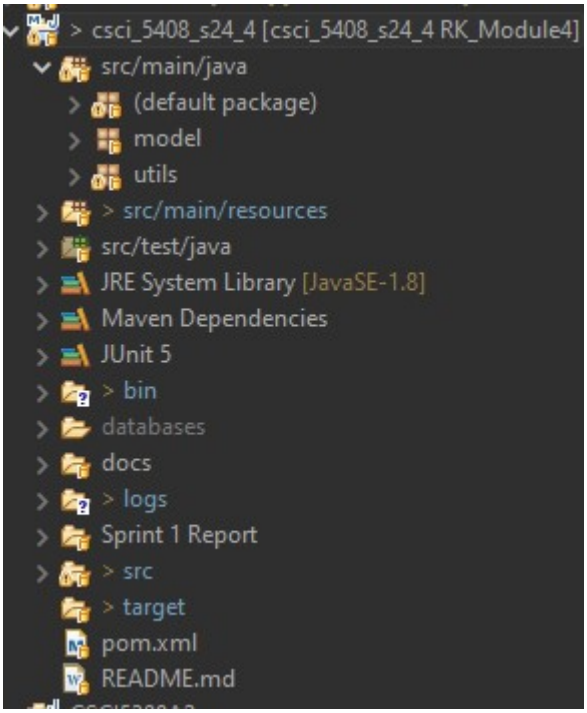

Figure 5 : Project folder structure after logs directory created

Description: As there are different log categories defined in the enum so different log categories folders were created in the log directory

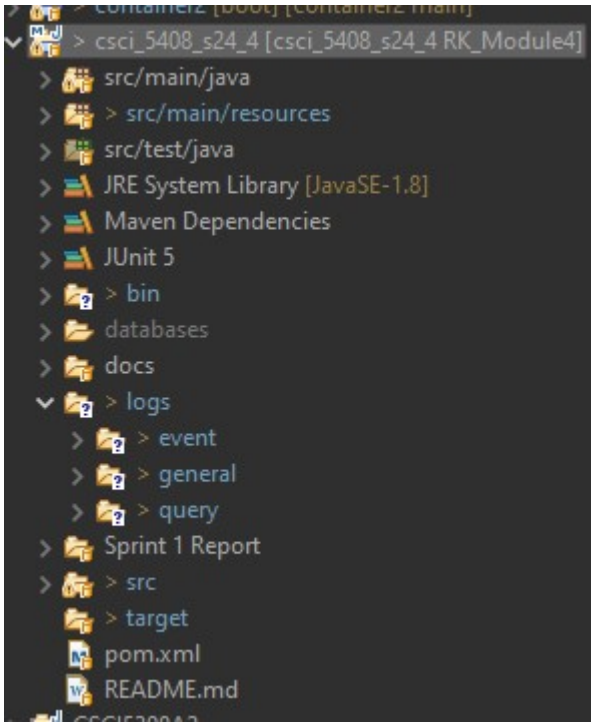

Figure 6 : different categories of logs created

Description: The naming convention of the log text files that stores logs. These are synchronized and create new log text file for every day.

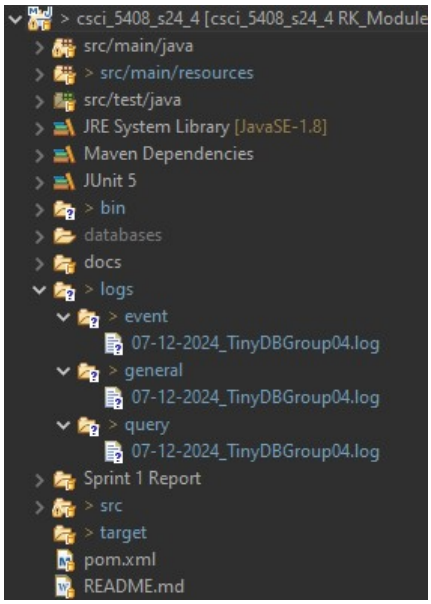

Figure 7 : The file name format for each category of log

Description: As the application was started and series of activates are done like registering of user, writing queries etc .so the logs are recorded as below in general log file

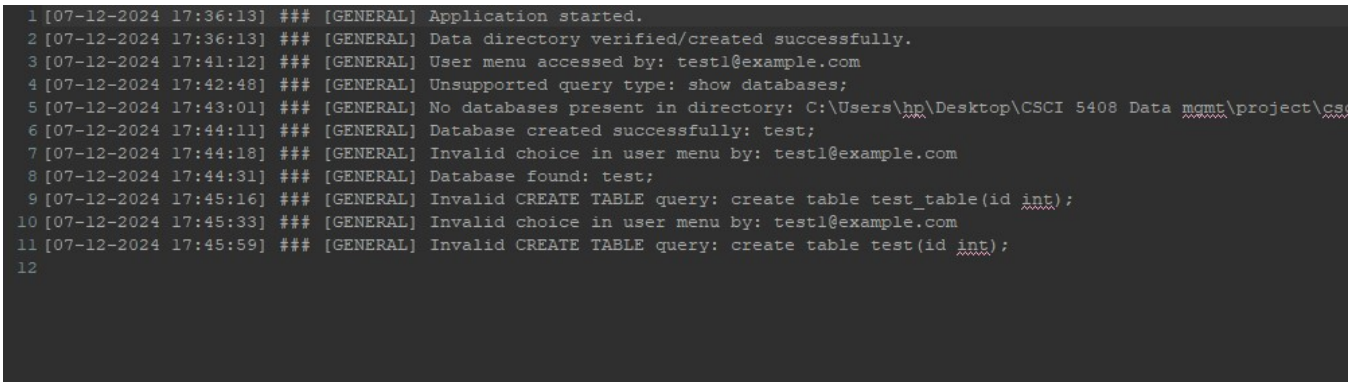

Figure 8 : The general log file

Description: As the application was started and series of activates are done like registering of user, writing queries etc .so the log are recorded as below in query log file

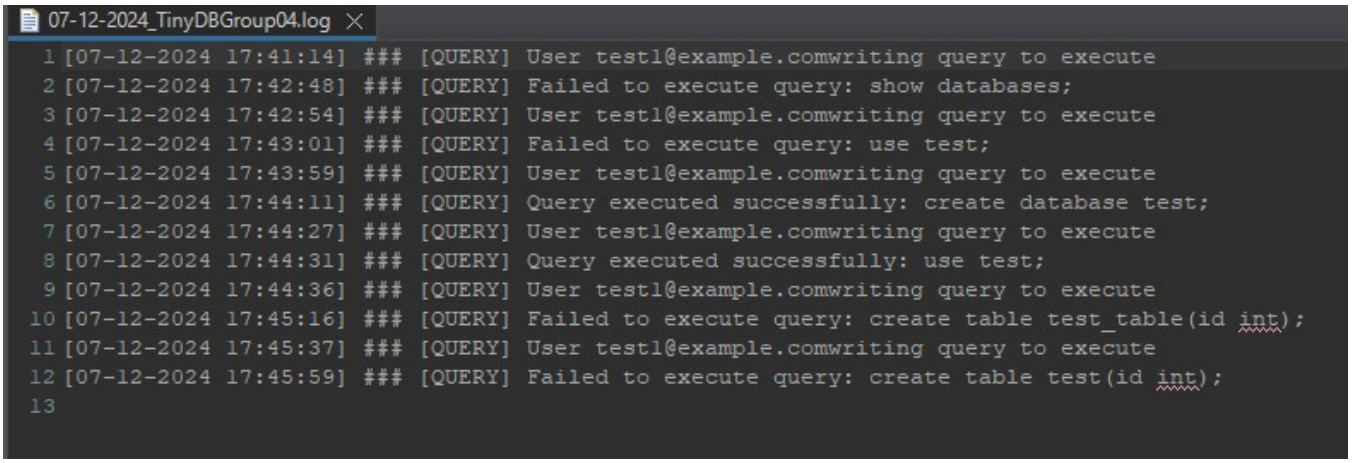

Figure 9 : The query log file after executing some queries in the application

Description: As the application was started and series of activates are done like registering of user ,writing queries etc .so the log are recorded as below in event log file

```
54 [07-12-2024 17:40:38] ### [EVENT] User login started.
55 [07-12-2024 17:41:05] ### [EVENT] User deserialized from string: ramyak@example.com
56 [07-12-2024 17:41:05] ### [EVENT] User created: ramyak@example.com
57 [07-12-2024 17:41:05] ### [EVENT] User deserialized from string: test@example.com
58 [07-12-2024 17:41:05] ### [EVENT] User created: test@example.com
59 [07-12-2024 17:41:05] ### [EVENT] User deserialized from string: p@g.com
60 [07-12-2024 17:41:05] ### [EVENT] User created: p@g.com
61 [07-12-2024 17:41:05] ### [EVENT] User deserialized from string: test1@example.com
62 [07-12-2024 17:41:05] ### [EVENT] User created: test1@example.com
63 [07-12-2024 17:41:05] ### [EVENT] User deserialized from string: test5@example.com
64 [07-12-2024 17:41:05] ### [EVENT] User created: test5@example.com
65 [07-12-2024 17:41:05] ### [EVENT] Users loaded successfully from file: src/main/resources/users.txt
66 [07-12-2024 17:41:12] ### [EVENT] Login successful: test1@example.com
67 [07-12-2024 17:43:01] ### [EVENT] Executing USE DATABASE query: use test;
68 [07-12-2024 17:44:11] ### [EVENT] Executing CREATE DATABASE query: create database test;
69 [07-12-2024 17:44:31] ### [EVENT] Executing USE DATABASE query: use test;
70 [07-12-2024 17:45:16] ### [EVENT] Executing CREATE TABLE query: create table test_table(id int);
71 [07-12-2024 17:45:59] ### [EVENT] Executing CREATE TABLE query: create table test(id int);
72
```

Figure 10 : Event logs file after the events has happened like user registration and login etc

Description: Creating table with reference keyword.

```
User Menu
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your query: create table course (courseId int,name varchar,studentId varchar references student.id)
Query executed successfully.
```

Figure 11 :Create table with references keyword

Description: Inserting table into just created table.

```
User Menu
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your query: insert into course (1,Advanced topics in software development,1)
Data inserted successfully into table: course
Query executed successfully.
```

Figure 12 Inserting data into course table

Description: Metadata of the referenced table , which is student table.

```
id int
name varchar
```

Figure 13 Student table metadata

Description: Data of the referenced table(student). Here the course table references to the student table.

```
id ### name
1 ### PARTH
2 ### ALEX
3 ### BOB
```

Figure 14 Student table data

Description: Metadata of the course table(student).The course table references to the id column of the student table.

```
courseId int
name varchar
studentId varchar references student.id
```

Figure 15 Course table metadata

Description: Data of the course table. This table references to the id column of the student table.

```
courseId ### name ### studentId
1 ### Advanced topics in software development ### 1
2 ### Database management ### 2
3 ### Cloud Architecture ### 3
```

Figure 16 Course table data

Description: Files created for both the tables.

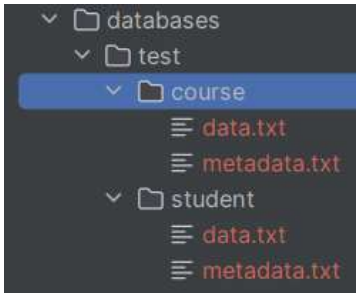

Figure 17 Both table folders created

Description: Asking user which database to export the erd..

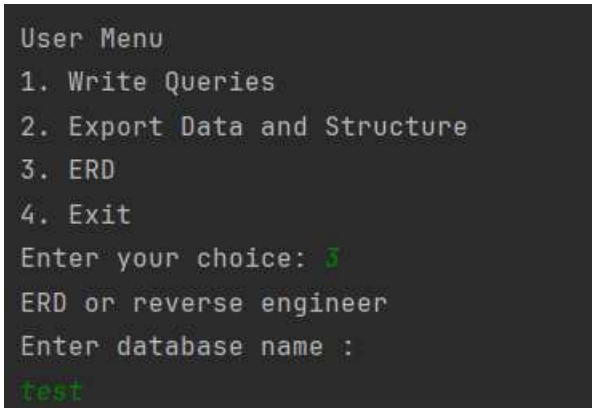

Figure 18 Asking user which database to export erd

Description: erd.txt file created.

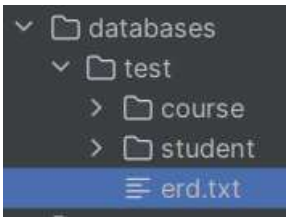

Figure 19 ERD file created

Description: Content of the erd file. The erd file contains the information like name and fields of the table, the cardinality of the table, and which field is referencing to which table.

```
Table : course
Cardinality : student 1-to-1 course
    courseId int
    name varchar
    studentId varchar references student.id
**************************
Table : student
    id int
    name varchar

**************************
```

Figure 20 ERD file

Description: Trying to reference a table which does not exist in the database. Here activity table is tryind to refer fafda table, which does not exist in the database.

```
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your query: create table activity (activityId int,name varchar,studentId int references fafda.id)
Table does not exist
Referenced table does not exist
```

Figure 21 Referenced table does not exist

Description: Trying to reference a table field which does not exist. Here the activity table is trying to reference to the idd column of student table. But the student table does not have a column with name idd.

```
User Menu
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your query: create table activity (activityId int,name varchar,studentId int references student.idd)
Referenced column does not exist
```

Figure 22 Referenced column does not exist in student table

Description: Export Empty Database

Here I'm trying to export the database which does not have any tables

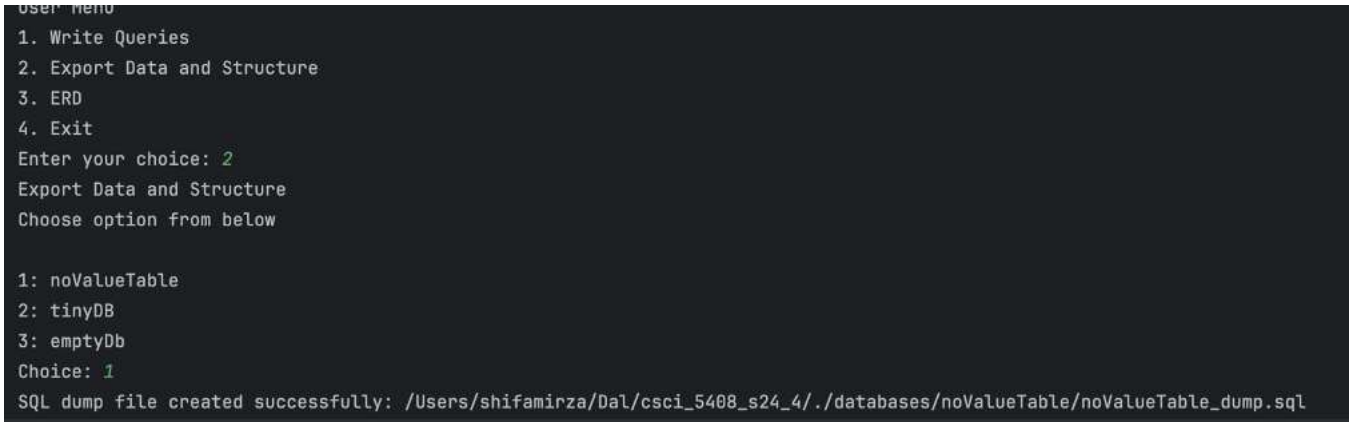

Fig 23: Empty Database

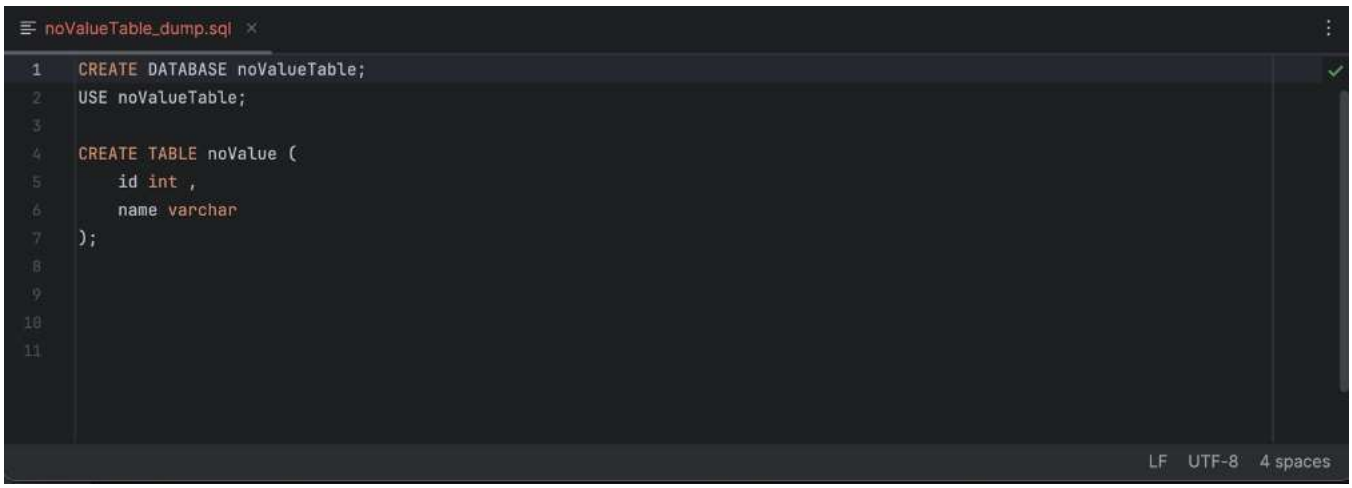

Fig 24: SQL dump for Empty Database

Description: Export table without data

Here's I'm trying to export database which has table but doesn't have any data inside the table

Fig 25: Empty Table

noValueTable_dump.sql

```sql
CREATE DATABASE noValueTable;
USE noValueTable;

CREATE TABLE noValue (
    id int ,
    name varchar
);
```

LF UTF-8 4 spaces

Fig 26: Empty Database Dump File

Description: Export table with Data

Here I'm trying to export database which as values in it

```
Enter your query: insert into student (1,'S')
Data inserted successfully into table: student
Query executed successfully.
User Menu
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 2
Export Data and Structure
Choose option from below

1: noValueTable
```

Fig 27: Table with values

tinyDB_dump.sql

```sql
CREATE DATABASE tinyDB;
USE tinyDB;

CREATE TABLE student (
    id int PRIMARY KEY,
    name varchar
);

INSERT INTO student VALUES (1, 'S');
```

Fig 28: Table with Values Dump file

Description: Dump already exists case

Here if the dump already exists then it will delete the existing dump and create new dump file

```
Enter your query: insert into student (2, 'M')
Data inserted successfully into table: student
Query executed successfully.
User Menu
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Fig 29: Insert command into table

```
Delete existing dump
SQL dump file created successfully: /Users/shifamirza/Dal/csci_5408_s24_4/./databases/tinyDB/tinyDB_dump.sql
User Menu
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Fig 30: Deleting existing dump

Description: Invalid choice

If the user selects and invalid choice amongst displayed

```
User Menu
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 2
Export Data and Structure
Choose option from below

1: noValueTable
2: tinyDB
3: TableWithValues
4: emptyDb
Choice: 5
Invalid choice
User Menu
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
```

Fig 31: User selects invalid choice

Description: Database with multiple tables

ID a database has multiple tables it will give the dump for both the tables

```
Enter your query: insert into professor (1,"Dr.Ajay")
Data inserted successfully into table: professor
Query executed successfully.
User Menu
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 2
Export Data and Structure
Choose option from below

1: noValueTable
2: tinyDB
3: TableWithValues
4: twoTableDb
5: emptyDb
Choice: 2
Delete existing dump
SQL dump file created successfully: /Users/shifamirza/Dal/csci_5408_s24_4/./databases/tinyDB/tinyDB_dump.sql
```

Fig 32: Multiple table dump

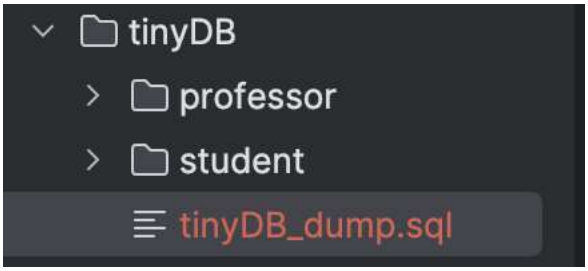

Fig 33: Multiple table dump

tinyDB_dump.sql

```sql
CREATE DATABASE tinyDB;
USE tinyDB;

CREATE TABLE student (
    id int PRIMARY KEY,
    name varchar
);

INSERT INTO student VALUES (1, 'S');
INSERT INTO student VALUES (2, 'M');


CREATE TABLE professor (
    id int PRIMARY KEY,
    name varchar
);

INSERT INTO professor VALUES (1, 'Dr.Ajay');

```

Fig 34: Multiple table dump file

Team Meeting details:

Date	Time	Agenda	Attendees	Meeting Type	Meeting Recording Link
July 05	1:30 PM	Plan for sprint 2	Parth Madhvani, Shifa Mirza, Ramya Kommalapati	Online	sprint2