CSCI-5408

DATA MANAGEMENT, WAREHOUSING, & ANALYTICS

ASSIGNMENT – 1

Banner ID: B00977669

GitLab Assignment Link:
https://git.cs.dal.ca/saji/csci5408_w24_b00977669_christin_saji

Problem 1: Build a Conceptual Model

Task 1: List of Entities and Attributes

Table 1 20 Entity set with their attributes and reasoning

S.No.	Entity	Attributes	Reason
1.	Student	StudentID, FirstName, LastName, DateOfBirth, Email	To uniquely identify each student, and communicate with them
2.	Faculty	FacultyID, FirstName, LastName, DepartmentID, Email, OfficeNumber	To uniquely identify faculty members, associate them with departments, and provide contact information
3.	Course	CourseID, CourseName, DepartmentID, CreditHours	To catalog courses, link them to departments, and define their weight
4.	Program	ProgramID, ProgramName, DepartmentID, DegreeLevel, TotalCreditHours	To define the various academic programs and their requirements
5.	Department	DepartmentID, DepartmentName, FacultyCount, Budget	To manage the administrative aspects of academic departments
6.	Lecture Hall	HallID, BuildingName,	To manage the physical spaces where courses are taught and their features

		Capacity, EquipmentDetails	
7.	Enrollment	EnrollmentID, StudentID, CourseID, Semester, Grade	To record what courses students are taking and their performance
8.	Academic Advisor	AdvisorID, FirstName, LastName, Email, DepartmentID	To provide information on individuals who guide students academically
9.	Research Group	GroupID, GroupName, FocusArea, LeadFacultyID, DepartmentID	To organize research efforts and link them to academic departments
10.	Research Project	ProjectID, Title, Summary, StartDate, EndDate	To manage individual research initiatives and track their timelines
11.	Library	LibraryID, Name, Location, OpenHours, ResourceCount	To manage the operations of university libraries
12.	Book	ISBN, Title, Author, LibraryID, CheckOutStatus	To catalog library books and track their availability
13.	Club	ClubID, Name, Purpose, FacultyAdvisorID	To manage student clubs and their activities
14.	Event	EventID, Title, Description, StartTime, EndTime	To schedule and promote university events
15.	Sports Team	TeamID, Sport, CaptainStudentID, TeamRanking	To manage sports teams and their competitive standings
16.	Scholarship	ScholarshipID, Name, EligibilityCriteria,	To manage financial aid offerings and their recipients

		Amount, ApplicationDeadline	
17.	Services	ServiceID, Name, Description, Availability, DepartmentID	To detail the services provided by the university to students and staff
18.	Job	JobID, Title, DepartmentID, Description, SalaryRange	To manage employment opportunities available within the university
19.	Alumni	AlumniID, FirstName, LastName, GraduationYear	To maintain a network of former students
20.	Dormitory	DormID, Name, Capacity, Amenities, RoomType	To manage student housing options and their features

Task 2: Initial Conceptual Model (ERD_Initial)

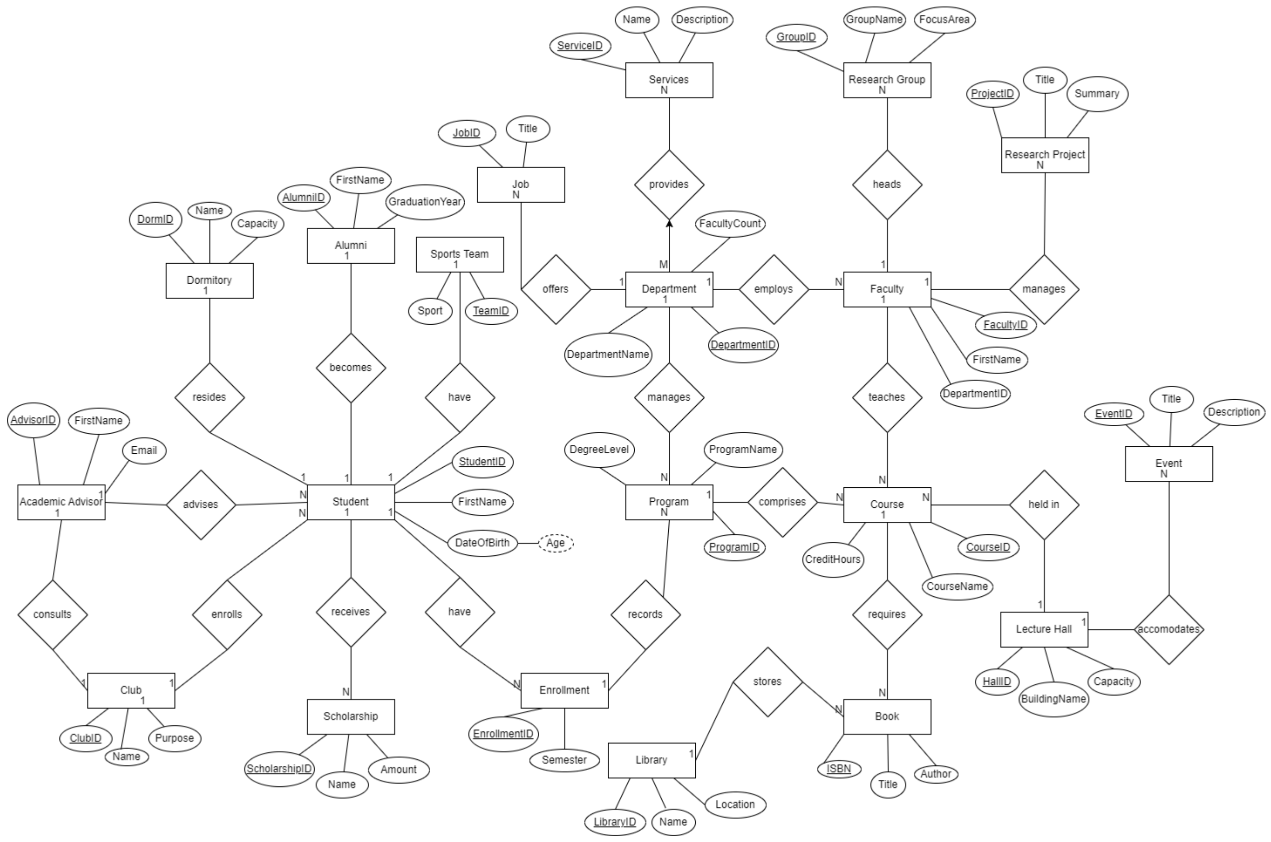


Figure 1 Initial ER Diagram for Halifax City University

Task 3: Design Issues Identification and Justification

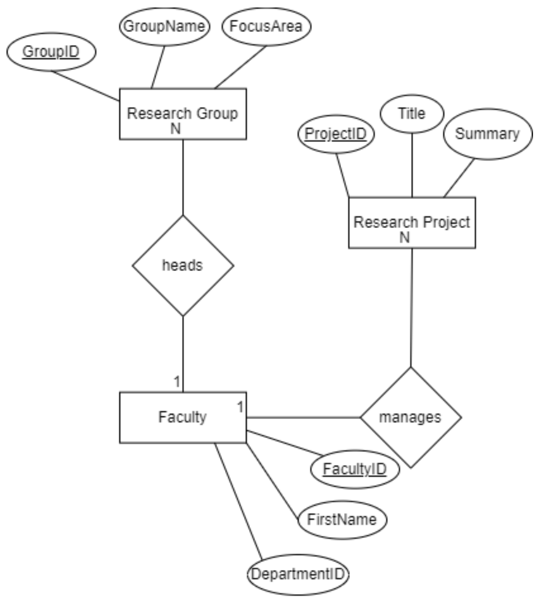


Figure 2 Chasm Trap between Research Group, Faculty and Research Project

The design issue present here is the chasm trap. Because of this, we can't determine which Research Group is working on which Research Project, and Research Groups and Research Projects can exist independently, which doesn't satisfy the requirements.

To solve this, we need to enforce a direct relationship between Research Group and Research Project.

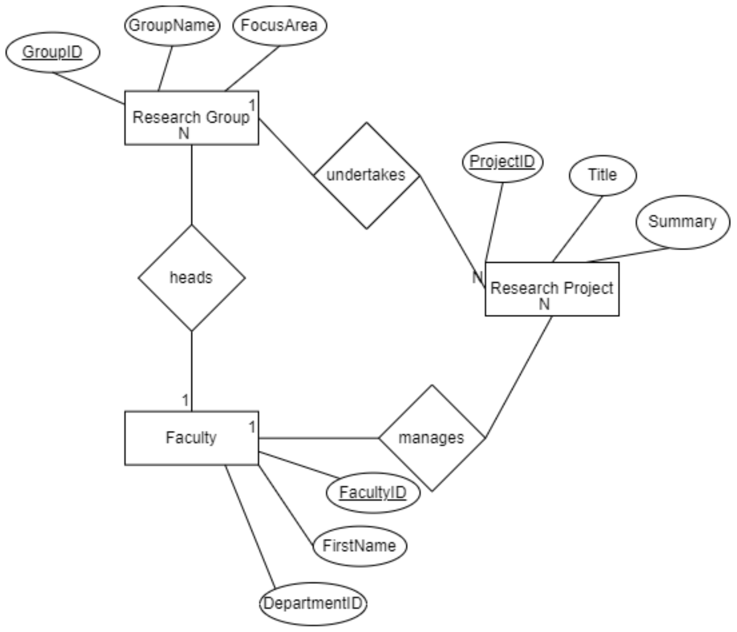


Figure 3 Solution for the Research Group, Faculty and Research Project chasm trap

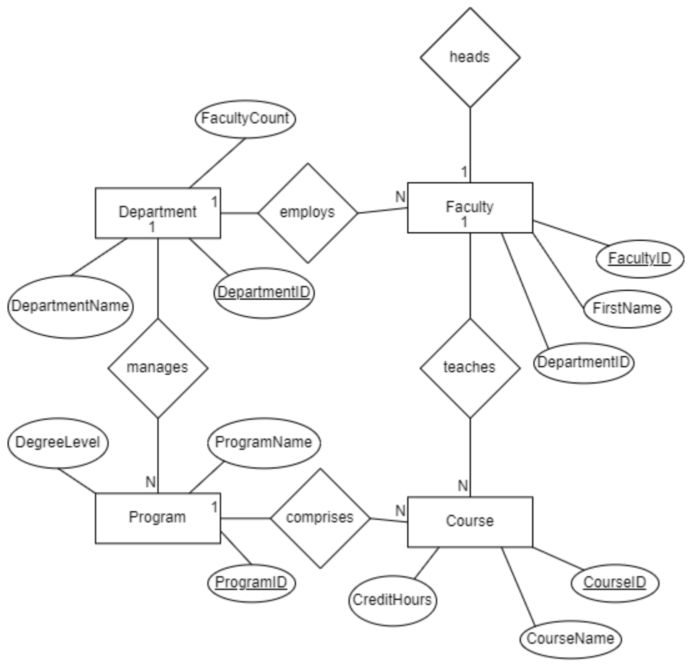


Figure 4 Chasm Trap between Department, Faculty and Course

Here, the chasm trap is present because not all faculty who head departments teach courses, and not all faculty who teach courses are part of the department.

To solve this design issue, we need to form relationships between Department and Course.

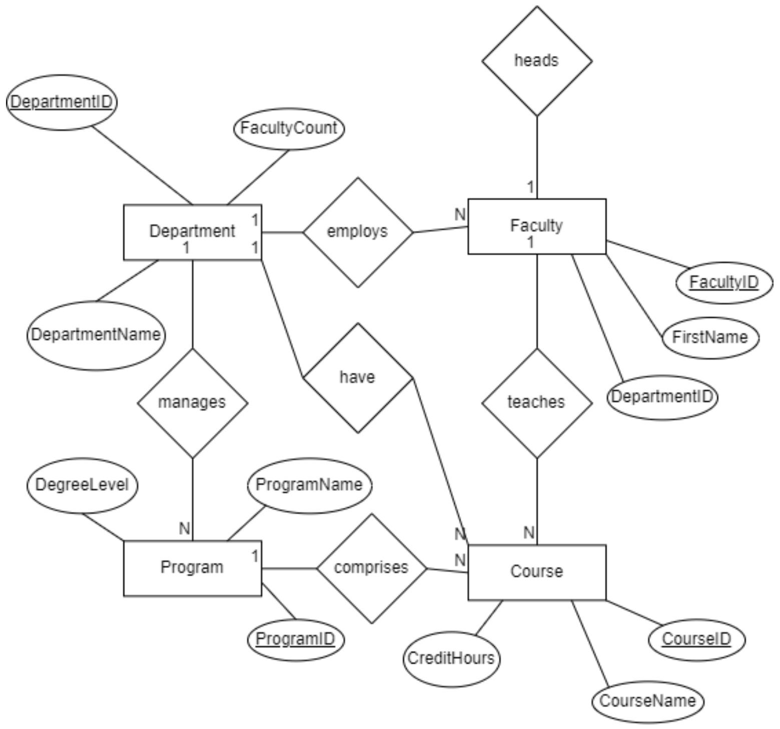


Figure 5 Solution for Department, Faculty and Course for chasm trap

Task 4: Final ERD Model (ERD_Final)

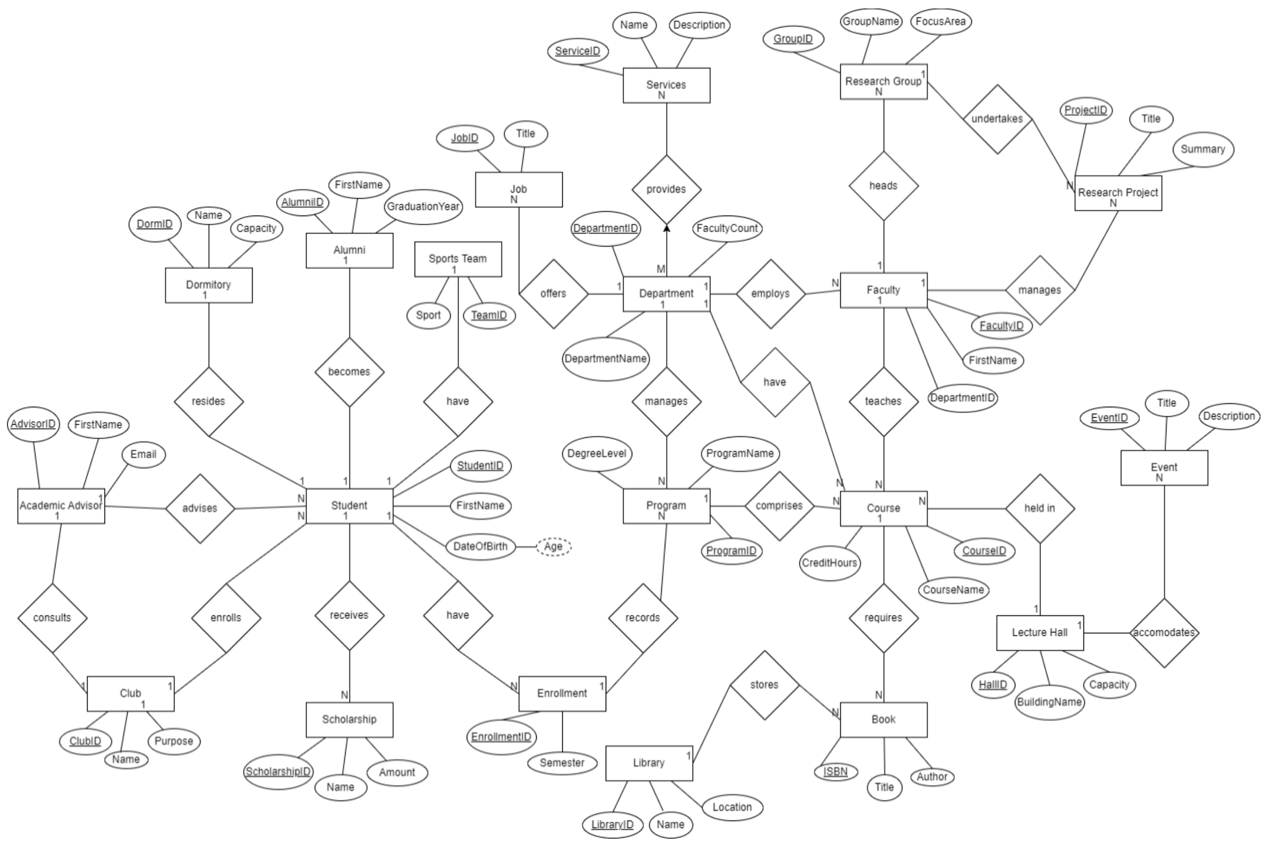


Figure 6 Final ER Diagram for Halifax City University

Task 5: Final ERRD Model (Final_ERRD)

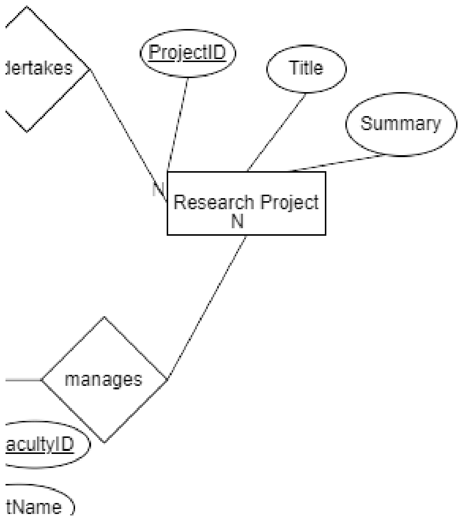


Figure 7 Research Project entity to be extended

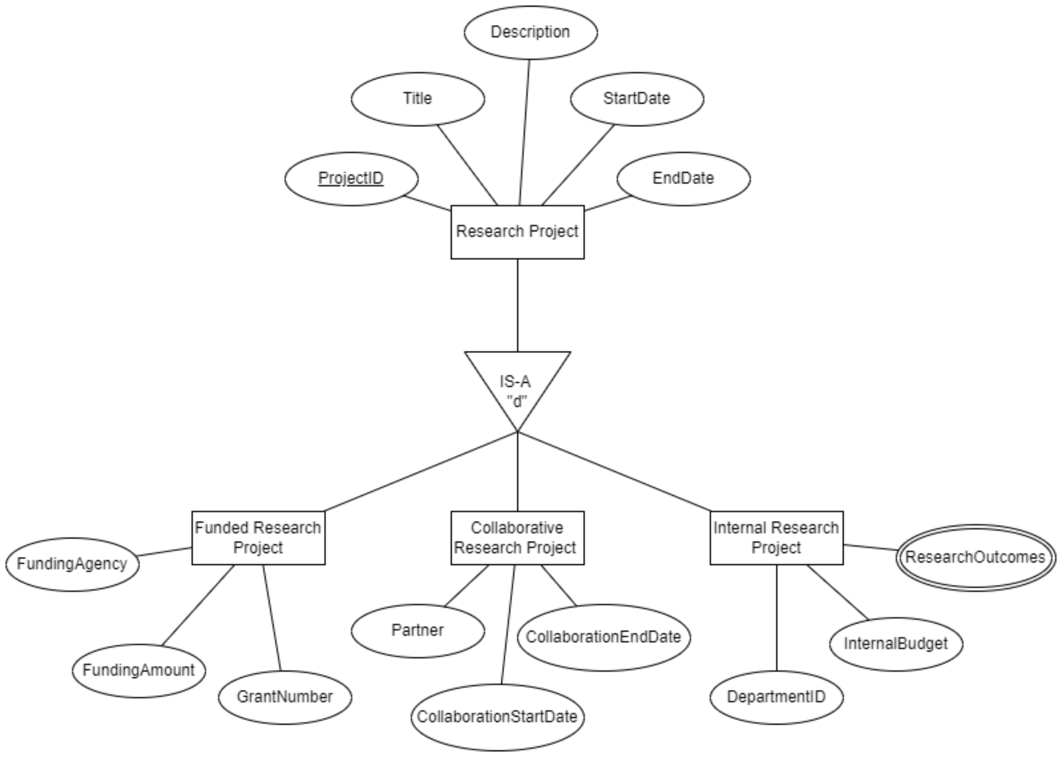


Figure 8 Enhanced ER Diagram for Research Project

Research Project (superclass) can be split into Funded Research Project, Collaborative Research Project, and Internal Research Project (subclasses). Each subclasses inherit the attributes from the Research Project, and they have their own attributes. They represent disjoint constraint because instance of the superclass can be a member of only one of the subclasses at a time.

Problem 2: Custom DBMS Using Java

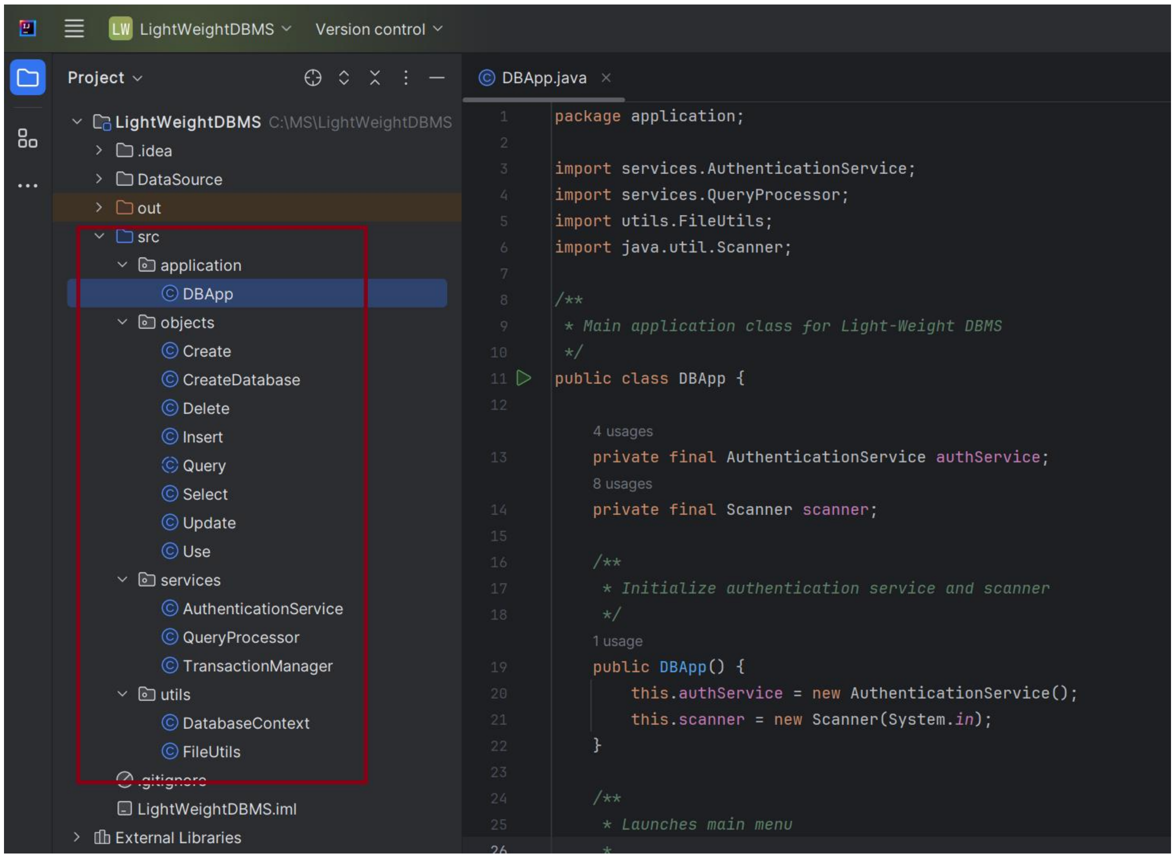
Task 1: Documentation of Design Principle

Application Architecture

The lightweight database management system is structured using a **layered architecture** style, with this the system is divided into a number of layers, each of which is responsible for a different task and interacts with the layers above and below it [1].

Application is structured as follows:

- **Presentation Layer (application)** – This package is the entry point of the application which handles user interaction and provides a command-line interface.
- **Business Logic Layer (services)** – This package encapsulates the core business logic and operations of the application.
- **Data Access Layer (objects)** – This package contains various query types which directly manipulates the database files.
- **Utility Layer (utils)** – This package provides supporting functionalities across the application.



```
LightWeightDBMS C:\MS\LightWeightDBMS
  .idea
  DataSource
  out
  src
    application
      DBApp
    objects
      Create
      CreateDatabase
      Delete
      Insert
      Query
      Select
      Update
      Use
    services
      AuthenticationService
      QueryProcessor
      TransactionManager
    utils
      DatabaseContext
      FileUtils
  .gitignore
  LightWeightDBMS.iml
External Libraries
```

```java
package application;

import services.AuthenticationService;
import services.QueryProcessor;
import utils.FileUtils;
import java.util.Scanner;

/**
 * Main application class for Light-Weight DBMS
 */
public class DBApp {

    4 usages
    private final AuthenticationService authService;
    8 usages
    private final Scanner scanner;

    /**
     * Initialize authentication service and scanner
     */
    1 usage
    public DBApp() {
        this.authService = new AuthenticationService();
        this.scanner = new Scanner(System.in);
    }

    /**
     * Launches main menu
```

Figure 9 Application Architecture for lightweight database management system.

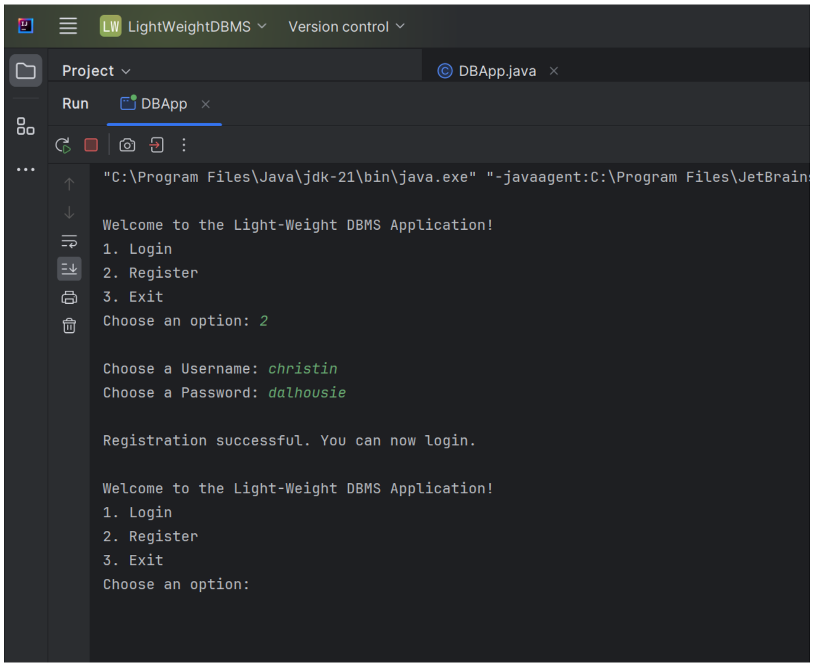
Following Single Responsibility Principle, each class is responsible for one job [2]. For example, ‘QueryProcessor’ is responsible for interpreting and directing SQL queries, ‘AuthenticationService’ handles user authentication, and ‘FileUtils’ manages file operations.

‘Query’ class is abstract, allowing for extension to handle various types of queries (‘Select’, ‘Insert’, ‘Update’, ‘Delete’) without modifying the base class this satisfies Open/Closed Principle. Also, from ‘Query’ class all classes inherit this ensures than any subclass can replace the ‘Query’ class without affecting the correctness of the program satisfying Liskov Substitution Principle.

Task 2: User Authentication

Register Process

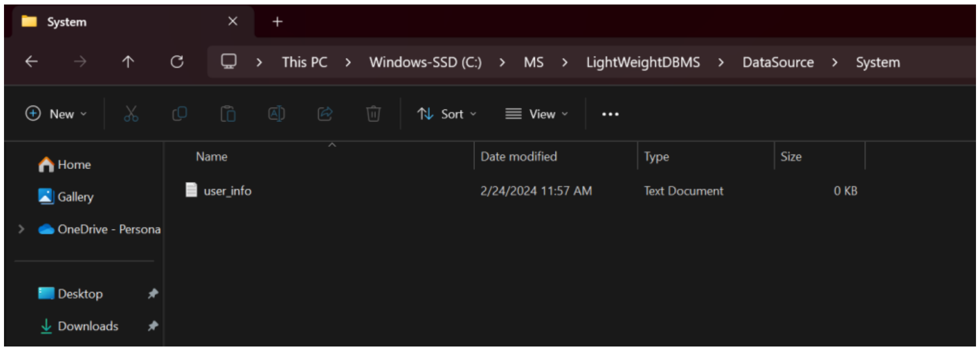
The menu of the lightweight database management system is displayed when the application is run. When user selects option 2 to register as shown in Figure 10, the user is able to successfully register with the system.



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrain

Welcome to the Light-Weight DBMS Application!
1. Login
2. Register
3. Exit
Choose an option: 2

Choose a Username: christin
Choose a Password: dalhousie

Registration successful. You can now login.

Welcome to the Light-Weight DBMS Application!
1. Login
2. Register
3. Exit
Choose an option:
```

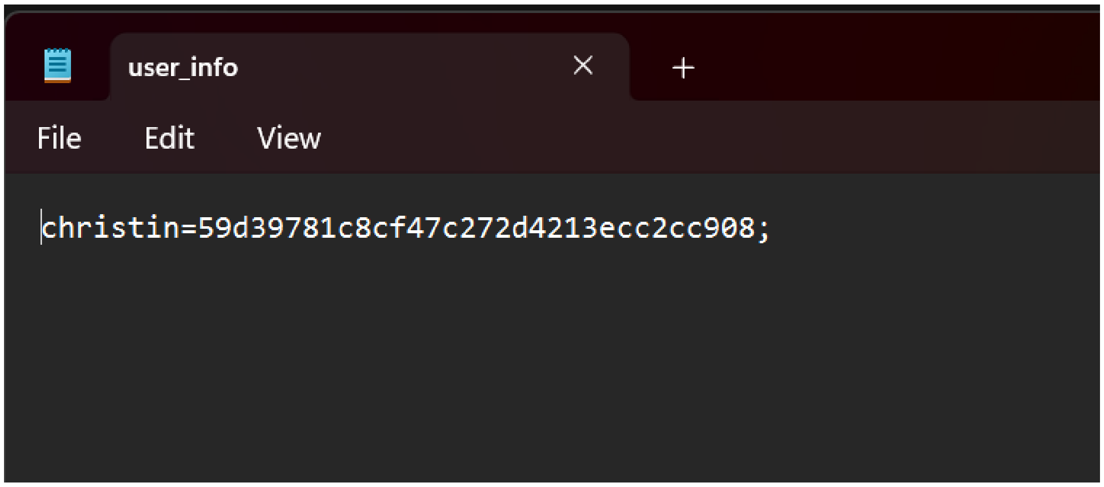
Figure 10 Successful registration process

User credentials are stored safely in a `user_info` text file, and the password is hashed using the MD5 algorithm.


System
This PC > Windows-SSD (C:) > MS > LightWeightDBMS > DataSource > System
New
Sort
View
Home
Gallery
OneDrive - Persona
Desktop
Downloads

Name	Date modified	Type	Size
user_info	2/24/2024 11:57 AM	Text Document	0 KB

Figure 11 'user_info' contains all the user credentials

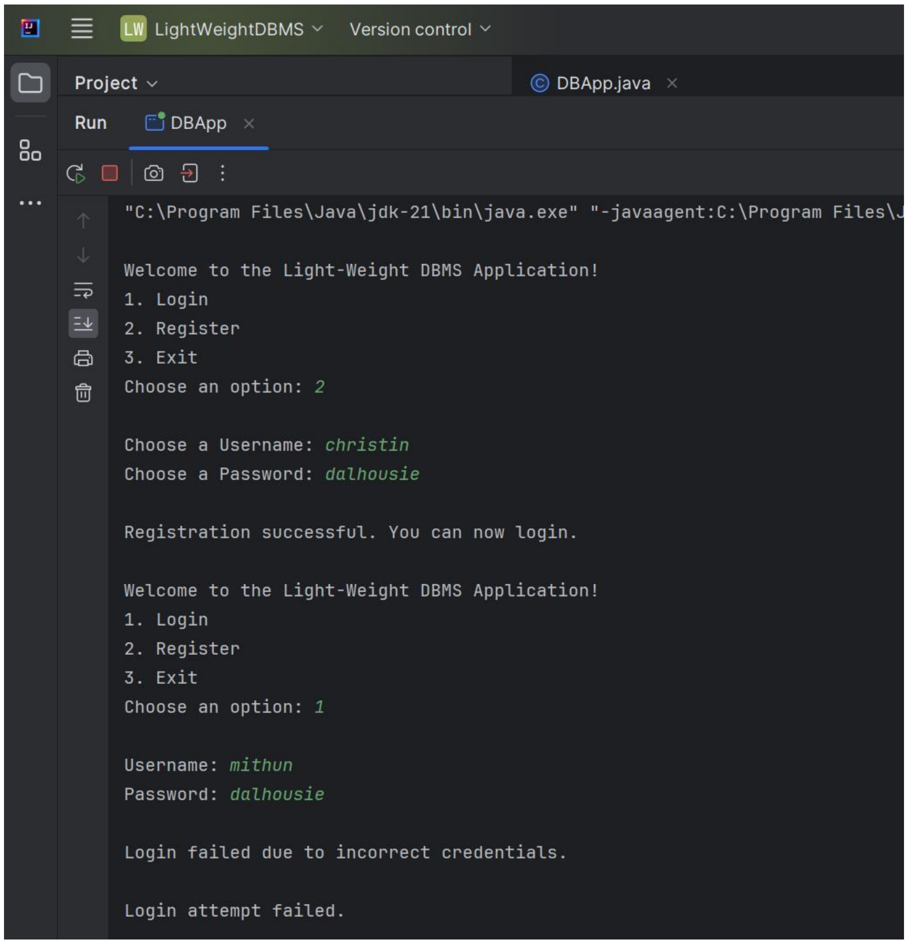


```
christin=59d39781c8cf47c272d4213ecc2cc908;
```

Figure 12 'user_info' contains username with MD5 hashed password

Login Process

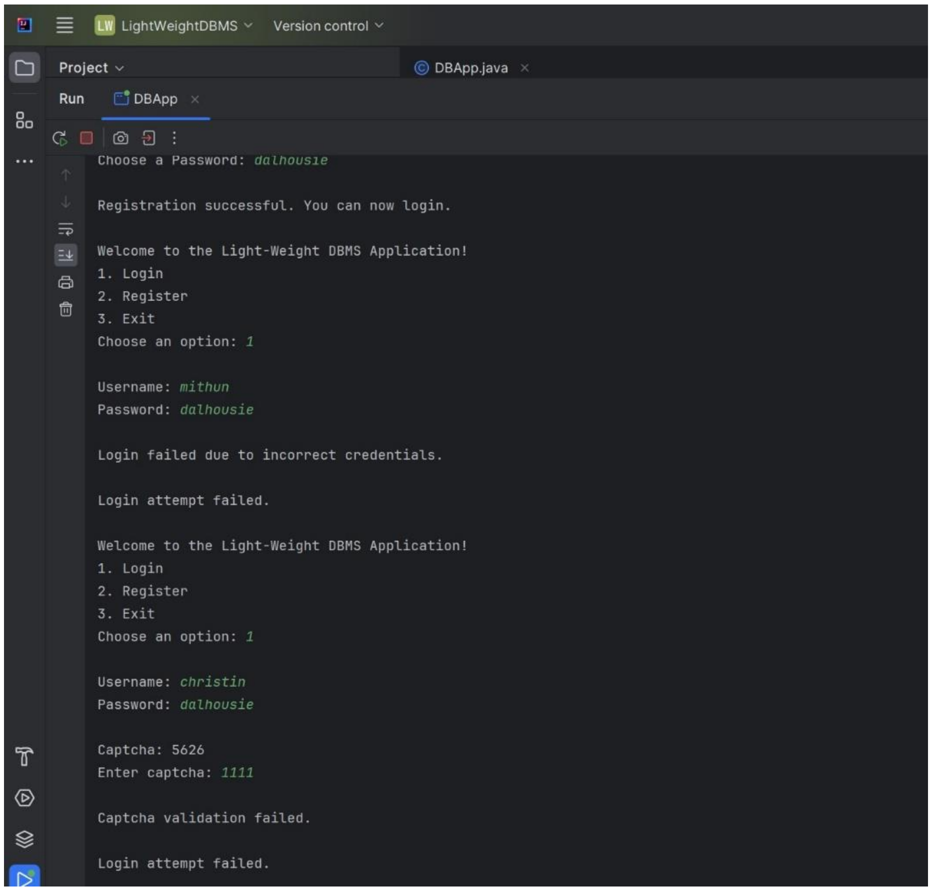
When a user enters incorrect credentials, they won't be able to log in to the system.



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\J

Welcome to the Light-Weight DBMS Application!
1. Login
2. Register
3. Exit
Choose an option: 2

Choose a Username: christin
Choose a Password: dalhousie

Registration successful. You can now login.

Welcome to the Light-Weight DBMS Application!
1. Login
2. Register
3. Exit
Choose an option: 1

Username: mithun
Password: dalhousie

Login failed due to incorrect credentials.

Login attempt failed.
```

Figure 13 Login process failed due to incorrect credentials

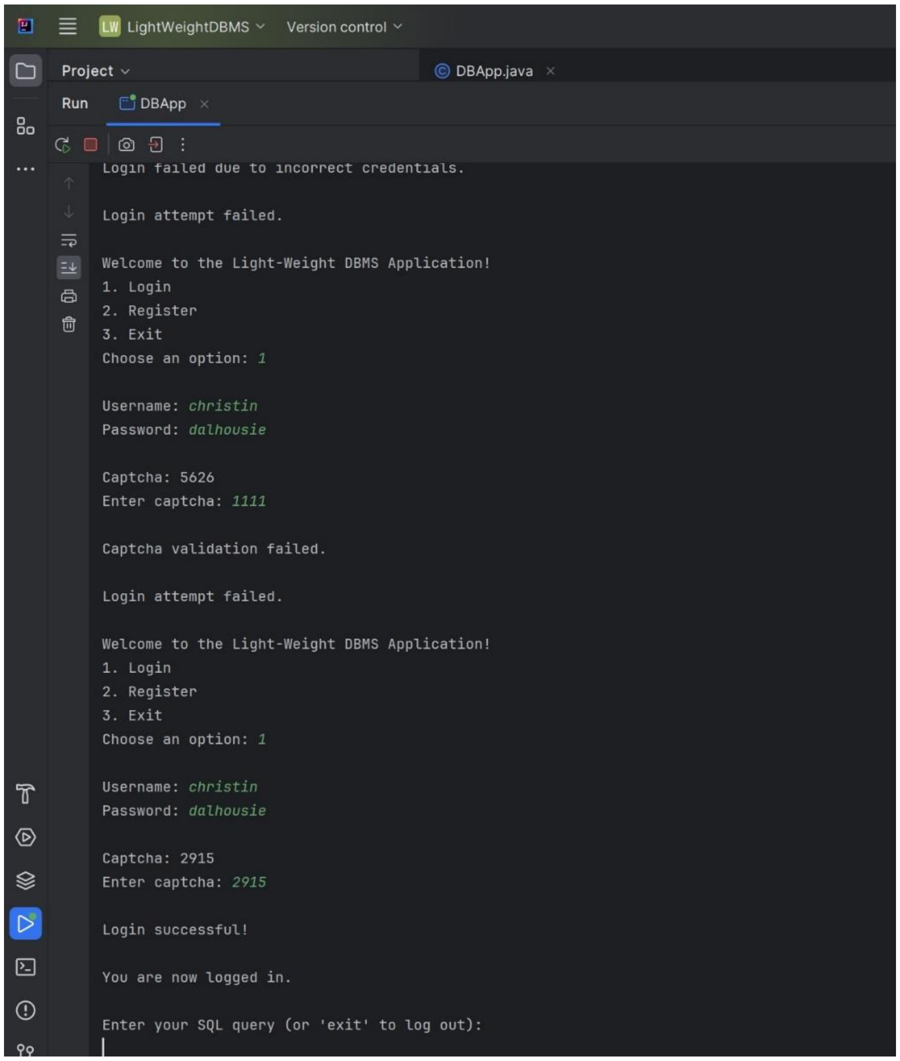
After entering the correct credentials to log in, the user will be prompted to enter a randomly generated 4-digit captcha. If they enter the incorrect captcha, they won't be allowed into the system.



```
Choose a Password: dalhousie

Registration successful. You can now login.

Welcome to the Light-Weight DBMS Application!
1. Login
2. Register
3. Exit
Choose an option: 1

Username: mithun
Password: dalhousie

Login failed due to incorrect credentials.

Login attempt failed.

Welcome to the Light-Weight DBMS Application!
1. Login
2. Register
3. Exit
Choose an option: 1

Username: christin
Password: dalhousie

Captcha: 5626
Enter captcha: 1111

Captcha validation failed.

Login attempt failed.
```

Figure 14 Incorrect captcha

After successfully entering the correct credentials and captcha, the user is allowed to perform SQL operations in the system.



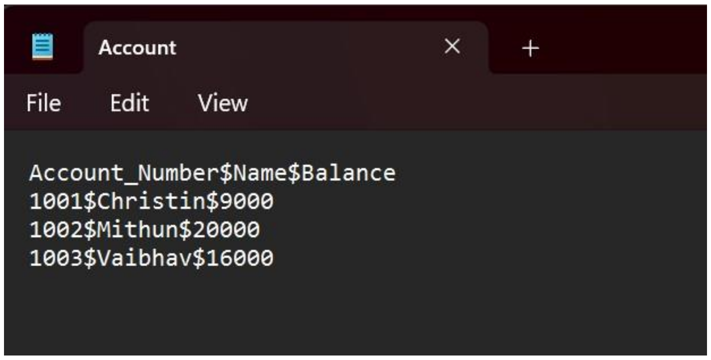
```
Login failed due to incorrect credentials.

Login attempt failed.

Welcome to the Light-Weight DBMS Application!
1. Login
2. Register
3. Exit
Choose an option: 1

Username: christin
Password: dalhousie

Captcha: 5626
Enter captcha: 1111

Captcha validation failed.

Login attempt failed.

Welcome to the Light-Weight DBMS Application!
1. Login
2. Register
3. Exit
Choose an option: 1

Username: christin
Password: dalhousie

Captcha: 2915
Enter captcha: 2915

Login successful!

You are now logged in.

Enter your SQL query (or 'exit' to log out):
```

Figure 15 Successful login after entering correct credentials with captcha

Task 3:

Task 3.1: Design of Persistent Storage

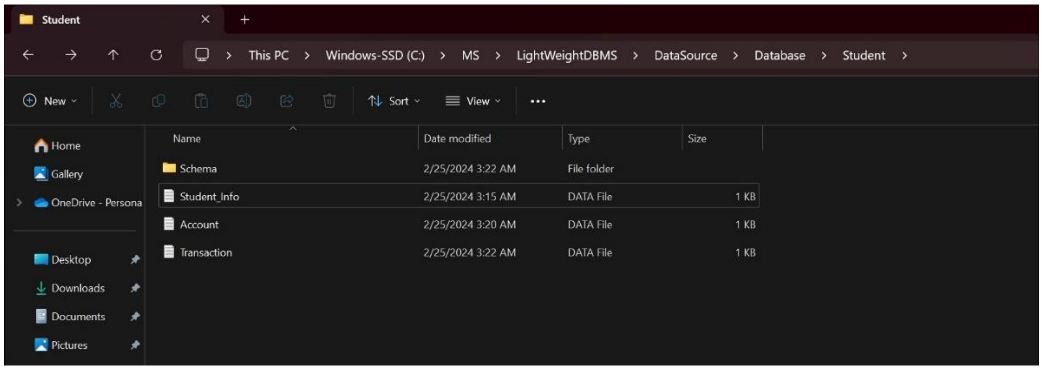
Novelty Task: Table data is stored using a custom delimiter separated by $.



```
Account_Number$Name$Balance
1001$Christin$9000
1002$Mithun$20000
1003$Vaibhav$16000
```

Figure 16 Table data stored using custom delimiter

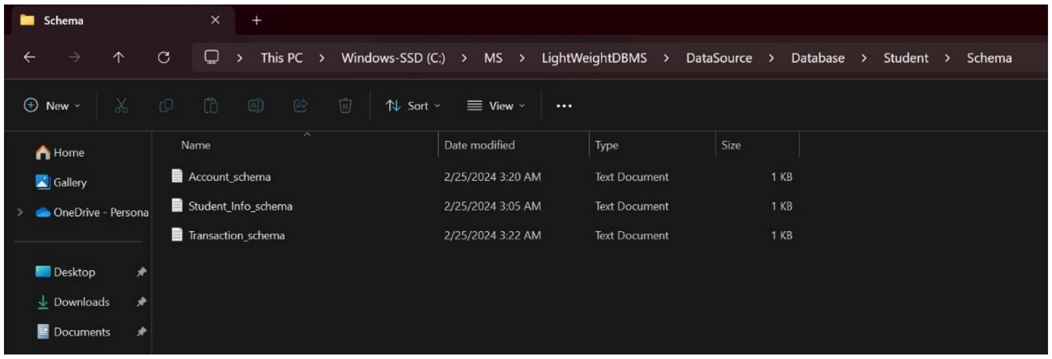
When each table is created, it creates a `(table_name).data` file extension, and its metadata is stored in the `Schema` folder as `(table_name)_schema.txt`.



Student
This PC > Windows-SSD (C:) > MS > LightWeightDBMS > DataSource > Database > Student >
New
Sort
View
Home
Gallery
OneDrive - Persona
Desktop
Downloads
Documents
Pictures

Name	Date modified	Type	Size
Schema	2/25/2024 3:22 AM	File folder	
Student_Info	2/25/2024 3:15 AM	DATA File	1 KB
Account	2/25/2024 3:20 AM	DATA File	1 KB
Transaction	2/25/2024 3:22 AM	DATA File	1 KB

Figure 17 Table data are stored in (table_name).data file extension



Schema
This PC > Windows-SSD (C:) > MS > LightWeightDBMS > DataSource > Database > Student > Schema
New
Sort
View
Home
Gallery
OneDrive - Persona
Desktop
Downloads
Documents

Name	Date modified	Type	Size
Account_schema	2/25/2024 3:20 AM	Text Document	1 KB
Student_Info_schema	2/25/2024 3:05 AM	Text Document	1 KB
Transaction_schema	2/25/2024 3:22 AM	Text Document	1 KB

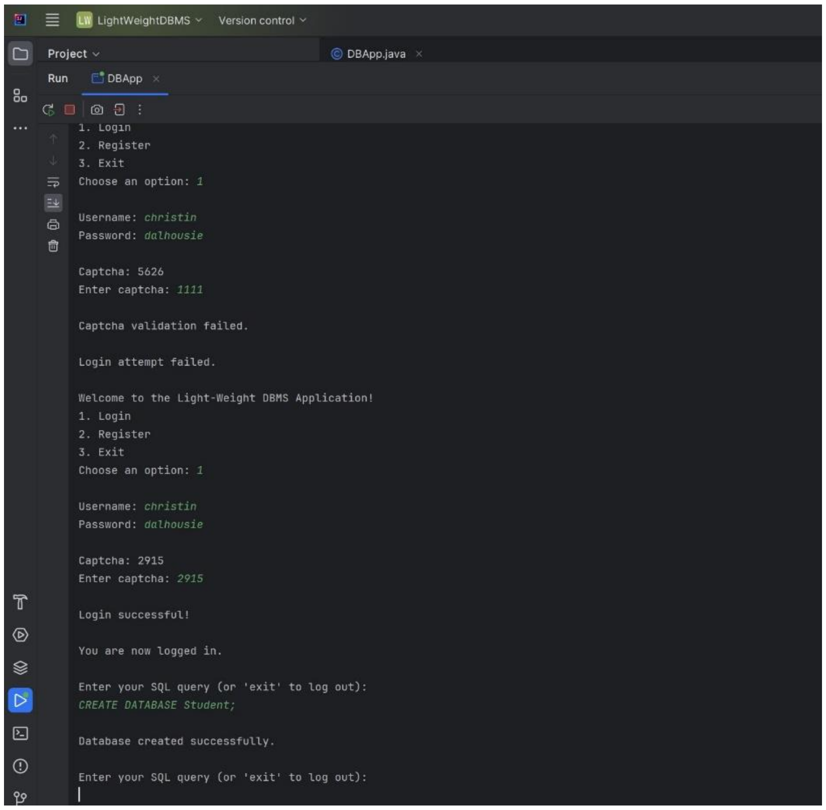
Figure 18 Each table has their own schema file

Task 3.2: Implementation of Queries (DDL & DML)

CREATE DATABASE

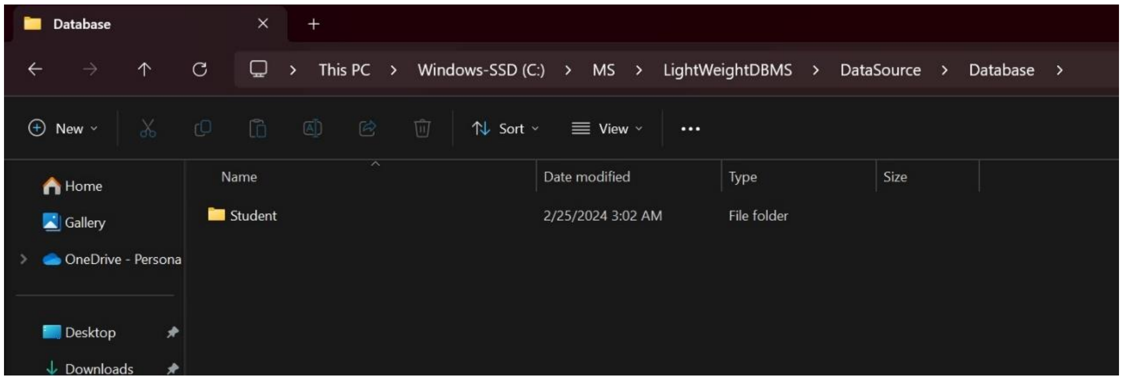
Query: CREATE DATABASE Student;

It creates a folder named after the `Database_Name` in the file system.



```
1. Login
2. Register
3. Exit
Choose an option: 1

Username: christin
Password: dalhousie

Captcha: 5626
Enter captcha: 1111

Captcha validation failed.

Login attempt failed.

Welcome to the Light-Weight DBMS Application!
1. Login
2. Register
3. Exit
Choose an option: 1

Username: christin
Password: dalhousie

Captcha: 2915
Enter captcha: 2915

Login successful!

You are now logged in.

Enter your SQL query (or 'exit' to log out):
CREATE DATABASE Student;

Database created successfully.

Enter your SQL query (or 'exit' to log out):
```

Figure 19 CREATE DATABASE operation



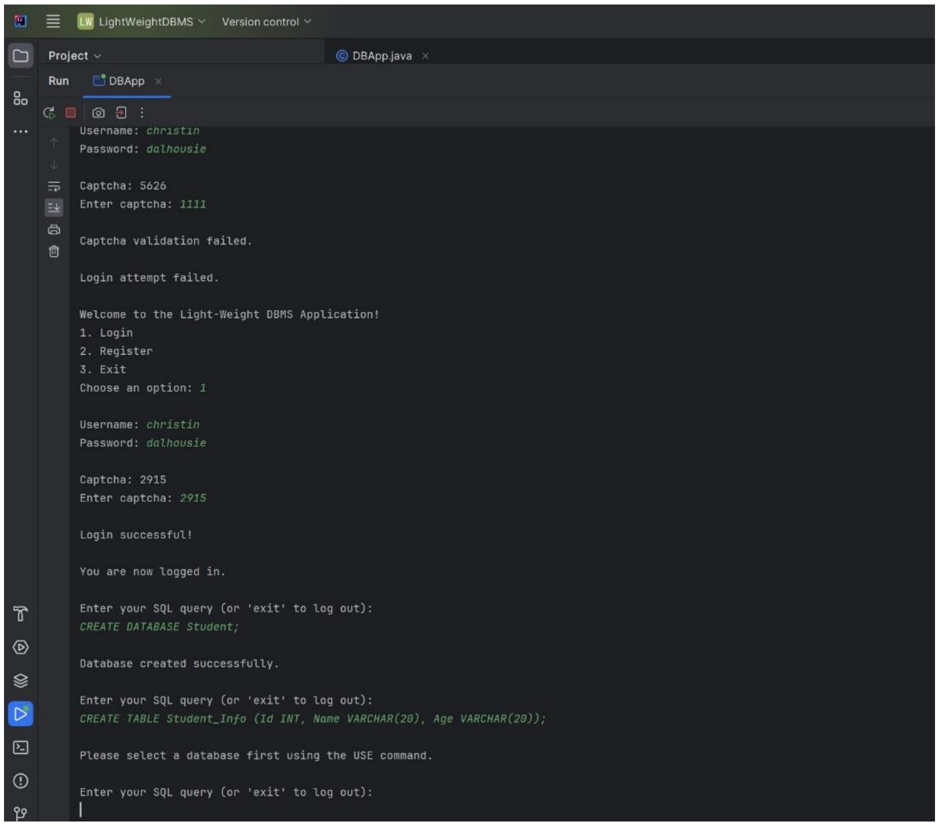
Database
This PC > Windows-SSD (C:) > MS > LightWeightDBMS > DataSource > Database >
New
Sort
View
Home
Gallery
OneDrive - Persona
Desktop
Downloads

Name	Date modified	Type	Size
Student	2/25/2024 3:02 AM	File folder	

Figure 20 Student folder is generated for the database

USE

If a user tries to access using a DML command, they will be prompted to first select the database.

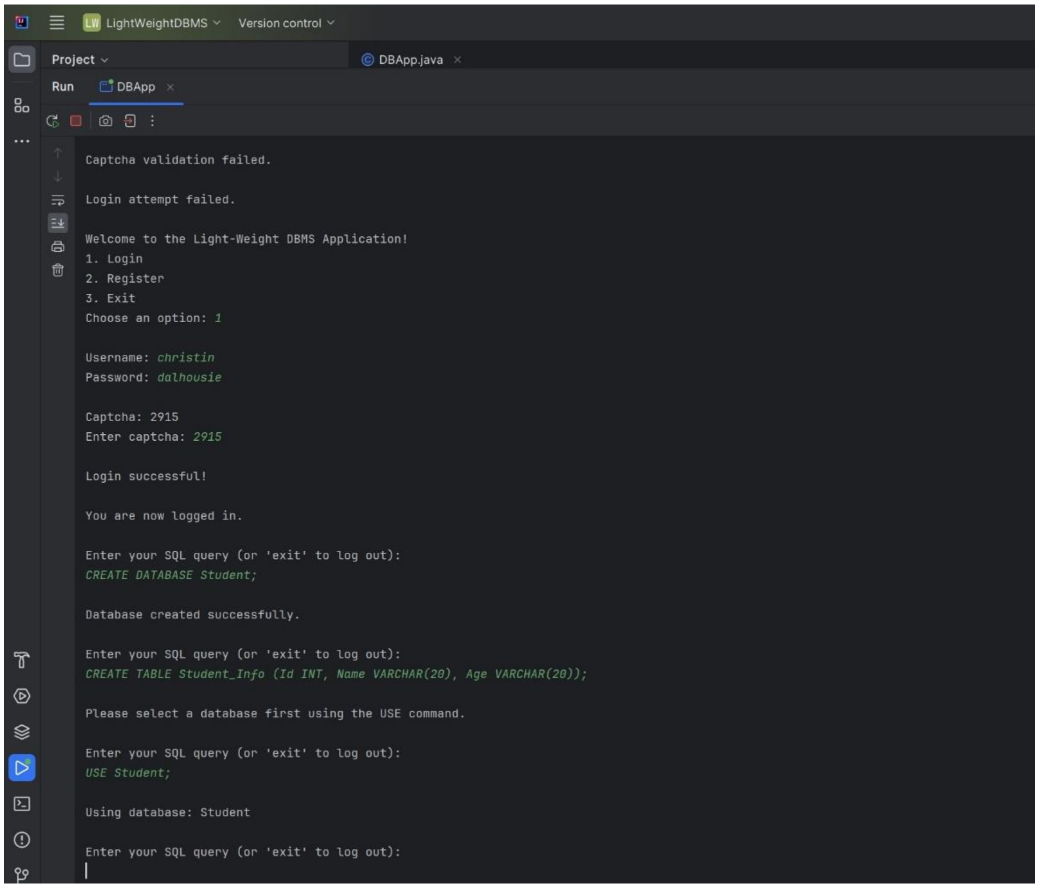


```
Username: christin
Password: dalhousie

Captcha: 5626
Enter captcha: 1111

Captcha validation failed.

Login attempt failed.

Welcome to the Light-Weight DBMS Application!
1. Login
2. Register
3. Exit
Choose an option: 1

Username: christin
Password: dalhousie

Captcha: 2915
Enter captcha: 2915

Login successful!

You are now logged in.

Enter your SQL query (or 'exit' to log out):
CREATE DATABASE Student;

Database created successfully.

Enter your SQL query (or 'exit' to log out):
CREATE TABLE Student_Info (Id INT, Name VARCHAR(20), Age VARCHAR(20));

Please select a database first using the USE command.

Enter your SQL query (or 'exit' to log out):
```

Figure 21 Using DML commands without selecting database

Query: USE DATABASE Student

Selecting the database makes it the active database for the DML operations.



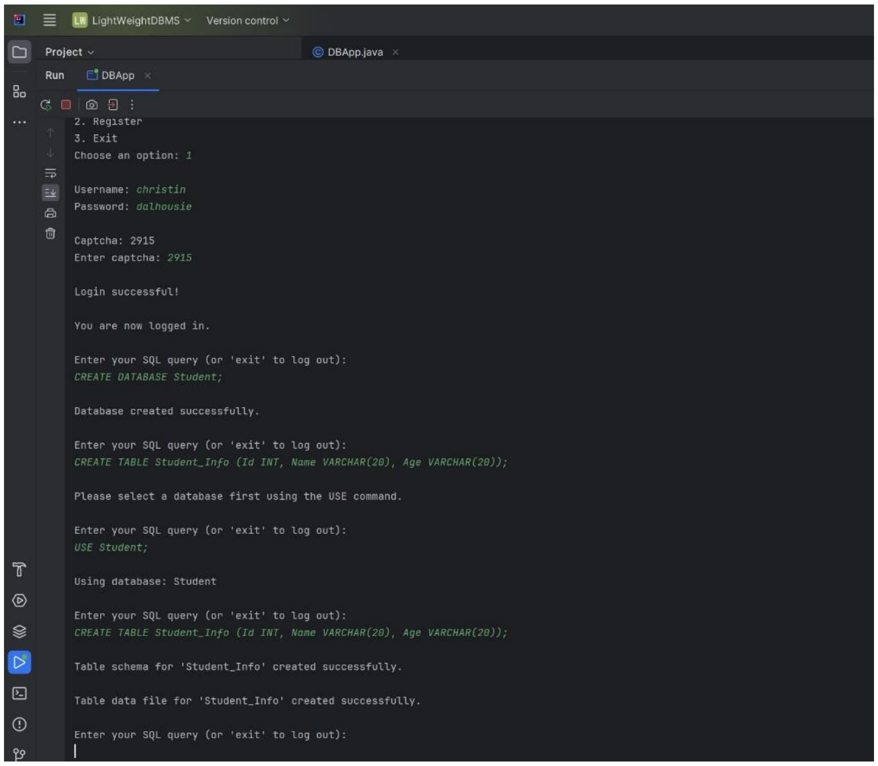
```
Captcha validation failed.

Login attempt failed.

Welcome to the Light-Weight DBMS Application!
1. Login
2. Register
3. Exit
Choose an option: 1

Username: christin
Password: dalhousie

Captcha: 2915
Enter captcha: 2915

Login successful!

You are now logged in.

Enter your SQL query (or 'exit' to log out):
CREATE DATABASE Student;

Database created successfully.

Enter your SQL query (or 'exit' to log out):
CREATE TABLE Student_Info (Id INT, Name VARCHAR(20), Age VARCHAR(20));

Please select a database first using the USE command.

Enter your SQL query (or 'exit' to log out):
USE Student;

Using database: Student

Enter your SQL query (or 'exit' to log out):
```

Figure 22 USE Student to select Student database

CREATE TABLE

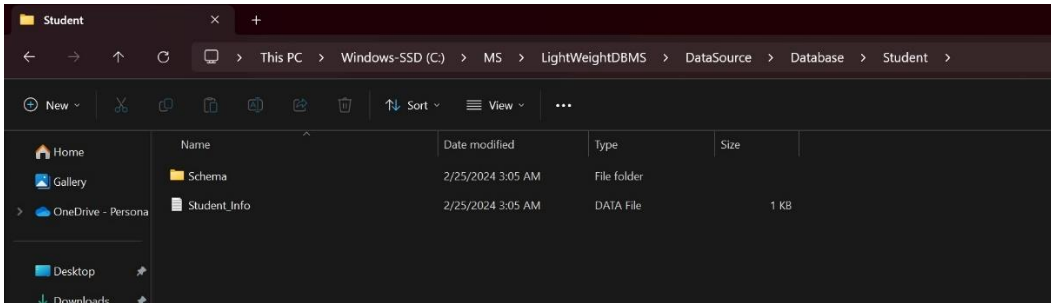
Query: CREATE TABLE Student_Info (Id INT, Name VARCHAR(20), Age VARCHAR(20));

This creates a `Student_info.data` file to store the table data inside the `Student` folder, and the metadata of the table is stored inside the `Schema` folder.



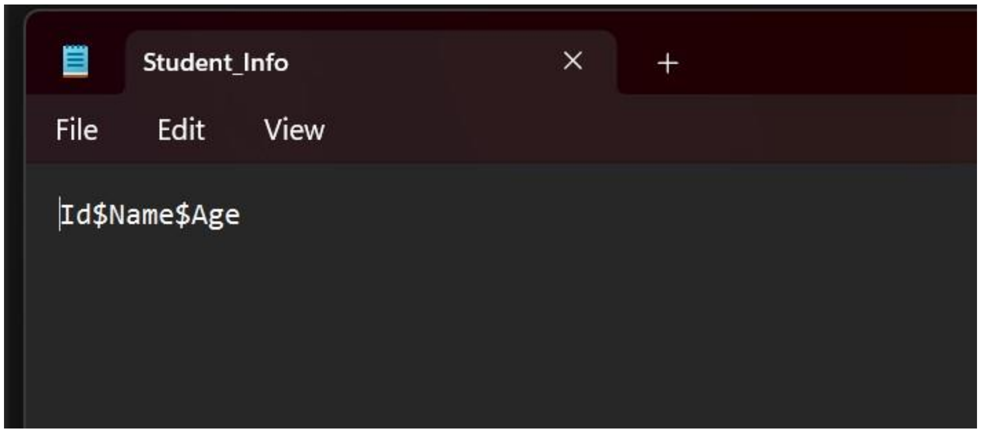
```
2. Register
3. Exit
Choose an option: 1

Username: christin
Password: dalhousie

Captcha: 2915
Enter captcha: 2915

Login successful!

You are now logged in.

Enter your SQL query (or 'exit' to log out):
CREATE DATABASE Student;

Database created successfully.

Enter your SQL query (or 'exit' to log out):
CREATE TABLE Student_Info (Id INT, Name VARCHAR(20), Age VARCHAR(20));

Please select a database first using the USE command.

Enter your SQL query (or 'exit' to log out):
USE Student;

Using database: Student

Enter your SQL query (or 'exit' to log out):
CREATE TABLE Student_Info (Id INT, Name VARCHAR(20), Age VARCHAR(20));

Table schema for 'Student_Info' created successfully.

Table data file for 'Student_Info' created successfully.

Enter your SQL query (or 'exit' to log out):
```

Figure 23 CREATE TABLE operation


Student
This PC > Windows-SSD (C:) > MS > LightWeightDBMS > DataSource > Database > Student >
New
Sort
View
Home
Gallery
OneDrive - Persona
Desktop
Downloads

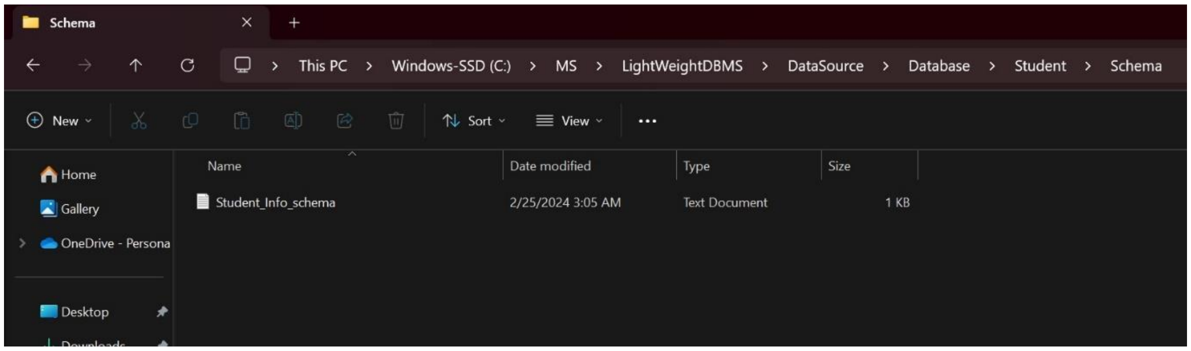
Name	Date modified	Type	Size
Schema	2/25/2024 3:05 AM	File folder	
Student_Info	2/25/2024 3:05 AM	DATA File	1 KB

Figure 24 Creates Student_Info.data to store table data



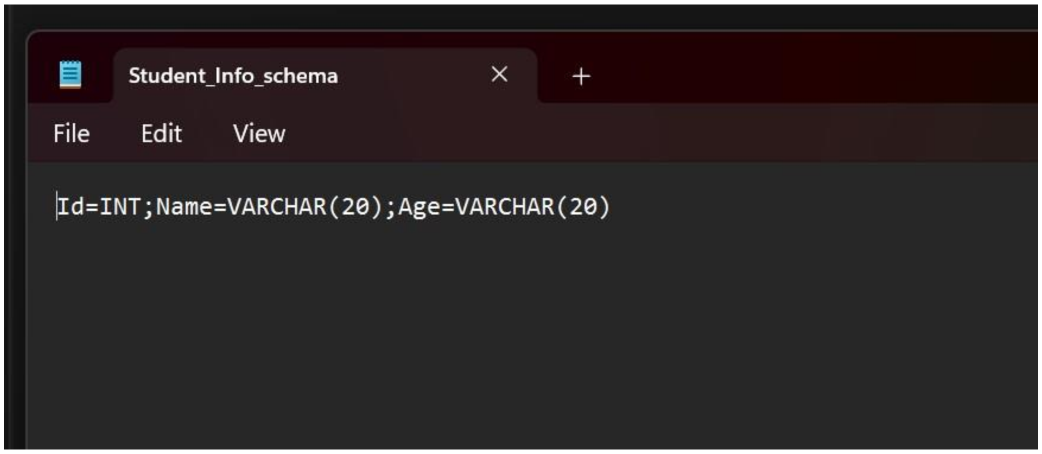
```
Id$Name$Age
```

Figure 25 Student_info with header


Schema
This PC > Windows-SSD (C:) > MS > LightWeightDBMS > DataSource > Database > Student > Schema
New
Sort
View
Home
Gallery
OneDrive - Persona
Desktop

Name	Date modified	Type	Size
Student_Info_schema	2/25/2024 3:05 AM	Text Document	1 KB

Figure 26 Student_Info_schema to store meta-data



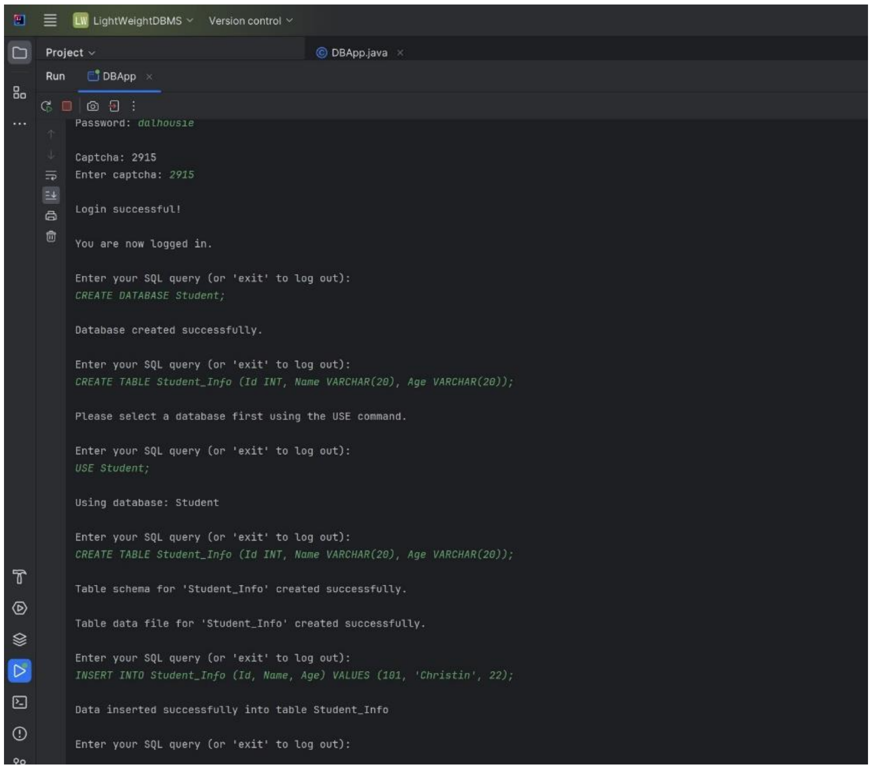
```
Id=INT;Name=VARCHAR(20);Age=VARCHAR(20)
```

Figure 27 Meta-data of Student_Info table

INSERT

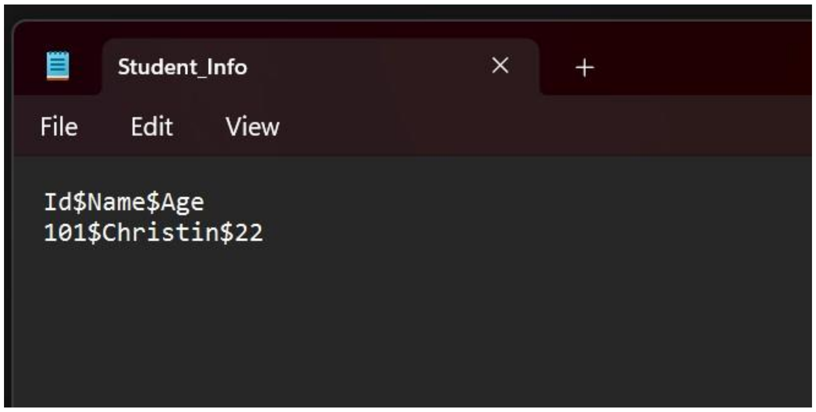
Query: INSERT INTO Student_Info (Id, Name, Age) VALUES (101, ‘Christin’, 22);

The insert operation puts the data inside the `Student_Info.data`, with each value separated by the custom delimiter $.



```
Password: dalhousie

Captcha: 2915
Enter captcha: 2915

Login successful!

You are now logged in.

Enter your SQL query (or 'exit' to log out):
CREATE DATABASE Student;

Database created successfully.

Enter your SQL query (or 'exit' to log out):
CREATE TABLE Student_Info (Id INT, Name VARCHAR(20), Age VARCHAR(20));

Please select a database first using the USE command.

Enter your SQL query (or 'exit' to log out):
USE Student;

Using database: Student

Enter your SQL query (or 'exit' to log out):
CREATE TABLE Student_Info (Id INT, Name VARCHAR(20), Age VARCHAR(20));

Table schema for 'Student_Info' created successfully.

Table data file for 'Student_Info' created successfully.

Enter your SQL query (or 'exit' to log out):
INSERT INTO Student_Info (Id, Name, Age) VALUES (101, 'Christin', 22);

Data inserted successfully into table Student_Info

Enter your SQL query (or 'exit' to log out):
```

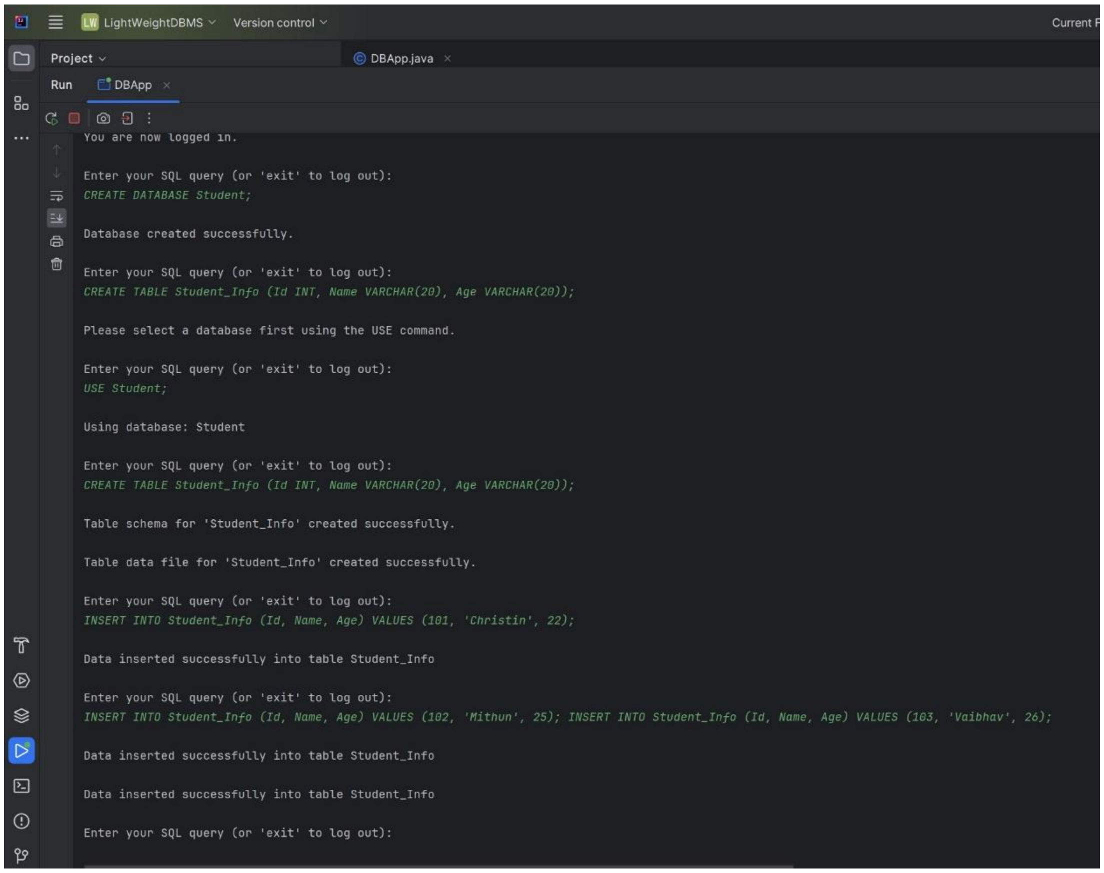
Figure 28 INSERT operation to the Student_Info table



```
Id$Name$Age
101$Christin$22
```

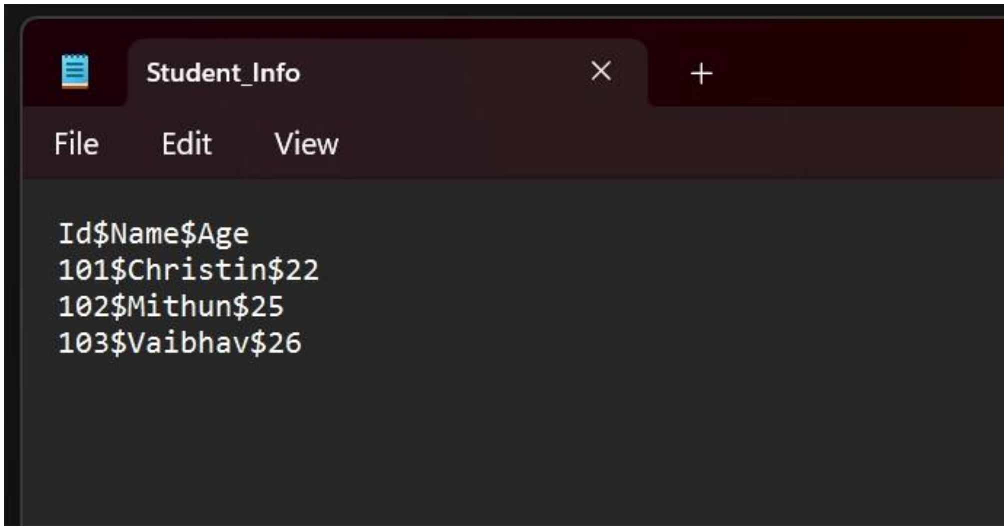
Figure 29 Data stored in Student_Info using custom delimiter

The insert operation also supports multiple insert statements to be entered together, and the values are stored in the data file.



```
You are now logged in.

Enter your SQL query (or 'exit' to log out):
CREATE DATABASE Student;

Database created successfully.

Enter your SQL query (or 'exit' to log out):
CREATE TABLE Student_Info (Id INT, Name VARCHAR(20), Age VARCHAR(20));

Please select a database first using the USE command.

Enter your SQL query (or 'exit' to log out):
USE Student;

Using database: Student

Enter your SQL query (or 'exit' to log out):
CREATE TABLE Student_Info (Id INT, Name VARCHAR(20), Age VARCHAR(20));

Table schema for 'Student_Info' created successfully.

Table data file for 'Student_Info' created successfully.

Enter your SQL query (or 'exit' to log out):
INSERT INTO Student_Info (Id, Name, Age) VALUES (101, 'Christin', 22);

Data inserted successfully into table Student_Info

Enter your SQL query (or 'exit' to log out):
INSERT INTO Student_Info (Id, Name, Age) VALUES (102, 'Mithun', 25); INSERT INTO Student_Info (Id, Name, Age) VALUES (103, 'Vaibhav', 26);

Data inserted successfully into table Student_Info

Data inserted successfully into table Student_Info

Enter your SQL query (or 'exit' to log out):
```

Figure 30 Inserting multiple data to Student_Info table



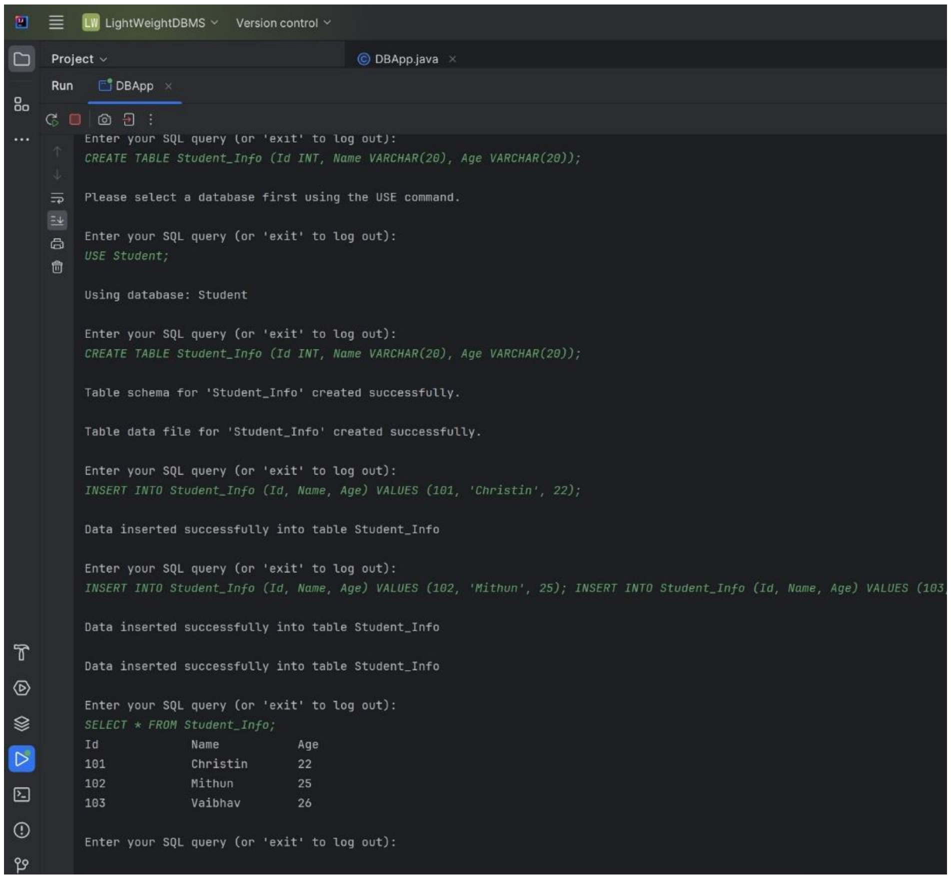
```
Id$Name$Age
101$Christin$22
102$Mithun$25
103$Vaibhav$26
```

Figure 31 Multiple rows of data in Student_Info table seperated by custom delimiter

SELECT

Query: SELECT * FROM Student_Info;

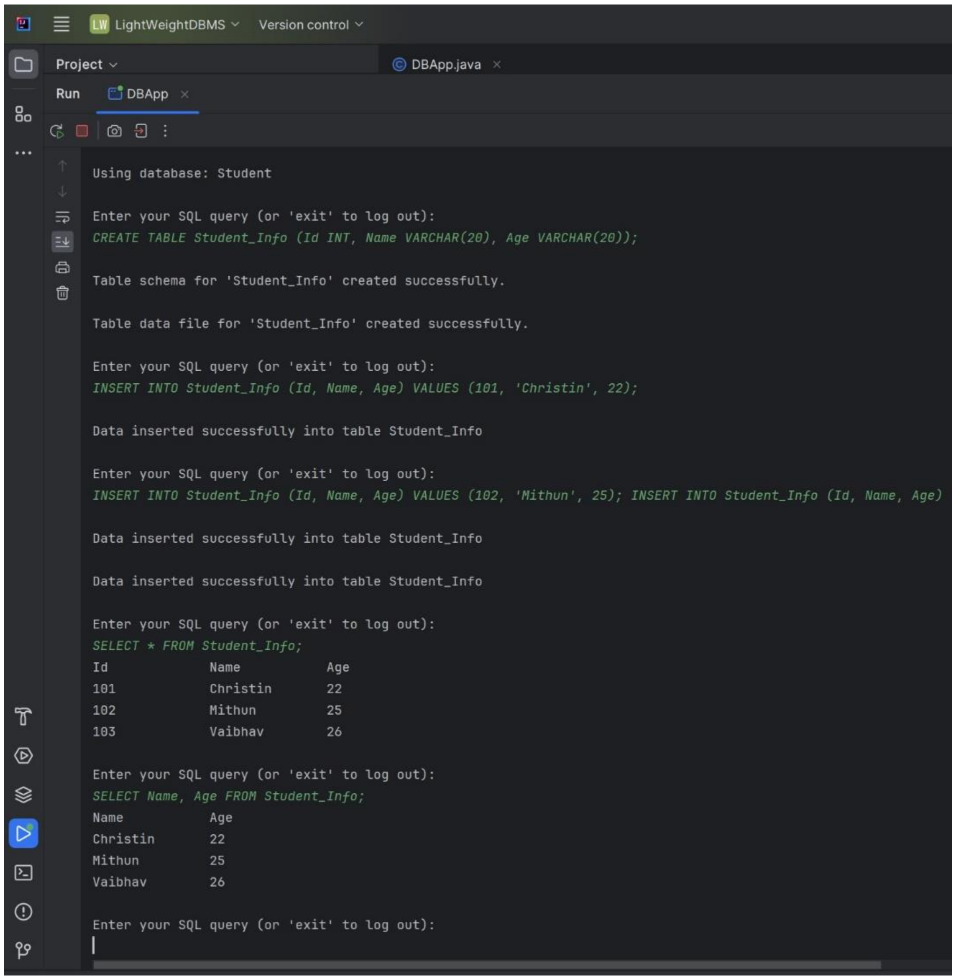
It selects all the columns and displays them nicely spaced.



```
Enter your SQL query (or 'exit' to log out):
CREATE TABLE Student_Info (Id INT, Name VARCHAR(20), Age VARCHAR(20));

Please select a database first using the USE command.

Enter your SQL query (or 'exit' to log out):
USE Student;

Using database: Student

Enter your SQL query (or 'exit' to log out):
CREATE TABLE Student_Info (Id INT, Name VARCHAR(20), Age VARCHAR(20));

Table schema for 'Student_Info' created successfully.

Table data file for 'Student_Info' created successfully.

Enter your SQL query (or 'exit' to log out):
INSERT INTO Student_Info (Id, Name, Age) VALUES (101, 'Christin', 22);

Data inserted successfully into table Student_Info

Enter your SQL query (or 'exit' to log out):
INSERT INTO Student_Info (Id, Name, Age) VALUES (102, 'Mithun', 25); INSERT INTO Student_Info (Id, Name, Age) VALUES (103

Data inserted successfully into table Student_Info

Data inserted successfully into table Student_Info

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Student_Info;
Id             Name           Age
101            Christin       22
102            Mithun         25
103            Vaibhav        26

Enter your SQL query (or 'exit' to log out):
```

*Figure 32 SELECT * to select all the columns and display it*

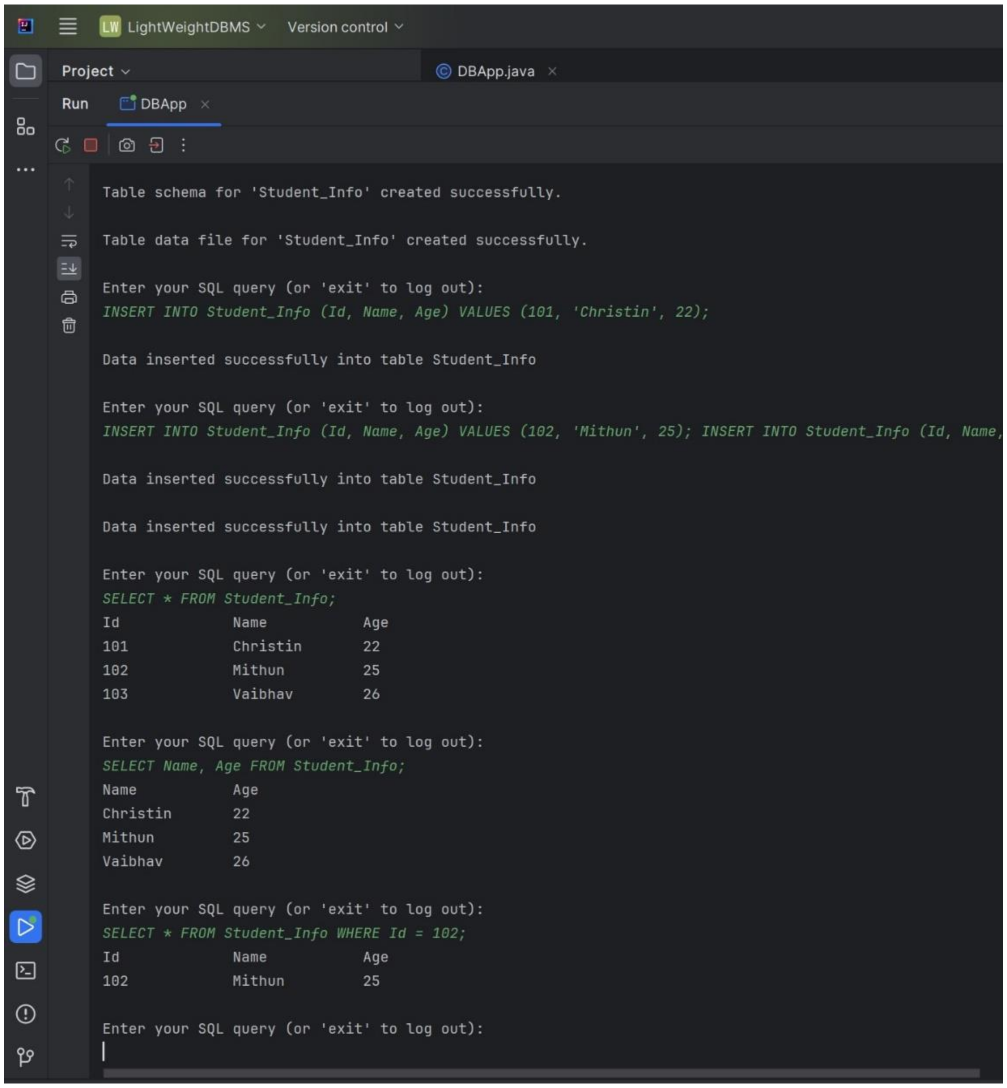
We can select particular columns, and it will display that information.



```
Using database: Student

Enter your SQL query (or 'exit' to log out):
CREATE TABLE Student_Info (Id INT, Name VARCHAR(20), Age VARCHAR(20));

Table schema for 'Student_Info' created successfully.

Table data file for 'Student_Info' created successfully.

Enter your SQL query (or 'exit' to log out):
INSERT INTO Student_Info (Id, Name, Age) VALUES (101, 'Christin', 22);

Data inserted successfully into table Student_Info

Enter your SQL query (or 'exit' to log out):
INSERT INTO Student_Info (Id, Name, Age) VALUES (102, 'Mithun', 25); INSERT INTO Student_Info (Id, Name, Age)

Data inserted successfully into table Student_Info

Data inserted successfully into table Student_Info

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Student_Info;
Id             Name           Age
101            Christin       22
102            Mithun         25
103            Vaibhav        26

Enter your SQL query (or 'exit' to log out):
SELECT Name, Age FROM Student_Info;
Name           Age
Christin       22
Mithun         25
Vaibhav        26

Enter your SQL query (or 'exit' to log out):
```

Figure 33 Selecting particular columns

Using the WHERE clause, we can display the particular information we need.



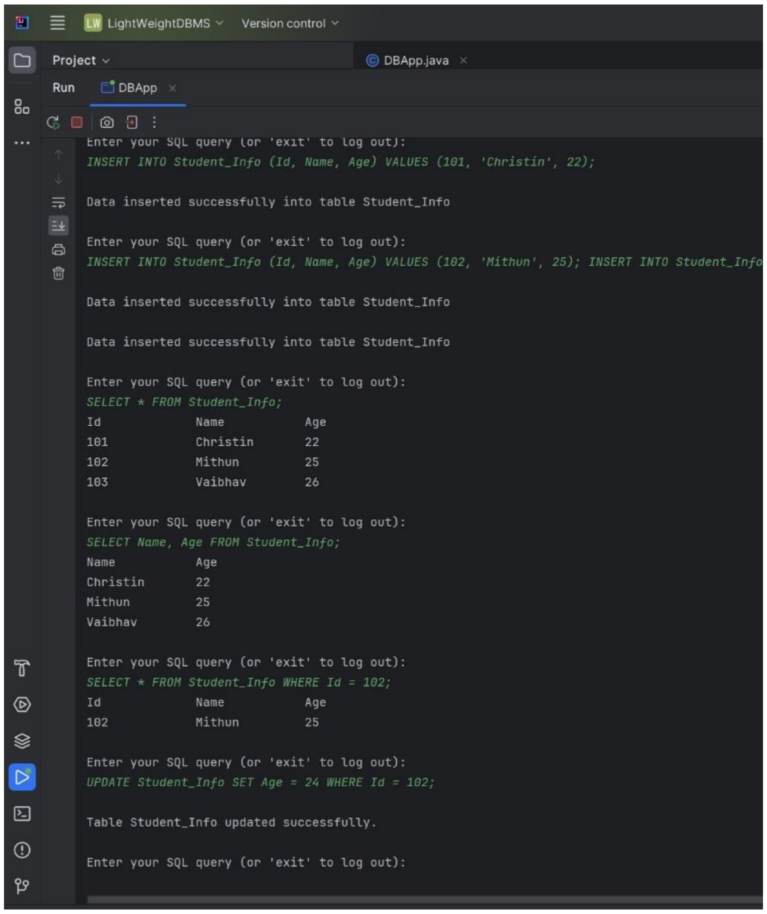
```
Table schema for 'Student_Info' created successfully.

Table data file for 'Student_Info' created successfully.

Enter your SQL query (or 'exit' to log out):
INSERT INTO Student_Info (Id, Name, Age) VALUES (101, 'Christin', 22);

Data inserted successfully into table Student_Info

Enter your SQL query (or 'exit' to log out):
INSERT INTO Student_Info (Id, Name, Age) VALUES (102, 'Mithun', 25); INSERT INTO Student_Info (Id, Name,

Data inserted successfully into table Student_Info

Data inserted successfully into table Student_Info

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Student_Info;
Id             Name           Age
101            Christin       22
102            Mithun         25
103            Vaibhav        26

Enter your SQL query (or 'exit' to log out):
SELECT Name, Age FROM Student_Info;
Name           Age
Christin       22
Mithun         25
Vaibhav        26

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Student_Info WHERE Id = 102;
Id             Name           Age
102            Mithun         25

Enter your SQL query (or 'exit' to log out):
```

Figure 34 Using WHERE clause to display particular information

UPDATE

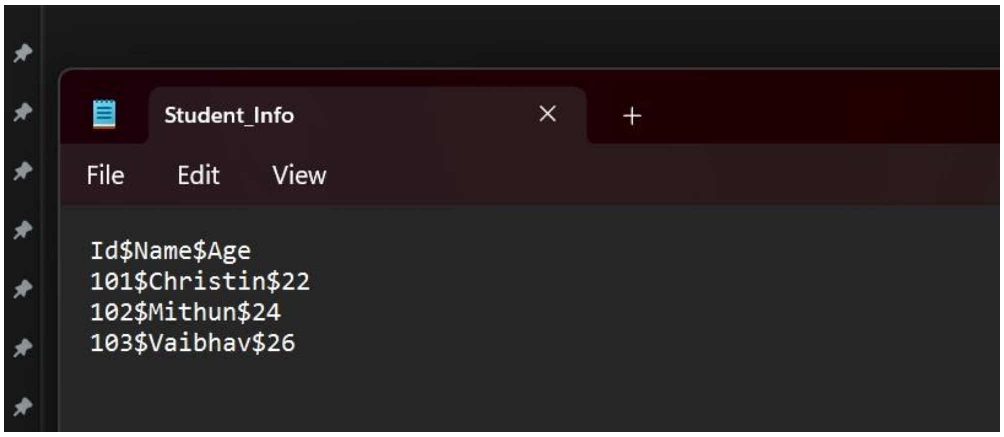
Query: UPDATE Student_Info SET Age = 24 WHERE Id = 102;

After a successful operation, it updates the data file.



```
Enter your SQL query (or 'exit' to log out):
INSERT INTO Student_Info (Id, Name, Age) VALUES (101, 'Christin', 22);

Data inserted successfully into table Student_Info

Enter your SQL query (or 'exit' to log out):
INSERT INTO Student_Info (Id, Name, Age) VALUES (102, 'Mithun', 25); INSERT INTO Student_Info

Data inserted successfully into table Student_Info

Data inserted successfully into table Student_Info

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Student_Info;
Id              Name            Age
101             Christin        22
102             Mithun          25
103             Vaibhav         26

Enter your SQL query (or 'exit' to log out):
SELECT Name, Age FROM Student_Info;
Name            Age
Christin        22
Mithun          25
Vaibhav         26

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Student_Info WHERE Id = 102;
Id              Name            Age
102             Mithun          25

Enter your SQL query (or 'exit' to log out):
UPDATE Student_Info SET Age = 24 WHERE Id = 102;

Table Student_Info updated successfully.

Enter your SQL query (or 'exit' to log out):
```

Figure 35 UPDATE operation into Student_Info table



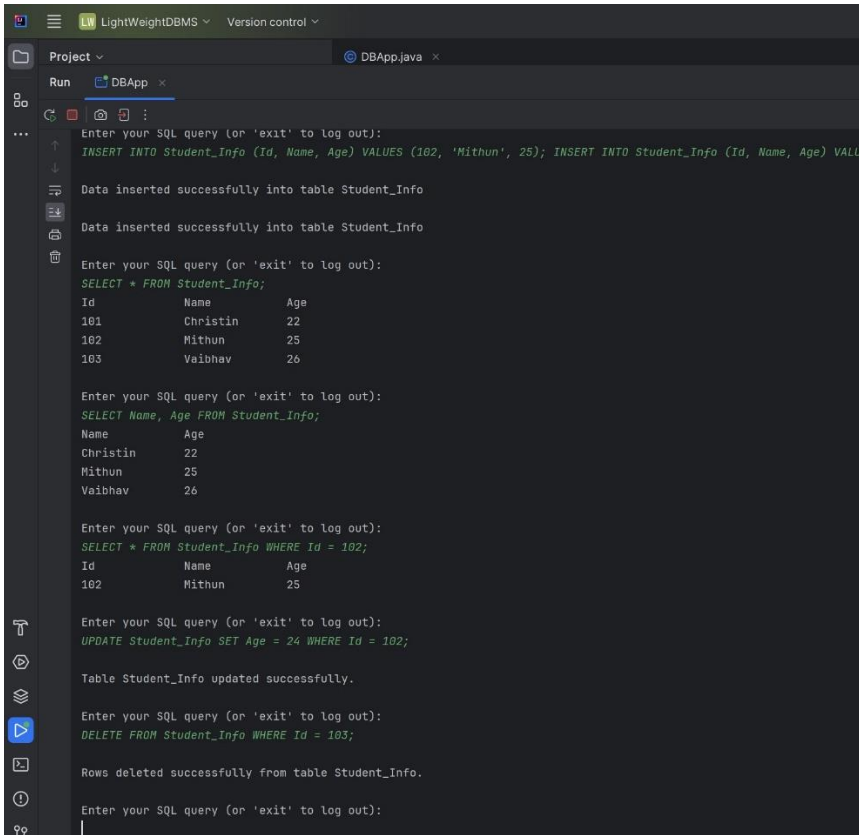
```
Id$Name$Age
101$Christin$22
102$Mithun$24
103$Vaibhav$26
```

Figure 36 Updated data in Student_Info

DELETE

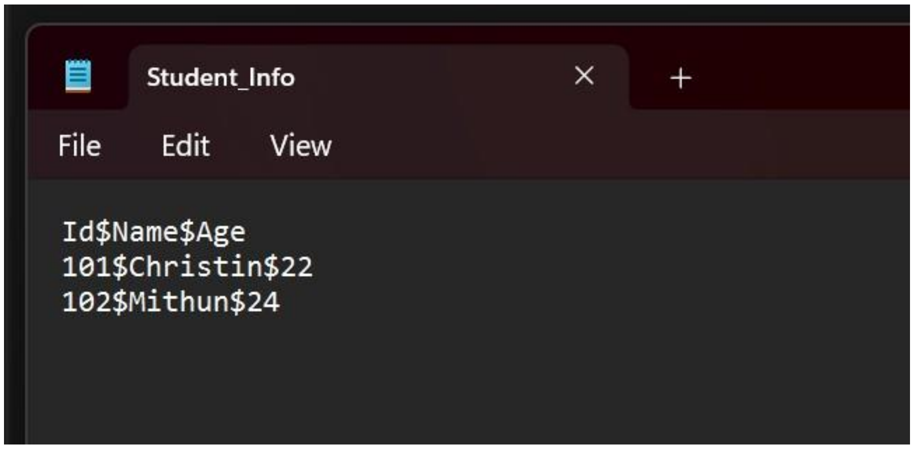
Query: DELETE FROM Student_Info WHERE Id = 103;

After successful execution, the particular information that is selected is removed from the data file.



```
Enter your SQL query (or 'exit' to log out):
INSERT INTO Student_Info (Id, Name, Age) VALUES (102, 'Mithun', 25); INSERT INTO Student_Info (Id, Name, Age) VALU

Data inserted successfully into table Student_Info

Data inserted successfully into table Student_Info

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Student_Info;
Id             Name           Age
101            Christin       22
102            Mithun         25
103            Vaibhav        26

Enter your SQL query (or 'exit' to log out):
SELECT Name, Age FROM Student_Info;
Name           Age
Christin       22
Mithun         25
Vaibhav        26

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Student_Info WHERE Id = 102;
Id             Name           Age
102            Mithun         25

Enter your SQL query (or 'exit' to log out):
UPDATE Student_Info SET Age = 24 WHERE Id = 102;

Table Student_Info updated successfully.

Enter your SQL query (or 'exit' to log out):
DELETE FROM Student_Info WHERE Id = 103;

Rows deleted successfully from table Student_Info.

Enter your SQL query (or 'exit' to log out):
```

Figure 37 Delete operation on Student_Info table



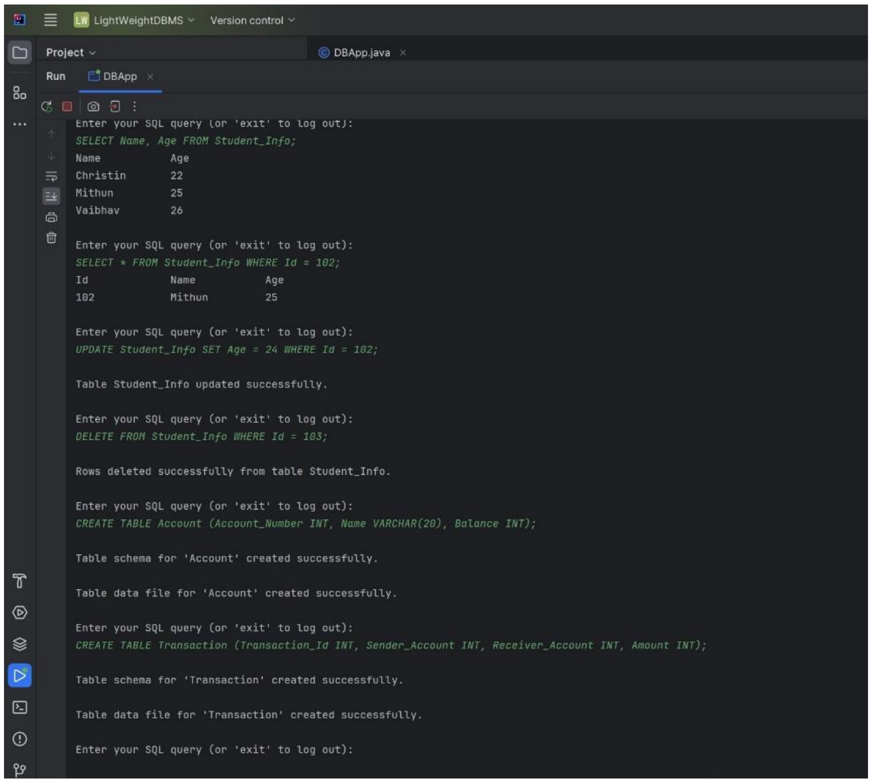
```
Id$Name$Age
101$Christin$22
102$Mithun$24
```

Figure 38 Update data in Student_Info table after delete operation

Task 4: Implementation of Transaction

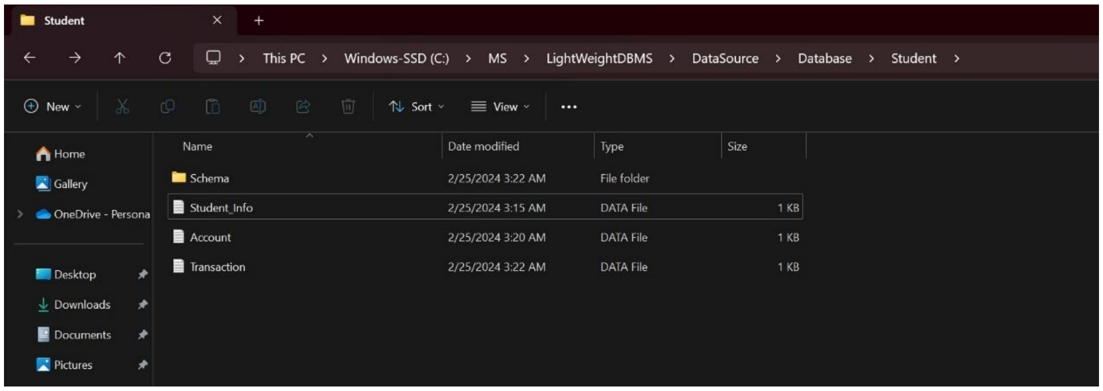
Setup

I created two tables, Account and Transaction, which I will use as base tables to run transaction operations on them. This will create `Account.data` and `Transaction.data` files, along with their respective metadata files in the Schema folder.



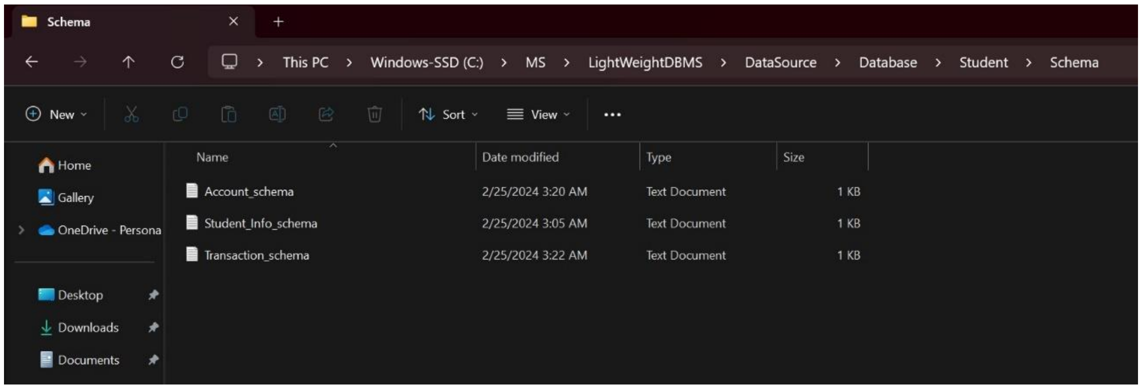
```
Enter your SQL query (or 'exit' to log out):
SELECT Name, Age FROM Student_Info;
Name        Age
Christin    22
Mithun      25
Vaibhav     26

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Student_Info WHERE Id = 102;
Id          Name        Age
102         Mithun      25

Enter your SQL query (or 'exit' to log out):
UPDATE Student_Info SET Age = 24 WHERE Id = 102;

Table Student_Info updated successfully.

Enter your SQL query (or 'exit' to log out):
DELETE FROM Student_Info WHERE Id = 103;

Rows deleted successfully from table Student_Info.

Enter your SQL query (or 'exit' to log out):
CREATE TABLE Account (Account_Number INT, Name VARCHAR(20), Balance INT);

Table schema for 'Account' created successfully.

Table data file for 'Account' created successfully.

Enter your SQL query (or 'exit' to log out):
CREATE TABLE Transaction (Transaction_Id INT, Sender_Account INT, Receiver_Account INT, Amount INT);

Table schema for 'Transaction' created successfully.

Table data file for 'Transaction' created successfully.

Enter your SQL query (or 'exit' to log out):
```

Figure 39 Created Account and Transaction table


Student
This PC > Windows-SSD (C:) > MS > LightWeightDBMS > DataSource > Database > Student >
New
Sort
View
Home
Gallery
OneDrive - Persona
Desktop
Downloads
Documents
Pictures

Name	Date modified	Type	Size
Schema	2/25/2024 3:22 AM	File folder	
Student_Info	2/25/2024 3:15 AM	DATA File	1 KB
Account	2/25/2024 3:20 AM	DATA File	1 KB
Transaction	2/25/2024 3:22 AM	DATA File	1 KB

Figure 40 Account.data and Transaction.data generated

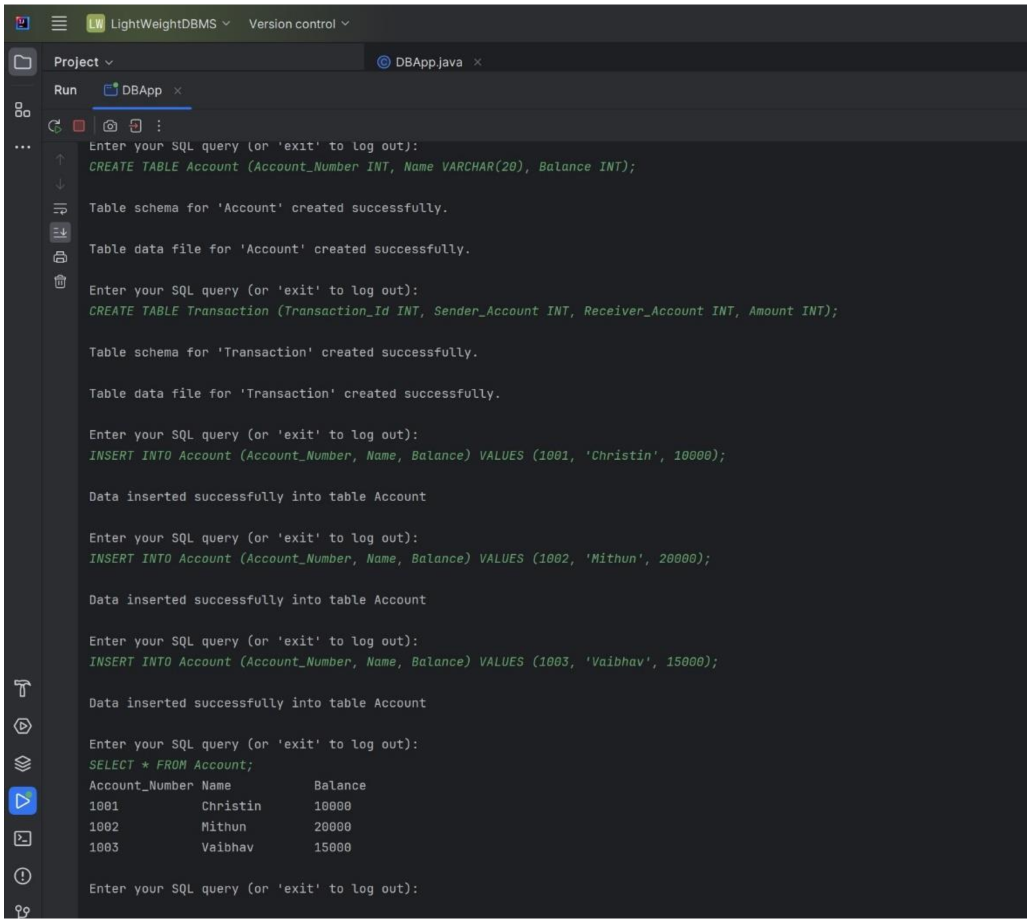


Name	Date modified	Type	Size
Account_schema	2/25/2024 3:20 AM	Text Document	1 KB
Student_Info_schema	2/25/2024 3:05 AM	Text Document	1 KB
Transaction_schema	2/25/2024 3:22 AM	Text Document	1 KB

Schema
This PC > Windows-SSD (C:) > MS > LightWeightDBMS > DataSource > Database > Student > Schema
New
Sort
View
Home
Gallery
OneDrive - Persona
Desktop
Downloads
Documents

Figure 41 Account_schema and Transaction_schema generated

I added some sample data to the Account table to run the operation on.



```
Enter your SQL query (or 'exit' to log out):
CREATE TABLE Account (Account_Number INT, Name VARCHAR(20), Balance INT);

Table schema for 'Account' created successfully.

Table data file for 'Account' created successfully.

Enter your SQL query (or 'exit' to log out):
CREATE TABLE Transaction (Transaction_Id INT, Sender_Account INT, Receiver_Account INT, Amount INT);

Table schema for 'Transaction' created successfully.

Table data file for 'Transaction' created successfully.

Enter your SQL query (or 'exit' to log out):
INSERT INTO Account (Account_Number, Name, Balance) VALUES (1001, 'Christin', 10000);

Data inserted successfully into table Account

Enter your SQL query (or 'exit' to log out):
INSERT INTO Account (Account_Number, Name, Balance) VALUES (1002, 'Mithun', 20000);

Data inserted successfully into table Account

Enter your SQL query (or 'exit' to log out):
INSERT INTO Account (Account_Number, Name, Balance) VALUES (1003, 'Vaibhav', 15000);

Data inserted successfully into table Account

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Account;
Account_Number Name          Balance
1001           Christin      10000
1002           Mithun        20000
1003           Vaibhav       15000

Enter your SQL query (or 'exit' to log out):
```

Figure 42 Sample data in Account Table

COMMIT

Query:

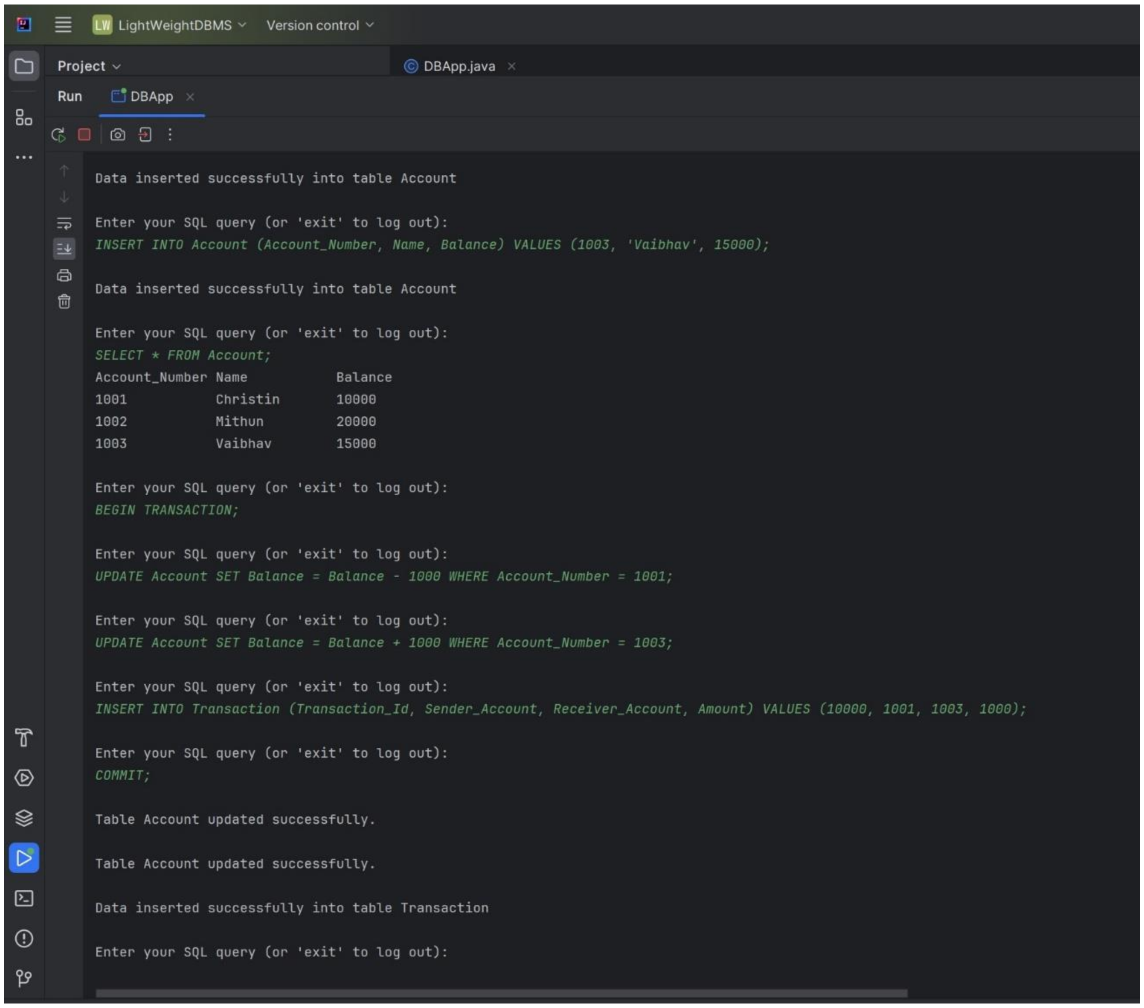
BEGIN TRANSACTION;

UPDATE Account SET Balance = Balance – 1000 WHERE Account_Number = 1001;

UPDATE Account SET Balance = Balance + 1000 WHERE Account_Number = 1002;

INSERT INTO Transaction (Transaction_Id, Sender_Account, Receiver_Account, Amount) VALUES (10000, 1001, 1003, 1000);

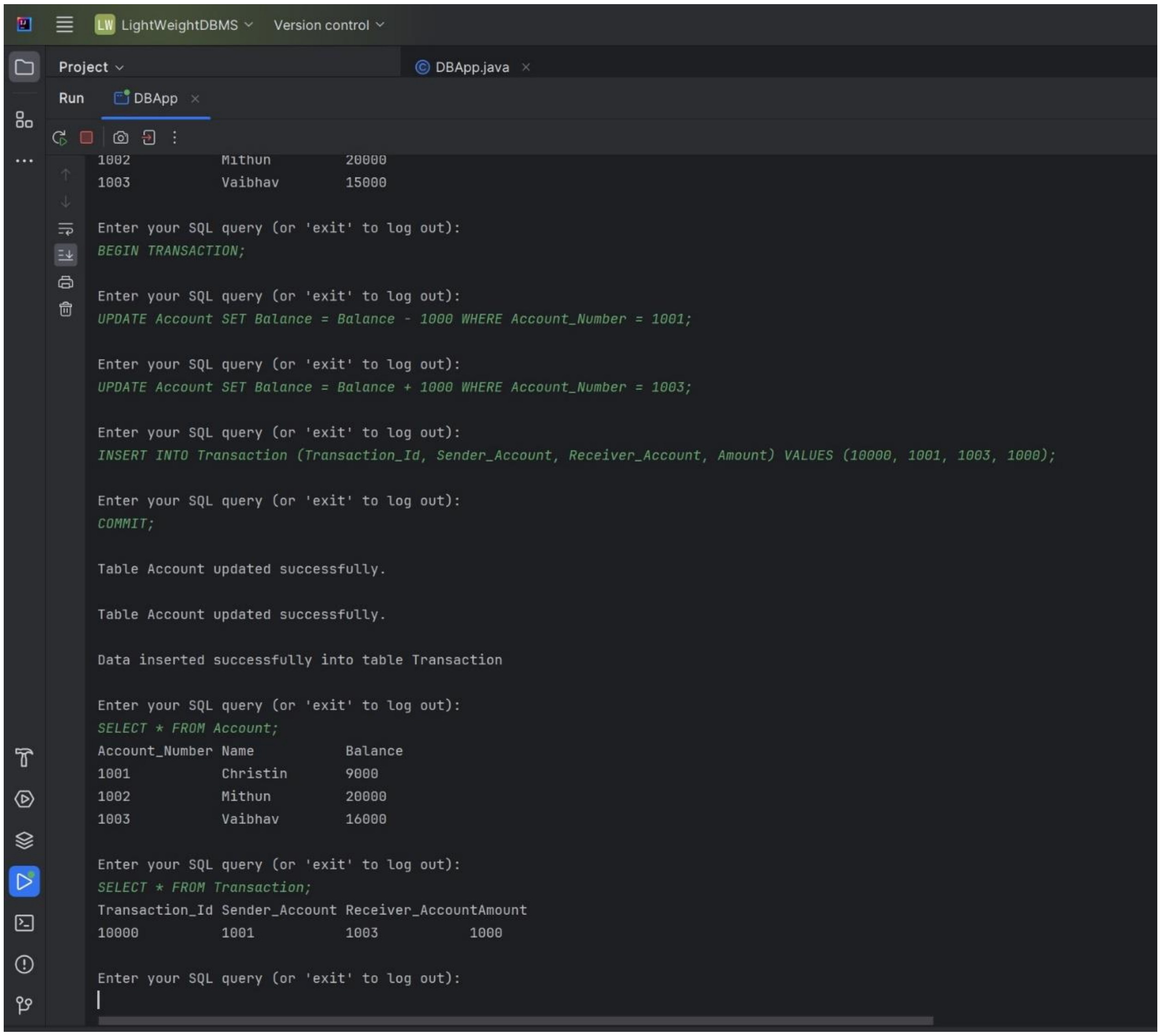
COMMIT;



```
Data inserted successfully into table Account

Enter your SQL query (or 'exit' to log out):
INSERT INTO Account (Account_Number, Name, Balance) VALUES (1003, 'Vaibhav', 15000);

Data inserted successfully into table Account

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Account;
Account_Number Name           Balance
1001           Christin       10000
1002           Mithun         20000
1003           Vaibhav        15000

Enter your SQL query (or 'exit' to log out):
BEGIN TRANSACTION;

Enter your SQL query (or 'exit' to log out):
UPDATE Account SET Balance = Balance - 1000 WHERE Account_Number = 1001;

Enter your SQL query (or 'exit' to log out):
UPDATE Account SET Balance = Balance + 1000 WHERE Account_Number = 1003;

Enter your SQL query (or 'exit' to log out):
INSERT INTO Transaction (Transaction_Id, Sender_Account, Receiver_Account, Amount) VALUES (10000, 1001, 1003, 1000);

Enter your SQL query (or 'exit' to log out):
COMMIT;

Table Account updated successfully.

Table Account updated successfully.

Data inserted successfully into table Transaction

Enter your SQL query (or 'exit' to log out):
```

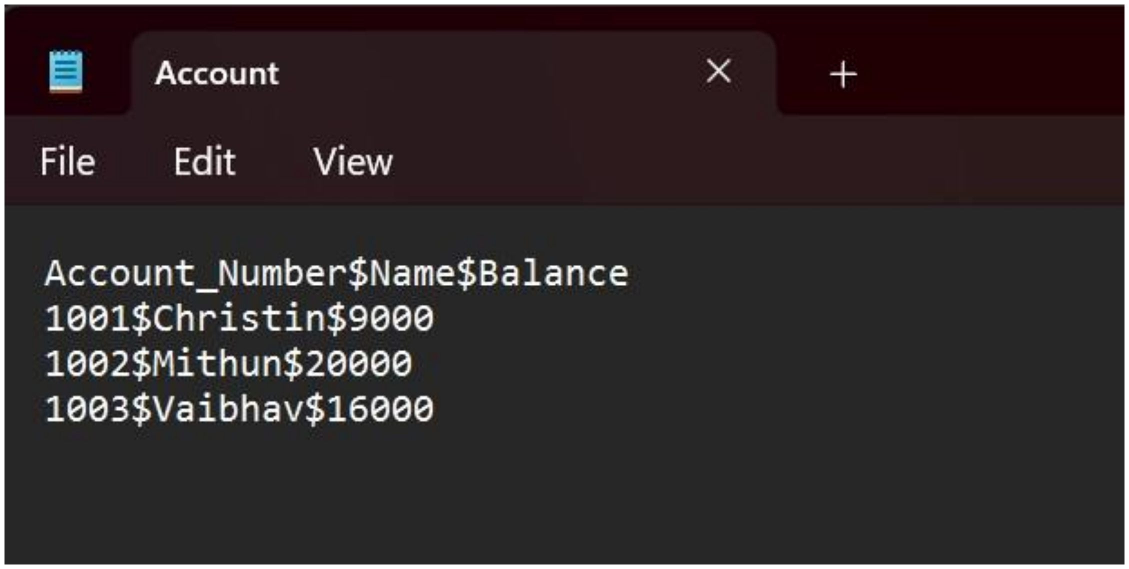
Figure 43 COMMIT operation for the transaction

After the COMMIT command, all the transactions are updated and reflected in the database file.



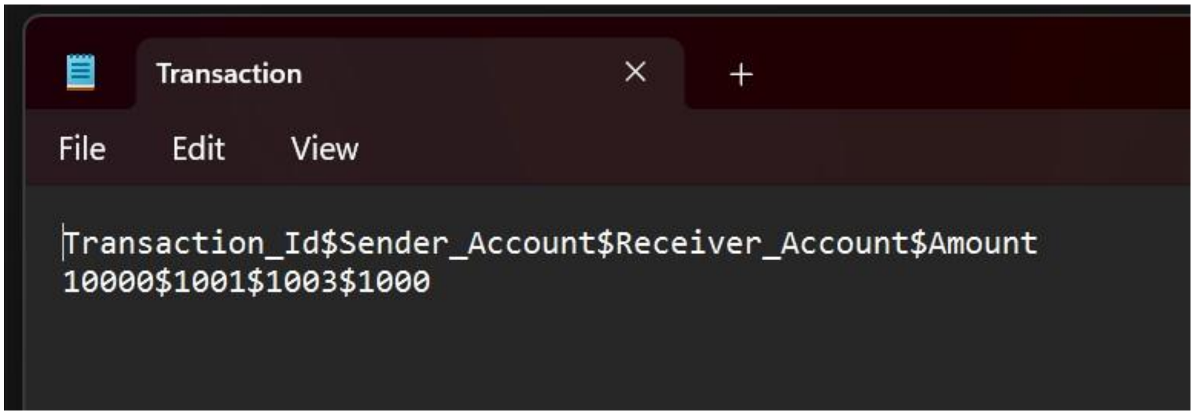
```
1002          Mithun        20000
1003          Vaibhav       15000

Enter your SQL query (or 'exit' to log out):
BEGIN TRANSACTION;

Enter your SQL query (or 'exit' to log out):
UPDATE Account SET Balance = Balance - 1000 WHERE Account_Number = 1001;

Enter your SQL query (or 'exit' to log out):
UPDATE Account SET Balance = Balance + 1000 WHERE Account_Number = 1003;

Enter your SQL query (or 'exit' to log out):
INSERT INTO Transaction (Transaction_Id, Sender_Account, Receiver_Account, Amount) VALUES (10000, 1001, 1003, 1000);

Enter your SQL query (or 'exit' to log out):
COMMIT;

Table Account updated successfully.

Table Account updated successfully.

Data inserted successfully into table Transaction

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Account;
Account_Number Name          Balance
1001          Christin      9000
1002          Mithun        20000
1003          Vaibhav       16000

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Transaction;
Transaction_Id Sender_Account Receiver_AccountAmount
10000         1001          1003          1000

Enter your SQL query (or 'exit' to log out):
```

Figure 44 Updated information in the table after successful commit



```
Account_Number$Name$Balance
1001$Christin$9000
1002$Mithun$20000
1003$Vaibhav$16000
```

Figure 45 Account table after commit operation



```
Transaction_Id$Sender_Account$Receiver_Account$Amount
10000$1001$1003$1000
```

Figure 46 Transaction table after commit operation

ROLLBACK

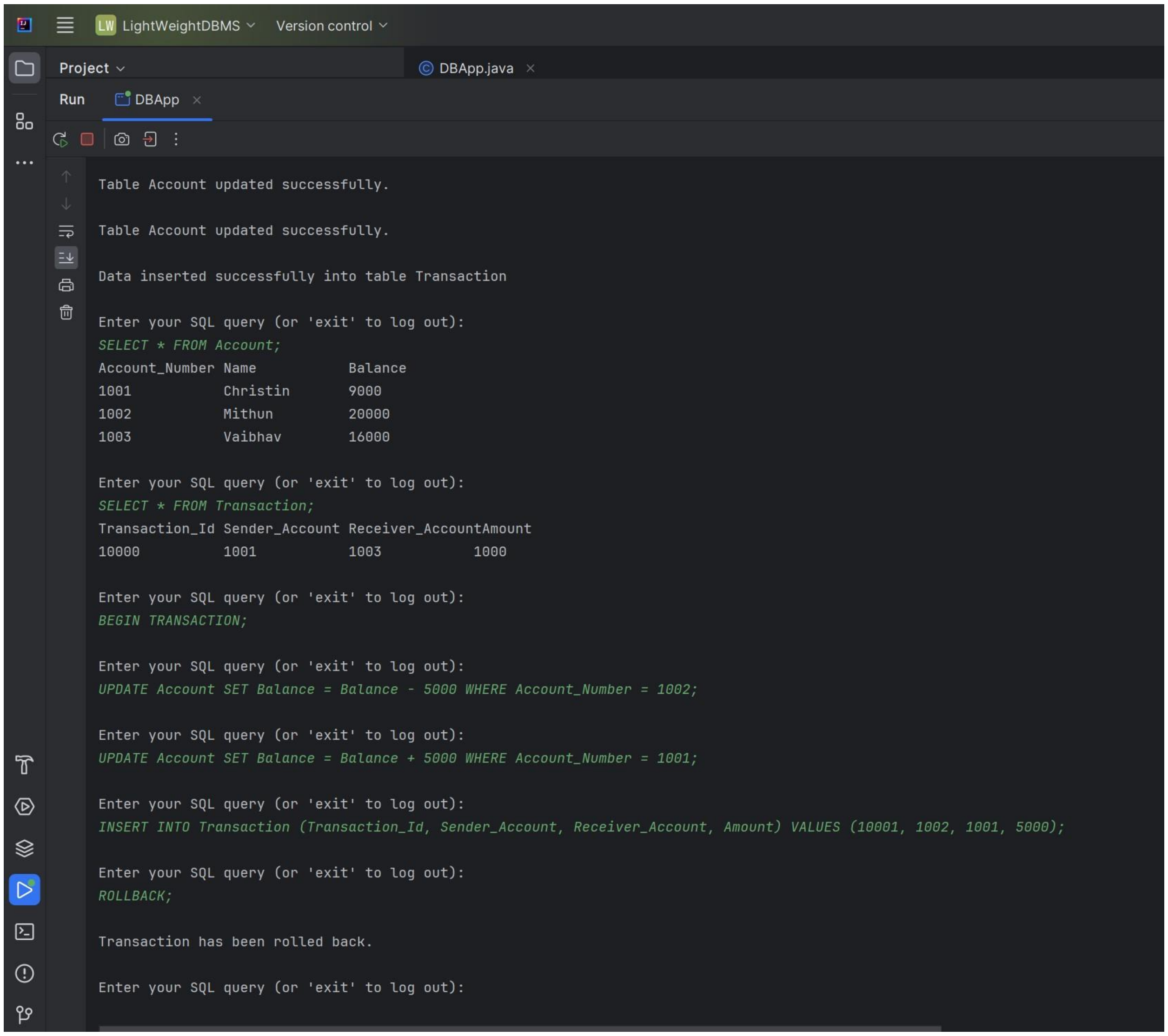
Query:

BEGIN TRANSACTION;

UPDATE Account SET Balance = Balance – 5000 WHERE Account_Number = 1002;

UPDATE Account SET Balance = Balance + 5000 WHERE Account_Number = 1001;

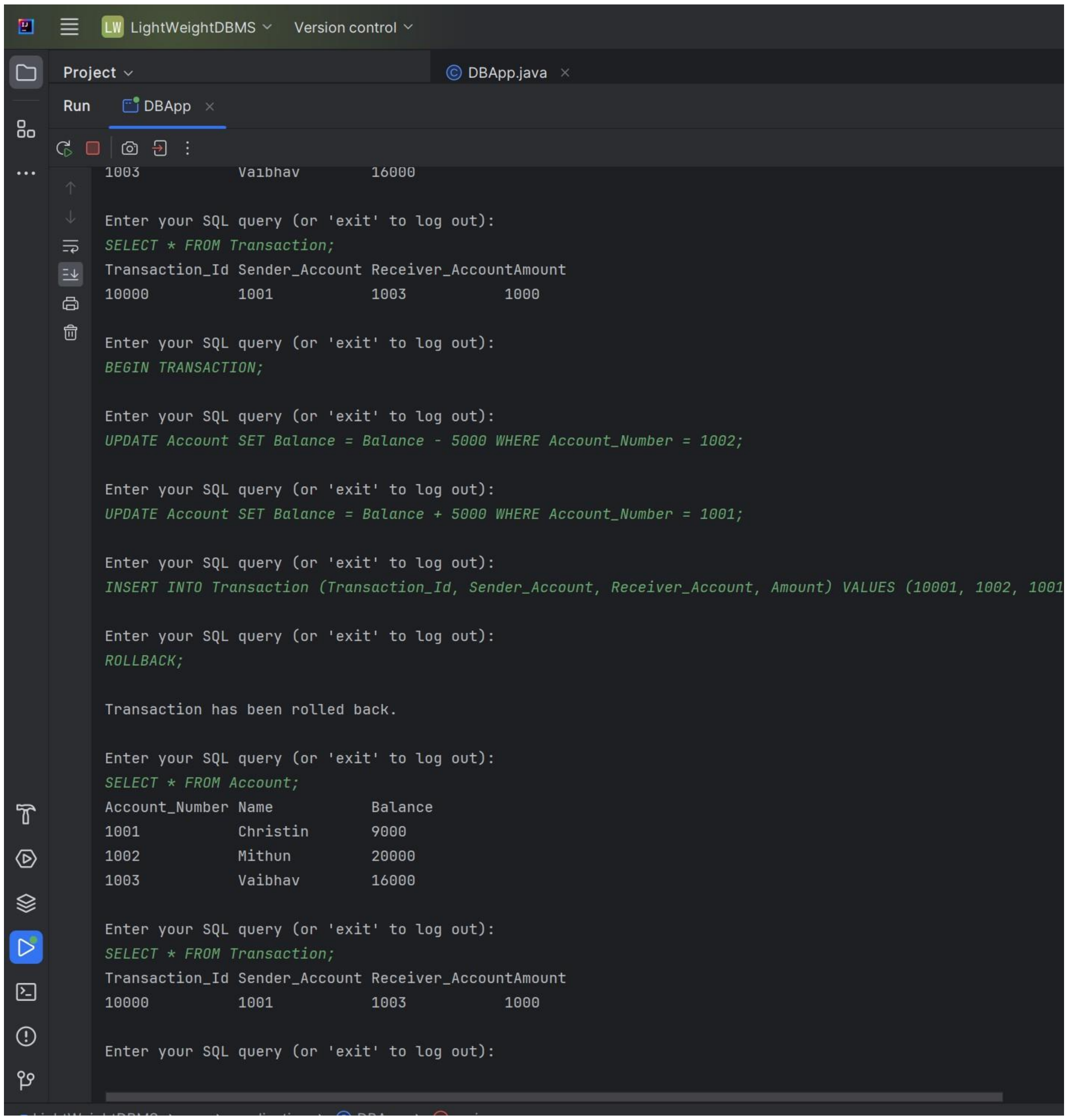
INSERT INTO Transaction (Transaction_Id, Sender_Account, Receiver_Account, Amount) VALUES (10001, 1002, 1001, 5000);



```
Table Account updated successfully.

Table Account updated successfully.

Data inserted successfully into table Transaction

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Account;
Account_Number Name           Balance
1001           Christin       9000
1002           Mithun         20000
1003           Vaibhav        16000

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Transaction;
Transaction_Id Sender_Account Receiver_AccountAmount
10000          1001           1003           1000

Enter your SQL query (or 'exit' to log out):
BEGIN TRANSACTION;

Enter your SQL query (or 'exit' to log out):
UPDATE Account SET Balance = Balance - 5000 WHERE Account_Number = 1002;

Enter your SQL query (or 'exit' to log out):
UPDATE Account SET Balance = Balance + 5000 WHERE Account_Number = 1001;

Enter your SQL query (or 'exit' to log out):
INSERT INTO Transaction (Transaction_Id, Sender_Account, Receiver_Account, Amount) VALUES (10001, 1002, 1001, 5000);

Enter your SQL query (or 'exit' to log out):
ROLLBACK;

Transaction has been rolled back.

Enter your SQL query (or 'exit' to log out):
```

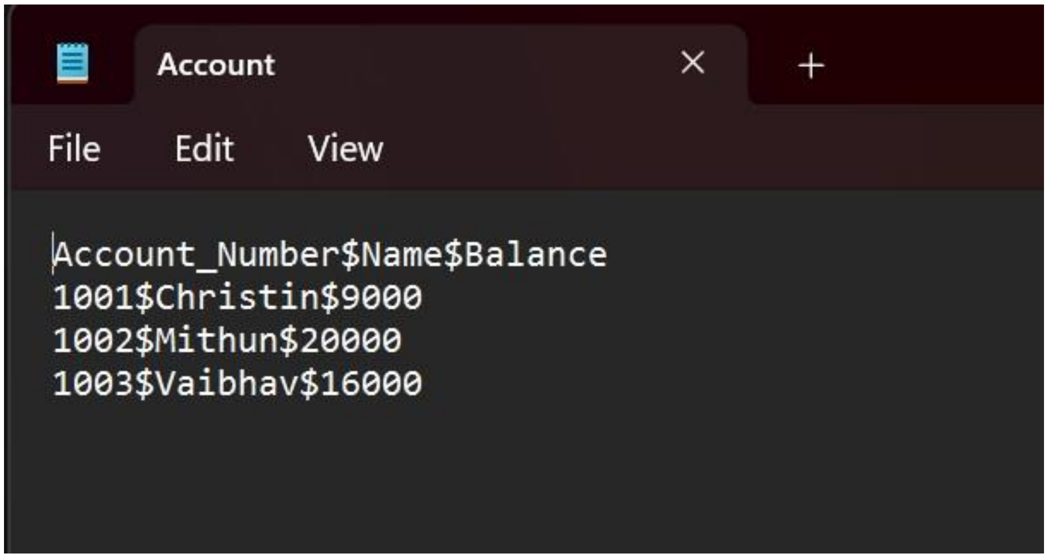
Figure 47 Rollback operation revert back the transaction

As soon as I enter ROLLBACK, all the operations from the point I started the transaction revert without updating the table.



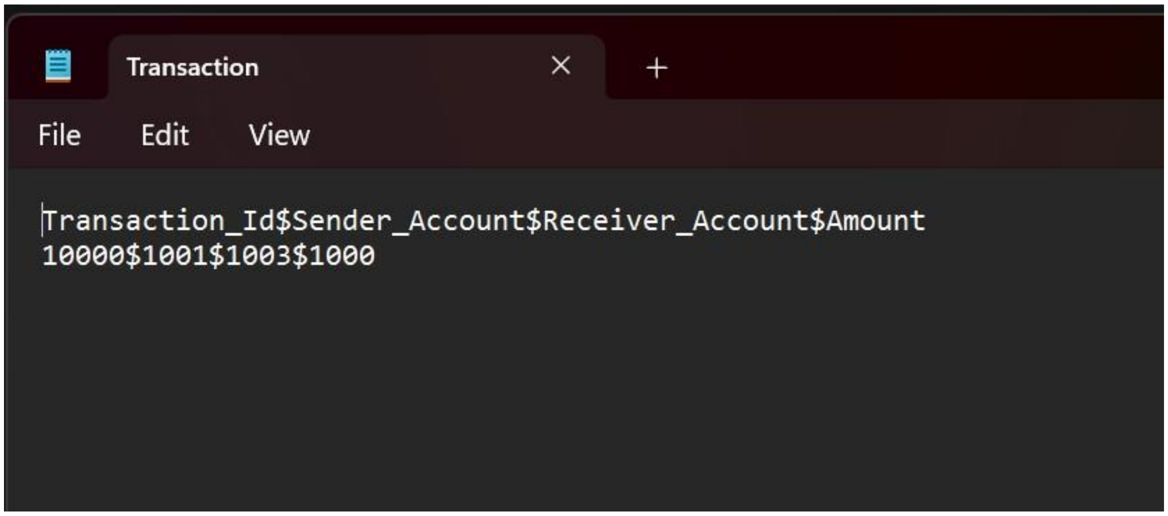
```
1003          Vaibhav       16000

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Transaction;
Transaction_Id Sender_Account Receiver_AccountAmount
10000          1001           1003           1000

Enter your SQL query (or 'exit' to log out):
BEGIN TRANSACTION;

Enter your SQL query (or 'exit' to log out):
UPDATE Account SET Balance = Balance - 5000 WHERE Account_Number = 1002;

Enter your SQL query (or 'exit' to log out):
UPDATE Account SET Balance = Balance + 5000 WHERE Account_Number = 1001;

Enter your SQL query (or 'exit' to log out):
INSERT INTO Transaction (Transaction_Id, Sender_Account, Receiver_Account, Amount) VALUES (10001, 1002, 1001

Enter your SQL query (or 'exit' to log out):
ROLLBACK;

Transaction has been rolled back.

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Account;
Account_Number Name          Balance
1001           Christin      9000
1002           Mithun        20000
1003           Vaibhav       16000

Enter your SQL query (or 'exit' to log out):
SELECT * FROM Transaction;
Transaction_Id Sender_Account Receiver_AccountAmount
10000          1001           1003           1000

Enter your SQL query (or 'exit' to log out):
```

Figure 48 No updates on the Account and Transaction table after rollback



```
Account_Number$Name$Balance
1001$Christin$9000
1002$Mithun$20000
1003$Vaibhav$16000
```

Figure 49 No update on Account table after rollback



```
Transaction_Id$Sender_Account$Receiver_Account$Amount
10000$1001$1003$1000
```

Figure 50 No update on Transaction table after rollback

References

[1] Sardar Mudassar Ali Khan, “Layered Architecture Used in Software Development,” *DEV Community*, [Online], June 14, 2023. Available: https://dev.to/sardarmudassaralikhan/layered-architecture-used-in-software-development-8jd [Accessed: February 24, 2024].

[2] Samuel Oloruntoba, “Solid: The First 5 Principles of Object Oriented Design,” *DigitalOcean*, November 30, 2021. Available: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design [Accessed: February 24, 2024].