CSCI 5408
Data Management and Warehousing

Sprint Report - 3

Group members:

Kenee Ashok Patel (B00969805)

Vraj Sunilkumar Shah (B00979965)

Shivani Uppe (B00976573)

GitLab Project Link: https://git.cs.dal.ca/kenee/dbms-builder-11

Table of Contents

Table of Figures

1. Pseudocode

1.1. ERD Generator

Class ERDGenerator

Attributes:

- database: The Database object
- tableRelationships: A map that holds the relationships for each table

Constructor ERDGenerator(database, tableRelationships):

- Initialize the database attribute with the provided database
- Initialize the tableRelationships attribute with the provided tableRelationships

Method generateERD():

- Create an empty StringBuilder called erd
- Append the database name to the erd string with the heading "Entity-Relationship Diagram for Database:"
- Add a couple of newlines for formatting

- For each table in the database:
  - Append the table name to the erd string with the heading "Table:"
  - Add a subheading "Columns:"

  - For each column in the table:
    - Append the column name and type to the erd string
    - If the column has constraints, append them to the erd string
    - Add a newline for each column

    - If the column is a foreign key (i.e., it references another table):
      - Append the foreign key information to the erd string
      - Find the referenced table and column in the database

- If the referenced table or column doesn't exist, skip to the next column

- Create a new Relationship object using the column and its referenced column
- Add this relationship to the tableRelationships map for the current table

- Add a newline after listing all columns for the current table

- Append a heading "Relationships and Cardinality:" to the erd string

- For each table and its relationships in the tableRelationships map:
  - Append the table name to the erd string with the heading "Table:"

  - For each relationship in the table:
    - Append the relationship details, including column names and their cardinalities, to the erd string

  - Add a newline after listing all relationships for the current table

- Return the erd string as a formatted ERD representation

End Class

2. Link to Git repository and meeting logs

Git Repository: https://git.cs.dal.ca/kenee/dbms-builder-11
Meeting logs: Data Meeting Logs.xlsx

3. Test cases and evidence of testing for whole project

3.1.Create Database

Create a database

```
/Users/lib-user/.local/share/mise/installs/java/21.0.2/b
Welcome to TinyDb, please start writing queries below.

dbms_builder_11 > create database simple_name;
Database created: simple_name
```

Figure 1: CREATE TABLE demonstration by creating a table with name 'simple_name'

Figure 2: Folder creation succession after creating the database

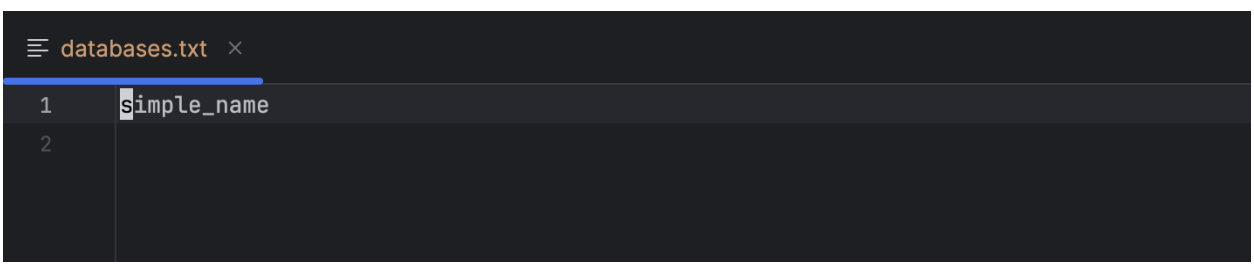

Figure 3: Contents of databases.txt

```
dbms_builder_11 > create database
Error: Invalid CREATE DATABASE query
```

Figure 4: Create database and pass invalid database name

3.2. Use Database

Using a database which was created earlier with create database statement

```
dbms_builder_11 > use simple_name;
Using database: simple_name
```

Figure 5: USE `DATABASE_NAME` demonstration by using simple_name database

```
dbms_builder_11 > use this_does_not_exist;
Error: Database does not exist: this_does_not_exist
```

Figure 6: Demonstration of USE `DATABASE_NAME` failing due to the database not existing

For the main functionality of using the correct database for table insertion, we will be creating a database named dbms, create a table named db_table in it and select * from it. Then we will try selecting everything from db_table but from simple_name database which should fail as there is no table named db_table in that.

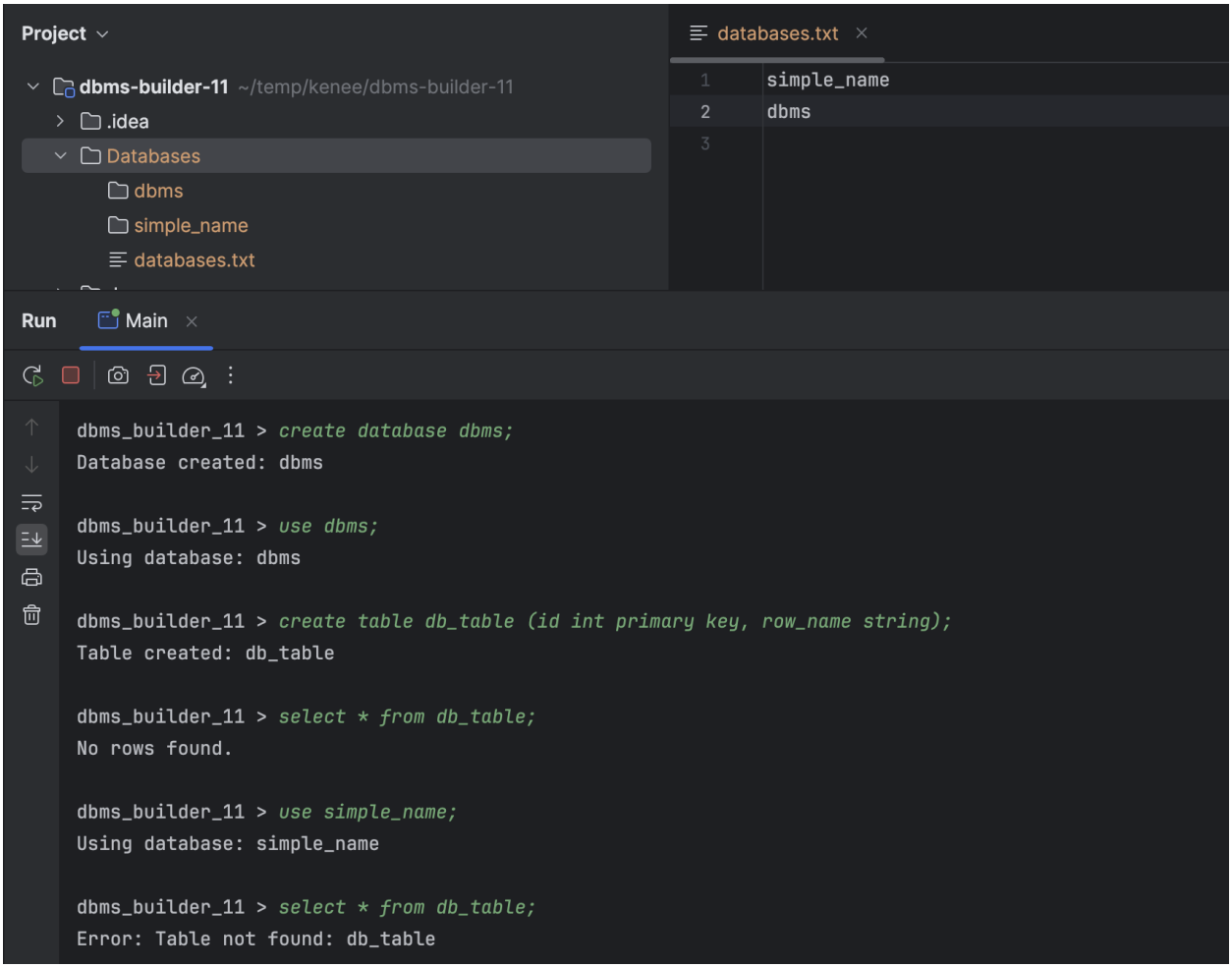

Figure 7: Demonstration of using database management

3.3. Create table

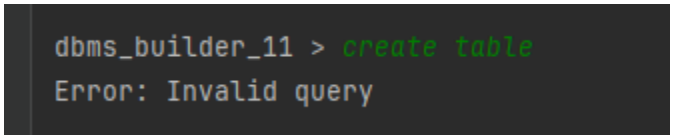

Figure 8: Create table with empty name

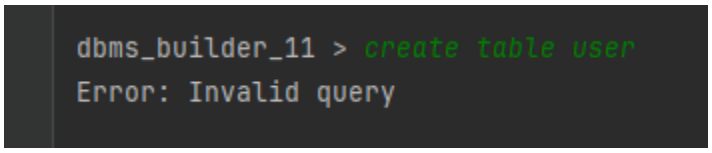

Figure 9: Create table without column definition

```
dbms_builder_11 > create table student (id)
Error: Invalid column definition: id
```

Figure 10: Create table with invalid column definition

```
dbms_builder_11 > create table student (id int, name varchar(100))
Error: Invalid column type: varchar(100. Allowed types: [int, string, double]
```

Figure 11: Create table with non-allowed data type

```
dbms_builder_11 > create table student (id int PRIMARY KEY, name varchar(100))
Error: Invalid constraint: PRIMARY. Allowed constraints: [primary_key, non_null, auto_increment, unique]
```

Figure 12: Create table with non-allowed column constraint

```
dbms_builder_11 > create table student (id int primary_key auto_increment, name string, user_id int foreign_key noTable.id)
Error: Invalid table in foreign key definition
```

Figure 13: Create table with invalid table in foreign key

```
dbms_builder_11 > create table student (id int primary_key auto_increment, name string, user_id int foreign_key user.non_existing_id)
Error: Invalid column in foreign key definition
```

Figure 14: Invalid column in foreign key definition

User table with 3 attributes i.e. id, name and age.

```
dbms_builder_11 > CREATE TABLE User (id INT PRIMARY KEY, name STRING, age INT);
Table created: User

dbms_builder_11 >
```

Figure 15: CREATE TABLE demonstration using User table

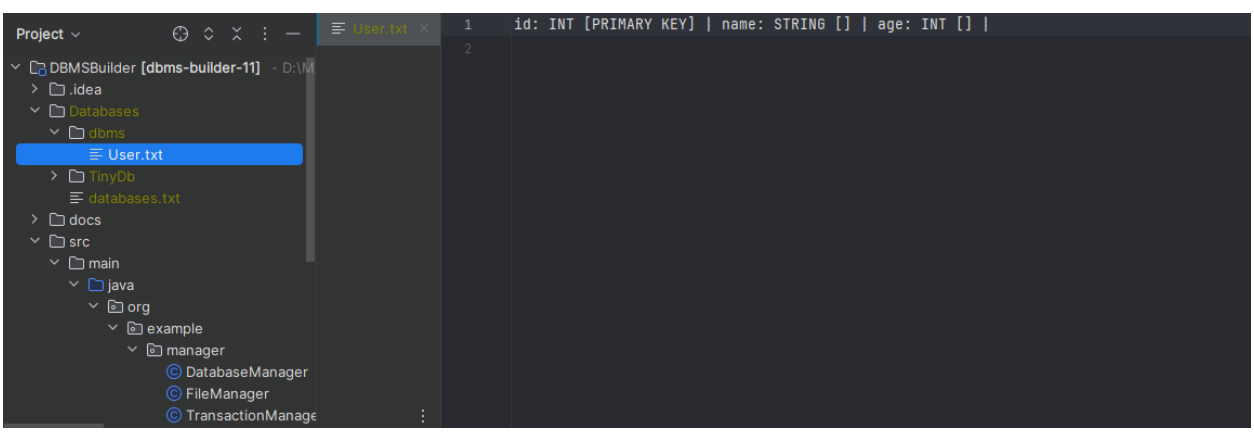

Figure 16: File creation succession after creating table

3.4. Insert into table

Insert a row into user table

```
dbms_builder_11 > INSERT INTO User (id, name, age) VALUES (1, "Vraj Shah", 24);
Row added successfully.

dbms_builder_11 >
```

Figure 17: INSERT INTO demonstration by inserting a row into User

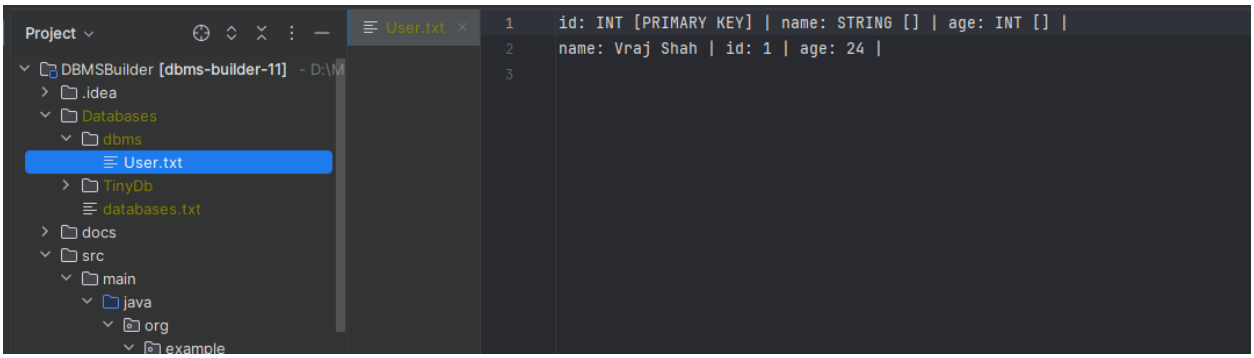

Figure 18: File row addition successful after inserting data

```
dbms_builder_11 > insert into user (id, name) values (1, "Vraj")
Error: Column email must have a value
```

Figure 19: Inserting into table where a column must have a value and not provided

```
dbms_builder_11 > insert into user (id, email) values (4, a@bgmail.com)
email has primary_key/unique constraint so it must have unique value.
Error: Invalid value: a@bgmail.com for column: Column[name=email, type=string, constraints=[unique, non_null], foreignKeyTable=null, foreignKeyColumn=null]
```

Figure 20: Inserting redundant value into column with unique constraint

3.5. Select from table

Select all columns from User.

```
dbms_builder_11 > SELECT * FROM User;
name      | id | age |
----------+----+-----+-
Vraj Shah | 1  | 24  |

dbms_builder_11 >
```

Figure 21: Selecting all columns from User table

Selecting specific columns from User.

```
dbms_builder_11 > SELECT name, age FROM User;
name      | age |
----------+-----+-
Vraj Shah | 24  |

dbms_builder_11 >
```

Figure 22: Select specific columns from User table

Selecting columns from User table where age = 24

```
dbms_builder_11 > SELECT * FROM User where age = 24;
name      | id | age |
----------+----+-----+-
Vraj Shah | 1  | 24  |

dbms_builder_11 >
```

Figure 23: Select from User where age is 24

Selecting columns from User table where age != 24

```
dbms_builder_11 > SELECT * FROM User where age != 24;
No rows found.

dbms_builder_11 >
```

Figure 24: Select from User where age is not equal to 24

Selecting columns from User table where age < 24

```
dbms_builder_11 > SELECT * FROM User where age < 24;
No rows found.

dbms_builder_11 >
```

Figure 25: Select from User where age is less than 24

Selecting columns from User table where age <= 24

```
dbms_builder_11 > SELECT * FROM User where age <= 24;
name      | id | age |
----------+----+-----+-
Vraj Shah | 1  | 24  |

dbms_builder_11 >
```

Figure 26: Select from User where age is less than or equal to 24

Selecting columns from User table where age > 24

```
dbms_builder_11 > SELECT * FROM User where age > 24;
No rows found.

dbms_builder_11 >
```

Figure 27: Select from User where age is greater than 24

Selecting columns from User table where age >= 24

```
dbms_builder_11 > SELECT * FROM User where age >= 24;
name      | id | age |
----------+----+-----+-
Vraj Shah | 1  | 24  |

dbms_builder_11 >
```

Figure 28: Select from User where age is greater than or equal to 24

Selecting columns from User table where age IN 24, 20, 21

```
dbms_builder_11 > SELECT * FROM User where age IN (24, 20, 21);
name      | id | age |
----------+----+-----+-
Vraj Shah | 1  | 24  |

dbms_builder_11 >
```

Figure 29: Select from User where age is either 24 or 20 or 21

3.6. Update table

Table before updating

```
dbms_builder_11 > select * from User;
name         | id | age |
-------------+----+-----+-
Vraj Shah    | 1  | 24  |
Shivani Uppe | 2  | 22  |
Kenne Patel  | 3  | 23  |
```

Figure 30: Table before performing update

Updating User where id = 2

```
dbms_builder_11 > update User set age = 25 where id = 2;
1 row(s) affected.

dbms_builder_11 > select * from User;
name         | id | age |
-------------+----+-----+-
Vraj Shah    | 1  | 24  |
Shivani Uppe | 2  | 25  |
Kenne Patel  | 3  | 23  |
```

Figure 31: Updating age where id = 2

Updating User who does not exist

```
dbms_builder_11 > update User set age = 26 where id = 4;
0 row(s) affected.

dbms_builder_11 > select * from User;
name         | id | age |
-------------+----+-----+-
Vraj Shah    | 1  | 24  |
Shivani Uppe | 2  | 25  |
Kenne Patel  | 3  | 23  |
```

Figure 32: Updating age for an id which does not exist

3.7. Delete from table

Deleting from User where id = 3

```
dbms_builder_11 > delete from User where id = 3;
1 row(s) deleted successfully.

dbms_builder_11 > select * from User;
name         | id | age |
-------------+----+-----+-
Vraj Shah    | 1  | 24  |
Shivani Uppe | 2  | 25  |
```

Figure 33: Delete from User where id = 3

Deleting from User where id does not exist

```
dbms_builder_11 > delete from User where id = 5;
0 row(s) deleted successfully.

dbms_builder_11 > select * from User;
name         | id | age |
-------------+----+-----+-
Vraj Shah    | 1  | 24  |
Shivani Uppe | 2  | 25  |
```

Figure 34: Deleting a User with an id which does not exist

3.8. Drop table

Dropping Table

```
dbms_builder_11 > drop table User;
Table dropped: User
```

Figure 35: Dropping table

3.9. Transaction

Start transaction, insert row

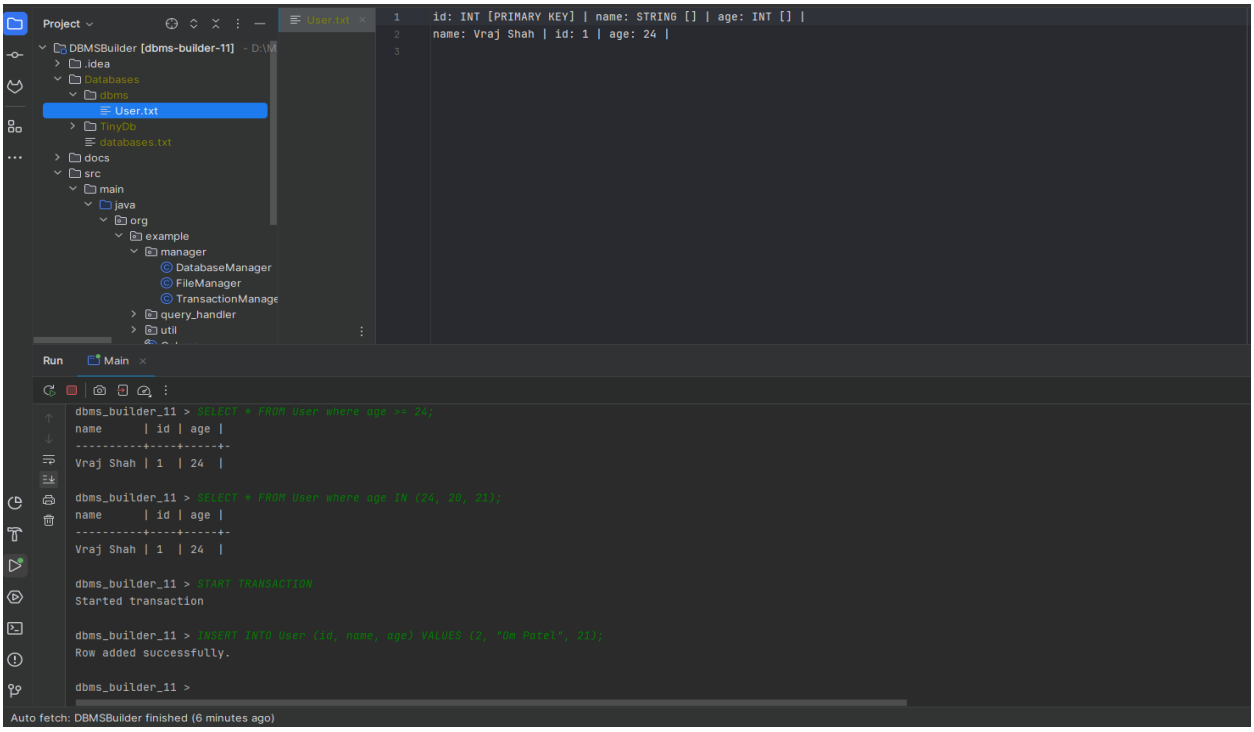

Figure 36: Starting transaction and inserting a row

Rollback changes

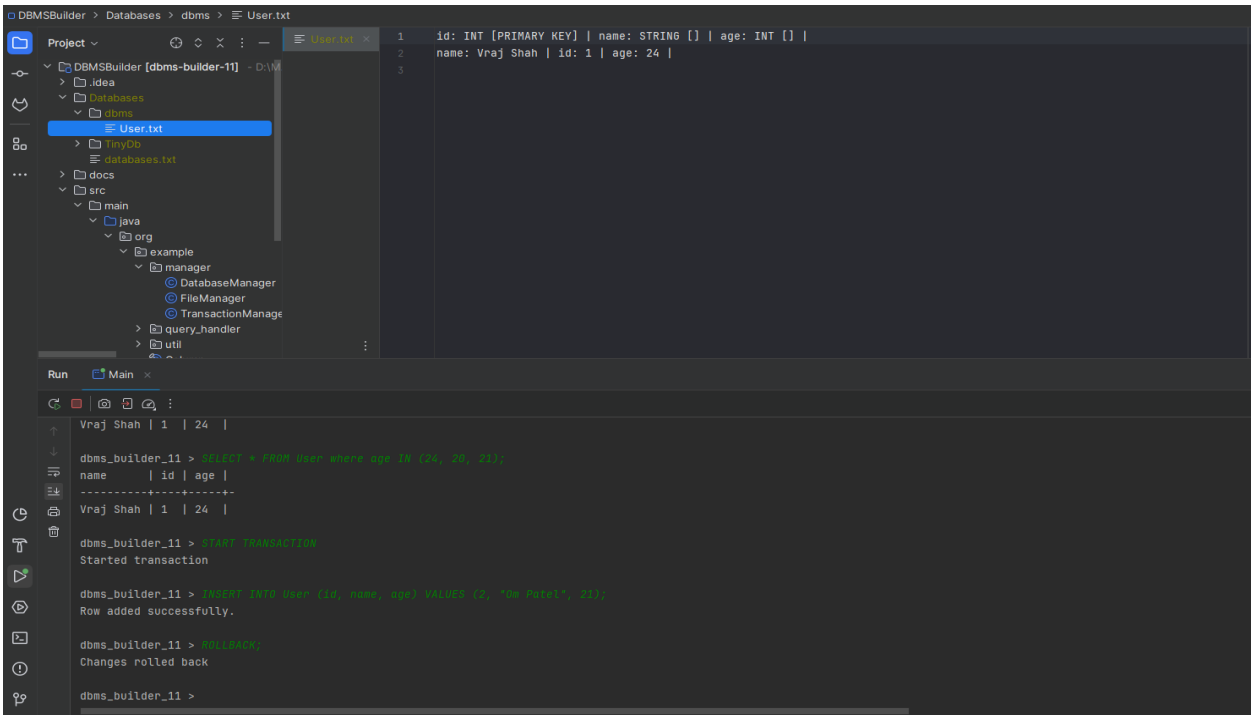

Figure 37: Rollback transaction and discard buffer data

Commit changes

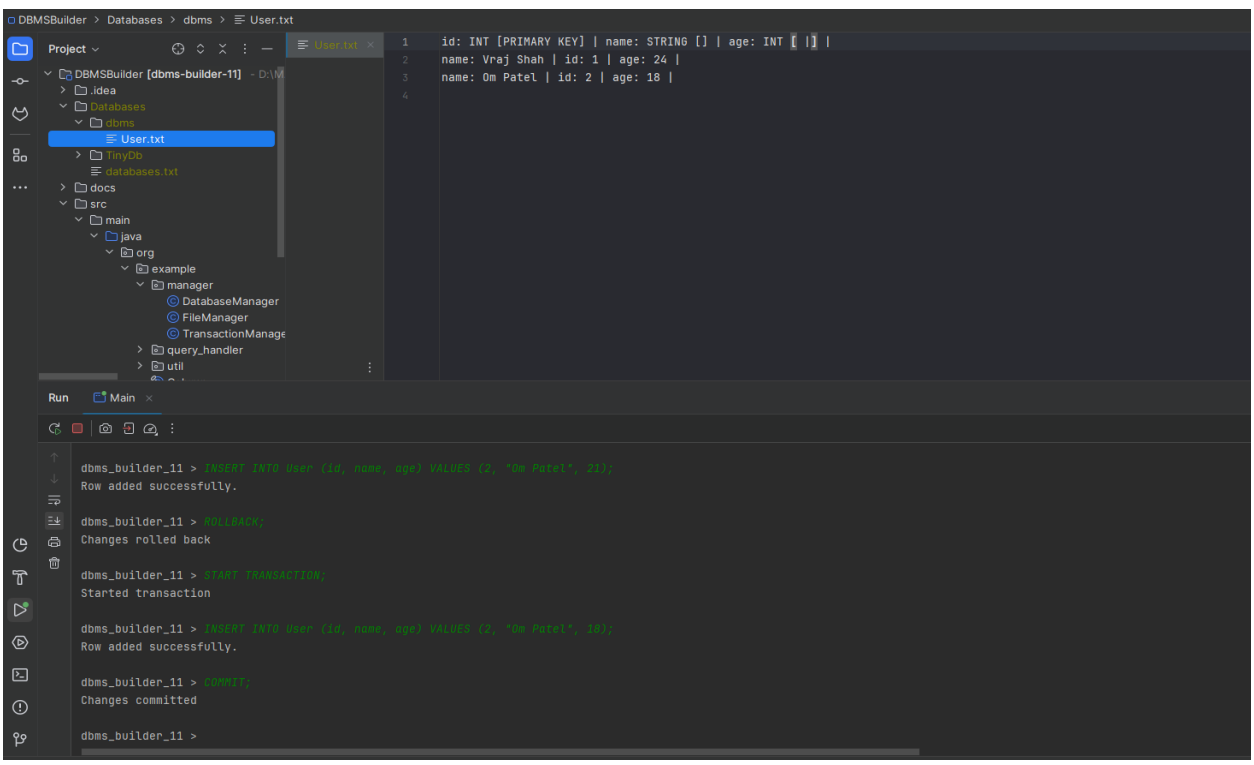

Figure 38: Commit changes to User table

3.10. Log Management

Logging queries

```
dbms_builder_11 > create database dbms;
Database created: dbms

dbms_builder_11 > use dbms;
Using database: dbms

dbms_builder_11 > create table people (id int primary key, name string);
Table created: people

dbms_builder_11 > insert into (id, name) values (1, "Kenny");
Error: Table not found: (id,

dbms_builder_11 > insert into people (id, name) values (1, "Kenny");
Row added successfully.

dbms_builder_11 > select * from people;
name  | id |
------+----+-
Kenny | 1  |

dbms_builder_11 > drop table people;
Table dropped: people
```

Figure 39 Queries ran to generate logs


```
{"query":"create database dbms;","execution_time":31,"timestamp":"2024-07-13T21:52:22.90826
{"query":"use dbms;","execution_time":1,"timestamp":"2024-07-13T21:52:26.247108"}
{"query":"create table people (id int primary key, name string);","execution_time":15,"timestamp":"20
{"query":"insert into (id, name) values (1, \"Kenny\");","execution_time":2,"timestamp":"2024-07-13T2
{"query":"insert into people (id, name) values (1, \"Kenny\");","execution_time":16,"timestamp":"2024
{"query":"select * from people;","execution_time":11,"timestamp":"2024-07-13T21:54:30.760922"}
{"query":"drop table people;","execution_time":2,"timestamp":"2024-07-13T21:54:39.992920"}
{"query":"exit","execution_time":0,"timestamp":"2024-07-13T21:54:51.212682"}
```


Figure 40 Content that was filled in query_logs.json


```
{"executionTime":11,"dbState":{"people":1},"action":"insert into table executed"}
{"executionTime":10,"dbState":{"people":1},"action":"select from table executed"}
```


Figure 41 Content that was filled in general_logs.json

```
{"details":"new database was created: dbms","eventType":"database created","timestamp":"202
{"details":"database changed to: dbms","eventType":"database changed","timestamp":"2024-07-13T21:52:2
{"details":"new table was created","eventType":"table created","timestamp":"2024-07-13T21:53:20.74141
{"details":"a table was deleted","eventType":"table deleted","timestamp":"2024-07-13T21:54:39.991650"
```


Figure 42 Content that was filled in event_logs.json

Logging of Transactional queries

```
dbms_builder_11 > select * from people;
No rows found.

dbms_builder_11 > start transaction;
Started transaction

dbms_builder_11 > insert into people (id, name) values (1, "Kenny");
Row added successfully.

dbms_builder_11 > select * from people;
name  | id |
------+----+-
Kenny | 1  |

dbms_builder_11 > rollback;
Changes rolled back

dbms_builder_11 > select * from people;
No rows found.

dbms_builder_11 > drop table people;
Table dropped: people
```

Figure 43 Queries ran for transactional logging

```
{"query":"use dbms;","execution_time":37,"timestamp":"2024-07-13T22:04:36.892571"}
{"query":"create table people (id int primary key, name string);","execution_time":28,"timestamp":"
{"query":"select * from people;","execution_time":10,"timestamp":"2024-07-13T22:05:51.941747"}
{"query":"start transaction;","execution_time":7,"timestamp":"2024-07-13T22:06:02.911560"}
{"query":"insert into people (id, name) values (1, \"Kenny\");","execution_time":11,"timestamp":"20
{"query":"select * from people;","execution_time":11,"timestamp":"2024-07-13T22:06:41.294013"}
{"query":"rollback;","execution_time":3,"timestamp":"2024-07-13T22:06:50.421979"}
{"query":"select * from people;","execution_time":3,"timestamp":"2024-07-13T22:06:55.750194"}
{"query":"drop table people;","execution_time":1,"timestamp":"2024-07-13T22:07:03.363949"}
{"query":"exit","execution_time":0,"timestamp":"2024-07-13T22:07:17.964032"}
```


Figure 44 query_logs.json after running the transactional queries


```
{"executionTime":8,"dbState":{"people":0},"action":"select from table executed"}
{"executionTime":9,"dbState":{"people":1},"action":"insert into table executed"}
{"executionTime":10,"dbState":{"people":1},"action":"select from table executed"}
{"executionTime":2,"dbState":{"people":0},"action":"select from table executed"}
```


Figure 45 general_logs.json after running the transactional queries

You can notice the number of records in people table, which is 0 after rollback was performed, indicating reflection of transaction rollback in the number of records.


```
{"details":"database changed to: dbms","eventType":"database changed","timestamp":"2024-07-
{"details":"new table was created","eventType":"table created","timestamp":"2024-07-13T22:05:30.69
{"details":"A transaction is started","eventType":"Transaction status modified","timestamp":"2024-
{"details":"Changes were rolled back","eventType":"Rollback performed","timestamp":"2024-07-13T22:
{"details":"a table was deleted","eventType":"table deleted","timestamp":"2024-07-13T22:07:03.3626
```


Figure 46 event_logs.json after running the transactional queries

3.11. Export structure and value

Exporting an empty database

```
dbms_builder_11 > create database uppe;
Database created: uppe

dbms_builder_11 > use uppe;
Using database: uppe

dbms_builder_11 > exit

-------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
-------------------------------------
Select an option between 1 and 4: 2

Enter Database name: uppe
CREATE DATABASE uppe;
USE uppe;

Database exported.
```

Figure 47: Exporting an empty database.

Exporting a database which has empty tables

```
dbms_builder_11 > use uppe;
Using database: uppe

dbms_builder_11 > create table users (id int primary_key, name string);
Table created: users

dbms_builder_11 > create table courses (int course_id primary_key, course_name string);
Table created: courses

dbms_builder_11 > exit;

-------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
-------------------------------------
Select an option between 1 and 4: 2

Enter Database name: uppe
CREATE DATABASE uppe;
USE uppe;
CREATE TABLE courses (int course_id primary_key, course_name string );
CREATE TABLE users (id int primary_key, name string );

Database exported.
```

Figure 48: Exporting a database which has empty tables.

Exporting a database which has tables and rows in the tables

```
Welcome to TinyDb, please start writing queries below.

dbms_builder_11 > use uppe;
Using database: uppe

dbms_builder_11 > insert into users (id, name) values (1, "uppe");
Row added successfully.

dbms_builder_11 > insert into users (id, name) values (2, "shivani");
Row added successfully.

dbms_builder_11 > insert into courses (id, name) values (5408, "Data Management");
Row added successfully.

dbms_builder_11 > insert into courses (id, name) values (5308, "ASDC");
Row added successfully.

dbms_builder_11 > exit;
```

Figure 49: Inserting values into tables

```
-------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
-------------------------------------
Select an option between 1 and 4: 2

Enter Database name: uppe
CREATE DATABASE uppe;
USE uppe;
CREATE TABLE courses (int course_id primary_key, course_name string );
INSERT INTO courses (course_name, int) VALUES ('null', 'null');
INSERT INTO courses (course_name, int) VALUES ('null', 'null');
CREATE TABLE users (id int primary_key, name string );
INSERT INTO users (name, id) VALUES ('uppe', '1');
INSERT INTO users (name, id) VALUES ('shivani', '2');

Database exported.
-------------------------------------
```

Figure 50: Exporting a database which has tables and rows in the tables.

Exporting a database after updating a table

```
dbms_builder_11 > use uppe;
Using database: uppe

dbms_builder_11 > update users set name = "shiv" where id = 2;
1 row(s) affected.

dbms_builder_11 > exit;

-------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
-------------------------------------
Select an option between 1 and 4: 2

Enter Database name: uppe
CREATE DATABASE uppe;
USE uppe;
CREATE TABLE courses (int course_id primary_key, course_name string );
INSERT INTO courses (course_name, int) VALUES ('null', 'null');
INSERT INTO courses (course_name, int) VALUES ('null', 'null');
CREATE TABLE users (id int primary_key, name string );
INSERT INTO users (name, id) VALUES ('uppe', '1');
INSERT INTO users (name, id) VALUES ('shiv', '2');

Database exported
```

Figure 51: Exporting a database after updating a table.

Exporting a database which does not exist

```
-------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
-------------------------------------
Select an option between 1 and 4: 2

Enter Database name: shivani
Database not found! Please try again.
```

Figure 52: Exporting a database which does not exist.

3.12. User interface and Login security

Landing menu

User inputs wrong number for menu selection:

```
C:\Users\shahv\.jdks\openjdk-21.0.2\bin\java.exe "-
-------------------------------------
1. Register
2. Login
3. Exit
-------------------------------------
Select an option between 1 and 3: 4
Oops wrong input provided, please try again.
```

Figure 53: Landing menu when user inputs invalid number - 4

```
-------------------------------------
1. Register
2. Login
3. Exit
-------------------------------------
Select an option between 1 and 3: abcd
Oops wrong input provided, please try again.
```

Figure 54: Landing menu when user inputs invalid input "abcd"

User provides empty user id:

```
-------------------------------------
1. Register
2. Login
3. Exit
-------------------------------------
Select an option between 1 and 3: 1

Enter UserID:
UserID is empty, please try again.
```

Figure 55: Landing menu when user inputs empty user id

User tries to register with already registered user id:

```
-------------------------------------
1. Register
2. Login
3. Exit
-------------------------------------
Select an option between 1 and 3: 1

Enter UserID: 1
User is already registered
```

Figure 56: Landing menu when user inputs registered user id

User provides empty password:

```
-------------------------------------
1. Register
2. Login
3. Exit
-------------------------------------
Select an option between 1 and 3: 1

Enter UserID: 2
Enter Password:
Password is empty, please try again.
```

Figure 57: Landing menu when user inputs empty password

User provides empty security question:

```
-------------------------------------
1. Register
2. Login
3. Exit
-------------------------------------
Select an option between 1 and 3: 1

Enter UserID: 2
Enter Password: abcd
Enter Security Question:
Security question is empty, please try again.
```

Figure 58: Landing menu when user inputs empty security question

User provides empty security question's answer:

```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 1

Enter UserID: 2
Enter Password: abcd
Enter Security Question: question
Enter Security Answer:
Security answer is empty, please try again.
```

Figure 59: Landing menu when user inputs empty security question's answer

User provides valid credentials:

```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 1

Enter UserID: 2
Enter Password: vraj
Enter Security Question: question
Enter Security Answer: answer
User registered successfully!
```

Figure 60: Successful registration from landing menu

User_profile.txt file after registration

```
$2a$10$8qpLWuqqDBlOOKAUGwxdPOo8ilKBOXCViYBRcYolv0Z7mY1TvMUiK | $2a$10$YWBfUUMu3Dv6C.itKVKStezI8uFEwOOhirZd9VH3SURMOojuaYco2 | a | b
$2a$10$f6Db7jAOIOVACpHJEUfD3OZpNEVk5m7KSWcqYmoqz6MinXzHjeXay | $2a$10$dMzUp.JI7/Kx21UQ3ZrHKOXfl7/qPvHj3y5slk5Sk/3Rs6/2aFGlS | question | answer
```

Figure 61: User profile text file after successful registration

User tries to login with user id which is not registered:

```
-------------------------------------
1. Register
2. Login
3. Exit
-------------------------------------
Select an option between 1 and 3: 2

Enter UserID: 3
User not found! Please login with valid user ID.
```

Figure 62: Landing menu when user tries to login with unregistered user id

User provides wrong password during login:

```
-------------------------------------
1. Register
2. Login
3. Exit
-------------------------------------
Select an option between 1 and 3: 2

Enter UserID: 2
Enter Password: password
Invalid password! Please try again with valid password.
```

Figure 63: Landing menu when user tries to login with invalid password

User provides wrong answer to security question:

```
-------------------------------------
1. Register
2. Login
3. Exit
-------------------------------------
Select an option between 1 and 3: 2

Enter UserID: 2
Enter Password: vraj
Please answer this question: question
wrong answer
Security answer invalid! Please try again with valid security answer.
```

Figure 64: Landing menu when user tries to login with invalid security answer

User provides valid credentials and answer to login:

```
-------------------------------------
1. Register
2. Login
3. Exit
-------------------------------------
Select an option between 1 and 3: 2

Enter UserID: 2
Enter Password: vraj
Please answer this question: question
answer
-------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
-------------------------------------
Select an option between 1 and 4:
```

Figure 65: Successful login from landing menu

User selects exit option:

```
-------------------------------------
1. Register
2. Login
3. Exit
-------------------------------------
Select an option between 1 and 3: 3



Process finished with exit code 0
```

Figure 66: Landing menu after user selects Exit option

Main menu

User inputs wrong number for menu selection:

```
-------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
-------------------------------------
Select an option between 1 and 4: 5
Oops wrong input provided, please try again.
```

Figure 67: Main menu when user inputs invalid number - 5

```
-------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
-------------------------------------
Select an option between 1 and 4: abcd
Oops wrong input provided, please try again.
```

Figure 68: Main menu when user inputs invalid input "abcd"

Like landing menu, main menu selection will divert execution to individual functionalities i.e. 1 will start accepting queries, 2 will prompt user to input database to export, 3 will be implemented in upcoming sprint and 4 for exiting the application.

```
-------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
-------------------------------------
Select an option between 1 and 4: 1

Welcome to TinyDb, please start writing queries below.

dbms_builder_11 >
```

Figure 69: Main menu after user selects Write Queries option

```
--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4: 2

Enter Database name:
```

Figure 70: Main menu after user selects Export Structure and Value option

```
--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4: 3

Currently, generating ERD is not supported
```

Figure 71: Main menu after user selects ERD option

```
--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4: 4



Process finished with exit code 0
```

Figure 72: Main menu after user selects Exit option

3.13. ERD Generator

```
-------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
-------------------------------------
Select an option between 1 and 4: 3

Enter Database name: dbms
Database not found! Please try again.
```

Figure 73: Exporting database that is not created

```
-------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
-------------------------------------
Select an option between 1 and 4: 3

Enter Database name: test
ERD generated and saved to Databases/test_ERD.txt
```

Figure 74: Exporting database that is already created

test_ERD.txt

```
Entity-Relationship Diagram for Database: test

Table: service
Columns:
  - id (int)
  - user_id (int)
    Foreign Key -> user.id

Table: user
Columns:
  - id (int) primary_key, auto_increment
  - email (string) unique, non_null

Relationships and Cardinality:
Table: service
  - user_id (MANY) ->  (ONE) id
```

Figure 75: ERD text file after exporting