DALHOUSIE UNIVERSITY

Data Management, Warehousing and Analytics

Jaskaran Singh

MACS

B00948857

js356337@dal.ca

Assignment 2

Gitlab Repository link:

https://git.cs.dal.ca/singh16/csci5408_s23_b00948857_jaskaran_singh.git

PROBLEM 1

- **Summary:**

The paper introduces distributed database systems and focuses on the concept of data fragmentation. It explains that distributed systems are networks of computers that share essential characteristics, such as the Internet and various other networks. Distributed databases are logical databases spread physically across multiple locations connected by a data communications network. The paper discusses the design process of distributed databases, which includes initial design, redesign, and materialization of the redesign.

The related works section presents several research papers that have focused on fragmentation schemes and allocation strategies in distributed database systems. Each study proposes different algorithms and techniques to improve the reliability, availability, and performance of database systems.

Distributed database system design primary concern is to make fragmentation of classes in case of object-oriented databases, or the relations in case of relational database, replication and allocation of the fragments in different sites of the distributed system, and local optimization in each site. Fragmentation is a technique to split a single class or relation of a database into two or more partitions, also; the combination of the partitions supports the original database without any loss of information. The paper delves into the various types of fragmentations: horizontal fragmentation, vertical fragmentation, and mixed fragmentation.

- Horizontal fragmentation involves partitioning a relation into disjoint tuples or instances.
- Vertical fragmentation partitions a relation into separate sets of columns or attributes.
- Mixed fragmentation is a combination of horizontal and vertical strategies.

Advantages and disadvantages of fragmentation are discussed in the paper, highlighting its benefits in terms of reliability, performance, storage capacity, communication costs, and security.

Advantages:

- Efficiency increase as data is stored in fragment close to the site of usage.
- Local query optimization techniques are sufficient since data is locally available.
- Privacy and security can be easily maintained as irrelevant data is not available.

Disadvantages:

- When data from multiple fragments is required, latency is high.
- Recursive fragmentation can be expensive.
- Lack of back-up copies of data in fragments can corrupt the database.

The decision-making process for fragmentation is explained, considering quantitative and qualitative information.

The paper concludes by discussing correctness rules of fragmentation, including completeness, reconstruction, and disjointness. It provides a comparison of fragmentation strategies for each fragment of a relation and highlights the conditions under which each strategy is suitable. These rules are:

- Completeness: if each data item in Relation R into fragments can be found in Ri fragment.
- Reconstruction: Relation R should b reconstructable from individual Ri fragments.
- Disjointness: Data item, Di present in fragment Ri should not be visible in fragment Rj.

Improvements:

- The paper discusses that there are three phases of distributed database design phases, namely, initial design, redesign, and materialization of the design. The paper should include what factors should be considered in deciding these 3 phases.
- Also, in the initial design, paper has not discussed fragment and allocation algorithms in this stage.
- In the redesign phase, what factors should be considered for generating new fragments?

References:

A. H. Al-Sanhani, A. Hamdan, A. B. Al-Thaher and A. Al-Dahoud, "A comparative analysis of data fragmentation in distributed database," 2017 8th International Conference on Information Technology (ICIT), Amman, Jordan, 2017, pp. 724-729, doi: 10.1109/ICITECH.2017.8079934. Link: https://ieeexplore-ieee-org.ezproxy.library.dal.ca/document/8079934 (Available).

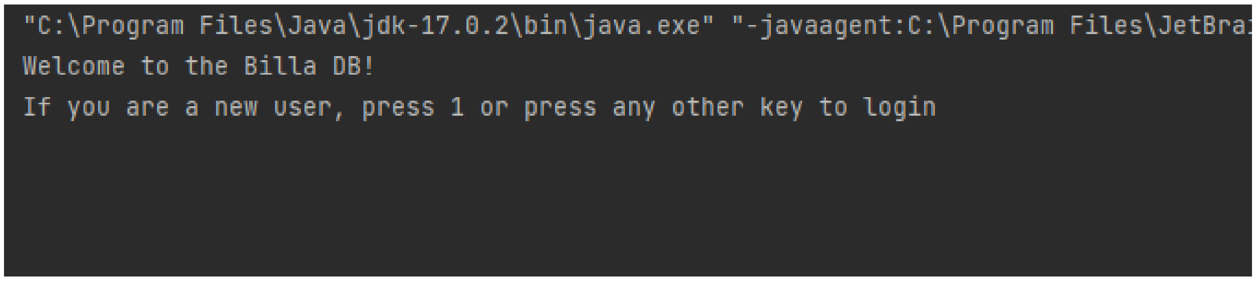
PROBLEM 2

BillaDB

Source code: The source code for this project is present in the gitlab repository (link mentioned on the first page) under problem2 folder.

- This project has been developed using JDK 17 and using a maven project.
- The code has provided required comments for different classes and methods in the project.
- The project has been developed considering SOLID principles. Mainly, there are 3 layers in the project architecture, repository, service, and views.
- Repository layers provide methods to store data in the database.
- Service layer provides methods for CRUD operations, user authentication, transaction managing and ERD of the database.
- In addition, there is a 'util' package which helps to encrypt and decrypt files and also password encoders for the project.
- Moreover, there is a model package which contain all model classes in the project.

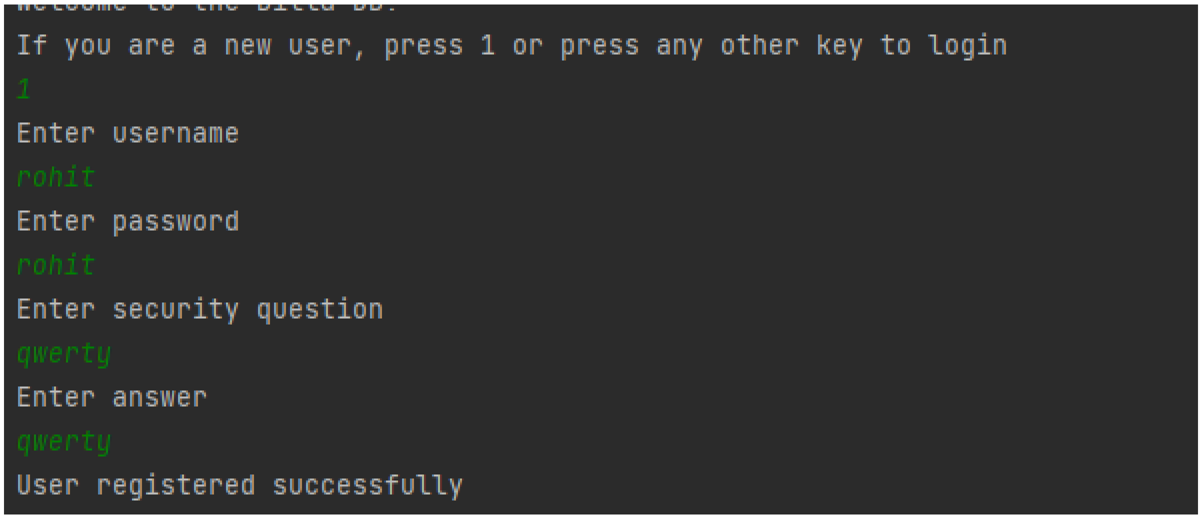
1. **Authentication:**



```
"C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBra
Welcome to the Billa DB!
If you are a new user, press 1 or press any other key to login
```

Fig 1: Home Screen of Billa DB

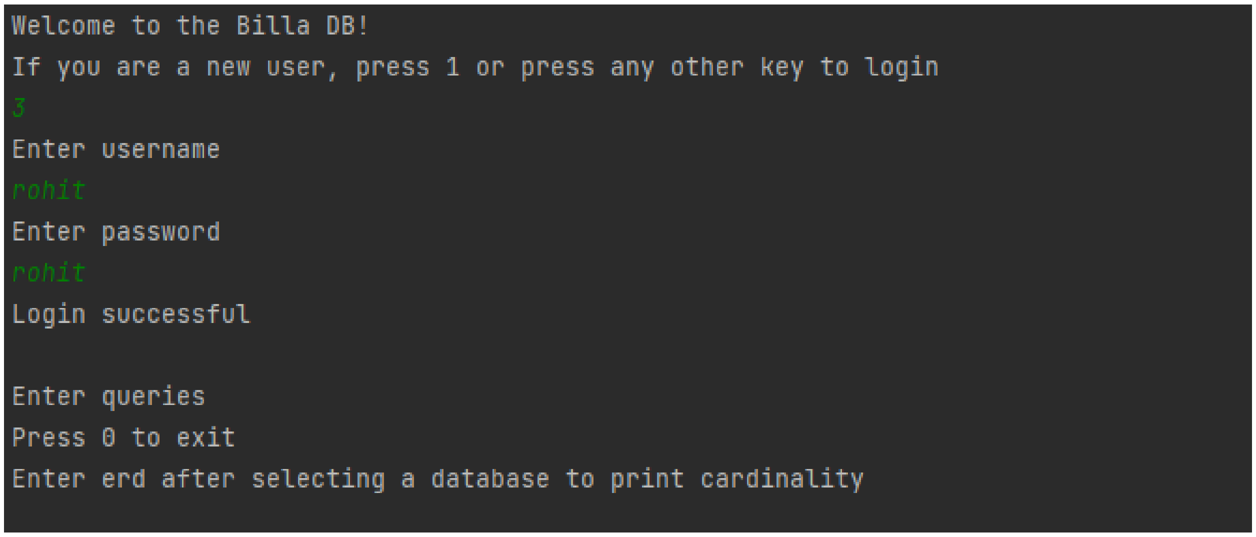
- For new users, press 1 to register.



```
If you are a new user, press 1 or press any other key to login
1
Enter username
rohit
Enter password
rohit
Enter security question
qwerty
Enter answer
qwerty
User registered successfully
```

Fig 2 User Registration

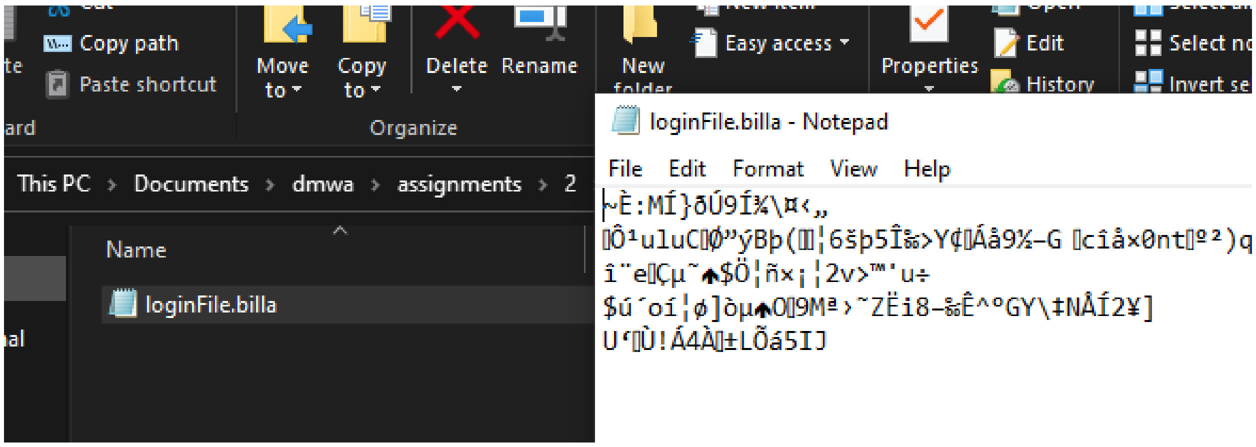
- Login



```
Welcome to the Billa DB!
If you are a new user, press 1 or press any other key to login
3
Enter username
rohit
Enter password
rohit
Login successful

Enter queries
Press 0 to exit
Enter erd after selecting a database to print cardinality
```

Fig 3 Login Successful screen

- All the users data is stored in User folder under target in 'loginFile.billa' in encrypted format for security reasons.
- For password encoding, MD5 library has been used and for file encryption, SHA-1 library has been used. All the related codebase is present in util package.

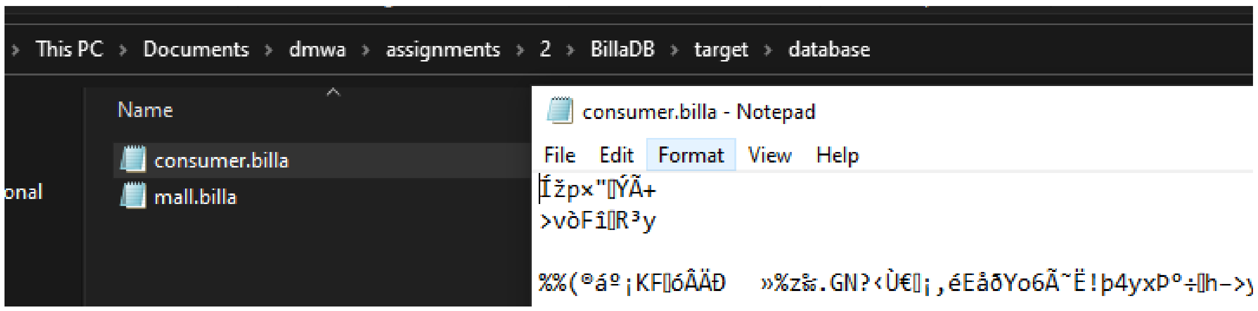


```
~È:MÍ}ðÚ9Î¾\¤‹„
Ô¹uluCØ"ýBþ(¦6šþ5Î‰>Y¢Áå9½–G cîå×0nt º²)q
î¨eÇµ˜$Ö¦ñ×¡¦2v>™'u÷
$ú´oí¦ø]òµ09Mª›˜ZËi8–‰Ê^°GY\‡NÅÍ2¥]
U'Ù!Á4À±LÕá5IJ
```

Fig 4 – User Authentication data in 'loginFile.billa'

2. Data encryption and storage

- All the data on file system is stored in encrypted format, be it users data or data related to database.
- '.billa' extension has been created for storing data and customized delimiters has been created for each model class, be it database, table, user or Role class.


This PC › Documents › dmwa › assignments › 2 › BillaDB › target › database
Name
consumer.billa
mall.billa
consumer.billa - Notepad
File Edit Format View Help

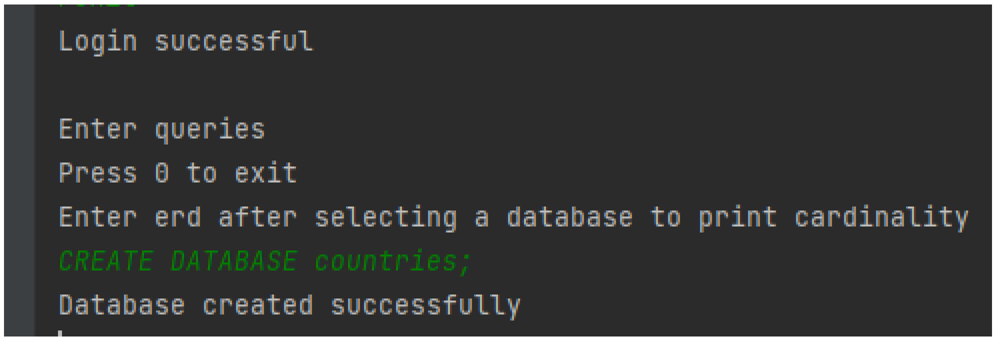
```
Ížpx"ÝÃ+
>vòFî R³y

%%(®áº¡KFóÂÄÐ    »%z‰.GN?‹Ù€¡,éEåðYo6Ã˜Ë!þ4yxÞ°÷h->
```

Fig 5 – Tables data

- All the data entered by the user in the form of sql queries is first stored in java objects (using model classes), and then using repository layer methods, it is stored in '.billa' files.
- Additionally, while fetching these stored data, firstly data is decrypted, then using mapping classes, it is converted into Java objects which can again be manipulated according to the user actions and then stored again in the file system.
- These mapping classes are present in "repository" folder.

3. DDL & DML

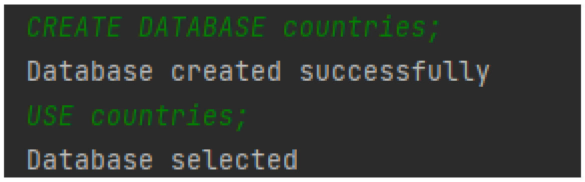
- **'CREATE DATABASE' statement:**



```
Login successful

Enter queries
Press 0 to exit
Enter erd after selecting a database to print cardinality
CREATE DATABASE countries;
Database created successfully
```

Fig 6: CREATE Statement

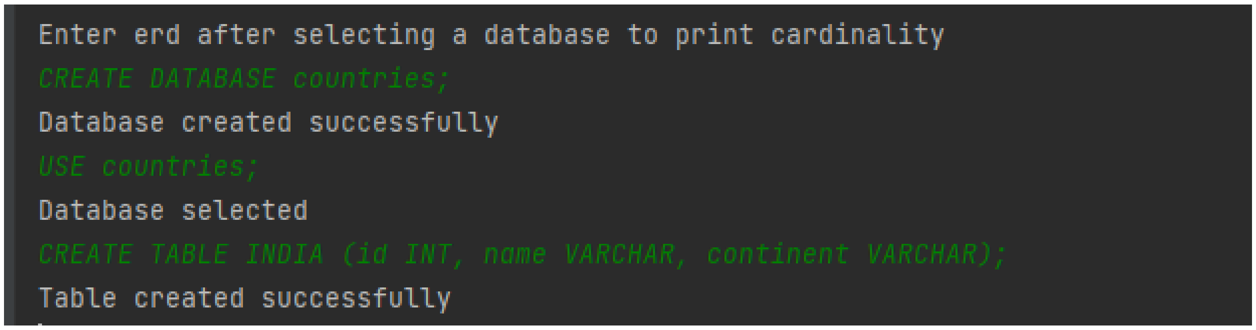
- **‘Use’ statement:**



```
CREATE DATABASE countries;
Database created successfully
USE countries;
Database selected
```

Fig 7: Use statement to select a database

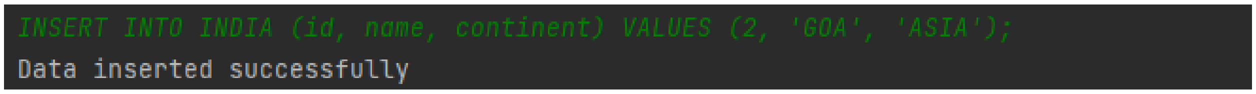
- **‘CREATE Table’:**



```
Enter erd after selecting a database to print cardinality
CREATE DATABASE countries;
Database created successfully
USE countries;
Database selected
CREATE TABLE INDIA (id INT, name VARCHAR, continent VARCHAR);
Table created successfully
```

Fig 8: “Create Table” query

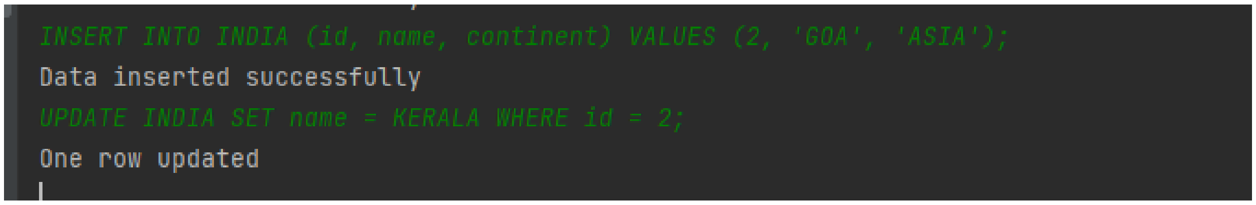
- **‘INSERT’ statement:**



```
INSERT INTO INDIA (id, name, continent) VALUES (2, 'GOA', 'ASIA');
Data inserted successfully
```

Fig 9: “INSERT” statement

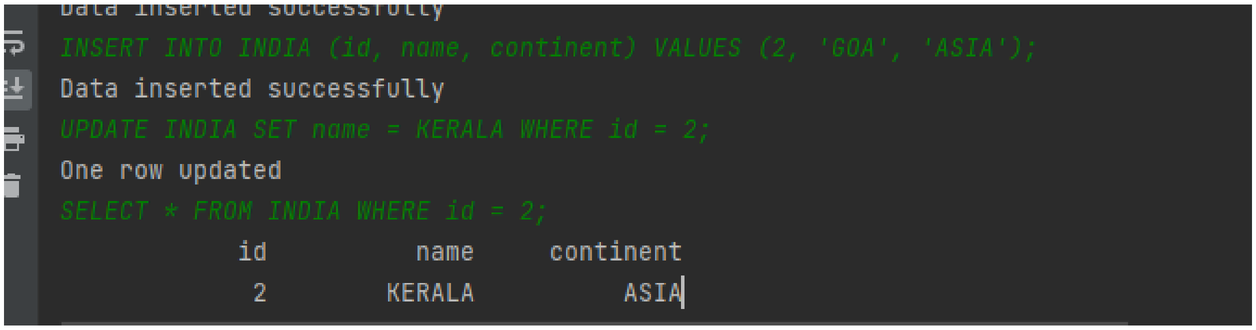
- **“UPDATE” statement:**



```
INSERT INTO INDIA (id, name, continent) VALUES (2, 'GOA', 'ASIA');
Data inserted successfully
UPDATE INDIA SET name = KERALA WHERE id = 2;
One row updated
```

Fig 10 : “UPDATE” statement

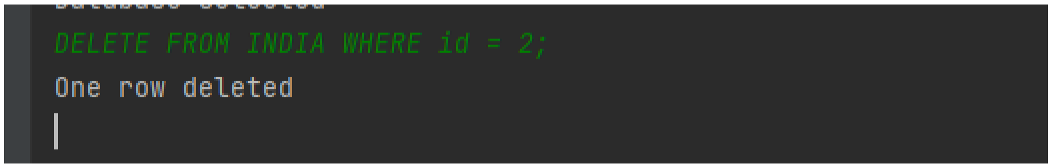
- **"SELECT" statement:**



```
INSERT INTO INDIA (id, name, continent) VALUES (2, 'GOA', 'ASIA');
Data inserted successfully
UPDATE INDIA SET name = KERALA WHERE id = 2;
One row updated
SELECT * FROM INDIA WHERE id = 2;
            id           name     continent
             2         KERALA          ASIA
```

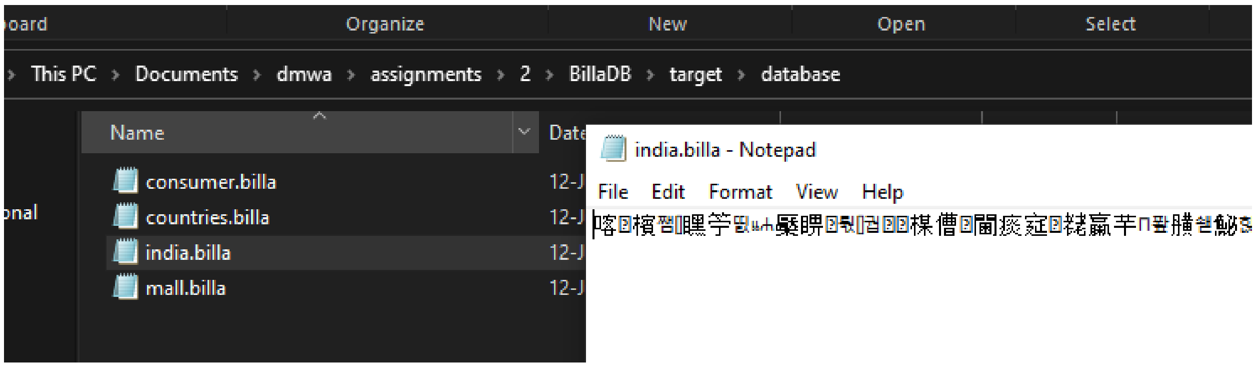
Fig 10 : "SELECT" statement

- **"DELETE" statement:**



```
DELETE FROM INDIA WHERE id = 2;
One row deleted
```

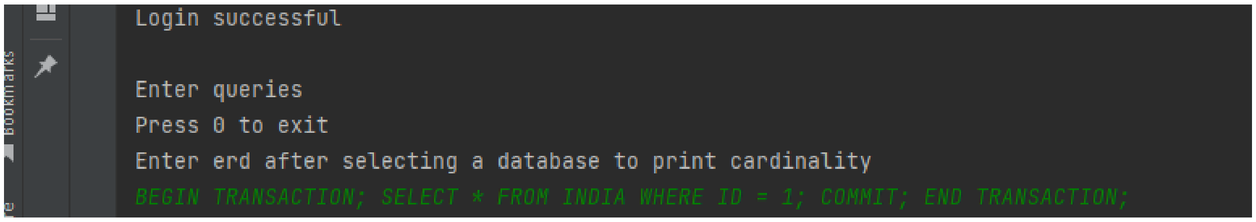
Fig 11: "DELETE" statement



Organize
New
Open
Select
This PC › Documents › dmwa › assignments › 2 › BillaDB › target › database
Name
Date
consumer.billa
countries.billa
india.billa
mall.billa
india.billa - Notepad
File Edit Format View Help

Fig 12: Table 'INDIA' and 'countries' database files

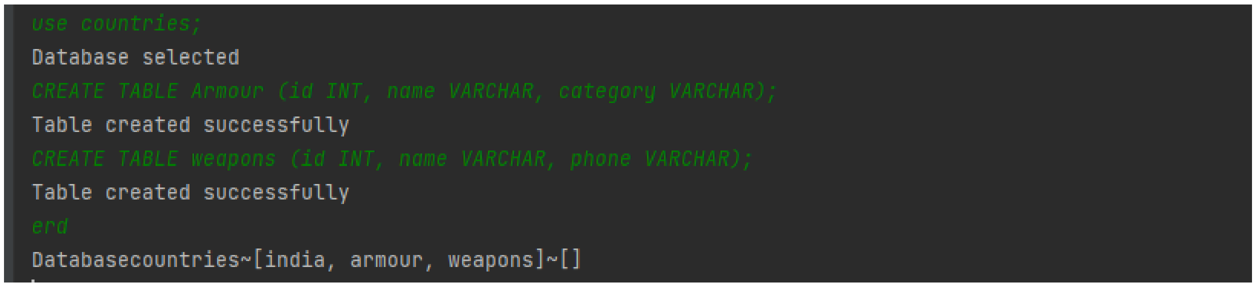
4. TRANSACTION MANAGEMENT



```
Login successful

Enter queries
Press 0 to exit
Enter erd after selecting a database to print cardinality
BEGIN TRANSACTION; SELECT * FROM INDIA WHERE ID = 1; COMMIT; END TRANSACTION;
```

Fig 13: Transactions

5. ERD



```
use countries;
Database selected
CREATE TABLE Armour (id INT, name VARCHAR, category VARCHAR);
Table created successfully
CREATE TABLE weapons (id INT, name VARCHAR, phone VARCHAR);
Table created successfully
erd
Databasecountries~[india, armour, weapons]~[]
```

Fig 14 ERD of database