NAME: Junaid Girkar
SAP ID: 60004190057
DIVISION: SE A (A3) COMPS

EXPERIMENT NO - 9

AIM:

Implement Transaction and Concurrency control techniques

THEORY:

A transaction is a logical unit of processing in a DBMS which entails one or more database access operations. In a nutshell, database transactions represent real-world events of any enterprise.
All types of database access operations which are held between the beginning and end transaction statements are considered as a single logical transaction. During the transaction the database is inconsistent. Only once the database is committed the state is changed from one consistent state to another.

Concurrency control is the procedure in DBMS for managing simultaneous operations without conflicting with each other. Concurrent access is quite easy if all users are just reading data. There is no way they can interfere with one another. Though for any practical database, would have a mix of reading and **WRITE** operations and hence the concurrency is a challenge.

Concurrency control is used to address such conflicts which mostly occur with a multi-user system. It helps you to make sure that database transactions are performed concurrently without violating the data integrity of respective databases.

Therefore, concurrency control is a most important element for the proper functioning of a system where two or multiple database transactions that require access to the same data, are executed simultaneously.

Concurrency Control Protocols
Different concurrency control protocols offer different benefits between the amount of concurrency they allow and the amount of overhead that they impose.

- Lock-Based Protocols

- Two Phase
- Timestamp-Based Protocols
- Validation-Based Protocols

Concurrency control using Mysql Locks:

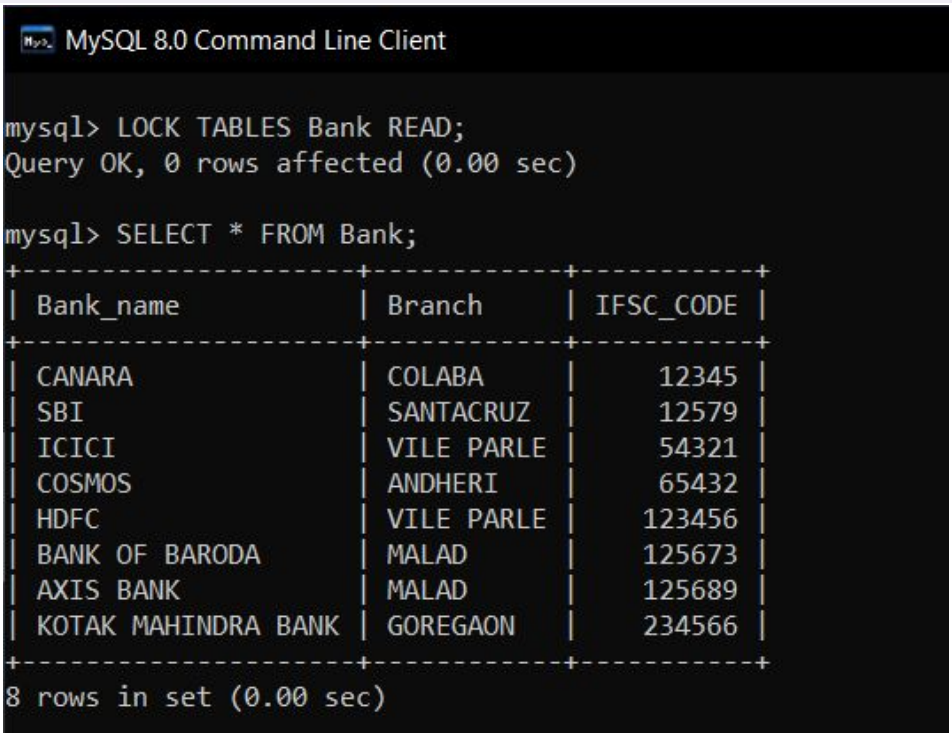
COMMANDS AND OUTPUT:

1) Read Lock by User-1

A) Reading by User-1

User-1:

```
LOCK TABLES Bank READ;
SELECT * FROM Bank;
```



B) WRITING BY USER-1

User-1:

```
LOCK TABLES Bank READ;
INSERT INTO Bank VALUES ('TEST', 'EARTH', 99999);
```

```
mysql> LOCK TABLES Bank READ;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Bank VALUES ('TEST', 'EARTH', 99999);
ERROR 1099 (HY000): Table 'Bank' was locked with a READ lock and can't be updated
mysql>
```

User-2:

```
SELECT * FROM Bank;
```

```
mysql> use credit;
Database changed
mysql> SELECT * FROM Bank;
+--------------------+------------+-----------+
| Bank_name          | Branch     | IFSC_CODE |
+--------------------+------------+-----------+
| CANARA             | COLABA     |     12345 |
| SBI                | SANTACRUZ  |     12579 |
| ICICI              | VILE PARLE |     54321 |
| COSMOS             | ANDHERI    |     65432 |
| HDFC               | VILE PARLE |    123456 |
| BANK OF BARODA     | MALAD      |    125673 |
| AXIS BANK          | MALAD      |    125689 |
| KOTAK MAHINDRA BANK | GOREGAON   |    234566 |
+--------------------+------------+-----------+
8 rows in set (0.00 sec)
```

User-1:

```
UNLOCK TABLES;
```

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

D) WRITING BY User-2

User-1:

```
LOCK TABLES Bank READ;
```

```
mysql> LOCK TABLES Bank READ;
Query OK, 0 rows affected (0.00 sec)
```

User-2:

```
INSERT INTO Bank VALUES ('TEST', 'EARTH', 99999);
```

```
mysql> INSERT INTO Bank VALUES ('TEST', 'EARTH', 99999);
```

User-1:

```
UNLOCK TABLES;
```

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> INSERT INTO Bank VALUES ('TEST', 'EARTH', 99999);
Query OK, 1 row affected (1 min 13.50 sec)
```

TAKE – 2

User-1:

```
LOCK TABLES Bank READ;
```

User-2:

```
INSERT INTO Bank VALUES ('Test3', 'Gringotts', 99994);
```

User-1:

```
mysql> LOCK TABLES Bank READ;
Query OK, 0 rows affected (0.00 sec)

mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

User-2:

```
mysql> INSERT INTO Bank VALUES ('Test3', 'Gringotts', 99994);
Query OK, 1 row affected (11.93 sec)
```

2) WRITE LOCK BY User-1

A) READING BY User-1

User-1:

```
LOCK TABLES Bank WRITE;
SELECT * FROM Bank;
```

```
mysql> LOCK TABLES Bank WRITE;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM Bank;
+--------------------+------------+------------+
| Bank_name          | Branch     | IFSC_CODE  |
+--------------------+------------+------------+
| CANARA             | COLABA     |      12345 |
| SBI                | SANTACRUZ  |      12579 |
| ICICI              | VILE PARLE |      54321 |
| COSMOS             | ANDHERI    |      65432 |
| Test3              | Gringotts  |      99994 |
| TEST2              | EARTH      |      99998 |
| TEST               | EARTH      |      99999 |
| HDFC               | VILE PARLE |     123456 |
| BANK OF BARODA     | MALAD      |     125673 |
| AXIS BANK          | MALAD      |     125689 |
| KOTAK MAHINDRA BANK | GOREGAON   |     234566 |
+--------------------+------------+------------+
11 rows in set (0.00 sec)
```

B) WRITING BY User-1

User-1:

```
LOCK TABLES Bank WRITE;
INSERT INTO Bank VALUES ('TEST4', 'GRINGOTTS', 99993);
```

```
mysql> LOCK TABLES Bank WRITE;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Bank VALUES ('TEST4', 'GRINGOTTS', 99993);
Query OK, 1 row affected (0.01 sec)
```

C) READING BY User-2

User-1:

```
LOCK TABLES Bank WRITE;
```

```
mysql> LOCK TABLES Bank WRITE;
Query OK, 0 rows affected (0.00 sec)
```

User-2:

```
SELECT * FROM Bank;
```

```
mysql> SELECT * FROM Bank;
```

User-1:

```
UNLOCK TABLES;
```

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT * FROM Bank;
+--------------------+------------+-----------+
| Bank_name          | Branch     | IFSC_CODE |
+--------------------+------------+-----------+
| CANARA             | COLABA     |     12345 |
| SBI                | SANTACRUZ  |     12579 |
| ICICI              | VILE PARLE |     54321 |
| COSMOS             | ANDHERI    |     65432 |
| TEST4              | GRINGOTTS  |     99993 |
| Test3              | Gringotts  |     99994 |
| TEST2              | EARTH      |     99998 |
| TEST               | EARTH      |     99999 |
| HDFC               | VILE PARLE |    123456 |
| BANK OF BARODA     | MALAD      |    125673 |
| AXIS BANK          | MALAD      |    125689 |
| KOTAK MAHINDRA BANK | GOREGAON   |    234566 |
+--------------------+------------+-----------+
12 rows in set (32.67 sec)
```

D) WRITING BY User-2:

User-1:

```
LOCK TABLE Bank WRITE;
```

```
mysql> LOCK TABLE Bank WRITE;
Query OK, 0 rows affected (0.00 sec)
```

User-2:

```
INSERT INTO Bank VALUES ('TEST5', 'MILKY WAY', 99991);
```

```
mysql> INSERT INTO Bank VALUES ('TEST5', 'MILKY WAY', 99991);
```

User-1:

```
UNLOCK TABLES;
```

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> INSERT INTO Bank VALUES ('TEST5', 'MILKY WAY', 99991);
Query OK, 1 row affected (39.02 sec)
```

```
mysql> SELECT * FROM Bank;
+--------------------+------------+-----------+
| Bank_name          | Branch     | IFSC_CODE |
+--------------------+------------+-----------+
| CANARA             | COLABA     |     12345 |
| SBI                | SANTACRUZ  |     12579 |
| ICICI              | VILE PARLE |     54321 |
| COSMOS             | ANDHERI    |     65432 |
| TEST5              | MILKY WAY  |     99991 |
| TEST4              | GRINGOTTS  |     99993 |
| Test3              | Gringotts  |     99994 |
| TEST2              | EARTH      |     99998 |
| TEST               | EARTH      |     99999 |
| HDFC               | VILE PARLE |    123456 |
| BANK OF BARODA     | MALAD      |    125673 |
| AXIS BANK          | MALAD      |    125689 |
| KOTAK MAHINDRA BANK | GOREGAON   |    234566 |
+--------------------+------------+-----------+
13 rows in set (0.00 sec)
```

CONCLUSION:

Thus we have performed experiments on using various types of locks on a table in MySQL Command Line Client and how it is used to demonstrate data concurrency.