TRANSACTIONS AND LOCKING

A transaction is a unit of work that is performed against a database. Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program.

A transaction is the propagation of one or more changes to the database. For example, if you are creating a record or updating a record or deleting a record from the table, then you are performing a transaction on that table. It is important to control these transactions to ensure the data integrity and to handle database errors.

Practically, you will club many SQL queries into a group and you will execute all of them together as a part of a transaction.

Properties of Transactions

Transactions have the following four standard properties, usually referred to by the acronym **ACID.**

- **Atomicity:** ensures that all operations within the work unit are completed successfully. Otherwise, the transaction is aborted at the point of failure and all the previous operations are rolled back to their former state.
- **Consistency:** ensures that the database properly changes states upon a successfully committed transaction.
- **Isolation:** enables transactions to operate independently of and transparent to each other.
- **Durability:** ensures that the result or effect of a committed transaction persists in case of a system failure.

Transaction Control

The following commands are used to control transactions.

- **COMMIT:** to save the changes.
- **ROLLBACK:** to roll back the changes.
- **SAVEPOINT:** creates points within the groups of transactions in which to ROLLBACK.
- **SET TRANSACTION:** Places a name on a transaction.

Transactional Control Commands

Transactional control commands are only used with the **DML Commands** such as -INSERT, UPDATE and DELETE only. They cannot be used while creating tables or dropping them because these operations are automatically committed in the database.

The COMMIT Command

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database. The COMMIT command saves all the transactions to the database since the last COMMIT or ROLLBACK command.

The syntax for the COMMIT command is as follows.

```
COMMIT;
```

Example

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is an example which would delete those records from the table which have age = 25 and then COMMIT the changes in the database.

```
mysql> DELETE FROM CUSTOMERS
       WHERE AGE = 25;
mysql> COMMIT;
```

Thus, two rows from the table would be deleted and the SELECT statement would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
3	Kaushik	23	Kota	2000.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The ROLLBACK Command

The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database. This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

The syntax for a ROLLBACK command is as follows:

```
ROLLBACK;
```

Example

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is an example, which would delete those records from the table which have the age = 25 and then ROLLBACK the changes in the database.

```
mysql> DELETE FROM CUSTOMERS
       WHERE AGE = 25;

mysql > ROLLBACK;
```

Thus, the delete operation would not impact the table and the SELECT statement would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The SAVEPOINT Command

A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.

The Syntax of a SAVEPOINT command is as shown below.

```
SAVEPOINT SAVEPOINT_NAME;
```

This command serves only in the creation of a SAVEPOINT among all the transactional statements. The ROLLBACK command is used to undo a group of transactions.

The syntax for rolling back to a SAVEPOINT is as shown below.

```
ROLLBACK TO SAVEPOINT_NAME;
```

Following is an example where you plan to delete the three different records from the CUSTOMERS table. You want to create a SAVEPOINT before each delete, so that you can ROLLBACK to any SAVEPOINT at any time to return the appropriate data to its original state.

Example

Consider the CUSTOMERS table having the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following code block contains the series of operations.

```
mysql> SAVEPOINT  SP1;
Savepoint  created.
mysql> DELETE  FROM
CUSTOMERS  WHERE ID=1;
 1 row deleted.
mysql > SAVEPOINT SP2;
Savepoint created.
mysql> DELETE  FROM
CUSTOMERS WHERE ID=2;
1 row deleted.
mysql> SAVEPOINT SP3;
Savepoint created.
mysql> DELETE  FROM
CUSTOMERS  WHERE ID=3;
1 row deleted.
```

Now that the three deletions have taken place, let us assume that you have changed your mind and decided to ROLLBACK to the SAVEPOINT that you identified as SP2. Because SP2 was created after the first deletion, the last two deletions are undone:

```
mysql> ROLLBACK TO
SP2;

Rollback complete.
```

Notice that only the first deletion took place since you rolled back to SP2.

```
mysql> Select * from
CUSTOMERS;
```

ID	NAME	AGE	ADDRESS	SALARY
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00

5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00
6 rows selected				

The RELEASE SAVEPOINT Command

The RELEASE SAVEPOINT command is used to remove a SAVEPOINT that you have created.

The syntax for a RELEASE SAVEPOINT command is as follows.

```
RELEASE  SAVEPOINT  SAVEPOINT_NAME;
```

Once a SAVEPOINT has been released, you can no longer use the ROLLBACK command to undo transactions performed since the last SAVEPOINT.

The SET TRANSACTION Command

The SET TRANSACTION command can be used to initiate a database transaction. This command is used to specify characteristics for the transaction that follows. For example, you can specify a transaction to be read only or read write.

The syntax for a SET TRANSACTION command is as follows.

```
SET TRANSACTION [ READ WRITE | READ ONLY ];
```

Table Locking

Locking is important in many scenarios to prevent other sessions from modifying tables during periods when a session requires exclusive access to them. for example altering table definition online or any kind of table definition changes. Mysql provides an option to lock table/s with different types of locks, depends on need.

The Syntax for locking is shown as below:

```
LOCK TABLES
    tbl_name [[AS] alias] lock_type
    [, tbl_name [[AS] alias] lock_type] ...

lock_type:
    READ [LOCAL]
  | [LOW_PRIORITY] WRITE

UNLOCK TABLES
```

Consider the CUSTOMERS table having the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Now let's lock the table with a read lock:

```
mysql> lock table CUSTOMERS read;
Query OK, 0 rows affected (0.00 sec)
```

As we try to insert the a new record into the CUSTOMERS table

```
session1> select count(*) from t1;

count(*)
-----------
7
```

```
mysql> insert into CUSTOMERS values( 10, 'Rakesh',28,Patna,15000.00);
ERROR 1099 (HY000): Table 'CUSTOMERS' was locked with a READ
 Lock and can't be updated
```

The statements acquired READ lock on table CUSTOMERS explicitly. After applying READ lock on table users can read the table but not write it.

Unlocking of table can be shown in the syntax below:

```
mysql>UNLOCK TABLES;
```

Subsequently trying to insert the record will result in successful insertion of record in table CUSTOMERS.

```
mysql> insert into CUSTOMERS values( 10, 'Rakesh',28,Patna,15000.00);
mysql> select * from CUSTOMERS;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00
10	Rakesh	28	Patna	15000.00