### 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;

### 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;

### 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 for 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;

**Commit, Rollback and Savepoint SQL commands**

Transaction Control Language(TCL) commands are used to manage transactions in the [database](https://www.studytonight.com/dbms/overview-of-dbms.php).

## COMMIT command

COMMIT command is used to permanently save any transaction into the database.

When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed, the changes made by these commands can be rolled back.

To avoid that, we use the COMMIT command to mark the changes as permanent.

**COMMIT;**

## ROLLBACK command

This command restores the database to last commited state. It is also used with SAVEPOINT command to jump to a savepoint in an ongoing transaction.

If we have used the UPDATE command to make some changes into the database, and realise that those changes were not required, then we can use the ROLLBACK command to rollback those changes, if they were not commited using the COMMIT command.

## ROLLBACK ;

**ROLLBACK TO savepoint\_name;**

## SAVEPOINT command

SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

**SAVEPOINT savepoint\_name;**

mysql> create table customers12( id int not null, name varchar (20) not null, age int not null, address char (25) , salary decimal (18, 2), primary key (id));

Query OK, 0 rows affected (0.09 sec)

mysql> insert into customers (id,name,age,address,salary) values (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),(7,'muffy',24,'indore',10000);

Query OK, 5 rows affected (0.05 sec)

Records: 5 Duplicates: 0 Warnings: 0

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 |

+----+----------+-----+-----------+---------+

5 rows in set (0.00 sec)

mysql> start transaction;

Query OK, 0 rows affected (0.00 sec)

mysql> insert into customers12 (id,name,age,address,salary) values (6,'komal',22,'mp',4500),(7,'muffy',24,'indore',10000);

Query OK, 2 rows affected (0.00 sec)

Records: 2 Duplicates: 0 Warnings: 0

mysql> commit;

Query OK, 0 rows affected (0.05 sec)

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 |

+----+----------+-----+-----------+----------+

7 rows in set (0.00 sec)

mysql> start transaction;

Query OK, 0 rows affected (0.00 sec)

mysql> delete from customers;

Query OK, 7 rows affected (0.00 sec)

mysql> select \* from customers;

Empty set (0.00 sec)

mysql> rollback;

Query OK, 0 rows affected (0.04 sec)

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 |

+----+----------+-----+-----------+----------+

7 rows in set (0.00 sec)

mysql> savepoint sp1;

Query OK, 0 rows affected (0.00 sec)

mysql> insert into customers (id,name,age,address,salary) values (8,'khilanil',28,'agra',2500.00);

Query OK, 1 row affected (0.00 sec)

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 |

| 8 | khilanil | 28 | agra | 2500.00 |

+----+----------+-----+-----------+----------+

8 rows in set (0.00 sec)

mysql> rollback to savepoint sp1;

Query OK, 0 rows affected (0.00 sec)

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 |

+----+----------+-----+-----------+----------+

7 rows in set (0.00 sec)

mysql> insert into customers (id,name,age,address,salary) values (8,'khilanil',28,'agra',2500.00);

Query OK, 1 row affected (0.00 sec)

mysql> commit;

Query OK, 0 rows affected (0.08 sec)

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 |

| 8 | khilanil | 28 | agra | 2500.00 |

+----+----------+-----+-----------+----------+

8 rows in set (0.00 sec)

mysql> start transaction;

Query OK, 0 rows affected (0.00 sec)

mysql> savepoint sp2;

Query OK, 0 rows affected (0.00 sec)

mysql> update customers set name='ramu' where id=1;

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> select \* from customers;

+----+----------+-----+-----------+----------+

| id | name | age | address | salary |

+----+----------+-----+-----------+----------+

| 1 | ramu | 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 |

+----+----------+-----+-----------+----------+

7 rows in set (0.00 sec)

mysql> release savepoint sp2;

Query OK, 0 rows affected (0.00 sec)

mysql> commit;

Query OK, 0 rows affected (0.14 sec)

mysql> select \* from customers;

+----+----------+-----+-----------+----------+

| id | name | age | address | salary |

+----+----------+-----+-----------+----------+

| 1 | ramu | 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 |

+----+----------+-----+-----------+----------+

7 rows in set (0.00 sec)

## What is a Stored Procedure?

A stored procedure is a procedure (like a subprogram in a regular computing language) that is stored (in the database). Correctly speaking, MySQL supports "routines" and there are two kinds of routines: stored procedures which you call, or functions whose return values you use in other SQL statements the same way that you use pre-installed MySQL functions like pi(). I'll use the word "stored procedures" more frequently than "routines" because it's what we've used in the past, and what people expect us to use.

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

mysql> create procedure p1(c\_age int) select id,name,age,salary from customers where age>c\_age;

Query OK, 0 rows affected (0.13 sec)

mysql> call p1(25);

+----+--------+-----+---------+

| id | name | age | salary |

+----+--------+-----+---------+

| 1 | ramesh | 32 | 2000.00 |

| 5 | hardik | 27 | 8500.00 |

+----+--------+-----+---------+

2 rows in set (0.08 sec)

Query OK, 0 rows affected (0.09 sec)

# What is Stored Procedures in SQL ?

**Stored Procedures** are created to perform one or more DML operations on Database. It is nothing but the group of SQL statements that accepts some input in the form of parameters and performs some task and may or may not returns a value.

**Syntax :** Creating a Procedure 

CREATE or REPLACE PROCEDURE name(parameters)

IS

variables;

BEGIN

//statements;

END;

The most important part is parameters. Parameters are used to pass values to the Procedure. There are 3 different types of parameters, they are as follows: 

1. **IN:**   
   This is the Default Parameter for the procedure. It always receives the values from calling program.
2. **OUT:**   
   This parameter always sends the values to the calling program.
3. **IN OUT:**   
   This parameter performs both the operations. It Receives value from as well as sends the values to the calling program.

**Example:**   
  
Imagine a table named with emp\_table stored in Database. We are Writing a Procedure to update a Salary of Employee with 1000. 

CREATE or REPLACE PROCEDURE INC\_SAL(eno IN NUMBER, up\_sal OUT NUMBER)

IS

BEGIN

UPDATE emp\_table SET salary = salary+1000 WHERE emp\_no = eno;

COMMIT;

SELECT sal INTO up\_sal FROM emp\_table WHERE emp\_no = eno;

END;

* Declare a Variable to Store the value coming out from Procedure :

VARIABLE v NUMBER;

* Execution of the Procedure:

EXECUTE INC\_SAL(1002, :v);

* To check the updated salary use SELECT statement:

SELECT \* FROM emp\_table WHERE emp\_no = 1002;

* or Use print statement :

PRINT :v

**Data control language (DCL)**

Data control language (DCL) is used to access the stored data. It is mainly used for revoke and to grant the user the required access to a database. In the database, this language does not have the feature of rollback.

* It is a part of the structured query language (SQL).
* It helps in controlling access to information stored in a database. It complements the data manipulation language (DML) and the data definition language (DDL).
* It is the simplest among three commands.
* It provides the administrators, to remove and set database permissions to desired users as needed.
* These commands are employed to grant, remove and deny permissions to users for retrieving and manipulating a database.

## DDL Commands

The Data Definition Language (DDL) commands are as follows −

### GRANT Command

It is employed to grant a privilege to a user. GRANT command allows specified users to perform specified tasks

**Syntax**

GRANT privilege\_name on objectname to user;

Here,

* privilege names are SELECT,UPDATE,DELETE,INSERT,ALTER,ALL
* objectname is table name
* user is the name of the user to whom we grant privileges

### REVOKE Command

It is employed to remove a privilege from a user. REVOKE helps the owner to cancel previously granted permissions.

**Syntax**

 REVOKE privilege\_name on objectname from user;

Here,

* privilege names are SELECT,UPDATE,DELETE,INSERT,ALTER,ALL
* objectname is table name
* user is the name of the user whose privileges are removing

### Example

GRANT SELECT, UPDATE ON employees TO Bhanu

Explanation − Firstly, to give the permissions to user, we have to use GRANT command. The privileges are SELECT because to view the records and UPDATE to modify the records. The objectname is table name which is Employee. The user name is bhanu.

REVOKE SELECT, UPDATE ON employees TO Bhanu

Explanation − Firstly, to revoke the permissions to user, we have to use REVOKE command. The privileges Need to revoke are SELECT because to view the records and UPDATE to modify the records. The objectname is table name which is Employee. The user name is Bhanu.