SQL commands

- 4 major categories:

  DDL- Data definition language

  DML- Data manipulation language

  DCL – Data control language

  TCL – Transaction control language

DDL Commands:

- CREATE
- DROP
- ALTER
- RENAME
- TRUNCATE

DML Commands:

- SELECT
- INSERT
- UPDATE
- DELETE

DCL Commands

- GRANT
- REVOKE

TCL Commands:

- COMMIT
- ROLLBACK
- SAVEPOINT
- SET TRANSACTION

DDL Commands

DDL : Data Definition Language

All DDL commands are auto-committed. That means it saves all the changes permanently in the database.

Command	Description
create	to create new table or database
alter	for alteration
truncate	delete data from table
drop	to drop a table
rename	to rename a table

DML CommandS

DML : Data Manipulation Language

DML commands are not auto-committed. It means changes are not permanent to database, they can be rolled back.

Command	Description
insert	to insert a new row
update	to update existing row
delete	to delete a row
merge	merging two rows or two tables

TCL Commands

TCL : Transaction Control Language

These commands are to keep a check on other commands and their affect on the database. These commands can annul changes made by other commands by rolling back to original state. It can also make changes permanent.

Command	Description
commit	to permanently save
rollback	to undo change
savepoint	to save temporarily

DCL Command

DCL : Data Control Language

Data control language provides command to grant and take back authority.

Command	Description
grant	grant permission of right
revoke	take back permission.

DQL Command

DQL : Data Query Language

Command	Description
select	retrieve records from one or more table

DDL COMMANDS

Create Command

Creating a Database

To create a database in RDBMS, *create* command is uses. Following is the Syntax,

```
create database database-name;
```

Example for Creating Database

```
create database Test;
```

The above command will create a database named **Test**.

Create Command

Creating a Table

create command is also used to create a table. We can specify names and datatypes of various columns along.Following is the Syntax,

```
create table table-name
(
 column-name1 datatype1,
 column-name2 datatype2,
 column-name3 datatype3,
 column-name4 datatype4
);
```

create table command will tell the database system to create a new table with given table name and column information.

Example for creating Table

```
create table Student(id int, name varchar, age int);
```

The above command will create a new table **Student** in database system with 3 columns, namely id, name and age.

Alter Command

alter command

alter command is used for alteration of table structures. There are various uses of *alter* command, such as,

- to add a column to existing table
- to rename any existing column
- to change datatype of any column or to modify its size.
- *alter* is also used to drop a column.

To Add Column to existing Table

Using alter command we can add a column to an existing table. Following is the Syntax,

```
alter table table-name add(column-name datatype);
```

Here is an Example for this,

```
alter table Student add(address char);
```

The above command will add a new column *address* to the **Student** table

Alter Command

To Add Multiple Column to existing Table

Using alter command we can even add multiple columns to an existing table. Following is the Syntax,

```
alter table table-name add(column-name1 datatype1, column-name2 datatype2, column-name3
datatype3);
```

Here is an Example for this,

```
alter table Student add(father-name varchar(60), mother-name varchar(60), dob date);
```

The above command will add three new columns to the **Student** table

Alter Command

To Add column with Default Value

alter command can add a new column to an existing table with default values. Following is the Syntax,

```
alter table table-name add(column-name1 datatype1 default data);
```

Here is an Example for this,

```
alter table Student add(dob date default '1-Jan-99');
```

The above command will add a new column with default value to the **Student** table

Alter Command

To Modify an existing Column

alter command is used to modify data type of an existing column . Following is the Syntax,

```
alter table table-name modify(column-name datatype);
```

Here is an Example for this,

```
alter table Student modify(address varchar(30));
```

The above command will modify *address* column of the **Student table**

Alter Command

To Rename a column

Using alter command you can rename an existing column. Following is the Syntax,

```
alter table table-name rename old-column-name to column-name;
```

Here is an Example for this,

```
alter table Student rename address to Location;
```

The above command will rename *address* column to *Location*.

Alter Command

To Drop a Column

alter command is also used to drop columns also. Following is the Syntax,

```
alter table table-name drop(column-name);
```

Here is an Example for this,

```
alter table Student drop(address);
```

The above command will drop *address* column from the **Student table**

SQL queries to Truncate, Drop or Rename a Table

truncate command

truncate command removes all records from a table. But this command will not destroy the table's structure. When we apply truncate command on a table its Primary key is initialized. Following is its Syntax,

```
truncate table table-name
```

Here is an Example explaining it.

```
truncate table Student;
```

The above query will delete all the records of **Student** table.

truncate command is different from **delete** command. delete command will delete all the rows from a table whereas truncate command re-initializes a table(like a newly created table).

For eg. If you have a table with 10 rows and an auto_increment primary key, if you use *delete* command to delete all the rows, it will delete all the rows, but will not initialize the primary key, hence if you will insert any row after using delete command, the auto_increment primary key will start from 11. But in case of *truncate* command, primary key is re-initialized.

SQL queries to Truncate, Drop or Rename a Table

drop command

drop query completely removes a table from database. This command will also destroy the table structure. Following is its Syntax,

```
drop table table-name
```

Here is an Example explaining it.

```
drop table Student;
```

The above query will delete the **Student** table completely. It can also be used on Databases. For Example, to drop a database,

```
drop database Test;
```

The above query will drop a database named **Test** from the system.

SQL queries to Truncate, Drop or Rename a Table

drop command

drop query completely removes a table from database. This command will also destroy the table structure. Following is its Syntax,

```
drop table table-name
```

Here is an Example explaining it.

```
drop table Student;
```

The above query will delete the **Student** table completely. It can also be used on Databases. For Example, to drop a database,

```
drop database Test;
```

The above query will drop a database named **Test** from the system.

SQL queries to Truncate, Drop or Rename a Table

rename query

rename command is used to rename a table. Following is its Syntax,

```
rename table old-table-name to new-table-name
```

Here is an Example explaining it.

```
rename table Student to Student-record;
```

The above query will rename **Student** table to **Student-record**.

DATABASE CONSTRAINTS

SQL Constraints

SQl Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside table.

Constraints can be divided into following two types,

- **Column level constraints** : limits only column data
- **Table level constraints :** limits whole table data

Constraints are used to make sure that the integrity of data is maintained in the database. Following are the most used constraints that can be applied to a table.

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

SQL Constraints

NOT NULL Constraint

NOT NULL constraint restricts a column from having a NULL value. Once **NOT NULL** constraint is applied to a column, you cannot pass a null value to that column. It enforces a column to contain a proper value. One important point to note about NOT NULL constraint is that it cannot be defined at table level.

Example using NOT NULL constraint

```
CREATE table Student(s_id int NOT NULL, Name varchar(60), Age int);
```

The above query will declare that the **s_id** field of **Student** table will not take NULL value.

SQL Constraints

UNIQUE Constraint

UNIQUE constraint ensures that a field or column will only have unique values. A UNIQUE constraint field will not have duplicate data. UNIQUE constraint can be applied at column level or table level.

Example using UNIQUE constraint when creating a Table (Table Level)

```
CREATE table Student(s_id int NOT NULL UNIQUE, Name varchar(60), Age int);
```

The above query will declare that the **s_id** field of **Student** table will only have unique values and wont take NULL value.

Example using UNIQUE constraint after Table is created (Column Level)

```
ALTER table Student add UNIQUE(s_id);
```

The above query specifies that **s_id** field of **Student** table will only have unique value.

SQL Constraints

Primary Key Constraint

Primary key constraint uniquely identifies each record in a database. A Primary Key must contain unique value and it must not contain null value. Usually Primary Key is used to index the data inside the table.

Example using PRIMARY KEY constraint at Table Level

```
CREATE table Student (s_id int PRIMARY KEY, Name varchar(60) NOT NULL, Age int);
```

The above command will creates a PRIMARY KEY on the `s_id`.

Example using PRIMARY KEY constraint at Column Level

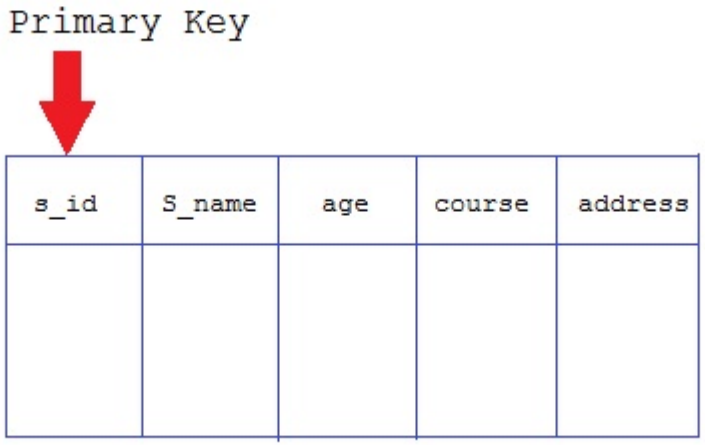
```
ALTER table Student add PRIMARY KEY (s_id);
```

The above command will creates a PRIMARY KEY on the `s_id`.

Database Keys

Primary Key

Primary key is a candidate key that is most appropriate to become main key of the table. It is a key that uniquely identify each record in a table.



Database Keys

Composite Key

Key that consist of two or more attributes that uniquely identify an entity occurance is called **Composite key**. But any attribute that makes up the **Composite key** is not a simple key in its own.

SQL Constraints

Foreign Key Constraint

FOREIGN KEY is used to relate two tables. FOREIGN KEY constraint is also used to restrict actions that would destroy links between tables. To understand FOREIGN KEY, let's see it using two table.

Customer_Detail Table :

c_id	Customer_Name	address
101	Adam	Noida
102	Alex	Delhi
103	Stuart	Rohtak

Order_Detail Table :

Order_id	Order_Name	c_id
10	Order1	101
11	Order2	103
12	Order3	102

In **Customer_Detail** table, c_id is the primary key which is set as foreign key in **Order_Detail** table. The value that is entered in c_id which is set as foreign key in **Order_Detail** table must be present in **Customer_Detail**table where it is set as primary key. This prevents invalid data to be inserted into c_id column of **Order_Detail**table.

SQL Constraints

Example using FOREIGN KEY constraint at Table Level

```
CREATE table Order_Detail(order_id int PRIMARY KEY,
order_name varchar(60) NOT NULL,
c_id int FOREIGN KEY REFERENCES Customer_Detail(c_id));
```

In this query, c_id in table Order_Detail is made as foriegn key, which is a reference of c_id column of Customer_Detail.

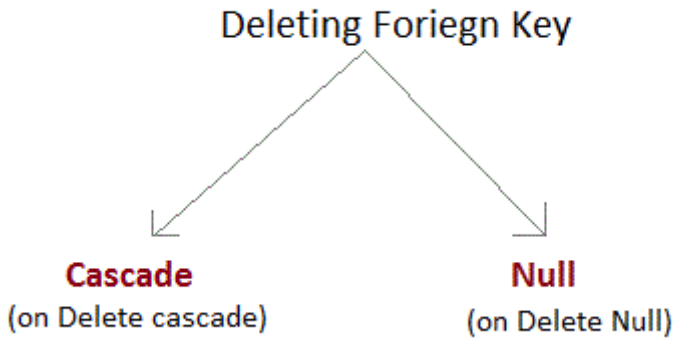
Example using FOREIGN KEY constraint at Column Level

```
ALTER table Order_Detail add FOREIGN KEY (c_id) REFERENCES Customer_Detail(c_id);
```

SQL Constraints

Behaviour of Foriegn Key Column on Delete

There are two ways to maintin the integrity of data in Child table, when a particular record is deleted in main table. When two tables are connected with Foriegn key, and certain data in the main table is deleted, for which record exit in child table too, then we must have some mechanism to save the integrity of data in child table.



- **On Delete Cascade :** This will remove the record from child table, if that value of foriegn key is deleted from the main table.
- **On Delete Null :** This will set all the values in that record of child table as NULL, for which the value of foriegn key is eleted from the main table.
- If we don't use any of the above, then we cannot delete data from the main table for which data in child table exists. We will get an error if we try to do so.

```
ERROR : Record in child table exist
```

SQL Constraints

CHECK Constraint

CHECK constraint is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database. Its like condition checking before saving data into a column.

Example using CHECK constraint at Table Level

```
create table Student(s_id int NOT NULL CHECK(s_id > 0),
Name varchar(60) NOT NULL,
Age int);
```

The above query will restrict the s_id value to be greater than zero.

Example using CHECK constraint at Column Level

```
ALTER table Student add CHECK(s_id > 0);
```

DML OPERATIONS

DML commands

Data Manipulation Language (DML) statements are used for managing data in database. DML commands are not auto-committed. It means changes made by DML command are not permanent to database, it can be rolled back.

Insert command

1) INSERT command

Insert command is used to insert data into a table. Following is its general syntax,

```
INSERT into table-name values(data1,data2,..)
```

Lets see an example,

Consider a table **Student** with following fields.

S_id	S_Name	age

```
INSERT into Student values(101,'Adam',15);
```

The above command will insert a record into **Student** table.

S_id	S_Name	age
101	Adam	15

Insert command

Example to Insert NULL value to a column

Both the statements below will insert NULL value into **age** column of the Student table.

```
INSERT into Student(id,name) values(102,'Alex');
```

Or,

```
INSERT into Student values(102,'Alex',null);
```

The above command will insert only two column value other column is set to null.

S_id	S_Name	age
101	Adam	15
102	Alex	

Insert Command

Example to Insert Default value to a column

```
INSERT into Student values(103,'Chris',default)
```

S_id	S_Name	age
101	Adam	15
102	Alex	
103	chris	14

Suppose the **age** column of student table has default value of 14.

Also, if you run the below query, it will insert default value into the age column, whatever the default value may be.

```
INSERT into Student values(103,'Chris')
```

Update Command

2) UPDATE command

Update command is used to update a row of a table. Following is its general syntax,

```
UPDATE table-name set column-name = value where condition;
```

Lets see an example,

```
update Student set age=18 where s_id=102;
```

S_id	S_Name	age
101	Adam	15
102	Alex	18
103	chris	14

Update Command

Example to Update multiple columns

```
UPDATE Student set s_name='Abhi',age=17 where s_id=103;
```

The above command will update two columns of a record.

S_id	S_Name	age
101	Adam	15
102	Alex	18
103	Abhi	17

Delete Command

3) Delete command

Delete command is used to delete data from a table. Delete command can also be used with condition to delete a particular row. Following is its general syntax,

```
DELETE from table-name;
```

Example to Delete all Records from a Table

```
DELETE from Student;
```

The above command will delete all the records from **Student** table.

Delete Command

Example to Delete a particular Record from a Table

Consider the following **Student** table

S_id	S_Name	age
101	Adam	15
102	Alex	18
103	Abhi	17

```
DELETE from Student where s_id=103;
```

The above command will delete the record where s_id is 103 from **Student** table.

S_id	S_Name	age
101	Adam	15
102	Alex	18

TCL OPERATIONS

TCL Command

Transaction Control Language(TCL) commands are used to manage transactions in database. These are used to manage the changes made by DML statements. It also allows statements to be grouped together into logical transactions.

TCL Command

Commit command

Commit command is used to permanently save any transaaction into database.

Following is Commit command's syntax,

```
commit;
```

Rollback command

This command restores the database to last commited state. It is also use with savepoint command to jump to a savepoint in a transaction.

Following is Rollback command's syntax,

```
rollback to savepoint-name;
```

Savepoint command

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

Following is savepoint command's syntax,

```
savepoint savepoint-name;
```

Example of Savepoint and Rollback

Following is the **class** table,

ID	NAME
1	abhi
2	adam
4	alex

Lets use some SQL queries on the above table and see the results.

```
INSERT into class values(5,'Rahul');
commit;
UPDATE class set name='abhijit' where id='5';
savepoint A;
INSERT into class values(6,'Chris');
savepoint B;
INSERT into class values(7,'Bravo');
savepoint C;
SELECT * from class;
```

Example of Savepoint and Rollback

```
INSERT into class values(5,'Rahul');
commit;
UPDATE class set name='abhijit' where id='5';
savepoint A;
INSERT into class values(6,'Chris');
savepoint B;
INSERT into class values(7,'Bravo');
savepoint C;
SELECT * from class;
```

The resultant table will look like,

ID	NAME
1	abhi
2	adam
4	alex
5	abhijit
6	chris
7	bravo

Now **rollback** to **savepoint B**

```
rollback to B;
SELECT * from class;
```

The resultant table will look like

ID	NAME
1	abhi
2	adam
4	alex
5	abhijit
6	chris

Now **rollback** to **savepoint A**

```
rollback to A;
SELECT * from class;
```

The result table will look like

ID	NAME
1	abhi
2	adam
4	alex
5	abhijit

DCL OPERATIONS

Data Control Language

The **Data Control Language** (**DCL**) component of the **SQL** language is used to create privileges to allow users access to, and manipulation of, the database.

- To perform any operation in the database, such as for creating tables, sequences or views we need privileges. Privileges are of two types,
- **System :** creating session, table etc are all types of system privilege.
- **Object :** any command or query to work on tables comes under object privilege.

There are two main commands:

- GRANT to grant a privilege to a user.
- REVOKE to revoke (remove) a privilege from a user. GRANT command.

Grant Command

To Allow a User to create Session

```
grant create session to username;
```

To Allow a User to create Table

```
grant create table to username;
```

Grant Command

To provide User with some Space on Tablespace to store Table

```
alter user username quota unlimited on system;
```

To Grant all privilege to a User

```
grant sysdba to username
```

To Grant permission to Create any Table

```
grant create any table to username
```

To Grant permission to Drop any Table

```
grant drop any table to username
```