Database

A database is an organized collection of data that allows easy access and management. It typically stores data in tables, with rows and columns, making it simple to find information.

Database management systems (DBMS) provide a way for users to interact with the database through a single software interface, ensuring data is managed effectively.

Databases are essential for handling large amounts of information, such as checking room availability in hotels on dynamic websites.

Some popular databases include MySQL, Oracle, MongoDB, PostgreSQL, and SQL Server.

DBMS(Database Management System)

A Database Management System is software that helps manage databases. Popular examples include MySQL and Oracle.

DBMS provides an interface for various tasks, such as creating databases, storing and updating data, and creating tables. It also ensures the security and protection of the data. When multiple users access the database, the DBMS helps maintain data consistency.

**Database Management System (DBMS)**

A Database Management System (DBMS) is software that helps manage databases and allows users to perform various tasks:

**1. Data Definition:** Create, modify, and remove definitions that organize data in the database.

**2. Data Updation:** Insert, modify, and delete actual data in the database.

**3. Data Retrieval:** Retrieve data for use by applications.

**4. User Administration:** Manage user registration, monitor users, maintain data integrity, enforce security, handle concurrency, monitor performance, and recover data from failures.

Characteristics of DBMS

**Digital Repository:** Stores and manages information on a server.

**Logical View:** Provides a clear view of data manipulation processes.

**Automatic Backup and Recovery:** Includes procedures for data backup and recovery.

**ACID Properties:** Ensures data integrity in case of failures.

**Simplified Relationships:** Reduces complexity in data relationships.

**Data Manipulation Support:** Facilitates data processing.

**Data Security:** Protects data from unauthorized access.

**Multiple Viewpoints:** Offers different views of the database based on user requirements.

Advantages of DBMS

1. Controls Redundancy: Minimizes data redundancy by storing data in a single database.

2. Data Sharing: Allows authorized users to share data easily.

3. Easy Maintenance: Centralized system makes maintenance simpler.

4. Reduced Development Time: Speeds up development and maintenance processes.

5. Backup and Recovery: Automatically backs up data and restores it when needed.

6. Multiple User Interfaces: Supports various interfaces, including graphical and programmatic options.

Disadvantages of DBMS

1. Cost: Requires high-speed processors and large memory, leading to increased hardware and software costs.

2. Size: Takes up considerable disk space and memory.

3. Complexity: Adds complexity to system requirements and management.

4. Risk of Failure: Centralized storage means that a single point of failure can result in significant data loss if the database is corrupted or damaged.

## RDBMS (Relational Database Management System)

The word RDBMS is termed as 'Relational Database Management System.' It is represented as a table that contains rows and column.

RDBMS is based on the Relational model; it was introduced by E. F. Codd.

**A relational database contains the following components:**

* Table
* Record/ Tuple
* Field/Column name /Attribute
* Instance
* Schema
* Keys

What is SQL?

SQL, short for Structured Query Language, is a programming language used to manage and manipulate data in relational database management systems (RDBMS). It is pronounced as "S-Q-L" or sometimes "See-Quell." SQL is essential for handling structured data, which is organized in tables.

SQL became an official standard by ANSI in 1986 and by ISO in 1987. It is crucial for anyone looking to work in data science, as major companies like Facebook, Instagram, and LinkedIn use SQL for data storage.

Why Use SQL?

SQL is widely used in data science and analytics for several reasons:

1. Data Management: Users can insert, update, and delete data in the database.

2. Data Retrieval: SQL allows users to easily retrieve data from the database.

3. Data Description: It helps describe the structure of the data.

4. Database Manipulation: Users can create, drop, and modify databases and their tables.

5. Views and Procedures: SQL enables the creation of views, stored procedures, and functions.

6. Data Definition: Users can define and modify the data stored in the database.

7. Permissions: SQL allows setting permissions and constraints on table columns, views, and stored procedures.

## Some SQL Commands

The SQL commands help in creating and managing the database. The most common SQL commands which are highly used are mentioned below:

1. CREATE command
2. UPDATE command
3. DELETE command
4. SELECT command
5. DROP command
6. INSERT command

### CREATE Command

This command helps in creating the new database, new table, table view, and other objects of the database.

### UPDATE Command

This command helps in updating or changing the stored data in the database.

### DELETE Command

This command helps in removing or erasing the saved records from the database tables. It erases single or multiple tuples from the tables of the database.

### SELECT Command

This command helps in accessing the single or multiple rows from one or multiple tables of the database. We can also use this command with the WHERE clause.

### DROP Command

This command helps in deleting the entire table, table view, and other objects from the database.

### INSERT Command

This command helps in inserting the data or records into the database tables. We can easily insert the records in single as well as multiple rows of the table.

SQL Syntax

To perform operations on data in a database, you need to write queries using SQL's predefined syntax. This syntax follows specific rules maintained by ISO and ANSI standards and is not case-sensitive.

Key Points About SQL Syntax:

**1. Case Insensitivity:** SQL keywords can be written in either uppercase or lowercase, but using uppercase improves readability.

**2. Text Lines:** SQL statements can span one or multiple lines.

**3. Actionable:** You can perform various actions in a database using SQL statements.

**4. Foundation:** SQL syntax is based on relational algebra and tuple relational calculus.

SQL Statements

SQL statements define what operations you want to perform on structured data. They are straightforward and resemble plain English but follow a specific syntax.

Example of an SQL Statement:

SELECT "column\_name" FROM "table\_name";

Every SQL statement starts with a keyword and ends with a semicolon (;), which separates multiple SQL statements in a single execution.

Important SQL Commands

1. SELECT: Retrieve data from a database.

2. UPDATE: Modify existing data.

3. DELETE: Remove data from a table.

4. CREATE TABLE: Create a new table.

5. ALTER TABLE: Modify an existing table.

6. DROP TABLE: Delete a table.

7. CREATE DATABASE: Create a new database.

8. DROP DATABASE: Delete a database.

9. INSERT INTO: Add new data to a table.

10. TRUNCATE TABLE: Remove all data from a table without deleting the table.

11. DESCRIBE: Show the structure of a table.

12. DISTINCT: Select unique values.

13. COMMIT: Save changes made during the current transaction.

14. ROLLBACK: Undo changes made during the current transaction.

15. CREATE INDEX: Improve the speed of data retrieval.

16. DROP INDEX: Remove an index.

17. USE: Select a database to work with.

1. SELECT Statement

This SQL statement reads the data from the SQL database and shows it as the output to the database user.

**Syntax of SELECT Statement:**

**SELECT** column\_name1, column\_name2, .…, column\_nameN

    [ **FROM** table\_name]

    [ **WHERE** condition]

[ **ORDER** **BY** order\_column\_name1 [ **ASC** | **DESC**], .... ];

**Example of SELECT Statement:**

1. **SELECT** Emp\_ID, First\_Name, Last\_Name, Salary, City
2. **FROM** Employee\_details
3. **WHERE** Salary = 100000
4. **ORDER** **BY** Last\_Name

This example shows the **Emp\_ID, First\_Name, Last\_Name, Salary, and City** of those employees from the **Employee\_details** table whose **Salary** is **100000**. The output shows all the specified details according to the ascending alphabetical order of **Last\_Name**.

### **2. UPDATE Statement**

This SQL statement changes or modifies the stored data in the SQL database.

**Syntax of UPDATE Statement:**

**UPDATE** table\_name

**SET** column\_name1 = new\_value\_1, column\_name2 = new\_value\_2, ...., column\_nameN = new\_value\_N

[**WHERE** CONDITION];

**Example of UPDATE Statement:**

1. **UPDATE** Employee\_details
2. **SET** Salary = 100000
3. **WHERE** Emp\_ID = 10;

This example changes the **Salary** of those employees of the **Employee\_detail**s table whose **Emp\_ID** is **10** in the table.

### **3. DELETE Statement**

This SQL statement deletes the stored data from the SQL database.

**Syntax of DELETE Statement:**

**DELETE** **FROM** table\_name

[ **WHERE** CONDITION];

**Example of DELETE Statement:**

1. **DELETE** **FROM** Employee\_details
2. **WHERE** First\_Name = 'Sumit';

This example deletes the record of those employees from the **Employee\_details** table whose **First\_Name** is **Sumit** in the table.

### **4. CREATE TABLE Statement**

This SQL statement creates the new table in the SQL database.

**Syntax of CREATE TABLE Statement:**

1. **CREATE** **TABLE** table\_name
2. (
3. column\_name1 data\_type [column1 **constraint**(s)],
4. column\_name2 data\_type [column2 **constraint**(s)],
5. .....
6. .....,
7. column\_nameN data\_type [columnN **constraint**(s)],
8. **PRIMARY** **KEY**(one or more col)
9. );

**Example of CREATE TABLE Statement:**

1. **CREATE** **TABLE** Employee\_details(
2. Emp\_Id NUMBER(4) NOT NULL,
3. First\_name **VARCHAR**(30),
4. Last\_name **VARCHAR**(30),
5. Salary Money,
6. City **VARCHAR**(30),
7. **PRIMARY** **KEY** (Emp\_Id)
8. );

This example creates the table **Employee\_details** with five columns or fields in the SQL database. The fields in the table are **Emp\_Id, First\_Name, Last\_Name, Salary,** and **City**. The **Emp\_Id** column in the table acts as a **primary key**, which means that the Emp\_Id column cannot contain duplicate values and null values.

### **5. ALTER TABLE Statement**

This SQL statement adds, deletes, and modifies the columns of the table in the SQL database.

**Syntax of ALTER TABLE Statement:**

1. **ALTER** **TABLE** table\_name **ADD** column\_name datatype[(**size**)];

The above SQL alter statement adds the column with its datatype in the existing database table.

1. **ALTER** **TABLE** table\_name **MODIFY** column\_name column\_datatype[(**size**)];

The above 'SQL alter statement' renames the old column name to the new column name of the existing database table.

1. **ALTER** **TABLE** table\_name **DROP** **COLUMN** column\_name;

The above SQL alter statement deletes the column of the existing database table.

**Example of ALTER TABLE Statement:**

1. **ALTER** **TABLE** Employee\_details
2. **ADD** Designation **VARCHAR**(18);

This example adds the new field whose name is **Designation** with size **18** in the **Employee\_details** table of the SQL database.

### **6. DROP TABLE Statement**

This SQL statement deletes or removes the table and the structure, views, permissions, and triggers associated with that table.

**Syntax of DROP TABLE Statement:**

1. **DROP** **TABLE** [ IF EXISTS ]
2. table\_name1, table\_name2, ……, table\_nameN;

The above syntax of the drop statement deletes specified tables completely if they exist in the database.

**Example of DROP TABLE Statement:**

1. **DROP** **TABLE** Employee\_details;

This example drops the **Employee\_details** table if it exists in the SQL database. This removes the complete information if available in the table.

### **7. CREATE DATABASE Statement**

This SQL statement creates the new database in the database management system.

**Syntax of CREATE DATABASE Statement:**

1. **CREATE** **DATABASE** database\_name;

**Example of CREATE DATABASE Statement:**

1. **CREATE** **DATABASE** Company;

The above example creates the company database in the system.

### **8. DROP DATABASE Statement**

This SQL statement deletes the existing database with all the data tables and views from the database management system.

**Syntax of DROP DATABASE Statement:**

1. **DROP** **DATABASE** database\_name;

**Example of DROP DATABASE Statement:**

1. **DROP** **DATABASE** Company;

The above example deletes the company database from the system.

### **9. INSERT INTO Statement**

This SQL statement inserts the data or records in the existing table of the SQL database. This statement can easily insert single and multiple records in a single query statement.

**Syntax of insert a single record:**

1. **INSERT** **INTO** table\_name
2. (
3. column\_name1,
4. column\_name2, .…,
5. column\_nameN
6. )
7. **VALUES**
8. (value\_1,
9. value\_2, ..…,
10. value\_N
11. );

**Example of insert a single record:**

1. **INSERT** **INTO** Employee\_details
2. (
3. Emp\_ID,
4. First\_name,
5. Last\_name,
6. Salary,
7. City
8. )
9. **VALUES**
10. (101,
11. Akhil,
12. Sharma,
13. 40000,
14. Bangalore
15. );

This example inserts **101** in the first column, **Akhil** in the second column, **Sharma** in the third column, **40000** in the fourth column, and **Bangalore** in the last column of the table **Employee\_details**.

**Syntax of inserting a multiple records in a single query:**

1. **INSERT** **INTO** table\_name
2. ( column\_name1, column\_name2, .…, column\_nameN)
3. **VALUES** (value\_1, value\_2, ..…, value\_N), (value\_1, value\_2, ..…, value\_N),….;

**Example of inserting multiple records in a single query:**

1. **INSERT** **INTO** Employee\_details
2. ( Emp\_ID, First\_name, Last\_name, Salary, City )
3. **VALUES** (101, Amit, Gupta, 50000, Mumbai), (101,  John, Aggarwal, 45000, Calcutta), (101, Sidhu, Arora, 55000, Mumbai);

This example inserts the records of three employees in the **Employee\_details** table in the single query statement.

### **10. TRUNCATE TABLE Statement**

This SQL statement deletes all the stored records from the table of the SQL database.

**Syntax of TRUNCATE TABLE Statement:**

1. **TRUNCATE** **TABLE** table\_name;

**Example of TRUNCATE TABLE Statement:**

1. **TRUNCATE** **TABLE** Employee\_details;

This example deletes the record of all employees from the Employee\_details table of the database.

### **11. DESCRIBE Statement**

This SQL statement tells something about the specified table or view in the query.

**Syntax of DESCRIBE Statement:**

1. DESCRIBE table\_name | view\_name;

**Example of DESCRIBE Statement:**

1. DESCRIBE Employee\_details;

This example explains the structure and other details about the **Employee\_details** table.

### **12. DISTINCT Clause**

This SQL statement shows the distinct values from the specified columns of the database table. This statement is used with the **SELECT** keyword.

**Syntax of DISTINCT Clause:**

1. **SELECT** **DISTINCT** column\_name1, column\_name2, ...
2. **FROM** table\_name;

**Example of DISTINCT Clause:**

1. **SELECT** **DISTINCT** City, Salary
2. **FROM** Employee\_details;

This example shows the distinct values of the **City** and **Salary** column from the **Employee\_details** table.

### **13. COMMIT Statement**

This SQL statement saves the changes permanently, which are done in the transaction of the SQL database.

**Syntax of COMMIT Statement:**

1. **COMMIT**

**Example of COMMIT Statement:**

1. **DELETE** **FROM** Employee\_details
2. **WHERE** salary = 30000;
3. **COMMIT**;

This example deletes the records of those employees whose **Salary** is **30000** and then saves the changes permanently in the database.

### **14. ROLLBACK Statement**

This SQL statement undo the transactions and operations which are not yet saved to the SQL database.

**Syntax of ROLLBACK Statement:**

1. **ROLLBACK**

**Example of ROLLBACK Statement:**

1. **DELETE** **FROM** Employee\_details
2. **WHERE** City = Mumbai;
3. **ROLLBACK**;

This example deletes the records of those employees whose **City** is **Mumbai** and then undo the changes in the database.

### **15. CREATE INDEX Statement**

This SQL statement creates the new index in the SQL database table.

**Syntax of CREATE INDEX Statement:**

1. **CREATE** **INDEX** index\_name
2. **ON** table\_name ( column\_name1, column\_name2, …, column\_nameN );

**Example of CREATE INDEX Statement:**

1. **CREATE** **INDEX** idx\_First\_Name
2. **ON** employee\_details (First\_Name);

This example creates an index **idx\_First\_Name** on the **First\_Name** column of the **Employee\_details** table.

### **16. DROP INDEX Statement**

This SQL statement deletes the existing index of the SQL database table.

**Syntax of DROP INDEX Statement:**

1. **DROP** **INDEX** index\_name;

**Example of DROP INDEX Statement:**

1. **DROP** **INDEX** idx\_First\_Name;

This example deletes the index **idx\_First\_Name** from the SQL database.

### **17. USE Statement**

This SQL statement selects the existing SQL database. Before performing the operations on the database table, you have to select the database from the multiple existing databases.

**Syntax of USE Statement:**

1. USE database\_name;

**Example of USE DATABASE Statement:**

1. USE Company;

This example uses the company database.

# **SQL Data Types**

Data types are used to represent the nature of the data that can be stored in the database table. For example, in a particular column of a table, if we want to store a string type of data then we will have to declare a string data type of this column.

Data types mainly classified into three categories for every database.

* String Data types
* Numeric Data types
* Date and time Data types

### **Data Types in MySQL, SQL Server and Oracle Databases**

### **MySQL Data Types**

A list of data types used in MySQL database. This is based on MySQL 8.0.

**MySQL String Data Types**

|  |  |
| --- | --- |
| **CHAR(Size)** | It is used to specify a fixed length string that can contain numbers, letters, and special characters. Its size can be 0 to 255 characters. Default is 1. |
| **VARCHAR(Size)** | It is used to specify a variable length string that can contain numbers, letters, and special characters. Its size can be from 0 to 65535 characters. |
| **BINARY(Size)** | It is equal to CHAR() but stores binary byte strings. Its size parameter specifies the column length in the bytes. Default is 1. |
| **VARBINARY(Size)** | It is equal to VARCHAR() but stores binary byte strings. Its size parameter specifies the maximum column length in bytes. |
| **TEXT(Size)** | It holds a string that can contain a maximum length of 255 characters. |
| **TINYTEXT** | It holds a string with a maximum length of 255 characters. |
| **MEDIUMTEXT** | It holds a string with a maximum length of 16,777,215. |
| **LONGTEXT** | It holds a string with a maximum length of 4,294,967,295 characters. |
| **ENUM(val1, val2, val3,...)** | It is used when a string object having only one value, chosen from a list of possible values. It contains 65535 values in an ENUM list. If you insert a value that is not in the list, a blank value will be inserted. |
| **SET( val1,val2,val3,....)** | It is used to specify a string that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values at one time in a SET list. |
| **BLOB(size)** | It is used for BLOBs (Binary Large Objects). It can hold up to 65,535 bytes. |

|  |  |
| --- | --- |
| **BIT(Size)** | It is used for a bit-value type. The number of bits per value is specified in size. Its size can be 1 to 64. The default value is 1. |
| **INT(size)** | It is used for the integer value. Its signed range varies from -2147483648 to 2147483647 and unsigned range varies from 0 to 4294967295. The size parameter specifies the max display width that is 255. |
| **INTEGER(size)** | It is equal to INT(size). |
| **FLOAT(size, d)** | It is used to specify a floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal point is specified by **d** parameter. |
| **FLOAT(p)** | It is used to specify a floating point number. MySQL used p parameter to determine whether to use FLOAT or DOUBLE. If p is between 0 to24, the data type becomes FLOAT (). If p is from 25 to 53, the data type becomes DOUBLE(). |
| **DOUBLE(size, d)** | It is a normal size floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal is specified by d parameter. |
| **DECIMAL(size, d)** | It is used to specify a fixed point number. Its size parameter specifies the total number of digits. The number of digits after the decimal parameter is specified by **d** parameter. The maximum value for the size is 65, and the default value is 10. The maximum value for **d** is 30, and the default value is 0. |
| **DEC(size, d)** | It is equal to DECIMAL(size, d). |
| **BOOL** | It is used to specify Boolean values true and false. Zero is considered as false, and nonzero values are considered as true. |

**MySQL Numeric Data Types**

|  |  |
| --- | --- |
| **DATE** | It is used to specify date format YYYY-MM-DD. Its supported range is from '1000-01-01' to '9999-12-31'. |
| **DATETIME(fsp)** | It is used to specify date and time combination. Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1000-01-01 00:00:00' to 9999-12-31 23:59:59'. |
| **TIMESTAMP(fsp)** | It is used to specify the timestamp. Its value is stored as the number of seconds since the Unix epoch('1970-01-01 00:00:00' UTC). Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. |
| **TIME(fsp)** | It is used to specify the time format. Its format is hh:mm:ss. Its supported range is from '-838:59:59' to '838:59:59' |
| **YEAR** | It is used to specify a year in four-digit format. Values allowed in four digit format from 1901 to 2155, and 0000. |

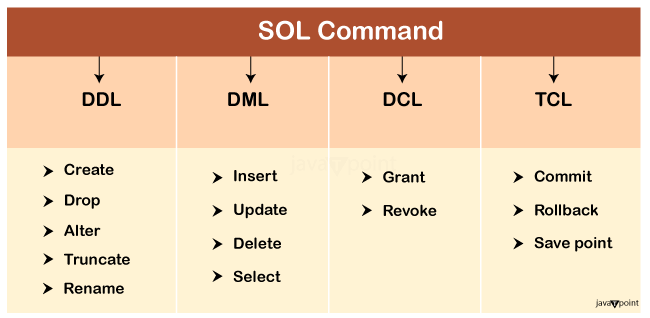
**MySQL Date and Time Data Types**

# **SQL Commands**

* SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
* SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

## Types of SQL Commands

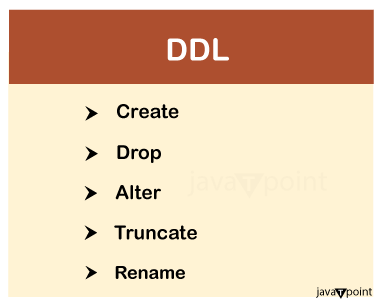
There are four types of SQL commands: DDL, DML, DCL, TCL.



### **1. Data Definition Language (DDL)**

* DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
* All the command of DDL is auto-committed that means it permanently save all the changes in the database.

**Following are some commands that come under DDL:**



**a. CREATE** It is used to create a new table in the database.

**Syntax:**

1. CREATE TABLE TABLE\_NAME (COLUMN\_NAMES DATATYPES [ ...]);

In above statement, TABLE\_NAME is the name of the table, COLUMN\_NAMES is the name of the columns and DATATYPES is used to define the type of data.

**Example:**

1. CREATE TABLE EMPLOYEE (Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);

**b. DROP:** It is used to delete both the structure and record stored in the table.

**Syntax: To DROP a table permanently from memory**

1. DROP TABLE table\_name [cascade constraint];

The cascade constraint is an optional parameter which is used for tables which have foreign keys that reference the table being dropped. If cascade constraint is not specified and used attempt to drop a table that has records in a child table, then an error will occur. So, by using cascade constraints, all child table foreign keys are dropped.

**Example**

1. DROP TABLE EMPLOYEE;

**c. ALTER:** It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

**Following are the list of modifications that can be done using ALTER command.**

* With the use of ALTER commands, we can add or drop one or more columns form existing tables.
* Increase or decrease the existing column width by changing the data type
* Make an existing mandatory column to optional.
* Enable or disable the integrity constraints in a table. We can also add, modify or delete the integrity constraints from a table.
* We can also specify a default value for existing column in a table.

**Adding new columns in Table:**

With the use of ALTER table command, we can add new columns existing table.

**Syntax:** **To add a new column in the table**

1. ALTER TABLE table\_name ADD column\_name column-definition;

**In the above syntax,** where table\_name corresponds to the name of the table, column-definition corresponds to the valid specifications for a column name and data type.

**EXAMPLE:**

1. ALTER TABLE STU\_DETAILS ADD (ADHAR\_NUM VARCHAR2 (15));

**Syntax: To ADD a multiple column from a table.**

ALTER TABLE table\_name ADD column\_name1, column\_name2;

**Example:**

Advertisement

1. ALTER TABLE STU\_DETAILS ADD ADHAR\_NUM, NAME;

**Adding constraints in a Table:**

You can also add constraints to an existing table. For example: If you forget to add a primary key constraint to a table during creation, you can add it using the ALTER TABLE statement.

Advertisement

**Syntax: To ADD a constraint from a table.**

1. ALTER TABLE table\_name ADD (column\_name column-definition CONSTRAINT constraint\_name);

**Example:**

1. ALTER TABLE STU\_DETAILS ADD (CONSTRAINT PK\_STU\_DETAILS PRIMARY KEY (STU\_ID);

**Following points should be kept in mind while adding new columns/relationships to existing tables.**

* No need for parentheses if you add only one column or constraints.
* You can add a column at any time if NULL is not specified. You can add a new column with NOT NULL if the table is empty.

**Modifying Column using ALTER:**

With the use of ALTER table we can modify column and constraint in the existing table. These statements can increase or decrease the column widths and changing a column from mandatory to optional.

**Syntax:**

1. ALTER TABLE table\_name MODIFY (column definitions....);

Advertisement

**Example:**

1. ALTER TABLE STU\_DETAILS MODIFY (ADHAR\_NUM VARCHAR2 (20));

SQL does not allow column widths to be reduced even if all column values are of valid length. So the values should be set to NULL to reduce the width of the columns. It is also not possible to reduce the width of the ADHAR\_NUM column from 18 to 12 even if all values in the ADHAR\_NUM column are less than 12 characters, unless all al values in the name column are null. You can modify the column form NULL to NOTNULL constraints if there is no record in that column in the table.

**Example:**

1. ALTER TABLE STU\_DETAILS MODIFY (ADHAR\_NUM VARCHAR2 (20) NOT NULL);

**Drop column and constraints using ALTER**

You cannot only modify columns but you can also drop them entirely if it is no longer required in a table. Using drop statement in alter command we can also remove the constraints form the table.

**Syntax: To drop a column from a table.**

1. ALTER TABLE table\_name DROP COLUMN column\_name;

**Example:**

1. ALTER TABLE STU\_DETAILS DROP COLUMN ADHAR\_NUM;

**Syntax: To drop a multiple column from a table.**

1. ALTER TABLE table\_name DROP COLUMN column\_name1, column\_name2;

**Example:**

1. ALTER TABLE STU\_DETAILS DROP COLUMN ADHAR\_NUM, NAME;

**Syntax: To drop a constraint from a table.**

1. ALTER TABLE table\_name DROP CONSTRAINT constraint\_name;

**Example:**

1. ALTER TABLE STU\_DETAILS DROP CONSTRAINT FK\_STU\_DETAILS;

**Following points should be kept in mind while deleting columns/associations:**

* You cannot drop columns in a table. If you want to drop a column from a table, the deletion is permanent so you cannot undo the column if you accidentally drop the wrong column.
* You cannot drop a column whose username is SYS.
* If you want to drop a primary key column unless you drop the foreign keys that belong to it then use cascade keyword for this.

**Example:**

Advertisement

1. ALTER TABLE STU\_DETAILS DROP PRIMARY KEY CASCADE;

* You can also enable or disable the key constraint in a table. It can be done in various situations such as: when loading large amount of data into table, performing batch operations, migrating the organizations legacy data.

**Example: To disable constraint**

1. ALTER TABLE STU\_DETAILS DISABLE CONSTRAINT FK\_STU\_DETAILS;

**Example: To Enable constraint**

1. ALTER TABLE STU\_DETAILS ENABLE CONSTRAINT FK\_STU\_DETAILS;

* Instead of dropping a column in a table, we can also make the column unused and drop it later on. It makes the response time faster. After a column has been marked as unused, the column and all its contents are no longer available and cannot be recovered in the future. The unused columns will not be retrieved using Select statement

**Example:**

1. ALTER TABLE STU\_DETAILS SET UNUSED COLUMN ADHAR\_NUM;

**RENAMING TABLE**

SQL provides the facility to change the name of the table by using a ALTER TABLE statement.

**Syntax:**

1. ALTER TABLE <OLD\_TABLENAME> Rename to <NEW\_TABLENAME>;

**Example:**

1. ALTER TABLE STU\_NAME Rename to STUDENT\_NAME;

**d. TRUNCATE:** It is used to delete all the rows from the table and free the space containing the table.

**Syntax:**

1. TRUNCATE TABLE table\_name;

**Example:**

1. TRUNCATE TABLE EMPLOYEE;

**e. Rename:** It is used to rename the table.

**Syntax:**

1. Rename <OLD\_TABLENAME> to <NEW\_TABLENAME>;

In the above syntax, Rename is a command, <OLD\_TABLENAME> is the name of the table and <NEW\_TABLENAME> is the name that you have changed.

**Example:**

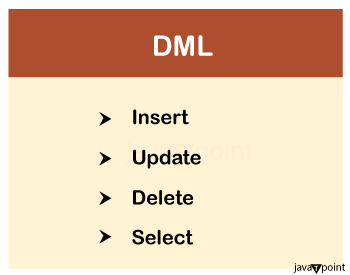
Advertisement

1. Rename STU\_NAME to STUDENT\_NAME;

### **2. Data Manipulation Language**

* DML commands are used to modify the database. It is responsible for all form of changes in the database.
* The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

**Following are the some commands that come under DML:**



**a. INSERT:** The INSERT statement is a SQL query. It is used to insert data into the row of a table. To insert a new row into a table you must be your on schema or INSERT privilege on the table.

**Following are the list of points should be considered while inserting the data into tables.**

* SQL uses all the columns by default if you do not specify the column name while inserting a row.
* The number of columns in the list of column name must match the number of values that appear in parenthesis after the word "values".
* The data type for a column and its corresponding value must match.

**Syntax: To add row in a table**

1. INSERT INTO TABLE\_NAME
2. (col1, col2, col3,.... col N)
3. VALUES (value1, value2, value3, .... valueN);

Or

1. INSERT INTO TABLE\_NAME
2. VALUES (value1, value2, value3, .... valueN);

In the above syntax, TABLE\_NAME is the name of the table in which the data will be inserted. The (col1, col2, col3, col N) are optional and name of the columns in which values will be inserted. The value1 corresponds to the value of be inserted in col1 and similarly value2 corresponds to the value of be inserted in col2 and so on.

**For example:**

1. INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");

**Syntax: To add multiple rows in a table**

1. INSERT INTO TABLE\_NAME
2. (col1, col2, col3,.... col N)
3. VALUES (value1, value2, value3, .... valueN), (value1, value2, value3, .... valueN);

**For example:**

1. INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS"), ("Raman", "DBMS"), ("Priya", "DBMS");

**b. UPDATE:** This command is used to update or modify the value of a column in the table.

**Syntax: To update record in a table**

1. UPDATE table\_name SET [column\_name1= value1,...column\_nameN = valueN] [WHERE CONDITION]

In the above syntax, table\_name is the name of the table, the column\_name is the name of column in the table to be modified, and value1 corresponds to the valid SQL values. The "WHERE" is a condition that restricts the rows updated for which the specified condition is true. If condition is not specified is not defined then SQL updates all the rows in the table. It contains comparison and logical operators etc.

**The following the list of points should be remembered while executing the UPDATE statement.**

* It references only one table.
* In the SET clause atleast one column must be assigned an expression for the update statement,
* In the where clause you could also give multiple conditions for update statement.

**For example:**

1. UPDATE students
2. SET User\_Name = 'Sonoo'
3. WHERE Student\_Id = '3'

**c. DELETE:** It is used to remove one or more row from a table. To delete rows from the table, it must be in your schema or you must have delete privilege.

**Syntax: To Delete a record from table**

1. DELETE FROM table\_name [WHERE condition];

In the above syntax, condition is used in the where clause to filter the records that are actually being removed. You can remove zero or more rows from a table. If you do not use where condition then DELETE statement will remove all the rows from the table. You can also use one or multiple conditions in WHERE clause.

**For example:**

1. DELETE FROM javatpoint
2. WHERE Author="Sonoo";

**d. SELECT:** This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

**Syntax: It is used for retrieve the records from table**

1. SELECT expressions
2. FROM TABLES
3. WHERE conditions;

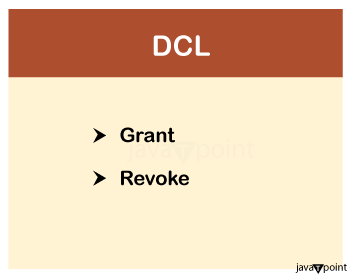
**For example:**

1. SELECT emp\_name
2. FROM employee
3. WHERE age > 20;

### **3. Data Control Language**

DCL commands are used to grant and take back authority from any database user.

Following are the some commands that come under DCL:



**Syntax:**

1. GRANT <obj\_priv> ON <obj\_name> To <username>;

In the above syntax, obj\_priv> is the DML statement like Insert, Delete , update and Select and <obj\_name> is a table, view etc. and username is the name of the authorized user.

**Example**

1. GRANT SELECT, UPDATE ON MY\_TABLE TO SOME\_USER, ANOTHER\_USER;

**b. Revoke:** It is used to take back permissions from the user.

**Syntax:**

1. REVOKE <obj\_priv> ON <obj\_name> FROM <username>;

In the above syntax, obj\_priv> is the DML statement like Insert, Delete , update and Select and <obj\_name> is a table, view etc. and username is the name of the user from whom the permission is revoked.

**Example**

1. REVOKE SELECT, UPDATE ON MY\_TABLE FROM USER1, USER2;

### **4. Transaction Control Language**

Transactions are atomic i.e. either every statement succeeds or none of statement succeeds. There are number of Transaction Control statements available that allow us to control this behavior. These statements ensure data consistency. TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

**Following are the some commands that come under TCL:**



**a. Commit:** Commit command is used to save all the transactions to the database. It makes your changes permanent and ends the transaction.

**Syntax: To permanently save the changes**

Advertisement

1. COMMIT;

**Example:**

1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. COMMIT;

**b. Rollback:** Rollback command is used to undo transactions that have not already been saved to the database. Rollback also serves to end the current transaction and begin a new one.

Consider a Situation where you have completed a series of INSERT, UPDATE or DELETE statements but have not yet explicitly committed them and yiu encounter a problem such as computer failure, then SQL will automatically rollback any uncommitted work.

**Syntax: To remove the changes**

1. ROLLBACK;

**Example:**

1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. ROLLBACK;

**c. SAVEPOINT:** It is used to roll the transaction back to a certain point without rolling back the entire transaction.

**Syntax:**

1. SAVEPOINT SAVEPOINT\_NAME;

In the above syntax, SAVEPOINT\_NAME is the name given to savepoint.

To selectively ROLLBACK a group of statements within a large transaction use the following command is used.

1. Rollback TO <save\_point\_name>

**Example:**

1. DELETE FROM CUSTOMERS WHERE AGE = 15;
2. SAVEPOINT A;
3. DELETE FROM CUSTOMERS WHERE AGE = 35;
4. ROLLBCAK TO A;