# **Database Languages**

* A DBMS must provide appropriate languages or interface for each category of users to express database queries and updates.
* So, DB Languages:

1. Communicates with the database
2. Used to create and maintain database on computer

* Mainly two types of DB Languages:
* Data Definition Language(DDL): to specify **database schema**
* Data Manipulation Language(DML): To express database **queries** and **updates**
* In practical, DDL and DML are not separate languages, instead they are the parts of a **single database language** such as widely used **SQL**(Structured Query Language)

## **Data Definition Language**

* DDL (stands for Data Definition Language) is used to specify the database schema.
* Used by the DBA and database designers to specify the conceptual schema of a database.
* DDL is used for creating tables, schema, indexes, constraints, etc in database.
* **DDL** is **used to store the information of metadata** like the **no. of tables** and **schemas**, **their names**, **indexes**, **columns in each table**, **constraints**, etc.
* **DDL compiler** generates **a set of table templates** stored in a **data dictionary**  (data dictionary contains data about data).
* Data dictionary contains metadata:
* Database Schema
* Table name
* Column names & types
* Integrity Constraints
* Primary Key
* Authorization
* Who can access what

DDL Commands in SQL

* **CREATE:** used to create the database instance
* **ALTER:** used to alter the structure of the database
* **DROP:** used to delete the database instance
* **TRUNCATE:** used to remove all records from a table
* **RENAME:** used to rename database instances

These commands either defines or update the database schema that’s why they come under Data Definition Language.

## **Data Manipulation Language (DML)**

* DML (Data Manipulation Language) is used for **accessing** and **manipulating data** in a database.
* It allows users to **insert**, **update**, **delete** and **retrieve** data from the database.
* **DML** is also known as **Query Languages**.
* A query is a statement requesting the retrieval of information
* Two classes of DML Languages:
* Procedural DMLs/High-level DMLs:
* User specifies what data is required and how to get the data
* Procedural DML is embedded into a high-level programming language like java, etc.
* Eg: PL/SQL
* Declarative (Non-Procedural) DMLs/ Low-level DMLs:
* User only specifies what data is required; without specifying how to get the data
* Declarative DMLs are usually easier to learn and use than procedural DMLs. However, since a user does not have to specify how to get all data, the database system has to figure out an alternative means of accessing data.
* Eg: SQL

DML commands in SQL

* **SELECT:** used to retrieve data from a database
* **INSERT:** used to insert data into a table
* **UPDATE:** used to update existing data within a table
* **DELETE:** used to delete one or some or all records from a table

## **Data Control Language**

* Data Control Language (DCL) is used to control privilege in database i.e., DCL is used for granting and revoking user access on a database (Authorization).
* To perform any operation in the database, such as for creating tables, sequences, or views, we need privileges.
* it is a component of SQL
* DCL commands in SQL
* **GRANT:** to give user access privileges to a database
* **REVOKE:** to take back permission from the user
* The operations for which privileges may be granted to or revoked from a user or role apply to both the **DDL** and **DML**.

## **Transaction Control Language (TCL)**

* TCL (Transcation Control Language) commands are used to manage transactions in the database.
* TCL is used to run the changes made by the **DML statement**.
* The changes in the database that we made using DML commands are either performed or rollbacked using TCL.
* TCL commands in SQL
* **COMMIT:** to save the transaction on the database
* **ROLLBACK:** to restore the database to original since the last commit
* **SAVEPOINT:** savepoint command is used to temporarily save a transaction so that you can rollback to that point whenever necessary.

# **Limit**

* The limit clause is used to specify the no. of records to return.
* Useful on large table with thousands of records b/c returning a large no of records can impact performance.
* Offset is used to return the result from specified record.

Syntax:

SELECT \* FROM employee LIMIT 3 OFFSET 3;

# **SQL Functions**

* SQL provides many built-in functions to perform operations on data.
* These functions are useful while performing mathematical calculations, string concatenation, sub-string etc.
* SQL Functions are divided into two categories:

1. Aggregate Functions
2. Scalar Functions

* Aggregate functions and Scalar functions both return a single value.
* Aggregate functions operate on many records while scalar functions operate on each records independently

## **Aggregate Functions in SQL**

* Aggregate functions performs on set of values and returns a single value
* Some commonly use aggregate functions are:

1. COUNT()
2. SUM()
3. AVG()
4. MAX()
5. MIN()

* Aggregate functions are often used by GROUP BY clause of the SELECT statement to group multiple rows together as input to form a single value output
* Aggregate functions ignore NULL values, except COUNT(\*).
* COUNT() – returns the no. of rows that matches the specified criteria.

**SELECT**

**COUNT(col\_name)**

**FROM**

**table\_name**

**WHERE condition;**

**SELECT COUNT(\*) FROM Student;**

**SELECT COUNT(Marks) FROM Student;**

**SELECT COUNT(DISTINCT Marks) FROM Student;**

* SUM() – returns total sum of a selected **numeric** column.

**SELECT**

**SUM(col\_name)**

**FROM**

**table\_name**

**WHERE condition;**

* AVG() – returns the average value of selected **numeric** column.

**SELECT**

**SUM(col\_name)**

**FROM**

**table\_name**

**WHERE condition;**

* MAX() – returns maximum value of selected column.
* MIN() – returns minimum value of selected column.

# **Sub Queries**

A query inside another query is called sub-query or nested query.

Based on the execution process of sub-query it is classified into two ways:

1. Non-Corelated Sub Query

* Here the inner query is executed first and then the outer query is executed.
* i.e., Outer query always depends on the result from the inner query.

1. Corelated Sub Query

* Here, the outer query is executed first and then the inner query is
* i.e., Inner query always depends on the result from the outer query.

Types of Non-Corelated Sub Query:

1. Simple/Single row sub query – when subquery returns one value
2. Multiple rows sub query – when subquery returns more than one value

# **JOINS**

* **Three** **retrieval** methods

1. **Selection** – with WHERE clause
2. **Projection** – without WHERE clause
3. **Join** – applied when we have to retrieve from more than one table

Kinds of Joins (Join is classified into two categories):

1. **ANSI-format joins (new style format)**

* With **ON** keyword join condition.
* Inner Join –
* Retrieve data from multiple tables based on an **equality** condition
* Common column is necessary and it’s datatype must be same.
* Result = Matching data/rows
* Outer Join – matching and un-matching data from the table

1. **Left outer** join – retrieve the **matching data from all tables** but **unmatching data from the left-hand side table** only.
2. **right outer** join - retrieve the **matching data from all tables** but **unmatching data from the right-hand side table** only.
3. **full outer** join – retrieve the **matching** and **unmatching** data from all tables.

* Cross join –
* the product of rows (row to row production MxN rows).
* Just for producing purpose.
* This is the only join which joins multiple tables info **without condition.**
* No requirement of common column.
* Natural join – to avoid duplicate column from join table. Not supported in MSSQL server but can be implemented using **EQUI-JOIN** to avoid duplicate column **.**

1. **Non-ANSI format joins (old style format)**

* Without **WHERE** keyword join condition.
* **EQUI JOIN**
* Retrieving data from the multiple table based on the equality condition ( = )
* But cannot use following operators ( <, >, <=, >=, !<, !>, !=)
* **NON-EQUI JOIN**
* Retrieving data from the multiple table based on any condition except equality condition( = )
* Cannot use =
* Can use <, >, <=, >=, !<, !>, !=, BETWEEN, AND, …
* **SELF JOIN**
* Joining a table data by itself.
* Can only be used using alias name(mandatory for self join mechanism)
* Can be implemented on single table.
* Scenario to use - When we compare the column value within the same table.

# **SET OPERATORS**

* When you want to combine two or more than two select operators.

1. UNION – to combine result of two or more select statement as a single unit of values **without** duplicate values.
2. UNION ALL – to combine result of two or more select statement as a single unit of values **with** duplicate values.
3. INTERSECTION – to returns common values from the tables.
4. EXCEPT – to return all the values from LHS table which are not found in RHS table.

RULES:

* No. of columns should be same within two select statement.
* Order of the column should be same.
* Data types of the column must match.