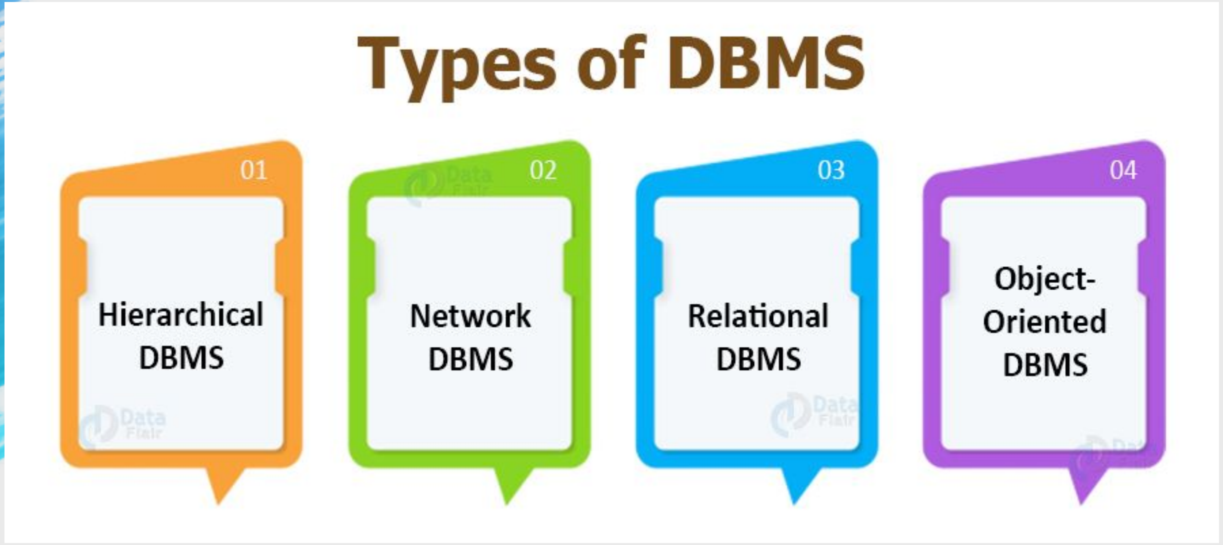
DBMS
PART - 2

TYPES OF DBMS


Types of DBMS

01 Hierarchical DBMS

02 Network DBMS

03 Relational DBMS

04 Object-Oriented DBMS

HIERARCHIAL

- In a Hierarchical database, model data is organized in a tree-like structure.
- Data is Stored Hierarchically (top down or bottom up) format.
- Data is represented using a parent-child relationship.

NETWORK

- ❖ The network database model allows each child to have multiple parents.
- ❖ It helps you to address the need to model more complex relationships like as the orders/parts many-to-many relationship.
- ❖ In this model, entities are organized in a graph which can be accessed through several paths.

RELATIONAL

- ★ Relational DBMS is the most widely used DBMS model because it is one of the easiest.
- ★ This model is based on normalizing data in the rows and columns of the tables.
- ★ Relational model stored in fixed structures and manipulated using SQL.

“

OBJECT ORIENTED

- ➔ In Object-oriented Model data stored in the form of objects.
- ➔ It defines a database as a collection of objects which stores both data members values and operations.

APPLICATIONS OF DBMS

- Banking
- Airlines
- Universities
- Telecommunication
- Finance
- Sales
- Manufacturing
- HR Management

ADVANTAGES OF DBMS

- ★ Reduced Redundancy
- ★ Shared data
- ★ Data integrity
- ★ Data security
- ★ Privacy
- ★ Backup & Recovery

CONSTRAINTS IN SQL

Constraints in SQL are used to specify the limit on the data type of the table.

It can be specified while creating or altering the table statement. The sample of constraints are:

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

AGGREGATE FUNCTIONS

- Aggregate functions in DBMS take multiple rows from the table and return a value according to the query.
- All the aggregate functions are used in Select statement.

```
SELECT <FUNCTION NAME> (<PARAMETER>) FROM <TABLE NAME>
```

- AVG
- COUNT
- MAX
- SUM
- STDDEV
- VARIANCE

AVG Function

This function returns the average value of the numeric column that is supplied as a parameter.

Example: Write a query to select average salary from employee table.

```
Select AVG(salary) from Employee
```

MAX Function

The MAX function is used to find maximum value in the column that is supplied as a parameter. It can be used on any type of data.

Example – Write a query to find the maximum salary in employee table.

```
Select MAX(salary) from Employee
```

COUNT Function

The count function returns the number of rows in the result. It does not count the null values.

Example: Write a query to return number of rows where salary > 20000.

```
Select COUNT(*) from Employee where Salary > 20000;
```

Types –

- COUNT(*): Counts all the number of rows of the table including null.
- COUNT(COLUMN_NAME): count number of non-null values in column.
- COUNT(DISTINCT COLUMN_NAME): count number of distinct values in a column.

SUM Function

This function sums up the values in the column supplied as a parameter.

Example: Write a query to get the total salary of employees.

```
Select SUM(salary) from Employee
```

STDDEV Function

The STDDEV function is used to find standard deviation of the column specified as argument.

Example – Write a query to find standard deviation of salary in Employee table.

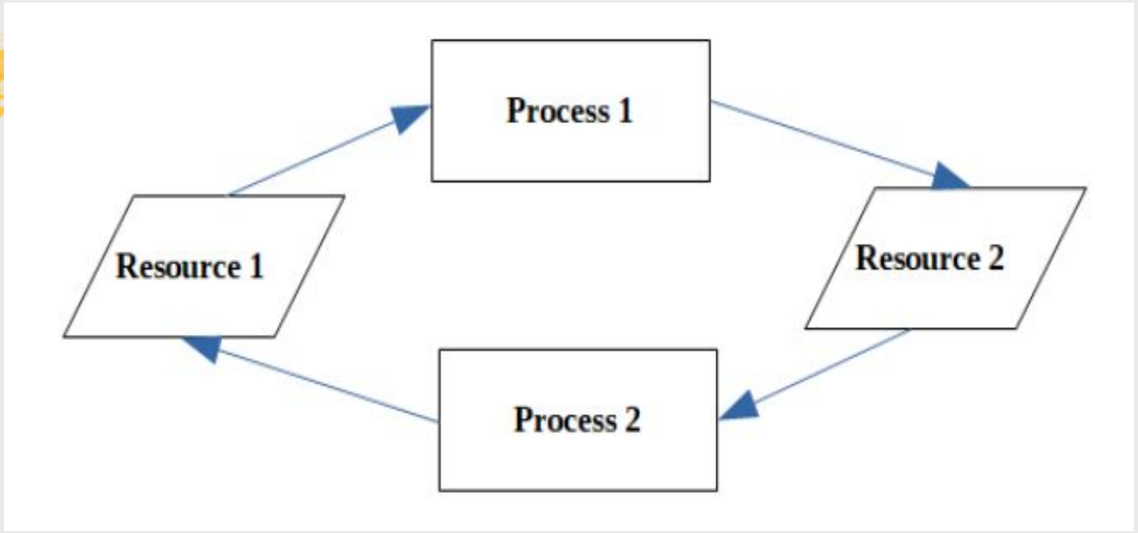
```
Select STDDEV(salary) from Employee
```

SCALAR FUNCTIONS IN DBMS

Function	Description
LCASE()	Used to convert string column values to lowercase
UCASE()	This function is used to convert a string column values to Uppercase.
LEN()	Returns the length of the text values in the column.
MID()	Extracts substrings in SQL from column values having String data type.
ROUND()	Rounds off a numeric value to the nearest integer.
NOW()	This function is used to return the current system date and time.
FORMAT()	Used to format how a field must be displayed.

DEADLOCKS IN DBMS

A deadlock occurs when two or more processes need some resource to complete their execution that is held by the other process.

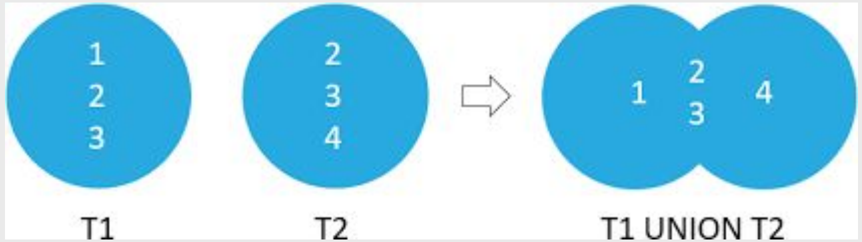


Deadlock Avoidance

- When a database is stuck in a deadlock state, then it is better to avoid the database rather than aborting or restating the database. This is a waste of time and resource.
- A method like "wait for graph" is used for detecting the deadlock situation but this method is suitable only for the smaller database.
- For the larger database, deadlock prevention method can be used.

UNION OPERATOR

The UNION operator is used to combine the result-set of two or more SELECT statements.



UNION Syntax

```
SELECT column_name(s) FROM table1

UNION

SELECT column_name(s) FROM table2;
```

UNION ALL OPERATOR

Same as Union Operator but allows duplicate values

```
SELECT City FROM Customers

UNION ALL

SELECT City FROM Suppliers

ORDER BY City;
```

EXCEPT & INTERSECT

- EXCEPT returns distinct rows from the left input query that aren't output by the right input query.
- INTERSECT returns distinct rows that are output by both the left and right input queries operator.

Now let's look at how you would return only the food you ate (or drank) for lunch, but did not have for dinner:

```
SELECT item FROM Lunch
EXCEPT
SELECT item FROM Dinner;
```

```
SELECT item FROM Lunch
INTERSECT
SELECT item FROM Dinner;
```

DISTINCT

The SELECT DISTINCT command returns only distinct (different) values in the result set.

The following SQL statement selects only the DISTINCT values from the "Country" column in the "Customers" table:

```
SELECT DISTINCT Country FROM Customers;
```

GROUP BY & ORDER BY

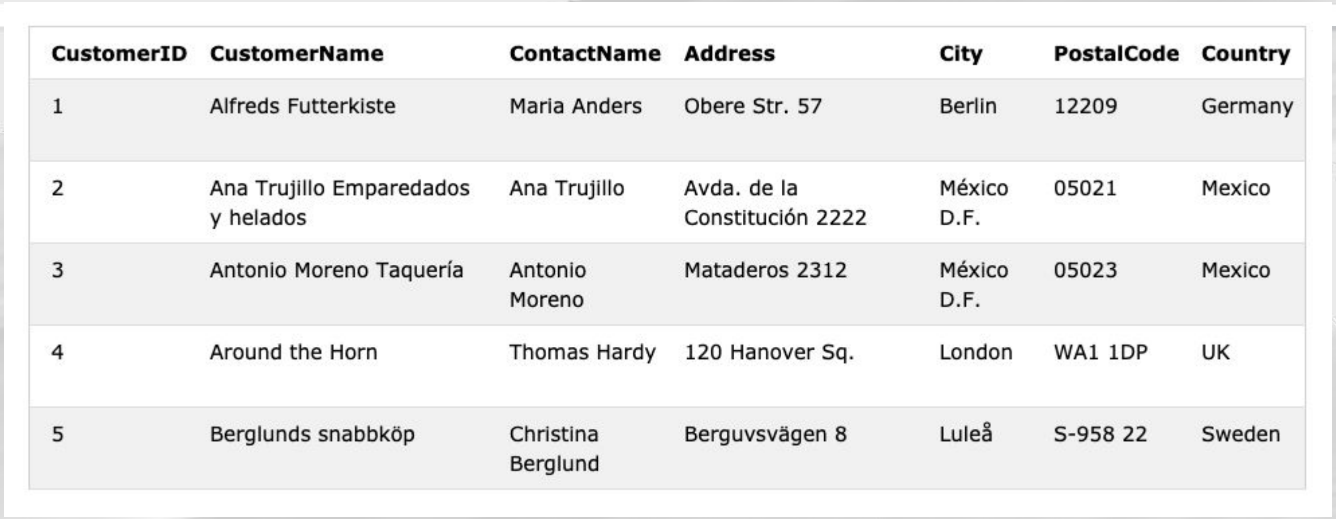
The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

If you want to know the total amount of the salary on each customer, then the GROUP BY query would be as follows.

```
SQL> SELECT NAME, SUM(SALARY) FROM CUSTOMERS
   GROUP BY NAME;
```



CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

```
SELECT COUNT(CustomerID), Country

FROM Customers

GROUP BY Country;

ORDER BY COUNT(CustomerID) DESC;
```

```
o/p:
Number of records 5
Count(Customer ID )   Country
2                    Mexico
1                    Germany
```

BETWEEN OPERATOR

- The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.
- The BETWEEN operator is inclusive: begin and end values are included.

```
SELECT * FROM Products

WHERE Price BETWEEN 10 AND 20;
```

LIKE Operator

The `LIKE` operator is used in a `WHERE` clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the `LIKE` operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

SYNTAX

```
SELECT column1, column2, ...

FROM table_name

WHERE columnN LIKE pattern;
```

LIKE Operator	**Description**
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

ALIAS

- ❖ SQL aliases are used to give a table, or a column in a table, a temporary name.
- ❖ Aliases are often used to make column names more readable.

Example:

```
SELECT column_name(s)

FROM table_name AS alias_name;
```

ANY OPERATOR

The ANY operator:

- returns a boolean value as a result
- returns TRUE if ANY of the subquery values meet the condition

ANY means that the condition will be true if the operation is true for any of the values in the range.

Example :

```
SELECT ProductName
FROM Products
WHERE ProductID = ANY
  (SELECT ProductID
  FROM OrderDetails
  WHERE Quantity = 10);
```

ALL OPERATOR

The ALL operator:

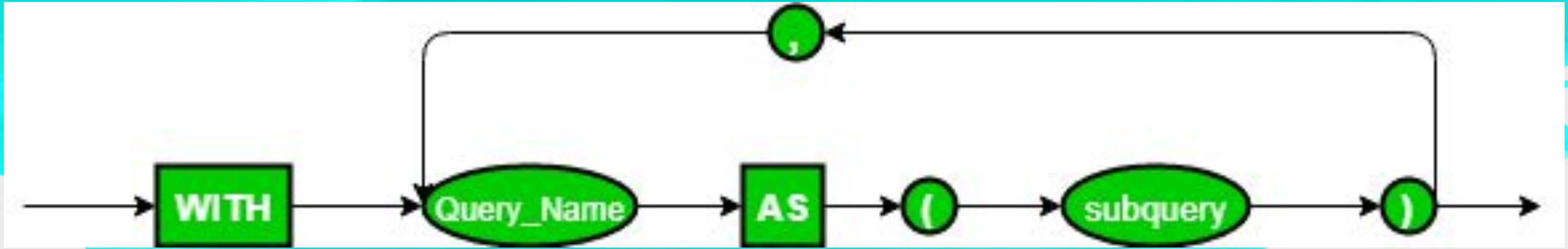
- returns a boolean value as a result
- returns TRUE if ALL of the subquery values meet the condition
- is used with SELECT, WHERE and HAVING statements

ALL means that the condition will be true only if the operation is true for all values in the range.

```
SELECT ProductName
FROM Products
WHERE ProductID = ALL
  (SELECT ProductID
  FROM OrderDetails
  WHERE Quantity = 10);
```

WITH CLAUSE

The clause is used for defining a temporary relation such that the output of this temporary relation is available and is used by the query that is associated with the WITH clause.



CASE STATEMENT

- ❑ The CASE statement goes through conditions and returns a value when the first condition is met (like an if-then-else statement).
- ❑ If there is no ELSE part and no conditions are true, it returns NULL.

```
SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN 'The quantity is greater than 30'
    WHEN Quantity = 30 THEN 'The quantity is 30'
    ELSE 'The quantity is under 30'
END AS QuantityText
FROM OrderDetails;
```

LIMIT

The SQL SELECT LIMIT statement is used to retrieve records from one or more tables in a database and limit the number of records returned based on a limit value.

```
SELECT expressions
FROM tables
[WHERE conditions]
[ORDER BY expression [ ASC | DESC ]]
LIMIT number_rows [ OFFSET offset_value ];
```

CURSORS

Cursor is a Temporary Memory or Temporary Work Station.

It is Allocated by Database Server at the Time of Performing DML operations on Table by User. Cursors are used to store Database Tables.

There are 2 types of Cursors:

- ➢ Implicit Cursors
- ➢ Explicit Cursors.

GROUP BY VS FILTERS

`GROUP BY` enables you to use aggregate functions on groups of data returned from a query.

`FILTER` is a modifier used on an aggregate function to limit the values used in an aggregation.

All the columns in the select statement that aren't aggregated should be specified in a `GROUP BY` clause in the query.

For example, let's say you wanted to know the average deal by sales agent for each of their customers. If you used the query:

```
 SELECT sales_agent,account,
 SUM(close_value) FROM sales_pipeline
 WHERE sales_pipeline.deal_stage = "Won"
 GROUP BY sales_agent
```

For example, if you wanted to know the number of those deals that had a value greater than 1000, you could use the query:

```
SELECT sales_agent,
      COUNT(sales_pipeline.close_value) AS total,
      COUNT(sales_pipeline.close_value)
FILTER(WHERE sales_pipeline.close_value > 1000) AS `over 1000`
```

Q-1. Write an SQL query to fetch “FIRST_NAME” from Worker table using the alias name as <WORKER_NAME>.

```
Select FIRST_NAME AS WORKER_NAME from Worker;
```

Q-2. Write an SQL query to fetch “FIRST_NAME” from Worker table in upper case.

```
Select upper(FIRST_NAME) from Worker;
```

Q-3. Write an SQL query to fetch unique values of DEPARTMENT from Worker table.

```
Select distinct DEPARTMENT from Worker;
```

Q-4. Write an SQL query to print the first three characters of FIRST_NAME from Worker table.

```
Select substring(FIRST_NAME,1,3) from Worker;
```

Q-5. Write an SQL query to find the position of the alphabet (‘a’) in the first name column ‘Amitabh’ from Worker table.

```
Select INSTR(FIRST_NAME, BINARY'a') from Worker where
FIRST_NAME = 'Amitabh';
```

Q-6. Write an SQL query to print the FIRST_NAME from Worker table after removing white spaces from the right side.

```
Select RTRIM(FIRST_NAME) from Worker;
```

Q-7. Write an SQL query to print the DEPARTMENT from Worker table after removing white spaces from the left side.

```
Select LTRIM(DEPARTMENT) from Worker;
```

Q-8. Write an SQL query that fetches the unique values of DEPARTMENT from Worker table and prints its length.

```
Select distinct length(DEPARTMENT) from Worker;
```

Q-9. Write an SQL query to print the FIRST_NAME from Worker table after replacing ‘a’ with ‘A’.

```
Select REPLACE(FIRST_NAME,'a','A') from Worker;
```

Q-10. Write an SQL query to print the FIRST_NAME and LAST_NAME from Worker table into a single column COMPLETE_NAME. A space char should separate them.

```
Select CONCAT(FIRST_NAME, ' ', LAST_NAME) AS
'COMPLETE_NAME' from Worker;
```

Q-12. Write an SQL query to print all Worker details from the Worker table order by FIRST_NAME Ascending and DEPARTMENT Descending.

```
Select * from Worker order by FIRST_NAME
asc,DEPARTMENT desc;
```

Q-13. Write an SQL query to print details for Workers with the first name as “Vipul” and “Satish” from Worker table.

```
Select * from Worker where FIRST_NAME in
('Vipul','Satish');
```

Q-14. Write an SQL query to print details of workers excluding first names, “Vipul” and “Satish” from Worker table.

```
Select * from Worker where FIRST_NAME not in
('Vipul','Satish');
```

Q-15. Write an SQL query to print details of Workers with DEPARTMENT name as “Admin”.

```
Select * from Worker where DEPARTMENT like 'Admin%';
```

Q-17. Write an SQL query to print details of the Workers whose FIRST_NAME ends with ‘a’.

Ans.

The required query is:

```
Select * from Worker where FIRST_NAME like '%a';
```

Q-16. Write an SQL query to print details of the Workers whose FIRST_NAME contains ‘a’.

The required query is:`Select * from Worker where FIRST_NAME like '%a%';`

Q-11. Write an SQL query to print all Worker details from the Worker table order by FIRST_NAME Ascending.

```
Select * from Worker order by FIRST_NAME asc;
```

Q-18. Write an SQL query to print details of the Workers whose FIRST_NAME ends with ‘h’ and contains six alphabets.

```
    Select * from Worker where FIRST_NAME like '_____h';
```

Q-19. Write an SQL query to print details of the Workers whose SALARY lies between 100000 and 500000.

```
    Select * from Worker where SALARY between 100000 and
500000;
```

Q-20. Write an SQL query to print details of the Workers who have joined in Feb’2014.

```
Select * from Worker where year(JOINING_DATE) = 2014 and
month(JOINING_DATE) = 2;
```

Q-21. Write an SQL query to fetch the count of employees working in the department ‘Admin’.

```
SELECT COUNT(*) FROM worker WHERE DEPARTMENT = 'Admin';
```

“

References

1. ***Javatpoint***
2. ***Tutorialspoint***
3. ***W3schools***
4. ***GFG***
5. ***Studytonight***

THANKS!