SQL

Predicate in DBMS

Predicate in DBMS

A Predicate in DBMS is a condition expression which evaluates and results in boolean value either true or false which enables decision making in retrieving and manipulating a record.

A predicate is a condition that is specified for:

- Filtering the data using the ***WHERE*** clause,
- Pattern matching in ***LIKE*** operator,
- Specifying a set of list for using ***IN*** operator,
- Manipulating a range of values using ***BETWEEN*** operator, etc

Consider a sample table 'emp'

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	500	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	500	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20
7369	SMITH	CLERK	7902	17-DEC-80	800	500	20

The predicate in where clause

```
select * from emp
where [job='MANAGER'];
```

O/P

```
3 rows selected.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO

7698	BLAKE	MANAGER	7839	01-MAY-81	2850	–	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20

In our ***emp*** table, there are three *managers* that's why only three records are displayed, because the condition is true for only those three rows i.e when the job is a ***manager.***

The predicate in 'IN' clause

It is used to specify a set of values and where manipulation is performed on all the values specified in the set and ***if any of the value that is present in the list matches with the values present in a table then it returns true and is operation is performed***

Example

```
select empno,job,sal,hiredate
from emp
where [ename in('SCOTT','FORD','SMITH','JONES')];
```

O/P

```
4 rows selected
```

EMPNO	JOB	SAL	HIREDATE
7566	MANAGER	2975	02-APR-81
7788	ANALYST	3000	19-APR-87
7902	ANALYST	3000	03-DEC-81
7369	CLERK	800	17-DEC-80

Records of all those employees that are specified in the list of ***in clause*** are displayed.

Predicate in 'BETWEEN CAUSE

It is used to perform ***data comparison and manipulation over a range of values*** present in the database table

Example

```
select empno,job,sal,hiredate
from emp
where [sal between 800 and 2900];
```

O/P

```
3 rows selected
```

EMPNO	JOB	SAL	HIREDATE
7698	MANAGER	2850	01-MAY-81
7782	MANAGER	2450	09-JUN-81
7369	CLERK	800	17-DEC-80

- The details of those employees whose salary is present in the range between 800/- to 2900/- are retrieved and it also considers specified values inclusive of the range
- In the case of between operator lower value is first specified and followed by the higher value & an ***and*** operator in between these higher and lower values.

The predicate in 'LIKE ' clause

The ***like*** operator is a pattern matching operator that returns ***those records that match with the specified data pattern***

Example

```
select empno,ename,hiredate,sal,job
from emp
where [ename like 'S%'];
```

O/P

```
2 rows selected
```

EMPNO	ENAME	HIREDATE	SAL	JOB
7788	SCOTT	19-APR-87	3000	ANALYST
7369	SMITH	17-DEC-80	800	CLERK

All the records of employees whose names starting with the letter 'S' are displayed.

Predicate in 'IS NULL' clause

All operations upon null values present in the table must be done using this ***is null*** operator, we cannot compare null value using the assignment operator(=)

```
select * from emp
where [comm is null]
```

O/P

```
4 rows selected
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20

The details of those employees whose commission value is Null are displayed.

The predicate in NOT clause

Not operator is negation operator which is used along with ***like, between, is null, in*** operators, It ***performs the reverse action*** of all these operators

Example

```
select * from emp
where [sal NOT between 800 and 2900 ];
```

O/P

```
4 rows selected
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20

The details of those employees whose salary does not fall in the range between 800 to 2900 are displayed.

SQL Operators in DBMS

SQL Operators in DBMS

SQL mainly provides the following
set of operators

- SQL Arithmetic Operators

- SQL Comparison Operators
- SQL Logical Operators
- SQL Special operators

SQL Arithmetic operators

SQL provides five basic arithmetic operators, assume if a =10, b=20.

Operator	Example
+	a + b will give 30
–	a – b will give -10
*	a * b will give 200
/	b / a will give 2
%	b % a will give 0

Consider the sample table '*emp*'

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	500	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	500	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20
7369	SMITH	CLERK	7902	17-DEC-80	800	500	20

Example

```
select empno,ename,sal,sal*12 as annsal,sal/12 as comm
from emp;
```

O/P

7 rows selected.

EMPNO	ENAME	SAL	ANNUAL	COMM
7839	KING	5000	60000	416.66
7698	BLAKE	2850	34200	237.5
7782	CLARK	2450	29400	204.16
7566	JONES	2975	35700	247.91
7788	SCOTT	3000	36000	250
7902	FORD	3000	36000	250
7369	SMITH	800	9600	66.66

The above query performs multiplication and division operation on each and every value of the salary column and displayed.

SQL Bitwise operators

Bitwise operators apply true false conditions on the individual binary bits of numbers

- **Bitwise AND(&)**: Returns ***true you only are both input bits are true*** otherwise false
- **Bitwise OR(|)**: Returns ***true if either of the input bits is true*** otherwise false
- **Bitwise XOR(^)**: Returns ***true only if both the input bits are different***

SQL comparison(Relational) operators

Comparison operators are generally used along with where clause ***for filtering the data as per the required condition*** specified Assume variable ***a*** holds 10 and variable ***b*** holds 20 then

Operator	Example
=	(a = b) is not true.
!=	(a != b) is true.
>	(a > b) is not true.
<	(a < b) is true.
>=	(a >= b) is not true.
<=	(a <= b) is true.

!<	(a !< b) is false.
!>	(a !> b) is true.

Example

```
select empno,ename,sal,job
from emp
where sal>2500;
```

O/P

```
5 rows selected.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	30
7566	JONES	MANAGER	7839	02-APR-81	2975	20

Here two conditions first employee must be a manager at the same time he must be getting salary greater than 2500 then only condition becomes true and the record gets displayed.

OR operator

Among multiple conditions specified in the ***where*** clause ***the transaction is performed if any of the condition becomes true***

Example

```
select * from emp
where JOB='ANALYST' OR JOB='MANAGER';
```

O/P

```
5 rows selected
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	10
7566	JONES	MANAGER	7839	02-APR-81	2975	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	20

In the example above, the data is fetched if the employee is an analyst or a manager i.e data is fetched even if any one condition is satisfied.

SQL Special Operators

IN operator

It is used to specify a set of values and operation n is performed on all the values specified in the set and ***if any of the value that is present in the list matches with the values present in a table then it returns true and is operation is performed***

Example

```
select empno,job,sal,hiredate from emp
where ename in('SCOTT','FORD','SMITH','JONES')
```

O/P

```
4 rows selected.
```

EMPNO	JOB	SAL	HIREDATE
7566	MANAGER	2975	02-APR-81
7788	ANALYST	3000	19-APR-87
7902	ANALYST	3000	03-DEC-81
7369	CLERK	800	17-DEC-80

Records of all those employees that are specified in the list of ***in*** are displayed.

BETWEEN Operator

It is used to ***perform data comparison and manipulation over a range of value***s present in the database table

Example

```
select empno,job,sal,hiredate
from emp
where sal between 800 and 2900;
```

O/P

```
3 rows selected.
```

EMPNO	JOB	SAL	HIREDATE
7698	MANAGER	2850	01-MAY-81
7782	MANAGER	2450	09-JUN-81
7369	CLERK	800	17-DEC-80

- The details of those employees whose salary is present in the range between 800 to 2900 retrieved and it also considered specified values inclusive of the range
- In the case of between operator lower value is first specified and followed by the higher value & ***and*** operator in between this higher and lower values.

LIKE operator

The like operator is a pattern matching operator and ***returns those records that match the specified pattern***

Example

```
select empno,ename,hiredate,sal,job
from emp
where ename like 'S%';
```

O/P

```
2 rows selected.
```

EMPNO	ENAME	HIREDATE	SAL	JOB
7788	SCOTT	19-APR-87	3000	ANALYST
7369	SMITH	17-DEC-80	800	CLERK

All the records of employees whose name starting with letter ‘S’ are displayed.

IS NULL operator

All ***operations upon null values present in the table must be done using this ‘is null’ operator*** .we cannot compare null value using the assignment operator

Example

```
select * from emp
where comm is null
```

O/P

```
4 rows selected.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20

The details of those employees whose commission value is Null are displayed.

NOT operator

Not operator is a negation operator which is used along with ***like between, is null, in*** operators, ***It performs reverse r action of all these operators.***

IS NOT NULL

```
select * from emp
where comm is not null;
```

O/P

```
3 rows selected.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	500	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	500	10
7369	SMITH	CLERK	7902	17-DEC-80	800	500	20

Details of all those employees whose Commission value is not null value are displayed.

NOT BETWEEN

```
select * from emp
where sal NOT between 800 and 2900
```

O/P

```
4 rows selected.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10

7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20

The details of those employees whose salary does not fall in the range between 800 to 2900 displayed.

NOT LIKE

```
select * from emp
where ename NOT like 'S%';
```

O/P

```
5 rows selected.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	500	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	500	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20

The details of all those employees whose name doesn't start with letter 'S' are displayed.

NOT IN

```
select * from emp
where ename not in('SCOTT','FORD','SMITH','JONES');
```

O/P

```
5 rows selected
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	500	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	500	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20

The details of all employees whose names are not among the list are displayed

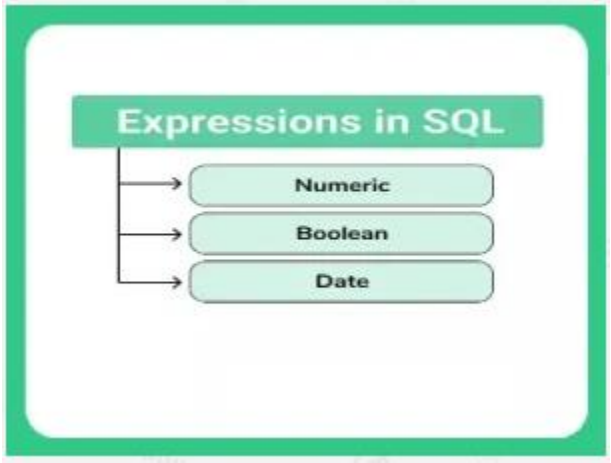
SQL Expressions in DBMS

SQL Expressions in DBMS

An expression is a combination of data, operators and other functions which finally computes to obtain a value

Basic Syntax for SQL expressions usage

```
SELECT column1, column2, columnN
FROM table_name
WHERE [CONDITION|EXPRESSION];
```



There are ***three different types of SQL expressions***, which are mentioned below

- Boolean
- Numeric
- Date

Consider the following sample table ***emp***

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	500	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	500	10

7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20
7369	SMITH	CLERK	7902	17-DEC-80	800	500	20

Boolean Expressions

- Boolean expressions generally correspond to a condition specified in the ***where*** clause for retrieving the required set of data
- Whenever the ***data you specified in the expression is matched with the data in the table then the expression evaluates to true*** and the record is obtained

Syntax

```
SELECT column1, column2, columnN
FROM table_name
WHERE SINGLE VALUE MATCHING EXPRESSION;
```

Example

```
select empno,ename,job,deptno,sal
from emp
where deptno=20;
```

O/P

```
4 rows selected
```

EMPNO	ENAME	JOB	DEPTNO	SAL
7566	JONES	MANAGER	20	2975
7788	SCOTT	ANALYST	20	3000
7902	FORD	ANALYST	20	3000
7369	SMITH	CLERK	20	800

The above query displays the rows of the table only if there is a match for the department number as specified in where clause i.e only when dept-no is 20.

Numeric expression

These expressions are used to ***perform arithmetical calculations on the data*** present in the table

Syntax

```
SELECT numerical_expression as OPERATION_NAME
[FROM table_name
WHERE CONDITION] ;
```

Example

```
select ename,job,sal,sal*12 as annualsalary ,sal/4 as bonus from emp;
```

O/P

```
7 rows selected.
```

ENAME	JOB	SAL	ANNUALSALARY	BONUS
KING	PRESIDENT	5000	60000	1250
BLAKE	MANAGER	2850	34200	712.5
CLARK	MANAGER	2450	29400	612.5
JONES	MANAGER	2975	35700	743.75
SCOTT	ANALYST	3000	36000	750
FORD	ANALYST	3000	36000	750
SMITH	CLERK	800	9600	200

The above query perform multiplication and division operation on the data of the ***salary column*** and displayed.

Date expression

- Date expression ***performs manipulation on 'date data type 'values***
- It is also used to return the current system, date timestamp and arithmetic operations on date values

Example

```
SELECT CURRENT_TIMESTAMP,sysdate from dual;
```

O/P

CURRENT_TIMESTAMP	SYSDATE
08-JUN-19 10.07.51.709047 AM US/PACIFIC	08-JUN-19

Example

We can also perform ***arithmetic operations on date values***

```
select ename,job,sal,hiredate,hiredate+2 from emp;
```

O/P

```
7 rows selected.
```

ENAME	JOB	SAL	HIREDATE	HIREDATE+2
KING	PRESIDENT	5000	17-NOV-81	19-NOV-81
BLAKE	MANAGER	2850	01-MAY-81	03-MAY-81
CLARK	MANAGER	2450	09-JUN-81	11-JUN-81
JONES	MANAGER	2975	02-APR-81	04-APR-81
SCOTT	ANALYST	3000	19-APR-87	21-APR-87
FORD	ANALYST	3000	03-DEC-81	05-DEC-81
SMITH	CLERK	800	17-DEC-80	19-DEC-80

Hiredate+2 is nothing but 2 days from current date value is advanced suppose it hiredate is *july 24* then *hiredate +2* will give *July 26*

Create Table in DBMS

CREATE TABLE

A database is nothing but the structured organization of data. For organizing the data in a database we need to create database tables as per the required structure

CREATE is the DDL(data definition language) commands used for the creation of the tables in a database

Syntax:

```
CREATE TABLE table_name(
column1 datatype,
column2 datatype,
.....
columnN datatype,
PRIMARY KEY(one or more columns)
);
```

CREATE TABLE

- ***Create*** command is used to create a table in the database with the structure specified by the user
- This structure includes the ***number of columns to be present in the table and the data type of the column, size of data,*** etc

Basic Syntax for CREATE

```
CREATE TABLE table_name
(
column1 datatype(size),
column2 datatype(size),
column3 datatype(size),
.....
columnN datatype(size),
PRIMARY KEY( one or more columns )
);
```

Example

```
create table emp(
empno number(4,0),
ename varchar2(10),
job varchar2(9),
mgr number(4,0),
hiredate date,
sal number(7,2),
deptno number(2,0)
PRIMARY KEY (ID)
);
```

- **Table name**:***emp***

- **Column names**: In the above table that we have created have ***7 columns namely empno, ename, job, mgr,hiredate, sal,deptno***
- **Data types:** What ***type of data should be entered for each column value***. for example, we have used ***number*** data type for the column empno , which means you must enter numerical values only while inserting data for emp column
- ***Size***: ***Specifies the length of the value that is inserted***, for example, we have used size 10 for ename as varchar2(10), which means the maximum number of characters that can be entered for the ename column is 10.

DESC command

- Describe command shows the ***structure of the table that we have created***
- It displays the column names, data types of column names, size of data , any constraints imposed the table, default values for each column, whether null values allowed or not for each column, etc

Example

```
desc emp;
```

O/P

Field	Type	Null	Key	Default	Extra
EMPNO	number(4,0)	N0	PRI		
ENAME	varchar2(10)	YES	–	NULL	
JOB	varchar2(9)	NO	–	NULL	
MGR	number(4,0)	NO	–	NULL	
HIREDATE	date	NO	–	NULL	
SAL	number(7,2)	NO	–	NULL	
DEPTNO	number(2,0)	NO	–	NULL	

Creating a new table from an existing table

- We c***an create a new table with exactly the same structure of any of the existing tables***
- Here an empty table with exactly the same structure of the existing table is created but data and constraints of the table are not copied

Example

```
create table
sample as (select * from emp);
```

O/P

```
table sample crated successfully
```

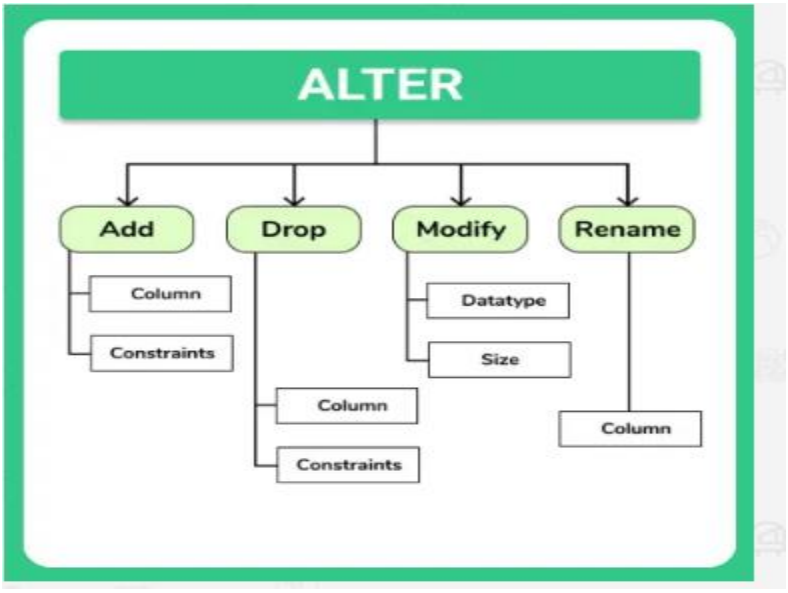
Not both ***sample*** and ***emp*** tables will have the same structure i.e same number of columns, column names, datatype, sizes, etc .you can also cross-check by using desc command.

ALTER

ALTER TABLE (ADD, DROP, MODIFY, RENAME)

In this article, we will learn about ALTER in DBMS.

- Alter command in SQL is used to make modifications to the columns in the existing table
- It is used to add columns, delete columns, drop constraints, renaming the columns, changing the data type and data type size of the column existing in the table.



ALTER Command: ADD

- ADD command is ***used to add one or more new columns to the existing database tables***
- The ***newly added columns will be empty*** and data can be entered by using insert command

Syntax

```
ALTER TABLE table_name ADD(column_name datatype);
```

Adding a single column

```
ALTER TABLE student ADD(address VARCHAR(100));
```

The above query will ***create a new column in the student table named address with varchar data type*** where each value in the address column can hold a maximum length of 200 characters

Adding multiple columns

Any number of columns can be added to the existing table by using add command separated with commas

```
ALTER TABLE student ADD (
father name VARCHAR(60),
mother name VARCHAR(60),
DOB DATE);
```

Adding constraints

We ***can add constraints like a primary key, foreign key*** to the existing table anytime

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

Now the ***ID column will become the primary key*** of the table

Adding a column with a default value

We ***can add a column to an existing table by applying the default clause***

```
ALTER TABLE student ADD(
gender char(1) default 'M'
);
```

A ***new column will be created*** with the name ***gender of char data type*** where all the values in that column ***will be by default 'M'*** which ***can also be customized*** using insert command.

ALTER command: MODIFY

By using the modify command, ***we can change the data type of the existing column or the size of the data type*** of the existing column

Syntax

```
ALTER TABLE table_name MODIFY (
column_name datatype
);
```

Modifying a single column

```
ALTER TABLE student MODIFY(
address varchar(75));
```

The above query ***changes the data type of the address column as varchar and size to 75*** overriding the existing size value and data type

Modifying multiple columns

We ***can change data types or sizes of the columns of multiple columns*** at a time separating with commas

```
ALTER TABLE student MODIFY(
address varchar(75),dob date);
```

Note:

- Modify ***can be used to change the data type of the existing column provided the new data type must be compatible*** with the existing column data ***if the existing column is empty then it can change it to any data type*** i.e a column that is consisting of character data cannot be changed it to number data type

- The ***size of the existing data type of a column can be increased without any condition*** but ***the size is decreased so that new size should be sufficient to all values in the existing column*** i.e size cannot be decreased not less than the size accommodated by the largest value present in the column
- For example, if the city column contains Hyderabad consisting of nine characters now the size can be increased to any value but cannot be decreased less than 9.

Alter command: Rename

Rename command in combination alter is ***used to rename the existing column name of a table***

Syntax

```
ALTER TABLE table_name RENAME
old_column_name TO new_column_name;
```

Example

```
ALTER TABLE student RENAME
address to location;
```

- Column ***address has been renamed to location***
- Now all ***manipulations on this column must be performed using the new name location.***

ALTER command: DROP

- Alter command is ***used to delete one or more existing columns*** present in the table
- We ***can also drop more than one column*** by separating the column names with commas

Syntax

```
ALTER TABLE table_name DROP(
column_name);
```

Dropping a single column

```
ALTER TABLE student DROP(
address);
```

Address column will be removed from the student table

Dropping multiple columns

We can also drop more than one column by separating the column names with commas

```
ALTER TABLE student DROP(
  Caste, religion);
```

Note:

In all databases we cannot drop all the columns using this drop command which means if there exists only one column in the table then drop command cannot be used to drop that column

Dropping constraints

Drop command is ***used to remove the constraints imposed on the columns***

```
ALTER TABLE emp
DROP CONSTRAINT PK_id;
```

For suppose ID is a primary key column of the table emp which is given a name of PK_id, ***upon dropping PK_id will remove the primary key of the table***

DROP/TRUNCATE/RENAME in DBMS

DROP/TRUNCATE/RENAME In DBMS

1. Dropping a table using DROP in DBMS

- Drop command delete the table existence completely i.e ***drop statement destroys the existing database object*** of that particular table, index or view
- After dropping a table if you try to use the table then compiler shows an error as " ***table or view does not exist***"

Syntax

```
DROP TABLE table_name;
```

Example

```
drop table emp;
```

2. TRUNCATE command

- The truncate command will make the table empty i.e all the table data will be deleted but the structure and database object is still alive ***and the table can be reused normally***
- The truncate command logically nothing but using delete command for deleting the records in the table without specifying the where condition i.e all the rows get deleted in that case

Syntax

```
TRUNCATE TABLE table_name;
```

Example

```
truncate table emp;
```

O/P

```
Table emp truncated successfully
```

After truncating the table if you try to display the table data using a select statement then you'll get a message as" ***0 rows selected***" i.e nothing to display because the table is empty without any rows

3. Renaming a table: RENAME COMMAND

The rename command is used to change the existing table name and give a new name to the table.

Syntax

```
rename old_table _name to new_table_name
```

Example

```
rename emp to Employees
```

O/P

```
Table emp  renamed successfully
```

The above query changes the emp table name from ***emp to employees*** all operations on the table must be done using the new name ***employees*** otherwise it will raise an error.

Queries

SELECT Query in DBMS

SELECT Query in DBMS

- A SELECT query is used to retrieve data(records) from the table
- We can retrieve complete table data, or specific records by specifying conditions using the WHERE clause.

Syntax

```
SELECT column1, column2, ...
FROM table_name;
```

Displaying only specific columns using SELECT Query in DBMS

Consider a table ***student*** where you need to display a student name, ID, and score

```
SELECT stuid,sname,score
from student;
```

O/P

stuid	sname	score
66	Trishaank	92
82	Srinivas	62
73	Prashanth	78
79	Sanjay	85

91	Chandana	90

Display all records present in the table using SELECT Query in DBMS

The '*' operator is used to display all the columns present in the table

```
SELECT * from student;
```

stuid	sname	branch	dob	score
66	Trishaank	computers	24-07-1998	92
82	Srinivas	computers	7-07-1996	62
73	Prasanth	Physics	17-08-1997	78
79	Sanjay	Maths	12-09-1997	85
91	Chandana	Biology	7-05-1997	90

All the columns in the table get displayed.

Retrieve records based on some specified condition

Only those ***records that satisfy the condition specified in the 'where' clause are displayed***

Display records of those students from the computers department

```
SELECT * from
student
where branch='computers';
```

stuid	sname	branch	dob	score
66	Trishaank	computers	24-07-1998	92
82	Srinivas	computers	7-07-1996	62

O/P

Performing some calculations on the column using SELECT

- We can use ***simple arithmetic operators upon column data while displaying records***
- But these changes are not affected in the actual database table they are just used for display purpose

Increase score of students by 5 marks who are having a score greater than 80

```
SELECT stuid,sname,score,score+5
from student
where score>80;
```

O/P

stuid	sname	score	score+5
66	Trishaank	92	97
79	Sanjay	85	90
91	Chandana	90	95

5 marks will be added to all students who are having 'score' greater than 80 are displayed

INSERT Query in DBMS

INSERT Query in DBMS

- INSERT is a widely used data manipulation language(DML)command for adding new data to the existing database table
- Insert command is used to add one or more rows of data to the database table with specified column values

Syntax for INSERT

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Consider a sample table ***student,*** initially as empty.

Simple INSERT Query in DBMS

- In this type of insert, ***all the values should be provided to all the columns that exist in the table without specifying the column name***
- The order of values declared in the values clause should follow the original order of the columns in the table

Example

```
insert into student
 values(66'trishaank','computers','24-07-1998',92);
```

O/P

```
1 row added
```

STUID	SNAME	BRANCH	DOB	SCORE
66	Trishaank	computers	24-07-1998	92

Inserting data into required columns

In this case, the order of columns declared in insert need not be the same as that of the original table order, ***only the specified column values are added and null values added to the remaining column values in that tuple***

Example

```
insert into student(stuid,sname,branch) values (82,'Srinivas','Computers');
```

O/P

```
Number of Records: 2
```

STUID	SNAME	BRANCH	DOB	SCORE
66	Trishaank	computers	24-07-1998	92
82	Srinivas	computers	null	null

In this case, the row added, values are inserted only for studid,sname, and branch, whereas for dob and score null values for inserted.

Dynamic INSERT Query in DBMS (using ampersand &)

In this type of insert, the ***values are entered by the user at the execution time***

Example

```
insert into student values(&stuid,'&sname','&branch','&dob',&score);
```

Action

```
enter value for stuid:73
enter value for sname:Prashanth
```

```
enter value for brance:Physics
enter value for dob:17-08-1997
enter value for score:78
```

O/P

```
Number of Records: 3
```

STUID	SNAME	BRANCH	DOB	SCORE
66	Trishaank	computers	24-07-1998	92
82	Srinivas	computers	null	null
73	Prashanth	Physics	17-08-1997	78

Using select in the INSERT command

Copying all columns of a table

- We can ***copy the data(rows) from one table and insert into another table by using this combination of select and insert***
- Consider the table ***sample1*** with following data and ***sample2*** which is empty as of now with the same structure as sample1, Now we can copy all the rows present in the ***sample1*** table to ***sample2*** table

Sample 1 table

Number of Records: 5

NAME	AGE
trish	20
prash	21
sanju	22
srinu	21
Chandana	20

Example

```
INSERT INTO sample2 SELECT * FROM sample1;
```

O/P

```
select * from sample2;
```

Number of Records: 5

NAME	AGE
trish	20
prash	21
Sanju	22
srinu	21
Chandana	20

Copying specific columns of a table

You ***can also copy only required columns from one table to another table*** using *insert into and select* combination

```
INSERT INTO sample2(name) SELECT name FROM sample2;
```

O/P

```
select * from sample2;
```

Number of Records: 5

NAME	AGE
trish	null
prash	null
Sanju	null
srinu	null
Chandana	null

Here only name column present in ***sample1*** gets copied into ***sample2***

Copying specific rows from a table

You can ***add only specific records by filtering the rows based on the condition specified in the where clause***

```
INSERT INTO sample2 SELECT * FROM sample1
WHERE age=20;
```

O/P

```
select * from sample2
```

Number of Records: 2

NAME	AGE
trish	20
Chandana	20

Only those rows where age is 20 from ***sample1*** are copied into the ***sample2*** table

WHERE Query in DBMS

WHERE Query in DBMS

- WHERE clause is used to specify a condition while retrieving and updating data from the database table and display only those records in the table for which the condition specified in the where clause becomes true
- The ***WHERE*** clause is most commonly used with a select, update and delete statements

Syntax:

```
SELECT Column1, Column2.....
From table_name
WHERE condition;
```

Consider the sample table *emp*

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	–	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20

7369	SMITH	CLERK	7902	17-DEC-80	800	–	20

Display the records of those employees whose designation is 'MANAGER'

```
select * from emp
where job='MANAGER';
```

O/P

3 rows selected.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	–	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20

In our ***emp*** table, there are three *managers* that are why only three records are displayed, because the condition is true for only those three rows i.e when the job is a ***manager.***

Display name, job, the salary of that employee whose salary is greater than 2900

```
select ename,job,sal
from emp
where sal>2900;
```

ENAME	JOB	SAL
KING	PRESIDENT	5000
JONES	MANAGER	2975
SCOTT	ANALYST	3000
FORD	ANALYST	3000

O/P

You can also display only specified columns and filter the records

Operators that can be used in Where query in DBMS

Not only comparison(=) operator ***many operators shown below can be used with where clause***

Operator	Description
=	Equal to

!=	Not Equal to
<	Less than
>	Greater than
<=	Less than or Equal to
>=	Greater than or Equal to
BETWEEN	Between a specified range of values
LIKE	This is used to search for a pattern in value.
IN	In a given set of values

Display the records of all employees other than managers

```
select ename,sal,job,deptno from emp
where job != 'MANAGER';
```

O/P 4 rows selected

ENAME	SAL	JOB	DEPTNO
KING	5000	PRESIDENT	10
SCOTT	3000	ANALYST	20
FORD	3000	ANALYST	20
SMITH	800	CLERK	20

Display records of all employees whose salary is between 1000/- to 2500/-

```
select ename,sal,job,deptno from emp
where SAL between 1000 AND 2500;
```

- Here we can make use of ***between*** operator ***where the condition is applied for all the values within the specified range***

- In the above example all the records of employees whose salary in between the range of 2000 and 2500 is displayed(inclusive of 2000 and 2500)
-

Deleting records based on the condition specified in where clause

Conditions can be specified by using where the ***where clause for deletion and updation of data***

```
delete from emp
where empno=7934;
```

O/P

```
1 row deleted
```

AND & OR in DBMS

AND & OR in DBMS

- Sometimes user requires ***more than one condition for filtering the data***, this purpose is served by AND / OR clauses
- The AND and OR operators are used with the where clause for precise filtration of data from the database tables by combining more than one condition along with select, update and delete queries

Syntax:

```
SELECT Column1, Column2....
FROM table_name
WHERE
condition1 AND/OR condition2,
AND/OR condition4.....;
```

AND clause

Definition: The AND results ***true*** only when ***all the conjunction of conditions specified after the where clause are satisfied***

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

Consider a sample table *emp*

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO

7839	KING	PRESIDENT	–	17-NOV-81	5000	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	10
7566	JONES	MANAGER	7839	02-APR-81	2975	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	20
7369	SMITH	CLERK	7902	17-DEC-80	800	20

Display the records of those employees who are working as a manager and getting salary greater than 2500 /-

```
select * from emp
where job='MANAGER' AND sal>2500;
```

O/P 2 rows selected.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	30
7566	JONES	MANAGER	7839	02-APR-81	2975	20

Here two conditions first employee must be a manager at the same time he must be getting salary greater than 2500 then only condition becomes true and the record gets displayed.

OR clause

Definition: Among multiple conditions specified in the ***where*** clause ***the transaction is performed if any of the condition becomes true***

Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...
```

Display all employees records who are working as analyst and managers

```
select * from emp
where JOB='ANALYST' OR JOB='MANAGER';
```

O/P **5 rows selected**

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	10
7566	JONES	MANAGER	7839	02-APR-81	2975	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	20

In the example above, the data is fetched if the employee is an analyst or a manager i.e data is fetched even if any one condition is satisfied.

Combining AND and OR

In the where clause it is possible that a ***condition is specified as a conjunction of both AND & OR***

Syntax:

```
SELECT * FROM table_name
WHERE condition1 AND|OR (condition2 AND|OR condition3....);
```

Example

```
select * from emp
where (sal>1500 OR job='MANAGER') AND (deptno=10 OR deptno=30);
```

O/P 3 rows selected.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	10

The above query displays the details of employees having a salary greater than 1500/- or if he is a manager which is exclusively from 10th dept and 30th departments.

UPDATE Query in DBMS

UPDATE Query In DBMS

- The UPDATE statement is used to modify or change the data of the existing table in the database
- We can update a single column as well as multiple columns as per our requirement

The general syntax of UPDATE

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

- The ***SET*** is Is used to set new values to the required column and the where Clause is used to filter the rows for which rows of data are needed to be updated

Updating a single column

Any row in the database table can be updated using update statement

Consider a sample table EMP as shown below

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	–	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20
7369	SMITH	CLERK	7902	17-DEC-80	800	–	20

Update the commission of all employees to 200/- whose salary is less than 3000/-

```
UPDATE emp SET comm=200
WHERE sal<3000;
```

O/P

```
7 ROWS SELECTED
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	200	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	200	10
7566	JONES	MANAGER	7839	02-APR-81	2975	200	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20
7369	SMITH	CLERK	7902	17-DEC-80	800	200	20

Initially, in the EMP table, values are NULLfor comm column for all the rows but using update statement we have changed the commission to 200 from NULL for those employees who are having salaries less than 3000.

Updating multiple columns

Using a single update statement with can ***parallelly update any number of column***

Update salary by 500 and change the commission to 1000 for all analysts working in the company

```
update emp set sal=sal+500 ,comm=1000
where job='ANALYST';
```

O/P

```
2 rows selected.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7788	SCOTT	ANALYST	7566	19-APR-87	3500	1000	20
7902	FORD	ANALYST	7566	03-DEC-81	3500	1000	20

We have two analysts, hence two rows will be updated i.e salary will be increased from 3000 to 3500 and commission will be changed to 1000 i.e both salary and commission columns get updated.

Updating without where clause

If you don't specify where clause, it means that no condition is required and no data filtering happens and ***all the rows present in the table will be updated for that column as specified***

Make the salary of all employees working in the company to 500

```
update emp
set sal=500;
```

O/P

```
7 rows selected
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	500	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	500	200	30
7782	CLARK	MANAGER	7839	09-JUN-81	500	200	10
7566	JONES	MANAGER	7839	02-APR-81	500	200	20
7788	SCOTT	ANALYST	7566	19-APR-87	500	–	20
7902	FORD	ANALYST	7566	03-DEC-81	500	–	20
7369	SMITH	CLERK	7902	17-DEC-80	500	200	20

We have seven rows present in the table, the value of a salary in every row becomes 500 because there is no condition for data change, hence all the rows will be updated

DELETE Query in DBMS

In Real world scenario if the user wants to delete his Facebook account the user just click the delete option but at the backend, DELETE command is executed and the record with particular details is been deleted

DELETE statement is used to delete single or multiple records present in the existing database table based on a specific condition

Basic Syntax for the DELETE statement

```
DELETE FROM table_name
WHERE condition;
```

Deleting a single record

Where clause is used to provide a condition for deleting a particular record in the table

Consider the sample table EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	–	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20
7369	SMITH	CLERK	7902	17-DEC-80	800	–	20

Delete the record of employee number 7698

```
delete from emp
where empno=7698;
```

O/P

```
1 ROW DELETED
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	–	30

- A single employee present in the company with employee number 7698 will be deleted from the database table
- The previous table we have 7 rows but here we are having only 6 records i.e one record has been deleted from the database table.

Deleting multiple records

Delete statement can be used to ***delete multiple rows at a time in a table***

***Delete the records of all employees whose salary is greater than 2500 expect the* president**

```
delete from emp
where sal>2500  and job!=  'PRESIDENT';
```

O/P

```
4 ROWS DELETED
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	–	30
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20

In the previous table, we have 7 records but after the execution of this query we are left with only three records i.e all those employees who are having a salary greater than 1000 except ***president*** will be deleted.

Deleting all records

- ***All the rows present in the table will be deleted if there is no condition specified by using where clause***
- In this type of delete, the table will become empty and no records will be present to display

```
delete from emp;
```

O/P

```
7 ROWS DELETED
```

- All the seven rows present in the ***emp*** table will be deleted and row-count becomes zero i.e nothing to display.

DELETE V/S TRUNCATE

- Both these commands **delete** and **truncate** make a table empty but there are certain differences where a super learner must be aware of

TRUNCATE command usage

```
truncate table emp;
```

- It removes all rows from a Table and makes a table empty
- This **operation cannot be rolled back (table cannot be restored) and no triggers will be fired**
- You ***cannot use a where clause and delete particular records you need to delete enter table data***
- Truncate is faster and does not use a match under space as delete

DELETE command usage

```
delete from emp;
```

- Delete command is also used to remove rows from a table
- ***You can delete whether only particular records by using a condition with where a clause or entire rows can be deleted without using where class***
- After performing delete operation the ***transactions can be rollbacked i.e that is a table can be restored*** *after deleting*
- To make the changes permanent in a delete operation need to commit the transaction commit or rollback

LIKE in DBMS

LIKE IN DBMS

- If you want to ***search all employees starting with letter P or names of all products which which consists of exactly 4 letter***s etc
- LIKE in DBMS operator used to search specified pattern in the data and retrieve the record when there is a pattern match as required

General Syntax for LIKE

```
SELECT column1, column2, ...
FROM table_name
WHERE column LIKE pattern;
```

Wildcard operators

% : ***Percent(%)***represents 0,1 or multiple characters **_** : ***Underscore(_)*** is used to match exactly a single character

This wild card operator is used in conjunction with where clause and filter the records based on specified patterns as required.

Different ways that we can use this LIKE clause

PATTERN	MEANING
'r%'	Matches strings which start with 'r'
'%r'	Matches strings with end with 'r'
'r%t'	Matches strings which contain the start with 'r' and end with 't'.
'%tri%'	Matches strings which contain the substring 'tri' in them at any position.
'_tri%'	Matches strings which contain the substring 'tri' in them at the second position.
'_r%'	Matches strings which contain 'r' at the second position.
'r_%_%'	Matches strings which start with 'r' and contain at least 2 more characters.

The following are the rules for pattern matching with the LIKE Clause:

Sample table *EMP*

ENAME	HIREDATE
KING	17-NOV-81
BLAKE	01-MAY-81
CLARK	09-JUN-81
JONES	02-APR-81
SCOTT	19-APR-87
FORD	03-DEC-81
SMITH	17-DEC-80
ALLEN	20-FEB-81
WARD	22-FEB-81
MARTIN	28-SEP-81
TURNER	08-SEP-81
ADAMS	23-MAY-87
JAMES	03-DEC-81
MILLER	23-JAN-82

Display the employees whose name start with 'M'

```
select ename  from emp
where ename like 'M%';
```

O/P

```
2 rows selected.
```

ENAME
MARTIN
MILLER

All employee names that start with letter **M** are displayed.

Display the names of all employees having M in any position in their name

```
select ename from emp
```

```
where ename like '%M%';
```

O/P

```
5 rows selected.
```

ENAME
SMITH
MARTIN
ADAMS
JAMES
MILLER

The above query discuss the employee name that contains one or more M anywhere in their names.

Display the names of employees whose name contain the second letter as L

```
select ename from emp
where ename like '_L%';
```

O/P

```
3 rows selected.
```

ENAME
BLAKE
CLARK
ALLEN

All string second letter as L are displayed.

Display the names of employees which contains the fourth letter as M

```
select ename from
emp where ename like '___M%';
```

O/P

```
1 ROW SELECTED
```

ENAME
ADAMS

In the above query, we have used 3 underscores(___) followed by letter M which mean that fourth letter should be M, like this, you can search for a specific character at any position based upon the number of underscores(_).

Display the employee names and hire dates for the employees joined in the month of December

```
select ename,hiredate from emp
where hiredate LIKE '%DEC%';
```

O/P

```
3 ROWS SELECTED
```

ENAME	HIREDATE
FORD	03-DEC-81
SMITH	17-DEC-80
JAMES	03-DEC-81

For all the values of ***hire date*** wherever it found a string with 'DEC', all those records are displayed.

Display names of all employees whose name contains exactly 4 letters

```
select ename from emp
where ename like '____';
```

O/P

```
3 rows selected
```

ENAME
KING
FORD
WARD

All employee names which contains four letters are displayed. Here we have used 4 underscores, you can display required length strings based on number of underscores (_).

Display the names of all employees whose name does not contain 'A' anywhere

```
select ename from emp
where ename not like '%A%';
```

O/P

```
7 rows selected
```

ENAME
KING
JONES
SCOTT

FORD

SMITH

TURNER

MILLER

Here all the names that are free from letter ‘A’ displayed

ORDER BY in DBMS

ORDER BY in DBMS

The order by clause is used to arrange the fetched data from the database table in ascending or descending order of data values based on one or more columns

- Sometimes the user may be interested in ***arranging the data in the table in some increasing or decreasing order of values***
- **Example**: If you want to display the details of all students based on descending order of their attendance or marks etc

Syntax

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

ASC: Displays ***data based on increasing order of values in the column DESC***: Displays ***data based on decreasing order of values in the column*** **Consider the following sample *EMP* table**

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	10
7566	JONES	MANAGER	7839	02-APR-81	2975	20

7788	SCOTT	ANALYST	7566	19-APR-87	3000	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	20
7369	SMITH	CLERK	7902	17-DEC-80	800	20

Sorting according to a single column

Here we ***use only a single column for data arrangement***

Display the names, salaries of all employees based on decreasing order of their salaries

```
select ename,sal from emp
ORDER BY sal DESC;
```

O/P

```
7 ROWS SELECTED
```

ENAME	SAL
KING	5000
FORD	3000
SCOTT	3000
JONES	2975
BLAKE	2850
CLARK	2450
SMITH	800

The row which contains the highest salary is displayed first, the row with the second highest salary is displayed next and so on

Display the department number and employee name as per increasing order of department numbers

```
select deptno,ename from emp
ORDER BY deptno;
```

O/P

```
7 rows selected
```

DEPTNO	ENAME
10	KING
10	CLARK
20	JONES
20	SMITH
20	SCOTT
20	FORD
30	BLAKE

As we have 3 departments in the employee table, first the rows of 10th department followed by 20 and 30 departments are displayed i.e increasing order of department numbers

Note:

ASC is optional and the Default value for order by, If you don't specify ASC or DSC by default data is arranged in ascending order, but for descending arrangement of data you must specify DESC in order by explicitly

Display all employees for working in a company based on seniority level

```
select ename,job,sal,hiredate
from emp
order by hiredate desc;
7 rows selected
```

ENAME	JOB	SAL	HIREDATE
SCOTT	ANALYST	3000	19-APR-87
FORD	ANALYST	3000	03-DEC-81
KING	PRESIDENT	5000	17-NOV-81
CLARK	MANAGER	2450	09-JUN-81
BLAKE	MANAGER	2850	01-MAY-81

JONES	MANAGER	2975	02-APR-81
SMITH	CLERK	800	17-DEC-80

Employees who are having an older hire date is displayed first followed by the second older hire date and so on.

Sorting according to more than one column

You can ***specify more than one column in the order by clause this is done when you further need sorting internally based on a certain column***

Display the details of employees based on increasing order of Departments and in each department salary should be further arranged in highest to lowest order

```
select ename,sal,deptno from emp
ORDER BY deptno,sal DESC;
```

O/P

```
7 rows selected
```

ENAME	SAL	DEPTNO
KING	5000	10
CLARK	2450	10
FORD	3000	20
SCOTT	3000	20
JONES	2975	20
SMITH	800	20
BLAKE	2850	30

Before displaying the rows of30th department, rows or sorted based on the salaries of30 department in decreasing order i.e that is first highest salary in the 30th department and the second-highest salary in the 30th department followed by the 20th department the first highest salary followed by 20th department second highest salary and so on.

Using where clause in ORDER BY

- You can also ***specify some conditions and filter the data by using where class and then sort the data***

- Here first ***where*** clause is to be followed by ***order by*** class.

Display the names, sal, jobs of employees who are working as a manager in highest to lowest order

```
select ename,job,sal from emp
where job='MANAGER'
order by sal desc ;
```

O/P

```
3 rows selected
```

ENAME	JOB	SAL
JONES	MANAGER	2975
BLAKE	MANAGER	2850
CLARK	MANAGER	2450

Here only records of managers are displayed as per decreasing order of their salaries.

Specifying column numbers in order by

Instead of column names, ***you can also use the position number of columns are specified in the select statement***

```
select ename, sal, job
from emp
order by 2 DESC;
```

O/P

```
7 rows selected
```

ENAME	SAL	JOB
KING	5000	PRESIDENT
FORD	3000	ANALYST
SCOTT	3000	ANALYST
JONES	2975	MANAGER
BLAKE	2850	MANAGER
CLARK	2450	MANAGER

SMITH	800	CLERK

The second column in the select statement is ‘sal’ hence in the above query sorting of rows is done as per increasing order of salaries.

Sorting data based on expressions

User can specify basic arithmetic expressions and sort the rows based on this column generated by the expression.

Display the names, job, annual salary of all employees based on decreasing order of their annual salary

```
select ename ,sal*12 annsal ,job,hiredate
from emp
order by annsal desc;
```

O/P

```
7 ROWS SELECTED
```

ENAME	ANNUAL	JOB	HIREDATE
KING	60000	PRESIDENT	17-NOV-81
FORD	36000	ANALYST	03-DEC-81
SCOTT	36000	ANALYST	19-APR-87
JONES	35700	MANAGER	02-APR-81
BLAKE	34200	MANAGER	01-MAY-81
CLARK	29400	MANAGER	09-JUN-81
SMITH	9600	CLERK	17-DEC-80

Here we have performed operations on the existing ***sal column*** and the rows are sorted based on this newly formed display purpose column.

Group By in DBMS

GROUP BY IN DBMS

If you want to display the total salaries of each department in a company or else to display the highest paid employees of each branch of that company. this purpose is also called a by using a group by clause.

Group by clause in SQL used to arrange logically related data into groups with help of some functions i.e if a particular column has the same type of data in different rows then they can be organized this into a logical groups.

General Syntax for GROUP BY in DBMS

```
SELECT column_name(s),function(column_name)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Simple example for GROUP BY

Consider a sample table *'emp'*

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	–	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20
7369	SMITH	CLERK	7902	17-DEC-80	800	–	20

Display the number of employees present in each department

```
select deptno,count(*)
from emp
group by deptno;
```

O/P

```
3 rows selected.
```

DEPTNO	COUNT(*)
30	1
10	2

20	4

The above query display group wise count of each department

Display the highest played employees in each department

```
select deptno,max(sal)
from emp
group by deptno
```

O/P

```
3 rows selected.
```

DEPTNO	MAX(SAL)
30	2850
10	5000
20	3000

Department wise highest salary i.e logically categorizing category employees into department wise.

Using WHERE clause in the GROUP BY

Using ***where clause rows can be pre excluded before dividing them into groups***, ***where*** clause must be specified before the ***group by*** clause when it is used in the query

Example

```
select deptno,max(sal)
from emp
where deptno!=30
group by deptno;
```

O/P

```
2 rows selected.
```

DEPTNO	MAX(SAL)
10	5000
20	3000

Highest salaries of all departments expect deptno 30 are displayed.

Using ORDER BY with GROUP BY

We can also ***display the rows in sorted order after logical organizing into groups using order by clause*** along with group by clause

Example

```
select deptno,max(sal)
from emp
group by deptno
order by deptno;
```

O/P

```
3 ROWS SELECTED
```

DEPTNO	MAX(SAL)
10	5000
20	3000
30	2850

The above query displays the highest salaries in each department but first deptno 30 details followed by deptno 20 and deptno 30 i.e increasing order of department numbers.

Points to note about GROUP BY clause

- ***GROUP BY clause is used only with the SELECT statement.***
- ***Where class is placed before group by class*** if it is used in the query.
- ***Order by class is placed after the group by class*** if it is used in the query group by class
- ***All the columns that are used in the select statement must be specified by using group by clause***
- ***If a group function is included in the select clause then we cannot use individual result columns.***

HAVING clause

- Having Clause is ***used to place conditions and decide which group will be part of the final result***
- You cannot use aggregate functions like sum() count() etc with where clause
- Hence we need to use having clause if you want to specify conditions using this aggregate functions

Example

```
SELECT ename , SUM(sal) FROM emp
GROUP BY ename
HAVING SUM(sal)>2000;
```

O/P

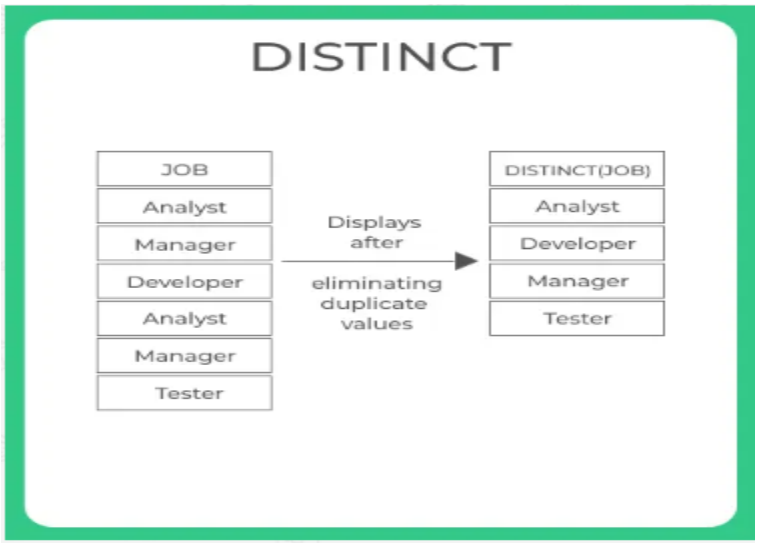
```
6 rows selected.
```

ENAME	SUM(SAL)
JONES	2975
KING	5000
CLARK	2450
SCOTT	3000
BLAKE	2850
FORD	3000

DISTINCT in DBMS

DISTINCT IN DBMS

- Inside table columns may contain many duplicate values and sometimes we require to list only unique values, this is done by using DISTINCT clause along with a select statement
- DISTINCT statement is used to return only unique values present in a column or combination of columns
- The DISTINCT clause is only ***for display purpose and will not affect the original database table***



DISTINCT Syntax

```
SELECT DISTINCT column1, column2,
FROM table name;
```

Consider a sample table EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	–	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20
7369	SMITH	CLERK	7902	17-DEC-80	800	–	20

DISTINCT on a single column

```
select DISTINCT job from emp;
```

Output

```
4 rows selected.
```

JOB

ANALYST

CLERK

MANAGER

PRESIDENT

The above query Returns unique values present in the job column i.e. ***even if a value is appeared for more than once but it is displayed only for one time.***

DISTINCT on more than one column

Whenever ***DISTINCT is applied*** on the select statement which contains ***more than one column then the combination of values of all the columns is considered as a single value*** even if ***any of the column value is different in that row it is considered as a DISTINCT value***

Example

```
select DISTINCT job,sal
from emp;
```

o/p

```
6 rows selected.
```

JOB	SAL
MANAGER	2850
CLERK	800
MANAGER	2975
PRESIDENT	5000
ANALYST	3000
MANAGER	2450

The above query Returns 6 values because each value is considered as a ***key combination of two values and [ANALYST, 3000] combination would appear twice*** and hence it is eliminated during display.

Another example for DISTINCT

```
select DISTINCT ename, job
from emp;
```

Output

6 rows selected

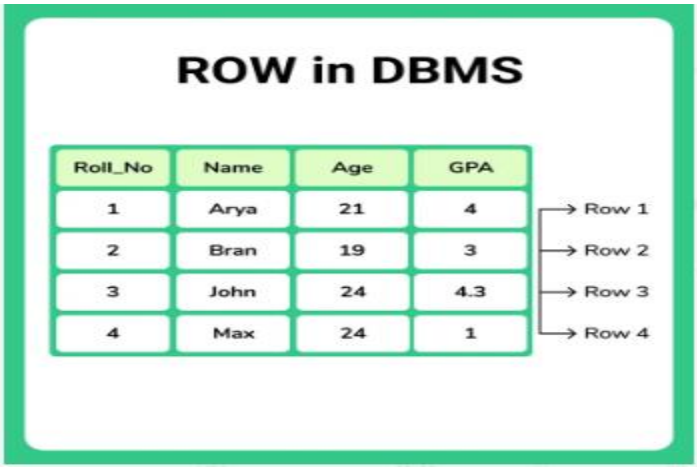
ENAME	JOB
FORD	ANALYST
CLARK	MANAGER
JONES	MANAGER
SMITH	CLERK
KING	PRESIDENT
SCOTT	ANALYST
BLAKE	MANAGER

The above query results in all the seven rows in the table because ***even if job column contains duplicate values name column contain only unique values hence the key combination both ename and job is considered as one entity*** and all values become unique

ROW in DBMS

ROW IN DBMS

- Generally, the database table is viewed as a collection of rows and columns
- The term row is often also called tuple and record in a database table.



ROW

Each row represents a complete record of the specific data item, each row stores different data with the same structure

Example

For example in an employee table, we ***need to store the details of 10 employees then the database table will consist of 10 rows*** every each row contains details of that particular employee i.e 1 row for storing

ENAME	JOB	SAL	HIREDATE	HIREDATE+2
KING	PRESIDENT	5000	17-NOV-81	19-NOV-81
BLAKE	MANAGER	2850	01-MAY-81	03-MAY-81
CLARK	MANAGER	2450	09-JUN-81	11-JUN-81
JONES	MANAGER	2975	02-APR-81	04-APR-81
SCOTT	ANALYST	3000	19-APR-87	21-APR-87
FORD	ANALYST	3000	03-DEC-81	05-DEC-81
SMITH	CLERK	800	17-DEC-80	19-DEC-80

the record of one employee **Consider a sample table '*emp*'**

Table contents seven rows i.e it stores the details of all the 7 employees in 7 separate rows.

SQL Query to retrieve particular rows

```
SELECT * FROM EMP
WHERE DEPTNO=20;
```

O/P

```
4 rows selected.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20

7369	SMITH	CLERK	7902	17-DEC-80	800	500	20

Only those rows corresponding to the details of Department number 20 are displayed.

ROW NUM in SQL

- The ***rownum*** clause in SQL is used to sp***ecify the number of rows present in a database table***
- Each row will be added a number starting from 1 to n so that the user can easily identify the number of rows present in the table

Example

```
select rownum,empno,ename,job,sal,deptno
FROM EMP;
```

O/P

```
7 rows selected.
```

ROWNUM	EMPNO	ENAME	JOB	SAL	DEPTNO
1	7839	KING	PRESIDENT	5000	10
2	7698	BLAKE	MANAGER	2850	30
3	7782	CLARK	MANAGER	2450	10
4	7566	JONES	MANAGER	2975	20
5	7788	SCOTT	ANALYST	3000	20
6	7902	FORD	ANALYST	3000	20
7	7369	SMITH	CLERK	800	20

Displaying top N rows

Suppose if you want to ***display the top three rows present in emp table*** then you can use this ***rownum*** along with the relational operator

Example

```
select rownum,empno,ename,job,sal,deptno
FROM EMP
where rownum<=3;
```

O/P

```
3 rows selected.
```

ROWNUM	EMPNO	ENAME	JOB	SAL	DEPTNO
1	7839	KING	PRESIDENT	5000	10
2	7698	BLAKE	MANAGER	2850	30
3	7782	CLARK	MANAGER	2450	10

Only the first 3 rows in a table or displayed

Truncate in DBMS

TRUNCATE IN DBMS

Sometimes the user wants to permanently delete the existing table or wants to delete the existing data alone or change the table name all these purposes are served using 3 DDL(data definition language) commands drop, truncate and rename respectively

TRUNCATE command

- The truncate command will make the table empty i.e all the table data will be deleted but the structure and database object is still alive ***and the table can be reused normally***
- The truncate command logically nothing but using delete command for deleting the records in the table without specifying the where condition i.e all the rows get deleted in that case

Syntax

```
TRUNCATE TABLE table_name;
```

Example

```
truncate table emp;
```

O/P

```
Table emp truncated successfully
```

After truncating the table if you try to display the table data using a select statement then you'll get a message as" ***0 rows selected***" i.e nothing to display because the table is empty without any rows

How TRUNCATE Is Different From DELETE?

DELETE	TRUNCATE
The DELETE command is used to delete specified rows(one or more).	While this command is used to delete all the rows from a table
DML COMMAND	DDL COMMAND(Autocommit)
Can use where condition with delete st	Can use where condition with truncate st
Delete specific rows"Where" condition	All rows will be removed from the table
Slower,because it used undo segment	Faster
"Delete"triggers will get fired	No triggers will get invoked
Won't reclaim the space used by table	Reclaim the space used by table
Won't reset the high level watermark	Reset the high level watermark

Syntax: DELETE command

```
DELETE FROM TableName
WHERE condition;
```

Rename in DBMS

RENAME IN DBMS

Sometimes the user wants to permanently delete the existing table or wants to delete the existing data alone or change the table name all these purposes are served using 3 DDL(data definition language) commands drop, truncate and rename respectively.

The rename command is used to change the existing table name and give a new name to the table.

Renaming a table: RENAME COMMAND

The rename command is used to change the existing table name and give a new name to the table.

Syntax

```
rename old_table _name to new_table_name
```

Example

```
rename emp to Employees
```

O/P

```
Table emp  renamed successfully
```

The above query changes the emp table name from ***emp to employees*** all operations on the table must be done using the new name ***employees*** otherwise it will raise an error.

Drop in DBMS

Dropping a table using DROP in DBMS

- Drop command delete the table existence completely i.e ***drop statement destroys the existing database object*** of that particular table, index or view
- After dropping a table if you try to use the table then compiler shows an error as " ***table or view does not exist***"

Syntax

```
DROP TABLE table_name;
```

Example

```
drop table emp;
```

O/P

```
Table emp dropped successfully
```

Restoring a dropped table using FLASHBACK Command

Before Oracle 10g whenever we use a drop command the table is permanently deleted but from ***Oracle 10g onwards we can get back the table from recycle bin by using flashback command.***

Syntax

```
flashback table table_ name to before drop
```

Example

```
flashback table emp to before drop
```

O/P

```
Table emp restored successfully
```

The above query gets the emp table back from the recycle bin.

Dropping a table permanently: PURGE Command

If you wish to ***delete the table permanently and not to allow the table to be restored from the recycle bin*** then you need to use '***purge*** '**‘**command along with drop command

Example

```
drop table emp purge
```

O/P

```
Table emp dropped permanently
```

The above query deletes the table object of emp permanently and we cannot use flashback command after purging

SQL Set Operations and Advanced SQL

Set Operations in DBMS

Set Operations in DBMS

SQL set operations are used for combining data from one or more tables

There are 3 set operations in SQL. They are

- UNION/UNION ALL
- INTERSECT
- MINUS

General Syntax for all SET operators

```
SELECT column name(s) FROM table1
UNION/UNION ALL/INTERSECT/MINUS
SELECT column name(s) FROM table2
```

Each SELECT statement that is used this set operators must follow these conditions

- The ***same number of columns***

- The columns must also have ***similar data types***
- The columns in each SELECT statement must also be in the ***same order***

UNION/UNION ALL

Union clause used to combine the result-set of two or more select queries

Customers table:

City	Country
Thimpu	Germany
Hyderabad	India
Hyderabad	India

Suppliers table:

City	Country
London	UK
California	USA
Texas	USA

Example for UNION

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```

Output:

City
Texas
Thimpu
London

	Hyderabad
	California

UNION all in SQL

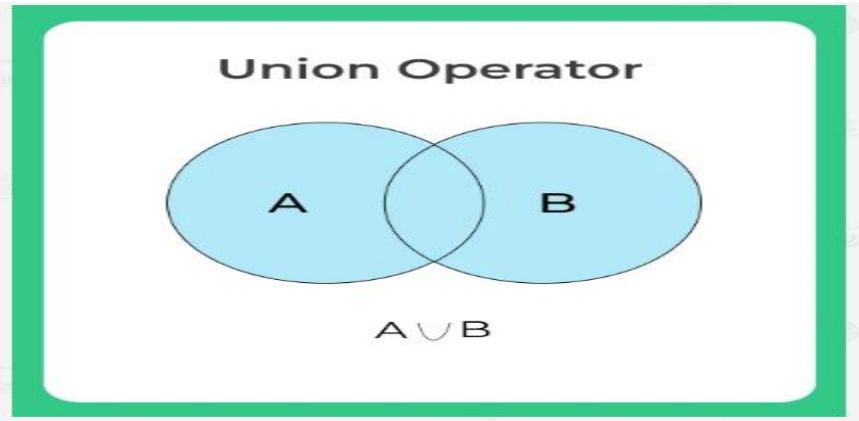
The only difference between UNION and UNION all Clause is that ***UNION removes duplicate values while comparing where UNION ALL allows duplicate values while combining***

Example for UNION ALL

```
SELECT City, Country FROM Customers;
UNION ALL
SELECT City, Country FROM Suppliers
```

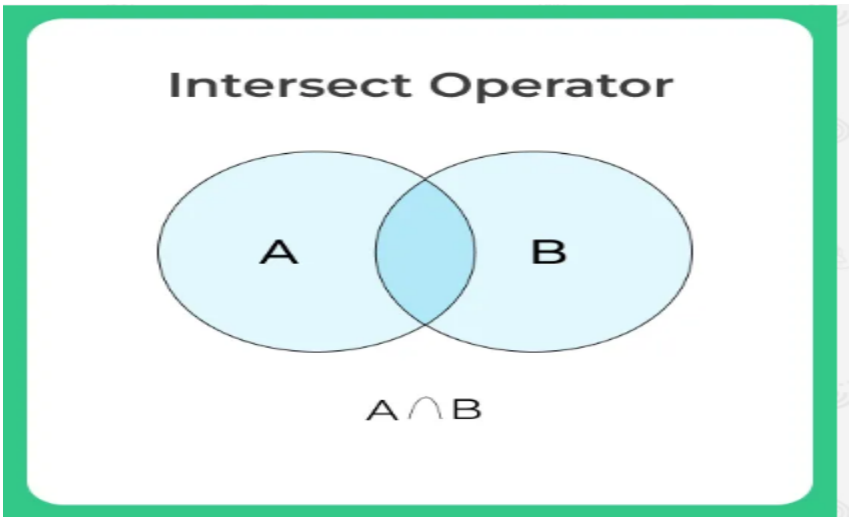
Output:

City	Country
Texas	USA
Thimpu	Germany
Londona	UK
Hyderabad	India
California	USA



INTERSECT

- Intersect statement combines result-set of two or more select queries and returns only those values that are common in both the result set
- Simply INTERSECT returns the common rows are values common in both the data sets and will not consider duplicate values



Example of INTERSECT

The **First** table,

ID	Name
1	rishi
2	trish
3	mahi

The **Second** table,

ID	Name
2	trish
3	Chester
4	mahi

INTERSECT query will be,

```
SELECT name FROM First
INTERSECT
SELECT name FROM Second;
```

The result set table will look like

Name
trish
mahi

MINUS

- Minus operator is used to subtract the result set obtained by the first SELECT query from the result set obtained by the second SELECT query.
- Simply, we can say that MINUS operator will return only those rows which are unique in only first SELECT query and not those rows which are common to both first and second SELECT queries.

Consider the two tables

Table1

Name	Address	Age
Priya	Hyderabad	19
Rahul	Chennai	20
Karthik	Mumbai	21
Payal	Delhi	20

Table2

Name	Course	Age
Divya	Java	22
Amitha	C++	19

Harshitha	Python	21
Payal	Java	20

Example

```
SELECT Name FROM Table1
MINUS
SELECT Name FROM Table2;
```

Output:

Name
Priya
Rahul
Karthik

Minus Operator in DBMS

Minus Operator in DBMS

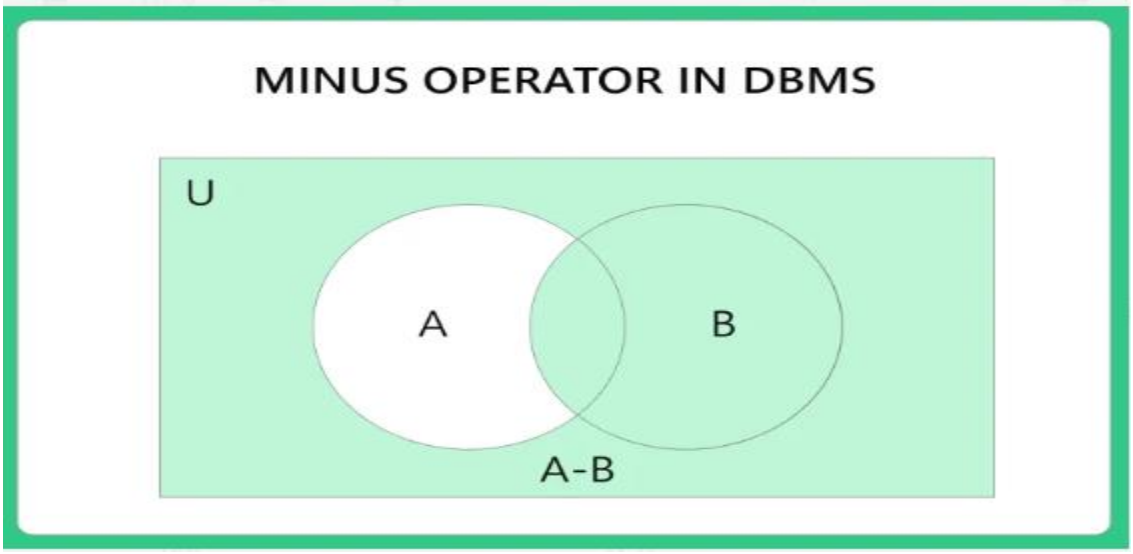
- MINUS operator is used to subtract the result set obtained by first SELECT query from the result set obtained by second SELECT query.
- Simply, we can say that MINUS operator will return only those rows which are unique in only first SELECT query and not those rows which are common to both first and second SELECT queries.

Syntax for MINUS

```
SELECT column name(s) FROM table1
MINUS
SELECT column name(s) FROM table2
```

The columns used in all the select statements must have the following

- the ***same number of columns***
- Similar or ***compatible data types***



Consider the two tables

Name	Address	Age	Grade
Priya	Hyderabad	19	A
Rahul	Chennai	20	B
Karthik	Mumbai	21	A
Payal	Delhi	20	B

Table1

Table2

Name	Course	Age	Grade
Divya	Java	22	B
Amitha	C++	19	A
Harshitha	Python	21	A

Payal	Java	20	B

Example

```
SELECT Name FROM Table1
MINUS
SELECT Name FROM Table2;
```

Name
Priya
Rahul
Karthik

Output:

Another Example

```
SELECT Name, Age,Grade FROM Table1
MINUS
SELECT Name,Age,Grade FROM Table2;
```

Name	Age	Grade
Priya	19	A
Rahul	20	B
Karthik	21	A

Output:

Note:

The MINUS operator is not supported with all databases. It is ***supported by Oracle database but not SQL server or PostgreSQL.***

ADVANCED SQL

Null Values in DBMS

Null Values in DBMS

- Special value that is supported by SQL is called as null which is used to represent values of attributes that are unknown or do not apply for that particular row
- For example age of a particular student is not available in the age column of student table then it is represented as null but not as zero
- It is important to know that ***null values is always different from zero value***
- A null value is used to represent ***the following different interpretations***
    - ***Value unknown*** (value exists but is not known)
    - ***Value not available*** (exists but is purposely hidden)
    - ***Attribute not applicable*** (undefined for that row)
- SQL provides special operators and functions to deal with data involving null values

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	500	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	500	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20
7369	SMITH	CLERK	7902	17-DEC-80	800	500	20

Consider the sample table '*emp*'

IS NULL operator

All ***operations upon null values present in the table must be done using this 'is null' operator*** .we cannot compare null value using the assignment operator

Example

```
select * from emp
where comm is null
```

O/P

```
4 rows selected.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20

The details of those employees whose commission value is Null are displayed.

IS NOT NULL

```
select * from emp
where comm is not null;
```

O/P

```
3 rows selected.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	500	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	500	10
7369	SMITH	CLERK	7902	17-DEC-80	800	500	20

Details of all those employees whose Commission value is not null value are displayed.

NOT NULL Constraint

- Not all constraints prevents a column to contain null values
- Once ***not null is applied to a particular column, you cannot enter null values to that column*** and restricted to maintain only some proper value other than null
- A ***not-null constraint cannot be applied at table level***

Example

```
CREATE TABLE STUDENT
(
  ID   INT           NOT NULL,
  NAME VARCHAR (20)     NOT NULL,
  AGE  INT           NOT NULL,
  ADDRESS  CHAR (25) ,
  SALARY   DECIMAL (18, 2),
  PRIMARY KEY (ID)
);
```

- In the above example, we have applied not null on three columns ID, name and age which means ***whenever a record is entered using insert statement all three columns should contain a value other than null***
- We have two other columns address and salary, ***where not null is not applied*** which means that ***you can leave the row as empty or use null value while inserting the record into the table.***

NVL() NULL Function

- Using NVL function you can ***substitute a value in the place of NULL values.***
- The substituted value then ***temporarily replaces the NULL values in your calculations or expression***. Remember that the substituted value ***only replaces the NULL value temporarily*** for the session and ***does not affect the value stored in the table.***
- Here is the syntax of NVL function.

```
NVL (exp, replacement-exp)
```

- As you can see NVL function ***takes two parameters exp and replacement exp***. ***First parameter exp can be a column name of a table or an arithmetic expression*** and the ***second parameter replacement expression will be the value which you want to substitute*** when a NULL value is encountered.
- Always remember the ***data type of both the parameters must match otherwise*** the compiler will raise an error.

Example

```
SELECT NVL (comm, 500) FROM employees
   WHERE salary>1000;
```

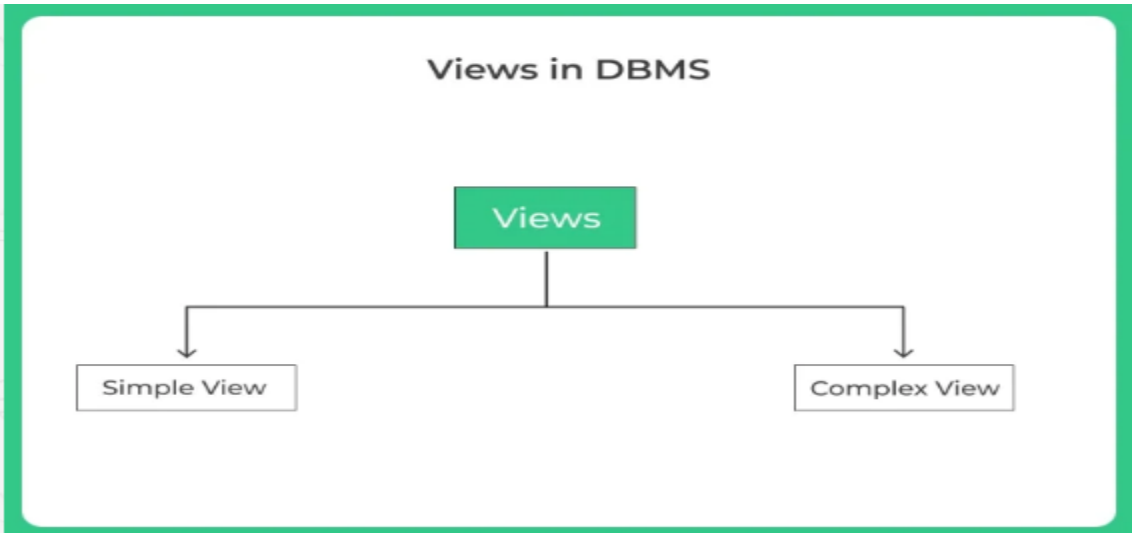
- On execution ***all the null values in the result set will get replaced by 500.***
- Similarly we can use NVL null function while performing arithmetic expression.
- Again let's take the same arithmetic expression which we used in the previous query where we added 100 to the values of commission column.

```
SELECT NVL(comm,100), NVL(comm,100)+100 FROM employees WHERE salary>1000;
```

Views in DBMS

Views in DBMS

- A view in SQL is a virtual table that is based upon the result-set of an SQL statement
- A view will also ***have rows and columns just like a real table*** in a database
- Simply a view is nothing but ***a stored SQL Query***
- A view can ***contain all the rows of a table or specific rows based on some condition***
- SQL functions conditions and join statements to a view and present the data just like the data is produced from a single table



Creating a view

A view is ***created by selecting fields from one or more tables*** present in a database

Syntax

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Note:

Whenever a user creates a view, database engine recreates the data using the views SQL statement i.e. ***view always shows upto date data***

Consider the tables StudentDetails and StudentMarks

StudentDetails

S_ID	NAME	ADDRESS
1	Harini	Kolkata
2	Preity	Hyderabad
3	Divya	Chennai
4	Kushi	Mumbai
5	Amitha	Bangalore

StudentMarks

ID	NAME	MARKS	AGE
1	Harini	96	20
2	Manisha	90	19
3	Divya	94	21
4	Kushi	92	19
5	Amitha	95	21

Simple Views in DBMS: Creating a view from a single table

In this example, we will create ***a view named as DetailsView from a single table StudentDetails***

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM StudentDetails
WHERE S_ID < 5;
```

The data present in a ***view can be seen just like a normal table select query***

```
SELECT * FROM DetailsView;
```

Output:

NAME	ADDRESS
Harini	Kolkata
Preity	Hyderabad
Divya	Chennai
Kushi	Mumbai

Complex view: Creating a view from multiple tables

- In this example will create a ***view named MarksView by taking data from both the table's student details and student marks***
- To create a View from multiple tables just simply ***include multiple tables in the SELECT statement.***

```
CREATE VIEW MarksView AS
SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS
FROM StudentDetails, StudentMarks
WHERE StudentDetails.NAME = StudentMarks.NAME;
```

To display data of View Marks:

```
SELECT * FROM MarksView;
```

NAME	MARKS	ADDRESS
Harini	96	Kolkata
Divya	94	Chennai
Kushi	92	Mumbai
Amitha	95	Bangalore

Output:

Deleting views in DBMS

- You can simply ***delete a view by using the Drop statement***
- That view is not used anymore

Syntax:

```
DROP VIEW view_name;
```

Example

```
DROP VIEW MarksView;
```

Updating views

Views are ***updated only if certain conditions are met*** otherwise if any one of the conditions are not met views will not be updated

Criteria for View Updating

- The ***select statement*** used in the create view statement ***should not include group by clause or order by clause***
- The select statement ***must not contain distinct keyword***
- A view ***should not be created from nested or Complex queries***
- A view should be ***created from a single table*** but ***if the view is created from more than one table then it is not allowed for updating***

CREATE OR REPLACE VIEW

Create or replace view statement is ***used to add or remove fields from existing views***

Syntax:

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1,coulmn2,..
FROM table_name
WHERE condition;
```

Update the view MarksView and add the field AGE to this View from StudentMarks Table,

```
CREATE OR REPLACE VIEW MarksView AS
SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS, StudentMarks. AGE
FROM StudentDetails, StudentMarks
WHERE StudentDetails.NAME = StudentMarks.NAME;
```

Fetch all the data from MarksView now as:

```
SELECT * FROM MarksView;
```

Output

NAME	ADDRESS	MARKS	AGE
HARINI	Kolkata	96	20

Divya	Chennai	94	21
Kushi	Mumbai	92	19
Amitha	Bangalore	95	21

Inserting a row into a view

We can use insert into statement of SQL ***to insert a row in a view just like inserting a row in an ordinary table***

Syntax:

```
INSERT view_name(column1, column2 , column3,..)
VALUES(value1, value2, value3..);
```

Example

```
INSERT INTO DetailsView(NAME, ADDRESS)
VALUES("Preity","Hyderabad");
```

Fetch all the data from DetailsView now as,

```
SELECT * FROM DetailsView;
```

NAME	ADDRESS
Harini	Kolkotta
Divya	Chennai
Kushi	Mumbai
Amitha	Bangalore
Preity	Hyderabad

Output

Deleting a row from a view

- A row in a view ***can be deleted just like simply deleting rows from a Table using delete statement***
- But remember a row in a view ***can be deleted only if the row is actually deleted in the original table from which it is created***

Syntax:

```
DELETE FROM view_name
WHERE condition;
```

Example

```
DELETE FROM DetailsView
WHERE NAME="Preity";
```

Fetch all the data from DetailsView now as,

```
SELECT * FROM DetailsView;
```

NAME	ADDRESS
Harini	Kolkotta
Divya	Chennai
Kushi	Mumbai
Amitha	Bangalore
Preity	Hyderabad

Output:

Advantages and disadvantages of views

Advantages

- **Enforce Business Rules:** By placing complicated or misunderstood business logic into the view, ***you can be sure to present a unified portrayal of the data*** which increases use and quality.
- **Consistency**: ***Once defined their calculations are referenced from the view rather than being restated in separate queries***. This makes for less mistakes and easier maintenance of code.

- **Security:** For example, ***you can restrict access*** to the employee table, that contains social security numbers, but allow access to a view containing name and phone number.
- **Simplicity:** Databases ***with many tables possess complex relationships***, which can be difficult to navigate if you aren't comfortable using Joins.
- **Space:** Views ***take up very little space***, as the data is stored once in the source table.

Limitations

- **Modifications:** Not all views support INSERT, UPDATE, or DELETE operations. ***Complex multi-table views are generally read-only.***
- **Performance:** ***Hugely complex*** job for the database engine. That is because each time a view is referenced, the query used to define it, is rerun.

DBMS Injection

- SQL injection is a technique used to exploit user's data through web page inputs by injecting SQL commands as statements where these statements can be used to manipulate the application web servers by the malicious users
- Simply SQL injection is nothing but ***introducing malicious code in SQL statements via web page input to destroy your database.***

Exploitation of SQL Injection in Web Applications

- Web servers are ***allowed to communicate with database servers at any time*** for storing and retrieving user's data into the database
- In this process, ***attackers design SQL statements so that they can be executed*** when the web server is retrieving content from the application server ***by compromising the security of the web application.***

Example of SQL injection

- An application that stores student records where any student can access and view his and her own records by entering his unique and private ID
- Student enters the following in the input field ***13332345 or 1=1.***
- As 1=1 holds true for all the records. Hence ***all the student details are accessed irrespective of ID*** details ***by the attacker which can be deleted or modified in the same way***

Consider the following SQL query.

```
SELECT * from USER where
USERNAME = “” and PASSWORD=””
```

Malicious user can simply make use of = and or operator

```
Select * from User where
(Username = “” or 1=1) AND
(Password=”” or 1=1).
```

As backend code has been changed and even if the password is right or wrong no matter because 1=1 always returns true and all the records of the students are compromised. As a result the query when executed ***can easily modify and access the private and secure information which is not intended to be shown to users.***

Impact of the SQL injection

- ***Hacker can retrieve all users' data*** present in the database like mobile number, credit card information which is highly confidential and can gain access to admin portals
- Almost all online shopping applications, definitely use backend database servers and ***if they are exploited through SQL injection entire server is compromised.***

Preventing SQL injection

- **Strong user authentication:** User input should be ***validated by predefining length and type of the input*** and then authenticating the user
- **Restricting access privileges:** Limiting the amount of information a user can view and ***restricting the user not to be granted permission to access everything in the database***
- **Using views:** By using views, ***Virtual Table is displayed to the users*** and becomes difficult to manipulate the original database table and by making most of the views as read-only
- **Secure Coding:** Secure coding is applied so that a ***strong level of authentication is done each time the code has been updated.***

Simple implementation example

- The login page is bypassed whereas ***injection can provide an attacker with unauthorized access*** to sensitive data like customer data trade secrets and intellectual property
- There is also an ***SQL Injection Automation tool sqlmap*** which is used ***to perform all type of SQL injection***.

```
$stmt = $dbConnection->prepare('SELECT count(*) FROM users WHERE username = ? AND password = ?');
$stmt->bind_param('ss', $username,$password);
```

```
$stmt->execute();
$result = $stmt->get_result();
echo $result;
?>
```