

SQL

Pratik Joshi
Data Engineer
Cloud Engineer (AWS Certified)
Android Developer (Google Certified)

Introduction

- Data
- Database
- DBMS
- RDBMS
- SQL
- No-SQL
- Big Data

Pratik Joshi



DBMS

Database Management System

Database: Collection of data

Management System: Set of program to store and retrieve data

DBMS: It is a collection data and set of program to access and store the data in an easy and efficient manner. It is used to manage databases.

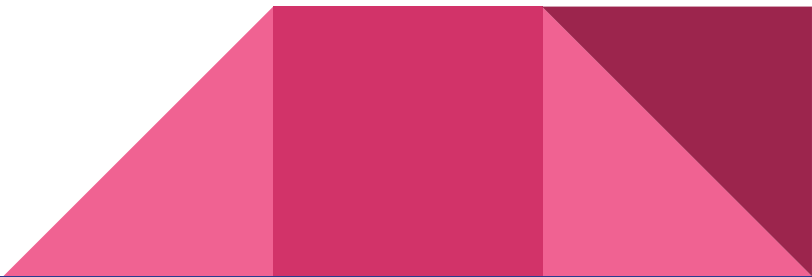
There are many databases present. Like Oracle, MySql, Tearadata, Microsoft SQL, SQLite, PostgreSQL

Pratik Joshi



Purpose of DBMS

Prior to DBMS data was managed in file-system. The advantages of DBMS over file system is:

- **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
 - **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
 - **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
 - **Reduce time:** It reduces development time and maintenance need.
 - **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- 

DBMS vs File System

DBMS	File System
DBMS is a collection of data. In DBMS, the user is not required to write the procedures.	File system is a collection of data. In this system, the user has to write the procedures for managing the database.
DBMS gives an abstract view of data that hides the details.	File system provides the detail of the data representation and storage of data.
DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from the system failure.	File system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will lost.
DBMS provides a good protection mechanism.	It is very difficult to protect a file under the file system.
DBMS contains a wide variety of sophisticated techniques to store and retrieve the data.	File system can't efficiently store and retrieve the data.
DBMS takes care of Concurrent access of data using some form of locking.	In the File system, concurrent access has many problems like redirecting the file while other deleting some information or updating some information.

RDBMS


It is Relational Database Management System. It is called Relational because here databases are related to each other. E.g Oracle, MySql, MS Sql

Components of RDBMS:

Table: RDBMS database uses tables to store data. A table is a collection of related data entries and contains rows and columns to store data.

Record: A row of a table is also called record. It contains the specific information of each individual entry in the table. It is a horizontal entity in the table.

Column: A column is a vertical entity in the table which contains all information associated with a specific field in a table.



DBMS Architecture

There are basically three types of DBMS Architecture.

1-Tier Architecture

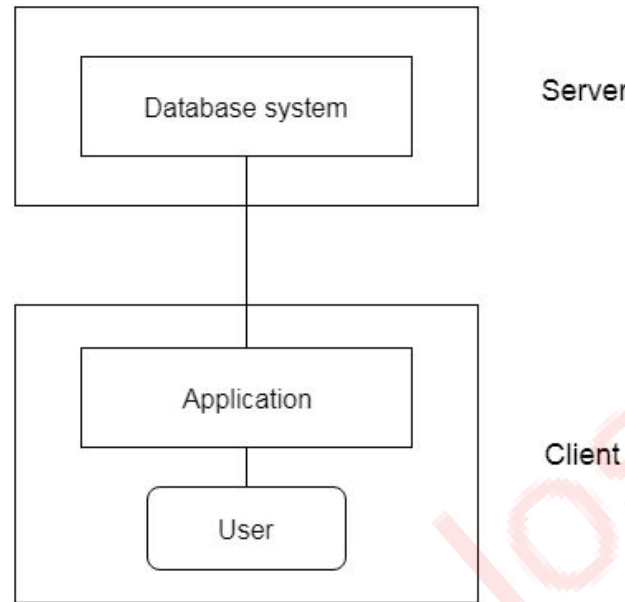
2-Tier Architecture

3-Tier Architecture

1-Tier Architecture:

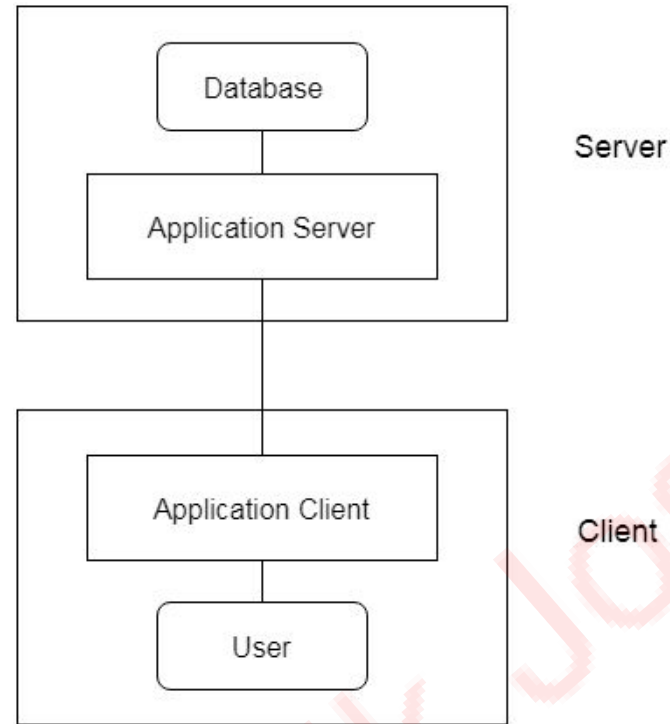
- In this architecture, the database is directly available to the user. It means the user can directly access the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

2-Tier Architecture:



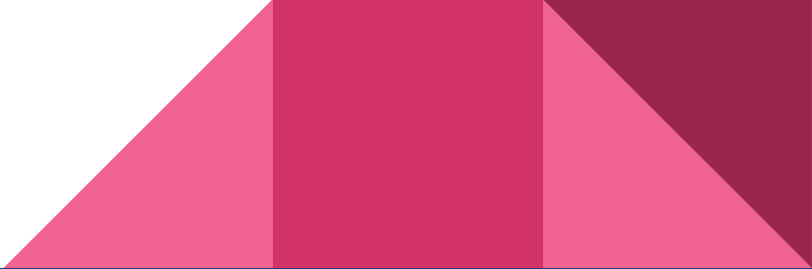
- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: ODBC, JDBC are used.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.

3-Tier Architecture:



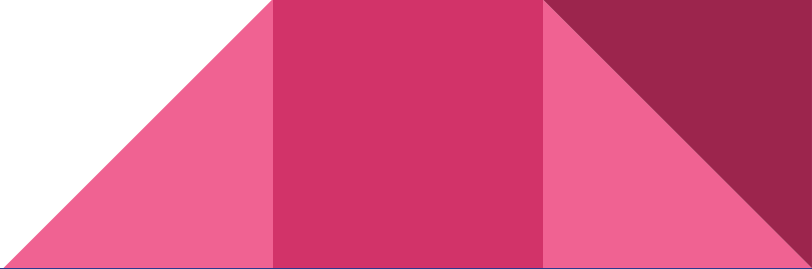
- The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.

Application of DBMS

- **Banking:** for storing customer information, account activities, deposits, payment details, loans, etc.
 - **Manufacturing:** for supply chain management, production tracking and inventory management
 - **Finance:** for storing information about stocks, sales, and purchases of financial instruments like stocks and bonds
 - **Education:** for student information, course registrations, payroll and grades
 - **Airlines:** for reservations, ticket booking and schedule information
- 

ACID Properties

ACID Properties are used for maintaining the integrity of database during transaction processing. ACID in DBMS stands for **A**tomicity, **C**onsistency, **I**solation, and **D**urability.

- **Atomicity:** A transaction is a single unit of operation. You either execute it entirely or do not execute it at all. There cannot be partial execution.
 - **Consistency:** Once the transaction is executed, it should move from one consistent state to another. The data should be always correct.
 - **Isolation:** Transaction should be executed in isolation from other transactions (no Locks). During concurrent transaction execution, intermediate transaction results from simultaneously executed transactions should not be made available to each other. (Level 0,1,2,3)
 - **Durability:** · After successful completion of a transaction, the changes in the database should persist. Even in the case of system failures. It ensures the permanency of data.
- 

SQL

Structured Query Language is a standard Database language which is used to create, maintain and retrieve the relational database.

- Standard Language for Managing almost any RDBMS
- Supports with set of commands for managing RDBMS
- Developed by IBM
- SQL is case insensitive. But it is a recommended practice to use keywords in capital letters and use user defined things (like table name, column name, etc) in small letters.
- We can write comments in SQL using “–” (double hyphen) at the beginning of any line.

Types of SQL: DDL, DML, DQL, DCL, TCL

DDL

Data Definition Language (DDL) statements are used to define the database structure or schema.

CREATE - to create objects in the database

ALTER - alters the structure of the database

DROP - delete objects from the database

TRUNCATE - remove all records from a table permanently

RENAME - rename an object

Pratik Joshi



DML and DQL

Data Manipulation Language (DML) statements are used for managing data within schema objects.

INSERT - insert data into a table

UPDATE - updates existing data within a table

DELETE - deletes unwanted/all records from a table

Data Query Language

SELECT - To Project the data



DCL

Data Control Language (DCL) statements are used for granting and denying access privileges to other users

GRANT - gives user's access privileges to database

REVOKE - withdraw access privileges given with the GRANT command

Pratik Joshi



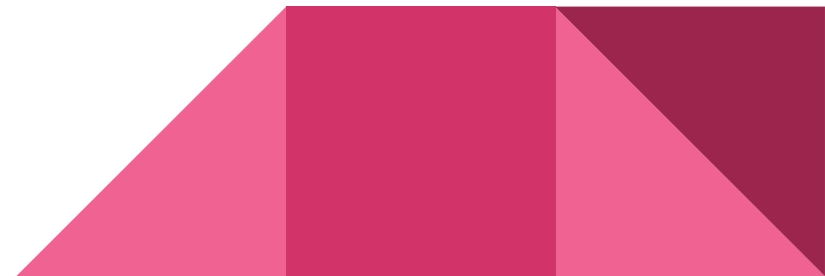
TCL

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

COMMIT - save work done

ROLLBACK - restore database to original since the last COMMIT
SAVEPOINT - identify a point in a transaction to which you can later roll back

Praik Joshi



Primary Key

A primary key is a single field or combination of fields that contains a unique record. It must be filled.

- None of the field of primary key can contain a null value.
- A table can have only one primary key.

Pratik Joshi



Starting With Oracle

Windows and Linux User:

<https://www.oracle.com/in/database/technologies/xe-downloads.html>

Mac User:

Oracle Cloud

Oracle Ducker

Oracle Cloud:

<https://livesql.oracle.com/>

Oracle Sql Developer IDE (Mac, Windows, Linux):

<https://www.oracle.com/tools/downloads/sqldev-downloads.html>

Starting with MySql

Windows User:

<https://dev.mysql.com/get/Downloads/MySQLInstaller/mysql-installer-web-community-8.0.25.0.msi>

Mac and Linux User:

<https://dev.mysql.com/downloads/mysql/>

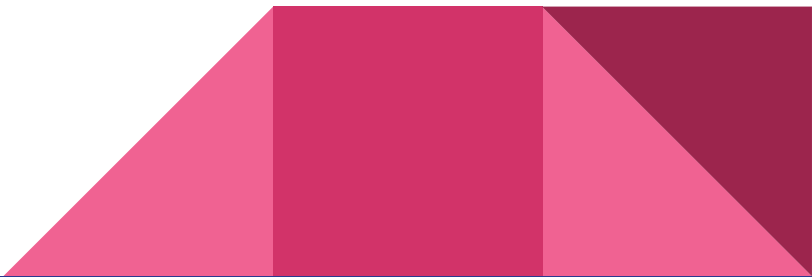
Praik Joshi



What is Oracle?

Oracle database is a relational database management system. It is known as Oracle database, OracleDB or simply Oracle. It is produced and marketed by Oracle Corporation.

Editions of Oracle DB:

- **Enterprise Edition:** It is the most robust and secure edition. It offers all features, including superior performance and security.
 - **Standard Edition:** It provides the base functionality for users that do not require Enterprise Edition's robust package.
 - **Express Edition (XE):** It is the lightweight, free and limited Windows and Linux edition.
 - **Oracle Lite:** It is designed for mobile devices.
- 

Data Types

String Datatype:

CHAR(size) It is used to store character data within the predefined length. It can be stored up to 2000 bytes.

VARCHAR2(size) It is used to store variable string data within the predefined length. It can be stored up to 4000 byte.

VARCHAR(SIZE) It is the same as VARCHAR2(size). You can also use VARCHAR(size), but it is suggested to use VARCHAR2(size)



Numeric Datatype:


NUMBER(p, s) : Total length: p, decimal: s

FLOAT(p) : It is a subtype of the NUMBER data type. The precision p can range from 1 to 126.

Date Type:

DATE: It is used to store a valid date-time format with a fixed length. Its range varies from January 1, 4712 BC to December 31, 9999 AD.

TIMESTAMP : It is used to store the valid date in YYYY-MM-DD with time hh:mm:ss format.



Oracle Large Object Data Types (LOB Types)

BLOB It is used to specify unstructured binary data. Its range goes up to $2^{32}-1$ bytes or 4 GB.

BFILE It is used to store binary data in an external file. Its range goes up to $2^{32}-1$ bytes or 4 GB.

CLOB It is used for single-byte character data. Its range goes up to $2^{32}-1$ bytes or 4 GB.

Pratik Joshi



DDLs in Oracle

CREATE

syntax: `CREATE TABLE table_name(col1 datatype,col2 datatype);`

e.g: `CREATE TABLE customer(cust_id NUMBER, customer_name VARCHAR2(50))`

To Create Table with Primary Key:

```
CREATE TABLE customers
( customer_id number(10) ,
  customer_name varchar2(50),
  city varchar2(50),
  CONSTRAINT customers_pk PRIMARY KEY (customer_id)
);
```


ALTER

To Add Column Name:

```
ALTER TABLE customers ADD customer_age varchar2(50);
```

To Add Multiple Column:

```
ALTER TABLE customers  
ADD (customer_type varchar2(50),  
     customer_address varchar2(50));
```

Modify Column:

```
ALTER TABLE customers MODIFY customer_name varchar2(100);
```

Drop Column:

```
ALTER TABLE customers DROP COLUMN customer_name;
```

Rename Column:

```
ALTER TABLE customers RENAME COLUMN customer_name to cname;
```

Rename Table:

```
ALTER TABLE customers RENAME TO retailers;
```



DROP AND TRUNCATE:

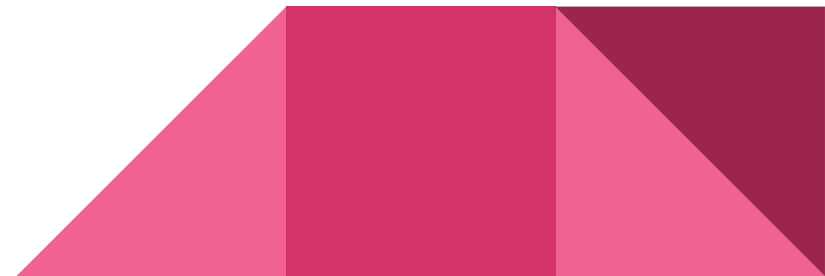
DROP

```
DROP TABLE table_name;  
-- Remove the table permanently from the DB
```

TRUNCATE

```
TRUNCATE TABLE table_name;  
-- Removed the data of the table, structure remains same
```

Pratik Joshi



Working with DMLs

Insert:

```
INSERT INTO customers(CUSTOMER_ID,CUSTOMER_NAME,CITY) VALUES(101,'Arvind','Delhi');
```

Select:

```
SELECT * FROM customers;
```

* is used to project all the data

Update:

```
UPDATE customers SET city='New Delhi' WHERE customer_id=101;
```

WHERE is used to filter the data.

Delete:

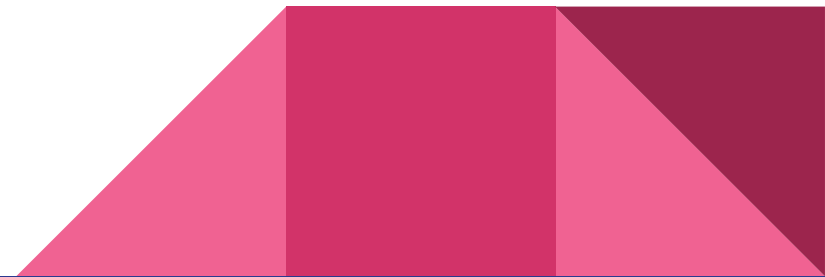
```
DELETE FROM customers WHERE customer_id=101;
```

DELETE FROM customers; --To delete entire data of table

DUAL Table

- The DUAL table is owned by the user SYS and can be accessed by all users.
- It contains one column, DUMMY, and one row with the value X.
- The DUAL table is useful when you want to return a value once only (for example, the value of a constant, pseudocolumn, or expression that is not derived from a table with user data).
- The DUAL table is generally used for SELECT clause syntax completeness

Pratibha Joshi



Operators

Arithmetic operators

+ : addition

- : subtraction

* : multiplication

/ : divide

= : equal to

!=, <>, ^= : not equal to

< : less than

<= : less than or equal to

> : greater than

>= : greater than or equal to

Pratik Joshi



Logical Operators

NOT : logical NOT operator

AND : logical AND operator

OR : logical OR operator

SQL operators

IN : list of values within ()

NOT IN : negation of IN

BETWEEN : range of values

NOT BETWEEN : negation of BETWEEN

LIKE : using “%” and “_” as meta characters

NOT LIKE : negation of LIKE

IS NULL : evaluating NULL's

IS NOT NULL : negation of IS NULL



Examples

```
SELECT ENAME,JOB,SAL,DEPTNO FROM EMP;
```

Arithmetic operation

```
SELECT ENAME,JOB,SAL,(SAL+250)*12 FROM EMP;
```

Alias column heading

```
SELECT EMPNO,ENAME AS EMPNAME,JOB DESIG,SAL  
Salary,(SAL+250)*12 "Annual Salary" FROM EMP;
```

Concatenation operator (||)

```
SELECT ENAME,JOB,ENAME||JOB FROM EMP;
```

```
SELECT ENAME||' works as '||JOB "EMP JOB" FROM EMP;
```



Handling Null

```
SELECT ENAME,SAL,COMM,SAL*12+COMM FROM EMP;
```

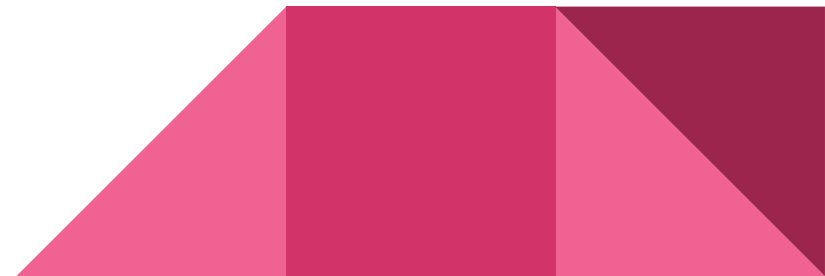
NVL ()

```
SELECT ENAME, SAL, COMM, SAL*12+NVL(COMM,0) FROM EMP;
```

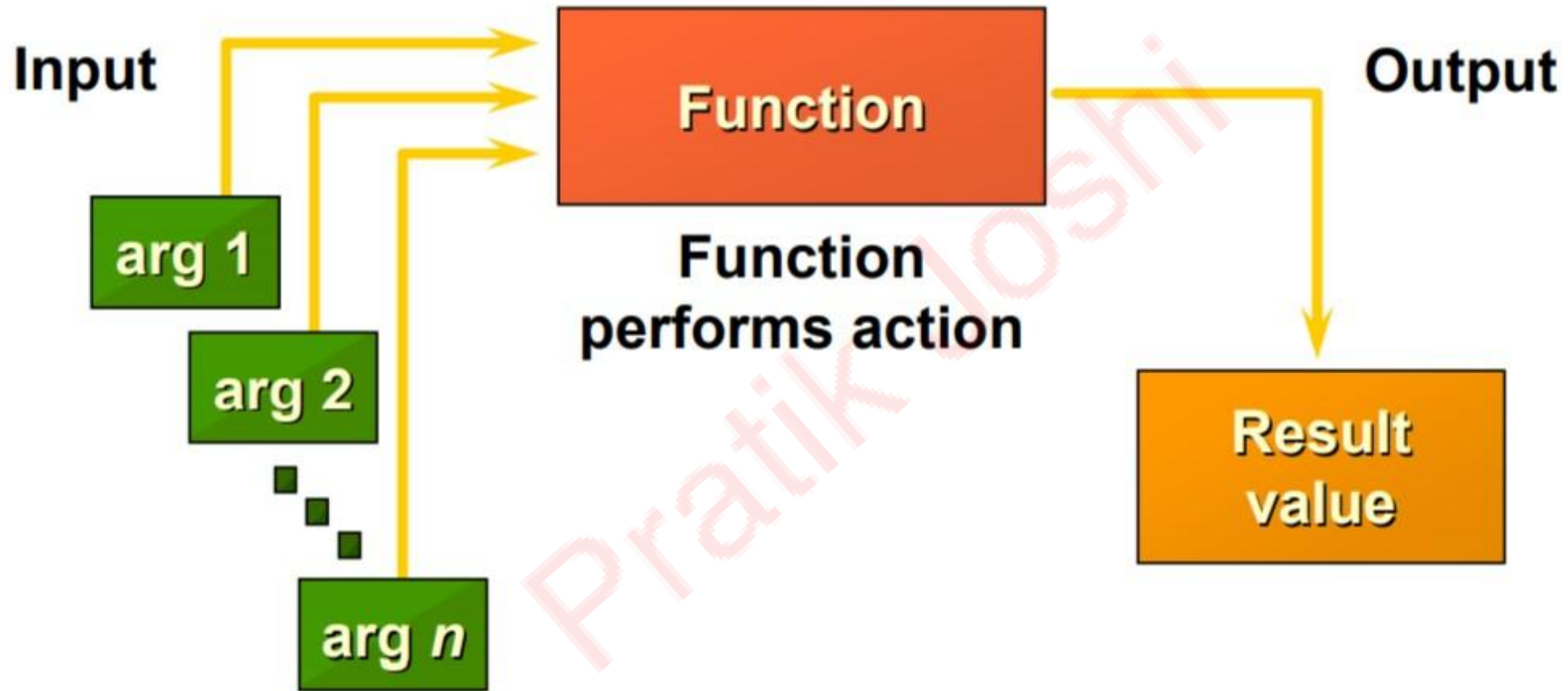
NVL2()

```
SELECT ENAME,SAL,COMM,NVL2(COMM,SAL*12+COMM,SAL*12) FROM EMP;
```

Pratik Joshi



Function



Character Functions

LOWER (string) – returns data in lower case

UPPER(string) – returns data in upper case

INITCAP(string) – returns with first character in caps for each word

CONCAT(string1,string2) – concatenates two strings

```
SQL> SELECT ENAME,LOWER(ENAME),UPPER('OrACle'),INITCAP(JOB),CONCAT(JOB,SAL)
FROM EMP WHERE DEPTNO=10;
```

Res:

CLARK	clark	ORACLE	Manager	MANAGER2450
KING	king	ORACLE	President	PRESIDENT5000
MILLER	miller	ORACLE	Clerk	CLERK1300

LENGTH(string)- returns length of the string

TRANSLATE(string,source,target) – overwrites source chars with target chars

```
SQL> SELECT DNAME,LENGTH(DNAME),  
TRANSLATE(DNAME,'A','X'),  
TRANSLATE(DNAME,'AS','XY')FROM DEPT;
```

Res:

ACCOUNTING	10	XCCOUNTING	XCCOUNTING
RESEARCH	8	RESEXRCH	REYEXRCH
SALES	5	SXLES	YXLEY
OPERATIONS	10	OPERXTIONS	OPERXTIONY

REPLACE(string,source,target) – replaces source string with target string

```
SQL> SELECT JOB, REPLACE(JOB,'SALESMAN','MARKETING') FROM EMP WHERE  
DEPTNO=30;
```

SALESMAN	MARKETING
SALESMAN	MARKETING
SALESMAN	MARKETING
MANAGER	MANAGER
SALESMAN	MARKETING
CLERK	CLERK

CHR(number) – returns char equi of ascii value

```
SELECT CHR(67)||CHR(65)||CHR(84) "Dog" FROM DUAL;
```

Number Function

ROUND()

```
SQL> SELECT ROUND(40.44,1), ROUND(40.45,1), ROUND(40.4),ROUND(40.5)  
FROM DUAL;
```

Op:

40.4 40.5 40 41

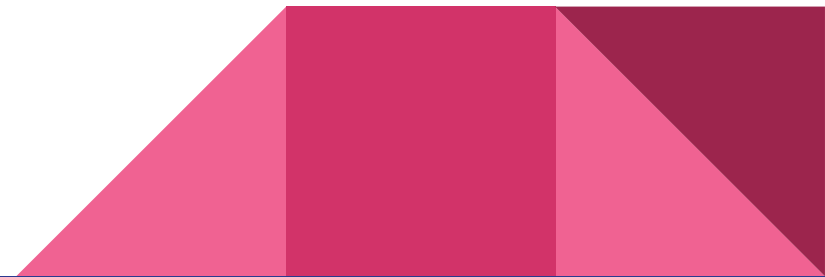
TRUNC()

```
SQL> SELECT TRUNC(40.41,1),TRUNC(40.49,1), TRUNC(40.1),TRUNC(40.9) FROM  
DUAL
```

Op:

40.4 40.4 40 40

Praik Joshi



CEIL() : Round the value upward

FLOOR() : Round the value downward

```
SQL> SELECT CEIL(99.6),CEIL(-11.5),FLOOR(101.76),FLOOR(-11.1) FROM DUAL;
```

Op:

```
CEIL(99.6) CEIL(-11.5) FLOOR(101.76) FLOOR(-11.1)
```

```
100 -11 101 -12
```

POWER()

```
SQL> SELECT POWER(SAL,2),POWER(5,3) FROM EMP  
WHERE DEPTNO=10;
```

Op:

```
POWER(SAL,2)
```

```
POWER(5,3)
```

```
6002500
```

```
125
```

```
25000000
```

```
125
```

```
1690000
```

```
125
```

Pratik Joshi



SQRT()

```
SQL> SELECT SQRT(SAL),SQRT(25) FROM EMP WHERE  
DEPTNO=10;
```

SQRT(SAL)	SQRT(25)
49.4974747	5
70.7106781	5
36.0555128	5

SIGN()

```
SQL> SELECT SAL-COMM,SIGN(SAL-COMM),  
COMM-SAL,SIGN(COMM-SAL),SIGN(5-5)  
FROM EMP WHERE DEPTNO=30;
```

1300	1	-1300	-1	0
750	1	-750	-1	0
-150	-1	150	1	0

MOD() : Modulus(%) of two number

ABS(): Absolute value of a number

SQL> SELECT MOD(10,3),MOD(10,2),ABS(45),ABS(-45) FROM DUAL;

Op:

MOD(10,3)	MOD(10,2)	ABS(45)	ABS(-45)
1	0	45	45

ASCII() : Returns the ASCII value of any character.

SQL> SELECT ASCII('K') "Ascii value" FROM DUAL;

Op:

Ascii value

75

Date Function

- Oracle database stores dates in an internal numeric format: century, year, month, day, hours, minutes, seconds.
- The default date display format is DD-MON-RR.

To get system date:

```
SQL> SELECT SYSDATE FROM DUAL;
```

```
SYSDATE
```

```
-----
```

```
17-JAN-21
```

To change system date format:

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT='DD-MON-YYYY HH24:MI:SS'
```

Arithmetic with Dates :

- Add or subtract a number to or from a date for a resultant date value.
- Subtract two dates to find the number of days between those dates.
- Add hours to a date by dividing the number of hours by 24.

```
SELECT HIREDATE, HIREDATE+7, HIREDATE-7, SYSDATE-HIREDATE FROM EMP WHERE DEPTNO=10;
```

```
09-JUN-81 16-JUN-81 02-JUN-81 6339.71453
```

```
17-NOV-81 24-NOV-81 10-NOV-81 6178.71453
```

```
23-JAN-82 30-JAN-82 16-JAN-82 6111.71453
```

```
SELECT ENAME, (SYSDATE-HIREDATE)/7 AS WEEKS FROM EMP;
```

```
--Add and Differ 12 hours
```

```
SELECT SYSDATE+1/2, SYSDATE-1/2 FROM DUAL;
```



Function	Description
MONTHS_BETWEEN	Number of months between two dates
ADD_MONTHS	Add calendar months to date
NEXT_DAY	Next day of the date specified
LAST_DAY	Last day of the month

MONTHS_BETWEEN()

```
SQL> SELECT MONTHS_BETWEEN(SYSDATE,HIREDATE),  
MONTHS_BETWEEN('01-JAN-1984','15-JAN-1998')  
FROM EMP WHERE DEPTNO=10;
```

ADD_MONTHS()

```
SQL> SELECT HIREDATE,ADD_MONTHS(HIREDATE,3),  
ADD_MONTHS(HIREDATE,-3)FROM EMP  
WHERE DEPTNO=10;
```

HIREDATE	ADD_MONTH	ADD_MONTH
-----	-----	-----
09-JUN-81	09-SEP-81	09-MAR-81
17-NOV-81	17-FEB-82	17-AUG-81
23-JAN-82	23-APR-82	23-OCT-81

NEXT_DAY()

```
SQL> SELECT HIREDATE,NEXT_DAY(HIREDATE,'FRIDAY'),  
NEXT_DAY(SYSDATE,'FRIDAY'),  
NEXT_DAY(SYSDATE,6) FROM EMP  
WHERE DEPTNO=10
```

LAST_DAY()

```
SQL> SELECT HIREDATE, LAST_DAY(HIREDATE),  
LAST_DAY(SYSDATE), LAST_DAY('12-FEB-88')  
FROM EMP WHERE DEPTNO=10;
```

HIREDATE	LAST_DAY(HIREDATE)	LAST_DAY(SYSDATE)	LAST_DAY('12-FEB-88')
17-NOV-81	30-NOV-81	30-JUN-21	29-FEB-88
09-JUN-81	30-JUN-81	30-JUN-21	29-FEB-88
23-JAN-82	31-JAN-82	30-JUN-21	29-FEB-88

Decode()

```
SQL> SELECT ENAME,JOB,  
DECODE(JOB,'MANAGER','Sr Manager',  
'CLERK','Supervisor',  
'ANALYST','Programmer',  
'Unassigned') NEW_JOB FROM EMP;
```

ENAME	JOB	NEW_JOB
KING	PRESIDENT	Unassigned
BLAKE	MANAGER	Sr Manager
CLARK	MANAGER	Sr Manager
JONES	MANAGER	Sr Manager
SCOTT	ANALYST	Programmer
FORD	ANALYST	Programmer
SMITH	CLERK	Supervisor
ALLEN	SALESMAN	Unassigned
WARD	SALESMAN	Unassigned
MARTIN	SALESMAN	Unassigned
TURNER	SALESMAN	Unassigned
ADAMS	CLERK	Supervisor
JAMES	CLERK	Supervisor
MILLER	CLERK	Supervisor

```
SQL> SELECT JOB,SAL,  
DECODE(JOB,'MANAGER',SAL*1.25,  
'CLERK',SAL*1.20,'SALESMAN',SAL*1.10,  
SAL) DECODED_SAL FROM EMP;
```

JOB	SAL	DECODED_SAL
-----	-----	-----
PRESIDENT	5000	5000
MANAGER	2850	3562.5
MANAGER	2450	3062.5
MANAGER	2975	3718.75
ANALYST	3000	3000
ANALYST	3000	3000
CLERK	800	960
SALESMAN	1600	1760
SALESMAN	1250	1375
SALESMAN	1250	1375
SALESMAN	1500	1650
CLERK	1100	1320
CLERK	950	1140
CLERK	1300	1560

Pratik Joshi



Greatest()

```
SELECT GREATEST(10, 50, 22) FROM dual;
```

Op :

50

Least()

```
SELECT LEAST(52, 15, 8, 89) FROM dual;
```

Op :

8



Case Statements

Simple CASE – statement

Syntax:

CASE parameter

WHEN parameter value THEN result1

WHEN parameter value THEN result2

WHEN parameter value THEN resultN

ELSE result

END;

Example Query:

SELECT ENAME, JOB, SAL,

CASE JOB

WHEN 'PRESIDENT' THEN SAL*1.25

WHEN 'MANAGER' THEN SAL*1.20

WHEN 'CLERK' THEN SAL*1.15

ELSE SAL*1.10

END "NEW SALARY"

FROM EMP



Searched CASE – statement

Syntax:

```
CASE  
    WHEN condition1 THEN result1  
    WHEN condition2 THEN result2  
    WHEN conditionN THEN resultN  
    ELSE result  
END;
```

Example Query:

```
SELECT ENAME,JOB,SAL,  
CASE  
    WHEN JOB IN('PRESIDENT','ANALYST') THEN 'Platinum Card'  
    WHEN SAL > 3000 THEN 'Gold Card'  
    WHEN SAL BETWEEN 2000 AND 3000 THEN 'Silver Card'  
    ELSE 'Not Eligible for card'  
END "Credit Card"  
FROM EMP
```

Group Function

The functions which deal with multi row data to aggregate the values are called Group functions or Aggregate Functions.

Some examples of Group Functions:

MIN()

MAX()

AVG()

SUM()

COUNT()

Pratik Joshi



```
SQL> SELECT MIN(SAL) FROM EMP;
```

MIN(SAL)

800

```
SQL> SELECT MIN(SAL),MAX(SAL),AVG(SAL),  
SUM(SAL),COUNT(*)FROM EMP;
```

MIN(SAL)	MAX(SAL)	AVG(SAL)	SUM(SAL)	COUNT(*)
-----	-----	-----	-----	-----
800	5000	2073.21429	29025	14

```
SQL> SELECT MIN(SAL) FROM EMP WHERE JOB='CLERK';  
MIN(SAL)
```

800

```
SQL> SELECT COUNT(*) FROM EMP WHERE DEPTNO=30;  
COUNT(*)
```

6



Group By Clause

The GROUP BY statement is often used with aggregate functions to group the result-set by one or more columns.

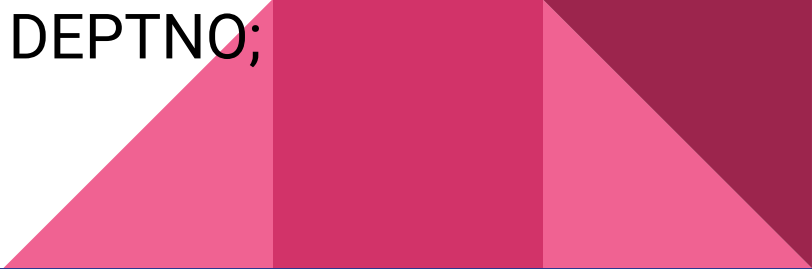
Any column or expression in the SELECT list that is not an aggregate function must be in the GROUP BY clause.

To display JOB wise sum of salary:

```
SQL> SELECT JOB,SUM(SAL) FROM EMP GROUP BY JOB;
```

To display sum of salary department no. wise:

```
SQL> SELECT DEPTNO,SUM(SAL) FROM EMP GROUP BY DEPTNO;
```



Having Clause

Having clause is used to filter on top of aggregated values.

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

To display department wise average salary where average salary > 2000:

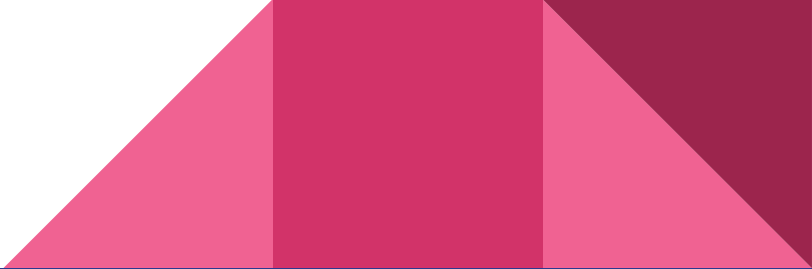
```
SQL> SELECT DEPTNO,AVG(SAL) FROM EMP GROUP BY DEPTNO  
      WHERE AVG(SAL) > 2000;
```

ERROR at line 2:

ORA-00934: group function is not allowed here

```
SQL> SELECT DEPTNO,AVG(SAL) FROM EMP GROUP BY DEPTNO  
      HAVING AVG(SAL) > 2000;
```

Problem Statements

1. Calculate total numbers of employee.
 2. Calculate total number of employee department wise.
 3. Display dept having more than 1 employee in their department.
 4. Calculate dept wise sum of salary.
 5. Calculate maximum of department wise salary sum.
 6. Find the maximum sal of any department greater than 3000
 7. Find the total number of MANAGER present in the EMP table.
 8. Find the total number of SALESMAN in the EMP table whose salary > 1500.
 9. Find the job having average salary more than 1500.
 10. Find the job having average salary more than 1000 and there are at least three employee.
- 

Order By Clause

In Oracle, ORDER BY Clause is used to sort or re-arrange the records in the result set. The ORDER BY clause is only used with SELECT statement.

Syntax:

```
SELECT *  
FROM table_name  
ORDER BY col_name;
```

ASC: To sort the records in ascending order.

DESC: To sort the records in descending order.



Distinct Clause

Oracle DISTINCT clause is used to remove the duplicate records from the result set. It is only used with SELECT statement.

Syntax:

```
SELECT DISTINCT expressions  
FROM tables  
WHERE conditions;
```

E.g:

```
SELECT DISTINCT deptno FROM emp;
```



Joins

JOINS are used to retrieve information from multiple tables.

Join Conditions: There may be at least one join condition. It compares two columns from different tables and combines the rows where the join condition is true.

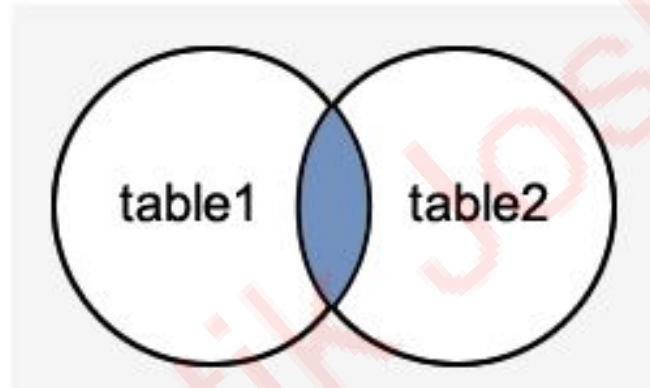
Types of Joins:

- Inner Joins (Simple Join)
- Outer Joins
 - Left Outer Join (Left Join)
 - Right Outer Join (Right Join)
 - Full Outer Join (Full Join)
- Self Joins
- Cross Joins (Cartesian Products)



Inner Join

Inner Join is the simplest and most common type of join. It is also known as simple join or equi join.



E.g:

```
SELECT EMP.ENAME,EMP.JOB,EMP.DEPTNO, DEPT.DNAME,DEPT.LOC  
FROM EMP,DEPT  
WHERE EMP.DEPTNO=DEPT.DEPTNO; (Joining Condition)
```

```
SQL> SELECT E.ENAME,E.JOB,E.DEPTNO,D.DNAME,D.LOC  
FROM EMP E,DEPT D  
WHERE E.DEPTNO=D.DEPTNO;
```

Here alias of EMP and DEPT table created.

EMP e and DEPT d

e and d also contains all the attributes of EMP and DEPT table respectively.

Pratik Joshi

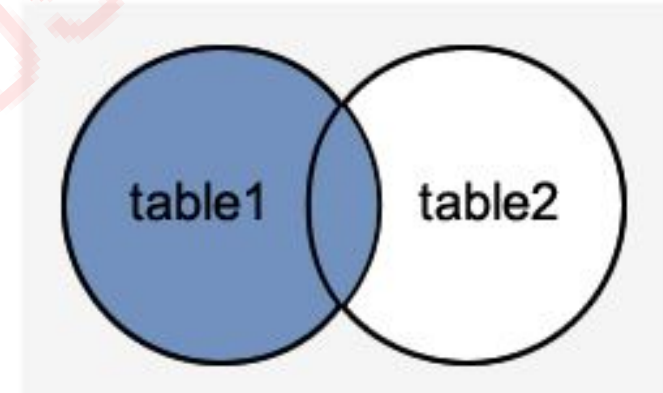


Left Outer Join

Left Outer Join returns all rows from the left (first) table specified in the ON condition and only those rows from the right (second) table where the join condition is met.

Syntax:

```
SELECT columns  
FROM table1  
LEFT OUTER JOIN table2  
ON table1.column = table2.column;
```



```
select ename, dname, emp.deptno, dept.deptno,dname,loc  
from EMP LEFT outer join DEPT  
on emp.deptno = dept.deptno;
```

Pratik Joshi

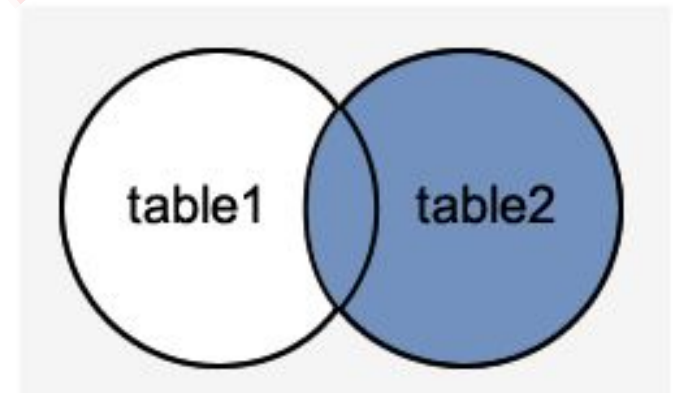


Right Outer Join

The Right Outer Join returns all rows from the right-hand table specified in the ON condition and only those rows from the other table where the join condition is met.

Syntax:

```
SELECT columns  
FROM table1  
RIGHT OUTER JOIN table2  
ON table1.column = table2.column;
```



E.g:

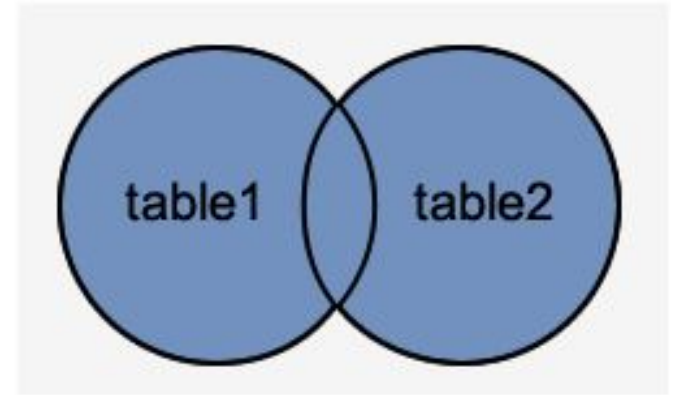
```
select ename,job,sal, emp.deptno, dept.deptno,dname,loc  
from EMP RIGHT outer join DEPT  
on emp.deptno = dept.deptno
```


Full Outer Join

The Full Outer Join returns all rows from the left hand table and right hand table. It places NULL where the join condition is not met.

Syntax:

```
SELECT columns  
FROM table1  
FULL [OUTER] JOIN table2  
ON table1.column = table2.column;
```



Example:

```
select ename,job,sal,emp.deptno,dept.deptno,dname,loc  
from emp full outer join dept on emp.deptno=dept.deptno
```

Self Join

Self Join is a specific type of Join. In Self Join, a table is joined with itself (Unary relationship). A self join simply specifies that each rows of a table is combined with itself and every other row of the table.

Syntax:

```
SELECT a.column_name, b.column_name...  
FROM table1 a, table1 b  
WHERE a.common_field = b.common_field;
```

Pratik Joshi



```
SQL> SELECT E.ENAME EMP_NAME,  
        M.ENAME MGR_NAME  
FROM EMP E,EMP M  
WHERE E.MGR=M.EMPNO;
```

EMP_NAME	MGR_NAME
----------	----------

SMITH	FORD
ALLEN	BLAKE
WARD	BLAKE
JONES	KING
MARTIN	BLAKE
BLAKE	KING
CLARK	KING

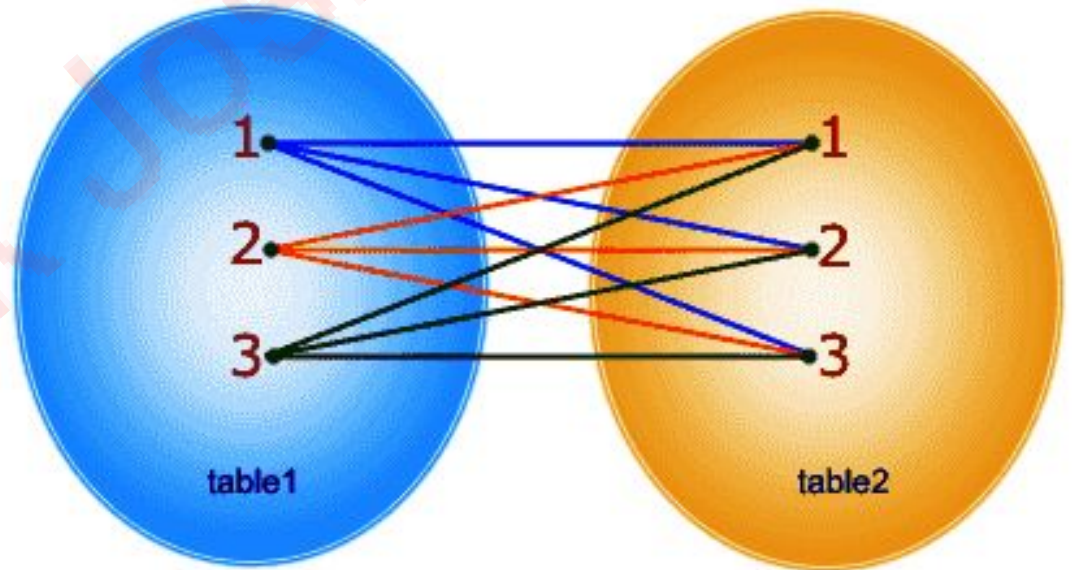
Cross Join

The CROSS JOIN specifies that all rows from first table join with all of the rows of second table. If there are "x" rows in table1 and "y" rows in table2 then the cross join result set have $x*y$ rows.

Syntax:

```
SELECT *  
FROM table1  
CROSS JOIN table2;
```

```
SELECT * FROM table1, table2
```



```
SELECT *  
FROM emp  
CROSS JOIN dept;
```

```
Select * from emp,dept;
```

Pratik Joshi



SET Operators

Set Operators work based on the set theory.

- UNION
- UNION ALL
- INTERSECT
- MINUS

Pratik Joshi



Union

In Oracle, UNION operator is used to combine the result sets of two or more Oracle SELECT statements. It combines the both SELECT statement and removes duplicate rows between them.

Each SELECT statement within the UNION operator must have the same number of fields in the result sets with similar data types.

Syntax:

```
SELECT col1, col2, ... coln  
FROM table1  
UNION  
SELECT col1, col2, ... coln  
FROM table2 ;
```



SQL> SELECT JOB FROM EMP WHERE DEPTNO=10;

MANAGER
PRESIDENT
CLERK

SQL> SELECT JOB FROM EMP WHERE DEPTNO=20;

CLERK
MANAGER
ANALYST
CLERK
ANALYST

SQL> SELECT JOB FROM EMP WHERE DEPTNO=10

UNION

SELECT JOB FROM EMP WHERE DEPTNO=20;

ANALYST
CLERK
MANAGER
PRESIDENT



Union All

In Oracle, the UNION ALL operator is used to combine the result sets of 2 or more SELECT statements. It is different from UNION operator in a way that it does not remove duplicate rows between the various SELECT statements.

Difference between UNION and UNION ALL operators:

UNION operator removes duplicate rows while UNION ALL operator does not remove duplicate rows.

Syntax:

```
SELECT col1, col2, ... coln  
FROM table1  
UNION ALL  
SELECT col1, col2, ... coln  
FROM table2 ;
```



```
SQL> SELECT JOB FROM EMP WHERE DEPTNO=10  
      UNION ALL  
      SELECT JOB FROM EMP WHERE DEPTNO=20;
```

JOB

MANAGER

PRESIDENT

CLERK

CLERK

MANAGER

ANALYST

CLERK

ANALYST

Pratik Joshi



Intersect

It is used to common or intersecting records from two SELECT queries.

Syntax:

```
SELECT expression1, expression2, ... expression_n  
FROM table1  
INTERSECT  
SELECT expression1, expression2, ... expression_n  
FROM table2 ;
```

Pratik Joshi



```
SQL> SELECT JOB FROM EMP WHERE DEPTNO=10  
INTERSECT  
SELECT JOB FROM EMP WHERE DEPTNO=20;
```

JOB

CLERK

MANAGER

Pratik Joshi

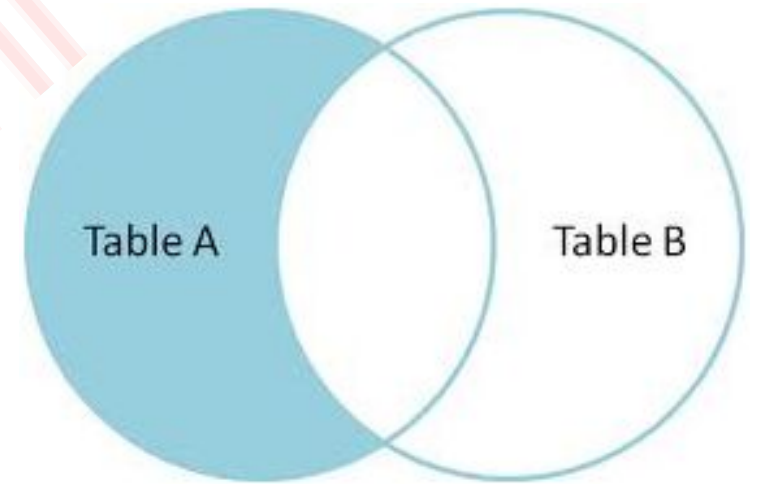


Minus

MINUS operator is used to return all rows in the first SELECT statement that are not returned by the second SELECT statement.

Syntax:

```
SELECT expression1, expression2, ... expression_n  
FROM table1  
MINUS  
SELECT expression1, expression2, ... expression_n  
FROM table2 ;
```



```
SQL> SELECT JOB FROM EMP WHERE DEPTNO=10  
      MINUS  
      SELECT JOB FROM EMP WHERE DEPTNO=20;  
JOB
```

```
-----  
PRESIDENT
```

Pratik Joshi

