JOINS :-

→ 'JOINS' are used to retrieve data from multiple tables at a time.

→ "ORACLE" supports the following two types of "JOINS" are

1. NON-ANSI JOINS (8i version JOINS)
2. ANSI JOINS (9i version JOINS)

1. NON-ANSI JOINS (WHERE)

1. EQUI JOIN.
2. NON EQUI JOIN.
3. SELF JOIN.

2. ANSI JOINS (ANSI)

1. INNER JOIN.
2. OUTER JOIN.
   - * LEFT OUTER JOIN.
   - * RIGHT OUTER JOIN.
   - * FULL OUTER JOIN.
3. CROSS JOIN / CARTISEAN JOIN.
4. NATURAL JOIN.

→ NON-ANSI JOINS ARE CALLED AS ORACLE 8i VERSION JOINS AND THESE ARE JOINING TABLES WITH "WHERE CLAUSE" CONDITION.

→ ANSI JOINS ARE CALLED AS ORACLE 9i VERSION JOINS AND THESE ARE JOINING TABLES WITH "ON CLAUSE" CONDITION.

Syntax for NON-ANSI JOINS :-

SELECT * FROM <TN1>, <TN2> WHERE <JOINING_CONDITION>;

Syntax for ANSI JOINS:

SELECT * FROM <TN1> <JOIN KEY> <TN2>
ON <JOINING CONDITION>;

JOINS TABLES:

SELECT * FROM STUDENT;

STID	SNAME	CID
1021	SAI	10
1022	ADAMS	20
1023	JONES	30

SELECT * FROM COURSE;

CID	CNAME	CFEE
10	C	4500
20	JAVA	6000
40	ORACLE	5000

05/06/2020

EQUI JOIN:

→ Retrieving data based on an equal operation is called as "EQUI JOIN".
→ we should have atleast one common column and those column data type must be matched.
→ we should use common column in IN JOIN condition level.
→ It always retrieve matching data or rows from tables.

Syntax:

WHERE <TN1>.<condition column> = <TN2>.<CC>;

(OR)

WHERE <TN ALIAS NAME>.<CC> = <TN2>.<C.C>;

Ex: write a query to display student and corresponding course details from student, source tables.

Ex:
SELECT * FROM STUDENT, COURSE WHERE
STUDENT.CID = COURSE.CID;

NOTE:- To avoide the above problem we should use a table name or an identity to the ambiguously column "CID" like below

SELECT * FROM STUDENT, COURSE WHERE CID = CID;
ORA-00918: column ambiguously defined

SELECT * FROM STUDENT, COURSE WHERE
STUDENT.CID = COURSE.CID;
SELECT * FROM STUDENT S, COURSE C WHERE
S.CID = C.CID;

RULE OF JOIN EXECUTION:-

→ row in a first table is comparing with all rows of second table

WHERE STUDENT.CID = COURSE.CID;

S.CID	=	C.CID
10	——	10
20	——	20
30	—X—	40

EX:- WAQ, to display student, course details from tables if course id = 20

SELECT * FROM STUDENT S, COURSE C WHERE
S.CID = C.CID AND C.CID = 20;

S.CID	matching	C.CID
10		10
20		20
30		40

EX:- WAQ to display employee name, Job and corresponding dept name FROM EMP, DEPT TABLES.

SELECT ENAME, JOB, DNAME FROM EMP E, DEPT D
WHERE E.DEPTNO = D.DEPTNO;

Ex: WAQ to display employee details who are working in "newyork"

SELECT * FROM EMP E, DEPTNO D WHERE
E.DEPTNO = D.DEPTNO AND LOC = 'NEW YORK';

16/6/2020

* WAQ find out total sal.

Sol:
SELECT DNAME, SUM(SAL) FROM EMP E, DEPT D
WHERE E.DEPTNO = D.DEPTNO GROUP BY DNAME;

DNAME	SUM(SAL)
ACCOUNTING	6300
RESEARCH	8378
SALES	9400

Ex: WAQ to display Department names and SUM of SAL DEPT is more than 8000

SELECT DNAME, SUM(SAL) FROM EMP E, DEPT D
WHERE E.DEPTNO = D.DEPTNO GROUP BY DNAME
HAVING SUM(SAL) > 8000;

DNAME	SUM(SAL)
RESEARCH	8378
SALES	9400

USING ROLLUP & CUBE CLAUSES IN JOINS:

SELECT DNAME, COUNT(*) FROM EMP E, DEPTNO D.
WHERE E.DEPTNO = D.DEPTNO GROUP BY ROLLUP(DNAME);

DNAME	COUNT(*)
ACCOUNTING	2
RESEARCH	2
SALES	6
	13

SELECT D.DEPTNO, DNAME, COUNT(*) FROM EMP E, DEPTNO D, WHERE E.DEPTNO = D.DEPTNO GROUP BY CUBE (D.DEPTNO, DNAME);

* INNER JOIN (ANSI JOIN)

→ It is ANSI formate JOIN similar to EQUI JOIN. IN NON ANSI JOIN'S

Syntax:

ON <TN1>.<COMON COLUMN> = <TN2>.<COMMON COLUMN>

(OR)

ON <TN1 ALIAS NAME>.<CC> = <TN2 ALIAS NAME>.<C.C>;

Ex: WAQ Display student, course details by using inner join,

SELECT * FROM STUDENT S INNER JOIN COURSE C ON S.CID = C.CID;

STID	SNAME	CID	CID	CNAME	FEE
1021	SA	10	10	C	1500
1022	NAVEEN	20	20	JAVA	6000

Ex: WAQ to display employ details who are working in the location is new York by using "INNER JOIN";

SELECT * FROM EMP E INNER JOIN DEPTNO D ON E.DEPTNO = D.DEPTNO AND LOC = 'NEW YORK';

ON → MANDATORY USING ON (MANDATORY) (OR) WHERE (OPTIONAL)

NOTE: the ADVANTAGE of ANSI formate JOINS are PORTABILITY. that means we can move ANSI formate Query's from one data base to from other Data base without making any changes as it is the Query can be Executed.

In other data base.

7/6/2020
8/6/2020

OUTER

* LEFT JOIN :-

SELECT * FROM STUDENT S LEFT OUTER COURSE C ON S.CID = C.CID; → ANSI FORMAT

SELECT * FROM STUDENT S, COURSE C WHERE S.CID = C.CID(+); → NON-ANSI FORMAT

Note: Heare (+) is called as outer join operator which can be used only one side at a time within join condition.

* RIGHT OUTER JOIN :-

→ Retriving matching rows from both tables and unmatching rows from right side table only

Ex:- SELECT * FROM STUDENT S RIGHT OUTER JOIN COURSE C ON S.CID = C.CID; → ANSI FORMAT

SELECT * FROM STUDENT S, COURSE C WHERE S.CID (+) = C.CID; → NON-ANSI

FULL OUTER JOIN :-

→ combination of LEFT RIGHT JOIN

→ By using full outer join we are retrieving matching rows of connecting rows from multiple table at a time

Ex :-

SELECT * FROM STUDENT S FULL OUTER JOIN COURSE C ON S.CID = C.CID; → ANS1

SELECT * FROM STUDENT S, COURSE C WHERE S.CID = C.CID (+)
UNION
SELECT * FROM STUDENT S, COURSE C WHERE S.CID (+) = C.CID; → NON-ANS1

NON-EQUI JOIN :-

→ Retrieving data from multiple tables based on any condition except AN "=" operator

→ In this non-equi join the following operators are <, >, <=, >=, !=, BETWEEN, AND etc.

SELECT * FROM TEST1

SNO	NAME
10	A
20	B

SELECT * FROM TEST2

SNO	SAL
10	25000
20	25000

SELECT * FROM TEST T1, TEST T2 WHERE T1.SNO>T2.SNO;

SNO	NAME	SNO	SAL
20	B	10	25000

Ex:- To display Emply detail who's salary is/s between Low sal and high sal

SELECT ENAME, SAL, LOSAL, HISAL FROM EMP, SALGRADE WHERE SAL BETWEEN LOSAL AND HISAL;

(OR)

SELECT ENAME, SAL, LOSAL, HISAL FROM EMP, SALGRADE WHERE (SAL>=LOSAL) AND (SAL<=HISAL);

SELECT ENAME, SAL, LOSAL, HISAL, GRADE FROM EMP, SALGRADE WHERE SAL BETWEEN

SELECT ENAME, SAL, LOSAL, HISAL, GRADE FROM EMP JOIN SALGRADE ON SAL BETWEEN LOSAL AND HISAL AND GRADE=2; -NON-ANSI (OTHER — NON-ANSI)

CROSS JOIN / CARTESIAN JOIN:-

-> JOINING two or more than two tables without any condition.

-> In CROSS JOIN MECHANISM each row of a table will join's each row of another table

-> a table is having m no. of rows AND Another table having n no. of rows then the result is m CROSS n CROSS

Ex: SELECT * FROM STUDENT CROSS JOIN COURSES; -> ANSI — JOIN

(or)

Ex: SELECT * FROM STUDENT, COURSES; - NON

9/6/2020

NATURAL JOIN :-

-> Natural JOIN is similar to "EQUI JOIN".

-> Whenever we use NATURAL JOIN's we should have common column name in both tables.

-> When we use NATURAL JOIN's there is no need to write a "JOIN" condition by the user because internally oracle database is preparing a JOIN condition based on common column name with an equal operator automatically.

-> So, that natural joins also retrieving only matching rows from the tables just like EQUI (or) INNER JOIN's.

By using NATURAL JOIN's we can avoide duplicate column names, while retrieving data tables

SELECT * FROM STUDENT S NATURAL JOIN COURSES C;

SID	STID	SNAME	CNAME	CFEE
10	1021	SAI	C	1500.
20	1022	ADAMS	JAVA	6000.

NOTE: this JOIN is possible in "ANSI" format only.

SELF-JOIN:

-> Joining a table by itself is called as "SELF JOIN".

-> whenever we use self JOIN we should create alias names on a table.

-> we can create any numbers of alias names on a single table but every alias name should be different.

-> when we create alias name on a table internaly oracle database is preparing virtual copy table each alias name and stored in buffer memory (temp memory).

-> there are two situations to use SELF JOIN mechanism.

1. comparing a single column value by itself in table
2. comparing two diff columns value to each other within the same table.

ex: single colm value:

WAQ to display employee who are working in the location of the employee sal

SELECT * FROM TEST;

ENAME	LOC
SAI	HYD
JONES	MUMBAI
ALLEN	CHENNAI
WARD	HYD

SELECT T1.ENAME, T1.LOC FROM TEST T1, TEST T1 WHERE T1.LOC=T2.LOC AND T2.ENAME='SAI';

ENAME	LOC
SAI	HYD
WARD	HYD

Ex: WAQ to display employee who's employee salary is same as the employee SCOTT sal

SELECT E1.ENAME, E1.SAL FROM EMP E1, EMP E2 WHERE E1.SAL=E2.SAL AND E2.ENAME='SCOTT';

(NON-ANSI)

ENAME	SAL
SCOTT	3000
FORD	3000

SELECT E1.ENAME, E1.SAL FROM EMP E1 JOIN EMP E2 ON E1.SAL = E2.SAL AND E2.ENAME='SCOTT';

(ANSI)

11/6/20

Ex: WAQ to display employee who are joined before their managers.

SELECT M.ENAME AS MANAGERS, M.HIREDATE AS MGRDOJ, E.ENAME AS EMPLOYEES, E.HIREDATE AS EMPDOJ FROM EMP E, EMP M WHERE M.EMPNO = E.MGR AND E.HIREDATE < M.HIREDATE;

Using clause in JOINS:

→ It was introduced oracle 9i version. which is used in place of ON CLUASE in ANSI-FORMATE JOIN.

→ when we use "USING" CLAUSE common column name must be same

SELECT * FROM STUDENT S INNER JOIN COURSE C USING (CID);

Note:- when we using class a comman column name should not be prifixed with it table alias name

USING (S.CID) → NOT ALLOWED.

USING (CID) → ALLOWED.

* How to join more than two tables:-

Syntax FOR NON-ANSI FORMAT:-

SELECT * FROM <TN1>, <TN2>, <TN3> ——

WHERE <CONDITION1> AND <CONDITION2> AND <C> ——

Syntax for ANSI format:

SELECT * FROM <TN1> <JOIN KEY> <TN2>
ON <CONDITION1> <JOIN KEY> <TN3>
ON <CONDITION2> <JOIN KEY> <TN4>
ON <CONDITION3> ----;

EQUI JOIN:

SELECT * FROM REGISTER;

REGNO	REGDATE	CID
1	05-APR-20	10
2	06-APR-20	20
3	07-APR-20	30

SELECT * FROM STUDENT S, COURSE C, REGISTER R
WHERE S.CID = C.CID AND C.CID = R.CID;

INNER JOIN:

SELECT * FROM STUDENT S INNER JOIN COURSE C
ON S.CID = C.CID INNER JOIN REGISTER R ON
C.CID = R.CID;

LEFT OUTER JOIN:

SELECT * FROM STUDENT S LEFT OUTER JOIN
COURSE C ON S.CID = C.CID LEFT OUTER JOIN
REGISTER R ON C.CID = R.CID;

LOGIC:

	LEFT	MID	RIGHT
CASE1:	S	C	R
CASE2:	C	R	S
CASE3:	R	S	C

Diff B/W SET OPERATORS & JOINS

SO (Set operators)	JOINS
1. To combine data Vertically 2. To combine rows wise	1. To combine data Horizontally 2. To combine colum wise