

DISPLAYING DATA FROM MULTIPLE TABLES -- CHAPTER 4 --

- Obtaining data from multiple tables: Sometimes you need to use data from more than one table. In the slide example, the report displays data from two separate tables.
- **Join:** When data from more than one table in a database is required, a join condition is used. Rows in one table can be joined to rows in another table according to common values existing in corresponding columns, that is, usually primary and foreign key columns. The syntax used within the WHERE clause to join information from two tables is as follow...

```
WHERE...
    ...Table1.column...
    ...table1.column1 = table2.column2;
```

- In the statements above, Table1.column denotes the column from which the information is to be extracted, and the statement table1.column1 = table2.column2 is the condition that must be true before an element from table1.column is selected.
- Cartesian Product: The combination of all rows between two table. It is formed when...
 - A join condition is omitted
 - A join condition is invalid
 - All rows in the first table are joined to all rows in the second table
- Types of joins: Equijoin, Non-equijoin, Outer join, Self join, Set Operators.
 - **Equijoin:** values in the two tables are equal...(i.e., the use of a primary key and a foreign key).
 - Qualifying column names with table name scan be very time consuming, particularly if table names are lengthy. You can use table aliases instead of table names. Just as a column alias gives a column another name, a table alias gives a table another name.

```
SQL> r
1  select e.empno, e.ename, e.deptno, d.deptno, d.loc
2  from emp e, dept d
3* where e.deptno = d.deptno
```

EMPNO	ENAME	DEPTNO	DEPTNO	LOC
7369	SMITH	20	20	DALLAS
7499	ALLEN	30	20	CHICAGO
7521	WARD	30	30	CHICAGO
...				

- Note that the table alias is declared in the second line in the “FROM” clause. While table aliases can be up to 30 characters long, the point is usually to make them as small as possible for efficiency.
 - **Non-Equijoins:** Table joins where there isn’t necessarily a shared column between tables. These tables are united by using an operator other than ‘=’ such as “>, <, BETWEEN” etc. Here’s a small example...

SEE BELOW...

```
SQL> r
  1 select e.ename, e.sal, s.grade
  2 from emp e, salgrade s
  3* where e.sal between s.losal and s.hisal
```

ENAME	SAL	GRADE
ALLEN	1000	1
WARD	1000	1
TURNER	1000	1
BLACK	1000	1
...		

- Recall that when using the BETWEEN clause, the order of the arguments does matter.
 - Outer Joins:** Can be used to return records with no direct match between tables. The operator used for these joins is the plus operator (+). Here's the syntax used...

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.colulmn(+) = table2.column;
Or
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.colulmn = table2.columnn(+);
```

- The plus operator is placed on the side of the join that is deficient of information. This operator has the effect of creating one or more null rows, to which one or more rows from the non-deficient table can be joined. Let's look at an example...**

```
SQL> r
  1 select e.deptno, e.ename, d.deptno, d.dname
  2 from emp e, dept d
  3* where e.mgr(+) = d.deptno
```

DEPTNO	ENAME	DEPTNO	DNAME
		10	ACCOUNTING
		20	RESEARCH
		30	SALES
		40	OPERATIONS
		50	DEVELOPMENT

- In the example above, none of the records meet the criteria, but since this is an outer join, the all of the records from e.mgr are included with their respective null columns. Note that DEPTNO is a column shared by both tables, and therefore, this column does appear here.**

- Note that only one side in the WHERE clause may have the outer join operator, and this is the side that lacks the information. Note also that a condition involving the outer join, cannot use the IN operator or be linked to another condition using the OR operator.

- **Self Joins:** Sometimes you need to join a table to itself. To find the name of each employee's manager, you need to join the emp table to itself, or perform a self join.

```
SQL> r
1 select worker.empno, worker.ename, worker.mgr,
2 worker.ename || ' works for ' || manager.ename
3 from emp worker, emp manager
4* where worker.mgr = manager.empno
```

EMPNO	ENAME	MGR	WORKER.ENAME 'WORKSFOR' MANAG
7499	ALLEN	7698	ALLEN works for BLAKE
7521	WARD	7698	WARD works for BLAKE
7566	JONES	7839	JONES works for KING
7698	BLAKE	7839	BLAKE works for KING
7782	CLARK	7839	CLARK works for KING
7788	SCOTT	7566	SCOTT works for JONES
7844	TURNER	7698	TURNER works for BLAKE
7876	ADAMS	7788	ADAMS works for SCOTT
7096	BLACK	7782	BLACK works for CLARK
9666	SHANK	9300	SHANK works for REDCORNER

- Note that not all of the managers are listed here. For example, SHANK at the bottom works for REDCORNER, but REDCORNER does not appear on the left...this is because REDCORNER does not have a manager listed, according to this report. Only those employees who have a manager appear here.