Introduction to Oracle SQL
Working with Joins

Sukhjit Singh

Session 3

Goals:

- Data from multiple tables using Joins
  - Join Condition
  - Types of Joins
  - Oracle 9i new join syntax
  - Join View
- Data dictionary views

Joins

- Joins are used to query data from multiple tables or views
- Rows in one table can be joined to rows in another table (primary and foreign key)
- The join condition is written in the WHERE clause

Joins

- In a join,
  - FROM clause of a query has two or more table names separated by commas
  - The SELECT list includes columns from any of the tables in the FROM clause
  - WHERE clause gives the join condition

Joins

- Syntax:

```
SQL> SELECT a.column, b.column
     FROM table1 a, table2 b
     WHERE a.column = b.column;
```

- Joins commonly involve the use of table aliases like a for table1, b for table2
- Two or more columns with the same name should be qualified with the table alias to avoid ambiguouity

Cartesian Product

- Cartesian Product is the result set of joining two tables, without the join condition
- Number of rows in Cartesian Product is the product of number of rows in each table.

```
SQL> SELECT e.ename, e.empno, d.name
     FROM emp e, dept d;
```

Join Condition

- Condition in the WHERE clause, which relates two tables

```
SQL> SELECT e.ename, d.name
     FROM emp e, dept d
     WHERE e.deptno = d.deptno;
```

- Usually, join condition specified on the primary key column of one table and foreign key column of the other table

Join Condition

- Total # of join conditions = # of tables – 1
- Must involve columns with the compatible datatypes
- Columns in the join condition need not be there in the SELECT list
- Involves (=) operator or other operators
- Can have additional search conditions in the WHERE clause, besides the join condition.

Join Condition

- Determines if a join is Equijoin or Non-Equijoin
- Equijoin involves join condition relating two tables by an (=) operator

```
SQL> SELECT e.ename, d.name
     FROM emp e, dept d
     WHERE e.deptno = d.deptno;
```

Join Condition

- Non-Equijoin involves join condition relating two tables by an operator , other than equality

```
SQL> SELECT e.ename, d.name
     FROM emp e, salgrade s, dept d
     WHERE  e.deptno = d.deptno AND
     e.sal BETWEEN s.losal AND
     s.highsal;
```

Types of Joins

- Inner Joins
- Outer Joins
- Self Joins

Types of Joins

```
SQL> SELECT * FROM emp;
EMPNO   ENAME    DEPTNO HIREDATE    JOB        COMM
-----  --------  ------ ----------  ---------  -----
7566   FORD          20 10-MAR-01   CLERK
7567   SMITH         30 1-JAN-01    MANAGER      100
7568   MARTIN        10 1-MAR-01    ANALYST

SQL> SELECT * FROM dept;
DEPTNO   NAME         LOCATION
-------  -----------  ---------
     10  ACCOUNTING   BOSTON
     20  SALES        SAN FRANCISCO
     30  TRAINING     HOUSTON
     40  MARKETING    SEATTLE
```

Inner Joins

- Regular joins, which satisfy the join condition
- Each row returned by an inner join contains data from all tables involved in the join

Inner Joins

```
SQL> SELECT e.ename, e.empno, d.name,
     e.hiredate
     FROM emp e, dept d
     WHERE e.deptno = d.deptno
     AND e.ename = 'FORD';

ENAME EMPNO  NAME   HIREDATE
----- ------ ------ ----------
FORD    7566 SALES  10-MAR-01
```

Outer Joins

- Extension to the inner joins
- Returns the rows, which satisfy the join condition and also rows from one table, for which no corresponding rows exist in the other table
- Use an outer join operator (+).
  - Right outer join
  - Left outer join (depending on where + operator is)
- The operator is placed on the side of the join that is **deficient** in information.

Outer Joins

- Syntax:

```
SQL> SELECT e.name, d.deptno
FROM emp e, dept d
WHERE e.deptno (+)= d.deptno;

ENAME      DEPTNO
-----      -------
FORD            30
SMITH           20
MARTIN          10
                40 - No employees in dept.
```

Restrictions on Outer Joins

- Outer join operator can appear on only one side of an expression in the join condition
- This query will generate error:

```
SQL> SELECT e.name, d.name
     FROM emp e, dept d
     WHERE e.deptno (+) =
     d.deptno(+);
```

Restrictions on Outer Joins

- If a join involves more than two tables, then one table can't be outer joined with more than one table in the query. Error query:

```
SQL> SELECT e.ename, d.name
     FROM emp e, dept d, salgrade s
     WHERE e.deptno (+) = d.deptno
     AND e.grade (+) = s.grade;
```

  - To workaround this issue, create a view with an outer join between two tables and then outer join the view with the third table.

Restrictions on Outer Joins

- Outer join condition, containing the (+) operator cannot use IN operator
- Invalid Query

```
SQL> SELECT e.name, e.hiredate,d.name
     FROM emp e, dept d
     WHERE e.deptno (+) IN (20, 30);
```

Restrictions on Outer Joins

- Outer join condition may not be combined with another condition using the OR operator. Invalid query:

```
SQL> SELECT e.name, e.hiredate,d.name
     FROM emp e, dept d
     WHERE e.deptno (+) = d.deptno
     OR d.deptno = 20;
```

Restrictions on Outer Joins

- A condition involving the (+) operator cannot involve a subquery (will be discussed in later sessions). Invalid query:

```
SQL> SELECT e.name, e.hiredate,d.name
     FROM emp e, dept d
     WHERE e.deptno (+) = (SELECT
     deptno FROM dept WHERE name =
     'ACCOUNTING');
```

Restrictions on Outer Joins

- As a workaround for the above query:

```
SQL> SELECT e.name, e.hiredate,d.name
     FROM emp e, (SELECT deptno FROM dept
     WHERE name = 'ACCOUNTING') v
     WHERE e.deptno(+) = v.deptno;

ENAME        HIREDATE     NAME
--------     ---------    ---------
MARTIN       1-MAR-01     ACCOUNTING
```

Full Outer Joins

- Used for including rows from table A and B, which are:
  - From the result of the inner join
  - From A, that don't have corresponding rows in B
  - From B, that don't have corresponding rows in A

Full Outer Joins

```
SQL> DESC location
Name                 Null?      Type
-----------------    --------  -----------
Location_Id          NOT NULL Number(3)
Regional_Group                 Varchar2(20)

SQL> SELECT * FROM location;
LOCATION_ID REGIONAL_GROUP
----------- ---------------
          1 NEW YORK
          2 BOSTON
          3 CHICAGO
```

Full Outer Joins

```
SQL> DESC department
Name                 Null?      Type
-----------------    --------  -----
Dept_Id              NOT NULL Number(2)
Name                          Varchar2(14)
Location_Id                   Number(3)

SQL> SELECT * FROM department;
DEPT_ID       NAME          LOCATION_ID
-------       ------------  -----------
      10      SALES                   1
      20      TRAINING                1
      30      ACCOUNTING              2
      40      OPERATIONS
```

Full Outer Joins

- Assume there are locations in the LOCATION table, that don't have corresponding departments in the DEPARTMENT table
- At the same time, there are departments in the DEPARTMENT table, that don't a corresponding LOCATION_ID in the LOCATION table.

Full Outer Joins

- This inner join will retrieve only the departments and locations that have corresponding rows in both the tables.

```
SQL> SELECT d.dept_id, d.name, l.regional_group
     FROM department d, location l
     WHERE d.location_id = l.location_id;
DEPT_ID     NAME          REGIONAL_GROUP
-------     -----------   --------------
     10     SALES         NEW YORK
      20    TRAINING      NEW YORK
      30    ACCOUNTING    BOSTON
```

Full Outer Joins

- This outer join makes the DEPARTMENT table optional in the query
- Retrieves locations that don't have any departments.

```
SQL> SELECT d.dept_id, d.name, l.regional_group
     FROM department d, location l
     WHERE d.location_id(+) = l.location_id;
DEPT_ID    NAME         REGIONAL_GROUP
-------    -----------  --------------
    10     SALES        NEW YORK
    20     TRAINING     NEW YORK
    30     ACCOUNTING   BOSTON
                        CHICAGO
```

Full Outer Joins

- This outer join makes the LOCATION table optional in the query.
  - Retrieves all departments, including that don't belong to any location.

```
SQL> SELECT d.dept_id, d.name, l.regional_group
     FROM department d, location l
     WHERE d.location_id = l.location_id(+);
DEPT_ID    NAME          REGIONAL_GROUP
-------    -----------   --------------
     10    SALES         NEW YORK
     20    TRAINING      NEW YORK
     30    ACCOUNTING    BOSTON
     40    OPERATIONS
```

Full Outer Joins

- Two-sided outer join are not allowed in Oracle, to retrieve departments without locations and locations without departments.
- Workaround for this problem: UNION
- UNION includes all the rows from both the tables (as would be expected in a full outer join) and eliminates the duplicate rows

Union

```
SQL> SELECT d.dept_id, d.name,
     l.regional_group
     FROM department d, location l
     WHERE d.location_id = l.location_id(+)
     UNION
     SELECT d.dept_id, d.name,
     l.regional_group
     FROM department d, location l
     WHERE d.location_id(+) =
     l.location_id;
```

Union

DEPT_ID	NAME	REGIONAL_GROUP
10	SALES	NEW YORK
20	TRAINING	NEW YORK
30	ACCOUNTING	BOSTON
40	OPERATIONS	
		CHICAGO