

DBMS SQL

Lesson 6: Joins and Subqueries



Lesson Objectives

- To understand the following topics:
 - Join
 - Oracle Proprietary Joins
 - SQL: 1999 Compliant Joins
 - Sub-queries
 - Co-related sub-query
 - Exists / Non-Exists Operators
 - CONNECT BY and START WITH clauses





6.1: Joins

What are Joins?

- If we require data from more than one table in the database, then a join is used.
 - Tables are joined on columns, which have the same “data type” and “data width” in the tables.
 - The JOIN operator specifies how to relate tables in the query.
 - When you join two tables a Cartesian product is formed, by default.
 - Oracle supports
 - Oracle Proprietary
 - SQL: 1999 Compliant Joins



6.1: Joins

Types of Joins

➤ Given below is a list of JOINS supported by Oracle:

Oracle Proprietary Joins	SQL: 1999 Compliant Joins
Cartesian Product	Cross Joins
Equijoin	Inner Joins (Natural Joins)
Outer-join	Left, Right, Full outer joins
Non-equijoin	Join on
Self-join	Join on



Cartesian Joins

- A Cartesian product is a product of all the rows of all the tables in the query.
- A Cartesian product is formed when the join condition is omitted or it is invalid
- To avoid having Cartesian product always include a valid join condition

Example

```
SELECT Student_Name, Dept_Name  
FROM Student_Master, Department_Master;
```



Guidelines for Joining Tables

- The JOIN condition is written in the WHERE clause
- The column names which appear in more than one table should be prefixed with the table name
- To improve performance of the query, table name prefix can be include for the other selected columns too



EquiJoin

- In an Equijoin, the WHERE statement compares two columns from two tables with the equivalence operator "=".
- This JOIN returns all rows from both tables, where there is a match.

Syntax :

```
SELECT <col1>, <col2>, ...  
FROM <table1>, <table2>  
Where <table1>.<col1>=<table2>.<col2>  
[AND <condition>] [ORDER BY <col1>, <col2>, ...]
```



6.2: Oracle Proprietary Joins

EquiJoin - Example

Example 1: To display student code and name along with the department name to which they belong

```
SELECT Student_Code, Student_name, Dept_name  
FROM Student_Master , Department_Master  
WHERE Student_Master.Dept_code =  
       Department_Master.Dept_code;
```

Example 2: To display student and staff name along with the department name to which they belong

```
SELECT student_name, staff_name, dept_name  
FROM student_master, department_master, staff_master  
WHERE student_master.dept_code=department_master.dept_code  
and staff_master.dept_code=department_master.dept_code;
```




Non-EquiJoin

- A non-equi join is based on condition other than an equality operator
- Example: To display details of staff_members who receive salary
- in the range defined as per grade

```
SELECT s.staff_name,s.staff_sal,sl.grade  
FROM staff_master s,salgrade sl  
WHERE staff_sal BETWEEN sl.losal and sl.hisal
```



Outer Join

- If a row does not satisfy a JOIN condition, then the row will not appear in the query result.
- The missing row(s) can be returned by using OUTER JOIN operator in the JOIN condition.
- The operator is PLUS sign enclosed in parentheses (+), and is placed on the side of the join(table), which is deficient in information.

WHERE table1 <OUTER JOIN INDICATOR> = table 2



Outer Join

Syntax

- `Table1.column = table2.column (+)` means OUTER join is taken on table1.
- The (+) sign must be kept on the side of the join that is deficient in information
- Depending on the position of the outer join (+), it can be denoted as Left Outer or Right outer Join



Outer Join - Example

- To display Department details which have staff members and also display department details who do not have any staff members

```
SELECT  
staff.staff_code,staff.Dept_Code,dept.Dept_name  
FROM Staff_master staff, Department_Master dept  
WHERE staff.Dept_Code(+) = dept.Dept_Code
```



Self Join

- In Self Join, two rows from the “same table” combine to form a “resultant row”.
- It is possible to join a table to itself, as if they were two separate tables, by using aliases for table names.
 - This allows joining of rows in the same table.

Example: To display staff member information along with their
➤ manager information

```
SELECT staff.staff_code, staff.staff_name,  
       mgr.staff_code, mgr.staff_name  
FROM staff_master staff, staff_master mgr  
WHERE staff.mgr_code = mgr.staff_code;
```



SQL: 1999 Compliant Joins - Syntax

➤ Syntax:

```
SELECT table1.column, table2.column  
FROM table1  
[CROSS JOIN table2] |  
[NATURAL JOIN table2] |  
[JOIN table2 USING (column_name)] |  
[JOIN table2 ON (table1.column_name =  
                  table2.column_name)] |  
[LEFT|RIGHT|FULL OUTER JOIN table2  
ON (table1.column_name = table2.column_name)];
```



Cross Join

➤ The Cross Join and Cartesian product are the same, which produces the cross-product of the tables

Example: Cross Join on Student_Master and Department_Master

```
SELECT student_name, dept_name  
FROM student_master  
CROSS JOIN department_master;
```



Natural Join

- The Natural Join is based on the all columns that have same name and datatype in the tables include in the query
- All the rows that have equal values in the matched columns are fetched

Example: To display student details along with their department details

```
SELECT Student_Code, Student_name, Dept_Code,  
       Dept_name  
FROM Student_Master  
NATURAL JOIN Department_Master
```




USING clause

- The USING clause can be replace the NATURAL JOIN if the columns have same names but data types do not match.
- The table name or aliases should not be used in the referenced columns
- This clause should be used to match only one column when there are more than one column matches



USING clause - Example

Example 1: To display student details along with their department details. The department code does not match in datatype, hence the join is performed with the USING clause

```
SELECT student_code, student_name, dept_code, dept_name  
FROM student_master  
JOIN department_master  
USING (dept_code, dept_code);
```



ON clause

- Explicit join condition can be specified by using ON clause
- Other search conditions can be specified in addition to join condition

Example: To display student along with department details from Computer Science department

```
SELECT student.student_code, student.student_name,  
       student.dept_code, dept.dept_name  
FROM student_master student  
JOIN department_master dept  
ON (student.dept_Code = dept.dept_Code)  
AND dept.dept_Name ='Computer Science' ;
```



6.3: SQL:1999 Compliant Joins

LEFT, RIGHT & FULL Outer Join

- A join between two tables that return rows that match the join condition and also unmatched rows from left table is LEFT OUTER JOIN
- A join between two tables that return rows that match the join condition and unmatched rows from the right table is RIGHT OUTER JOIN
- A join between two tables that return rows that match the join condition and returns unmatched rows of both left and right table is a full outer join



LEFT, RIGHT & FULL Outer Join - Example

Example 1: Display student & department details and also those departments who do have students

```
SELECT s.student_code, s.dept_code, d.dept_name  
FROM student_master s  
RIGHT OUTER JOIN department_master d  
ON (s.dept_code = d.dept_code);
```

Example 2 Display student & department details, also those students who are not assigned to any department

```
SELECT s.student_code, s.dept_code, d.dept_name  
FROM student_master s  
LEFT OUTER JOIN department_master d  
ON (s.dept_code = d.dept_code);
```

LEFT, RIGHT & FULL Outer Join - Example

Example 3: Display student & department details. Also those departments who do have students and students who are not assigned to any department

```
SELECT s.student_code,s.dept_code,d.dept_name  
FROM student_master s  
FULL OUTER JOIN department_master d  
ON (s.dept_code = d.dept_code );
```



6.4: Subqueries

What is a SubQuery?

- A sub-query is a form of an SQL statement that appears inside another SQL statement.
 - It is also called as a “nested query”.
- The statement, which contains the sub-query, is called the “parent statement”.
- The “parent statement” uses the rows returned by the sub-query.



6.4: Subqueries

Subquery - Examples

Example 1: To display name of students from "Mechanics" department.

Method 1:

```
SELECT Dept_Code FROM Department_Master  
WHERE Dept_name = 'Mechanics';
```

O/P : 40

```
SELECT student_code,student_name FROM student_master  
WHERE dept_code=40;
```


Subquery - Examples

Example 1 (contd.):

Method 2: Using sub-query

```
SELECT student_code, student_name
FROM student_master
WHERE dept_code = (SELECT dept_code
                   FROM department_master
                   WHERE dept_name = 'Mechanics');
```



Where to use Subqueries?

- Subqueries can be used for the following purpose :
 - To insert records in a target table.
 - To create tables and insert records in the table created.
 - To update records in the target table.
 - To create views.
 - To provide values for conditions in the clauses, like WHERE, HAVING, IN, etc., which are used with SELECT, UPDATE and DELETE statements.



6.3: Subqueries

Comparison Operators for Subqueries

➤ Types of SubQueries

- Single Row Subquery
- Multiple Row Subquery.

➤ Some comparison operators for subqueries:

Operator	Description
IN	Equals to any member of
NOT IN	Not equal to any member of
*ANY	compare value to every value returned by sub-query using operator *
*ALL	compare value to all values returned by sub-query using operator *



Using Comparison Operators - Examples

Example 1: To display all staff details of who earn salary least

Salary

```
SELECT staff_name, staff_code, staff_sal
FROM staff_master
WHERE staff_sal = (SELECT MIN(staff_sal)
                   FROM staff_master) ;
```

Example 2: To display staff details who earn salary greater than average salary earned in dept 10

```
SELECT staff_code,staff_sal FROM staff_master
WHERE staff_sal > ANY(SELECT AVG(staff_sal)
FROM staff_master WHERE dept_code=10);
```



What is a Co-related Subquery?

- A sub-query becomes “co-related”, when the sub-query references a column from a table in the “parent query”.
 - A co-related sub-query is evaluated once for each row processed by the “parent statement”, which can be either SELECT, UPDATE, or DELETE statement.
 - A co-related sub-query is used whenever a sub-query must return a “different result” for each “candidate row” considered by the “parent query”.



Co-related Subquery -Examples

- Example 2: To display staff details whose salary is greater than the average salary in their own department:

```
SELECT staff_name, staff_sal , dept_code  
FROM staff_Master s  
WHERE staff_sal > (SELECT AVG(staff_sal)  
FROM staff_Master m  
WHERE s.dept_code = m.dept_code );
```



EXISTS/ NOT EXISTS Operator

- The EXISTS / NOT EXISTS operator enables to test whether a value retrieved by the Outer query exists in the result-set of the values retrieved by the Inner query.
 - The EXISTS / NOT EXISTS operator is usually used with a co-related sub-query.
 - If the query returns at least one row, the operator returns TRUE.
 - If the value does not exist, it returns FALSE.
 - The NOT EXISTS operator enables to test whether a value retrieved by the Outer query is not a part of the result-set of the values retrieved by the Inner query.



EXISTS/ NOT EXISTS Operator - Examples

- Example 1: To display details of employees who have some other employees reporting to them.

```
SELECT staff_code, staff_name FROM staff_master staff
WHERE EXISTS (SELECT mgr_code FROM staff_master mgr WHERE
mgr.mgr_code = staff.staff_code) ;
```

- Example 2: To display details of departments which have employees working in it.

```
SELECT dept_code,dept_name FROM department_master
WHERE EXISTS ( SELECT dept_code FROM staff_master
                WHERE staff_master.dept_code =
                department_master.dept_code) ;
```



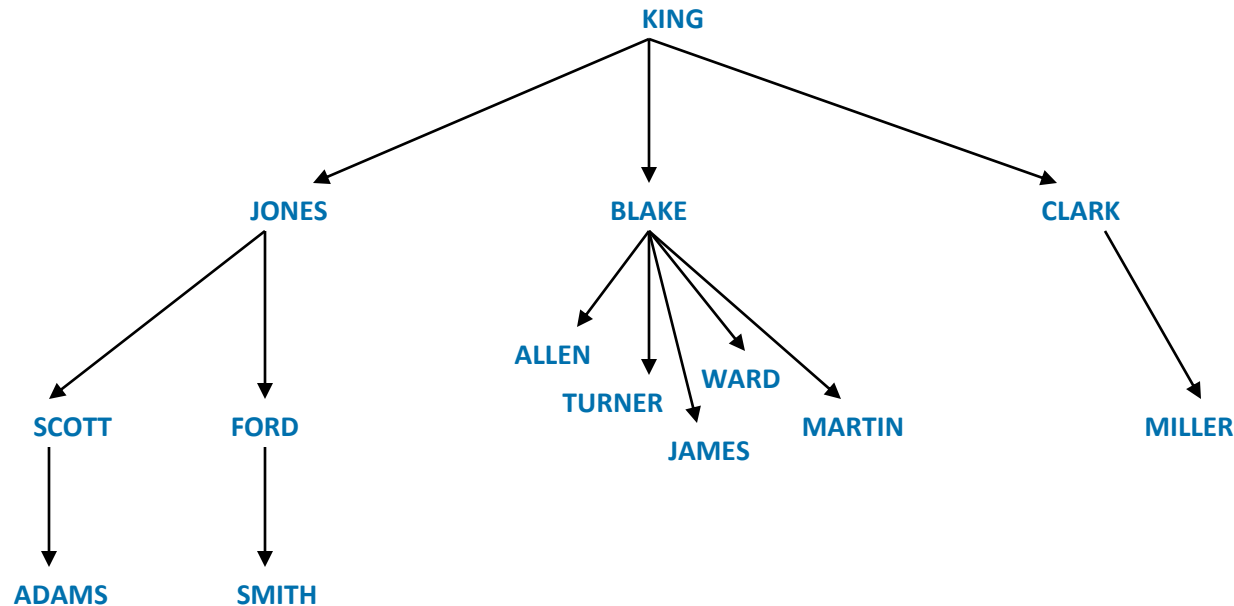

CONNECT BY and START WITH Clauses

- The START WITH .. CONNECT BY clause can be used to select data that has a hierarchical relationship
 - Usually, they have some sort of parent-child relationship.
 - They are used to retrieve rows, which are connected to each other through a tree-like structure.



CONNECT BY and START WITH Clauses

- The earliest ancestor in the tree is called the root-node called as a trunk. Extending from the trunk are branches, which have other branches.





CONNECT BY and START WITH Clauses

- The restrictions on SELECT statements performing hierarchical queries are as follows :
 - A SELECT statement that performs a hierarchical query cannot perform a JOIN.
 - If an ORDER BY clause is used in a hierarchical query, then Oracle orders rows using the ORDER BY clause rather than in a hierarchical fashion.

CONNECT BY, START WITH Clauses-Examples



➤ Example 1: To list "Allen" and his subordinates

```
SELECT staff_name, staff_code, mgr_code  
FROM staff_master  
CONNECT BY PRIOR staff_code = mgr_code  
START WITH staff_name = 'Allen';
```

Note: If START WITH clause is omitted, then the tree structure is generated for each of the rows in the EMP table.



Quick Guidelines

➤ For Using Subqueries

- Should be enclosed in parenthesis
- They should be placed on the right side of the comparison condition
- Cannot use ORDER By clause in subquery unless performing top-n analysis
- Use operator carefully. Single Row operators for Single Row Subquery and Multiple Row operator for Multiple Row Subquery





Quick Guidelines

- Restrict using the NOT IN clause, which offers poor performance because the optimizer has to use a nested table scan to perform this activity.
- Instead try to use one of the following options, all of which offer better performance:
 - Use EXISTS or NOT EXISTS
 - Use IN
 - Perform a LEFT OUTER JOIN and check for a NULL condition





Quick Guidelines

- If you have a choice of using the IN or the EXISTS clauses in your SQL, use the EXISTS clause as it is usually more efficient and performs faster.
 - Consider EXISTS in place of table joins.
 - Consider NOT EXISTS in place of NOT IN.





Summary

- In this lesson, you have learnt:
 - Joins
 - Oracle Proprietary Joins
 - SQL: 1999 Compliant Joins
- Sub-queries
 - Co-related sub-query
 - Exists / Non-Exists Operators
- CONNECT BY and START WITH clauses





Review – Match the Following

1. Equi Join	a. is based on any other operator other than equality
2. Non-equijoin	b. Is based on equality operator
3. Outer Join	c. Joins the table to itself
4. Self Join	d. includes a "+" operator with equality operator





Review – Questions

- Question 1: The SQL compliant join which is same as EquiJoin.
 - Option 1: Cross Join
 - Option 2: Natural Join
 - Option 3: Full Outer Join

- Question 2: A sub-query is also sometimes termed as ____.





Review – Questions

- Question 3: A sub-query can be used for creating and inserting records.
 - True / False

- Question 4: If a sub-query returns multiple values, then the valid operators is/are _____.
 - Option 1: =
 - Option 2: IN
 - Option 3: >
 - Option 4: Any

