



Inner, Outer, Full? Oracle9i Join Syntax

Presented to: New York Oracle Users Group

Presented on: September 26, 2002

Presented by: John Jay King
King Training Resources
john@kingtraining.com

Download this paper and code examples from:

<http://www.kingtraining.com>



- Become familiar with ISO/ANSI standard join syntax added to Oracle9i
- Know how to use new inner-join semantics
- Become familiar with the separation of join criteria from other row selection criteria
- Understand how the ISO/ANSI syntax improves upon the Oracle Outer Join operator (+)
- Know the difference between Left, Right, and Full Outer Join



- Inner Join
- Cross Join, Natural Join
- Outer Join
- Left, Right, Full Outer Join



- Joins combine data from one table with data from one or more other tables (or views, or synonyms)
- Tables are "joined" two at a time making a new table containing all possible combinations of rows from the original two tables (sometimes "cartesian product")
- A "join condition" is usually used to limit the combinations of table data to just those rows containing columns that match columns in the other table
- A table may be "joined" to another table, tables, or itself!
- Whenever two or more tables/views/synonyms are listed in a FROM clause a join results
- Join conditions serve the purpose of limiting the number of rows returned by the join



- The original join semantics in SQL include two (or more) table/view/synonym names in a FROM clause, the WHERE clause describes the join condition

```
select distinct nvl(dname, 'No Dept'), count(empno) nbr_emps
  from emp, dept
 where emp.deptno = dept.deptno
       and emp.sal between 2000 and 3000
       and emp.job = 'SALESMAN'
 group by dname;
```



- ISO/ANSI Join syntax has been used for several years in some non-Oracle SQL environments
- Oracle invented the original Outer-join syntax and was slow to accept the new style
- ISO/ANSI Join syntax is supported by third party SQL tools
- The new semantics separate join criteria from other row selection criteria



- Cross Join is the same as when comma-delimited, requiring specification of join conditions in the WHERE clause:

```
select ename,dname  
  from emp cross join dept  
 where emp.deptno = dept.deptno
```



- Natural joins indicate an equi-join automatically using any column names match to join
- Natural joins may also specify ISO/ANSI join types (INNER, LEFT, RIGHT, FULL; discussed later...)
- Additional criteria may be specified using the WHERE clause.

```
select ename,dname  
from emp natural join dept
```




- When join column names are the same, the new syntax now allows the USING clause

```
select dname,ename  
from dept join emp  
using (deptno)
```

- To join using multiple columns, use a comma-delimited list: “(using col1, col2, col3)”




- Traditional Inner Joins match rows of tables
- The older syntax names all tables in comma-delimited form and uses the WHERE clause to specify Join criteria
- Note that in the example below Join criteria is mixed with row selection criteria:

```
select distinct nvl(dname, 'No Dept'),  
               count(empno) nbr_emps  
from emp, dept  
where emp.deptno = dept.deptno  
       and emp.job in ('MANAGER', 'SALESMAN', 'ANALYST')  
group by dname;
```



- Use INNER JOIN (or simply JOIN) between the table(s) involved and specify one-or-more Join criteria with the ON/USING clause
- Correlation (alias) table names may be specified
- The WHERE clause names only non-Join criteria

```
select distinct nvl(dname, 'No Dept'),  
               count(empno) nbr_ems  
from emp join dept  
   on emp.deptno = dept.deptno  
where emp.job in ('MANAGER', 'SALESMAN', 'ANALYST')  
group by dname;
```



Joining More Than Two



```
select distinct nvl(dname, 'No Dept') dept
, count(empno) nbr_emps
, round(avg(grade), 1) avg_paygrade
from      emp
  join    dept
        on emp.deptno = dept.deptno
  join    salgrade
        on emp.sal between losal and hisal
where emp.job in ('MANAGER', 'SALESMAN', 'ANALYST')
group by dname
```

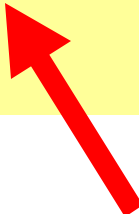


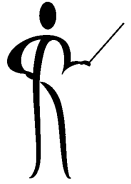
- It is also possible that a user might be interested in rows that DO NOT match rows in the other table(s)
- Finding rows without matches is often referred to as Outer Join (sometimes Anti-Join)



- Oracle invented the first syntax for solving the outer Join issue years ago
- This is the “(+)” notation used on the side of the Join criteria WHERE clause where null rows are to be created to match the other table

```
select distinct nvl(dname, 'No Dept'),  
               count(empno) nbr_emp  
from emp, dept  
where emp.deptno(+) = dept.deptno  
group by dname;
```





- The new ISO/ANSI Join syntax provides three separate capabilities: LEFT, RIGHT, and FULL OUTER JOIN (the word OUTER is redundant and usually omitted)
- With the new syntax, LEFT and RIGHT indicate which side of the join represents the complete set, the opposite side is where null rows will be created



- The example below solves the same problem as the Oracle Outer Join operator example earlier:

```
select distinct nvl(dname, 'No Dept'),  
               count(empno) nbr_emps  
from emp right join dept  
   on emp.deptno = dept.deptno  
group by dname;
```




- To cause SQL to generate rows on both sides of the join required a UNION using the old Oracle Outer Join operator syntax:

```
select nvl(dname, 'No Dept') deptname,  
       count(empno) nbr_emps  
from emp, dept  
where emp.deptno(+) = dept.deptno  
group by dname  
  
union  
  
select nvl(dname, 'No Dept') deptname,  
       count(empno) nbr_emps  
from emp, dept  
where emp.deptno = dept.deptno(+)  
group by dname;
```



- The new ISO/ANSI Outer Join mechanism is simpler to code
- To cause rows to be created on either side of a Join as required to align the two tables use the FULL OUTER JOIN (FULL JOIN) syntax:

```
select distinct nvl(dname, 'No Dept')
deptname, count(empno) nbr_emps
from emp full join dept
on dept.deptno = emp.deptno -- using (deptno)
group by dname;
```



- COALESCE is similar to NVL, but, returns first non-null value:

```
COALESCE(qtr4, qtr3, qtr2, qtr1)
```

- NULLIF returns NULL if the specified value is matched

```
NULLIF(PREFCODE, 'N/A')
```



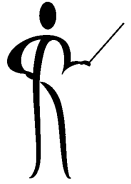
- Oracle recommends the new ISO/ANSI Outer Join
- Queries using the Oracle Outer Join operator “(+)” are subject to rules and restrictions, that do not apply to the ANSI syntax:
 1. Queries may not mix Oracle Outer Join with new ISO/ANSI semantics
 2. If multiple join conditions are present the Oracle Outer Join operator must be specified for every condition
 3. Oracle Outer Join may not apply to an expression
 4. Oracle Outer Join operator may not be combined with another condition using the OR operator
 5. Oracle Outer Join operator may not be used with IN
 6. Oracle Outer Join operator may only be used for one table in a query



- In Oracle 9.0 & 9.2 it appears that Inner Joins behave differently using the comma-delimited syntax vs. when using the new syntax
- In at least some cases, the new syntax creates a different plan and sometimes does not perform as well as the earlier methods – test both
- Right-Left Outer Joins in 9.0 & 9.2 appear to use exactly the same execution plan for queries using either the older Oracle Outer Join operator or the new Outer Join semantics, there should be no performance difference – it never hurts to test!
- Full Join syntax in 9.0 seems the same as the old method, under 9.2 Full Join usually generates a better plan



- In Oracle version 9.0 there seems to be a bug in FULL JOIN when a View is named rather than a Table
(system cannot find view...)
- This problem disappears under Oracle 9.2



- Oracle9.0 out of the box has a **HUGE BUG!**
- ANSI/ISO Join syntax (inner or outer join) allows access to **ANY** table or view in the database!!!!!! (no fooling!!!)
- Patches are available for all platforms except Windows (whoops!)
- Oracle 9.2 fixes this problem (hurrah!)



- Using ISO/ANSI-standard syntax rather than vendor-specific code makes good sense
- The number of programming, code-generation, development, and code review tools recognizing the new syntax is growing
- With Right/Left Outer Joins the choice of the new syntax is made easier since performance does not seem to differ, you should still test
- With Full Outer Joins the choice seems obvious since performance appears to be the same in Oracle 9.0 and seems to improve with Oracle 9.2
- The simpler syntax of the new semantics especially with FULL JOIN is probably worth adopting the new style alone
- In the case of Inner Joins, the performance issues cannot be ignored by many larger installations -- YOUR MILEAGE MAY VARY, there is no substitute for testing



To contact the author:

John King

King Training Resources

6341 South Williams Street

Littleton, CO 80121-2627 USA

1.800.252.0652 - 1.303.798.5727

Email: john@kingtraining.com



Thanks for your attention!

Today's slides and examples are on the web:

<http://www.kingtraining.com>