9

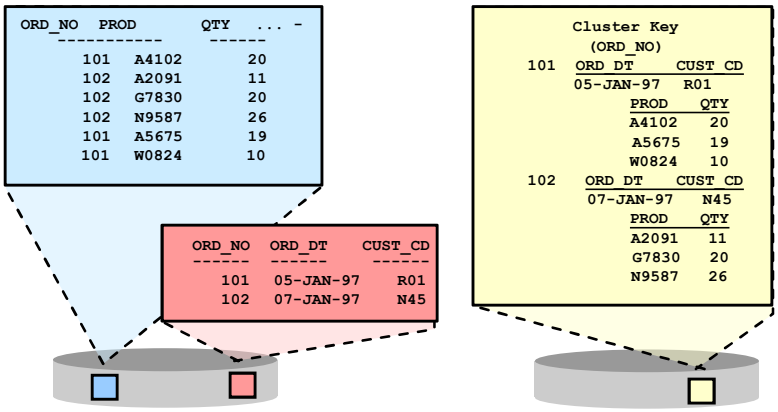
Other Optimizer Operators

Objectives

After completing this lesson, you should be able to:

- Describe SQL operators for:
  - Clusters
  - In-List
  - Sorts
  - Filters
  - Set Operations
- Result cache operators

Clusters



Unclustered `ORDERS` and `ORDER_ITEMS` tables

Clustered `ORDERS` and `ORDER_ITEMS` tables

When Are Clusters Useful?

- Index cluster:
  - Tables are always joined on the same keys.
  - The size of the table is not known.
  - It can be used in any type of search.
- Hash cluster:
  - Tables are always joined on the same keys.
  - Storage for all cluster keys is allocated initially.
  - It can be used in either equality (=) or nonequality (<>) searches.

When Are Clusters Useful?

- Single-table hash cluster:
  - Fastest way to access a large table with an equality search
- Sorted hash cluster:
  - Is used only for equality search
  - Avoid sorts on batch reporting
  - Avoid overhead probe on the branch blocks of an IOT

Cluster Access Path: Examples

1.20700002 seconds

```
select * from bigemp_fact where deptno=10;
```

Autotrace

1.207 seconds

OPERATION	OBJECT_NAME	COST	LAST_CR_BUFFER_GETS
SELECT STATEMENT		1	
TABLE ACCESS HASH	BIGEMP_FACT	1	
Access Predicates			
DEPTNO=10			

1.25800002 seconds

```
select * from emp,dept where emp.deptno=dept.deptno and emp.deptno > 800;
```

Autotrace

1.258 seconds

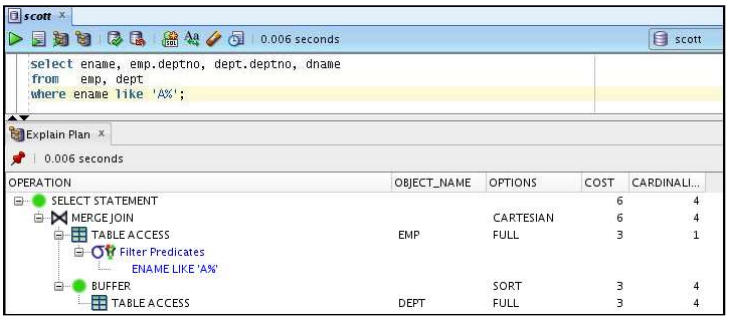
OPERATION	OBJECT_NAME	COST	LAST_CR_BUFFER_GETS
SELECT STATEMENT		11520	
NESTED LOOPS		11520	69
TABLE ACCESS CLUSTER	DEPT	174	60
INDEX RANGE SCAN	EMP_DEPT_IN...	2	2
Access Predicates			
DEPT.DEPTNO>800			
TABLE ACCESS CLUSTER	EMP	57	9
Filter Predicates			
AND			
EMP.DEPTNO>800			
EMP.DEPTNO=DEPT.D			

Sorting Operators

- `SORT` operator:
  - `AGGREGATE`: Retrieves a single row from group function
  - `UNIQUE`: Removes duplicates
  - `JOIN`: Precedes a merge join
  - `GROUP BY`, `ORDER BY`: For these operators
- `HASH` operator:
  - `GROUP BY`: For this operator
  - `UNIQUE`: Equivalent to `SORT UNIQUE`
- If you want ordered results, *always* use `ORDER BY`.

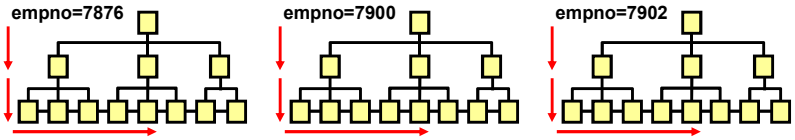
Buffer Sort Operator

```
SELECT ename, emp.deptno, dept.deptno, dname
FROM emp, dept
WHERE ename like 'A%';
```



Inlist Iterator

Every value executed separately




```
SELECT * FROM emp WHERE empno IN (7876, 7900, 7902);
```

scott | 0.007 seconds | scott

```
SELECT * FROM emp WHERE empno IN (7876, 7900, 7902);
```

Explain Plan | 0.007 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALI...
SELECT STATEMENT			2	3
INLIST ITERATOR				
TABLE ACCESS	EMP	BY INDEX RO...	2	3
INDEX	PK_EMP	UNIQUE SCAN	1	3
Access Predicates				
OR				
EMPNO=7876				
EMPNO=7900				
EMPNO=7902				

View Operator

```
create view V as select /*+ NO_MERGE */ DEPTNO, sal from emp ;
select * from V;
```

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALI...
SELECT STATEMENT			3	14
VIEW	V		3	14
TABLE ACCESS	EMP	FULL	3	14

```
select v.*,d.dname from (select DEPTNO, sum(sal) SUM_SAL
from emp group by deptno) v, dept d where v.deptno=d.deptno;
```

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALI...
SELECT STATEMENT			7	3
MERGE JOIN			7	3
TABLE ACCESS	DEPT	BY INDEX ...	2	4
INDEX	PK_DEPT	FULL SCAN	1	4
SORT		JOIN	5	3
Access Predicates				
V.DEPTNO=D.DEPT				
Filter Predicates				
V.DEPTNO=D.DEPT				
VIEW			4	3
HASH		GROUP BY	4	3
TABLE ACCESS	EMP	FULL	3	14

Count Stop Key Operator

```
SELECT count(*)
FROM (SELECT /*+ NO_MERGE */ *
      FROM emp WHERE empno ='1' and rownum < 10);
```

scott | 0.007 seconds | scott

```
select count(*)
from (select /*+ NO_MERGE */ *
      from emp where empno ='1' and rownum < 10);
```

Autotrace | Script Output | Explain Plan

0.007 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALI...
SELECT STATEMENT			0	1
SORT		AGGREGATE		1
VIEW			0	1
COUNT		STOPKEY		
Filter Predicates				
ROWNUM<10				
INDEX	PK_EMP	UNIQUE SCAN	0	1
Access Predicates				
EMPNO=1				

Min/Max and First Row Operators

```
SELECT MIN(quantity_on_hand)
FROM INVENTORIES
WHERE quantity_on_hand < 500;
```

oe_connection | 1.079 seconds | oe_connection

```
SELECT MIN(quantity_on_hand)
FROM INVENTORIES
WHERE quantity_on_hand < 500;
```

Script Output | Explain Plan

1.079 seconds

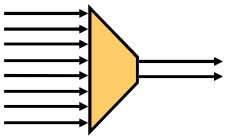
OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALI...
SELECT STATEMENT			2	1
SORT		AGGREGATE		1
FIRST ROW			2	1
INDEX	INV_QTY_INDEX	RANGE SCAN ...	2	1
Access Predicates				
QUANTITY_ON_HAND<500				

Other N-Array Operations

- FILTER
- CONCATENATION
- UNION ALL/UNION
- INTERSECT
- MINUS

FILTER Operations

- Accepts a set of rows
- Eliminates some of them
- Returns the rest



```
SELECT  deptno, sum(sal) SUM_SAL FROM emp
GROUP BY deptno HAVING sum(sal) > 9000;
```

SELECT STATEMENT			4	3
FILTER				
Filter Predicates				
SUM(SAL)>9000				
HASH		GROUP BY	4	3
TABLE ACCESS	EMP	FULL	3	14

1

```
SELECT deptno, dname FROM dept d WHERE NOT EXISTS
(select 1 from emp e where e.deptno=d.deptno);
```

SELECT STATEMENT			6	1
MERGE JOIN		ANTI	6	1
TABLE ACCESS	DEPT	BY INDEX ...	2	4
INDEX	PK_DEPT	FULL SCAN	1	4
SORT		UNIQUE	4	14
Access Predicate:				
E.DEPTNO=D.D				
Filter Predicates				
E.DEPTNO=D.D				
TABLE ACCESS	EMP	FULL	3	14

2

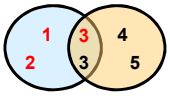
Concatenation Operation

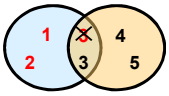
```
SELECT * FROM emp WHERE deptno=1 or sal=2;

-----------------------------------------------------------------------
| Id  | Operation                    | Name     | Rows  | Bytes |
-----------------------------------------------------------------------
|   0 | SELECT STATEMENT             |          |     8 |   696 |
|   1 |  CONCATENATION               |          |       |       |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP      |     4 |   348 |
|   3 |    INDEX RANGE SCAN          | I_SAL    |     2 |       |
|   4 |   TABLE ACCESS BY INDEX ROWID| EMP      |     4 |   348 |
|   5 |    INDEX RANGE SCAN          | I_DEPTNO |     2 |       |
-----------------------------------------------------------------------
        Predicate Information (identified by operation id):
        ---------------------------------------------------
                        3 - access("SAL"=2)
                  4 - filter(LNNVL("SAL"=2))
                    5 - access("DEPTNO"=1)
```

UNION [ALL], INTERSECT, MINUS

UNION
UNION ALL

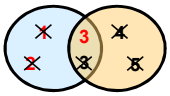
```
SORT UNIQUE
UNION-ALL
INDEX FULL SCAN
INDEX FAST FULL SCAN
```





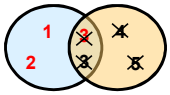
INTERSECT

```
INTERSECTION
SORT UNIQUE NOSORT
INDEX FULL SCAN
SORT UNIQUE
INDEX FAST FULL SCAN
```



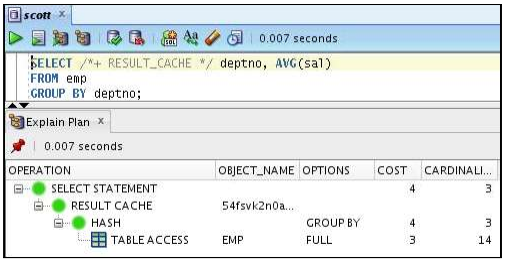
MINUS

```
MINUS
SORT UNIQUE NOSORT
INDEX FULL SCAN
SORT UNIQUE
INDEX FAST FULL SCAN
```



Result Cache Operator

```
SELECT /*+ RESULT_CACHE */ deptno, AVG(sal)
FROM emp
GROUP BY deptno;
```



Quiz

Hash clusters are a better choice than indexed tables or index clusters when a table is queried frequently with equality queries.

a. True
b. False

Quiz

The _____ operator uses a temporary table to store intermediate data.

a. Buffer Sort Operator
b. Inlist
c. Min/Max
d. N-Array

Quiz

The following query uses the _____ operator:

```
SELECT * FROM emp WHERE empno IN (7876, 7900,
7902);
```

a. Buffer Sort Operator
b. Inlist
c. Min/Max
d. N-Array

Quiz

A `FILTER` operation retrieves rows returned by another statement.

a. True
b. False

Summary

In this lesson, you should have learned to:

- Describe SQL operators for:
  - Clusters
  - In-List
  - Sorts
  - Filters
  - Set Operations
- Result cache operators

Practice 9: Overview

This practice covers the following topics:

- Using different access paths for better optimization (case 14 to case 16)
- Using the result cache