Joins

Macneil Fernandes©2005

Joins Overview

- Joins are statements that retrieve data from more than one table.
- A join is characterized by multiple tables in the FROM clause, and the relationship between the tables is defined through the existence of a join condition in the WHERE clause.

Macneil Fernandes©2005

Join Overview

Cartesian Join.

- Consider the Example Query given below

```
SELECT ... FROM Orders, Order_Details;
```

- With no WHERE clause at all, the database has no instructions on how to combine rows from these two large tables, and it does the logically simplest thing: it returns every possible combination of rows from the tables.
- This type of join is know as a *Cartesian Join.*
- The result, every combination of elements from two or more sets, is known as the *Cartesian product*.

Macneil Fernandes©2005

Join Overview

Inner Join.

- Consider the example query as show below:

```
SELECT ... FROM Orders O, Order_Details D WHERE
  O.Order_ID=D.Order_ID;
```

```
Or, shown in the newer-style notation:
```

```
SELECT ... FROM Orders O INNER JOIN Order_Details D ON
  O.Order_ID=D.Order_ID;
```

- The query above gives all the combinations of orders and order details, but discards those combinations that fail to share the same order ID.
- Such a join which only returns the combinations of rows after discarding the rows that fail to satisfy the join condition is referred to as an inner join.

Macneil Fernandes©2005

Join Overview

Outer Join.

- An outer join is easiest to describe in procedural terms:
    - Start with rows from a driving table
    - Find matching rows, where possible, from an outer-joined table,
    - Create an artificial all-nulls matching row from the outer-joined table whenever the database finds no physical matching row.
- There are three forms of the join operation, namely
    - Left outer join
    - Right outer join
    - Full outer join.

Macneil Fernandes©2005

Join Overview

- The syntax of the outer join is a bit different from that of the inner join, because it includes a special operator called the outer join operator. The outer join operator is a plus sign enclosed in parentheses, i.e., (+). This operator is used in the join condition in the WHERE clause following a field name from the table that you wish to be considered the optional table.
- In the suppliers and parts example given below, the PART table doesn't have information for one supplier. Therefore, we will simply add a (+) operator to the join condition on the side of the PART table. The query and the result set look as follows:

```
SELECT S.SUPPLIER_ID, S.NAME SUPPLIER_NAME,
  P.PART_NBR, P.NAME PART_NAME FROM SUPPLIER S, PART
  P WHERE S.SUPPLIER_ID = P.SUPPLIER_ID (+);
```

- Note the (+) operator following P.SUPPLIER_ID. That makes PART the optional table in this join. If a supplier does not currently supply any parts, Oracle will fabricate a PART record with all NULLs for that supplier. Thus, the query results can include all suppliers, regardless of whether they currently supply parts.

Macneil Fernandes©2005

Join Overview

- The outer join operator (+) can appear on either the left or the right side of the join condition. However, make sure you apply this operator to the appropriate table in the context of your query. For example, it makes no difference to the result if you switch the two sides of the equality operator in the previous example:

```
SELECT S.SUPPLIER_ID, S.NAME SUPPLIER_NAME,
  P.PART_NBR, P.NAME PART_NAME FROM SUPPLIER S, PART P
  WHERE P.SUPPLIER_ID (+) = S.SUPPLIER_ID;
```

Macneil Fernandes©2005

Restrictions on Outer Joins:

- The outer join operator can appear on only one side of an expression in the join condition.

```
SELECT .............. FROM SUPPLIER S, PART P
WHERE S.SUPPLIER_ID (+) = P.SUPPLIER_ID (+);
```

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

```
SELECT .............. FROM EMPLOYEE E, JOB J, DEPARTMENT D
WHERE E.JOB_ID (+) = J.JOB_ID AND E.DEPT_ID (+) =
  D.DEPT_ID;
```

- An outer join condition containing the (+) operator may not use the IN operator

```
SELECT .............. FROM EMPLOYEE E, JOB J
WHERE E.JOB_ID (+) IN (668, 670, 667);
```

- An outer join condition containing the OR operator may not be combined with another condition using the OR operator.

```
SELECT ..............  FROM EMPLOYEE E, DEPARTMENT D
WHERE E.DEPT_ID = D.DEPT_ID (+) OR D.DEPT_ID = 10;
```

- A condition containing the (+) operator may not involve a subquery

```
SELECT .............. FROM EMPLOYEE E
WHERE E.DEPT_ID (+) = (SELECT DEPT_ID FROM DEPARTMENT
  WHERE NAME = 'ACCOUNTING');
```

Full Outer Joins

Full Outer Joins

- The full outer join does the operation of both the right outer join and the left outer join. i.e. It pads the rows from the left relation that did not match any from the right relation, as well as pad the rows got from the right relation that did not match any from the left relation.
- Consider a department and a location table. If you want to include the departments without a location as well as the locations without a department, you will probably try to use a two-sided outer join, correctly termed a full outer join, as:

```
SELECT ………………… FROM DEPARTMENT D, LOCATION L
WHERE D.LOCATION_ID (+) = L.LOCATION_ID (+);
```

- However Oracle Doesn’t allow a 2-sided outer join. A UNION of two SELECT statements is a work around for this problem.
- The above query can be rewritten as :

```
SELECT ………………… FROM DEPARTMENT D, LOCATION L
WHERE D.LOCATION_ID (+) = L.LOCATION_ID
UNION
SELECT ………………… FROM DEPARTMENT D, LOCATION L
WHERE D.LOCATION_ID = L.LOCATION_ID (+) ;
```

Macneil Fernandes©2005

Join Execution Methods

- The join types determine which results a query requires but do not specify how a database should execute those queries.
- Join Method To join each pair of row sources, Oracle must perform one of these operations:
  - Nested loops (NL) join (Nested Loops Outer Join)
  - Sort merge join (Sort merge Outer Join)
  - Hash join (Hash Joins Outer Join)
  - Cartesian Joins
  - Full Outer Joins

Nested-loops Joins

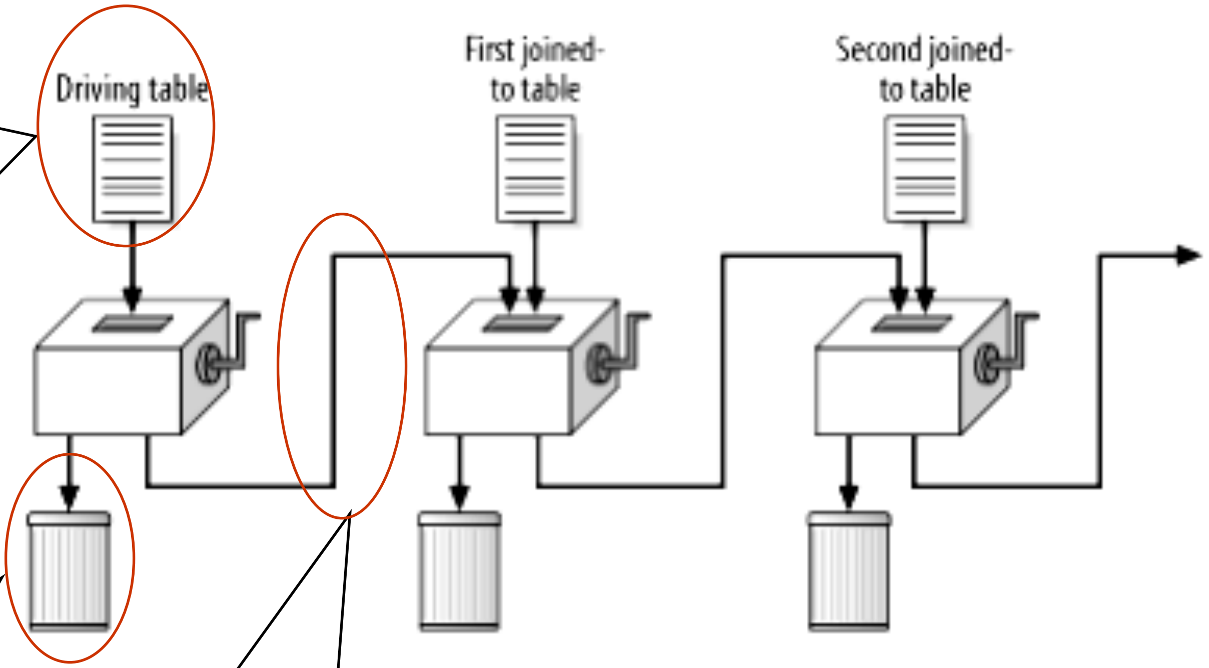
- Nested loop (NL) joins are useful for joining small subsets of data and if the join condition is an efficient way of accessing the second table.
- For nested loop joins:
  1. The optimizer determines a driving table (the outer loop).
  2. The other table is designated the inner table.
  3. The database implements this execution plan as a nested series of loops—the outermost loop reads rows off the driving table, the next loop finds matching rows in the first joined table, and so on.

```
NESTED LOOP
<Outer Loop>
<Inner Loop>
<…………….>
```

Nested-loops Joins

- The simplest method of performing an efficient join between two or more tables is the nested-loops join, illustrated below

The query execution begins with what amounts to a single-table query of the driving table (the first table the database reads), using only the conditions that refer solely to that table.



From the Driving Tables rows that don't satisfy the single-table conditions are separated from rows that satisfy the single-table query conditions.

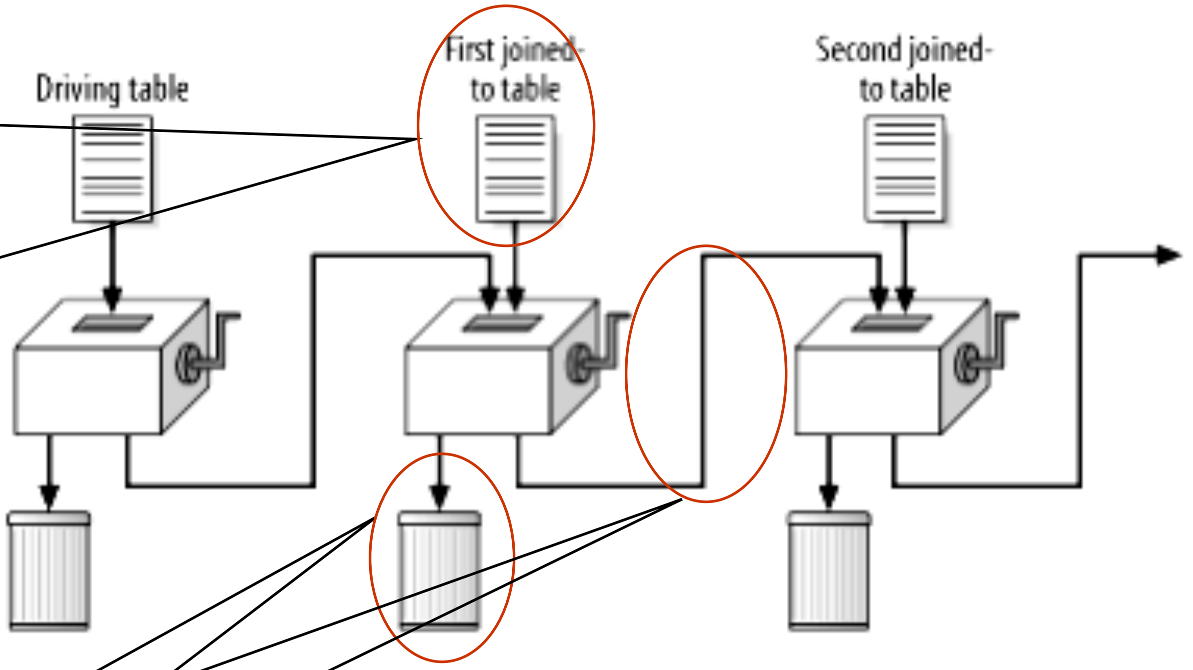
Since the query is a join, the database does not stop here. Instead, it passes the result rows from that first box to the next box.

Macneil Fernandes©2005

Nested-loops Joins

- The simplest method of performing an efficient join between two or more tables is the nested-loops join, illustrated below

The job of the second box is to take rows from the first box, one at a time, find matching rows from the first joined-to table, then again separate the rows that don't satisfy the condition from the rows that satisfy the conditions.



The rows that don't meet the query conditions on the tables touched so far are discarded while the rows that meet these conditions are passed on.

Nested-loops Outer Joins

- When the join is an inner join, the database discards rows from the earlier step that fail to find matches in the joined-to table.
- When the join is an outer join, the database fills in joined-to table values with nulls when it fails to find a match, retaining all rows from the earlier step.
- This process continues with the rest of the tables in the same way until the query is finished, leaving a fully joined result that satisfies all query conditions and joins.

Macneil Fernandes©2005

Hash Joins

- Sometimes, the database accesses the joined tables independently and then match rows where it can and discard unmatchable rows.
- The database has two ways to do this: *hash joins*, and *sort-merge joins*
- The optimizer uses the smaller of the two tables/data sources to build a hash table on the join key in memory.
- It then scans the larger table, probing the hash table to find the joined rows.
- You cannot use hash joins unless there is an equality condition.

Macneil Fernandes©2005

Hash Joins

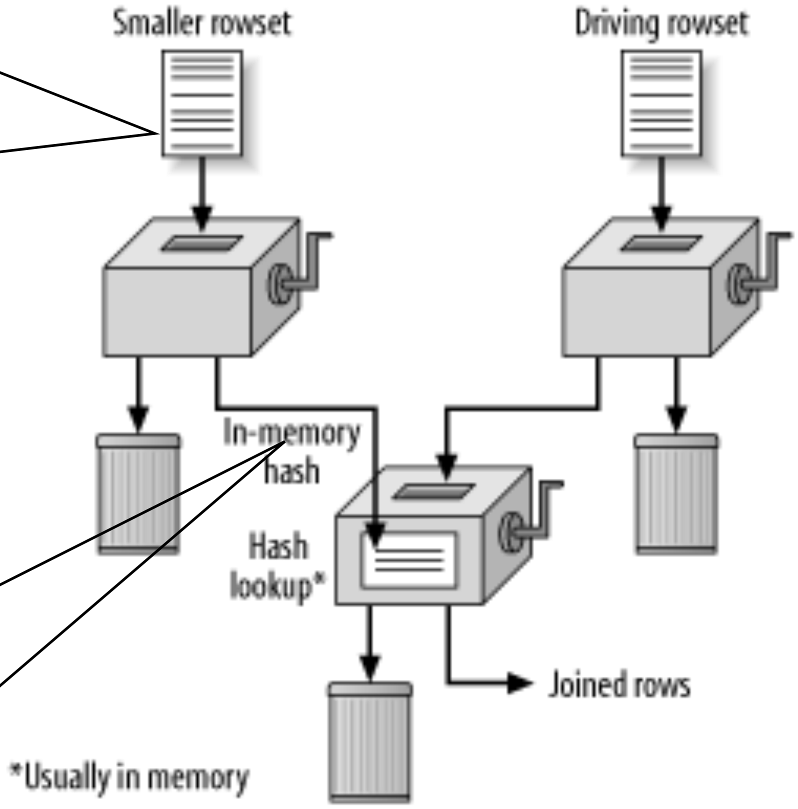
- The figure below illustrates a hash join.

•As shown, each of the top two boxes with cranks acts like an independently optimized single-table query.

•Based on table and index statistics, the cost-based optimizer estimates which of these two independent tables will return fewer rows after discarding filtered rows.

•It chooses to hash the complete result from that single-table query.

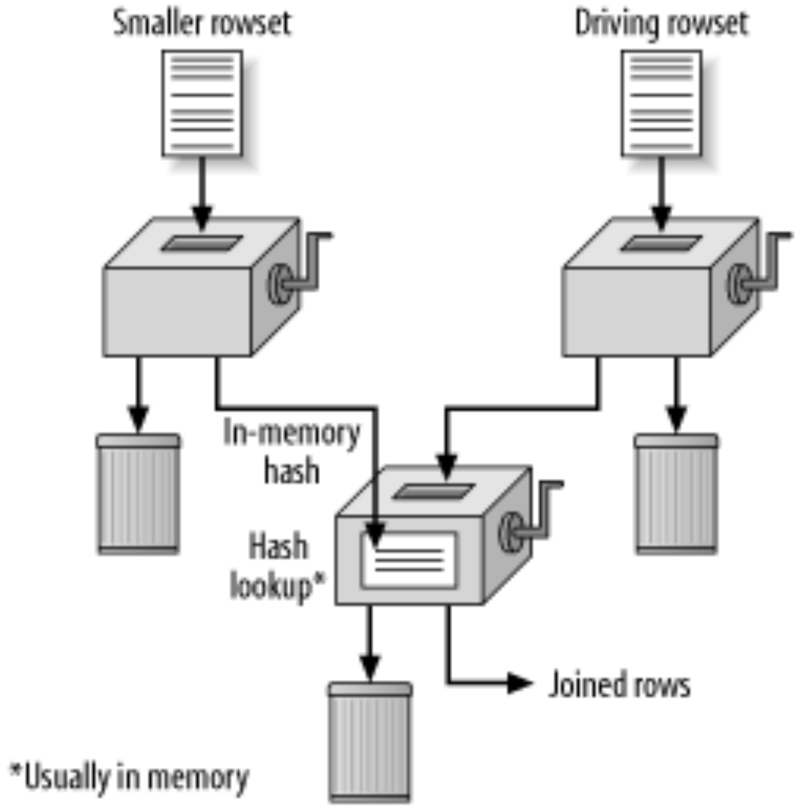
•In other words, it performs some randomizing mathematical function on the join key and uses the result of that function to assign each row to a hash bucket.



Macneil Fernandes©2005

Hash Joins

- It then executes the larger query in returning the driving rowset.
- As each row exits this step, the database executes the same hash function in its join key and uses the hash-function result to go directly to the corresponding hash bucket for the other rowset.
- When it reaches the right hash bucket, the database searches the tiny list of rows in that bucket to find matches.



Hash Joins Outer Joins

- This operation is used for outer joins.
- The outer table (whose rows are being preserved) is used to build the hash table, and the inner table is used to probe the hash table.
- The optimizer uses hash joins for processing an outer join if the data volume is high enough to make hash join method efficient or if it is not possible to drive from the outer table to inner table.

Macneil Fernandes©2005

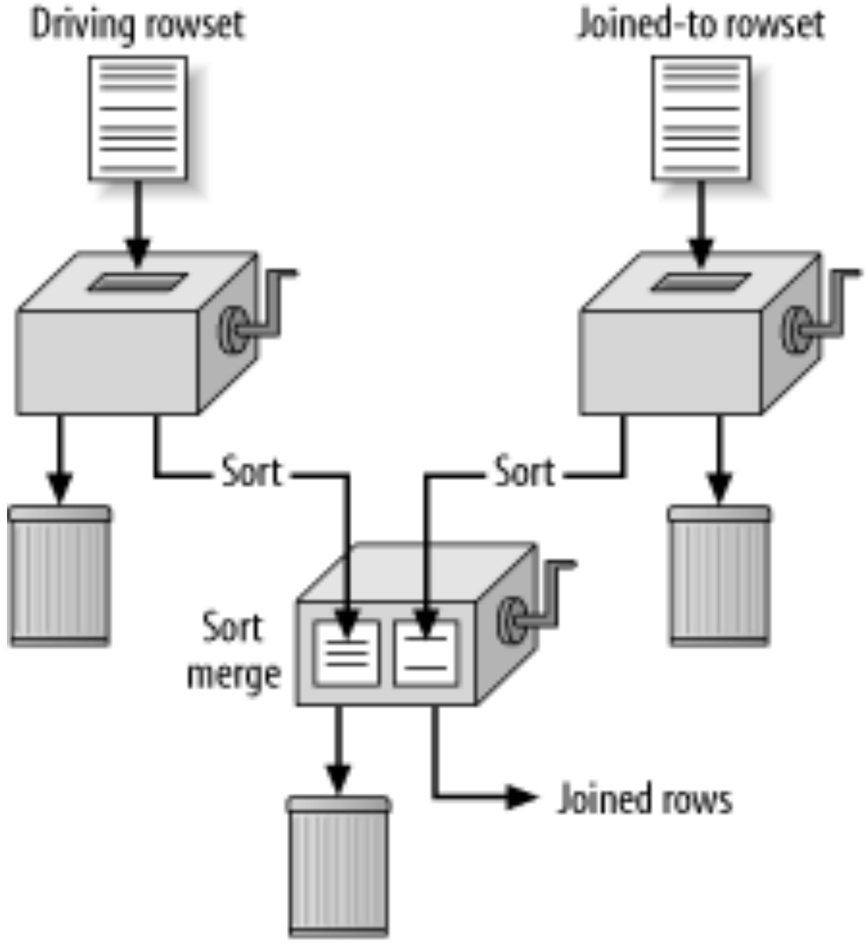
Sort Merge Joins

- Sort merge joins can be used to join rows from two independent sources.
- Sort merge joins can perform better than hash joins if the row sources are sorted already, and if a sort operation does not have to be done.
- Sort merge joins are useful when the join condition between two tables is an inequality condition (but not a nonequality) like <, <=, >, or >=.
- In a merge join, there is no concept of a driving table.
    - Both the inputs are sorted on the join key - sort join operation
    - The sorted lists are merged together - merge join operation

Macneil Fernandes©2005

Sort Merge Joins

- The sort-merge join, shown below reads the two tables independently, but, instead of matching rows by hashing, it presorts both rowsets on the join key and merges the sorted lists.



•In the simplest implementation, you can imagine listing the rowsets side-by-side, sorted by the join keys.

•The database alternately walks down the two lists, in a single pass down each list, comparing top rows, discarding rows that fall earlier in the sort order than the top of the other list, and returning matching rows.

Joins Execution

How the CBO Chooses Execution Plans for Joins

- With the CBO, the optimizer generates a set of execution plans based on the possible join orders, join methods, and available access paths.
- The optimizer then estimates the cost of each plan and chooses the one with the lowest cost.

How CBO chooses Execution Plans for Joins

The optimizer estimates costs in these ways:

- The cost of a nested loops operation is based on the cost of reading each selected row of the outer table and each of its matching rows of the inner table into memory. The optimizer estimates these costs using the statistics in the data dictionary.
- The cost of a sort merge join is based largely on the cost of reading all the sources into memory and sorting them.
- The optimizer also considers other factors when determining the cost of each operation. For example:
  - A smaller sort area size is likely to increase the cost for a sort merge join because sorting takes more CPU time and I/O in a smaller sort area. Sort area size is specified by the initialization parameter SORT_AREA_SIZE.
  - A larger multiblock read count is likely to decrease the cost for a sort merge join in relation to a nested loops join. If a large number of sequential blocks can be read from disk in a single I/O, then an index on the inner table for the nested loops join is less likely to improve performance over a full table scan. The multiblock read count is specified by the initialization parameter DB_FILE_MULTIBLOCK_READ_COUNT.

Macneil Fernandes©2005

Anti-Joins

How the CBO Executes Anti-Joins

- An *anti-join* returns rows from the left side of the predicate for which there is no corresponding row on the right side of the predicate.
- That is, it returns rows that fail to match (NOT IN) the subquery on the right side.
- For example, an anti-join can select a list of employees who are not in a particular set of departments:

```
SELECT * FROM emp
WHERE deptno NOT IN
(SELECT deptno FROM dept
WHERE loc = ’HEADQUARTERS’);
```

- The optimizer uses a nested-loops algorithm for NOT IN subqueries by default, unless the MERGE_AJ, HASH_AJ, or NL_AJ hint is used which allow the transformation of the NOT IN uncorrelated subquery into a sort-merge or hash anti-join.