Joins

Mirunalini.P

SSNCE

March 22, 2022

Session Objective

- To learn about joins
- To learn different join functions

Table of Contents

Cartesian product

A Cartesian product is formed when:

- A join condition is omitted
- A join condition is invalid
- All rows in the first table are joined to all rows in the second table
- To avoid a Cartesian product, include a valid join condition in a WHERE clause.

Cartesian product

Ex:Employees

Emp_id	*Last_name*	*Dept_id*
100	Higgins	10
101	John	20
102	Ramesh	20

Departments

Dept_id	*D_name*
10	Administration
20	Marketing

Cartesian product

Employees X Departments >(Cross product – without any JOIN condition)

Emp_id	*Last_name*	*Dept_id*	*Dept_id*	*D_name*
100	Higgins	10	10	Administration
100	Higgins	**10**	**20**	Marketing
101	John	**20**	**10**	Administration
101	John	20	20	Marketing
102	Ramesh	**20**	**10**	Administration
102	Ramesh	20	20	Marketing

Joins

- Write the join condition in the **WHERE clause**
- **Prefix the column name** with the table name when the same column name appears in more than one table.
- Rows in one table Joined to rows in another table according to **common values** existing in corresponding columns (usually primary and foreign key columns)
- To join n tables together, need minimum of **n-1 join conditions**

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column1 = table2.column2 ;
```

Joins - Oracle 8i and Prior

- Equijoin (Simple join, Inner join) (=)
- Nonequijoin(other than =)
- Outer join (+)
- Self join

Tables

Emplovees

Emp_id	*Last_name*	*Dept_id*
100	Higgins	10
101	John	20
102	Ramesh	30

Departments

Dept_id	*D_name*	*Location_id*
10	Administration	1400
20	Marketing	1500
30	Accounting	1600
40	Shipping	1700

Locations

Location_id	*City*
1400	Toronto
1500	Oxford
1600	New Jersy
1700	New York

EQUI – JOIN

- Joins based on the = operator
- Equi joins removes null values.
- Only matching tuples will appear in the result.

EQUI – JOIN

1. Display the details of employees along with their department name.

```
SELECT e.lastname, d.d_name, d.dept_id
FROM employees e, departments d
WHERE e.dept_id = d.dept_id
```

2. Display the location name of the dept where the employees works

```
SELECT e.lastname, d.d_name, l.city
FROM employees e, departments d, locations l
WHERE e.dept_id = d.dept_id
AND d.location_id = l.location_id ;
```

EQUI – JOIN

```
SELECT * FROM employees e, departments d, WHERE e.dept_id =
d.dept_id;
```

Emp_id	Last_name	Dept_id	Dept_id	Dept_name	location
100	Higgins	10	10	admin	1400
101	John	20	20	Marketing	1500
102	Ramesh	30	30	Accounting	1600

Tables

Employees

Last_name	*Salary*
Higgins	4600
John	2500
Ramesh	9400

Job_Grades

Grade	*Lowest_sal*	*Highest_sal*
A	1000	2999
B	3000	5999
C	6000	10000

NON-EQUIJOINS

Join condition containing something other than an equality operator (=)

Retrieve the grade of the employees

```
SELECT e.last_name, e.salary, j.grade
FROM employees e, job_grades j
WHERE e.salary BETWEEN
j.lowest_sal AND j.highest_sal
```

OUTER JOINS

- If a row does not satisfy a join condition, that row will not appear in the result
- Use an outer join to see the rows that do not meet the join condition
- The outer join operator is the plus sign (+) and it is placed on the "side" of the join that is deficient in information (no matching rows).
- Append NULL values for the unmatched tuples.

Tables

Emplovees

Emp_id	*Last_name*	*Dept_id*
100	Higgins	10
101	John	20
102	Ramesh	30

Departments

Dept_id	*D_name*	*Location_id*
10	Administration	1400
20	Marketing	1500
30	Accounting	1600
40	Shipping	1700

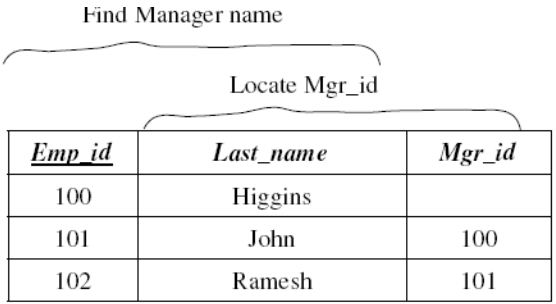
OUTER JOINS

```
SELECT e.last_name, e.dept_id, d.d_name FROM employees e,
departments d WHERE e.dept_id (+) = d.dept_id
```

Last_name	*Dept_id*	*D_name*
Higgins	10	Administration
John	20	Marketing
Ramesh	30	Accounting
		Shipping

SELF JOINS

Joining a table to itself is self join - searching the same table twice



Emp_id	*Last_name*	*Mgr_id*
100	Higgins	
101	John	100
102	Ramesh	101

List the name of each employee's manager

```
SELECT worker.last_name emp_name,
manager.last_name mgr_name
FROM employees worker, employees manager
WHERE worker.mgr_id = manager.emp_id ;
```

JOIN TYPES

- **CROSS JOIN :** Returns a Cartesian product
- **NATURAL JOIN :**Joins two tables based on the same column name
- **USING (column_name) :** Performs an equijoin based on the column name
- **ON (condition) :** Performs an equijoin based on the condition in the ON clause
- **LEFT | RIGHT |FULL OUTER** JOIN table2 - Retrieve both matching and unmatching tuples

CROSS JOIN

Same as a Cartesian product between the two tables

EMPLOYEE

Emp_id	*Last_name*	*Dept_id*
100	Higgins	10
101	John	20

Departments

Dept_id	*D_name*	*Location_id*
10	Administration	1400
20	Marketing	1500
30	Accounting	1600

CROSS JOIN

```
SELECT last_name, d_name
FROM employees CROSS JOIN departments ;
```

Last_name	D_name
Higgins	Administration
John	Administration
Higgins	Marketing
John	Marketing
Higgins	Accounting
John	Accounting

NATURAL JOIN

- NATURAL JOIN clause is based on all columns in the two tables that have the same name
- Specified on the columns which have same names and data types in both tables
- Same name, but different data type ¿ an error

Tables

Departments

Dept_id	*D_name*	*Location_id*
10	Administration	1400
20	Marketing	1500
30	Accounting	1600
40	Shipping	1700

Locations

Location_id	*City*
1400	Toronto
1500	Oxford
1600	New Jersy
1700	New York

NATURAL JOIN

```
SELECT dept_id, d_name, location_id, city
FROM departments NATURAL JOIN locations ;
```

Dept_id	D_name	Location_id	city
10	Administration	1400	Toronto
20	Marketing	1500	Oxford
30	Accounting	1600	New Jersy
40	Shipping	1700	New York

JOINS with the USING clause

- JOINS with the USING clause
- Natural Join uses all columns with same names and data types to join the tables
- USING – Used to specify only those columns that should be used for joins
- Do not use a table name or alias in the referenced columns

Departments

Dept_id	*D_name*	*Location_id*
10	Administration	1400
20	Marketing	1500
30	Accounting	1600
40	Shipping	1700

Locations

Location_id	*City*
1400	Toronto
1500	Oxford
1600	New Jersy
1700	New York

JOINS with the USING clause

```
SELECT * FROM departments JOIN locations
USING (location_id) ;
```

Dept_id	D_name	Location_id	city
10	Administration	1400	Toronto
20	Marketing	1500	Oxford
30	Accounting	1600	New Jersy
40	Shipping	1700	New York

JOINS with the ON clause

ON clause used to specify a join condition or to join columns that have different names

```
SELECT * FROM departments d JOIN locations l
ON (d.location_id = l.location_id) ;
```

Dept_id	D_name	Location_id	Location_id	city
10	Administration	1400	1400	Toronto
20	Marketing	1500	1500	Oxford
30	Accounting	1600	1600	New Jersy
40	Shipping	1700	1700	New York

Columns with same values repeated

Tables - Outer Join

Employees

Emp_id	*Last_name*	*Dept_id*
100	Higgins	10
101	John	20
102	George	

Departments

Dept_id	*D_name*	*Location_id*
10	Administration	1400
20	Marketing	1500
30	Accounting	1600

LEFT OUTER JOIN

Join that returns the results of the matching tuples as well as the unmatched tuples in left relation

```
SELECT e.last_name, e.dept_id, d.d_name
FROM employees e
LEFT OUTER JOIN departments d
ON ( e.dept_id = d.dept_id ) ;
```

Last_name	Dept_id	D_name
Higgins	10	Administration
John	20	Marketing
George		

RIGHT OUTER JOIN

Join that returns the results of the matching tuples as well as the unmatched tuples in right relation

```
SELECT e.last_name, e.dept_id, d.d_name
FROM employees e
RIGHT OUTER JOIN departments d
ON ( e.dept_id = d.dept_id ) ;
```

Last_name	Dept_id	D_name
Higgins	10	Administration
John	20	Marketing
		Accounting

FULL OUTER JOIN

Join that returns the results of the matching tuples as well as the unmatched tuples for both left and right relation

```
SELECT e.last_name, e.dept_id, d.d_name
FROM employees e
FULL OUTER JOIN departments d
ON ( e.dept_id = d.dept_id ) ;
```

Last_name	**Dept_id**	**D_name**
Higgins	10	Administration
John	20	Marketing
George		
		Accounting

Reference

- Fundamentals of Database systems 7^{th} Edition by Ramez Elmasri.
- Oracle 9i SQL