Persistent

Nugget 6: Combining Data

Persistent University

Key Learning Points

1. **Set Operators:**
2. **Joins**

Persistent

Sample Data

Table Name : Employee

EmployeeId	FirseName	LastName	Email	PhoneNumber	HireDate	JobId	Salary	CommitionPct	ManagerId	DepartmentId
1	John	Demn	JohnD@yahoo.com	9898780979	1/10/2001	IT_PROF	70000	0.5	NULL	10
2	Ken	Dale	kendaleD@gmail.com	7877787655	4/1/2001	SALES_HEAD	50000	NULL	NULL	10
3	James	Walton	JW@yahoo.com	5787887888	1/1/2001	IT_REP	30000	0.2	1	20
4	robin	sngal	robin@gmail.com	4990988839	5/1/2001	SALES_REP	40000	0.3	2	20
5	ajay	ghosala	ghosala@hotmail.com	9809888898	6/10/2002	SALES_REP	30000	0.4	2	20
6	John	Reddies	John@gmail.com	6878900989	6/10/2003	M_per	50000	NULL	NULL	NULL

Table Name : Department

DEPARTMENTID	DEPARTMENTNAME	MANAGERID	LOCATIONID
10	Sales	1	1
20	IT	2	2
30	Marketing	(null)	1

Table Name : Location

LOCATIONID	CITY
1	Pune
2	Mumbai

Table Name : Job_History

EMPLOYEEID	START_DATE	END_DATE	JOBID	DEPARTMENTID
1	10-Jan-01	31-Dec-01	IT_PROF	10
1	1-Jan-02	31-Aug-05	IT_REP	10
2	1-Apr-01	31-Jan-02	SALES_REP	20
2	1-Feb-02	31-Jan-05	SALES_PERS	10

Set operators

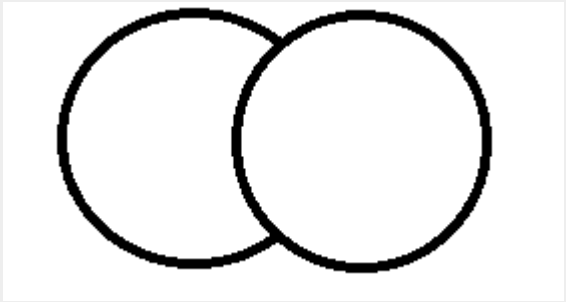
- SET Operators combine result of two or more queries into one result. Queries containing SET operators are called as compound queries.
- There are 4 SET Operators.
  - UNION : All distinct rows selected by either query.
  - UNION ALL : All rows selected by either query; including duplicates.
  - INTERSECT : All distinct rows selected by both the queries.
  - MINUS : All distinct rows selected by first SELECT statement and not selected by 2nd query.

- All SET operators have equal precedence, If SQL statement contains multiple SET operators, they will be evaluated from left to right, if no parenthesis explicitly specifies any other order.

UNION operator

- UNION Operator returns results from both the queries after eliminating the duplicates.

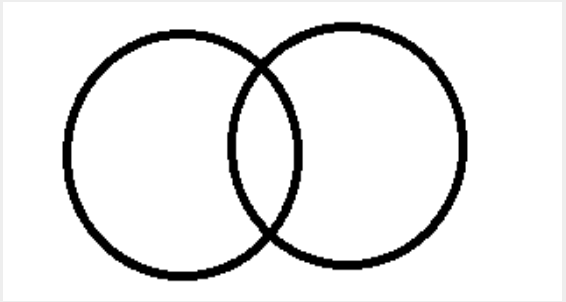
- Guidelines:
  - The number of columns and data type of columns being selected must be identical in all the SELECT statements used in the query. The column names need not be identical.
  - UNION operates over all the columns being selected.
  - By default, output is sorted in ascending order of first column of the SELECT clause.



UNION ALL operator

- UNION ALL Operator returns results from both the queries, including all the duplicates.

- Guidelines:
  - The number of columns and data type of columns being selected must be identical in all the SELECT statements used in the query. The column names need not be identical.
  - UNION ALL operates over all the columns being selected.
  - Duplicate values are not eliminated.
  - Output is not sorted by default.
  - DISTINCT keyword can not be used.



Examples of UNION and UNION ALL operator

- Requirement: Display current and previous job details of all employees . Display each employee/job linking only once.

```
Select employeeid, jobid From employee

UNION

Select employeeid, jobid From job_history;
```

EMPLOYEEID	JOBID
1	**IT_PROF**
1	IT_REP
2	SALES_HEAD
2	SALES_PERS
2	SALES_REP
3	IT_REP
4	SALES_REP
5	SALES_REP
6	M_per

→ Duplicate rows are displayed only once
→ Data is sorted by default

- Requirement: Display current and previous job details of all employees .

```
Select employeeid, jobid From employee

UNION all

Select employeeid, jobid From job_history;
```

EMPLOYEEID	JOBID
1	**IT_PROF**
2	SALES_HEAD
6	M_per
3	IT_REP
4	SALES_REP
5	SALES_REP
1	**IT_PROF**
1	IT_REP
2	SALES_REP
2	SALES_PERS

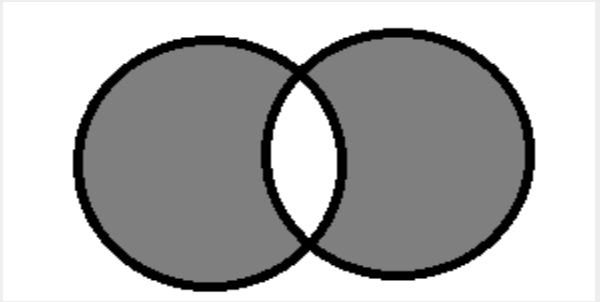
→ Duplicate rows are not eliminated.
→ Data is not sorted. We need to use Order By to sort the data.

Persistent

INTERSECT Operator

- INTERSECT Operator returns all the rows common to both the queries.

- Guidelines:
  - The number of columns and data type of columns being selected must be identical in all the SELECT statements used in the query. The column names need not be identical.
  - Reversing the order of intersected tables does not alter the result.

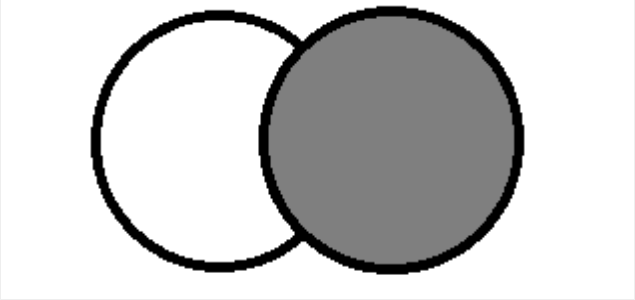


MINUS Operator

- MINUS Operator returns rows returned by the first query which are not present in rows returned by the second query

  (First SELECT statement minus second SELECT statement).

- **Guidelines:**
  - The number of columns and data type of columns being selected must be identical in all the SELECT statements used in the query. The column names need not be identical.
  - All the columns in the WHERE clause must be in the SELECT clause for the MINUS operator to work.



Examples of INTERSECT and MINUS operator

- Requirement: Display employee id and job id of all employees which currently have a job title which they already hold before in the same company.

```
Select employeeid,jobid from employee

INTERSECT

Select employeeid,jobid from job_history;
```

EMPLOYEEID	JOBID
1	IT_PROF

- Requirement: Display employee ids who have not changed their job even once.

```
Select employeeid From employee

MINUS

Select employeeid From job_history;
```

EMPLOYEEID
3
4
5
6

SET Operator Guidelines

- The expressions in SELECT statement must match in number and data type.
- Parenthesis can be used to alter the sequence of execution.
- Column names from First query appears in the result.
- The ORDER BY Clause:
  - Can appear at the very end of the statement.
  - Will accept column name , aliases from first SELECT statement, or the positional notation.
- SET Operators can be used in Subqueries.
- The output is sorted in ascending order by default; except in UNION ALL operator.
- Duplicate values are eliminated, except in case of UNION ALL.
- The set operators are not valid on columns of type BLOB, CLOB, BFILE, VARRAY, or nested table.
- The UNION, INTERSECT, and MINUS operators are not valid on LONG columns.

Joining tables

- When data from more than one table in database is required, a JOIN condition is used. Rows in one table can be joined to rows in another table according to common values existing in corresponding column, usually, primary and foreign key columns.
- Syntax:

```
SELECT table1.column, table2.column
From table1, table2
Where table1.column1=table2.column2;    -- WHERE clause contains the condition that joins the tables together.
```

- Guidelines:
  - When writing the SELECT statement that joins the tables, precede the column name with table name for clarity and enhance database access.
  - If same column name appears in more than one table, column name must be prefixed with table name.
  - To join n tables, you need minimum of n-1 join conditions. This rule may not apply if your table has composite primary key.

Table Aliases

- Qualifying column names with table names can be very time consuming; particularly if table names are lengthy. User can use table aliases instead of table names.
- As column alias gives a column another name, table alias gives table another name.
- Table aliases helps to keep SQL code smaller and thereby saving the memory.
- Table aliases are specified as : From Employee Emp, department Dept
- Guidelines:
  - Can be up to 30 characters in length; but shorter is better.
  - If the table alias is used for a particular table name in FROM clause then that table alias must be substituted for the table name throughout the SELECT statement.
  - Table aliases should be meaningful.
  - Table alias is valid only for the current SELECT statement.

Inner join

- Consider our example database.

To display the employee's department name, we need to compare the value in department id column of EMPLOYEE table with department id column in DEPARTMENT table. This relationship between EMPLOYEE and DEPARTMENT table is an Equi Join or Inner Join.

- User can add more search condition using the WHERE clause in such Inner Join query.
- Example:

```
Select employee.employeeid, employee.lastname, employee.departmentid, department.departmentname

From Employee, Department

Where employee.departmentid=department.departmentid;
```

EMPLOYEEID	LASTNAME	DEPARTMENTID	DEPARTMENTNAME
1	Demn	10	Sales
2	Dale	20	IT
3	Walton	10	Sales
4	sngal	20	IT
5	ghosala	20	IT

Inner join

- Use of aliases and filter in Inner Join

```
Select E.employeeid,E.lastname, E.departmentid, D.departmentname
From employee E, department D
Where E.departmentid=D.departmentid
and E.lastname like 'D%'
```

EMPLOYEEID	LASTNAME	DEPARTMENTID	DEPARTMENTNAME
1	Demn	10	Sales
2	Dale	20	IT

- Joining more than 2 tables.

```
Select E.employeeid, E.lastname, E.departmentid, D.departmentname, L.city
From employee E, department D, location L
Where E.departmentid=D.departmentid
and D.locationid= L.locationid order by employeeid;
```

EMPLOYEEID	LASTNAME	DEPARTMENTID	DEPARTMENTNAME	CITY
1	Demn	10	Sales	Pune
2	Dale	20	IT	Mumbai
3	Walton	10	Sales	Pune
4	sngal	20	IT	Mumbai
5	ghosala	20	IT	Mumbai

Cartesian Product

- A Cartesian product is formed when:
  - A Join condition is omitted
  - A Join condition is invalid.
- In this case, all rows in the first table are joined with all the rows in the second table.
- To avoid Cartesian product, always use a valid join condition in WHERE clause, unless user has specific need to combine all rows from all the tables.
- Example:

```
Select E.lastname, D.departmentname

from employee E  , department D;
```

As Employee table contains 6 rows and Department table contains 3 rows, the output will contain total 18 rows.

In this case, every row in Employee table will join with all the rows in Department table.

Natural join (SQL:1999 compliant Syntax)

- Oracle 9i and further versions also supports SQL:1999 JOIN syntax.
- In Oracle releases prior to 9i, It was not possible to join without explicitly specifying the columns in the corresponding tables. In 9i, it is possible to let the join be performed automatically based on columns in 2 tables which have matching datatype and name, using the keyword NATURAL JOIN. This clause is used to perform INNER Join.
- The Natural Join is based on the columns that have the same name and same data type. Columns that have same name in both the tables must be used without any qualifiers(table name or alias).
- If the column having same name and different data type, error is returned.
- It selects rows from the 2 tables that have equal values in all the matched columns.
- Example:

```
Select departmentId, departmentName, city

from Department NATURAL JOIN Location;
```

DEPARTMENTID	DEPARTMENTNAME	CITY
10	Sales	Pune
20	IT	Mumbai
30	Marketing	Pune

Join with ‘ON clause’ (SQL:1999 compliant Syntax)

- The Join condition for the natural join is basically an equi join of all the columns with the same name. To Specify arbitrary condition or to specify columns to join, the ON clause is used.
- This clause lets user specify the Join condition separate from any search or filter condition in WHERE clause. The ON clause makes code easy to understand.
- Example:

```
SELECT e.employeeid, d.departmentId, d.departmentName, l.city
FROM Employee e
JOIN Department d
ON e.departmentid= d.departmentid
JOIN Location l
ON d.locationid= l.locationid
```

EMPLOYEEID	DEPARTMENTID	DEPARTMENTNAME	CITY
3	10	Sales	Pune
1	10	Sales	Pune
5	20	IT	Mumbai
4	20	IT	Mumbai
2	20	IT	Mumbai

Self Join

- Joining a **Table to itself**: Sometimes User needs to join table to itself.

  For example: To find employee's manager name, user need to join Employee table to itself. It is called as Self Join.

- To Find Robin's manager:
  - Find details of Robin in Employee table
  - Find the manager number of Robin in employee table. Robin's manager id is 2.
  - Find the name of the employee with employee id 2. 2 is the employee id of Ken. So Ken is Robin's manager.

    Thus, in this process, we need to look into same table twice.

- Example:

```
Select e.employeeid, e.firstname||' '||e.lastname employeename,
m.employeeid managerId, m.firstname ||' '|| m.lastname manager_name
from employee e, employee m
where e.managerid= m.employeeid;
```

EMPLOYEEID	EMPLOYEE NAME	MANAGERID	MANAGER_NAME
3	James Walton	1	John Demn
5	ajay ghosala	2	Ken Dale
4	robin sngal	2	Ken Dale

Persistent

Outer Join: Oracle proprietary Syntax

- If a particular row does not satisfy the Join condition, it will not appear in the query result. Thus, to include such rows in the result, user can use outer join condition.
- Outer Join operator is (+) : plus sign enclosed in parenthesis.
- This (+) operator has the effect of creating one or more null rows, to which one or more rows from non deficient table can be joined.
- Syntax:

```
SELECT table1.column, table2.column

From table1, table2

Where table1.column1=table2.column2(+);
```

- (+) Operator can be placed on any side of Join condition; but not on the both sides.
- A condition involving an Outer join can not use the IN operator or be linked with another condition using OR operator.

Outer Join: Oracle proprietary Syntax

- Example:

```
Select E.employeeid, E.lastname, D.departmentid, D.departmentname

From employee E, department D

Where E.departmentid(+) =D.departmentid;
```

EMPLOYEEID	LASTNAME	DEPARTMENTID	DEPARTMENTNAME
1	Demn	10	Sales
2	Dale	20	IT
3	Walton	10	Sales
4	sngal	20	IT
5	ghosala	20	IT
(null)	(null)	30	Marketing

-- Output contains all the Departments whether or not it contains any Employee.

OUTER Join: ANSI SQL : 1999 syntax

- Oracle 9i and further versions also supports SQL:1999 JOIN syntax.
- Syntax:

```
Select table1.column, table2.column

From table1 LEFT/FULL/RIGHT OUTER JOIN  table2

ON (table1.column=table2.column);
```

- LEFT Outer Join retrieves all the rows from the table which is at the Left hand Side(LHS) , even if there is no match in right table. It is same as

```
..From table1, table2 Where table1.column=table2.column(+)
```

- RIGHT Outer Join retrieves all the rows from the table which is at the Right hand Side (RHS), even if there is no match in left table. It is same as

```
..From table1, table2 Where table1.column(+)=table2.column
```

Outer Join: ANSI SQL : 1999 syntax

- FULL Outer Join retrieves all the rows; matched as well as unmatched from both the tables. It is same as

```
Select table1.column, table2.column
From table1, table2 Where table1.column(+)=table2.column
UNION
Select table1.column, table2.column
From table1, table2 Where table1.column=table2.column(+)
```

- Left Outer Join Example:

```
Select E.employeeid, E.lastname, D.departmentid, D.departmentname
from employee E  Left outer join department D
on E.departmentid =D.departmentid;
```

EMPLOYEEID	LASTNAME	DEPARTMENTID	DEPARTMENTNAME
3	Walton	10	Sales
1	Demn	10	Sales
5	ghosala	20	IT
4	sngal	20	IT
2	Dale	20	IT
6	Reddies	null	null

Outer Join: ANSI SQL : 1999 syntax

- Right Outer Join Example:

```
Select E.employeeid, E.lastname, D.departmentid, D.departmentname
From employee E  Right outer join department D
On E.departmentid =D.departmentid;
```

EMPLOYEEID	LASTNAME	DEPARTMENTID	DEPARTMENTNAME
1	Demn	10	Sales
2	Dale	20	IT
3	Walton	10	Sales
4	sngal	20	IT
5	ghosala	20	IT
(null)	(null)	30	Marketing

- Full Outer Join Example:

```
Select E.employeeid, E.lastname, D.departmentid, D.departmentname
From employee E  FULL outer join department D
On E.departmentid = D.departmentid;
```

EMPLOYEEID	LASTNAME	DEPARTMENTID	DEPARTMENTNAME
1	Demn	10	Sales
2	Dale	20	IT
6	Reddies	null	null
3	Walton	10	Sales
4	sngal	20	IT
5	ghosala	20	IT
null	null	30	Marketing

Reference Material: Sites

http://docs.oracle.com/cd/B19306_01/server.102/b14200/queries006.htm

http://beginner-sql-tutorial.com/sql-joins.htm

http://docs.oracle.com/cd/B19306_01/server.102/b14200/queries004.htm

http://oreilly.com/catalog/mastorasql/chapter/ch07.html

http://www.techonthenet.com/sql/joins.php

http://psoug.org/definition/JOIN.htm

Session 6: Summary

With this we have come to an end of our Sixth session where we discussed SET operators and various types of Joins with examples.

- At the end of Nugget 6, we see that you are now able to answer following questions:
  - Explain “Differences between UNION and UNION ALL”.
  - Explain “Various SET Operators with example”
  - Explain “ Inner Join”
  - Explain : Concept of Natural Join”
  - Explain “Concept of Self Join with example”
  - Explain “OUTER Join syntax with examples”

Persistent

Thank you!

Persistent University