AGENDA

1. Joins I
2. Joins II
3. Set Operators

Data Science Academy

Joins I

Data Science Academy

Oracle JOINS: Retrieving Data From Multiple Tables

- Oracle join is used to combine columns from two or more tables based on values of the related columns. The related columns are typically the primary key column(s) of the first table and foreign key column(s) of the second table.

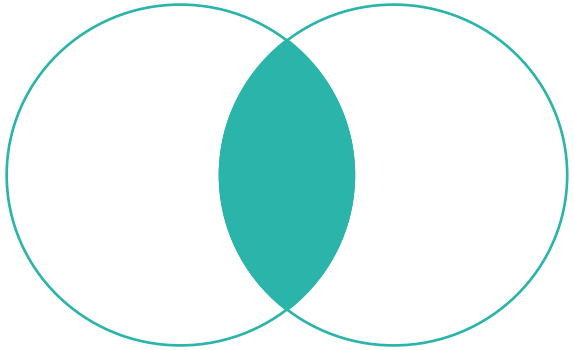
- Oracle supports inner join, left join, right join, full outer join and cross join.

- Note that you can join a table to itself to query hierarchical data using an inner join, left join, or right join. This kind of join is known as self-join.

ORACLE INNER JOIN

- An INNER JOIN combines data from two tables where there is a match on the joining column(s) in both tables.

```
1 SELECT table1.column, table2.column
2 FROM table1
3 INNER JOIN table2
4 ON table1.column_name = table2.column_name;
```



INNER JOIN

ORACLE INNER JOIN

- In the examples below, we are returning the DEPARTMENT_NAME and the EMPLOYEE_NAME for each employee. Remember, the INNER keyword is optional.

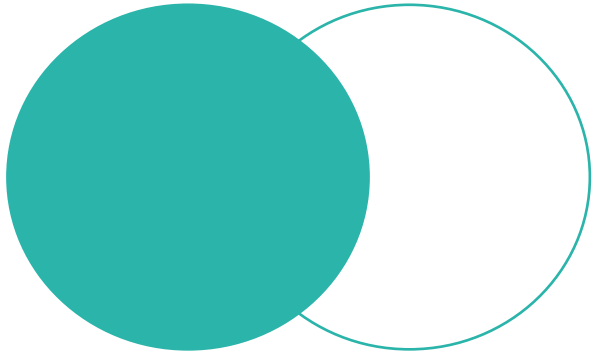
```
1 SELECT d.department_name, e.first_name
2 FROM hr.departments d
3 JOIN hr.employees e
4 ON d.department_id = e.department_id
5 WHERE d.department_id >= 30
6 ORDER BY d.department_name;
```

Department_name	First_name
Accounting	William
Accounting	Shelley
Executive	Neena
Executive	Lex
Executive	Steven
Finance	Daniel
...	

ORACLE LEFT JOIN

- The LEFT [OUTER] JOIN returns all rows from the left table with the matching rows if available from the right table. If there is no matching row found from the right table, the left join will have null values for the columns of the right table:

```
1 SELECT  d.department_name, e.first_name
2 FROM hr.departments d
3 LEFT OUTER JOIN hr.employees e
4 ON d.department_id = e.department_id
5 WHERE d.department_id >= 30
6 ORDER BY d.department_name, e.first_name;
```



LEFT OUTER JOIN

ORACLE LEFT JOIN

- Adding filters to columns returned from an outer joined table is a common cause for confusion.
- If you test for a specific value, for example "salary >= 2000", but the value for the SALARY column is NULL because the row is missing, a regular condition in the WHERE clause will throw the row away, therefore defeating the object of doing an outer join.
- Using the ANSI join syntax, filters on columns from the outer joined table are included in the join itself, rather than being placed in the WHERE clause.

```
1  SELECT  d.department_name, e.first_name
2  FROM hr.departments d
3  LEFT OUTER JOIN hr.employees e
4  ON d.department_id = e.department_id
5  AND e.salary >= 2000
6  WHERE d.department_id >= 30
   ORDER BY d.department_name, e.first_name;
```

Department_name	First_name
Accounting	Shelley
Accounting	William
Benefits	(null)
Construction	(null)
Contracting	(null)
Control And Credit	(null)
...	

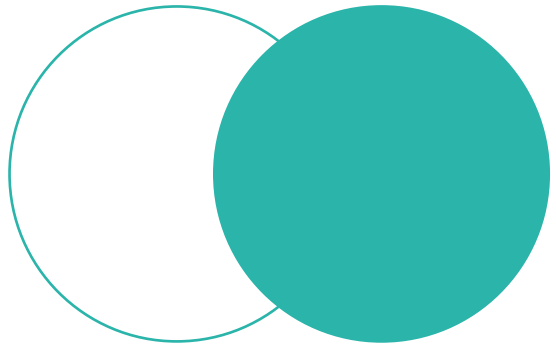
ORACLE RIGHT JOIN

- The right join or right outer join is a reversed version of the left join. The right join makes a result set that contains all rows from the right table with the matching rows from the left table. If there is no match, the left side will have nulls.

- The following example use right join to join the left table to the right table:

```
1 SELECT  d.department_name, e.first_name
2 FROM hr.employees e
3 RIGHT OUTER JOIN hr.departments d
4 ON e.department_id = d.department_id
5 WHERE d.department_id >= 30
6 ORDER BY d.department_name, e.first_name;
```

Department_name	First_name
Accounting	Shelley
Accounting	William
Benefits	(null)
Construction	(null)
Contracting	(null)
Control And Credit	(null)
...	



RIGHT OUTER JOIN

II

Joins II

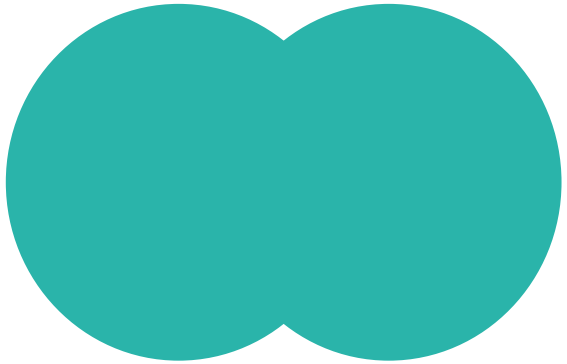
Data Science Academy

ORACLE FULL OUTER JOIN

- Oracle FULL OUTER JOIN or FULL JOIN returns a result set that contains all rows from both left and right tables, with the matching rows from both sides where available. If there is no match, the missing side will have nulls.

```
1 SELECT  d.department_name, e.first_name
2 FROM hr.employees e
3 FULL OUTER JOIN hr.departments d
4 ON e.department_id = d.department_id
5 ORDER BY d.department_name, e.first_name;
```

Department_name	First_name
Accounting	Shelley
Accounting	William
Administration	Jennifer
Benefits	(null)
Construction	(null)
Contracting	(null)
...	



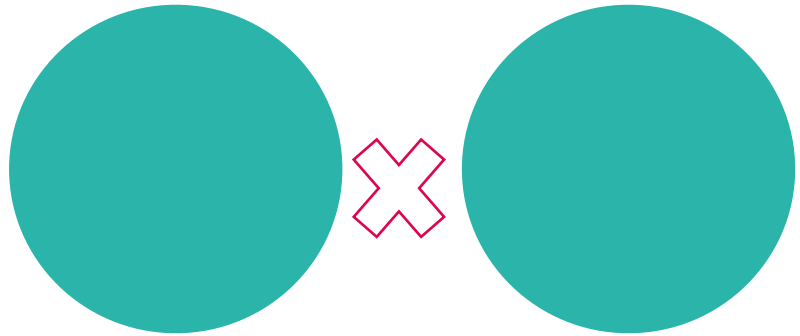
FULL OUTER JOIN

CROSS JOIN

- A CROSS JOIN is the deliberate creation of a Cartesian product. There are no join columns specified, so every possible combination of rows between the two tables is produced.

```
1 SELECT e.first_name, d.department_name
2 FROM hr.employees e
3 CROSS JOIN hr.departments d
4 ORDER BY e.first_name, d.department_name;
```

Department_name	First_name
Accounting	Adam
Administration	Adam
Benefits	Adam
Construction	Adam
Contracting	Adam
Control And Credit	Adam
...	



CROSS JOIN

[INNER] JOIN … USING

- The INNER JOIN … USING is almost a half-way house between a conventional INNER JOIN and a NATURAL JOIN. The join is made using columns with matching names in each table, but you have to specify the columns to be used, not the whole condition. This allows you to join on a subset of the columns common to both tables.

```
1 SELECT e.first_name, d.department_name
2 FROM hr.employees e
3 JOIN hr.departments d
4 USING(department_id)
5 ORDER BY e.first_name;
```

First_name	Department_name
Adam	Shipping
Alana	Shipping
Alberto	Sales
Alexander	IT
Alexander	Purchasing
Alexis	Shipping
…	

JOINING MULTIPLE TABLES

- Display department name, manager name, and city:

```
1 SELECT department_name, first_name, city
2 FROM hr.departments d
3 JOIN hr.employees e
4 ON (d.manager_id=e.employee_id)
5 JOIN hr.locations l
6   USING (location_id)
```

Department_name	First_name	City
Public Relations	Hermann	Munich
Shipping	Adam	San Francisco
Finance	Nancy	Seattle
Marketing	Michael	Toronto
Accounting	Shelley	Seattle
…		

- Display employee name, job title for the jobs employee did in the past where the job was done less than six months:

```
1 SELECT first_name, job_title
2 FROM hr.employees e
3 JOIN hr.job_history jh
4 ON (jh.employee_id = e.employee_id)
5 JOIN hr.jobs j
6 ON (jh.job_id = j.job_id)
7 WHERE months_between(end_date,start_date) > 6
```

First_name	Job_Title
Lex	Programmer
Neena	Public Accountant
Neena	Accounting Manager
Michael	Marketing Representative
Den	Stock Clerk
…	

ORACLE SELF JOIN

- A self join is a join that joins a table with itself. A self join is useful for comparing rows within a table or querying hierarchical data.
- A self join uses other joins such as inner join and left join. In addition, it uses the table alias to assign the table different names in the same query.
- Note that referencing the same table more than once in a query without using table aliases cause an error.
- The following illustrates how the table T is joined with itself:

```
SELECT column_list
FROM T alias_1
INNER JOIN T alias_2
ON join_predicate;
```

ORACLE SELF JOIN

- To retrieve the employee and manager data from the employees table, you use a self join as shown in the following statement:

```
1  SELECT
2    (e.first_name || ' ' || e.last_name) employee,
3    (m.first_name || '  ' || m.last_name) manager,
4     e.job_title
5  FROM
6     hr.employees e
7  LEFT JOIN hr.employees m ON
8     m.employee_id = e.manager_id
9  ORDER BY
10    manager;
```

Employee	Manager
Steven King	Adam Fripp
Laura Bissot	Adam Fripp
Mozhe Atkinson	Adam Fripp
James Marlow	Adam Fripp
TJ Olson	Adam Fripp
Anthony Cabrio	Adam Fripp

- This query references to the employees table twice: one as e (for employee) and another as m (for manager). The join predicate matches employees and managers using the employee_id and manager_id columns.

III

Set Operators

Data Science Academy

SET OPERATORS

- Set operators combine the results of two component queries into a single result. Queries containing set operators are called compound queries

Operator	Returns
UNION	All distinct rows selected by either query
UNION ALL	All rows selected by either query, including all duplicates
INTERSECT	All distinct rows selected by both queries
MINUS	All distinct rows selected by the first query but not the second

SET OPERATORS UNION

- The following statement combines the results of two queries with the UNION operator, which eliminates duplicate selected rows.

```
1 SELECT department_id
2 FROM hr.employees emp
3 UNION SELECT department_id
4 FROM hr.departments dep;
```

Department_id
10
20
30
40
50
…

SET OPERATORS UNION ALL

- The UNION operator returns only distinct rows that appear in either result, while the UNION ALL operator returns all rows. The UNION ALL operator does not eliminate duplicate selected rows:

```
1 SELECT location_id
2 FROM hr.locations
3 UNION ALL SELECT location_id
4 FROM hr.departments;
```

```
Location_id
    1000
    1100
    1200
    1300
    1400
...
```

SET OPERATORS INTERSECT

- The following statement combines the results with the INTERSECT operator, which returns only those rows returned by both queries:

```
1 SELECT jobs_id
2 FROM hr.employees emp
3 INTERSECT SELECT jobs_id
4 FROM hr.jobs j;
```

```
   Job_id
AC_ACCOUNT
AC_MGR
AD_ASST
AD_PRES
AD_VP
...
```

SET OPERATORS MINUS

- The following statement combines results with the MINUS operator, which returns only unique rows returned by the first query but not by the second:

```
1 SELECT job_id
2 FROM hr.employees
3 MINUS SELECT job_id
4 FROM hr.job_history;
```

```
 Job_id
AD_PRES
AD_VP
FI_ACCOUNT
FI_MGR
HR_REP
...
```

SUBQUERIES

- A subquery is a SELECT statement nested inside another statement such as SELECT, INSERT, UPDATE or DELETE.
- Typically, you can use a subquery anywhere that you use an expression.
- By using a subquery, we can nest the first query inside the second one as shown in the following statement:
- In this example Oracle evaluates the whole query above in two steps:
  - First, execute the subquery.
  - Second, use the result of the subquery in the outer query.

```
SELECT
    column1,
    column2,
    column3
FROM
    table1
WHERE
    column3 = (
        SELECT
        MAX(table2co
        lumn3 ) FROM
            table2);

```

SUBQUERIES

- It represents the subquery returns a set of rows to find all departments that do actually have one or more employees assigned to them.

```
1 SELECT department_name
2  FROM hr.departments
3 WHERE hr.departments
4 IN (SELECT DISTINCT(department_id)
5 FROM hr.employees);
```

Thank you!

Data Science
Academy