



# **Visualización de Datos de Varias Tablas Utilizando Uniones**

# Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Escribir sentencias `SELECT` para acceder a datos de más de una tabla mediante uniones igualitarias y no igualitarias
- Unir una tabla consigo misma mediante autounión
- Ver datos que normalmente no cumplen una condición de unión mediante uniones `OUTER`
- Generar un producto cartesiano de todas las filas de una o más tablas

# Agenda

- Tipos de JOINS y sintaxis
- Unión natural:
  - Cláusula USING
  - Cláusula ON
- Autounión
- Uniones no igualitarias
- Unión OUTER:
  - Unión LEFT OUTER
  - Unión RIGHT OUTER
  - Unión FULL OUTER
- Producto cartesiano
  - Unión cruzada


# Obtención de Datos de Varias Tablas

**EMPLOYEES**

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
...			
18	174	Abel	80
19	176	Taylor	80
20	178	Grant	(null)

**DEPARTMENTS**

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700



	EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	10	Administration
2	201	20	Marketing
3	202	20	Marketing
4	124	50	Shipping
...			
18	205	110	Accounting
19	206	110	Accounting

# Tipos de Uniones

Las uniones compatibles con el estándar SQL:1999 incluyen los siguientes elementos:

- Uniones naturales:
  - Cláusula `NATURAL JOIN`
  - Cláusula `USING`
  - Cláusula `ON`
- Uniones `OUTER`:
  - `LEFT OUTER JOIN`
  - `RIGHT OUTER JOIN`
  - `FULL OUTER JOIN`
- Uniones cruzadas

# Unión de Tablas mediante la Sintaxis SQL:1999

Utilizar una unión para consultar datos de más de una tabla:

```
SELECT    table1.column, table2.column
FROM      table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)] |
[CROSS JOIN table2];
```

# Cualificación de Nombres de Columna Ambiguos

- Utilizar prefijos de tabla para cualificar los nombres de columna que están en varias tablas.
- Utilizar prefijos de tabla para mejorar el rendimiento.
- En lugar de prefijos de nombre de tabla completos, utilizar alias de tabla.
- Los alias de tablas proporciona un nombre más corto de una tabla:
  - Mantiene el código SQL más pequeño, utiliza menos memoria
- Utilizar alias de columna para distinguir columnas que tienen nombres idénticos, pero que residen en diferentes tablas.

# Agenda

- Tipos de JOINS y sintaxis
- Unión natural:
  - Cláusula USING
  - Cláusula ON
- Autounión
- Uniones no igualitarias
- Unión OUTER:
  - Unión LEFT OUTER
  - Unión RIGHT OUTER
  - Unión FULL OUTER
- Producto cartesiano
  - Unión cruzada



# Creación de Uniones Naturales

- La cláusula `NATURAL JOIN` está basada en todas las columnas de las dos tablas que tienen el mismo nombre.
- Selecciona filas de las dos tablas que tienen valores iguales en todas las columnas coincidentes.
- Si las columnas que tienen el mismo nombre tienen tipos de dato diferentes, se devolverá un error.

# Recuperación de Registros con Uniones Naturales

```
SELECT  department_id, department_name,  
        location_id, city  
FROM    departments  
NATURAL JOIN locations ;
```



	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	60	IT	1400	Southlake
2	50	Shipping	1500	South San Francisco
3	10	Administration	1700	Seattle
4	90	Executive	1700	Seattle
5	110	Accounting	1700	Seattle
6	190	Contracting	1700	Seattle
7	20	Marketing	1800	Toronto
8	80	Sales	2500	Oxford

# Creación de Uniones con la Cláusula USING

- Si varias columnas tienen el mismo nombre pero los tipos de dato no coinciden, utilizar la cláusula `USING` para especificar las columnas para la unión igualitaria.
- Utilizar `USING` para que sólo coincida una columna en caso de que coincida más de una.
- Las cláusulas `NATURAL JOIN` y `USING` se excluyen mutuamente.

# Unión de Nombres de Columna



**EMPLOYEES**

	 EMPLOYEE_ID	 DEPARTMENT_ID
1	200	10
2	201	20
3	202	20
4	205	110
5	206	110
6	100	90
7	101	90
8	102	90
9	103	60
10	104	60

...

**Clave ajena**





**DEPARTMENTS**

	 DEPARTMENT_ID	 DEPARTMENT_NAME
1	10	Administration
2	20	Marketing
3	50	Shipping
4	60	IT
5	80	Sales
6	90	Executive
7	110	Accounting
8	190	Contracting

**Primary key**

# Recuperación de Registros con la Cláusula USING

```
SELECT employee_id, last_name,  
       location_id, department_id  
FROM   employees JOIN departments  
USING (department_id) ;
```

	 EMPLOYEE_ID	 LAST_NAME	 LOCATION_ID	 DEPARTMENT_ID
1	200	Whalen	1700	10
2	201	Hartstein	1800	20
3	202	Fay	1800	20
4	144	Vargas	1500	50
5	143	Matos	1500	50
6	142	Davies	1500	50
7	141	Rajs	1500	50
8	124	Mourgos	1500	50

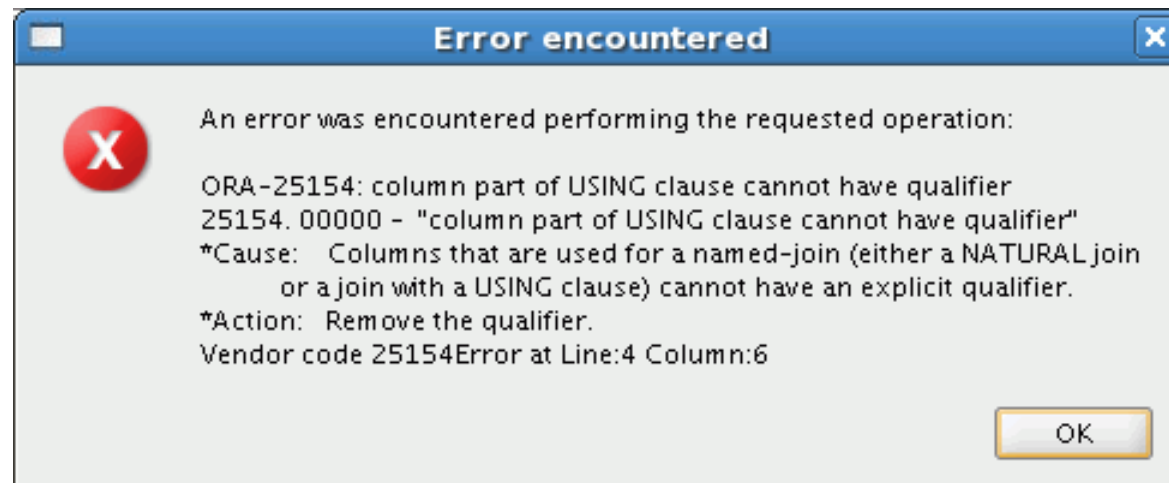
...

18	206	Gietz	1700	110
19	205	Higgins	1700	110

# Uso de Alias de Tabla con la Cláusula USING

- No cualificar una columna que se utilice en la cláusula USING.
- Si la misma columna se utiliza en otro lugar de la sentencia SQL, no se le puede agregar un alias.

```
SELECT l.city, d.department_name  
FROM   locations l JOIN departments d  
USING (location_id)  
WHERE  d.location_id = 1400;
```



# Creación de Uniones con la Cláusula ON

- La condición de unión de la unión natural es básicamente una unión igualitaria de todas las columnas con el mismo nombre.
- Utilizar la cláusula `ON` para especificar condiciones arbitrarias o columnas que se van a unir.
- La condición de unión está separada de otras condiciones de búsqueda.
- La cláusula `ON` facilita la comprensión del código.

# Recuperación de Registros con la Cláusula ON

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id);
```


	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	144	Vargas	50	50	1500
5	143	Matos	50	50	1500
6	142	Davies	50	50	1500
7	141	Rajs	50	50	1500
8	124	Mourgos	50	50	1500
9	103	Hunold	60	60	1400
10	104	Ernst	60	60	1400
11	107	Lorentz	60	60	1400

...



# Creación de Uniones en 3 Direcciones con la Cláusula ON

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations l
ON     d.location_id = l.location_id;
```

	 EMPLOYEE_ID	 CITY	 DEPARTMENT_NAME
1	100	Seattle	Executive
2	101	Seattle	Executive
3	102	Seattle	Executive
4	103	Southlake	IT
5	104	Southlake	IT
6	107	Southlake	IT
7	124	South San Francisco	Shipping
8	141	South San Francisco	Shipping
9	142	South San Francisco	Shipping

...

# Aplicación de Condiciones Adicionales a una Unión

Uso de la cláusula `AND` o la cláusula `WHERE` para aplicar condiciones adicionales:

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id)  
AND    e.manager_id = 149 ;
```

O bien

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id)  
WHERE  e.manager_id = 149 ;
```

# Agenda

- Tipos de JOINS y sintaxis
- Unión natural:
  - Cláusula USING
  - Cláusula ON
- Autounión
- Uniones no igualitarias
- Unión OUTER:
  - Cláusula LEFT OUTER
  - Unión RIGHT OUTER
  - Unión FULL OUTER
- Producto cartesiano
  - Unión cruzada

# Unión de una Tabla consigo Misma

**EMPLOYEES (WORKER)**

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
200	Whalen	101
201	Hartstein	100
202	Fay	201
205	Higgins	101
206	Gietz	205
100	King	(null)
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103

...

**EMPLOYEES (MANAGER)**

EMPLOYEE_ID	LAST_NAME
200	Whalen
201	Hartstein
202	Fay
205	Higgins
206	Gietz
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst

...



**MANAGER\_ID en la tabla WORKER es igual a  
EMPLOYEE\_ID en la tabla MANAGER.**

# Autouniones que Utilizan la Cláusula ON

```
SELECT worker.last_name emp, manager.last_name mgr
FROM   employees worker JOIN employees manager
ON     (worker.manager_id = manager.employee_id);
```

	 EMP	 MGR
1	Hunold	De Haan
2	Fay	Hartstein
3	Gietz	Higgins
4	Lorentz	Hunold
5	Ernst	Hunold
6	Zlotkey	King
7	Mourgos	King

...

# Agenda

- Tipos de JOINS y sintaxis
- Unión natural:
  - Cláusula USING
  - Cláusula ON
- Autounión
- Uniones no igualitarias
- Unión OUTER:
  - Cláusula LEFT OUTER
  - Unión RIGHT OUTER
  - Unión FULL OUTER
- Producto cartesiano
  - Unión cruzada

# Uniones no igualitarias

**EMPLOYEES**

RZ	LAST_NAME	RZ	SALARY
1	Whalen		4400
2	Hartstein		13000
3	Fay		6000
4	Higgins		12000
5	Gietz		8300
6	King		24000
7	Kochhar		17000
8	De Haan		17000
9	Hunold		9000
10	Ernst		6000
...			
19	Taylor		8600
20	Grant		7000

**JOB\_GRADES**

RZ	GRADE_LEVEL	RZ	LOWEST_SAL	RZ	HIGHEST_SAL
1	A		1000		2999
2	B		3000		5999
	C		6000		9999
4	D		10000		14999
5	E		15000		24999
6	F		25000		40000

**JOB\_GRADES define el rango de valores de LOWEST\_SAL y HIGHEST\_SAL de cada GRADE\_LEVEL. Por lo tanto, la columna GRADE\_LEVEL se puede utilizar para asignar grados a cada empleado.**

# Recuperación de Registros con Uniones no Igualitarias

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e JOIN job_grades j
ON     e.salary
      BETWEEN j.lowest_sal AND j.highest_sal;
```

	LAST_NAME	SALARY	GRADE_LEVEL
1	Vargas	2500	A
2	Matos	2600	A
3	Davies	3100	B
4	Rajs	3500	B
5	Lorentz	4200	B
6	Whalen	4400	B
7	Mourgos	5800	B
8	Ernst	6000	C
9	Fay	6000	C
10	Grant	7000	C

...



# Agenda

- Tipos de JOINS y sintaxis
- Unión natural:
  - Cláusula USING
  - Cláusula ON
- Autounión
- Uniones no igualitarias
- **Unión OUTER:**
  - Cláusula LEFT OUTER
  - Unión RIGHT OUTER
  - Unión FULL OUTER
- Producto cartesiano
  - Unión cruzada

# Devolución de Registros sin Coincidencia Directa con las uniones OUTER

**DEPARTMENTS**

	DEPARTMENT_NAME	DEPARTMENT_ID
1	Administration	10
2	Marketing	20
3	Shipping	50
4	IT	60
5	Sales	80
6	Executive	90
7	Accounting	110
8	Contracting	190

**No hay ningún empleado en el departamento 190.**

**Al empleado “Grant” no se le ha asignado un ID departamento.**

**Uniones Igualitarias con EMPLOYEES**

	DEPARTMENT_ID	LAST_NAME
1	10	Whalen
2	20	Hartstein
3	20	Fay
4	110	Higgins
5	110	Gietz
6	90	King
7	90	Kochhar
8	90	De Haan
9	60	Hunold
10	60	Ernst

...

18	80	Abel
19	80	Taylor

# Uniones INNER frente a Uniones OUTER

- En SQL:1999, la unión de dos tablas que devuelven sólo filas coincidentes se denomina unión INNER.
- Una unión entre dos tablas que devuelve los resultados de la unión INNER y las filas no coincidentes de las tablas izquierda (o derecha) se denomina una unión OUTER.
- Una unión entre dos tablas que devuelve los resultados de una unión INNER y los resultados de una unión izquierda y derecha da como resultado una unión OUTER completa.

# LEFT OUTER JOIN

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

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Vargas	50	Shipping
5	Matos	50	Shipping

...

16	Kochhar	90	Executive
17	King	90	Executive
18	Gietz	110	Accounting
19	Higgins	110	Accounting
20	Grant	(null)	(null)

# RIGHT OUTER JOIN

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

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping
6	Rajs	50	Shipping
7	Mourgos	50	Shipping
8	Matos	50	Shipping

...

18	Higgins	110	Accounting
19	Gietz	110	Accounting
20	(null)	190	Contracting

# FULL OUTER JOIN

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

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting

...

17	Zlotkey	80	Sales
18	Abel	80	Sales
19	Taylor	80	Sales
20	Grant	(null)	(null)
21	(null)	190	Contracting

# Agenda

- Tipos de JOINS y sintaxis
- Unión natural:
  - Cláusula USING
  - Cláusula ON
- Autounión
- Unión no igualitaria
- Unión OUTER:
  - Cláusula LEFT OUTER
  - Unión RIGHT OUTER
  - Unión FULL OUTER
- Producto cartesiano
  - Unión cruzada

# Productos Cartesianos

- Un producto cartesiano se forma cuando:
  - Se omite una condición de unión
  - Una condición de unión no es válida
  - Todas las filas de la primera tabla se unen a todas las filas de la segunda tabla
- Se incluye siempre una condición de unión válida si desea evitar un producto cartesiano.



# Generación de un Producto Cartesiano

**EMPLOYEES (20 filas)**

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
4	205	Higgins	110
...			
19	176	Taylor	80
20	178	Grant	(null)

**DEPARTMENTS (8 filas)**

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

**Producto cartesiano:**  
**20 x 8 = 160 filas**

	EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
1	200	10	1700
2	201	20	1700
...			
21	200	10	1800
22	201	20	1800
...			
159	176	80	1700
160	178	(null)	1700

# Creación de Uniones Cruzadas

- La cláusula `CROSS JOIN` produce el producto combinado de dos tablas.
- Esto también se denomina un producto cartesiano entre dos tablas.

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

	LAST_NAME	DEPARTMENT_NAME
1	Abel	Administration
2	Davies	Administration
3	De Haan	Administration
4	Ernst	Administration
5	Fay	Administration

...

158	Vargas	Contracting
159	Whalen	Contracting
160	Zlotkey	Contracting

# Prueba

La sintaxis de unión estándar SQL:1999 soporta los siguientes tipos de uniones. ¿Qué tipos de unión no soporta la sintaxis de unión de Oracle?

1. Uniones igualitarias
2. Uniones no igualitarias
3. Unión `OUTER` izquierda
4. Unión `OUTER` derecha
5. Unión `OUTER` completa
6. Autouniones
7. Uniones naturales
8. Productos cartesianos

# Resumen

En esta lección debe haber aprendido a utilizar uniones para mostrar los datos de varias tablas utilizando:

- Uniones igualitarias
- Uniones no igualitarias
- Uniones `OUTER`
- Autouniones
- Uniones cruzadas
- Uniones naturales
- Unión `OUTER` completa (o de dos lados)

# Práctica 6: Visión General

En esta práctica se abordan los siguientes temas:

- Unión de tablas con una unión igualitaria
- Realización de uniones externas y autouniones
- Adición de condiciones