



Autor: **Nicola Strappazon**
Web: <http://nicola.strappazon.me>
Revisión: 17/11/11

Lección 5

Recibiendo datos desde múltiples tablas

Este material se encuentra basado en el curso de Fundamentos a SQL de ORACLE, el cual es adaptado para el producto PostgreSQL, todos los ejemplos, códigos fuentes y la Base de Datos HR es propiedad de ORACLE.

Objetivos

Al completar esta lección usted podrá entender los siguientes puntos:

- Escribir sentencias **SELECT** para poder combinar diversas tablas que se encuentran relacionadas.


Obteniendo datos de múltiples tablas

EMPLOYEES

employee_id	last_name	department_id
145	Russell	80
146	Partners	80
147	Errazuriz	80
148	Cambrault	80
149	Zlotkey	80
150	Tucker	80
151	Bernstein	80
152	Hall	80
153	Olsen	80
154	Cambrault	80

DEPARTMENTS

department_id	department_name	location_id
10	Administration	1700
20	Marketing	1800
30	Purchasing	1700
40	Human Resources	2400
50	Shipping	1500
60	IT	1400
70	Public Relations	2700
80	Sales	2500
90	Executive	1700
100	Finance	1700
110	Accounting	1700
120	Treasury	1700
130	Corporate Tax	1700



employee_id	department_id	department_name
202	20	Marketing
119	30	Purchasing
187	50	Shipping
107	60	IT
155	80	Sales
102	90	Executive
101	90	Executive
113	100	Finance
206	110	Accounting
(32 rows)		

Tipos de JOIN

- [INNER] JOIN
- LEFT [OUTER] JOIN
- RIGHT [OUTER] JOIN
- FULL [OUTER] JOIN
- CROSS JOIN
- NATURAL JOIN

Sintaxis básica

```
SELECT table_1.expression, table_2.expresion [, ...]
FROM   table_1
[NATURAL JOIN table_2] |
[JOIN table_2 USING(expression)] |
[JOIN table_2 ON table_2.expression = table_1.expression] |
[LEFT | RIGHT | FULL OUTER JOIN table_2
  ON table_2.expression = table_1.expression] |
[CROSS JOIN table_2];
```

Usando la Cláusula **NATURAL JOIN**

- La cláusula **NATURAL JOIN** une dos tablas mediante todas las columnas que tengan el mismo nombre.
- Selecciona los registros que provienen de las dos tablas siempre y cuando los valores de las columnas que tienen el mismo nombre coincidan.
- Si el tipo de dato es diferente en las columnas que tienen el mismo nombre, se produce un error.

Usando la cláusula NATURAL JOIN

```
SELECT employee_id, department_id, department_name
FROM employees
NATURAL JOIN departments;
```

employee_id	department_id	department_name
202	20	Marketing
119	30	Purchasing
118	30	Purchasing
117	30	Purchasing
116	30	Purchasing
115	30	Purchasing
187	50	Shipping
186	50	Shipping
185	50	Shipping
184	50	Shipping
...		
(32 rows)		

Uniendo tablas usando la cláusula **USING**

- En caso de existir varias columnas con el mismo nombre y su tipo de dato no coincide, se utiliza la cláusula **JOIN** acompañada de la cláusula **USING** para especificar cuales de las columnas serán utilizadas para unir las dos tablas.
- Utilizar la cláusula **USING** solamente cuando existe mas de una columna que coincide.
- No puede usar nombres de tablas o alias para hacer referencia a una columna que tenga nombres en común.

Uniendo columnas con el mismo nombre

EMPLOYEES

employee_id	last_name	department_id
145	Russell	80
146	Partners	80
147	Errazuriz	80
148	Cambrault	80
149	Zlotkey	80
150	Tucker	80
151	Bernstein	80
152	Hall	80
153	Olsen	80
154	Cambrault	80

Clave Foránea

DEPARTMENTS

department_id	department_name	location_id
10	Administration	1700
20	Marketing	1800
30	Purchasing	1700
40	Human Resources	2400
50	Shipping	1500
60	IT	1400
70	Public Relations	2700
80	Sales	2500
90	Executive	1700
100	Finance	1700

Clave Primaria

Uniendo columnas con el mismo nombre

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

employee_id	department_id	department_name
200	10	Administration
202	20	Marketing
201	20	Marketing
119	30	Purchasing
118	30	Purchasing
117	30	Purchasing
116	30	Purchasing
115	30	Purchasing
114	30	Purchasing
203	40	Human Resources
...		
(106 rows)		

Columnas Ambiguas

- Usar los alias en las tablas para identificar correctamente las columnas entre las tablas.
- Usar los alias en las tablas mejora el rendimiento.
- Usar los alias en las columnas que son del mismo nombre y están en distintas tablas.
- Usar los alias simplifican las consultas.

Usando el alias en las tablas

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

employee_id	last_name	department_id	department_id	location_id
200	Whalen	10	10	1700
202	Fay	20	20	1800
201	Hartstein	20	20	1800
119	Colmenares	30	30	1700
118	Himuro	30	30	1700
117	Tobias	30	30	1700
116	Baida	30	30	1700
115	Khoo	30	30	1700
114	Raphaely	30	30	1700
203	Mavris	40	40	2400
...				

(106 rows)

Creando JOIN con la cláusula ON

- Se utiliza la cláusula **ON** para especificar la unión de las columnas con distinto nombre o condiciones arbitrarias.
- Las condiciones en la cláusula **ON** se encuentran separadas de otras condiciones de búsqueda.
- La cláusula **ON** permite un código mas fácil de entender.

Usando la cláusula ON

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

employee_id	last_name	location_id	department_id
200	Whalen	1700	10
202	Fay	1800	20
201	Hartstein	1800	20
119	Colmenares	1700	30
118	Himuro	1700	30
117	Tobias	1700	30
116	Baida	1700	30
115	Khoo	1700	30
203	Mavris	2400	40
...			
(106 rows)			

Uso libre del JOIN con la cláusula ON

EMPLOYEES (Trabajadores)

employee_id	last_name	manager_id
100	King	
121	Fripp	100
120	Weiss	100
108	Greenberg	101
103	Hunold	102
106	Pataballa	103
104	Ernst	103
113	Popp	108
117	Tobias	114
116	Baida	114
126	Mikkilineni	120
129	Bissot	121
130	Atkinson	121

(107 rows)

Clave Foránea

EMPLOYEES (Jefes)

employee_id	last_name
100	King
101	Kochhar
102	De Haan
114	Raphaely
120	Weiss
121	Fripp
122	Kaufling
123	Vollman
124	Mourgos
145	Russell
146	Partners
147	Errazuriz
148	Cambrault

(15 rows)

Clave Primaria

Uso libre del JOIN con la cláusula ON

```
SELECT e.last_name AS "Trabajador",  
       m.last_name AS "Jefe"  
FROM   employees e  
JOIN   employees m ON (e.manager_id = m.employee_id);
```

Trabajador		Jefe
-----	+	-----
Russell		King
Partners		King
Errazuriz		King
Cambrault		King
Zlotkey		King
Tucker		Russell
Bernstein		Russell
Hall		Russell
Olsen		Russell
...		
(106 rows)		

Condiciones adicionales en la cláusula JOIN

```
SELECT e.last_name AS "Trabajador",  
       m.last_name AS "Jefe"  
FROM   employees e  
JOIN   employees m ON ( e.manager_id = m.employee_id  
                        AND e.manager_id = 100) ;
```

Trabajador	Jefe
Russell	King
Partners	King
Errazuriz	King
Cambrault	King
Zlotkey	King
Kochhar	King
De Haan	King
Raphaely	King
Weiss	King
Fripp	King
Kaufling	King
Vollman	King
Mourgos	King
■ ■ ■	

(14 rows)

Usando varias cláusulas JOIN juntas

```
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
200	Seattle	Administration
202	Toronto	Marketing
201	Toronto	Marketing
119	Seattle	Purchasing
118	Seattle	Purchasing
117	Seattle	Purchasing
116	Seattle	Purchasing
115	Seattle	Purchasing
114	Seattle	Purchasing
203	London	Human Resources
199	South San Francisco	Shipping
198	South San Francisco	Shipping
197	South San Francisco	Shipping
196	South San Francisco	Shipping

■ ■ ■

(106 rows)

Combinaciones de desigualdad

DEPARTMENTS

department_name	department_id
Administration	10
Marketing	20
Purchasing	30
Human Resources	40
Shipping	50
IT	60
Public Relations	70
Sales	80
Executive	90
Finance	100
Accounting	110
Contracting	190
...	
(27 rows)	

EMPLOYEES

department_id	last_name
	Grant
110	Gietz
110	Higgins
100	Urman
100	Faviet
100	Chen
100	Sciarra
100	Popp
100	Greenberg
90	King
90	De Haan
90	Kochhar
80	Russell
80	McEwen
80	Smith
80	Doran
...	
(107 rows)	

**El departamento 190 no se encuentra
asignado a ningún empleado**

Combinaciones INNER Versus OUTER

- Las combinaciones entre tablas usando las cláusulas **NATURAL JOIN**, **USING** y **ON** retorna un resultado de tipo **INNER JOIN**, donde siempre existe una relación de valores.
- Para retornar aquellos registros que no coinciden mediante una relación entre tablas, se utiliza la cláusula **OUTER JOIN**, donde podemos especificar de cual lado falta una coincidencia.
 - **LEFT OUTER JOIN**
 - **RIGHT OUTER JOIN**
 - **FULL OUTER JOIN**
- La clave foránea se encuentren con el valor **NULL**.

Usando la cláusula 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
ORDER BY d.department_id DESC;
```

last_name	department_id	department_name
Grant		
Gietz	110	Accounting
Higgins	110	Accounting
Urman	100	Finance
Faviet	100	Finance
Chen	100	Finance
Sciarra	100	Finance
Popp	100	Finance
Greenberg	100	Finance
King	90	Executive
De Haan	90	Executive
Kochhar	90	Executive
Russell	80	Sales
McEwen	80	Sales
...		

(107 rows)

Usando la cláusula RIGHT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e
RIGHT OUTER JOIN departments d
        ON e.department_id = d.department_id
ORDER BY d.department_id DESC;
```

last_name	department_id	department_name
		Payroll
		Recruiting
		Retail Sales
		Government Sales
		IT Helpdesk
		NOC
		IT Support
		Operations
...		
		Shareholder Services
		Control And Credit
		Corporate Tax
		Treasury
Higgins	110	Accounting
Gietz	110	Accounting

(122 rows)

Usando la cláusula FULL OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e
FULL OUTER JOIN departments d
        ON e.department_id = d.department_id
ORDER BY d.department_id DESC;
```

last_name	department_id	department_name
Grant		Payroll
		Recruiting
		Retail Sales
		Government Sales
		IT Helpdesk
		NOC
		IT Support
...		
		Shareholder Services
		Control And Credit
		Corporate Tax
		Treasury
Higgins	110	Accounting
Gietz	110	Accounting

(123 rows)

Producto Cartesiano

- El producto cartesiano se forma cuando:
 - Se omite una condición en el **JOIN**.
 - La condición del **JOIN** es invalida.
 - Todos los registros de la primera tabla se relacionan con todos los registros de la segunda tabla.
 - Se utiliza la cláusula **CROSS JOIN**.
- Para evitar un producto cartesiano, siempre debe incluir una condición valida dentro del **JOIN**.

Generando el producto cartesiano

DEPARTMENTS

department_name	department_id
Administration	10
Marketing	20
Purchasing	30
Human Resources	40
Shipping	50
IT	60
Public Relations	70
Sales	80
■ ■ ■	
(27 rows)	

EMPLOYEES

department_id	last_name
	Grant
110	Gietz
110	Higgins
100	Urman
100	Faviet
100	Chen
100	Sciarra
100	Popp
■ ■ ■	
(107 rows)	

Producto cartesiano:
27 x 107 = 2889 registros

employee_id	department_id	department_name
145	10	Administration
145	20	Marketing
145	30	Purchasing
145	40	Human Resources
145	50	Shipping
145	60	IT
145	70	Public Relations
■ ■ ■		
(2889 rows)		

Usando la cláusula CROSS JOIN

```
SELECT e.employee_id, d.department_id, d.department_name
FROM   employees e
CROSS JOIN departments d;
```

employee_id	department_id	department_name
145	10	Administration
145	20	Marketing
145	30	Purchasing
145	40	Human Resources
145	50	Shipping
145	60	IT
145	70	Public Relations
145	80	Sales
145	90	Executive
145	100	Finance
...		

(106 rows)

Resumen

En esta lección, usted debió entender como usar la cláusula JOIN para mostrar los datos que provienen de distintas tablas usando:

- [INNER] JOIN
- LEFT [OUTER] JOIN
- RIGHT [OUTER] JOIN
- FULL [OUTER] JOIN
- CROSS JOIN
- NATURAL JOIN