BASES DE DATOS

Consultas sobre varias tablas

BASES DE DATOS

Consultas sobre una tabla

Introducción

Producto cartesiano

Concatenación SQL 92

Concatenación SQL 99

- NATURAL JOIN
- CROSS JOIN
- INNER JOIN

Concatenación externa SQL 99

- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

Introducción

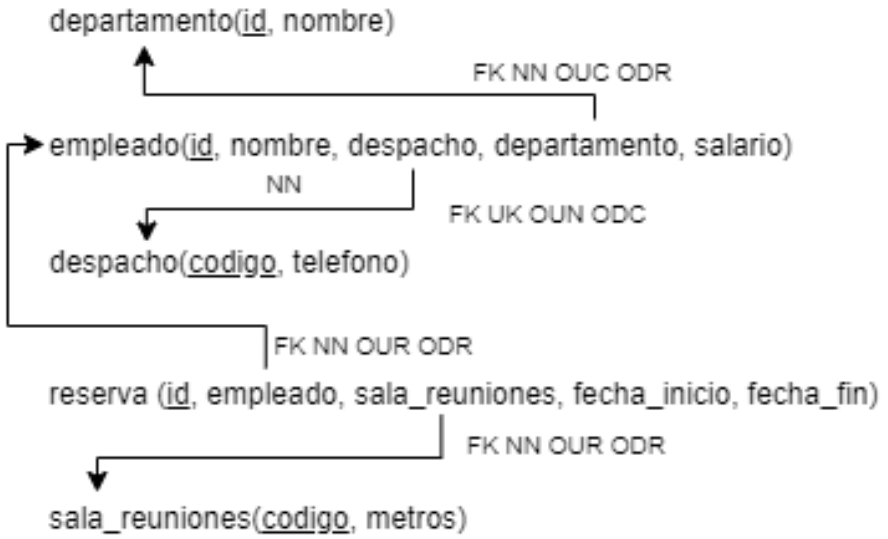
Suele ser habitual necesitar que una consulta obtenga datos que se encuentran en diferentes tablas.

Para integrar o unir los datos de varias tablas es necesario relacionarlas mediante algunas de sus columnas.

Generalmente estas columnas serán la clave ajena y la clave primaria a la que referencia.

Introducción

Todos los ejemplos de este tema hacen uso de la base de datos "oficina" cuyo grafo relacional es el siguiente:



- La fecha de fin de reserva debe ser posterior a la fecha de inicio.
- El salario por defecto es 1200€.

Puedes crear la base de datos ejecutando el *script oficina-con-datos.sql*

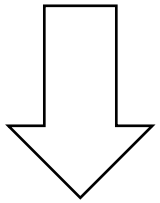
BASES DE DATOS

Consultas sobre una tabla

Producto cartesiano

Un producto cartesiano o cruzado, combina todos los registros de una tabla con todos los registros de la otra tabla.

Ejemplo: Obtén el nombre del empleado y el nombre del departamento en el que trabaja.



Los datos están en dos tablas: empleado y departamento.

Producto cartesiano

```
SELECT * FROM empleado, departamento;
```

id	nombre	despacho	departamento	salario
1	Juan	NULL	2	1200.00
2	Sara	DS01	3	2700.00
3	Sergio	NULL	1	1200.00
4	Marta	NULL	1	1400.00
5	Elena	DS02	4	1700.00
6	Sebastián	NULL	4	1600.00
7	Sara	NULL	4	1300.00
8	Martín	DS03	1	1300.00
9	Laura	NULL	1	1200.00
10	Pedro	NULL	2	1200.00
11	Leopoldo	DS04	5	1700.00

X

id	nombre
1	Informática
2	RRHH
3	Dirección
4	Administración
5	Comercial

Producto cartesiano

```
SELECT * FROM empleado, departamento;
```

id	nombre	despacho	departamento	salario	id	nombre
1	Juan	NULL	2	1200.00	1	Informática
1	Juan	NULL	2	1200.00	2	RRHH
1	Juan	NULL	2	1200.00	3	Dirección
1	Juan	NULL	2	1200.00	4	Administración
1	Juan	NULL	2	1200.00	5	Comercial
2	Sara	DS01	3	2700.00	1	Informática
2	Sara	DS01	3	2700.00	2	RRHH
2	Sara	DS01	3	2700.00	3	Dirección
2	Sara	DS01	3	2700.00	4	Administración
2	Sara	DS01	3	2700.00	5	Comercial
3	Sergio	NULL	1	1200.00	1	Informática
3	Sergio	NULL	1	1200.00	2	RRHH
3	Sergio	NULL	1	1200.00	3	Dirección
3	Sergio	NULL	1	1200.00	4	Administración
3	Sergio	NULL	1	1200.00	5	Comercial

...

Producto cartesiano

No todas las filas resultantes del producto cartesiano resultan interesantes.

id	nombre	despacho	departamento	salario	id	nombre
1	Juan	NULL	2	1200.00	1	Informática
1	Juan	NULL	2	1200.00	2	RRHH
1	Juan	NULL	2	1200.00	3	Dirección
1	Juan	NULL	2	1200.00	4	Administración
1	Juan	NULL	2	1200.00	5	Comercial

¿Para qué queremos los datos de Juan relacionados con los datos de los departamentos donde no trabaja?

A continuación veremos mecanismos para concatenar tablas según el estándar SQL 92 y SQL 99 y evitar filas no deseadas.

BASES DE DATOS

Consultas sobre una tabla

Introducción

Producto cartesiano

Concatenación SQL 92

Concatenación SQL 99

- NATURAL JOIN
- CROSS JOIN
- INNER JOIN

Concatenación externa SQL 99

- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

Concatenación de tablas

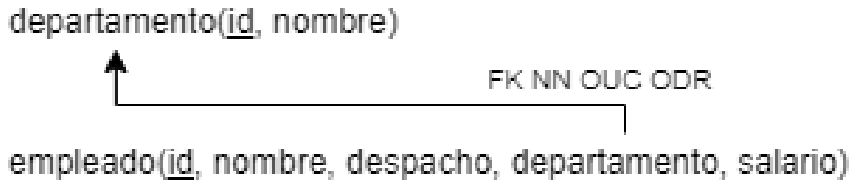
SQL 92

Se basa en el uso de la condición WHERE para unir las tablas.

Generalmente se compara la clave ajena y la clave primaria a la que referencian. Podríamos pensar en algo así:

id	nombre	despacho	departamento	salario	id	nombre
1	Juan	NULL	2	1200.00	1	Informática
1	Juan	NULL	2	1200.00	2	RRHH
1	Juan	NULL	2	1200.00	3	Dirección
1	Juan	NULL	2	1200.00	4	Administración
1	Juan	NULL	2	1200.00	5	Comercial
2	Sara	DS01	3	2700.00	1	Informática
2	Sara	DS01	3	2700.00	2	RRHH

```
SELECT * FROM empleado, departamento
WHERE departamento = id;
```



Concatenación de tablas

```
SELECT * FROM empleado, departamento
WHERE departamento = id;
```

Pero si observamos el resultado de realizar el produto cartesiano existen dos columnas llamadas *id*: una referente al *id* del empleado y otra referente al *id* del departamento.

id	nombre	despacho	departamento	salario	id	nombre
1	Juan	NULL	2	1200.00	1	Informática
1	Juan	NULL	2	1200.00	2	RRHH
1	Juan	NULL	2	1200.00	3	Dirección
1	Juan	NULL	2	1200.00	4	Administración
1	Juan	NULL	2	1200.00	5	Comercial
2	Sara	DS01	3	2700.00	1	Informática
2	Sara	DS01	3	2700.00	2	RRHH

Concatenación de tablas

Para evitar la ambigüedad podemos utilizar el nombre de la tabla como prefijo a la hora de referirnos a la columna:

```
SELECT * FROM empleado, departamento
WHERE departamento = departamento.id;
```

id	nombre	despacho	departamento	salario	id	nombre
1	Juan	NULL	2	1200.00	1	Informática
1	Juan	NULL	2	1200.00	2	RRHH
1	Juan	NULL	2	1200.00	3	Dirección
1	Juan	NULL	2	1200.00	4	Administración
1	Juan	NULL	2	1200.00	5	Comercial
2	Sara	DS01	3	2700.00	1	Informática
2	Sara	DS01	3	2700.00	2	RRHH

Concatenación de tablas

```
SELECT * FROM empleado, departamento
WHERE departamento = departamento.id;
```

id	nombre	despacho	departamento	salario	id	nombre
1	Juan	NULL	2	1200.00	1	Informática
1	Juan	NULL	2	1200.00	2	RRHH
1	Juan	NULL	2	1200.00	3	Dirección
1	Juan	NULL	2	1200.00	4	Administración
1	Juan	NULL	2	1200.00	5	Comercial
2	Sara	DS01	3	2700.00	1	Informática
2	Sara	DS01	3	2700.00	2	RRHH
2	Sara	DS01	3	2700.00	3	Dirección
2	Sara	DS01	3	2700.00	4	Administración
2	Sara	DS01	3	2700.00	5	Comercial
3	Sergio	NULL	1	1200.00	1	Informática
3	Sergio	NULL	1	1200.00	2	RRHH
3	Sergio	NULL	1	1200.00	3	Dirección

...

Concatenación de tablas

```
SELECT * FROM empleado, departamento
WHERE departamento = departamento.id;
```

id	nombre	despacho	departamento	salario	id	nombre
3	Sergio	NULL	1	1200.00	1	Informática
4	Marta	NULL	1	1400.00	1	Informática
8	Martín	DS03	1	1300.00	1	Informática
9	Laura	NULL	1	1200.00	1	Informática
1	Juan	NULL	2	1200.00	2	RRHH
10	Pedro	NULL	2	1200.00	2	RRHH
2	Sara	DS01	3	2700.00	3	Dirección
5	Elena	DS02	4	1700.00	4	Administración
6	Sebastián	NULL	4	1600.00	4	Administración
7	Sara	NULL	4	1300.00	4	Administración
11	Leopoldo	DS04	5	1700.00	5	Comercial

Concatenación de tablas

El enunciado nos pedía obtener únicamente el nombre del empleado y el del departamento donde trabaja.

De nuevo tenemos un problema de ambigüedad:

```
SELECT nombre, nombre FROM empleado, departamento
WHERE departamento = departamento.id;
```

id	nombre	despacho	departamento	salario	id	nombre
3	Sergio	NULL	1	1200.00	1	Informática
4	Marta	NULL	1	1400.00	1	Informática
8	Martín	DS03	1	1300.00	1	Informática
9	Laura	NULL	1	1200.00	1	Informática
1	Juan	NULL	2	1200.00	2	RRHH
10	Pedro	NULL	2	1200.00	2	RRHH
2	Sara	DS01	3	2700.00	3	Dirección
5	Elena	DS02	4	1700.00	4	Administración
6	Sebastián	NULL	4	1600.00	4	Administración

Concatenación de tablas

```
SELECT empleado.nombre, departamento.nombre
FROM empleado, departamento
WHERE departamento = departamento.id;
```

id	nombre	despacho	departamento	salario	id	nombre
3	Sergio	NULL	1	1200.00	1	Informática
4	Marta	NULL	1	1400.00	1	Informática
8	Martín	DS03	1	1300.00	1	Informática
9	Laura	NULL	1	1200.00	1	Informática
1	Juan	NULL	2	1200.00	2	RRHH
10	Pedro	NULL	2	1200.00	2	RRHH
2	Sara	DS01	3	2700.00	3	Dirección
5	Elena	DS02	4	1700.00	4	Administración
6	Sebastián	NULL	4	1600.00	4	Administración

Concatenación de tablas

```
SELECT empleado.nombre, departamento.nombre
FROM empleado, departamento
WHERE departamento = departamento.id;
```

nombre	nombre
Sergio	Informática
Marta	Informática
Martín	Informática
Laura	Informática
Juan	RRHH
Pedro	RRHH
Sara	Dirección
Elena	Administración
Sebastián	Administración
Sara	Administración
Leopoldo	Comercial

Concatenación de tablas

Para evitar tener que escribir repetidas veces el nombre de las tablas podemos utilizar alias de tablas:

```
SELECT emp.nombre, dep.nombre
FROM empleado AS emp, departamento AS dep
WHERE departamento = dep.id;
```

Podemos seguir utilizando alias en las columnas:

```
SELECT emp.nombre AS empleado, dep.nombre AS departamento
FROM empleado AS emp, departamento AS dep
WHERE departamento = dep.id;
```

BASES DE DATOS

Consultas sobre una tabla

Concatenación de tablas

SQL 99

NATURAL JOIN

Establece una relación de igualdad entre las tablas a través de los campos que tengan el mismo nombre en ambas tablas.

```
SELECT * FROM empleado NATURAL JOIN reserva;
```

id	nombre	despacho	departamento	salario
1	Juan	NULL	2	1200.00
2	Sara	DS01	3	2700.00
3	Sergio	NULL	1	1200.00
4	Marta	NULL	1	1400.00
5	Elena	DS02	4	1700.00
6	Sebastián	NULL	4	1600.00
7	Sara	NULL	4	1300.00
8	Martín	DS03	1	1300.00
9	Laura	NULL	1	1200.00
10	Pedro	NULL	2	1200.00

id	empleado	sala_reuniones	fecha_inicio	fecha_fin
1	1	SR01	2021-03-25 18:30:00	2021-03-25 19:30:00
2	2	SR01	2021-07-23 09:00:00	2021-07-23 10:30:00
3	2	SR02	2021-11-10 09:00:00	2021-11-10 10:00:00
4	3	SR03	2021-05-07 12:00:00	2021-05-07 13:00:00

Concatenación de tablas

SQL 99

NATURAL JOIN

Establece una relación de igualdad entre las tablas a través de los campos que tengan el mismo nombre en ambas tablas.

```
SELECT * FROM empleado NATURAL JOIN reserva;
```

id	nombre	despacho	departamento	salario	empleado	sala_reuniones	fecha_inicio	fecha_fin
1	Juan	NULL	2	1200.00	1	SR01	2021-03-25 18:30:00	2021-03-25 19:30:00
2	Sara	DS01	3	2700.00	2	SR01	2021-07-23 09:00:00	2021-07-23 10:30:00
3	Sergio	NULL	1	1200.00	2	SR02	2021-11-10 09:00:00	2021-11-10 10:00:00
4	Marta	NULL	1	1400.00	3	SR03	2021-05-07 12:00:00	2021-05-07 13:00:00

Es equivalente a:

```
SELECT * FROM empleado AS e, reserva AS r
WHERE e.id = r.id;
```

Concatenación de tablas

NATURAL JOIN

Es poco utilizado ya que obliga durante el diseño de la base de datos a asignar el mismo nombre a claves ajenas y a claves primarias.

Además, relaciona todas las columnas que tengan el mismo nombre, lo cual puede ser problemático. Por ejemplo, la siguiente sentencia no devuelve ninguna fila:

```
SELECT * FROM empleado NATURAL JOIN departamento;
```

Puesto que existen varias columnas con el mismo nombre (id y nombre) el equivalente con condición WHERE sería:

```
SELECT * FROM empleado AS emp, departamento AS dep
WHERE emp.id = dep.id AND emp.nombre = dep.nombre;
```

Concatenación de tablas

CROSS JOIN

Realiza un producto cartesiano (ya visto anteriormente) entre las tablas indicadas.

Eso significa que cada fila de la primera tabla se combina con cada fila de la segunda tabla.

No es una operación muy utilizada, aunque posibilita resolver consultas extremadamente complicadas.

Ejemplo: Realiza el producto cartesiano entre las tablas empleado y departamento.

```
SELECT * FROM empleado CROSS JOIN departamento;
```

Concatenación de tablas

INNER JOIN

Establece relaciones cuya condición se establece manualmente, lo que permite realizar asociaciones más complejas o bien asociaciones cuyos campos en las tablas no tienen el mismo nombre.

Ejemplo: Obtén el nombre de cada empleado y el del departamento donde trabaja.

```
SELECT emp.nombre AS empleado, dep.nombre AS departamento
FROM empleado AS emp
INNER JOIN departamento AS dep ON emp.departamento = dep.id;
```

La condición se establece tras la cláusula ON y generalmente se compara la clave ajena con la clave primaria a la que referencia.

Concatenación de tablas

INNER JOIN

El concepto es el mismo que en la concatenación de tablas mediante la cláusula WHERE, lo único que cambia es la sintaxis.

Ejemplo: Obtén el número de teléfono de aquellos empleados que tengan despacho.

```
SELECT e.nombre, d.telefono
FROM empleado e
INNER JOIN despacho d ON e.despacho = d.codigo;
```

BASES DE DATOS

Consultas sobre una tabla

Concatenación externa

La concatenación como la hemos visto hasta ahora hace que se "pierdan" filas si no existen filas relacionadas en la otra tabla.

Supongamos que tenemos las siguientes tablas:

cliente		
codigo	**nombre**	**municipio**
100	Alberto	1000
200	Carlos	1000
300	Sonia	
400	Laura	3000

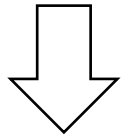
municipio	
codigo	**nombre**
1000	Elche
2000	Alicante
3000	Orihuela

Concatenación externa

cliente		
codigo	**nombre**	**municipio**
100	Alberto	1000
200	Carlos	1000
300	Sonia	
400	Laura	3000

municipio	
codigo	**nombre**
1000	Elche
2000	Alicante
3000	Orihuela

```
SELECT c.nombre AS cliente, m.nombre AS municipio
FROM cliente AS c INNER JOIN municipio AS m ON c.municipio = m.codigo;
```



cliente	municipio
Alberto	Elche
Carlos	Elche
Laura	Orihuela

Concatenación externa

```
SELECT c.nombre AS cliente, m.nombre AS municipio
FROM cliente AS c INNER JOIN municipio AS m ON c.municipio = m.codigo;
```

cliente	municipio
Alberto	Elche
Carlos	Elche
Laura	Orihuela

En el resultado final no aparece:

- La clienta Sonia, ya que no tiene municipio asignado.
- El municipio de Alicante, ya que ningún cliente es de este municipio.

Concatenación externa

En ocasiones resulta muy interesante que no se pierda ninguna fila de una u otra tabla al realizar la concatenación.

Para evitarlo es necesario hacer una concatenación externa.

La sintaxis es muy similar a la interna.

La única diferencia es que evita que se pierdan filas que no están relacionadas.

Las concatenaciones externas son: LEFT JOIN, RIGHT JOIN, FULL JOIN.

BASES DE DATOS

Consultas sobre una tabla

- Introducción
- Producto cartesiano
- Concatenación SQL 92
- Concatenación SQL 99
  - NATURAL JOIN
  - CROSS JOIN
  - INNER JOIN
- Concatenación externa SQL 99
  - **LEFT JOIN**
  - RIGHT JOIN
  - FULL JOIN

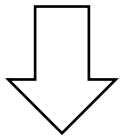
Concatenación externa

LEFT JOIN

cliente		
codigo	**nombre**	**municipio**
100	Alberto	1000
200	Carlos	1000
300	Sonia	
400	Laura	3000

municipio	
codigo	**nombre**
1000	Elche
2000	Alicante
3000	Orihuela

```
SELECT c.nombre AS cliente, m.nombre AS municipio
FROM cliente AS c LEFT JOIN municipio AS m ON c.municipio = m.codigo;
```



cliente	municipio
Alberto	Elche
Carlos	Elche
Sonia	
Laura	Orihuela

Concatenación externa

LEFT JOIN

```
SELECT c.nombre AS cliente, m.nombre AS municipio
FROM cliente AS c LEFT JOIN municipio AS m ON c.municipio = m.codigo;
```

cliente	municipio
Alberto	Elche
Carlos	Elche
Sonia	
Laura	Orihuela

En el resultado final contiene todas las filas de la tabla de la izquierda, esto es, todos los clientes, incluidos aquellos sin municipio.

En el resultado no aparece Alicante ya que no existe ningún cliente que pertenezca a éste municipio.

BASES DE DATOS

Consultas sobre una tabla

- Introducción
- Producto cartesiano
- Concatenación SQL 92
- Concatenación SQL 99
  - NATURAL JOIN
  - CROSS JOIN
  - INNER JOIN
- Concatenación externa SQL 99
  - LEFT JOIN
  - **RIGHT JOIN**
  - FULL JOIN

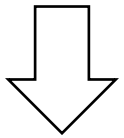
Concatenación externa

RIGHT JOIN

cliente		
codigo	**nombre**	**municipio**
100	Alberto	1000
200	Carlos	1000
300	Sonia	
400	Laura	3000

municipio	
codigo	**nombre**
1000	Elche
2000	Alicante
3000	Orihuela

```
SELECT c.nombre AS cliente, m.nombre AS municipio
FROM cliente AS c RIGHT JOIN municipio AS m ON c.municipio = m.codigo;
```



cliente	municipio
Alberto	Elche
Carlos	Elche
	Alicante
Laura	Orihuela

Concatenación externa

RIGHT JOIN

```
SELECT c.nombre AS cliente, m.nombre AS municipio
FROM cliente AS c RIGHT JOIN municipio AS m ON c.municipio = m.codigo;
```

cliente	municipio
Alberto	Elche
Carlos	Elche
	Alicante
Laura	Orihuela

En el resultado final contiene todas las filas de la tabla de la derecha, esto es, todos los municipios, incluidos aquellos sin clientes asociados.

En el resultado no aparece Sonia ya es una clienta sin municipio asociado.

BASES DE DATOS

Consultas sobre una tabla

- Introducción
- Producto cartesiano
- Concatenación SQL 92
- Concatenación SQL 99
  - NATURAL JOIN
  - CROSS JOIN
  - INNER JOIN
- Concatenación externa SQL 99
  - LEFT JOIN
  - RIGHT JOIN
  - **FULL JOIN**

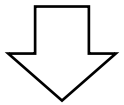
Concatenación externa

FULL JOIN

cliente		
codigo	**nombre**	**municipio**
100	Alberto	1000
200	Carlos	1000
300	Sonia	
400	Laura	3000

municipio	
codigo	**nombre**
1000	Elche
2000	Alicante
3000	Orihuela

```
SELECT c.nombre AS cliente, m.nombre AS municipio
FROM cliente AS c FULL JOIN municipio AS m ON c.municipio = m.codigo;
```



cliente	municipio
Alberto	Elche
Carlos	Elche
Sonia	
	Alicante
Laura	Orihuela

Concatenación externa

FULL JOIN

```
SELECT c.nombre AS cliente, m.nombre AS municipio
FROM cliente AS c FULL JOIN municipio AS m ON c.municipio = m.codigo;
```

cliente	municipio
Alberto	Elche
Carlos	Elche
Sonia	
	Alicante
Laura	Orihuela

El resultado final contiene todas las filas tanto de la tabla de la izquierda (cliente) como de la derecha (municipio).