Base de Datos Relacionales

Clase 9

Bases de Datos

- Uso de JOIN
- Tipos de JOIN
- Subconsultas

Sentencia JOIN

En ***programación de bases de datos***, la diferencia entre **JOIN y WHERE** es que **JOIN** se utiliza para combinar dos o más tablas en una sola tabla virtual basada en las columnas relacionadas entre ellas, mientras que **WHERE** se utiliza para filtrar filas de una tabla según una condición específica. En otras palabras, **JOIN** se utiliza para combinar tablas y WHERE se utiliza para filtrar filas.

JOIN es más eficiente que **WHERE** ya que **JOIN** combina las tablas antes de aplicar el filtro, lo que reduce el número de filas que deben ser filtradas. WHERE, por otro lado, aplica el filtro a todas las filas antes de combinar las tablas.

INNER JOIN

Esta cláusula de combinación devuelve solo aquellas filas que tienen una coincidencia en ambas tablas combinadas. Por ejemplo, puede unir las tablas de empleados y departamentos para crear un conjunto de resultados que muestre el nombre del departamento junto a cada empleado.

```
SELECT t1.codcliente, t1.nombre, t1.apellido, t2.nrofact
FROM clientes AS t1
INNER JOIN factura AS t2
ON t1.codcliente = t2.codcliente
ORDER BY t1.codcliente;
```

```
SELECT t1.codcliente, t1.nombre, t1.apellido, t2.nrofact
FROM clientes AS t1, factura AS t2
WHERE t1.codcliente = t2.codcliente
ORDER BY t1.codcliente;
```

clientes

codcliente	apellido	nombre	codpostal	condiva
100	Perez	Gabriel	5500	0
101	Gomez	Catalina	5501	0
102	Barroso	Lautaro	5500	0
104	Baez	Juan	5502	0
105	Baez	Carlos	5501	*NULL*
107	Fabres	Juan	1001	*NULL*
108	*NULL*	*NULL*	*NULL*	*NULL*

factura

nrofact	codcliente	fecha	hora
1000	100	2023-03-29 00:00:00	10:00:00
1001	101	2023-03-29 00:00:00	10:15:00
1002	102	2023-03-30 00:00:00	10:20:00
1003	101	2023-03-30 00:00:00	09:20:00
1004	102	2023-03-29 00:00:00	09:00:00
1005	104	2023-03-30 00:00:00	09:10:00
1006	101	2023-04-01 00:00:00	10:30:00
1007	103	2023-04-01 00:00:00	10:35:00
1008	105	2023-04-01 00:00:00	11:00:00
1010	102	2023-03-23 00:00:00	09:00:00

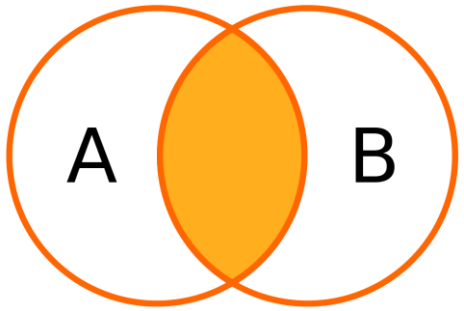


Diagrama de Venn representando el **Inner Join**, entre las tablas A y B, de una sentencia SQL

SQL:2003 especifica dos formas diferentes para expresar estas combinaciones. La primera, conocida como *explícita*, usa la palabra JOIN junto con las condiciones después de la palabra reservada *ON*. La segunda es *implícita* y usa las comas para separar las tablas a combinar en la sentencia *FROM*, y se usa la sentencia *WHERE* para establecer las condiciones, la cual entonces es obligatoria para el *INNER JOIN* pues de lo contrario la sentencia sería un *CROSS JOIN* (ver más abajo).

LEFT JOIN

Volvamos a mirar la imagen de las dos tablas (clientes y factura). Esto nos permitirá entender cómo debemos interpretar el uso de la consulta **LEFT JOIN**.

La cláusula en cuestión nos devuelve todas las filas de la tabla izquierda junto con las filas combinadas *(aquellas que establecimos como resultante de la relación)* de la tabla derecha, siempre que se cumple la condición de combinación.

```
SELECT t1.codcliente, t1.nombre, t1.apellido, t2.nrofact
FROM clientes AS t1
 LEFT JOIN factura AS t2
ON t1.codcliente = t2.codcliente
ORDER BY t1.codcliente;
```

codcliente ▲ 1	nombre	apellido	nrofact
100	Gabriel	Perez	1000
101	Catalina	Gomez	1001
101	Catalina	Gomez	1003
101	Catalina	Gomez	1006
102	Lautaro	Barroso	1002
102	Lautaro	Barroso	1004
102	Lautaro	Barroso	1010
104	Juan	Baez	1005
105	Carlos	Baez	1008
107	Juan	Fabres	*NULL*
108	*NULL*	*NULL*	*NULL*

clientes

codcliente	apellido	nombre	codpostal	condiva
100	Perez	Gabriel	5500	0
101	Gomez	Catalina	5501	0
102	Barroso	Lautaro	5500	0
104	Baez	Juan	5502	0
105	Baez	Carlos	5501	*NULL*
107	Fabres	Juan	1001	*NULL*
108	*NULL*	*NULL*	*NULL*	*NULL*

factura

nrofact	codcliente	fecha	hora
1000	100	2023-03-29 00:00:00	10:00:00
1001	101	2023-03-29 00:00:00	10:15:00
1002	102	2023-03-30 00:00:00	10:20:00
1003	101	2023-03-30 00:00:00	09:20:00
1004	102	2023-03-29 00:00:00	09:00:00
1005	104	2023-03-30 00:00:00	09:10:00
1006	101	2023-04-01 00:00:00	10:30:00
1007	103	2023-04-01 00:00:00	10:35:00
1008	105	2023-04-01 00:00:00	11:00:00
1010	102	2023-03-23 00:00:00	09:00:00

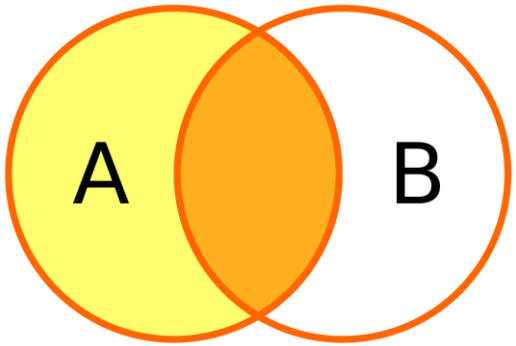


Diagrama de Venn representando el **Left Join**, entre las tablas A y B, de una sentencia SQL

LEFT JOIN

Diagrama de Venn representando el **Left Join**, entre las tablas A y B, de una sentencia SQL

El resultado de esta operación siempre contiene todos los registros de la tabla de la izquierda (la primera tabla que se menciona en la consulta), mas los elementos communes de la tabla de derecha.

retorna un valor nulo **NULL** en los campos de la tabla derecha cuando no haya correspondencia.

RIGHT JOIN

Y como podemos imaginarnos, RIGHT JOIN es lo opuesto al ejemplo anterior. En este caso, se devolverán todas las filas de la tabla de la derecha junto con las filas de la tabla izquierda para que se cumpla la condición.

```
SELECT t1.codcliente, t1.nombre, t1.apellido, t2.nrofact
FROM clientes AS t1
RIGHT JOIN factura AS t2
ON t1.codcliente = t2.codcliente
ORDER BY t1.codcliente;
```

codcliente ▲ 1	nombre	apellido	nrofact
NULL	*NULL*	*NULL*	1007
100	Gabriel	Perez	1000
101	Catalina	Gomez	1003
101	Catalina	Gomez	1006
101	Catalina	Gomez	1001
102	Lautaro	Barroso	1002
102	Lautaro	Barroso	1004
102	Lautaro	Barroso	1010
104	Juan	Baez	1005
105	Carlos	Baez	1008

clientes

codcliente	apellido	nombre	codpostal	condiva
100	Perez	Gabriel	5500	0
101	Gomez	Catalina	5501	0
102	Barroso	Lautaro	5500	0
104	Baez	Juan	5502	0
105	Baez	Carlos	5501	*NULL*
107	Fabres	Juan	1001	*NULL*
108	*NULL*	*NULL*	*NULL*	*NULL*

factura

nrofact	codcliente	fecha	hora
1000	100	2023-03-29 00:00:00	10:00:00
1001	101	2023-03-29 00:00:00	10:15:00
1002	102	2023-03-30 00:00:00	10:20:00
1003	101	2023-03-30 00:00:00	09:20:00
1004	102	2023-03-29 00:00:00	09:00:00
1005	104	2023-03-30 00:00:00	09:10:00
1006	101	2023-04-01 00:00:00	10:30:00
1007	103	2023-04-01 00:00:00	10:35:00
1008	105	2023-04-01 00:00:00	11:00:00
1010	102	2023-03-23 00:00:00	09:00:00

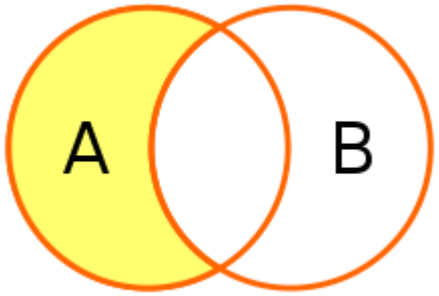
OUTER JOIN

OUTER JOIN, o unión externa, realiza una combinación de las tablas y devuelve las filas de ambas tablas vinculadas, incluso si no hay filas relacionadas entre ambas tablas combinadas.

Existen tres tipos de combinaciones externas:

- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

LEFT JOIN excluyendo la intersección



LEFT OUTER JOIN

Diagrama de Venn representando el Left Join, entre las tablas A y B, agregando una condición donde las claves de B son nulas. Si se quieren mostrar solo los registros de la primera tabla que no tengan correspondientes en la segunda, se puede agregar la condición adecuada en la cláusula *WHERE*. Esto nos dará los empleados que no estén asignados a ningún departamento, que en el diagrama de la derecha se representan en amarillo.

codcliente ▲ 1	nombre	apellido	nrofact
100	Gabriel	Perez	1000
101	Catalina	Gomez	1003
101	Catalina	Gomez	1006
101	Catalina	Gomez	1001
102	Lautaro	Barroso	1002
102	Lautaro	Barroso	1004
102	Lautaro	Barroso	1010
104	Juan	Baez	1005
105	Carlos	Baez	1008
107	Juan	Fabres	*NULL*
108	*NULL*	*NULL*	*NULL*

```
SELECT t1.codcliente, t1.nombre, t1.apellido, t2.nrofact
FROM clientes AS t1
LEFT OUTER JOIN factura AS t2
ON t1.codcliente = t2.codcliente
WHERE  t1.codcliente IS NOT NULL
ORDER BY t1.codcliente;
```

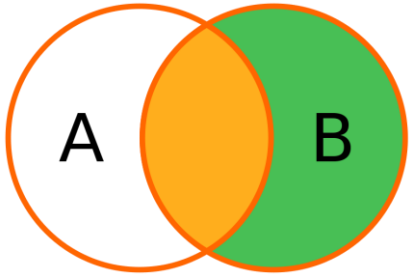


Diagrama de Venn representando el **Right Join**, entre las tablas A y B, de una sentencia SQL

```
SELECT t1.codcliente, t1.nombre, t1.apellido, t2.nrofact
FROM clientes AS t1
RIGHT OUTER JOIN factura AS t2
ON t1.codcliente = t2.codcliente
ORDER BY t1.codcliente;
```

Esta operación es una imagen refleja de la anterior; el resultado de esta operación siempre contiene **todos los registros de la tabla de la derecha** *(la segunda tabla que se menciona en la consulta),* independientemente de si existe o no un registro correspondiente en la tabla de la izquierda.

FULL JOIN (o FULL OUTER JOIN)

Devuelve todas las filas de las tablas combinadas, coincidan en una relación - o no. Lo que básicamente realiza FULL JOIN es una combinación de LEFT JOIN y RIGHT JOIN.

De igual forma a los casos anteriores, donde no se encuentre relación de un registro para con el de la otra tabla, se completarán los campos de la tabla opuesta, definiendo los datos con NULL.

Usamos la sentencia **UNION** retorna todos los valores de la tabla derecha con los valores de la tabla de la izquierda correspondientes.

```
                1                    2
SELECT clientes.codcliente, nombre FROM `clientes`
UNION
SELECT factura.codcliente, factura.nrofact FROM `factura`;
                1                    2
```

Condición: los campos mencionados para hacer la UNION deben ser del mismo tipo.
Y la cantidad de columnas deben ser iguales.

UNION Y UNION ALL

La diferencia entre Union y Union all es que **Union all no eliminará las filas duplicadas**, en su lugar, solo extrae todas las filas de todas las tablas que se ajustan a los detalles de su consulta y las combina en una tabla. Una UNION declaración efectivamente hace un SELECT DISTINCT en el conjunto de resultados.

```
SELECT clientes.codcliente FROM `clientes`
UNION ALL
SELECT factura.codcliente FROM `factura`;
```

Gracias…