



LENGUAJE SQL

Lenguaje de manipulación de datos





CONSULTAS

Consultas: Sintaxis

- SELECT permite realizar la consulta de datos contenidos en las tablas. Su sintaxis es:

```
SELECT [DISTINCT]  
[campo1, campo2, ... | *]  
FROM [NomTab1, NomTab2, ...]  
[WHERE condición]  
[ORDER BY expr_col1 [DESC|ASC] [, expr_col2 [DESC|ASC]...] ;
```

Nota: Se utiliza WHERE para conectar las distintas tablas de FROM.

Consultas: Sintaxis

Donde:

- **SELECT**, indica los campos que devuelve la consulta.
- **FROM**, la tabla o tablas de donde se extraen los datos.
- **WHERE**, las condiciones que cumplen los registros devueltos.
- **ORDER BY**, la ordenación que se quiere de los registros.
- **DISTINCT**: omite los registros que contienen datos duplicados en los campos seleccionados. Por ejemplo, varios empleados contenidos en una tabla Empleados pueden tener el mismo apellido.

Consultas: Parámetros

*: Se seleccionan todos los campos de la tabla o tablas especificadas.

NomTab1: El nombre de la tabla/s que contiene los campos.

campo1, campo2: Los nombres de los campos que contienen los datos que desea recuperar. Si incluye más de un campo, éstos se recuperan en el orden enumerado.

Si queremos que en lugar del nombre del campo aparezca otro encabezado utilizaremos la opción campo1 AS alias1, campo2 AS alias2 siendo alias1, alias2....

condición: que deben cumplir los registros que devuelve la consulta (a estas condiciones le llamamos filtros y vamos a detallarlo más adelante).

expr_col1: ordena los registros de salida por el campo que se indica en la cláusula ORDER BY. ASC o DESC permiten ordenación ascendente o descendente respectivamente.



FILTROS

Consultas: Filtros

En SQL se seleccionan los registros que se quieren mostrar con WHERE:

SELECT * FROM coches WHERE marca = 'seat';

Los filtros se construyen mediante expresiones. Una expresión es una combinación de operadores, operandos y funciones que producen un resultado.

OPERANDOS

Pueden ser constantes (números enteros: 3, reales :2.5, cadenas de caracteres: 'Europa', fechas: '2010-01-02') o variables.

Consultas: Filtros

OPERADORES

- Aritméticos: permiten formar expresiones con constantes, columnas y funciones (+, -, *, /). Ej:

*SELECT col1 * col2 FROM ALUMNOS WHERE col1-col2=15;*

- Lógicos: AND, OR, NOT.

- De comparación: =, <, >, >=, <=, <>

- De valores nulos: IS [NOT] NULL. Para averiguar si un campo no tiene ningún valor. Ej.:

SELECT Nombre FROM ALUMNOS WHERE Teléfono IS NULL;

Consultas: Filtros

De cadenas de caracteres:

- **=:** podemos comparar cadenas de caracteres completas (Nombre='Paula') pero no identificar patrones de texto incompletos.
- **LIKE** : permite identificar con patrones de texto. Admite los comodines % y _ (el % indica cualquier número de caracteres, incluso cero, mientras que _ indica un carácter exactamente).

SELECT nombre, nota FROM alumnos WHERE nombre LIKE '_a%'

(obtendría el nombre y la nota de los alumnos cuyo nombre tiene una a en la segunda letra)

Consultas: Filtros

De conjuntos de valores:

- In(valor1, valor2, valor3,...). Admite NOT delante.

```
SELECT * FROM ALUMNOS
```

```
WHERE Nombre NOT IN ('Gabriel', 'Miguel');
```

- Between: comprueba si un valor está en el rango especificado. Su formato es: [NOT] Between valor_inicial and valor_final.

```
SELECT * FROM ALUMNOS
```

```
WHERE Edad Between 15 and 17;
```

Consultas: Filtros

Por el número de registros:

Si queremos limitar el número de registros que devuelve una consulta utilizamos la opción OFFSET ... FETCH ... ROWS ONLY.

```
SELECT nombre FROM jugadores  
FETCH FIRST 1 ROWS ONLY;
```

Obtenemos el nombre de los cuatro primeros jugadores

```
SELECT nombre FROM jugadores  
OFFSET 6 ROWS FETCH NEXT 3 ROWS;
```

Obtenemos 3 filas a partir de la sexta

Consultas: Filtros

Orden de los registros:

ORDER BY expr_col1[DESC|ASC] [, expr_col2 [DESC|ASC]...];

- Permite ordenar el resultado de una consulta por los valores de una o más expresiones
- Se pueden especificar varias expresiones. Si existe igualdad para la primera expresión para dos filas se irán evaluando las siguientes expresiones para intentar resolver la igualdad.
- Por defecto es ascendente (ASC), opción descendente (DESC).

SELECT Apellidos, Nombre FROM Empleados

ORDER BY Apellidos, Nombre;



FUNCIONES AGRUPADAS



Consultas: Funciones de grupo/Agregadas

- Retornan un único resultado por cada conjunto de filas
- NO SE PUEDEN UTILIZAR EN CLAUSULAS WHERE
- Estas funciones agregadas ignoran el valor NULL y calculan resultados a pesar de su existencia

COUNT (* | [DISTINCT | ALL] columna)

- Cuenta todas las tuplas de la tabla, aunque tengan valores NULL (incluso en todos los atributos).

SELECT COUNT() FROM Alumnos;*

(Retorna el número de filas de la tabla alumnos)

Consultas: Funciones de grupo/Agregadas

COUNT (DISTINCT columna)

- Devuelve el número de valores distintos para la columna, no incluyendo NULL.

SELECT COUNT(DISTINCT Edad) FROM Alumnos;

(Número de edades distintas que hay en la tabla alumnos.)

COUNT (columna)

- Devuelve el número de valores de la columna, no incluyendo los NULL.

SELECT COUNT(Edad) FROM Alumnos;

- (Retorna el numero de alumnos para los que esta definida (no es NULL) la edad)

Consultas: Funciones de grupo/Agregadas

AVG([DISTINCT] columna| expresión)

- Devuelve la media aritmética de los valores de la columna o de los devueltos por la expresión (los valores NULL son ignorados)
- La columna o expresión deben ser numéricas.

SELECT AVG(Edad) FROM Alumnos;

(Media de edad de todos los alumnos ignora los alumnos con edad a NULL)

SELECT AVG(NVL(Edad,0)) FROM Alumnos;

(Retorna la media de edad de todos los alumnos y utiliza edad 0 en los alumnos con edad NULL)

Consultas: Funciones de grupo/Agregadas

SUM ([DISTINCT] columna | expresión)

- Devuelve la suma de los valores de la columna o de los devueltos por expresión (los valores NULL son ignorados)
- La columna o expresión debe ser numéricas.

SELECT SUM(Edad) FROM Alumnos;

(Retorna la suma de la edad de todos los alumnos, ignorando alumnos con edad a NULL)

SELECT SUM(DISTINCT Edad) FROM Alumnos;

(Retorna la suma de las edades distintas de todos los alumnos, se omiten las edades repetidas y se ignora los alumnos con edad a NULL)

Consultas: Funciones de grupo/Agregadas

MAX ([DISTINCT] columna | expresión)

- Devuelve el valor máximo de la columna o la expresión (los valores NULL son ignorados)
- La columna o expresión puede ser numéricas, cadenas de caracteres o fechas.


`SELECT MAX(Edad) FROM Alumnos;`

(Retorna la edad máxima los alumnos, ignorando edad NULL)

MIN ([DISTINCT] columna | expresión)

- Función contraria a MAX, con las mismas condiciones.

`SELECT MAX(Edad) FROM Alumnos;`



CONSULTAS DE INFORMACIÓN AGRUPADA



Consultas: Datos agrupados

Consiste en la realización de consultas obteniendo un resultado para cada grupo de datos. Las filas de la consulta son agrupadas por una o varias columnas, las cuales tienen un mismo valor.

Sintaxis

```
SELECT [ALL, DISTINCT]{[*|esquema.tabla.*],expr1[,expr2,...]}  
FROM tabla1 [, tabla2, tablas3 ...]  
[WHERE condición(s)]  
[GROUP BY exp1,[exp2, ...] ]  
[HAVING condición(s) ]  
[ORDER BY exp1|pos1 [,exp2|pos2,exp3|pos3, ...] [ASC|DESC] ]
```

Consultas: Datos agrupados

Consiste en la realización de consultas obteniendo un resultado para cada grupo de datos. Las filas de la consulta son agrupadas por una o varias columnas, las cuales tienen un mismo valor.

Sintaxis

```
SELECT [ALL, DISTINCT]{[*][esquema.tabla.*],expr1[,expr2,...]}  
FROM tabla1 [, tabla2, tablas3 ...]  
[WHERE condición(s)]  
[GROUP BY exp1,[exp2, ...] ]  
[HAVING condición(s) ]  
[ORDER BY exp1|pos1 [,exp2|pos2,exp3|pos3, ...] [ASC|DESC] ]
```

Consultas: GROUP BY

- Se utiliza para formar agrupaciones de tuplas en función de los valores de uno o varios atributos/expresiones.
- Permite agrupar la tabla en grupo de manera que cada grupo tiene el mismo valor para los atributos/expresiones elegidas en la cláusula GROUP BY.
- Pueden existir grupos de una sola fila.
- Los valores NULL se consideran iguales. Se incluyen en el mismo grupo.

Consultas: GROUP BY

Importante:

- Se utiliza con funciones de grupo o agregadas que se aplican a cada grupo
- Si se utiliza en el SELECT solo pueden aparecer
 - *Valores constantes*
 - *Los atributos/expresiones que aparecen en el GROUP BY*
 - *Funciones de grupo o agregadas (se aplican a cada grupo)*
- Las funciones de grupo o agregadas no pueden usarse en la cláusula WHERE.

Consultas: GROUP BY

```
SELECT mat, SUM(horas)
FROM trabajos
GROUP by mat
ORDER by mat;
```

TRABAJOS			
MAT	DNI	HORAS	FECHA REP
M3020KY	1111	1	23-FEB-96
M3020KY	2222	2.5	23-FEB-96
J1234Z	4444	7	19-MAR-97
J1234Z	2222	3	19-MAR-97
GR4321A	3333	2.1	1-JAN-98
B4444AC	3333	3.2	23-APR-96
CA0000AD	3333	8	23-APR-96
M3020KY	5555	2	23-FEB-96
J9999AB	6666	1	5-MAR-98
J9999AB	5555	0.6	5-MAR-98
J9999AB	2222	0.9	5-MAR-98
J1234Z	1111	2.2	19-MAR-97
GR1111AK	3333	5.5	1-JAN-98
J9999AB	3333	6.7	5-MAR-98
GR1111AK	5555	2.5	1-JAN-98
GR1111AK	7777	1	1-JAN-98

Consultas: GROUP BY

		MAT	HORAS	
		-----	-----	
MAT	SUM (HORAS)	B4444AC	3.2	SUM (horas) = 3.2
		CA0000AD	8	SUM (horas) = 8
		GR1111AK	5.5	SUM (horas) = 8
		GR1111AK	2.5	
		GR4321A	2.1	SUM (horas) = 2.1
		J1234Z	2.2	SUM (horas) = 12.2
-----	-----	J1234Z	3	
		J1234Z	7	
		J9999AB	1	SUM (horas) = 10.6
		J9999AB	7.7	
		J9999AB	.7	
		J9999AB	1.2	
B4444AC	3.2			
CA0000AD	8			
GR1111AK	8			
GR4321A	2.1			
J1234Z	12.2			
J9999AB	10.6			

Consultas: GROUP BY

```
SELECT Dept_No, Oficio, COUNT(*)  
FROM EMPLE  
GROUP BY Dept_No, Oficio;
```

10 DIRECTOR	1
10 EMPLEADO	1
10 PRESIDENTE	1
20 ANALISTA	2
20 DIRECTOR	1
20 EMPLEADO	2
30 DIRECTOR	1
30 EMPLEADO	1
30 VENDEDOR	4

Consultas: HAVING

- Las condiciones son similares a las utilizadas en WHERE, pero se pueden utilizar funciones de grupo
- Permite seleccionar los grupos que cumplan unas determinadas condiciones
- “Hace con los grupos lo que el WHERE hace con las tuplas.”
- Actúa como un filtro sobre filas agrupadas, a diferencia de la cláusula WHERE que actúa sobre las filas antes de la agrupación.
- Si se utiliza HAVING sin GROUP BY las condiciones se aplican a un único grupo formado por todas las tuplas.

Consultas: HAVING

```
SELECT mat, SUM(horas)
FROM trabajos
GROUP by mat
HAVING SUM(horas) > 10
ORDER by mat;
```

MAT	SUM (HORAS)
-----	-----
J1234Z	12.2
J9999AB	10.6

Consultas: HAVING

Orden de ejecución de las consultas:

- FROM -> Se eligen las tablas sobre las que se aplica la consulta
- WHERE -> Se selecciona las filas con el WHERE
- GROUP BY -> Se agrupan las filas según los parámetros de la cláusula GROUP BY
- HAVING -> Se seleccionan los grupos que cumplen las condiciones del HAVING
- ORDER BY -> Se ordenan las filas
- SELECT -> Se seleccionan las expresiones indicadas.

Consultas: HAVING

```
SELECT mat, SUM(horas)
FROM trabajos
WHERE mat != 'J1235Z'
HAVING SUM(horas) > 10
GROUP by mat
ORDER by mat;
```

MAT	SUM (HORAS)
-----	-----
J9999AB	10.6



SUBCONSULTAS

Lenguaje DML: Subconsultas

- Las subconsultas o consultas anidadas consisten en consultas las cuales utilizan en su filtro (where o having) una comparación entre una columna y el resultado de otra SELECT.

Equipos que no tengan jugadores españoles

```
SELECT Nombre  
FROM equipos  
WHERE NOT EXISTS (SELECT * FROM jugadores  
                   WHERE equipos.Nombre= jugadores.Nombre_equipo  
                   AND procedencia='Spain');
```




COMBINACIONES

Introducción

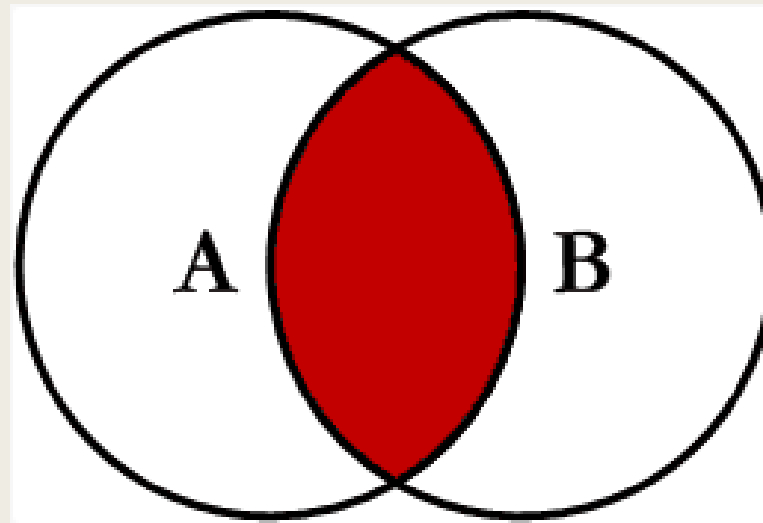
- Las bases de datos permiten la realización de consultas que impliquen a más de una tabla al mismo tiempo. Estas son llamadas multitabla.
- Es habitual encontrar consultas multitabla que únicamente incluyen aquellos registros que han sido encontrados en ambas tablas.
- Es posible, en ocasiones, que se necesite recuperar registros de una tabla, aunque éstos no tengan coincidencia en la otra.



INNER JOIN

Uniones entre tablas: INNER JOIN

- Son las uniones más utilizadas y muestran la intersección entre tablas, es decir, únicamente se muestra los registros que han podido encontrarse en ambas tablas (existe una conexión entre tablas). Puede utilizarse la abreviatura “JOIN”.



Uniones entre tablas: INNER JOIN

```
SELECT * FROM clientes c INNER JOIN empleados e  
ON c.codigoempleadorepventas = e.codigoempleado;
```

⚡ NOMBRECLIENTE	⚡ CODIGOEMPLEADOREPVENTAS	⚡ CODIGOEMPLEADO	⚡ NOMBRE
DGPRODUCTIONS...	19	19	Walter Santiago
Gardening Ass...	19	19	Walter Santiago
Gerudo Valley	22	22	Lorena
Tendo Garden	22	22	Lorena
Lasas S.A.	8	8	Mariano
Beragua	11	11	Emmanuel
Club Golf Pue...	11	11	Emmanuel
Naturagua	11	11	Emmanuel
DaraDistribuc...	11	11	Emmanuel
Madrileña de ...	11	11	Emmanuel



OUTER JOIN

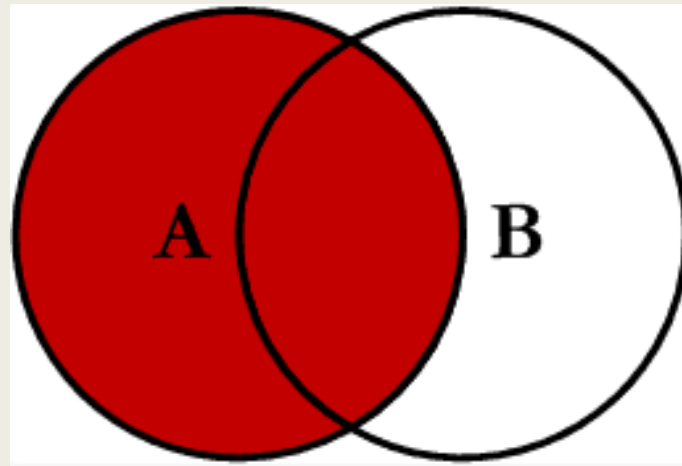


Uniones entre tablas: OUTER

- Las uniones “OUTER” van acompañadas de “LEFT/RIGHT”, ya que nos permiten recuperar registros que no tengan coincidencia en ambas tablas.
- Es muy importante saber que lugar ocupa cada tabla a la hora de recuperar registros (tabla left/right).
- Es posible utilizar una forma abreviada:
 - *LEFT OUTER JOIN* -> *LEFT JOIN*
 - *RIGHT OUTER JOIN* -> *RIGHT JOIN*

Uniones entre tablas: LEFT OUTER JOIN

- Son las uniones menos utilizadas y muestran la tabla de la izquierda por completo, además de las posibles coincidencias que pudiese haber con la tabla de su derecha.



Uniones entre tablas: INNER JOIN

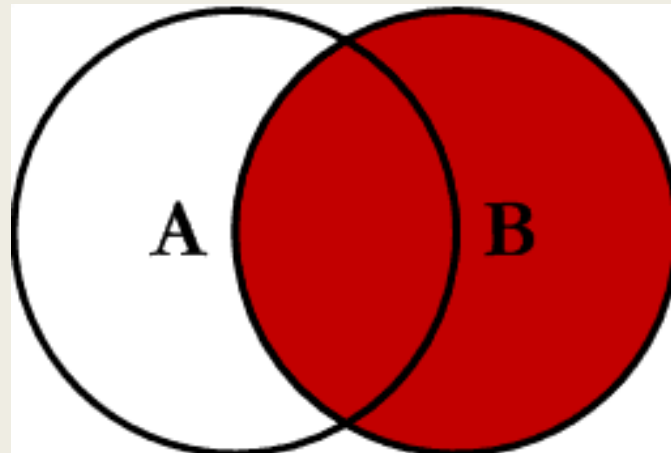
SELECT c.nombrecliente, c.codigoempleadorepventas,
e.codigoempleado, e.nombre

FROM empleados e LEFT OUTER JOIN clientes c ON
c.codigoempleadorepventas = e.codigoempleado;

NOMBRECLIENTE	CODIGOEMPLEADOREPVEN...	CODIGOEMPLEADO	NOMBRE
Campohermoso	30	30	Julian
Vivero Humanes	30	30	Julian
Naturajardin	30	30	Julian
Agrojardin	30	30	Julian
Jardin de Flores	30	30	Julian
El Jardin Viviente S.L	31	31	Mariko
Tutifruti S.A	31	31	Mariko
(null)	(null)	21	Marcus
(null)	(null)	4	Maria
(null)	(null)	10	Hilario

Uniones entre tablas: RIGHT OUTER JOIN

- Hacen lo opuesto a las anteriores, y no suelen utilizarse, ya que cambiando el sentido de la unión pueden escribirse como sentencias “LEFT OUTER JOIN”.



Uniones entre tablas: INNER JOIN

SELECT c.nombrecliente, c.codigoempleadorepventas,
e.codigoempleado, e.nombre

FROM clientes c RIGHT OUTER JOIN empleados e ON
c.codigoempleadorepventas = e.codigoempleado;

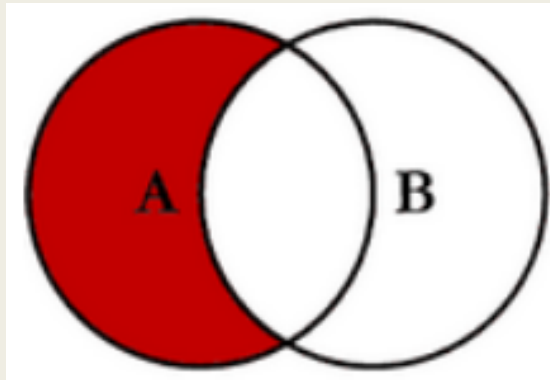
⚡ NOMBRECLIENTE	⚡ CODIGOEMPLEADOREPVEN...	⚡ CODIGOEMPLEADO	⚡ NOMBRE
FLORES S.L.	18	18	Michael
THE MAGIC GARDEN	18	18	Michael
El Jardin Viv...	31	31	Mariko
(null)	(null)	6	Juan Carlos
(null)	(null)	23	Nei
(null)	(null)	14	Oscar
(null)	(null)	27	Larry



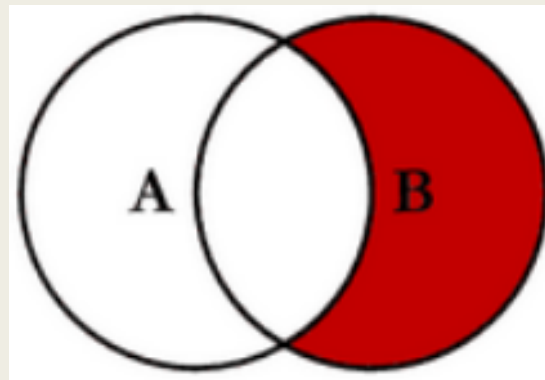
VARIANTES OUTER



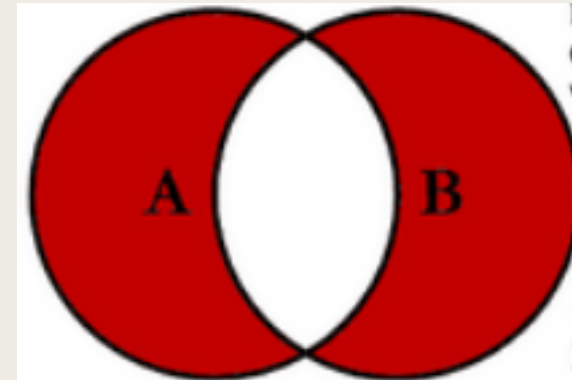
Uniones entre tablas: Otras variantes



```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL.
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL.
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL,  
OR B.Key IS NULL
```

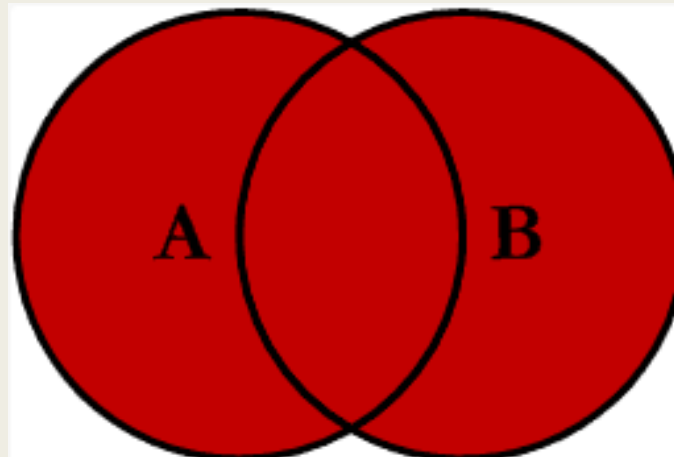


A diagram illustrating a full join operation. It features two large, thick black L-shaped brackets. The first bracket is on the left, with its vertical arm extending downwards and its horizontal arm extending to the right. The second bracket is on the right, with its vertical arm extending upwards and its horizontal arm extending to the left. These two brackets are positioned such that they appear to be enclosing a central area. In the center of this area, the text "FULL JOIN" is written in a bold, black, sans-serif font.

FULL JOIN

Uniones entre tablas: FULL JOIN

- Mientras que la cláusula JOIN muestra la intersección, y OUTER muestra la intersección acompañada de una tabla (de las incluidas), FULL JOIN muestra todos los registros de la unión, tengan o no coincidencias de cualquier tipo.



Uniones entre tablas: INNER JOIN

```
SELECT c.nombrecliente, c.codigoempleadorepventas,  
       e.codigoempleado, e.nombre  
FROM clientes c FULL JOIN empleados e  
ON c.codigoempleadorepventas = e.codigoempleado;
```

⚡ NOMBRECLIENTE	⚡ CODIGOEMPLEADOREPVENTAS	⚡ CODIGOEMPLEADO	⚡ NOMBRE
Tutifruti S.A	31	31	Mariko
FLORES S.L.	18	18	Michael
THE MAGIC GARDEN	18	18	Michael
El Jardin Vivie...	31	31	Mariko
(null)	(null)	6	Juan Carlos
(null)	(null)	23	Nei
(null)	(null)	14	Oscar
(null)	(null)	27	Larry