BBDD

UT4: DML

José Antonio
Benítez Chacón

Índice

1. Introducción

DML (Data Manipulation Language) es un componente de SQL que permite realizar operaciones sobre los datos almacenados en una base de datos.

Se centra en las operaciones de inserción consulta actualización y eliminación de información dentro de una base de datos relacional. Es lo que se conoce como **CRUD**.

Create = Insert

Read = Select

Update = Update

Delete = Delete

2. INSERT

La sentencia INSERT se utiliza para añadir una o más filas a una tabla

Sintaxis básica:

```
INSERT INTO nombre_tabla (columna1, columna2, columna3, ...)
VALUES (valor1, valor2, valor3, ...) ;
```

Ejemplo:

```
INSERT INTO estudiantes (id, nombre, apellido 1, apellido 2, edad, curso)
VALUES (2, 'Juan', 'Pérez', 'Perea', 20, 'Matemáticas');
```

2. INSERT

Variantes:

1) Insertar datos sin especificar columnas (ha de incluirse un valor para cada columna de la tabla, y en el orden en que están definidas:

```
INSERT INTO estudiantes
VALUES (2, 'Juan', 'Pérez', 'Perea', 20, 'Matemáticas');
```

2) Insertar múltiples filas:

```
INSERT INTO estudiantes (id, nombre, apellido 1, apellido 2, edad, curso)
VALUES
(3, 'María', 'López', 'Sánchez', 21, 'Física'),
(4, 'Antonio', 'Benítez', 'Domínguez', 22, 'Química'),
(5, 'Penélope', 'Cruz', 'Martín', 19, 'Teatro');
```

2. INSERT

Variantes:

3) Insertar datos a partir de una consulta a otra tabla (u otras tablas):

```
INSERT INTO ofertas (id, nombre, precio_oferta)
SELECT id, nombre, precio * 0.9
FROM productos
WHERE categoria = 'Electrónica';
```

3. UPDATE

La sentencia UPDATE se utiliza para modificar los valores de una o más filas.

Sintaxis básica:

```
UPDATE nombre_tabla
SET columna1 = valor1, columna2 = valor2, ...
WHERE condiciones;
```

Ejemplo:

```
UPDATE estudiantes
SET edad = 23
WHERE id = 1;
```

Nota: La cláusula **WHERE** es opcional, pero no usarla actualiza todas las filas de la tabla.

3. UPDATE

Actualizaciones de más de un campo:

```
UPDATE empleados
SET salario = salario * 1.10,
   departamento = 'Administración'
WHERE departamento = 'Ventas';
```

4. DELETE

La sentencia DELETE se usa para eliminar filas de una tabla.

Sintaxis básica:

```
DELETE FROM nombre_tabla
WHERE condiciones;
```

Ejemplo:

```
DELETE FROM estudiantes
WHERE curso = 'Química';
```

Nota: La cláusula **WHERE** es opcional, pero al no usarla se eliminan todas las filas de la tabla.

5. Transacciones en DML

Las sentencias DML se suelen ejecutar dentro de transacciones para garantizar la consistencia de los datos. Esto incluye el uso de:

- **BEGIN TRANSACTION**: Inicia una transacción.
- **COMMIT**: Confirma los cambios realizados.
- **ROLLBACK**: Revierte los cambios realizados durante la transacción.

```
BEGIN TRANSACTION;
INSERT INTO estudiantes (id, nombre, edad, curso)
VALUES (4, 'Ana Torres', 19, 'Biología');
UPDATE estudiantes
SET curso = 'Física'
WHERE nombre = 'Carlos Gómez';
ROLLBACK; -- Deshace los cambios
```

6. Restricciones y reglas

Las operaciones de DML respetan las relaciones entre tablas definidas entre tablas por claves principales y ajenas.
Ej: No se puede eliminar una fila referenciada por otra tabla, a menos que se permita con ON DELETE CASCADE o con ON DELETE NULL.

Además, los datos insertados o actualizados deben cumplir con las restricciones definidas en el esquema de la base de datos, como:

- NOT NULL
- UNIQUE
- CHECK

7. SELECT

La sentencia SELECT se utiliza para consultar datos de una o varias tablas.

Sintaxis básica:

```
SELECT columnas
FROM nombre_tabla
WHERE condiciones;
```

Ejemplo:

```
SELECT nombre, curso
FROM estudiantes
WHERE edad > 20;
```

Nota: La cláusula **WHERE** es opcional, pero no usarla recupera todas las filas de la tabla.

7. SELECT

Partes de una consulta:

```
SELECT nombre, curso
FROM estudiantes
WHERE edad > 20;
```

SELECT: Sentencia que indica que estamos ante una consulta y que precede a la **información que desea mostrarse** (sean valores directos de columnas, cálculos sobre los mismos u otras operaciones). Si lo que desea es extraerse la información de todas las columnas, se puede utilizar * (asterisco). Si en la consulta se recupera información de una o varias tablas, se puede usar * precedido por el nombre o alias de la tabla.

FROM: Palabra clave para indicar el **origen u orígenes de los datos** que se van a consultar y que se indicarán justo tras dicha palabra.

WHERE: Cláusula para aplicar **condiciones** a la consulta

7. SELECT

Condiciones

- Las condiciones se utilizan para aplicar filtros de búsqueda y, así, acotar los resultados.
- Vienen precedidas de la cláusula WHERE y se unen entre sí por operadores lógicos (AND, OR)
- Las condiciones pueden ser de diferentes tipos. Aunque normalmente se aplican a columnas de las tablas involucradas en la consulta (buscando un valor concreto, evitándolo, etc.), pueden tenerse en cuenta factores externos, como puede ser la hora del día, el día de la semana, el usuario que lanza la consulta, etc.
- Entre los operadores de condiciones más comunes están el **=** y el **like**.

Operador	Descripción	Operador	Descripción
=	Igual a	<>	Distinto de (en algunos SGBD: !=)
>	Mayor que	BETWEEN	Valor en un rango
<	Menor que	LIKE	como (un patrón sencillo)
>=	Mayor o igual que	IN	en (pertenencia)
<=	Menor o igual que		

7. SELECT

Ejemplo:

```
select ca.nombre_es "com. autónoma", p.nombre provincia, m.nombre municipio, pa.pob_m, pa.pob_f
from
comunidades_autonomas ca
join provincias p on (p.id_com_auto = ca.id)
join municipios m on (m.id_provincia = p.id)
join poblacion_anyo pa on (pa.id_municipio = m.id)
where pob_m < pob_f
and pob_m between 200 and 500
and (ca.nombre_es like '%da%' or p.nombre_es in ('Badajoz','Cuenca'))
and substr(p.nombre,1,1)>'B';
```

7. SELECT

Ordenación de resultados

Los resultados devueltos por una consulta no tienen por qué seguir el orden deseado.
Para que se muestren en el orden que necesitemos, utilizaremos la cláusula **ORDER BY**.

Tras ORDER BY indicaremos el criterio de ordenación (generalmente el valor de una columna o una operación sobre el mismo), seguido de **ASC** o **DESC** si queremos indicar específicamente que el orden sea ascendente (de menor a mayor) o descendente (de mayor a menor) respectivamente. Si no se incluye nada, por defecto es ascendente.

Para indicar la columna o criterio que queremos utilizar para la ordenación utilizaremos su nombre o cálculo completo o (y esto solo si aparece como resultado de la consulta) el número que representa el orden en el que aparece en la consulta.

Si queremos ordenar por más de un criterio, los separamos por comas, teniendo mayor peso mientras más cerca del ORDER BY se esté.

7. SELECT

Ejemplos

```
select p.nombre provincia, m.nombre municipio
from provincias p
join municipios m on (p.id = m.id_provincia)
where p.nombre in ('Huelva','Cádiz')
and length(m.nombre) < 6
order by length(m.nombre), m.nombre desc;
```

```
select p.nombre provincia, m.nombre municipio
from provincias p
join municipios m on (p.id = m.id_provincia)
where p.nombre in ('Sevilla','Cádiz')
and upper(m.nombre) like '%ILL%'
order by 1 asc, 2 desc;
```

7. SELECT

Limitación de resultados

Las consultas en BBDD devuelve el número de resultados completo según nuestros criterios de búsqueda (salvo que paginen estos resultados para no ofrecerlos todos de golpe).

Por rendimiento u otros motivos nos puede interesar limitar la búsqueda a los primeros resultados.

En estos casos, tenemos varias opciones:

En bases de datos como MariaDB, MySQL o PostgreSQL, podemos utilizar **LIMIT** n (siendo n un número), que limitará el resultado de la búsqueda a n resultados (como mucho, si los hubiera).

En otras bases de datos que no admiten LIMIT (al no ser un estándar de SQL), como Oracle, podemos utilizar alternativas, como **ROWNUM** <= n, para quedarnos solo con lo n primeros resultados devueltos (si los hubiese).

7. SELECT

Ejemplo:

```
select p.nombre provincia, m.nombre municipio, (pa.pob_m+pa.pob_f) "población"
from provincias p
join municipios m on (p.id = m.id_provincia)
join poblacion_anyo pa ON (pa.id_municipio = m.id)
where p.nombre = 'Sevilla'
order by 3 desc
limit 5;
```

7. SELECT

Agrupación de resultados

Los resultados se pueden agrupar para obtener una serie de datos relevantes (suma de valores, número total de registros, media de valores, etc.).

Para agrupar resultados se utiliza la cláusula **GROUP BY**, que viene seguida de las columnas que se desean agrupar para obtener los resultados.

Funciones de Agregado	
Función	Descripción
AVG	Calcula la media de los valores de un campo
COUNT	Cuenta el número de registros
SUM	Suma todos los valores de un campo
MAX	Valor más alto de un campo
MIN	Valor más bajo de un campo

7. SELECT

Agrupación de resultados

Si tenemos esta tabla, por ejemplo:

id	marca	modelo	matricula	precio
1	KIA	Sportage	1234-BCB	16.500
2	FORD	Focus	4321-BMN	9.490
3	SEAT	Ibiza	9999-DDD	9.490
4	KIA	Optima	6547-CDC	14.499,95
5	KIA	Carnival	1111-CCC	13.500
6	FORD	Mondeo	3522-DMN	10.320,25

```
select marca, avg(precio) precio_medio, max(precio) precio_maximo, min(precio) precio_minimo,
            count(1) coches_registrados, sum(precio) suma_precios
            from coches
            group by marca;
```

	marca	precio_medio	precio_maximo	precio_minimo	coches_registrados	suma_precios
1	FORD	9.905,125	10.320,25	9.490	2	19.810,25
2	KIA	14.833,316667	16.500	13.500	3	44.499,95
3	SEAT	9.490	9.490	9.490	1	9.490

7. SELECT

Si queremos que solo se muestren los resultados de una agrupación que cumplan alguna condición, se puede añadir la cláusula **HAVING** (ligada a GROUP BY), para conseguir esto:

```
select marca, avg(precio) precio_medio, max(precio) precio_maximo, min(precio) precio_minimo,
            count(1) coches_registrados, sum(precio) suma_precios
            from coches
            group by marca
            having avg(precio) > 9500;
```

	marca	precio_medio	precio_maximo	precio_minimo	coches_registrados	suma_precios
1	FORD	9.905,125	10.320,25	9.490	2	19.810,25
2	KIA	14.833,316667	16.500	13.500	3	44.499,95

8. Funciones de uso común

Hay una serie de funciones predefinidas que pueden utilizarse sobre cadenas, números y fechas. Se verán a continuación las más comunes.

Funciones para cadenas

LENGTH y **CHAR_LENGTH**: Ambas devuelven la longitud de una cadena. LENGTH devuelve la longitud en bytes y CHAR_LENGTH, en caracteres.

```
select marca, length(marca), char_length(marca) from
coches
group by marca;
```

	marca	leng	char_len
1	FORD	4	4
2	KIA	3	3
3	SEAT	4	4

```
select 'Día'  palabra, length('Día') l_bytes, char_length('Día') l_chars
union
select 'compañía', length('compañía'), char_length('compañía')
union
select 'ポケモン', length('ポケモン'), char_length('ポケモン');
```

	palabra	l_bytes	l_chars
	Día	4	3
	compañía	10	8
	ポケモン	12	4

8. Funciones de uso común

Funciones para cadenas

SUBSTR (o **SUBSTRING**): Sirve para extraer una subcadena de texto de una cadena existente. Substr(cadena,n,m) comienza en el carácter n desde la izquierda (o n desde la derecha si es negativo, y recupera hasta m caracteres), siempre que n esté en el rango de posiciones de la cadena de entrada (empezando en 1 o -1).
LEFT: Recupera los primeros n caracteres (desde la izquierda).
RIGHT: Recupera los últimos n caracteres (desde la derecha).

```
select substr('Saludo',1,100) s1, substr('Bravo',3,2) s2,
substr('Cadena',100,-1) s3, substr('Saludo',-3,12) s4,
left('Super Saiyajin',5) s5,right('Super Saiyajin',5) s6;
```

	A-Z s1	A-Z s2	A-Z s3	A-Z s4	A-Z s5	A-Z s6
1	Saludo	av		udo	Super	yajin

8. Funciones de uso común

Funciones para cadenas

UPPER: pasa a mayúsculas
LOWER: pasa a minúsculas
CONCAT: concatena N cadenas de texto.

```
select saludo, pokemon, lower(concat('A','E','I','O','U')) vocales,
     concat(    upper(substr(pokemon,1,1)),
                lower(substr(pokemon,2,char_length(pokemon)))) pokemon_ini_may
     from (
                select upper('Hola') saludo, lower('CHARIZARD') pokemon
     )aux;
```

saludo	pokemon	vocales	pokemon_ini_may
HOLA	charizard	aeiou	Charizard

8. Funciones de uso común

Funciones para cadenas

TRIM: elimina espacios al principio y final de la cadena
LTRIM: elimina espacios al principio (izquierda, ***L****eft*) de la cadena
RTRIM: elimina espacios al final (derecha, ***R****ight*) de la cadena

```
select cadena, concat('0',rtrim(cadena),'0') rtrimCad,
     concat('0',ltrim(cadena),'0') ltrimCad,
     concat('0',trim(cadena),'0') trimCad
from (
     select '      Cadena      ' cadena
)a1;
```

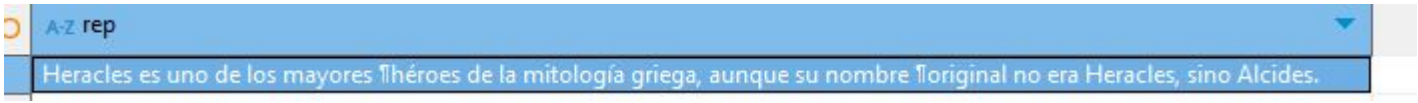
cadena	rtrimCad	ltrimCad	trimCad
Cadena	0 Cadena0	0Cadena 0	0Cadena0

8. Funciones de uso común

Funciones para cadenas

REPLACE: Sustituye en una cadena de entrada todas las ocurrencias de una secuencia de caracteres por la secuencia indicada.

```
select replace('Hércules es uno de los mayores
héroes de la mitología griega, aunque su nombre
original no era Hércules, sino Alcides.' ,
'Hércules', 'Heracles') rep;
```



8. Funciones de uso común

Funciones matemáticas

ROUND: Redondea un número a los decimales indicados.
CEIL: Techo de un número (redondeo entero al alza)
FLOOR: Suelo de un número (redondeo entero a la baja)
ABS: Valor absoluto de un número
POWER: Eleva un número a un exponente.

```
select 9/7, round(9/7,2), round(9/7), floor(9/7),
ceil(9/7), abs(2), abs(-2), power(2,4);
```

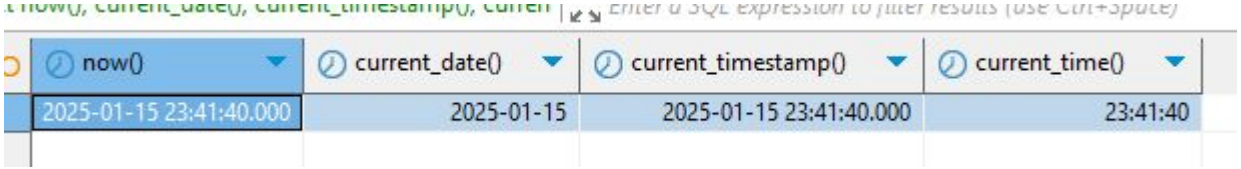
123 9/7	123 round(9/7,2)	123 round(9/7)	123 floor(9/7)	123 ceil(9/7)	123 abs(2)	123 abs(-2)	123 power(2,4)
1,2857	1,29	1	1	2	2	2	16

8. Funciones de uso común

Funciones de tiempo (fechas y horas)

NOW o **CURRENT_TIMESTAMP**: Devuelve fecha y hora actuales.
CURRENT_DATE: Devuelve la fecha actual.
CURRENT_TIME: Devuelve la hora actual.

```
select now(), current_date(),
current_timestamp(), current_time();
```

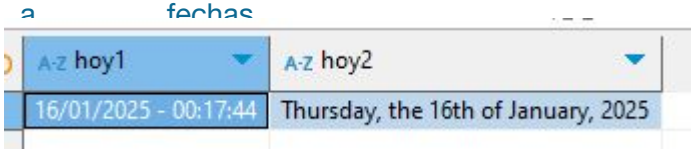


8. Funciones de uso común

Funciones de tiempo (fechas y horas)

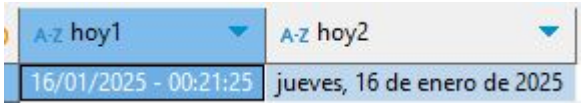
DATE_FORMAT: Para dar formato a fechas

```
select date_format(now(),'%d/%m/%Y - %H:%i:%s') hoy1,
date_format(now(),'%W, the %D of %M, %Y') hoy2;
```



NOTA: Los nombres de meses, días de la semana, etc. aparecen por defecto en formato inglés (salvo configurado de otra forma). Para que aparezca en formato español de España, podemos fijar los nombres a “es_ES” primero.

```
SET lc_time_names = 'es_ES';
select date_format(now(),'%d/%m/%Y - %H:%i:%s') hoy1,
date_format(now(),'%W, %d de %M de %Y') hoy2;
```



8. Funciones de uso común

Funciones de tiempo (fechas y horas)

Expresiones para los formatos de fecha

%Y : Año como valor numérico de 4 dígitos
%y : Año como valor numérico de 2 dígitos
%a : Nombre del día de la semana abreviado (de domingo a sábado)
%b : Nombre del mes abreviado (enero a diciembre)
%c : Mes como valor numérico (0 a 12)
%D: Día del mes como valor numérico, seguido de un sufijo (1º, 2º, 3º, ...)
%d : Día del mes como valor numérico (01 a 31)
%e : Día del mes como valor numérico (0 a 31)
%f : Microsegundos (000000 a 999999)
%H : Hora (00 a 23)
%h : Hora (00 a 12)
%I : Hora (00 a 12)
%i : Minutos (00 a 59)
%j : Día del año (001 a 366)
%k : Hora (00 a 23)
%l : Hora (1 a 12)
%M : Nombre completo del mes (enero a diciembre)
%m : Nombre del mes como valor numérico (00 a 12)
%p : AM o PM
%r : Hora en formato de 12 horas AM o PM (hh:mm:ss AM/PM)
%S : Segundos (00 a 59)
%s : Segundos (00 a 59)
%T : Hora en formato de 24 horas (hh:mm:ss)
%U : Semana donde el domingo es el primer día de la semana (00 a 53)
%u : Semana donde el lunes es el primer día de la semana (00 a 53)
%V : Semana donde el domingo es el primer día de la semana (01 a 53)
%v : Semana donde el lunes es el primer día de la semana (01 a 53)
%W : Nombre completo del día de la semana (de domingo a sábado)
%w : Día de la semana donde domingo=0 y sábado=6
%X : Año de la semana donde el domingo es el primer día de la semana
%x : Año de la semana donde el lunes es el primer día de la semana.

8. Funciones de uso común

Funciones de tiempo (fechas y horas)

EXTRACT (o **DATEPART** en otras BBDD): Extrae partes específicas de una fecha.

```
select now() ahora, extract(month from now()) mes,
extract(year from now()) año, extract(week from now()) semana,
extract (hour from now()) hora;
```

ahora	123 mes	123 año	123 semana	123 hora
2025-01-16 00:31:47.000	1	2.025	2	0

Posibles partes para extraer:

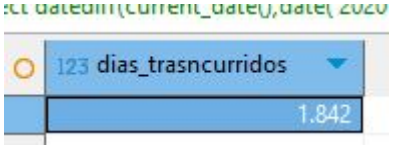
- MICROSECOND
- SECOND
- MINUTE
- HOUR
- DAY
- WEEK
- MONTH
- QUARTER
- YEAR
- SECOND_MICROSECOND
- MINUTE_MICROSECOND
- MINUTE_SECOND
- HOUR_MICROSECOND
- HOUR_SECOND
- HOUR_MINUTE
- DAY_MICROSECOND
- DAY_SECOND
- DAY_MINUTE
- DAY_HOUR
- YEAR_MONTH

8. Funciones de uso común

Funciones de tiempo (fechas y horas)

DATEDIFF: Calcula la diferencia entre dos fechas

```
select datediff(current_date(),date('2020-01-01')) dias_trasncurridos;
```



8. Funciones de uso común

Funciones de tiempo (fechas y horas)

ADDDATE o **DATE_ADD**: Añade o resta a una fecha una cantidad de tiempo determinada:

```
select current_date hoy, now() ahora,
adddate(current_date(), interval 1 week) en_una_semana,
date_add(current_date(), interval -100 day) hace_100_dias,
adddate(now(), interval 130 minute) en_130_minutos,
date_add(adddate(current_date(), interval -1000 year), interval -1 month) hace_1_milenio_y_1_mes;
```

hoy	ahora	en_una_semana	hace_100_dias	en_130_minutos	hace_1_milenio_y_1_mes
2025-01-16	2025-01-16 01:00:38.000	2025-01-23	2024-10-08	2025-01-16 03:10:38.000	1024-12-16

8. Funciones de uso común

Funciones condicionales

CASE: Estructura condicional

```
select animal,
case when animal in ('perro','gato') then 'mamífero'
       when animal in ('serpiente','tortuga') then 'reptil'
       when animal in ('canario','gorrión') then 'ave'
       when animal in ('rana', 'sapo') then 'anfibio'
       when animal in ('atún','mero') then 'pez'
       else 'no catalogado'
end categoria_vertebrados
from (
       select 'perro' animal union select 'gato' animal
       union select 'serpiente' animal union
       select 'gorrión' animal union
       select 'rana' animal union
       select 'rinoceronte' animal
)a1;
```

animal	categoria_vertebrados
perro	mamífero
gato	mamífero
serpiente	reptil
gorrión	ave
rana	anfibio
rinoceronte	no catalogado

9. JOIN

La cláusula JOIN en SQL sirve para combinar datos de dos o más tablas. En la mayoría de tipos de JOIN esta unión se realiza a través de uno o más campos en común entre ellas.

Generalmente suele utilizarse la equivalencia entre la primary key de una tabla con la foreign key de otra tabla que referencia a la primera tabla, tal que:

```
tabla1.primary_key = tabla2.foreign_key_a_tabla1
```

Si la primary key (y la foreign key) estuviera conformada por más de un campo, podría relacionarse así:

```
(tabla1.primary_keyc1 = tabla2.foreign_keyc1_a_tabla1
and tabla1.primary_keyc2 = tabla2.foreign_keyc2_a_tabla1
...
and tabla1.primary_keycn = tabla2.foreign_keycn_a_tabla1)
```

9. JOIN

Ejemplo:

Vehículos

id	marca	modelo	matrícula
1	KIA	Sportage	1234-AAA
2	FORD	Focus	4321-BBB
3	SEAT	Ibiza	9999-DDD
4	RENAULT	Clio	3535-FDG
5	PEUGEOT	206	4487-FPR
6	CITROEN	Xceed	7787-KJL

Propietarios

id	Nombre	DNI	id_vehiculo
1	Fran	12345R	2
2	Natalia	45245T	5
3	Clara	78445P	3
4	Antonio	14552B	NULL

9. JOIN

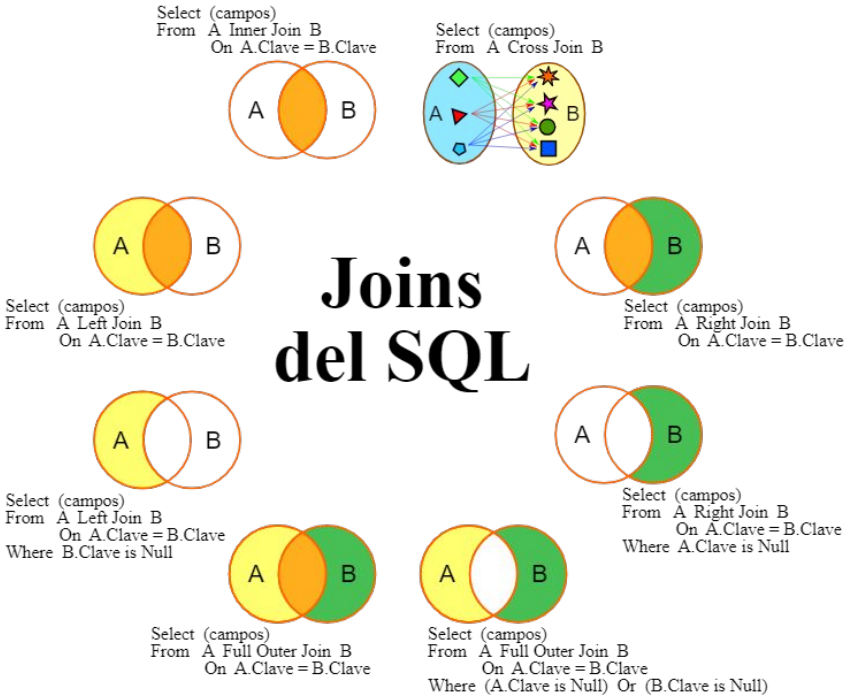
Ejemplo de JOIN

```
select  c.id , c.marca , c.modelo , p.nombre,
from  coche  c
join  propietario p on c.id = p.id_vehiculo;
```

c.id	c.marca	c.modelo	p.nombre
2	FORD	Focus	Fran
3	SEAT	Ibiza	Clara
5	PEUGEOT	206	Natalia

9. JOIN

Hay distintos tipos de JOIN, según el cruce que queramos hacer.
A destacar: INNER JOIN, LEFT (OUTER) JOIN, RIGHT (OUTER) JOIN, FULL (OUTER) JOIN y CROSS JOIN.



9. JOIN

INNER JOIN (o simplemente **JOIN**)

Devuelve solo las filas que coinciden en las dos tablas que une el join. Su sintaxis es INNER JOIN o, simplemente, JOIN:

	id	marca	modelo	matricula
1	1	KIA	Sportage	1234BBB
2	2	FORD	Focus	4321BBB
3	3	SEAT	Ibiza	9999DDD
4	4	RENAULT	Clio	3535FDG
5	5	PEUGEOT	206	4487FPR
6	6	CITROEN	Xceed	7787KJL

propietarios | Enter a SQL expression to filter results (use Ctrl+Space)

Grilla / Texto

	id	nombre	dni	id_vehiculo
1	1	Fran	53975551W	2
2	2	Natalia	11355987J	5
3	3	Clara	76451352L	3
4	4	Antonio	36562841V	[NULL]

```
select v.*, p.nombre
from vehiculos v
join propietarios p on (v.id = p.id_vehiculo);
```

select v.*, p.nombre from vehiculos v join propietarios p on (v.id = p | Enter a SQL expression to filter resul

Grilla / Texto

	id	marca	modelo	matricula	nombre
1	2	FORD	Focus	4321BBB	Fran
2	5	PEUGEOT	206	4487FPR	Natalia
3	3	SEAT	Ibiza	9999DDD	Clara

9. JOIN

LEFT JOIN (o **LEFT OUTER JOIN**)

Devuelve todas las filas de la tabla de la izquierda y las filas coincidentes de la tabla de la derecha. Si no hay coincidencia, los valores de las columnas de la tabla derecha serán NULL.

	id	marca	modelo	matricula
1	1	KIA	Sportage	1234BBB
2	2	FORD	Focus	4321BBB
3	3	SEAT	Ibiza	9999DDD
4	4	RENAULT	Clio	3535FDG
5	5	PEUGEOT	206	4487FPR
6	6	CITROEN	Xceed	7787KJL

propietarios | Enter a SQL expression to filter results (use Ctrl+Space)

Grilla | Texto

	id	nombre	dni	id_vehiculo
1	1	Fran	53975551W	2
2	2	Natalia	11355987J	5
3	3	Clara	76451352L	3
4	4	Antonio	36562841V	[NULL]

```
select v.*, p.nombre
from vehiculos v
left join propietarios p on (v.id = p.id_vehiculo);
```

select v.*, p.nombre from vehiculos v left join propietarios p on (v.id | Enter a SQL expression to filter results

Grilla | Texto | Record

	id	marca	modelo	matricula	nombre
1	1	KIA	Sportage	1234BBB	[NULL]
2	2	FORD	Focus	4321BBB	Fran
3	3	SEAT	Ibiza	9999DDD	Clara
4	4	RENAULT	Clio	3535FDG	[NULL]
5	5	PEUGEOT	206	4487FPR	Natalia
6	6	CITROEN	Xceed	7787KJL	[NULL]

9. JOIN

RIGHT JOIN (o **RIGHT OUTER JOIN**)

Es lo contrario del LEFT JOIN. Devuelve todas las filas de la tabla de la derecha y las filas coincidentes de la tabla de la izquierda. Si no hay coincidencia, los valores de las columnas de la tabla izquierda serán NULL.

	id	marca	modelo	matricula
1	1	KIA	Sportage	1234BBB
2	2	FORD	Focus	4321BBB
3	3	SEAT	Ibiza	9999DDD
4	4	RENAULT	Clio	3535FDG
5	5	PEUGEOT	206	4487FPR
6	6	CITROEN	Xceed	7787KJL

propietarios | Enter a SQL expression to filter results (use Ctrl+Space)

Grilla | Texto

	id	nombre	dni	id_vehiculo
1	1	Fran	53975551W	2
2	2	Natalia	11355987J	5
3	3	Clara	76451352L	3
4	4	Antonio	36562841V	[NULL]

```
select v.*, p.nombre
from vehiculos v
right join propietarios p on (v.id = p.id_vehiculo);
```

select v.*, p.nombre from vehiculos v right join propietarios p on (v.i | Enter a SQL expression to filte

	id	marca	modelo	matricula	nombre
1	2	FORD	Focus	4321BBB	Fran
2	5	PEUGEOT	206	4487FPR	Natalia
3	3	SEAT	Ibiza	9999DDD	Clara
4	[NULL]	[NULL]	[NULL]	[NULL]	Antonio

9. JOIN

RIGHT JOIN (o **RIGHT OUTER JOIN**)

Un RIGHT JOIN puede expresarse como un LEFT JOIN cambiando el orden de las tablas

	id	marca	modelo	matricula
1	1	KIA	Sportage	1234BBB
2	2	FORD	Focus	4321BBB
3	3	SEAT	Ibiza	9999DDD
4	4	RENAULT	Clio	3535FDG
5	5	PEUGEOT	206	4487FPR
6	6	CITROEN	Xceed	7787KJL

propietarios | Enter a SQL expression to filter results (use Ctrl+Space)

Grilla | Texto

	id	nombre	dni	id_vehiculo
1	1	Fran	53975551W	2
2	2	Natalia	11355987J	5
3	3	Clara	76451352L	3
4	4	Antonio	36562841V	[NULL]

```
select v.*, p.nombre
from vehiculos v
right join propietarios p on (v.id = p.id_vehiculo);
```

```
select v.*, p.nombre
from propietarios p
left join vehiculos v on (v.id = p.id_vehiculo);
```

select v.*, p.nombre from vehiculos v right join propietarios p on (v.i | Enter a SQL expression to filte

	id	marca	modelo	matricula	nombre
1	2	FORD	Focus	4321BBB	Fran
2	5	PEUGEOT	206	4487FPR	Natalia
3	3	SEAT	Ibiza	9999DDD	Clara
4	[NULL]	[NULL]	[NULL]	[NULL]	Antonio

9. JOIN

FULL JOIN (o **FULL OUTER JOIN**)

Devuelve todas las filas cuando hay una coincidencia en una de las tablas. Si no hay coincidencia, la parte que no tenga coincidencia será NULL.

NOTA: En MariaDB, FULL JOIN no es soportado de forma nativa, pero se puede lograr un comportamiento similar combinando LEFT JOIN y RIGHT JOIN con una unión (como UNION).

	id	marca	modelo	matricula
1	1	KIA	Sportage	1234BBB
2	2	FORD	Focus	4321BBB
3	3	SEAT	Ibiza	9999DDD
4	4	RENAULT	Clio	3535FDG
5	5	PEUGEOT	206	4487FPR
6	6	CITROEN	Xceed	7787KJL

propietarios | Enter a SQL expression to filter results (use Ctrl+Space)

Grilla | Texto

	id	nombre	dni	id_vehiculo
1	1	Fran	53975551W	2
2	2	Natalia	11355987J	5
3	3	Clara	76451352L	3
4	4	Antonio	36562841V	[NULL]

```
select v.*, p.nombre
from propietarios p
full join vehiculos v on (v.id = p.id_vehiculo);
```

vehiculos(+) 1

```
select v.*, p.nombre from propietarios p left join vehiculos v on (v.id
```

Enter a SQL expression to filter results (use Ctrl+Space)

SQL Error [1064] [42000]: (conn=13) You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'full join vehiculos v on (v.id = p.id_vehiculo)
LIMIT 0, 200' at line 3

9. JOIN

FULL JOIN (o **FULL OUTER JOIN**)

NOTA: En MariaDB, FULL JOIN no es soportado de forma nativa, pero se puede lograr un comportamiento similar combinando LEFT JOIN y RIGHT JOIN con una unión (como UNION).

	id	marca	modelo	matricula
1	1	KIA	Sportage	1234BBB
2	2	FORD	Focus	4321BBB
3	3	SEAT	Ibiza	9999DDD
4	4	RENAULT	Clio	3535FDG
5	5	PEUGEOT	206	4487FPR
6	6	CITROEN	Xceed	7787KJL

propietarios | Enter a SQL expression to filter results (use Ctrl+Space)

Grilla | Texto

	id	nombre	dni	id_vehiculo
1	1	Fran	53975551W	2
2	2	Natalia	11355987J	5
3	3	Clara	76451352L	3
4	4	Antonio	36562841V	[NULL]

```
select v.*, p.nombre
from propietarios p
left join vehiculos v on (v.id = p.id_vehiculo)
union
select v.*, p.nombre
from propietarios p
right join vehiculos v on (v.id = p.id_vehiculo);
```

select v.*, p.nombre from propietarios p left join vehiculos v on (v.id | Enter a SQL expression to filter results (use

Grilla | Texto | Record

	id	marca	modelo	matricula	nombre
1	2	FORD	Focus	4321BBB	Fran
2	5	PEUGEOT	206	4487FPR	Natalia
3	3	SEAT	Ibiza	9999DDD	Clara
4	[NULL]	[NULL]	[NULL]	[NULL]	Antonio
5	1	KIA	Sportage	1234BBB	[NULL]
6	4	RENAULT	Clio	3535FDG	[NULL]
7	6	CITROEN	Xceed	7787KJL	[NULL]

9. JOIN

CROSS JOIN

Devuelve el producto cartesiano de las dos tablas, es decir, combina cada fila de la tabla de la izquierda con cada fila de la tabla de la derecha. Este tipo de join puede generar un gran número de resultados, por lo que se usa con precaución.

	id	marca	modelo	matricula
1	1	KIA	Sportage	1234BBB
2	2	FORD	Focus	4321BBB
3	3	SEAT	Ibiza	9999DDD
4	4	RENAULT	Clio	3535FDG
5	5	PEUGEOT	206	4487FPR
6	6	CITROEN	Xceed	7787KJL

propietarios | Enter a SQL expression to filter results (use Ctrl+Space)

Grilla | Texto

	id	nombre	dni	id_vehiculo
1	1	Fran	53975551W	2
2	2	Natalia	11355987J	5
3	3	Clara	76451352L	3
4	4	Antonio	36562841V	[NULL]

```
select v.marca, p.nombre
from propietarios p
cross join vehiculos v;
```

vehiculos(+) 1

```
select v.marca, p.nombre from propietarios p cr
```

Grilla | Texto | Record

	marca	nombre
1	KIA	Fran
2	KIA	Natalia
3	KIA	Clara
4	KIA	Antonio
5	FORD	Fran
6	FORD	Natalia
7	FORD	Clara
8	FORD	Antonio
9	SEAT	Fran
10	SEAT	Natalia
11	SEAT	Clara
12	SEAT	Antonio
13	RENAULT	Fran
14	RENAULT	Natalia
15	RENAULT	Clara
16	RENAULT	Antonio
17	PEUGEOT	Fran
18	PEUGEOT	Natalia
19	PEUGEOT	Clara
20	PEUGEOT	Antonio
21	CITROEN	Fran
22	CITROEN	Natalia
23	CITROEN	Clara
24	CITROEN	Antonio