

Bases de Datos

Unidad 5: Realización de consultas avanzadas Sesión 1

Hasta ahora únicamente hemos visto consultas realizadas sobre una única tabla.

Es muy frecuente y necesario tener que realizar consultas sobre **combinaciones de tablas** dado que necesitamos obtener datos en la consulta de varias tablas y/o tener condiciones aplicadas a datos de varias tablas.

Por ejemplo, en la base de datos de alquileres, para obtener el nombre y apellidos de los clientes que han alquilado coches en enero, necesitamos usar o combinar dos tablas en la consulta: **clientes y contratos**.

En MySQL podemos usar las siguientes operaciones de combinación de tablas:

- ☐ Producto cartesiano o CROSS JOIN
- ☐ Combinación INNER JOIN
- ☐ Combinación LEFT JOIN
- ☐ Combinación RIGHT JOIN

Combinación de tablas

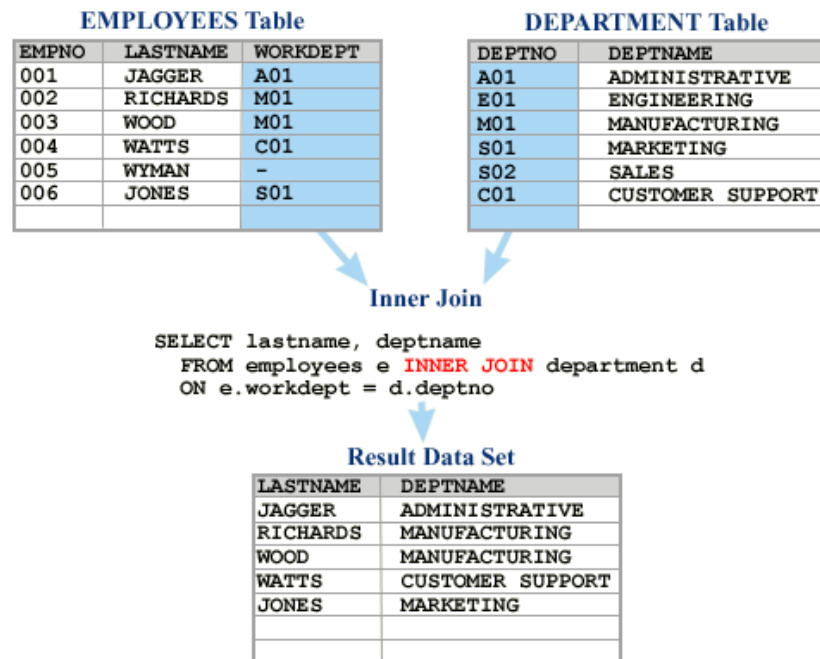
La reunión interna. INNER JOIN

Reunión Interna. INNER JOIN

Permite emparejar filas de dos tablas a través de una relación entre una columna de una tabla y otra columna de otra tabla.

Lo normal es que sean la clave principal de una tabla y la correspondiente clave ajena relacionada en la otra tabla, aunque pueden ser columnas que no tienen relación de clave ajena establecida.

En una consulta de este tipo, para cada fila de una de las tablas se busca en la otra tabla la fila o filas que cumplen la condición de relación que se quiera entre las dos columnas (normalmente se busca igualdad entre clave principal y clave ajena).



Reunión Interna. INNER JOIN

La sintaxis de esta operación dentro de una SELECT es:

```
SELECT ..... FROM tabla1 INNER JOIN tabla2 ON columna1  
condicion_relacion columna2
```

Tabla 1 y tabla 2 podrían ser incluso la misma tabla si hay alguna relación entre una columna de la tabla y la clave principal de la misma tabla. En este caso, al menos uno de los nombres de tabla tendría que ser un **alias**.

¿Por qué una de esos nombres debería ser un alias?

Columna1 y columna2 son las columnas que se emparejan o relacionan y deben tener el mismo tipo de datos o datos compatibles.

Condicion_relacion representa cualquier operación relacional, aunque normalmente se usa la igualdad. Se pueden combinar más de dos tablas usando varios INNER JOIN.

Reunión Interna. INNER JOIN

Sintaxis:

```
SELECT ..... FROM tabla1 INNER JOIN tabla2 ON columna1  
condicion_relacion columna2
```

Cuando coincida el nombre de las dos columnas relacionadas, tendremos que escribir nombres cualificados, escribiendo el nombre de la tabla a la que pertenecen, un punto y el nombre de la columna. En ese caso, también se puede usar y es más adecuada esta sintaxis:

```
SELECT ..... FROM tabla1 INNER JOIN tabla2 USING (columna);
```

Ejemplo: Obtener el número de contrato y la matrícula, marca y modelo de todos los automóviles que están contratados actualmente por algún cliente.

```
SELECT numcontrato, automoviles.matricula, marca, modelo FROM contratos  
INNER JOIN automoviles ON contratos.matricula=automoviles.matricula WHERE  
ffin IS NULL;
```

¿Por qué hemos escrito automoviles.matricula pero no automoviles.marca?

numcontrato	matricula	marca	modelo
18	2058JGF	Seat	Leon
19	3273JGH	Audi	A4
20	2123JTB	Renault	Megane
21	8795JTK	Mercedes	GLA

Reunión Interna. INNER JOIN

Ejemplo: Obtener el número de contrato y el nombre y apellidos de todos los clientes que tienen actualmente contrato algún automóvil.

SELECT numcontrato, nombre, apellidos **FROM** clientes **INNER JOIN** contratos **ON** dnicliente=dni **WHERE** ffin **IS NULL**;

¿Por qué ponemos **WHERE ffin IS NULL**?

numcontrato	nombre	apellidos
18	Fanny	Cepeda
19	Carlos Javier	Lopez Carvajal
20	Ismael	Poza Rincón
21	Alicia	de la Hoz Gomez

Reunión Interna. INNER JOIN

Ejemplo: De todos los contratos finalizados, obtener la matricula, marca y modelo de cada coche contratado, el nombre y apellidos del cliente que hizo cada contrato y los kilómetros recorridos por el coche en el contrato.

SELECT numcontrato, automoviles.matricula, marca, modelo, nombre, apellidos, kfin-kini **FROM** (contratos **INNER JOIN** automoviles **ON** contratos.matricula = automoviles.matricula) **INNER JOIN** clientes **ON** dnicliente=dni **WHERE** ffin **IS NOT NULL**;

numcontrato	matricula	marca	modelo	nombre	apellidos	kfin-kini
1	1234JMY	Mercedes	Clase C Coupe 170CV	Mariano	Dorado	361
2	7856JLD	BMW	318 TDI	Fanny	Cepeda	1000
3	5573JFS	Seat	Leon SW	Reyes	Sanz Lopez	2250
4	3273JGH	Audi	A4	Ismael	Poza Rincón	870
5	3765JSD	Seat	Ibiza	Ana Belén	Fuentes Rojas	833
6	1678JCN	Ford	Fiesta	Antonio	Díaz Vera	3328

Reunión Interna. INNER JOIN

Ejemplo: En una base de datos nba tenemos una tabla equipos. En la tabla equipos, entre otros datos, se tiene el nombre del equipo y la división en la que participa. Obtener todos los enfrentamientos o partidos posibles entre equipos de la división central sin usar la tabla partidos, buscando los distintos cruces.

```
SELECT a.nombre AS local, b.nombre AS visitante FROM equipos AS a INNER JOIN  
equipos AS b ON a.nombre <> b.nombre WHERE a.division='central' AND  
b.division='central';
```

local	visitante
Bulls	Bucks
Cavaliers	Bucks
Pacers	Bucks
Pistons	Bucks
Bucks	Bulls
Cavaliers	Bulls
Pacers	Bulls

Explicación: Tenemos una INNER JOIN entre dos tablas que son la misma. A la primera la renombramos como tabla a y la segunda como tabla b, pero las dos son equipos.

Ejemplo: Obtener el nombre y apellidos de los clientes que han contratado automóviles de la marca Seat.

```
SELECT DISTINCT nombre, apellidos FROM (contratos INNER  
JOIN automoviles ON contratos.matricula = automoviles.matricula) INNER JOIN  
clientes ON dncliente=dni WHERE marca='seat';
```

```
select nombre, apellidos from clientes inner join contratos as con on  
clientes.dni=con.dncliente inner join automoviles on  
con.matricula=automoviles.matricula where marca='seat';
```

nombre	apellidos
Anais	Rodriguez
Fanny	Cepeda
Ana Belén	Fuentes Roias
Reves	Sanz Lopez

*Si el uso de las operaciones de combinación de tablas permiten relacionar dos o más tablas por medio de diferentes **columnas independientemente de que estas sean claves primarias o foráneas...***

¿Por qué al diseñar una base de datos e implementar su diseño físico utilizamos las claves foráneas?

Motivos:

- 1. Integridad referencial. Al estar relacionadas las tuplas de dos tablas por medio de una FK, garantizamos la integridad de los datos. Ya que en caso de actualización o borrado estos serán tratados en consecuencia por medio de las restricciones correspondientes.***
- 2. Diseño estructural. Permiten una mayor claridad en la definición de la estructura de la base de datos. Facilitan conocer mejor las relaciones entre las tablas y ayudan a la hora de componer las consultas.***
- 3. Rendimiento. La rapidez al realizar las consultas aumenta, ya que una FK genera automáticamente un índice, por lo que las consultas serán más rápidas.***

Bases de Datos

Unidad 5: Realización de consultas Sesión 2

Producto cartesiano

El producto cartesiano de dos tablas permite obtener una tabla con las columnas de la primera tabla y las columnas de la segunda tabla (aunque tengan nombres iguales).

Las filas de la hoja de resultados resultante son todas las posibles combinaciones entre filas de la primera tabla y filas de la segunda tabla. Así, si una tabla tiene 6 filas y la otra tiene 8, el resultado del producto cartesiano es una tabla de 48 filas. Pero si una tabla tiene 6000 filas y otras 8000, se crea en memoria una tabla de 48 millones de filas, cada una de las cuales contiene varios bytes. Eso supone crear mucho espacio en memoria y puede ser un GRAVE PROBLEMA.

idfab	idproducto	fab	producto
bic	41672	aci	41004
imm	779c	aci	41002

COMPOSICION

idfab	idproducto	fab	producto
bic	41672	aci	41002
bic	41672	aci	41004
imm	779c	aci	41002
imm	779c	aci	41004

Producto cartesiano

Para obtener el producto cartesiano total entre dos tablas se escribe * (todas las columnas) después de SELECT y los nombres de las dos tablas separadas con coma después de FROM. No es muy normal tener que obtener el producto cartesiano total entre dos o más tablas. Lo normal es que sobre el resultado de un producto cartesiano apliquemos condiciones para extraer los datos combinados que queremos.

Por ejemplo, si ejecutamos:

SELECT * FROM automoviles, contratos;

matricula	marca	modelo	color	precio	kilometros	extras	alquilado	numcontrato	matricula	dnicliente	fini	ffin	kini	kfin
1234JMY	Mercedes	Clase C Coupe 170CV	Negro	165.78	22561	AUT,TS	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
1678JCN	Ford	Fiesta	Verde	68.64	9500	AA,EE,CC,RC,ABS	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
1732JBS	Seat	Leon	Negro	90.06	2500	TS,SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
1978JNT	Opel	Corsa	Azul	42.7	45876		0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
2058JGF	Seat	Leon	Rojo	93.64	9736	GPS,SN	1	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
2123JTB	Renault	Megane	Amarillo	92.65	34323	TS,SN	1	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
3273JGH	Audi	A4	Rojo	124.2	17368	AUT,GPS,WIFI	1	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
3765JSD	Seat	Ibiza	Rojo	70.56	7683	SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
4387JDD	Citroen	C3	Verde	62.67	23057		0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
4738JBJ	Audi	A3	Amarillo	118.76	8008	GPS,WIFI,SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
5031JHL	BMW	318 i	Azul	116.45	24796	GPS,WIFI,SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
5573JFS	Seat	Leon SW	Gris	102.63	28500	AUT,GPS	1	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
5678JRZ	Mercedes	Clase C	Blanco	123.65	7659	AUT,GPS,SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
6761JYM	Renault	Clio	Blanco	53.62	25672	SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
7839JDR	Ford	Focus	Blanco	87.62	15873	GPS,SN	1	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
7856JLD	BMW	318 TDI	Azul	121.79	35978	TS,GPS,SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561

Producto cartesiano

matricula	marca	modelo	color	precio	kilometros	extras	alquilado	numcontrato	matricula	dncliente	fini	ffin	kini	kfin
1234JMY	Mercedes	Clase C Coupe 170CV	Negro	165.78	22561	AUT,TS	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
1678JCN	Ford	Fiesta	Verde	68.64	9500	AA,EE,CC,RC,ABS	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
1732JBS	Seat	Leon	Negro	90.06	2500	TS,SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
1978JNT	Opel	Corsa	Azul	42.7	45876		0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
2058JGF	Seat	Leon	Rojo	93.64	9736	GPS,SN	1	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
2123JTB	Renault	Megane	Amarillo	92.65	34323	TS,SN	1	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
3273JGH	Audi	A4	Rojo	124.2	17368	AUT,GPS,WIFI	1	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
3765JSD	Seat	Ibiza	Rojo	70.56	7683	SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
4387JDD	Citroen	C3	Verde	62.67	23057		0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
4738JBJ	Audi	A3	Amarillo	118.76	8008	GPS,WIFI,SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
5031JHL	BMW	318 i	Azul	116.45	24796	GPS,WIFI,SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
5573JFS	Seat	Leon SW	Gris	102.63	28500	AUT,GPS	1	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
5678JRZ	Mercedes	Clase C	Blanco	123.65	7659	AUT,GPS,SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
6761JYM	Renault	Clio	Blanco	53.62	25672	SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
7839JDR	Ford	Focus	Blanco	87.62	15873	GPS,SN	1	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
7856JLD	BMW	318 TDI	Azul	121.79	35978	TS,GPS,SN	0	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561

Vemos que en el resultado vemos múltiples filas para el contrato 1. Vemos que ese contrato corresponde a la matrícula 12434JMY. Pero ese contrato se ha combinado con cada uno de los automóviles, es decir, se ha combinado con automóviles de matrículas distintas a las del contrato. Lo lógico es que, para cada contrato, relacionemos el contrato con los datos del automóvil que corresponda. Eso hay que hacerlo con una condición WHERE.

SELECT * FROM automoviles, contratos WHERE automoviles.matricula = contratos.matricula;

Aunque ahora veamos en el resultado una fila por cada contrato, realmente se ha creado en memoria una tabla intermedia con el producto cartesiano completo. 17

Producto cartesiano

Vamos a ver varios ejemplos en los que se realizan consultas en las que se puede usar el producto cartesiano. Se aplica la combinación producto cartesiano y se establecen condiciones de selección WHERE y se indican las columnas o expresiones que se quieren consultar.

Ejemplo: Obtener la matricula, marca, modelo, fecha inicial y fecha final del contrato número 1 (habrá que establecer la condición WHERE para que el contrato sea el número 1 y para que la matrícula del automóvil coincida con la del contrato).

```
SELECT contratos.matricula, marca, modelo, fini, ffin FROM automoviles,  
contratos WHERE numcontrato=1 AND  
contratos.matricula=automoviles.matricula;
```

matricula	marca	modelo	fini	ffin
1234JMY	Mercedes	Clase C Coupe 170CV	2016-12-12	2016-12-19

Vemos que en la instrucción se deben usar **cualificadores de tabla** en columnas que tienen el mismo nombre en las dos tablas.

Producto cartesiano

Cuando se combinan tablas puede ser útil a veces renombrarlas. Por ejemplo, la instrucción usada en la anterior diapositiva:

```
SELECT contratos.matricula, marca, modelo, fini, ffin FROM automoviles,  
contratos WHERE numcontrato=1 AND  
contratos.matricula=automoviles.matricula;
```

Sería equivalente a esta que usa renombrado de tablas:

```
SELECT c.matricula, marca, modelo, fini, ffin FROM automoviles AS a, contratos AS  
c WHERE numcontrato=1 AND c.matricula=a.matricula;
```

Producto cartesiano

Ejemplo: Suponiendo que tenemos en una base de datos una tabla con los módulos de un curso y otra tabla con los alumnos del curso, realizar una consulta que obtiene todas las posibles combinaciones de códigos o números de alumnos con todos los códigos de módulos del curso DAM1.

numalumn	nombre	apellidos	direccion	dni
001	Ana	Alonso	Alonso	11111111A
002	Beatriz	Bueno	Bueno	22222222B
003	Carlos	Campos	Campos	33333333C

codmodulo	nommodulo	horassemana	transversal
BD	Bases de ...	6	0
ED	Entornos ...	2	0
FOL	Formación...	3	1
LMSG	Lenguajes...	4	0
PRG	Programac...	8	0
SI	Sistemas I...	7	0

SELECT numalumn, codmodulo FROM alumnos, modulos ORDER BY codmodulo;

numalumn	codmodulo
001	BD
002	BD
003	BD
001	ED
002	ED
003	ED
001	FOL
002	FOL
003	FOL
001	LMSG

Producto cartesiano

Ejemplo: En la base de datos alquileres obtener la marca y modelo (sin repetir) de todos los automóviles contratados alguna vez en diciembre de 2017.

SELECT DISTINCT marca, modelo FROM automoviles, contratos WHERE automoviles.matricula=contratos.matricula AND fini LIKE '2017-12%';

marca	modelo
Mercedes	Clase C Coupe 170CV
BMW	318 TDI
Seat	Leon SW
Audi	A4
Seat	Ibiza
Ford	Fiesta
Renault	Clio
Opel	Corsa
Seat	Leon

Producto cartesiano

Ejemplo: Del contrato de alquiler de coches número 10, obtener el cliente que hizo el contrato, la matrícula, marca y modelo del coche y la duración del contrato.

SELECT apellidos, nombre, contratos.matricula, marca, modelo, fini, ffin **FROM** automoviles, contratos, clientes **WHERE** automoviles.matricula = contratos.matricula **AND** contratos.dnicliente=clientes.dni **AND** numcontrato=10;

apellidos	nombre	matricula	marca	modelo	fini	ffin
de la Hoz Gomez	Alicia	3273JGH	Audi	A4	2016-12-27	2017-01-02

Producto cartesiano

Ejemplo: En una base de datos nba tenemos una tabla equipos. En la tabla equipos, entre otros datos, se tiene el nombre del equipo y la división en la que participa. Obtener todos los enfrentamientos o partidos posibles entre equipos de la división central. Habrá que combinar la tabla equipos consigo misma evitando que el equipo local y el visitante sea el mismo.

SELECT a.nombre AS local, b.nombre AS visitante FROM equipos AS a, equipos AS b WHERE a.division='central' AND b.division='central' AND a.nombre <> b.nombre;

local	visitante
Bulls	Bucks
Cavaliers	Bucks
Pacers	Bucks
Pistons	Bucks
Bucks	Bulls
Cavaliers	Bulls
Pacers	Bulls
Pistons	Bulls
Bucks	Cavaliers
Bulls	Cavaliers

IMPORTANTE:

El producto cartesiano debe evitarse, siempre y cuando la consulta se pueda realizar con otra operación de combinación, cuando las tablas que se combinan tienen muchas filas.

La combinación produce el producto del número de filas combinadas y eso pueden ser muchísimas filas (y muchas columnas también). Todo eso se almacena temporalmente en RAM y ocupa mucho espacio.

Las consultas de los ejemplos de las diapositivas 7 y 10 son casos de una buena utilización del producto cartesiano ya que ahí si que queremos combinar todas las filas de una tabla con todas las de la otra.

Sin embargo, los otros ejemplos de este apartado se podrían realizar más óptimamente con otras operaciones de combinación.

Bases de Datos

Unidad 5: Realización de consultas Sesión 3

Combinación de tablas

Las reuniones externas.

LEFT JOIN

RIGHT JOIN

Reunión externa por la izquierda. LEFT JOIN

Permite emparejar filas de dos tablas a través de una relación entre una columna de una tabla y otra columna de otra tabla. Hasta aquí todo igual que INNER JOIN.

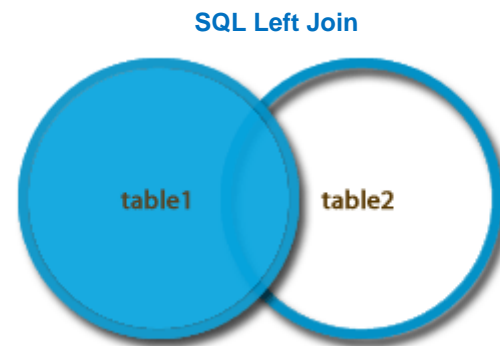
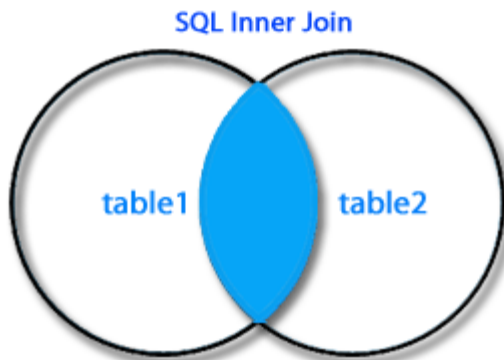
Además añade combinaciones de filas de la tabla de la izquierda con columnas vacías de la tabla de la derecha o a valores nulos para aquellas filas de la tabla de la izquierda que no tienen correspondencia con filas de la tabla de la derecha.

Por ejemplo, si se hace un **AUTOMOVILES LEFT JOIN CONTRATOS ON automoviles.matricula = contratos.matricula**

Para los automóviles que nunca han sido contratados, se generaría una fila con los datos del automóvil y todos los datos de contrato a valor NULL.

Sintaxis:

La misma que para INNER JOIN. Se pueden usar las cláusulas ON y USING.



Reunión externa por la izquierda. LEFT JOIN

*Ejemplo: Obtener la matrícula, marca y modelo de todos los automóviles junto con los datos de todos los contratos que se han realizado sobre esos automóviles. **Para los automóviles nunca contratados se debe obtener también una fila que no está relacionada con ningún contrato.***

SELECT automoviles.matricula, marca, modelo, contratos.* **FROM** automoviles **LEFT JOIN** contratos **USING** (matricula);

matricula	marca	modelo	numcontrato	matricula	dnidiente	fini	ffin	kini	kfin
1234JMY	Mercedes	Clase C Coupe 170CV	1	1234JMY	08785691K	2016-12-12	2016-12-19	22200	22561
1678JCN	Ford	Fiesta	6	1678JCN	24589635S	2016-12-22	2016-12-30	32650	35978
1732JBS	Seat	Leon	NULL	NULL	NULL	NULL	NULL	NULL	NULL
1978JNT	Opel	Corsa	8	1978JNT	13876715C	2016-12-25	2016-12-26	45650	45876
2058JGF	Seat	Leon	9	2058JGF	09856064L	2016-12-27	2016-12-30	8150	9736
2058JGF	Seat	Leon	18	2058JGf	07385709H	2017-01-08	NULL	9736	NULL
2123JTB	Renault	Megane	20	2123JTB	03549358G	2017-01-09	NULL	34323	NULL

Reunión externa por la izquierda. LEFT JOIN

Ejemplo: Obtener los datos de todos los automóviles que nunca han sido contratados.

SELECT automoviles.* FROM automoviles LEFT JOIN contratos USING (matricula) WHERE numcontrato IS NULL;

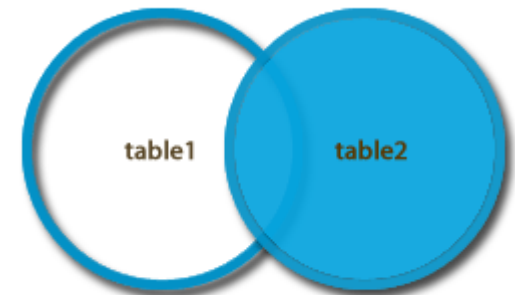
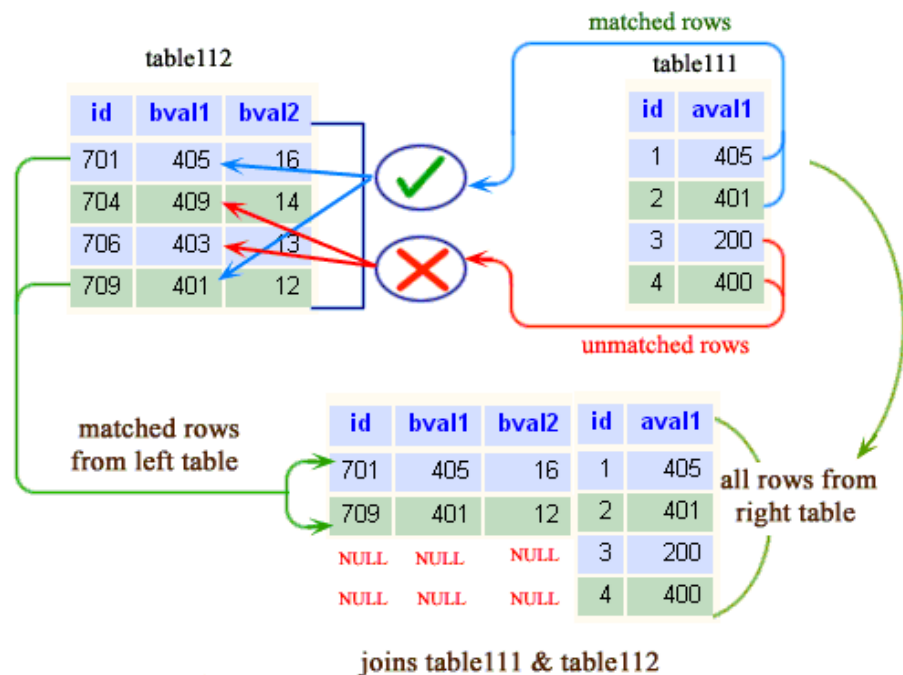
matricula	marca	modelo	color	precio	kilometros	extras	alquilado
1732JBS	Seat	Leon	Negro	90.06	2500	TS,SN	0
7839JDR	Ford	Focus	Blanco	87.62	15873	GPS,SN	1

Reunión externa por la derecha. RIGHT JOIN

Permite emparejar filas de dos tablas a través de una relación entre una columna de una tabla y otra columna de otra tabla. Hasta aquí todo igual que INNER JOIN.

Además añade combinaciones de filas de la tabla de la derecha con columnas vacías de la tabla de la izquierda o a valores nulos para aquellas filas de la tabla de la derecha que no tienen correspondencia con filas de la tabla de la izquierda.

MySQL RIGHT JOIN



Reunión externa por la derecha. RIGHT JOIN

Ejemplo: Obtener el DNI, nombre y apellidos de todos los clientes registrados junto con los datos de los contratos que han realizado. En la hoja de resultados se deben mostrar también los clientes que no han realizado ningún contrato.

SELECT clientes.dni, nombre, apellidos, contratos.* FROM contratos RIGHT JOIN clientes ON dni=dnicliente;

dni	nombre	apellidos	numcontrato	matricula	dnicliente	fini	ffin	kini	kfin
00371569B	Beatriz	Garcia Martin	7	6761JYM	00371569B	2016-12-24	2016-12-30	21500	25672
00445760C	Sandra	Flores Jorje	NULL	NULL	NULL	NULL	NULL	NULL	NULL
00740365D	Carlos Javier	Lopez Carvajal	12	8795JTK	00740365D	2017-01-06	2017-01-10	44850	46980
00740365D	Carlos Javier	Lopez Carvajal	19	3273JGH	00740365D	2017-01-09	NULL	17368	NULL
02748375F	Vanessa	Rodriguez	15	4738JBJ	02748375F	2017-01-08	2017-01-12	7965	8008
03549358G	Ismael	Poza Rincón	4	3273JGH	03549358G	2016-12-15	2016-12-22	15380	16250
03549358G	Ismael	Poza Rincón	20	2123JTB	03549358G	2017-01-09	NULL	34323	NULL

Reunión externa por la izquierda. RIGHT JOIN

Ejemplo: Obtener los datos de todos los clientes que nunca han hecho contratos.

SELECT clientes.* FROM contratos RIGHT JOIN clientes ON dni=dncliente WHERE numcontrato IS NULL;

dni	apellidos	nombre	direccion	localidad	carnet	fcarnet	fnac
00445760C	Flores Jorge	Sandra	C/ La Cañada 28	Madrid	B	1995-10-04	1974-06-02
11223344M	Garcia Garcia	Noelia	C/ Talavera 19	Toledo	B	2001-10-30	1982-03-13
28759595T	Ruiz Alonso	Ricardo	C/ Hortaleza, 56	Madrid	B	1999-10-13	1977-11-13
43809540X	Montoya	Natalia	C/ Verona, 3	Toledo	B	2005-10-14	1986-09-15

Otros tipos de reuniones

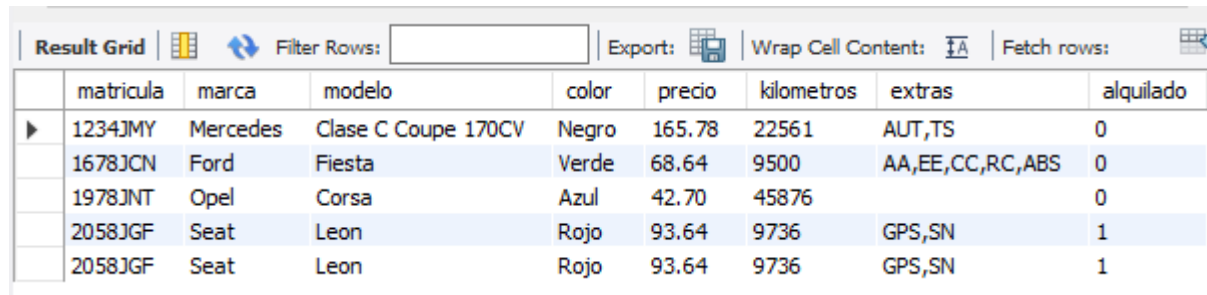
NATURAL JOIN

Permite combinar filas de dos tablas relacionadas por igualdad entre dos campos. Las condiciones son:

- Las tablas asociadas tienen uno o más pares de columnas con nombres idénticos.
- Las columnas deben ser del mismo tipo de datos.
- No se usa la cláusula ON.

Ejemplo: Obtener los datos de los automóviles que han realizado los cinco primeros contratos.

SELECT automoviles.* FROM automoviles NATURAL JOIN contratos LIMIT 5;



	matricula	marca	modelo	color	precio	kilometros	extras	alquilado
▶	1234JMY	Mercedes	Clase C Coupe 170CV	Negro	165.78	22561	AUT,TS	0
	1678JCN	Ford	Fiesta	Verde	68.64	9500	AA,EE,CC,RC,ABS	0
	1978JNT	Opel	Corsa	Azul	42.70	45876		0
	2058JGF	Seat	Leon	Rojo	93.64	9736	GPS,SN	1
	2058JGF	Seat	Leon	Rojo	93.64	9736	GPS,SN	1

¿Por medio de qué columna se están cruzando las tablas?

STRAIGHT_JOIN

Puede usarse con las cláusulas ON y USING para realizar lo mismo que INNER JOIN. Sin esas cláusulas realiza lo mismo que NATURAL JOIN.

Bases de Datos

Unidad 5: Realización de consultas avanzadas Sesión 4

Consultas de resumen o agregado

Agrupamiento de registros

Las **consultas de resumen o de agregado** permiten realizar cálculos de resumen o de grupo sobre las filas que tienen un valor igual en una o varias columnas.

Para realizar estos cálculos se usan las **funciones de agregado**.

- **Count(*expresión o columna*)**: Cuenta cuantas filas hay con la expresión o columna que no estén a valor nulo. Si en el argumento de la función escribimos *, se cuentan cuantas filas hay en la consulta. Si la expresión o columna vale **null**, no se cuenta.
- **Sum(*expresión o columna*)**: Calcula la suma de los valores numéricos indicados en el argumento. Si en la expresión o columna hay **null**, no se tiene en cuenta para la suma.
- **Min(*expresión o columna*)**: Obtiene el valor mínimo del argumento indicado.
- **Max(*expresión o columna*)**: Obtiene el valor máximo del argumento indicado.
- **Avg(*expresión o columna*)**: Obtiene la media aritmética del argumento indicado. No considera los valores nulos para el cálculo de la media.
- **Group_concat(*expresión o columna*)**: Obtiene la concatenación de todos los valores que se obtendrían en la consulta. No considera los valores nulos para la concatenación.

Ejemplo: Obtener cuantos contratos se han realizado:

```
SELECT count(*) FROM contratos;
```

Ejemplo: Obtener cuantos contratos realizados han finalizado.

```
SELECT count(ffin) FROM contratos;
```

Ejemplo: Obtener cuantos automóviles hay.

```
SELECT count(*) FROM automoviles;
```

Ejemplo: Obtener de cuantas marcas hay coches.

```
SELECT count(marca) FROM automoviles;
```

¿Es correcta esta última consulta?

Consultas de resumen

Ejemplo: Obtener la media de kilómetros realizados en los contratos finalizados, el máximo kilometraje realizado y el mínimo.

SELECT avg(kfin-kini), max(kfin-kini), min(kfin-kini) FROM contratos;

avg(kfin-kini)	max(kfin-kini)	min(kfin-kini)
1634.0667	5671	43

¿Es coherente la consulta mostrada con el ejemplo propuesto?
¿Qué tendría que incluir la consulta?

Consultas de resumen

Ejemplo: Obtener una cadena de caracteres concatenación de los nombres de todos los clientes de Toledo.

```
SELECT group_concat(nombre) FROM clientes WHERE localidad='toledo';
```

group_concat(nombre)
Fanny,Noelia,Natalia,Ana Belén

Consultas de resumen

Ejemplo: Obtener la suma total de kilómetros realizados en contratos finalizados por clientes de Madrid.

```
SELECT sum(kfin-kini) FROM contratos INNER JOIN clientes ON dnicliente=dni  
WHERE localidad='madrid';
```

sum(kfin-kini)
22293

Agrupamiento de registros

La cláusula **GROUP BY** permite agrupar varias filas de una consulta por una o varias expresiones. Todos los valores repetidos de las expresiones agrupadas, se mostrarán en una sola fila.

Ejemplo: Obtener la marca y modelo (sin repetir) de todos los automóviles que fueron contratados y cuya fecha de finalización de contrato está dentro del año 2018.

SELECT marca, modelo FROM automoviles INNER JOIN contratos ON contratos.matricula = automoviles.matricula WHERE year(ffin)=2018 GROUP BY marca, modelo;

¿Qué ocurre si pedimos que, además de marca y modelo, nos muestre el dni del cliente?

marca	modelo
Mercedes	Clase C Coupe 170CV
Ford	Fiesta
Opel	Corsa
Seat	Leon
Audi	A4
Seat	Ibiza
Seat	Leon SW
Renault	Clio
BMW	318 TDI

Agrupamiento de registros

Ejemplo: Obtener las localidades en las que se tienen clientes.

SELECT localidad FROM clientes GROUP BY localidad;

localidad
Cuenca
Madrid
Toledo

¿Qué utilidad podría tener esta consulta?

¿De qué forma alternativa podríamos obtener el mismo resultado?

Agrupamiento de registros

Ejemplo: Obtener el nombre y apellidos de los clientes que han realizado contratos a partir del 24 de diciembre de 2017. Los resultados deben estar ordenados ascendentemente por apellidos, nombre.

**SELECT nombre, apellidos FROM clientes INNER JOIN contratos ON dnicliente=dni
WHERE finis >='2017-12-24' GROUP BY **dnicliente** ORDER BY apellidos, nombre;**

nombre	apellidos
Soraya	Bats Corzo
Fanny	Cepeda
Eva	Coria García
Alicia	de la Hoz Gomez
Mariano	Dorado
Ana Belén	Fuentes Rojas
Beatriz	Garcia Martin
Carlos Javier	Lopez Carvajal
Ismael	Poza Rincón
Anais	Rodriguez
Vanessa	Rodriguez

¿Podríamos usar un **DISTINCT** en lugar del **GROUP BY**?

Agrupamiento de registros

Obtener cálculos sobre grupos de registros o filas

Cuando se realizan agrupamientos en una SELECT, podemos obtener cálculos sobre cada grupo con las funciones de resumen o agregado que hemos visto.

Ejemplo: Obtener cuantos automóviles hay de cada marca usando la función count. Hay que agrupar por marca en una consulta sobre la tabla automoviles.

SELECT marca, count(*) FROM automoviles GROUP BY marca;

marca	count(*)
Audi	2
BMW	2
Citroen	1
Ford	2
Mercedes	3
Opel	1
Renault	2
Seat	4

Agrupamiento de registros

Ejemplo: Obtener el nombre y apellidos de los clientes que han realizado contratos a partir del 24 de diciembre de 2017 y cuantos contratos han realizado desde esa fecha. Los resultados deben estar ordenados ascendentemente por apellidos, nombre.

SELECT nombre, apellidos, count(*) FROM clientes INNER JOIN contratos ON dnicliente=dni WHERE fini >='2017-12-24' GROUP BY dnicliente ORDER BY apellidos, nombre;

nombre	apellidos	count(*)
Soraya	Bats Corzo	1
Fanny	Cepeda	1
Eva	Coria García	1
Alicia	de la Hoz Gomez	2
Mariano	Dorado	1
Ana Belén	Fuentes Rojas	1
Beatriz	Garcia Martin	1
Carlos Javier	Lopez Carvajal	2

Agrupamiento de registros

Ejemplo: Obtener el precio medio, precio máximo y precio mínimo de los coches de cada marca ordenados por precio medio descendentemente.

**SELECT marca, avg(precio)AS medio, max(precio), min(precio) FROM automoviles
GROUP BY marca ORDER BY medio DESC;**

marca	medio	max(precio)	min(precio)
Mercedes	152.43333180745444	167.87	123.65
Audi	121.47999954223633	124.2	118.76
BMW	119.11999893188477	121.79	116.45
Seat	89.22249794006348	102.63	70.56
Ford	78.13000106811523	87.62	68.64
Renault	73.13500022888184	92.65	53.62
Citroen	62.66999816894531	62.67	62.67
Opel	42.70000076293945	42.7	42.7

Agrupamiento de registros

Ejemplo: Obtener el precio medio, precio máximo y precio mínimo de los coches de la marca SEAT.

```
SELECT avg(precio), max(precio), min(precio) FROM automoviles WHERE  
marca='SEAT';
```

avg(precio)	max(precio)	min(precio)
89.222500	102.63	70.56

Poner condiciones sobre resultados de funciones de agrupamiento. Cláusula HAVING:

En una consulta se pueden seleccionar filas que cumplan condiciones relativas al resultado de una función de agrupamiento.

Detrás de **HAVING** se ha de escribir una condición de selección.

En la condición de selección sólo se pueden usar funciones de agrupamiento o resumen, columnas de agrupación (las que se utilicen con GROUP BY) o cualquier expresión basada en estas columnas o en las funciones de agrupamiento.

Ejemplo: Obtener el número de clientes de cada localidad siempre que en la localidad haya más de tres clientes.

```
SELECT localidad,count(*) FROM clientes GROUP BY localidad HAVING count(*)>3;
```

¿En qué cambia la sentencia si quitamos el HAVING?

Agrupamiento de registros

Ejemplo: Obtener las marcas de coches cuyo precio medio de alquiler sea inferior a 105 Euros.

SELECT marca FROM automoviles GROUP BY marca HAVING avg(precio)<105;

marca
Ford
Seat

Agrupamiento de registros

Ejemplo: Obtener las marcas de coches y su precio medio de alquiler siempre que se cumpla que ese precio medio está comprendido entre 75 y 100 euros.

SELECT marca, avg(precio) AS media FROM automoviles GROUP BY marca **HAVING** media >=75 AND media<=100;

marca	media
Ford	78.130000
Seat	89.222500

¿Cómo podemos escribir la condición del HAVING sin utilizar operadores de comparación?

Bases de Datos

Unidad 5: Realización de consultas avanzadas Sesión 5

UNION:

UNION se usa para combinar los resultados de varias sentencias en un único conjunto de resultados.

Las columnas del resultado de ambas consultas deben ser del mismo tipo. El resultado final tendrá el nombre de columnas de la primera consulta.

Por defecto solo muestra las filas que son distintas (como si pusiéramos la cláusula DISTINCT). Podemos evitar esto con la cláusula ALL.

Ejemplo: Obtener el DNI de los clientes de la tabla contratos y de la tabla contratos2.

```
SELECT DISTINCT dnicliente FROM contratos UNION ALL SELECT DISTINCT  
dnicliente FROM contratos2;
```

El resultado será una tabla con los DNI de los clientes de ambas tablas. Si hay clientes con contratos en las dos tablas saldrán dos veces.

Ejemplo: Obtener la matrícula de los coches actualmente alquilados (ffin=NULL) y de los coches de marca Renault sin repetir matrículas.

```
SELECT matricula FROM contratos WHERE ffin IS NULL UNION SELECT matricula  
FROM automoviles WHERE marca="Renault";
```

¿Cómo podríamos obtener las matrículas, marcas, modelos y precios de alquiler de los automóviles que tienen un precio de alquiler superior al automóvil de matrícula 5031JHL?

SUBCONSULTAS

Subconsultas

Una subconsulta es una consulta **SELECT** que se hace dentro de otra consulta **SELECT**. Los datos que se obtienen de la subconsulta se usan en la consulta en la que se incluye.

También se pueden usar subconsultas dentro de las instrucciones **INSERT**, **UPDATE** y **DELETE**.

Si no existieran las subconsultas, para obtener las matrículas, marcas, modelos y precios de alquiler de los automóviles que tienen un precio de alquiler superior al automóvil de matrícula 5031JHL, posiblemente planteáramos esto con dos instrucciones;

1.- Obtener el precio de alquiler del automóvil de matrícula 5031JHL

SELECT precio FROM automoviles WHERE matricula='5031JHL';

2.- Obtener ahora las matrículas, marcas, modelos y precios de los automóviles con precio de alquiler superior a 116,45 Euros.

SELECT matricula, marca, modelo, precio FROM automoviles WHERE precio > 116.45;

Subconsultas

En el anterior ejemplo, lo que hemos hecho realmente es esto:

```
SELECT precio FROM automoviles WHERE matricula='5031JHL';
```



```
SELECT matricula, marca, modelo, precio FROM automoviles WHERE precio > 116.45;
```

Podemos modificar la instrucción segunda para que, en lugar del precio, use una subconsulta para obtener el precio del automóvil de la matrícula indicada.

A la primera de estas dos consultas la denominaremos **consulta interna** y a la segunda, **consulta externa**.

```
SELECT matricula, marca, modelo FROM automoviles WHERE precio>(SELECT  
precio FROM automoviles WHERE matricula = '5031JHL');
```


Subconsultas

Ejemplo: Obtener las matrículas, marcas, modelos y precios de alquiler de los automóviles que tienen un precio de alquiler superior al automóvil de matrícula 5031JHL.

SELECT matricula, marca, modelo FROM automoviles WHERE precio > (SELECT precio FROM automoviles WHERE matricula = '5031JHL') ;

matricula	marca	modelo
1234JMY	Mercedes	Clase C Coupe 170CV
3273JGH	Audi	A4
4738JBJ	Audi	A3
5678JRZ	Mercedes	Clase C
7856JLD	BMW	318 TDI
8795JTK	Mercedes	GLA

MUY IMPORTANTE: En subconsultas como esta anterior, que se usan para comparar con un valor, las subconsultas deben devolver únicamente un valor

Tipos de subconsultas

El estándar SQL define tres tipos de subconsultas:

- **Subconsultas escalares.** Son aquellas que devuelven una columna y una fila. Es decir, aquellas que devuelven un único valor concreto.
- **Subconsultas de fila.** Son aquellas que devuelven más de una columna pero una única fila.
- **Subconsultas de tabla.** Son aquellas que devuelve una o varias columnas y cero o varias filas.

Subconsultas

Tipos de subconsultas

También podemos distinguir las subconsultas dependiendo del lugar que ocupen dentro de la consulta externa:

- Subconsultas en la cláusula **SELECT**
- Subconsultas en la cláusula **WHERE**
- Subconsultas en la cláusula **HAVING**
- Subconsultas en la cláusula **FROM**

¿Qué tipos de subconsultas (escalares, de fila o de tabla) se utilizarán en las distintas cláusulas de una consulta?

Operadores

Los operadores que podemos usar en las subconsultas son los siguientes:

- Operadores básicos de comparación (>, >=, <, <=, !=, <>, =).
- Predicados ALL y ANY.
- Predicado IN y NOT IN.
- Predicado EXISTS y NOT EXISTS.

Subconsultas

Ejemplo: Obtener los números de contrato, matrículas, dni de cliente y nombre del cliente de todos los contratos.

SELECT numcontrato, matricula, dnicliente, (**select nombre from clientes where dni=dnicliente**) as nombreciente **FROM** contratos;

	numcontrato	matricula	dnicliente	nombreciente
►	1	1234JMY	08785691K	Mariano
	2	7856JLD	07385709H	Fanny
	3	5573JFS	37409800V	Reyes
	4	3273JGH	03549358G	Ismael
	5	3765JSD	58347695Z	Ana Belén

¿De qué otra manera podemos realizar la misma consulta?

¿Qué ocurre si a la subconsulta interna añadimos el campo apellidos? ¿Por qué?

Subconsultas

Ejemplo: Obtener las matrículas, marcas, modelos y precios de alquiler de los automóviles de color rojo que tienen un precio de alquiler superior al automóvil de matrícula 5031JHL.

SELECT matricula, marca, modelo **FROM** automoviles **WHERE** precio>(SELECT precio **FROM** automoviles **WHERE** matricula = '5031JHL') **AND** color='rojo';

matricula	marca	modelo
3273JGH	Audi	A4

Subconsultas

Ejemplo: Obtener la marca y modelo del coche de precio de alquiler más alto.

SELECT marca, modelo, precio **FROM** automoviles **WHERE** precio = (**SELECT** **max**(precio) **FROM** automoviles);

marca	modelo	precio
Mercedes	GLA	167.87

Subconsultas

Ejemplo: Obtener las marcas y sus precios medios de alquiler siempre que se cumpla que ese precio medio es inferior al precio de alquiler del automóvil de matrícula 5031JHL.

SELECT marca, avg(precio) **FROM** automoviles **GROUP BY** marca **HAVING**
avg(precio) < (SELECT precio FROM automoviles WHERE matricula = '5031JHL');

marca	avg(precio)
Citroen	62.670000
Ford	78.130000
Opel	42.700000
Renault	73.135000
Seat	89.222500

Ejemplo: Obtener la marca y modelo del coche correspondiente al contrato número 10.

SELECT marca, modelo FROM automoviles WHERE matricula = (SELECT matricula FROM contratos WHERE numcontrato=10);

Pero esto se puede hacer de la siguiente forma, y es más adecuado, ya que la consulta consume menos tiempo. En general las instrucciones que usan subconsultas llevan más tiempo que las que no las usan, aunque esto no siempre es así.

SELECT marca, modelo FROM automoviles INNER JOIN contratos USING (matricula) WHERE numcontrato=10;

marca	modelo
Audi	A4

Subconsultas

Hasta ahora hemos usado las subconsultas dentro de las cláusulas SELECT, WHERE y HAVING. También se pueden usar en la cláusula FROM, para obtener una hoja de resultados a partir de la que construimos una consulta.

Ejemplo: Obtener los datos de los clientes que tienen contratos en las dos tablas de contratos (contratos y contratos 2).

SELECT * FROM clientes INNER JOIN (SELECT DISTINCT dnicliente FROM contratos UNION ALL SELECT DISTINCT dnicliente FROM contratos2) AS t ON t.dnicliente=clientes.dni GROUP BY dni HAVING count(*)=2;

dni	apellidos	nombre	direccion	localidad	carnet	fcarnet	fnac	dnidiente
00740365D	Lopez Carvajal	Carlos Javier	C/ Salmoral 28	Madrid	D	1994-08-21	1973-12-11	00740365D
07385709H	Cepeda	Fanny	C/ Antonio Gala, 34	Toledo	B	2006-11-05	1983-01-24	07385709H

¿Por qué hemos incluido la cláusula ALL en el UNION?

¿Qué posibles valores devolverá el count(*) del HAVING de esta consulta?

Subconsultas

Ejemplo: En la base de datos ligatercera, obtener cuantos equipos han metido goles en la jornada 1.

Lo que vamos a hacer es una subconsulta con la unión de contar cuantos equipos locales han metido goles y cuantos equipos visitantes han metido goles. Esa unión la renombramos para tratarla como si fuera una tabla. De esa tabla, sumamos los valores que contiene, es decir, los equipos locales que han marcado goles y los visitantes que han marcado goles.

```
SELECT sum(marcaron) FROM (SELECT count(*) AS marcaron FROM partidos  
WHERE golesloc>0 AND numjornada=1 UNION ALL SELECT count(*) AS marcaron  
FROM partidos WHERE golesvis>0 AND numjornada=1) AS t;
```

sum(marcaron)
12

¿Por qué utilizamos el UNION?

¿Por qué hacemos un sum(marcaron) en lugar de count(marcaron)?

Comprobar si un dato está incluido en varios valores devueltos por una subconsulta

No se pueden usar ni el operador de igualdad (=) ni otros operadores relacionales para comparar resultados, con subconsultas que devuelven más de un valor. Si queremos comprobar que un valor está incluido dentro del conjunto de valores devueltos por la subconsulta, usaremos el operador IN.

Ejemplo: Obtener las matrículas, marcas y modelos de los coches alquilados desde el 1 de enero de 2018.

SELECT matricula, marca, modelo FROM automoviles WHERE matricula IN (SELECT matricula FROM contratos WHERE fini>='2018-01-01');

matricula	marca	modelo
2058JGF	Seat	Leon
2123JTB	Renault	Megane
3273JGH	Audi	A4
4387JDD	Citroen	C3
4738JBJ	Audi	A3
5031JHL	BMW	318 i
5678JRZ	Mercedes	Clase C
8795JTK	Mercedes	GLA

Subconsultas

Ejemplo: Obtener la marca y modelo de todos los coches que ha alquilado Ismael Poza Rincón.

SELECT marca, modelo FROM automoviles WHERE matricula IN (SELECT matricula FROM contratos WHERE dncliente = (SELECT dni FROM clientes WHERE nombre='Ismael' AND apellidos='Poza Rincón'));

marca	modelo
Audi	A4
Renault	Megane

Subconsultas

Ejemplo: Obtener los datos de los clientes que no han realizado ningún contrato.

SELECT * FROM clientes WHERE dni NOT IN (SELECT DISTINCT dnicliente FROM contratos);

dni	apellidos	nombre	direccion	localidad	carnet	fcarnet	fnac
00445760C	Flores Jorje	Sandra	C/ La Cañada 28	Madrid	B	1995-10-04	1974-06-02
11223344M	Garcia Garcia	Noelia	C/ Talavera 19	Toledo	B	2001-10-30	1982-03-13
28759595T	Ruiz Alonso	Ricardo	C/ Hortaleza, 56	Madrid	B	1999-10-13	1977-11-13
43809540X	Montoya	Natalia	C/ Verona, 3	Toledo	B	2005-10-14	1986-09-15

¡OJO! Cuando se utiliza el NOT IN, si la subconsulta devuelve algún nulo, el resultado es falso o nulo pero nunca verdadero.

¿En qué situaciones podríamos asegurar que el resultado no incluirá nulos?

Subconsultas

Uso del cuantificador ALL

En subconsultas que devuelven varios valores, el cuantificador ALL permite seleccionar las filas que cumplan con una determinada condición respecto de todos los valores devueltos por la subconsulta.

Es decir, el operador ALL selecciona los valores si todos los registros de subconsulta cumplen la condición.

Ejemplo: Obtener las marcas de coches de las que no se ha alquilado ningún coche en 2018.

```
SELECT marca FROM automoviles WHERE marca <> ALL (SELECT DISTINCT marca  
FROM contratos INNER JOIN automoviles USING (matricula) WHERE  
year(fini)=2018);
```

marca
Ford
Opel

¡OJO! Si la subconsulta no devuelve ningún valor, ALL devuelve el valor verdadero.

Uso del cuantificador ANY

En subconsultas que devuelven varios valores, el cuantificador ANY permite seleccionar las filas que cumplan con una determinada condición para al menos uno de los valores devueltos por la subconsulta.

Es decir, el operador ANY devuelve verdadero si alguno de los valores de la subconsulta cumple la condición.

Ejemplo: Obtener los datos de los coches con precio de alquiler menor que el de alguno de los coches SEAT.

SELECT * FROM automoviles WHERE precio < ANY (SELECT precio FROM automoviles WHERE marca='seat');

matricula	marca	modelo	color	precio	kilometros	extras	alquilado
1678JCN	Ford	Fiesta	Verde	68.64	9500	AA,EE,CC,RC,ABS	0
1732JBS	Seat	Leon	Negro	90.06	2500	TS,SN	0
1978JNT	Opel	Corsa	Azul	42.70	45876		0
2058JGF	Seat	Leon	Rojo	93.64	9736	GPS,SN	1
2123JTB	Renault	Megane	Amarillo	92.65	34323	TS,SN	1
3765JSD	Seat	Ibiza	Rojo	70.56	7683	SN	0
4387JDD	Citroen	C3	Verde	62.67	23057		0
6761JYM	Renault	Clio	Blanco	53.62	25672	SN	0
7839JDR	Ford	Focus	Blanco	87.62	15873	GPS,SN	1

Bases de Datos

Unidad 5: Realización de consultas avanzadas Sesión 6

FUNCIONES MYSQL

- Todos los SGBD incluyen un conjunto de funciones que pueden ser usadas para obtener fácilmente determinados resultados.
- Cada SGBD incluye un conjunto propio de funciones. Las funciones no forman parte del lenguaje SQL.
- La llamada a una función puede realizarse en las sentencias SELECT, INSERT, UPDATE y DELETE.
- Toda función devuelve un valor y opera con unos datos recibidos o parámetros.
- Para llamar a una función siempre se usa la sintaxis:

Nombre_funcion(param1, param1,...)

- Como parámetros pueden darse valores constantes, nombres de columnas, llamadas a otras funciones y operaciones entre los anteriores.

Las funciones MySQL pueden clasificarse en función de los tipos de datos con los que trabajan o del tipo de operación que realizan en:

- **Funciones matemáticas o numéricas**
- **Funciones de cadena de caracteres**
- Funciones de fecha y hora
- Funciones de búsqueda de texto
- Funciones de control de flujo
- Funciones de conversión
- Funciones de agregado o agrupación
- Otras funciones

FUNCIONES MATEMÁTICAS PRINCIPALES

- **pow(X,Y)** : Devuelve el resultado X elevado a Y.
- **sqrt(X)** : Devuelve la raíz cuadrada de X.
- **ceil(X)** : Redondea al entero más cercano por arriba.
- **floor(X)**: Redondea al entero más cercano por abajo.
- **round(X)** : Redondea al entero más cercano.
- **round(X,D)** : Redondea al número más cercano usando D decimales.
- **truncate(X,D)**: Obtiene el número X truncado a D decimales.
- **rand()** : Devuelve un número coma flotante aleatorio mayor o igual que cero y menor que 1.0.

FUNCIONES DE CADENA DE CARACTERES

char_length(cadena) : Devuelve el número de caracteres que tiene el contenido de la cadena.

concat(cad1, cad2,...) : Devuelve la cadena resultado de concatenar todas las cadenas pasadas. Se pueden pasar otros tipos de datos en cuyo caso los trata como cadenas de caracteres.

left(cad, N) : Devuelve los N primeros caracteres de cad.

right(cad, N) : Devuelve los N últimos caracteres de cad.

insert(cadena, posicion, longitud, nueva_cadena): Devuelve el resultado de sustituir con la nueva cadena los caracteres de cadena expresados en longitud desde la posición indicada.

FUNCIONES DE CADENA DE CARACTERES

Ejemplo: Obtener los nombres y apellidos de todos los clientes en una sola columna con el formato apellidos, nombre.

SELECT concat(apellidos, ", ", nombre) AS nombrecompleto FROM clientes;

EJEMPLO: Suponiendo que la columna localidad de CLIENTES contiene erróneamente SANTRDER para todos los alumnos de Santander, hacer lo necesario modificar el valor de esa columna usando la función insert.

SET SQL_SAFE_UPDATES = 0; # Esta instrucción desactiva el modo seguro para UPDATES y permite actualizar valores sin usar clave

UPDATE clientes SET localidad= insert(localidad, 5, 1, 'AN') WHERE localidad='SANTRDER';

SET SQL_SAFE_UPDATES = 1; # Esta instrucción reactiva el modo seguro para UPDATES

Aunque, por lógica, esto cualquiera lo haría así:

UPDATE clientes SET localidad= 'SANTANDER' WHERE localidad='SANTRDER';

FUNCIONES DE CADENA DE CARACTERES

locate(subcadena, cadena): Devuelve la posición a partir de la cual se encuentra subcadena en cadena, cero si no la encuentra.

locate(subcadena, cadena, pos): igual que la anterior buscando a partir de la posición pos.

lcase(cadena): Devuelve la cadena en minúsculas.

ucase(cadena): Devuelve cadena en mayúsculas.

lpad(cadena, N, subcadena): Devuelve cadena ocupando N caracteres, rellenando por la izquierda con subcadena si fuese necesario.

rpad(cadena, N, subcadena): igual que la anterior por la derecha.

ltrim(cadena): Devuelve cadena tras eliminarle los espacios por la izquierda si los tuviera.

rtrim(cadena): igual que la anterior por la derecha.

trim(subcadena FROM cadena): Devuelve la cadena tras eliminarle las apariciones de subcadena por la izquierda y por la derecha. Esta función admite otras sintaxis.

replace(cadena, subcadena a sustituir, subcadena a incluir): Sustituye una cadena de caracteres por otra.

FUNCIONES DE CADENA DE CARACTERES

Ejemplo: Obtener la calle en la que vive cada cliente. No hay que escribir otros datos de la dirección.

```
SELECT LEFT(direccion, LOCATE(',', direccion)-1) FROM clientes;
```

EJEMPLO: Obtener las matrículas y precios de los automóviles de forma que los precios ocupen 20 posiciones rellenando las sobrantes con '+.'

```
SELECT matricula, LPAD(precio, 20, '+.') FROM automoviles;
```

Ejemplo: Sustituir dos espacios en blanco por uno solo

```
SELECT replace(direccion, ' ', ' ') FROM clientes;
```

Bases de Datos

Unidad 5: Realización de consultas avanzadas Sesión 7

Funciones en MySQL 5.7

- Funciones de fecha y hora
- Funciones de búsqueda de texto
- Funciones de control de flujo
- Funciones de conversión
- Funciones de agregado o agrupación
- Otras funciones

Adddate(fecha, INTERVAL N tipo_intervalo): Devuelve la fecha incrementada en N el tipo de intervalo indicado.

El tipo de intervalo para fechas puede ser DAY, WEEK, MONTH, QUARTER, YEAR

EJEMPLO: Suponiendo que todas las fechas de los contratos tienen como año el 2007, modificarlas para que tengan como año el 2017.

```
UPDATE contratos SET finicial=adddate(finicial,INTERVAL 10 YEAR),  
ffinal=adddate(ffinal,INTERVAL 10 YEAR);
```

FUNCIONES DE FECHA Y HORA

Addtime(tiempo1, tiempo2): Devuelve el resultado de sumar los dos tiempos.

Subtime(tiempo1, tiempo2): Devuelve el resultado de tiempo1-tiempo2.

Curtime(): Devuelve la hora actual.

EJEMPLO: Obtener la hora que será dentro de 1 hora y 20 minutos y la que era hace 3 horas y 15 minutos.

```
SELECT addtime(curtime(),'1:20:0'), subtime(curtime(),'3:15:0');
```

FUNCIONES DE FECHA Y HORA

datediff(fecha1, fecha2): Devuelve los días transcurridos entre fecha2 y fecha1.

Subdate(fecha, INTERVAL N tipo_periodo): Devuelve la fecha resultado de restarle a fecha el tipo de periodo N veces.

Curdate(): Devuelve la fecha actual.

EJEMPLO: Obtener la fecha que era hace dos trimestres y cuantos días han transcurrido desde esa fecha.

```
SELECT subdate(curdate(), INTERVAL 2 QUARTER), datediff( curdate(),  
subdate(curdate(), INTERVAL 2 QUARTER));
```

FUNCIONES DE FECHA Y HORA

date(fechahora): Devuelve la fecha de una dato DATETIME.

time(fechahora): Devuelve la parte TIME de una dato DATETIME.

year(fecha): Devuelve el año de una fecha.

quarter(fecha): Devuelve el trimestre de una fecha.

month(fecha): Devuelve el mes numérico de una fecha.

monthname(fecha): Devuelve el nombre del mes de una fecha.

day(fecha): Devuelve el día del mes de una fecha.

dayname(fecha): Devuelve el nombre del día de la semana de una fecha.

dayofweek(fecha): Devuelve el número de día de la semana de una fecha. Semana comienza en Domingo con número 1.

dayofyear(fecha): Devuelve el número de día del año de una fecha.

weekofyear(fecha): Devuelve el número de semana del año de la fecha dada. Las semanas comienzan en domingo y la primera del año es la primera con comienzo en domingo.

FUNCIONES DE FECHA Y HORA

now(): Devuelve la fecha y hora actuales.

hour(tiempo): Devuelve la parte horas de tiempo.

minute(tiempo): Devuelve la parte minutos de tiempo.

second(tiempo): Devuelve la parte segundos de tiempo.

sec_to_time(segundos): Convierte los segundos pasados a dato TIME

time_to_sec(tiempo): Opuesta a la anterior

Ejemplo: Obtener la hora actual y cuantos minutos faltan para la siguiente hora en punto.

```
SELECT curtime();
```

```
SELECT 60-minute(curtime());
```


FUNCIONES DE CONVERSIÓN

convert(valor,tipo): Convierte el valor al tipo de dato que se indique en tipo.

BINARY valor: Realmente BINARY no es una función de MySQL sino un operador especial de MySQL. Hace que valor se interprete conforme a su codificación interna. Su principal aplicación es la de servir para comparar dos cadenas de caracteres diferenciando entre mayúsculas y minúsculas, entre palabras con acentos y sin acentos, etc.

Si ejecutamos lo siguiente para obtener los clientes cuyo nombre está guardado como SANDRA, obtenemos una clienta Sandra.

```
SELECT * FROM clientes WHERE nombre='SANDRA';
```

Sin embargo, si ejecutamos lo siguiente, no obtenemos ningún resultado pues no hay ningún nombre que sea exactamente SANDRA.

```
SELECT * FROM clientes WHERE BINARY nombre='SANDRA';
```

FUNCIONES DE CONTROL DE FLUJO

CASE *valor* WHEN [*valor1*] THEN *resultado1*
[WHEN [*valor2*] THEN *resultado2* ...]
[ELSE *resultado*]
END

Devuelve el resultado correspondiente al primer valorN que coincida con *valor*. Si ningún valorN coincide con *valor* se devuelve el resultado que hay tras la cláusula ELSE, y si no tuviera esta cláusula se devuelve NULL.

Ejemplo: Obtener el día de la semana que es hoy en español.

```
SELECT case dayofweek(curdate()) when 1 then 'domingo' when 2 then  
'lunes' when 3 then 'martes' when 4 then 'miercoles' when 5 then 'jueves'  
when 6 then 'viernes' when 7 then 'sabado' end;
```

FUNCIONES DE CONTROL DE FLUJO

CASE WHEN [condicion1] THEN resultado1
[WHEN [condicion2] THEN resultado2 ...]
[ELSE resultado]
END

Devuelve el resultado correspondiente a la primera condición que se cumpla.

Ejemplo: Obtener la calificación de los alumnos en formato alfanumérico. Debe preverse una calificación incorrecta.

```
SELECT nombre, apellidos, case when nota>=0 and nota<5 then 'suspenseo'  
when nota<6 then 'aprobado' when nota<7 then 'bien' when nota<9 then  
'notable' when nota<10 then 'sobresaliente' else 'calificacion incorrecta'  
end FROM alumnos;
```

FUNCIONES DE CONTROL DE FLUJO

IF(*expr1*,*expr2*,*expr3*)

Si *expr1* es verdadera (*expr1* <> 0 and *expr1* <> NULL), devuelve *expr2*, si no devuelve *expr3*.

Ejemplo: Obtener la matrícula marca y modelo de los automóviles junto con su estado (escribiendo alquilado o disponible).

```
SELECT matricula, marca, modelo, if(alquilado, 'alquilado', 'disponible')  
FROM automoviles;
```

OTRAS FUNCIONES

aes_encrypt(texto,clave): Permite encriptar información usando una clave de encriptación. Utiliza la técnica AES

aes_decrypt(texto,clave): Para desencriptar.

md5(texto): Para encriptar con algoritmo MD5. NO es reversible , es decir, no hay una función para desencriptar.

connection_id(): Devuelve el número de identificador de la conexión cliente MySQL al servidor.

current_user(): Devuelve el nombre del usuario y del equipo donde éste ha sido autenticado.

last_insert_id(): Devuelve el último valor insertado en una columna AUTO_INCREMENT.

row_count(): Devuelve el número de filas que se vieron afectadas por la operación precedente de borrado, inserción o modificación.

version(): Devuelve la versión del servidor MySQL

VARIABLES DEFINIDAS POR EL USUARIO DE MYSQL

Variables definidas por el usuario de MYSQL

Cuando se desea **pasar un valor de una instrucción SQL a otra** es necesario el uso de variables de usuario. Para ello, se **almacenará el valor en una variable definida por el usuario de MySQL en la primera instrucción y se hará referencia a él, en las instrucciones siguientes, por medio de dicha variable.**

Para crear una variable definida por el usuario **se debe respetar el formato** de caracteres alfanuméricos con una longitud máxima de 64 caracteres (a partir de MySQL 5.7.5):
@nombre_variable

Las variables definidas por el usuario **no distinguen entre mayúsculas y minúsculas.** Significa que @id y @ID son lo mismo

Se puede asignar la variable definida por el usuario a determinados **tipos de datos** como **integer, floating point, decimal, string o NULL.**

Un usuario no puede ver una variable definida por otro. En otras palabras, **una variable definida por el usuario es específica de la sesión.**

Hay que tener en cuenta que las variables definidas por el usuario son una **extensión específica de MySQL** del estándar SQL. Es posible que no estén disponibles en otros sistemas gestores de bases de datos.

Existen **dos maneras de asignar un valor a una variable** definida por el usuario.

La primera forma es utilizar la instrucción **SET** de la siguiente manera:

SET @nombre_variable:= valor;

Se puede utilizar := o = como operador de asignación en la instrucción SET.

Por ejemplo, la instrucción que asigna el número 100 a la variable @numcontrato:

SET @numcontrato := 100;

O el valor del número de contrato más alto:

SET @numcontrato = (select max(numcontrato) from contratos);

~~SET @numcontrato = max(numcontrato) from contratos;~~ (esto no funciona)

¿Por qué no podemos utilizar esta última asignación?

La segunda forma de asignar un valor a una variable es utilizar la instrucción **SELECT**.

En este caso, se debe utilizar el operador de asignación :=

```
SELECT @nombre_variable:= valor;
```

Después de la asignación, se puede utilizar la variable en la instrucción posterior donde se permite una expresión, por ejemplo, en la cláusula WHERE, la instrucción INSERT o UPDATE.

La siguiente instrucción **obtiene el valor** del numero de contrato más alto **y lo asigna a la variable** definida por el usuario (compárese con SET)

```
SELECT @maximo:=MAX(numcontrato) FROM contratos;
```

La siguiente instrucción **utiliza la variable** @maximo para consultar la información del contrato con el número más alto.

```
SELECT * FROM contratos WHERE numcontrato=@maximo;
```

Una variable definida por el usuario solo puede contener un único valor. Si la instrucción SELECT devuelve varios valores, la variable tomará el valor de la última fila del resultado.

```
SELECT  @numcontrato:=numcontrato as result  
FROM    contratos  
WHERE  
fini > '2018-01-06'  
ORDER BY fini;
```

```
SELECT @numcontrato;
```