



Tema 5.2 : Cont. Select



ÍNDICE

5.7.- Subconsultas

- 5.7.1.- Test de Comparación

- 5.7.2.- Test de pertenencia a un conjunto

- 5.7.3.- Test de Existencia

- 5.7.4.- Test cuantificados ALL y ANY

- 5.7.5.- Subconsultas anidadas

5.8.- Consultas multitas

- 5.8.1.- Consultas multitas SQL1

- 5.8.2.-Consultas multitas SQL2

5.9.- Consultas Reflexivas

5.10.- Consultas con Tablas Derivadas

Actividades 5.1 => 5.7

5.7.- Subconsultas

- Se utilizan para realizar filtrados con los datos de otra consulta. Estos filtros pueden aplicarse en la cláusula WHERE para seleccionar registros y en la HAVING para filtrar grupos.
- Ejemplo, codificar una consulta para ver los nombres de los jugadores de la división SouthWest:

```
mysql> select nombre from jugadores where nombre_equipo IN (SELECT nombre from
equipos where division='SouthWest');
```

nombre
Andre Brown
Kwame Brown
Brian Cardinal
Jason Collins
Mike Conley
Javaris Crittenton
Rudy Gay
Casey Jacobsen
Kyle Lowry
Aaron McKie
Darko Milicic
Mike Miller
Juan Carlos Navarro
Hakim Warrick
Chris Andersen
Hilton Armstrong
Ryan Bowen
Rasual Butler
Tyson Chandler
Melvin Ely
Mike James
Jannero Pargo
Chris Paul
Morris Peterson
Peja Stojakovic
Bonzi Wells
Tony Parker
Damon Stoudamire
Kurt Thomas
Ine Udoka
Jacque Vaughn

```
73 rows in set (0.00 sec)

mysql>
```

5.7.- Subconsultas

- Una sentencia subordinada de otra puede tener a su vez otras sentencias subordinadas a ella. Se llama sentencia externa a la primera de todas, la que no es subordinada de ninguna. Una sentencia es antecedente de otra cuando ésta es su subordinada directa o subordinada de sus subordinadas a cualquier nivel. A las sentencias subordinadas también se les llama anidadas.
- Las subordinadas pueden ser parte de los siguientes predicados:
 - ✓ Predicados básicos de comparación
 - ✓ Predicados cuantificados (ANY,SOME, ALL)
 - ✓ Predicados EXISTS
 - ✓ Predicado IN

5.7.1.- Test de Comparación

- Consiste en utilizar los operadores de relación =, >, >=, <, <=, <> para comparar el valor producido con un valor único generado por una subconsulta. Por ejemplo, para obtener el nombre del jugador de la NBA de mayor altura:

```
mysql> select nombre from jugadores where altura = (select max(altura) from jugadores);
+-----+
| nombre |
+-----+
| Yao Ming |
+-----+
1 row in set (0.01 sec)
```

- La subconsulta produce un único resultado. La subconsulta debe estar siempre al lado derecho del operador de comparación. **Campo <= subConsulta**

5.7.2.- Test de pertenencia a un conjunto

- Consiste en utilizar el operador IN para filtrar los registros cuya expresión coincida con algún valor producido por la subconsulta. Por ejemplo, extraer las divisiones de la nba donde juegan jugadores españoles, ordenados por división:

```
mysql> select division from equipos where nombre IN (select nombre_equipo from jugadores where procedencia = 'Spain') ORDER BY DIVISION ASC;
```

division
Atlantic
NorthWest
Pacific
SouthWest

```
4 rows in set (0.00 sec)
```

5.7.3.- Test de Existencia

- Permite filtrar los resultados de una consulta si existen filas en la consulta asociada, e.d, si la subconsulta genera un número de filas distinto de 0.

SELECT columnas FROM *tabla* WHERE EXISTS (subconsulta)

```
mysql> #Seleccionar los equipos que tengan jugadores españoles
mysql> select nombre from equipos where EXISTS (select nombre from jugadores where
equipos.nombre=jugadores.nombre_equipo and procedencia='Spain');
+-----+
| nombre |
+-----+
| Grizzlies |
| Lakers   |
| Raptors  |
| Trail Blazers |
+-----+
4 rows in set (0.00 sec)
```

```
mysql> select nombre from equipos where nombre IN (select nombre_equipo from jug
adores where procedencia='Spain');
+-----+
| nombre |
+-----+
| Grizzlies |
| Lakers   |
| Raptors  |
| Trail Blazers |
+-----+
4 rows in set (0.00 sec)
```

5.7.3.- Test de No Existencia

```
mysql> #Seleccionar los equipos que no tengan jugadores españoles
mysql> select nombre from equipos where NOT EXISTS (select nombre from jugadores
where equipos.nombre=jugadores.nombre_equipo and procedencia='Spain');
+-----+
| nombre |
+-----+
| 76ers  |
| Bobcats|
| Bucks  |
| Bulls  |
| Cavaliers|
| Celtics|
| Clippers|
| Hawks  |
| Heat    |
| Hornets|
| Jazz    |
| Kings   |
| Knicks  |
| Magic    |
| Mavericks|
| Nets     |
| Nuggets  |
| Pacers   |
| Pistons  |
| Rockets  |
| Spurs    |
| Suns     |
| Supersonics|
| Timberwolves|
| Warriors |
| Wizards |
+-----+
26 rows in set (0.00 sec)

mysql>
```

Es como si cada registro devuelto por la consulta principal provocara la ejecución de la subconsulta, si la consulta principal devuelve por ejemplo, 30 registros, se ejecutarían 30 subconsultas, pero en realidad el SGBD realiza sólo dos consultas y una operación de JOIN.

```
mysql> select nombre from equipos where nombre NOT IN (select nombre_equipo from
jugadores where procedencia='Spain');
+-----+
| nombre |
+-----+
| 76ers  |
| Bobcats|
| Bucks  |
| Bulls  |
| Cavaliers|
| Celtics|
+-----+
```


5.7.4.- Test cuantificados ALL y ANY

- Sirven para calcular la relación entre una expresión y todos los registros de la subconsulta (ALL) o algunos de los registros de la subconsulta (ANY o SOME).
- Por ejemplo, obtener todos los jugadores de la NBA que pesan más que todos los jugadores españoles.

```
mysql> select nombre,peso from jugadores where peso > ALL (select peso from jugadores where procedencia='Spain');
```

nombre	peso
Michael Doleac	262
Al Jefferson	265
Chris Richard	270
Elton brand	254
Paul Davis	270
Chris Kaman	265
Kwame Brown	270
Mehmet Okur	263
Ike Diogu	255
David Harrison	280
Jermaine O'Neal	260
Jason Maxiell	260
Nick Collison	255
Shaquille O'Neal	325
Brian Skinner	255

```
59 rows in set (0.00 sec)
```

5.7.5.- Subconsultas anidadas

- Se puede usar una subconsulta para filtrar el resultado de otra subconsulta. Por ejemplo, obtener el nombre de la ciudad donde juega el jugador más alto de la NBA:

```
mysql> select ciudad from equipos where nombre= (select nombre_equipo from jugadores where altura= (select max(altura) from jugadores));
```

ciudad
Houston

```
1 row in set (0.00 sec)
```

Los pasos serían:

1/ Obtener la altura máxima:

Select max (altura) from jugadores;

2/ Obtener el nombre del equipo , a través de la altura se localiza al jugador y por tanto el nombre de su equipo

3/ Obtener la ciudad del equipo

5.8.- Consultas multitaslas

Es aquella en la que se puede consultar información de más de una tabla. Se aprovechan los campos relacionados de las tablas para unirlos (join).

```
SELECT [DISTINCT ] select_expr [, select_expr ...]  
[FROM referencias_tablas]  
[WHERE filtro]  
[GROUP BY {expr esión [,expresión]...}  
[HAVING filtro_grupos]  
[ORDER BY {col_name | expr | position} [ASC | DESC], ...]
```

La diferencia con las consultas sencillas se halla en la cláusula FROM. En vez de una tabla se puede desarrollar el token *referencias_tablas*:

referencias_tablas

referencias_tablas:

referencia_tabla [, referencia_tabla] ...

| referencia_tabla [INNER | CROSS] JOIN referencia_tabla [ON Condición]

| referencia_tabla LEFT [OUTER] JOIN referencia_tabla ON Condición

| referencia_tabla RIGHT [OUTER] JOIN referencia_tabla ON Condición

referencia_tabla:

Nombre_tabla [[AS] alias]

La primera opción, referencia_tabla[, referencia_tabla]... es típica de SQL1 para las uniones, que consiste en un producto cartesiano más un filtro por las columnas relacionadas, y el resto de opciones son propias de SQL2.

5.8.1.- Consultas multitas SQL1

- El producto cartesiano de dos tablas son todas las combinaciones de las filas de una tabla unidas a las filas de la otra tabla. Por ejemplo, en la BD de veterinarios con dos tablas mascotas y propietarios.

```
mysql> select * from mascotas;
```

idMascota	nombre	especie	raza	pedigree	fechaNacimiento	sexo
1	kira	canina	dalmata	1	1970-01-01	h
2	terry	canina	pastor aleman	1	1973-01-01	m
3	lorato	ave	loro	1	1973-01-01	m
4	donna	canina	fox terrier	0	1980-01-01	h
5	dante	canina	cairn terrier	0	1980-01-01	m
6	boris	canina	bichon maltes	0	1998-01-01	m
7	laica	canina	boxer	0	0000-00-00	h
8	don	canina	dogo	0	1998-09-08	m
9	fox	canina	basset hound	1	2002-09-07	m
10	TOM	CANINA	CANICHE	0	2012-02-12	m

```
10 rows in set (0.00 sec)
```

```
mysql> select * from mascotas where propietario is null;
```

idMascota	nombre	especie	raza	pedigree	fechaNacimiento	sexo	propietario
7	laica	canina	boxer	0	0000-00-00	h	NU
10	TOM	CANINA	CANICHE	0	2012-02-12	m	NU

```
2 rows in set (0.00 sec)
```

5.8.1.- Consultas multitas SQL1

```
mysql> select * from propietarios;
```

```
+-----+-----+
| DNI      | NOMBRE      |
+-----+-----+
| 11111111A | Pepe Rodriguez |
| 11111111B | Luis Rodriguez |
| 11111111C | Ana Rodriguez  |
+-----+-----+
3 rows in set (0.01 sec)
```

```
mysql> select * from mascotas,propietarios;
```

```
+-----+-----+-----+-----+-----+-----+-----+
| idMascota | nombre | especie | raza      | pedigree | fechaNacimiento | s
| NOMBRE      |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | kira | canina | dalmata | 1 | 1970-01-01 | h
1111A | Pepe Rodriguez |
| 1 | kira | canina | dalmata | 1 | 1970-01-01 | h
```

```
| 10 | TOM | CANINA | CANICHE | 0 | 2012-02-12 | m
1111C | Ana Rodriguez |
+-----+-----+-----+-----+-----+-----+-----+
30 rows in set (0.00 sec)
```

5.8.1.- Consultas multitas SQL1

La operación genera un conjunto de resultados con todas las combinaciones posibles entre las filas de las dos tablas, y con todas las columnas.

```
mysql> select * from mascotas CROSS JOIN propietarios;
```

```
+-----+-----+-----+-----+-----+
| idMascota | nombre | especie | raza          | pedigree |
| NOMBRE    |        |         |               |          |
+-----+-----+-----+-----+-----+
```

```
| 10 | TOM | CANINA | CANICHE | 0 | 2012-02-12 | m
1111C | Ana Rodriguez |
+-----+-----+-----+-----+-----+
30 rows in set (0.00 sec)
```

PRODUCTO CARTESIANO+ FILTRO

- Si se aplica un filtro al producto cartesiano para obtener sólo las filas en las que el campo DNI coincide , se obtendría:

```
mysql> select * from mascotas,propietarios where mascotas.propietario=propietarios.dni;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+
| idMascota | nombre | especie | raza          | pedigree | fechaNacimiento | sexo | propietario | DNI          | NOMBRE |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+
| 1 | kira | canina | dalmata | 1 | 1970-01-01 | h | 111111111A | 111111111A | Pepe |
driguez |
| 2 | terry | canina | pastor aleman | 1 | 1973-01-01 | m | 111111111A | 111111111A | Pepe |
driguez |
| 3 | lorato | ave | loro | 1 | 1973-01-01 | m | 111111111B | 111111111B | Luis |
driguez |
| 5 | dante | canina | cairn terrier | 0 | 1980-01-01 | m | 111111111B | 111111111B | Luis |
driguez |
| 6 | boris | canina | bichon maltes | 0 | 1998-01-01 | m | 111111111C | 111111111C | Ana R |
driguez |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+
5 rows in set (0.00 sec)
```


INNER JOIN

```
mysql> select * from mascotas INNER JOIN propietarios ON mascotas.propietario=propietarios.dni;
```

idMascota	nombre	especie	raza	pedigree	fechaNacimiento	sexo	propietario	DNI	NOMBRE
1	kira	canina	dalmata	1	1970-01-01	h	111111111A	111111111A	Pepe R
2	terry	canina	pastor aleman	1	1973-01-01	m	111111111A	111111111A	Pepe R
3	lorato	ave	loro	1	1973-01-01	m	111111111B	111111111B	Luis R
5	dante	canina	cairn terrier	0	1980-01-01	m	111111111B	111111111B	Luis R
6	boris	canina	bichon maltes	0	1998-01-01	m	111111111C	111111111C	Ana Ro

5 rows in set (0.00 sec)

5.8.2.-Consultas multitas SQL2

- SQL2 introduce las joins o composiciones internas, externas y productos cartesianos (también llamadas composiciones cruzadas):

1/ JOIN INTERNA

- De Equivalencia (INNER JOIN)
- Natural (NATURAL JOIN)

2/ PRODUCTO CARTESIANO (CROSS JOIN)

3/ JOIN EXTERNA

- De tabla derecha (RIGHT OUTER JOIN)
- De tabla izquierda (LEFT OUTER JOIN)
- Completa (FULL OUTER JOIN)

JOIN INTERNA De Equivalencia (INNER JOIN)

- Hay dos formas de expresar la INNER JOIN o Composiciones internas: usando la palabra reservada JOIN o separando por coma las tablas a combinar en la sentencia FROM. Con esta operación se calcula el producto cartesiano de todos los registros, después cada registro en la primera tabla es combinado con cada registro de la segunda tabla, y sólo se seleccionan aquellos registros que satisfacen las condiciones que se especifican. Los valores nulos no se combinan.
- Por ejemplo, de todos los registros de la tabla de mascotas encontrar todas las combinaciones en la tabla de propietarios en los que el DNI coincida.

```
mysql> SELECT idMascota,mascotas.nombre,raza,dni from mascotas INNER JOIN propietarios ON mascotas.propietario=propietarios.dni;
```

idMascota	nombre	raza	dni
1	kira	dalmata	111111111A
2	terry	pastor aleman	111111111A
3	lorato	loro	111111111B
5	dante	cairn terrier	111111111B
6	boris	bichon maltes	111111111C

```
5 rows in set (0.00 sec)
```

JOIN INTERNA De Equivalencia (INNER JOIN)

- Hay que tener en cuenta que si hay un animal sin propietario no saldrá en el conjunto de resultados puesto que no tiene coincidencia en el filtro. En el ejemplo introducimos un animal sin propietario:

```
mysql> insert into mascotas values(null,'cat',default,'pastor aleman',false,now(),'m',null);
Query OK, 1 row affected, 1 warning (0.05 sec)

mysql>
```

```
mysql>
mysql> SELECT idMascota,mascotas.nombre,raza,dni from mascotas INNER JOIN propietarios ON mascotas.propietario=propietarios.dni;
+-----+-----+-----+-----+
| idMascota | nombre | raza      | dni      |
+-----+-----+-----+-----+
| 1 | kira | dalmata   | 11111111A |
| 2 | terry | pastor aleman | 11111111A |
| 3 | lorato | loro      | 11111111B |
| 5 | dante | cairn terrier | 11111111B |
| 6 | boris | bichon maltes | 11111111C |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Composiciones Naturales (NATURAL JOIN)

- Es una especialización de la INNER JOIN. En este caso se comparan todas las columnas que tengan el mismo nombre en ambas tablas, la tabla resultante tiene solo una columna por cada par de columnas con el mismo nombre.
- EJEMPLO: BD Jardineria

```
mysql> describe Empleados;
```

Field	Type	Null	Key	Default	Extra
CodigoEmpleado	int(11)	NO	PRI	NULL	
Nombre	varchar(50)	NO		NULL	
Apellido1	varchar(50)	NO		NULL	
Apellido2	varchar(50)	YES		NULL	
Extension	varchar(10)	NO		NULL	
Email	varchar(100)	NO		NULL	
CodigoOficina	varchar(10)	NO		NULL	
CodigoJefe	int(11)	YES		NULL	
Puesto	varchar(50)	YES		NULL	

rows in set <0.00 sec>

```
mysql> show tables;
```

Tables_in_jardineria
clientes
detallepedidos
empleados
gamasproductos
oficinas
pagos
pedidos
productos

8 rows in set <0.00 sec>

```
mysql> describe Oficinas;
```

Field	Type	Null	Key	Default	Extra
CodigoOficina	varchar(10)	NO	PRI	NULL	
Ciudad	varchar(30)	NO		NULL	
Pais	varchar(50)	NO		NULL	
Region	varchar(50)	YES		NULL	
CodigoPostal	varchar(10)	NO		NULL	
Telefono	varchar(20)	NO		NULL	
LineaDireccion1	varchar(50)	NO		NULL	
LineaDireccion2	varchar(50)	YES		NULL	

8 rows in set <0.01 sec>

NATURAL JOIN

```
mysql> select codigoEmpleado,nombre,Oficinas.codigoOficina,ciudad from Empleados
s NATURAL JOIN Oficinas;
```

codigoEmpleado	nombre	codigoOficina	ciudad
11	Emmanuel	BCN-ES	Barcelona
12	José Manuel	BCN-ES	Barcelona
13	David	BCN-ES	Barcelona
14	Oscar	BCN-ES	Barcelona
20	Hilary	BOS-USA	Boston
21	Marcus	BOS-USA	Boston
22	Lorena	BOS-USA	Boston
26	Amy	LON-UK	Londres
27	Larry	LON-UK	Londres
28	John	LON-UK	Londres
7	Carlos	MAD-ES	Madrid
8	Mariano	MAD-ES	Madrid
9	Lucio	MAD-ES	Madrid
10	Hilario	MAD-ES	Madrid
15	Francois	PAR-FR	Paris
16	Lionel	PAR-FR	Paris
17	Laurent	PAR-FR	Paris
18	Michael	SFC-USA	San Francisco
19	Walter Santiago	SFC-USA	San Francisco
29	Kevin	SYD-AU	Sydney
30	Julian	SYD-AU	Sydney
31	Mariko	SYD-AU	Sydney
1	Marcos	TAL-ES	Talavera de la Reina
2	Ruben	TAL-ES	Talavera de la Reina
3	Alberto	TAL-ES	Talavera de la Reina
4	Maria	TAL-ES	Talavera de la Reina
5	Felipe	TAL-ES	Talavera de la Reina
6	Juan Carlos	TAL-ES	Talavera de la Reina
23	Nei	TOK-JP	Tokyo
24	Narumi	TOK-JP	Tokyo
25	Takuma	TOK-JP	Tokyo

```
31 rows in set (0.02 sec)
```

Composiciones Externas OUTER JOIN

- Las tablas relacionadas no requieren que haya una equivalencia. El registro es seleccionado para ser mostrado aunque no haya otro registro que le corresponda. Outer JOIN se subdivide dependiendo de la tabla a la cual se admitirán los registros que no tienen correspondencia, ya sean de tabla izquierda, de tabla derecha o combinación completa. Si los registros que admiten no tener correspondencia son los que aparecen en la tabla de la izquierda se llama composición de tabla izquierda o LEFT JOIN (O LEFT OUTER JOIN).

Ejemplo LEFT OUTER JOIN

```
mysql> select * from mascotas left outer JOIN PROPIETARIOS ON mascotas.dni=
propietarios.dni;
```

idMascota	nombre	especie	raza	pedigree	fechaNacimiento
sexo	dni	nose	DNI	NOMBRE	nose
h	1 kira	canina	dalmata	1	1970-01-01
h	11111111A	NULL	11111111A	Pepe Rodriguez	NULL
m	2 terry	canina	pastor aleman	1	1973-01-01
m	11111111A	NULL	11111111A	Pepe Rodriguez	NULL
m	3 lorato	ave	loro	1	1973-01-01
m	11111111B	NULL	11111111B	Luis Rodriguez	NULL
m	5 dante	canina	cairn terrier	0	1980-01-01
m	11111111B	NULL	11111111B	Luis Rodriguez	NULL

Se observa que se incluye laica que no tiene propietario.

h	7 laica	canina	boxer	0	0000-00-00
h	NULL	NULL	NULL	NULL	NULL
m	8 don	canina	dogo	0	1998-09-08
m	222333444a	NULL	NULL	NULL	NULL
m	9 fox	canina	basset hound	1	2002-09-07
m	222333444a	NULL	NULL	NULL	NULL
m	10 TOM	CANINA	CANICHE	0	2012-02-12
m	NULL	NULL	NULL	NULL	NULL
m	11 cat	canina	pastor aleman	0	2015-02-12
m	NULL	NULL	NULL	NULL	NULL

11 rows in set (0.00 sec)

Ejemplo RIGHT OUTER JOIN

- Si los registros que admiten no tener correspondencia son los que aparecen en la tabla de la derecha, se llama composición de tabla derecha o RIGHT JOIN

```
mysql> select * from mascotas right outer JOIN PROPIETARIOS ON mascotas.dni=propietarios.dni;
```

idMascota	nombre	especie	raza	pedigree	fechaNacimiento
o sexo dni	nose	DNI	NOMBRE	nose	
1	kira	canina	dalmata	1	1970-01-01
h	11111111A	NULL	11111111A	Pepe Rodriguez	NULL
2	terry	canina	pastor aleman	1	1973-01-01
m	11111111A	NULL	11111111A	Pepe Rodriguez	NULL
3	lorato	ave	loro	1	1973-01-01
m	11111111B	NULL	11111111B	Luis Rodriguez	NULL
5	dante	canina	cairn terrier	0	1980-01-01
m	11111111B	NULL	11111111B	Luis Rodriguez	NULL
6	boris	canina	bichon maltes	0	1998-01-01
m	11111111C	NULL	11111111C	Ana Rodriguez	NULL

5 rows in set (0.00 sec)

En este caso aparecen todos los propietarios, aunque no tengan una mascota.

Ejemplo FULL OUTER JOIN

Composición externa completa

- Esta operación admite registros sin correspondencia tanto para la tabla izquierda como para la derecha, e.d, animales sin propietarios y propietarios sin animales. Presenta valores nulos para los registros sin pareja.

```
mysql> select * from mascotas FULL OUTER JOIN propietarios ON mascotas.propietario=propietarios.dni;  
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that  
corresponds to your MySQL server version for the right syntax to use near 'OUTER  
JOIN propietarios ON mascotas.propietario=propietarios.dni' at line 1  
mysql>  
mysql>  
mysql>
```

Como se observa, MySql no implementa FULL OUTER JOIN.

En SQL existe el operador UNION, que añade al conjunto de resultados producidos por una SELECT, los resultados de otra SELECT

SELECT ... FROM ...

UNION [ALL]

SELECT ... FROM ...

El parámetro ALL incluye todos los registros de las dos SELECT, incluyendo los que son iguales. Si no se indica ALL, se excluyen los duplicados.

Composición externa completa, mediante UNION

- MySql simula FULL OUTER JOIN, haciendo una UNION de los resultados de un LEFT OUTER JOIN y los resultados de un RIGHT OUTER JOIN, ya que UNION, sin la opción ALL, elimina los registros duplicados, por tanto, se podría codificar:

```
mysql> select * from mascotas right outer JOIN PROPIETARIOS ON mascotas.dni=propietarios.dni UNION select * from mascotas left outer join propietarios on mascotas.dni=propietarios.dni;
```

idMascota	nombre	especie	raza	pedigree	fechaNacimiento
o sexo dni	nose DNI	NOMBRE	nose		
1	kira	canina	dalmata	1	1970-01-01
h	111111111A	NULL	111111111A	Pepe Rodriguez	NULL
2	terry	canina	pastor aleman	1	1973-01-01
m	111111111A	NULL	111111111A	Pepe Rodriguez	NULL
3	lorato	ave	loro	1	1973-01-01
m	111111111B	NULL	111111111B	Luis Rodriguez	NULL
5	dante	canina	cairn terrier	0	1980-01-01
m	111111111B	NULL	111111111B	Luis Rodriguez	NULL
6	boris	canina	bichon maltes	0	1998-01-01

5.9.- Consultas Reflexivas

- A veces, es necesario obtener información de relaciones reflexivas, por ejemplo un informe de empleados donde junto a su nombre y apellidos apareciera el nombre y apellidos de su jefe. Para ello hay que hacer un JOIN entre registros de la misma tabla:

```
mysql>
mysql> use jardineria;
Database changed
mysql> describe empleados;
```

Field	Type	Null	Key	Default	Extra
CodigoEmpleado	int(11)	NO	PRI	NULL	
Nombre	varchar(50)	NO		NULL	
Apellido1	varchar(50)	NO		NULL	
Apellido2	varchar(50)	YES		NULL	
Extension	varchar(10)	NO		NULL	
Email	varchar(100)	NO		NULL	
CodigoOficina	varchar(10)	NO	MUL	NULL	
CodigoJefe	int(11)	YES	MUL	NULL	
Puesto	varchar(50)	YES		NULL	

```
rows in set (0.03 sec)

mysql>
mysql>
```

5.9.- Consultas Reflexivas

```
mysql> select concat(emp.nombre, ' ', emp.apellido1) as NEMPLEADO, concat(jef
e.nombre, ' ', jefe.apellido1) as NJEFE from empleados EMP INNER JOIN emplea
dos JEFE ON EMP.codigoEmpleado = JEFE.codigoJefe;
+-----+-----+
| NEMPLEADO          | NJEFE          |
+-----+-----+
| Marcos Maga??a     | Ruben L??pez   |
| Ruben L??pez       | Alberto Soria  |
| Ruben L??pez       | Maria Sol??s   |
| Alberto Soria      | Felipe Rosas   |
| Alberto Soria      | Juan Carlos Ortiz |
| Alberto Soria      | Carlos Soria   |
| Alberto Soria      | Emmanuel Maga??a |
| Alberto Soria      | Francois Fignon |
| Alberto Soria      | Michael Bolton |
| Alberto Soria      | Hilary Washington |
| Alberto Soria      | Nei Nishikori  |
| Alberto Soria      | Amy Johnson    |
| Alberto Soria      | Kevin Fallmer  |
| Carlos Soria       | Mariano L??pez |
| Carlos Soria       | Lucia Gonzalez |
+-----+-----+
| Amy Johnson        | John Walton    |
| Kevin Fallmer      | Julian Bellinelli |
| Kevin Fallmer      | Mariko Kishi    |
+-----+-----+
30 rows in set (0.00 sec)
```

- Se ha usado la tabla Empleados dos veces, una con un alias **emp** que representa los Empleados como Subordinados y otra con alias **jefe** que representa los Empleados como Jefes. Ambas tablas (aunque en realidad es una sólo) se unen en una JOIN a través de la relación CodigoEmpleado y CodigoJefe.
- Además el primer campo seleccionado es la concatenación del nombre y apellido del Empleado al que se le da un alias (**NEMPLEADO**) y lo mismo con (**NJEFE**).
- Se observa que en esta query no aparecen empleados sin jefe, ya que se ha utilizado un INNER JOIN.

5.10.- Consultas con Tablas Derivadas

- Las consultas con Tablas Derivadas o **inline views**, son aquellas que utilizan sentencias select en la cláusula FROM en lugar de nombres de tablas:

```
mysql> SELECT * FROM (SELECT CodigoEmpleado, Nombre FROM Empleados
-> WHERE CodigoOficina='TAL-ES') as tabla_derivada;
```

CodigoEmpleado	Nombre
1	Marcos
2	Ruben
3	Alberto
4	Maria
5	Felipe
6	Juan Carlos

```
6 rows in set (0.00 sec)
mysql>
```

La tabla derivada (select CodigoEmpleado, nombre FROM Empleados) tiene un alias 'tabla_derivada'. Es como una especie de tabla temporal resultado de ejecutar la consulta.

5.10.- Consultas con Tablas Derivadas

- Por ejemplo, en la BD jardinería, si se quiere obtener el importe del pedido de menor coste de todos los pedidos. Primero se obtiene el total de todos los pedidos y luego el pedido de menor coste con la función de columna MIN:

```
mysql>
mysql> select SUM(Cantidad*PrecioUnidad) as total,CodigoPedido
-> FROM DetallePedidos
-> Group BY CodigoPedido;
```

```
17888.00 104
15065.00 105
10776.00 106
32216.00 107
6660.00 108
5553.00 109
14399.00 110
1064.00 111
151.00 128
89 rows in set (0.00 sec)
```

```
mysql> SELECT MIN(total) FROM(
-> select SUM(Cantidad*PrecioUnidad) as total,CodigoPedido
-> FROM DetallePedidos
-> Group BY CodigoPedido) AS TotalPedidos;
```

```
+-----+
| MIN(total) |
+-----+
| 4.00 |
+-----+
1 row in set (0.00 sec)
```

```
mysql>
```

TotalPedidos es el alias de la tabla derivada, es obligatorio utilizar siempre un alias.

5.10.- Consultas con Tablas Derivadas

- Para determinar cuál es el código de pedido que le corresponde al importe menor de todos los pedidos:

```
mysql> select codigopedido from detallepedidos group by  
        codigopedido having sum(cantidad*preciounidad)=  
(select min(total) from (select sum(cantidad*preciounidad) as  
        total, codigopedido from detallepedidos group by  
        codigopedido) as tablad);
```

```
mysql> select codigopedido from detallepedidos group by codigopedido having  
        sum(cantidad*preciounidad) =(select min(total) from (select sum(cantidad*pr  
eciounidad) as total, codigopedido from detallepedidos group by codigopedido  
) as tablad);  
+-----+  
| codigopedido |  
+-----+  
|          42 |  
+-----+  
1 row in set (0.02 sec)
```


Actividad 5.1: BD Jardinería

- Codifica en mySql sentencias para obtener la siguiente información:
 - 1.- El código de oficina y la ciudad donde hay oficinas.
 - 2.- Cuántos empleados hay en la compañía?
 - 3.- Cuántos Clientes tiene cada país?

Actividad 5.1

- 4.- Cuál fue el pago medio en 2008 (usar la función YEAR)
- 5.- Cuántos pedidos están en cada estado ordenado descendente por el número de pedidos?
- 6.- Precio del producto más caro y del más barato
- 7.- El nombre del producto más caro.

Actividad 5.2

- 1.- Visualizar el nombre del representante de ventas de cada cliente.
- 2.- Nombre del cliente con mayor límite de crédito
- 3.- Nombre, apellido1 y cargo de los empleados que no representen a ningún cliente

Actividad 5.3

- 1.-Listado en el que figure el nombre de cada cliente, y el nombre y apellidos de su representante de ventas
- 2.- Mostrar el nombre de los clientes que no hayan realizado pagos junto con el nombre de sus representantes de ventas
- 3.-Listado de las ventas totales de los productos que hayan facturado más de 3000 euros. Se mostrará el nombre, unidades vendidas, total facturado y total facturado con impuestos (18% IVA)
- 4/ Listar la dirección de las oficinas que tengan clientes en Fuenlabrada.

Actividad 5.4

- Visualizar un listado en el que figure el nombre y apellido de cada empleado junto con el de su jefe

Actividad 5.5

- Visualizar la información de los empleados que pertenecen a la oficina 'TAL_ES'.
Utilizando una tabla derivada:

Actividad 5.6

- Sacar el importe del pedido de menor coste de todos los pedidos.

```
mysql> describe detallepedidos;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| CodigoPedido   | int(11)       | NO   | PRI | NULL    |       |
| CodigoProducto | varchar(15)   | NO   | PRI | NULL    |       |
| Cantidad       | int(11)       | NO   |     | NULL    |       |
| PrecioUnidad   | decimal(15,2) | NO   |     | NULL    |       |
| NumeroLinea    | smallint(6)   | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.05 sec)
```

Actividad 5.7

1/ Cliente que hizo el pedido de mayor cuantía

```
mysql> select nombrecliente from clientes c, pedidos p, detallepedidos dp
where c.codigocliente=p.codigocliente and p.codigopedido=dp.codigopedido
group by dp.codigopedido having sum(dp.cantidad*dp.preciounidad)>=all (select
sum(cantidad*preciounidad) from detallepedidos group by codigopedido);
```

```
mysql> select nombrecliente,c.codigocliente from clientes c, pedidos p, det
allepedidos dp where c.codigocliente=p.codigocliente and p.codigopedido=dp.
codigopedido group by dp.codigopedido having sum(dp.cantidad*dp.preciounida
d)>=all(select sum(cantidad*preciounidad) from detallepedidos group by codi
gopedido);
+-----+-----+
| nombrecliente | codigocliente |
+-----+-----+
| Gerudo Valley |             4 |
+-----+-----+
1 row in set (0.00 sec)
```


Actividad 5.7

- Con tablas derivadas:

```
mysql> SELECT TotalPedidos.CodigoCliente, nombreCliente FROM
-> (SELECT codigoPedido, codigoCliente, sum(Cantidad*precioUnidad) AS Total
-> FROM Pedidos NATURAL JOIN DetallePedidos GROUP BY codigoPedido, codigoCl
ente) TotalPedidos
-> INNER JOIN Clientes ON Clientes.codigoCliente=TotalPedidos.codigoCliente
-> WHERE Total=(SELECT MAX(Total) FROM (SELECT codigoPedido,codigoCliente,
-> sum(Cantidad*precioUnidad) as Total FROM Pedidos NATURAL JOIN
-> detallePedidos GROUP BY codigoPedido, codigoCliente) TotalPedidos);
+-----+-----+
| CodigoCliente | nombreCliente |
+-----+-----+
|          4   | Gerudo Valley |
+-----+-----+
1 row in set (0.05 sec)
```