



Tema 5 : Diseño físico de Bases de Datos. Realización de Consultas. DML



ÍNDICE

5.1.- El lenguaje DML

5.2.- Sentencia Select

5.3.- Consultas básicas

5.4.- Filtros

5.4.1.- Filtros con operadores de pertenencia a conjuntos

5.4.2.- Filtros con operador de rango

5.4.3.- Filtros con test de valor nulo

5.4.4.- Filtros con test de patrón

5.4.5.- Filtros por límite de número de registros

5.4.6.- Filtros por límite de número de registros

5.5.- Ordenación

5.6.- Consultas de resumen: Funciones columna

5.6.1.- Filtros de Grupos

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 multitaslas

5.8.1.- Consultas multitaslas SQL1

5.8.2.-Consultas multitaslas SQL2

5.9.- Consultas Reflexivas

5.10.- Consultas con Tablas Derivadas

Actividades 5.1 => 5.7

5.1.- El lenguaje DML

- Las sentencias DML del lenguaje SQL son las siguientes:
 - La sentencia **SELECT**, que se utiliza para extraer información de la BD, ya sea de una tabla o de varias.
 - La sentencia **INSERT**, para insertar uno o varios registros en una tabla.
 - La sentencia **DELETE**, que borra registros de una tabla.
 - La sentencia **UPDATE**, que modifica registros de una tabla.
- Cualquier ejecución de un comando en un SGBD es una consulta o Query, es decir, cualquier orden o petición que se realiza al SGBD para realizar una operación determinada ya sea de consulta, inserción, borrado o actualización es una Query o consulta.

5.2.- Sentencia Select

```
SELECT
[ALL | DISTINCT ]
select_expr [, select_expr ...]
[FROM table_references
[WHERE where_condition]
[GROUP BY {col_name | expr | position}
[ASC | DESC], ... [WITH ROLLUP]]
[HAVING where_condition]
[ORDER BY {col_name | expr | position}
[ASC | DESC], ...]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name' export_options
| INTO DUMPFILE 'file_name'
| INTO @var_name [, @var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]]
```

5.3.- Consultas básicas

- SELECT es la sentencia más versátil de todo el SQL.
- El formato básico para hacer una consulta es:

```
SELECT [DISTINCT ] select_expr [, select_expr ...]  
[FROM tabla]
```

*Donde **select_expr**:*

Nombre _columna [AS alias] | * | expresión

El parámetro opcional DISTINCT *fuerza a que solo se muestren los registros con valores distintos, es decir, que se supriman las repeticiones*

Ejemplos básicos BD veterinario

```
mysql> describe mascotas;
```

Field	Type	Null	Key	Default	Extra
idMascota	int(11)	NO	PRI	NULL	auto_increment
nombre	varchar(30)	NO	UNI	NULL	
especie	varchar(20)	NO		canina	
raza	varchar(20)	YES		NULL	
pedigree	tinyint(1)	YES		NULL	
fechaNacimiento	date	YES		NULL	
sexo	char(1)	YES		NULL	

```
7 rows in set (0.01 sec)
```

```
mysql> insert into mascotas values(null,'donna','canina','fox terrier',false,'1980-02-07','h');
Query OK, 1 row affected (0.02 sec)
```

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

idMascota	nombre	especie	raza	pedigree	fechaNacimiento	sexo
1	kira	canina	dalmata	1	1967-02-07	h
2	terry	canina	pastor aleman	0	1974-02-07	m
3	lorato	ave	loro	0	1972-02-07	h
4	donna	canina	fox terrier	0	1980-02-07	h

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

Ejemplos básicos BD veterinario

```
mysql>
mysql> insert into mascotas values(null,'dante',default,'Cairn terrier',true,'1988-05-08','m');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select nombre AS nombreAnimal, raza from mascotas where especie='canina'

+-----+-----+
| nombreAnimal | raza          |
+-----+-----+
| kira         | dalmata      |
| terry        | pastor aleman|
| donna        | fox terrier  |
| dante        | Cairn terrier|
| boris        | bichon maltes|
+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select nombre, concat(especie,'/',raza) as Animal from mascotas;

+-----+-----+
| nombre | Animal              |
+-----+-----+
| kira   | canina/dalmata     |
| terry  | canina/pastor aleman|
| lorato | ave/loro            |
| donna  | canina/fox terrier  |
| dante  | canina/Cairn terrier|
| boris  | canina/bichon maltes|
+-----+-----+
6 rows in set (0.00 sec)
```


Ejemplos básicos BD veterinario

```
mysql> select nombre from mascotas where pedigree=true;
+-----+
| nombre |
+-----+
| kira   |
| dante  |
| boris  |
+-----+
3 rows in set (0.00 sec)
```

```
mysql> select nombre, concat(especie,raza) as Animal,idMascota*9 from mascotas;
+-----+-----+-----+
| nombre | Animal          | idMascota*9 |
+-----+-----+-----+
| kira   | caninadalmata   | 9           |
| terry  | caninapastor aleman | 18          |
| lorato | aveloro         | 27          |
| donna  | caninafox terrier | 36          |
| dante  | caninaCairn terrier | 45          |
| boris  | caninabichon maltes | 54          |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> select especie from mascotas;
+-----+
| especie |
+-----+
| canina  |
| canina  |
| ave     |
| canina  |
| canina  |
| canina  |
+-----+
6 rows in set (0.00 sec)

mysql> select DISTINCT especie from mascotas;
+-----+
| especie |
+-----+
| canina  |
| ave     |
+-----+
2 rows in set (0.43 sec)

mysql> _
```

Ejemplos básicos BD veterinario

- Calcular la edad aproximada de cada mascota

```
mysql> select NOMBRE, (YEAR(NOW()) - YEAR(fechaNacimiento)) AS Edad from mascotas;
```

NOMBRE	Edad
kira	47
terry	40
lorato	42
donna	34
dante	26
boris	15

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

5.4.- Filtros

- Son condiciones que cualquier gestor de BD interpreta para seleccionar registros y mostrarlos como resultado de la consulta. En SQL la palabra clave es **WHERE**.

```
SELECT [DISTINCT ] select_expr [, select_expr ...]  
[FROM tabla] [WHERE filtro]
```

Filtro es una expresión que indica la condición o condiciones que deben satisfacer los registros para ser seleccionados.

```
mysql> select * from mascotas where nombre='kira';  
+-----+-----+-----+-----+-----+-----+-----+  
| idMascota | nombre | especie | raza   | pedigree | fechaNacimiento | sexo |  
+-----+-----+-----+-----+-----+-----+-----+  
| 1 | kira | canina | dalmata | 1 | 1967-02-07 | h |  
+-----+-----+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

Expresiones

- Ejemplo: La fecha de hoy menos 20 años:

```
mysql> select date_sub(now(), interval 20 year);
+-----+
| date_sub(now(), interval 20 year) |
+-----+
| 1994-12-31 18:44:28                |
+-----+
1 row in set (0.00 sec)
```

- Los elementos que pueden formar parte de las expresiones son:
 - ✓ Operandos (constantes o variables)
 - ✓ Operadores Aritméticos (+, -, *, /, %)
 - ✓ Operadores Relacionales (<, >, <=, >=, =, <>)
 - ✓ Operadores Lógicos (AND, OR, NOT)
 - ✓ () para dar máxima prioridad a la expresión
 - ✓ Funciones: date_add, concat, left, right ...

Expresiones

```
mysql> select left(raza,4) from mascotas;
+-----+
| left(raza,4) |
+-----+
| daln         |
| past         |
| loro         |
| fox          |
| Cair         |
| bich         |
| boxe         |
+-----+
7 rows in set (0.00 sec)
```

```
mysql> select (10)=10 AND 0<=1) +7;
+-----+
| (10)=10 AND 0<=1) +7 |
+-----+
| 8                       |
+-----+
1 row in set (0.00 sec)
```

```
mysql> set @var="hola que tal";
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select left(@var,4);
```

```
+-----+
| left(@var,4) |
+-----+
```

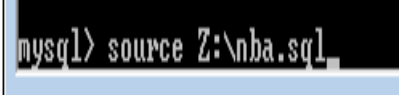
```
+-----+
| hola        |
+-----+
```

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

```
mysql> select (7+3)*1 and null;
+-----+
| (7+3)*1 and null |
+-----+
| NULL              |
+-----+
1 row in set (0.00 sec)

mysql> select (7+3)*1 or null;
+-----+
| (7+3)*1 or null  |
+-----+
| 1                 |
+-----+
1 row in set (0.00 sec)
```

Ejemplos NBA

- Ejemplos de construcción de filtros para una base de datos de jugadores de la liga americana de baloncesto (NBA). 
- Crear la BD NBA con el fichero nba.sql

```
mysql> show tables;
+-----+
| Tables_in_nba |
+-----+
| equipos        |
| estadisticas   |
| jugadores      |
| partidos       |
+-----+
4 rows in set (0.00 sec)
```

```
mysql> describe jugadores;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| codigo     | int(11)       | NO   | PRI  | NULL    |       |
| Nombre     | varchar(30)   | YES  |      | NULL    |       |
| Procedencia | varchar(20)   | YES  |      | NULL    |       |
| Altura     | varchar(4)    | YES  |      | NULL    |       |
| Peso       | int(11)       | YES  |      | NULL    |       |
| Posicion   | varchar(5)    | YES  |      | NULL    |       |
| Nombre_equipo | varchar(20)  | YES  | MUL  | NULL    |       |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.01 sec)
```

Ejemplos NBA

Seleccionar los nombres de los jugadores de los Lakers:

```
mysql> select nombre from jugadores where Nombre_equipo='Lakers';
```

nombre
Trevor Ariza
Kobe Bryant
Andrew Bynum
Jordan Farmar
Derek Fisher
Pau Gasol
Didier Ilunga-Mbenga
Coby Karl
Chris Mihm
Ira Newble
Lamar Odom
Vladimir Radmanovic
Ronny Turiaf
Sasha Vujacic
Luke Walton

```
15 rows in set (0.04 sec)
```

- Visualizar todos los nombres de equipos:

```
mysql> select DISTINCT Nombre_equipo from jugadores;
```

Nombre_equipo
76ers
Bobcats
Bucks
Bulls
Cavaliers
Celtics
Clippers

Ejemplos NBA

- Selecciona los jugadores españoles de los Lakers:

```
mysql> select nombre from jugadores where Nombre_equipo = 'Lakers' and Procedencia='spain';
```

nombre
Pau Gasol

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

Selecciona los jugadores españoles, eslovenos o serbios de los Lakers:

```
mysql> select nombre, procedencia, nombre_equipo from jugadores where nombre_equipo='Lakers' and (procedencia = 'spain' or procedencia='slovenia' or procedencia='serbia');
```

nombre	procedencia	nombre_equipo
Pau Gasol	Spain	Lakers
Sasha Vujacic	Slovenia	Lakers

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


5.4.1.- Filtros con operadores de pertenencia a conjuntos

Nombre_columna **IN** (valor1, valor2, ...)

- Este operador permite comprobar si una columna tiene un valor igual que cualquiera de los que están incluidos dentro del paréntesis.
- Por ejemplo, Selecciona los jugadores españoles eslovenos o serbios de los Lakers :

```
mysql> select nombre, procedencia, nombre_equipo from jugadores where nombre_equipo='Lakers' and procedencia IN ('spain','slovenia','serbia');
+-----+-----+-----+
| nombre      | procedencia | nombre_equipo |
+-----+-----+-----+
| Pau Gasol   | Spain      | Lakers        |
| Sasha Uujacic | Slovenia   | Lakers        |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

5.4.2.- Filtros con operador de rango

- El operador de rango **BETWEEN** permite seleccionar los registros que estén incluidos en un rango.

Nombre_columna BETWEEN valor1 AND valor2

- Por ejemplo, seleccionar a los jugadores de la NBA cuyo peso está entre 275 y 300 libras:

```
mysql> select nombre, peso, nombre_equipo from jugadores where peso between 275
and 300;
```

nombre	peso	nombre_equipo
Darko Milicic	275	Grizzlies
Andrew Bynum	285	Lakers
Eddy Curry	285	Knicks
Jerome James	285	Knicks
Glen Davis	289	Celtics
Scot Pollard	278	Celtics
Zaza Pachulia	280	Hawks
DeSagana Diop	280	Nets
Kyrylo Fesenko	288	Jazz
David Harrison	280	Pacers

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

5.4.2.- Filtros con operador de rango

- Sabiendo que una libra equivale a 0.4535 Kgr. Seleccionar los jugadores de la NBA cuyo peso está entre 275 y 300 libras, mostrando el peso en libras y Kgr:

```
mysql> select nombre, peso as libras, peso*0.4535 as kg, nombre_equipo from jugadores where peso between 275 and 300;
```

nombre	libras	kg	nombre_equipo
Darko Milicic	275	124.7125	Grizzlies
Andrew Bynum	285	129.2475	Lakers
Eddy Curry	285	129.2475	Knicks
Jerome James	285	129.2475	Knicks
Glen Davis	289	131.0615	Celtics
Scot Pollard	278	126.0730	Celtics
Zaza Pachulia	280	126.9800	Hawks
DeSagana Diop	280	126.9800	Nets
Kyrylo Fesenko	288	130.6080	Jazz
David Harrison	280	126.9800	Pacers

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

Seleccionar el peso en Kgr de los jugadores de la NBA que pesen entre 125 y 140 kgr.

```
mysql> select nombre, peso as libras, peso*0.4535 as kg, nombre_equipo from jugadores where peso*0.4535 between 125 and 140;
```

nombre	libras	kg	nombre_equipo
Andrew Bynum	285	129.2475	Lakers
Eddy Curry	285	129.2475	Knicks
Jerome James	285	129.2475	Knicks
Glen Davis	289	131.0615	Celtics
Scot Pollard	278	126.0730	Celtics
Zaza Pachulia	280	126.9800	Hawks
DeSagana Diop	280	126.9800	Nets
Kyrylo Fesenko	288	130.6080	Jazz
David Harrison	280	126.9800	Pacers

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

5.4.3.- Filtros con test de valor nulo

- Los operadores IS e **IS NOT** permiten verificar si un campo es o no es nulo respectivamente.
- Por ejemplo, determinar los jugadores cuya procedencia es desconocida:

```
mysql> select nombre, peso, nombre_equipo from jugadores where procedencia IS null;
+-----+-----+-----+
| nombre      | peso | nombre_equipo |
+-----+-----+-----+
| Kelenna Azubuike | 220 | Warriors      |
| Matt Barnes     | 226 | Warriors      |
| Marco Belinelli  | 192 | Warriors      |
| Andris Biedrns   | 230 | Warriors      |
| Austin Croshere  | 235 | Warriors      |
| Baron Davis      | 215 | Warriors      |
| Monta Ellis      | 177 | Warriors      |
| Al Harrington    | 250 | Warriors      |
| Stephen Jackson  | 218 | Warriors      |
| Patrick O'Bryant | 250 | Warriors      |
| Kosta Perovic    | 240 | Warriors      |
| Mickael Pietrus  | 215 | Warriors      |
| C.J. Watson      | 180 | Warriors      |
| Brandon Wright   | 205 | Warriors      |
+-----+-----+-----+
14 rows in set (0.00 sec)
```

La query contraria sacaría el resto de jugadores:

```

+-----+-----+-----+
| Alando Tucker | 205 | Suns          |
+-----+-----+-----+
418 rows in set (0.00 sec)

mysql>
mysql> select nombre, peso, nombre_equipo from jugadores where procedencia IS NOT NULL;
```

5.4.4.- Filtros con test de patrón

- Seleccionan los registros que cumplan una serie de características. Se pueden usar los caracteres comodines **%** y **_** para buscar una cadena de caracteres.
- Por ejemplo, seleccionar los jugadores cuya procedencia incluye la palabra 'monte':

```
mysql> select nombre, procedencia, nombre_equipo from jugadores where procedencia like '%monte%';
```

nombre	procedencia	nombre_equipo
Darko Milicic	Serbia Montenegro	Grizzlies
Vladimir Radmanovic	Serbia Montenegro	Lakers
Nenad Krstic	Serbia Montenegro	Nets

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

Por ejemplo, seleccionar los jugadores cuyo nombre empieza por 'P':

```
mysql> select nombre, procedencia, nombre_equipo from jugadores where nombre like 'P%';
```

nombre	procedencia	nombre_equipo
Paul Davis	Michigan	Clippers
Pau Gasol	Spain	Lakers
Primož Brezec	Slovenia	Raptors
P.J.Brown	Louisiana State	Celtics
Paul Pierce	Kansas	Celtics
Peja Stojakovic	Serbia	Hornets
Pat Garrity	Notre Dame	Magic
Paul Millsap	Loisiana Tech	Jazz
Patrick O'Bryant	NULL	Warriors

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

5.4.4.- Filtros con test de patrón

Visualizar la información de los equipos que tengan un nombre de 6 caracteres y empiecen por L y terminen por S:

```
mysql> select * from equipos where nombre like 'L____s';
```

Nombre	Ciudad	Conferencia	Division
Lakers	Los Angeles	West	Pacific

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

Equipos que en su nombre tengan una 'a' como segundo carácter:

```
mysql> select * from equipos where nombre like '_a%';
```

Nombre	Ciudad	Conferencia	Division
Cavaliers	Cleveland	East	Central
Hawks	Atlanta	East	SouthEast
Jazz	Utah	West	NorthWest
Lakers	Los Angeles	West	Pacific
Magic	Orlando	East	SouthEast
Mavericks	Dallas	West	SouthWest
Pacers	Indiana	East	Central
Raptors	Toronto	East	Atlantic
Warriors	Golden State	West	Pacific

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

5.4.5.- Filtros por límite de número de registros

- Este tipo de filtros no es estándar y su funcionamiento varía con el SGBD. Consiste en limitar el número de registros devuelto por una consulta.

[LIMIT [desplazamiento,] nfilas]

- Nfilas especifica el número de filas a devolver y desplazamiento especifica a partir de qué fila se empieza a contar.
- Por ejemplo, visualizar las 4 primeras filas de la tabla jugadores:

```
mysql> select nombre,nombre_equipo from jugadores limit 4;
```

nombre	nombre_equipo
Corey Brever	Timberwolves
Greg Buckner	Timberwolves
Michael Doleac	Timberwolves
Randy Foye	Timberwolves

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

5.4.5.- Filtros por límite de número de registros

- Visualizar 4 jugadores empezando desde el tercero:

```
mysql> select nombre,nombre_equipo from jugadores limit 2,4;
```

nombre	nombre_equipo
Michael Doleac	Timberwolves
Randy Foye	Timberwolves
Ryan Gomes	Timberwolves
Marko Jaric	Timberwolves

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

- Visualizar el nombre de 2 jugadores cuyo nombre contenga 'Brian' :

```
mysql> select nombre from jugadores where nombre like '%brian%' limit 2;
```

nombre
Brian Cardinal
Brian Scalabrino

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


5.5.- Ordenación

- Para mostrar ordenados un conjunto de registros se utiliza la cláusula ORDER BY.

```
SELECT [DISTINCT ] select_expr [, select_expr ...]  
[FROM tabla  
[WHERE filtro]  
[ORDER BY {nombre_columna | expr | position} [ASC | DESC], ... ]
```

Esta cláusula permite ordenar el conjunto de resultados de forma ascendente (ASC) o descendente (DESC) por una o varias columnas. Si no se especifica nada, por defecto es ASC. La columna por la que se quiere ordenar puede especificarse por su nombre, posición o por una expresión numérica.

5.5.- Ordenación

- Por ejemplo, dada la tabla de equipos:

```
mysql> describe equipos;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Nombre | varchar(20) | NO   | PRI | NULL    |       |
| Ciudad | varchar(20) | YES  |     | NULL    |       |
| Conferencia | varchar(4) | YES  |     | NULL    |       |
| Division | varchar(9) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

- Obtener los equipos de la conferencia oeste de la NBA ordenados por nombre:

```
mysql> select * from equipos where Conferencia='west' ORDER BY nombre;
+-----+-----+-----+-----+
| Nombre | Ciudad | Conferencia | Division |
+-----+-----+-----+-----+
| Clippers | Los Angeles | West | Pacific |
| Grizzlies | Memphis | West | SouthWest |
| Hornets | New Orleans | West | SouthWest |
| Jazz | Utah | West | NorthWest |
| Kings | Sacramento | West | Pacific |
| Lakers | Los Angeles | West | Pacific |
| Mavericks | Dallas | West | SouthWest |
| Nuggets | Denver | West | NorthWest |
| Rockets | Houston | West | SouthWest |
| Spurs | San Antonio | West | SouthWest |
| Suns | Phoenix | West | Pacific |
| Supersonics | Seattle | West | NorthWest |
| Timberwolves | Minnesota | West | NorthWest |
| Trail Blazers | Portland | West | NorthWest |
| Warriors | Golden State | West | Pacific |
+-----+-----+-----+-----+
15 rows in set (0.00 sec)
```

5.6.- Consultas de resumen: Funciones columna

- Permiten extraer información calculada de varios conjuntos de registros.
- Por ejemplo, obtener el número de registros de la tabla jugadores, o el peso máximo o medio.

```
mysql> select max(peso) from jugadores;
+-----+
| max(peso) |
+-----+
|      325 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT count(*) from jugadores;
+-----+
| count(*) |
+-----+
|      432 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> select avg(peso) from jugadores;
+-----+
| avg(peso) |
+-----+
|  223.1181 |
+-----+
1 row in set (0.00 sec)
```

5.6.- Funciones columna

SUM (Expresión) # Suma los valores indicados en el argumento

AVG (Expresión) # Calcula la media de los valores

MIN (Expresión) # Calcula el mínimo

MAX (Expresión) # Calcula el máximo

COUNT (nbColumna) # Cuenta el número de valores de una columna (excepto los nulos)

COUNT (*) # Cuenta el número de valores de una fila (incluyendo los nulos)

```
mysql> #altura del jugador más bajo de la NBA
mysql> select min(altura) from jugadores;
+-----+
| min(altura) |
+-----+
| 6-2         |
+-----+
1 row in set (0.01 sec)
```

```
mysql> #Cuanto pesan de media los jugadores de los Lakers
mysql> select avg(peso) from jugadores where nombre_equipo='Lakers';
+-----+
| avg(peso) |
+-----+
| 230.0000  |
+-----+
1 row in set (0.00 sec)
```

```
mysql> #Numero de jugadores de los lakers
mysql> select count(*) from jugadores where nombre_equipo='lakers';
+-----+
| count(*) |
+-----+
| 15       |
+-----+
1 row in set (0.01 sec)
```

5.6.- Agrupaciones de Registros

- Con las consultas de resumen se pueden realizar agrupaciones de registros, es decir, conjuntos de registros que tienen una o varias columnas con el mismo valor. A este grupo de registros se le puede aplicar una función de columna para realizar determinados cálculos, por ejemplo, contarlos:

```
mysql> #Divisiones distintas de los equipos
mysql> select DISTINCT division from equipos;
```

division
Atlantic
SouthEast
Central
Pacific
SouthWest
NorthWest

6 rows in set (0.00 sec)

```
mysql> select division,nombre from equipos group by division,nombre;
```

division	nombre
Atlantic	76ers
Atlantic	Celtics
Atlantic	Knicks
Atlantic	Nets
Atlantic	Raptors
Central	Bucks
Central	Bulls
Central	Cavaliers
Central	Pacers
Central	Pistons
NorthWest	Jazz
NorthWest	Nuggets
NorthWest	Supersonics

5.6.- Agrupaciones de Registros

```
SELECT [DISTINCT] select_expr [, select_expr ...]  
[FROM tabla]  
[WHERE filtro]  
[GROUP BY {expr [,expr] ...} ]  
[ORDER BY {nombre_columna | expr | position} [ASC | DESC], ...]
```

Se observa que GROUP BY va justo antes que la cláusula ORDER BY.

```
mysql> # visualizar para cada equipo su nombre, el numero de jugadores y el peso  
      medio  
mysql> SELECT nombre_equipo, COUNT(*) num_jugadores, avg(peso) 'peso medio' FROM  
      jugadores group by nombre_equipo;
```

nombre_equipo	num_jugadores	peso medio
76ers	14	220.8571
Bobcats	14	219.9286
Bucks	14	225.7143
Bulls	15	215.9333
Cavaliers	14	227.4286
Celtics	15	222.0667
Clippers	15	219.0667
Grizzlies	14	221.0000
Hawks	13	215.3846
Heat	16	221.8125

5.6.- Agrupaciones de Registros

```
mysql> #nombre del equipo y peso máximo del mismo
mysql> select nombre_equipo,max(peso) from jugadores group by nombre_equipo;
```

nombre_equipo	max(peso)
76ers	250
Bobcats	266
Bucks	260
Bulls	270
Cavaliers	260
Celtics	289
Clippers	270
Grizzlies	275
Hawks	280
Heat	261
Hornets	261

```
mysql> #Numero de equipos
mysql> select count(distinct nombre_equipo) from jugadores;
```

count(distinct nombre_equipo)
30

1 row in set (0.00 sec)

5.6.- Agrupaciones de Registros

```
mysql>
mysql> #Cuántos equipos tiene cada conferencia en la nba
mysql> select conferencia,count(*) from equipos GROUP BY conferencia;
+-----+-----+
| conferencia | count(*) |
+-----+-----+
| East        | 15       |
| West        | 15       |
+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql>
mysql> #Cuánto pesan de media los jugadores de España
mysql> select procedencia, avg(peso) from jugadores where procedencia='Spain' Group BY procedencia;
+-----+-----+
| procedencia | avg(peso) |
+-----+-----+
| Spain       | 208.6000  |
+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> #Cuánto pesan de media los jugadores de España, Francia e Italia
mysql> select procedencia, avg(peso) from jugadores where procedencia IN ('Spain', 'France', 'Italy') Group BY procedencia;
+-----+-----+
| procedencia | avg(peso) |
+-----+-----+
| France      | 219.2500  |
| Italy       | 250.0000  |
| Spain       | 208.6000  |
+-----+-----+
3 rows in set (0.00 sec)
```


5.6.- Agrupaciones de Registros

- Para cada agrupación, se ha seleccionado también el nombre de la columna por la cual se agrupa. Esto no es posible si no se incluye el GROUP BY.

```
mysql> SELECT COUNT(*), CONFERENCIA FROM EQUIPOS GROUP BY CONFERENCIA;
```

COUNT(*)	CONFERENCIA
15	East
15	West

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

Cuando agrupamos, en la línea select, sólo pueden aparecer campos por los que agrupamos, o funciones de columna.

En el **having** no podremos hacer referencia a campos por los que no agrupamos.

Se puede agrupar por campos que no son mostrados en la línea **select**.

5.6.- SUBCONSULTAS

```
mysql> select nombre,peso from jugadores where peso = (select max(peso) from jugadores);
```

nombre	peso
Shaquille O'Neal	325

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

```
mysql> #nombre del equipo,nombre del jugador y peso del mas pesado de cada equipo
mysql> select nombre_equipo, nombre, peso from jugadores j1 where peso IN (select max(peso) from jugadores j2 group by nombre_equipo having j1.nombre_equipo=j2.nombre_equipo);
```

nombre_equipo	nombre	peso
Timberwolves	Chris Richard	270
Clippers	Paul Davis	270
Grizzlies	Darko Milicic	275
Lakers	Andrew Bynum	285
Raptors	Primož Brezec	255
Raptors	Basho Nestorovic	255

5.6.1.- Filtros de Grupos

- Los filtros de grupos deben realizarse mediante el uso de la cláusula HAVING puesto que WHERE actúa antes de agrupar los registros. Es decir, si se desea filtrar resultados calculados mediante agrupaciones se debe usar la siguiente sintaxis:

```
SELECT [ DISTINCT ] select_expr [, select_expr ...]  
[FROM tabla]  
[WHERE filtro]  
[GROUP BY {expr [, expr]...}  
[HAVING filtro_grupos]  
[ORDER BY {nombre_columna | expr | posición} [ASC | DESC], ...]
```

5.6.1.- Filtros de Grupos

```
mysql> select avg(peso), nombre_equipo from jugadores group by nombre_equipo having avg(peso)>228;
```

avg(peso)	nombre_equipo
230.0714	Jazz
235.4667	Knicks
230.0000	Lakers
228.8462	Suns
229.6923	Wizards

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

```
mysql> #Seleccionar los equipos de la NBA que tienen jugadores españoles
mysql> select nombre_equipo, count(*) from jugadores where PROCEDENCIA='Spain' GROUP BY nombre_equipo HAVING COUNT(*) order by nombre_equipo DESC;
```

nombre_equipo	count(*)
Trail Blazers	1
Raptors	2
Lakers	1
Grizzlies	1

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

```
mysql> #Seleccionar qué equipos de la NBA tiene más de un jugador español
mysql> select nombre_equipo, count(*) from jugadores where PROCEDENCIA='Spain' GROUP BY nombre_equipo HAVING COUNT(*)>1;
```

nombre_equipo	count(*)
Raptors	2

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

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
Ime 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 jugadores 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 {expresión [,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:  
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 multitablas SQL1

La operación CROSS JOIN genera un conjunto de resultados con todas las combinaciones posibles entre las filas de las dos tablas, y con todas las columnas. Se trata de un error, ya que se combina cada fila de una tabla con todas las filas de la otra tabla, aunque no tengan nada en común.

```
mysql> select * from mascotas CROSS JOIN propietarios;
+-----+-----+-----+-----+-----+
| idMascota | nombre | especie | raza | pedigree |
| NOMBRE | | | | |
+-----+-----+-----+-----+-----+
```

```

1 |          10 | TOM       | CANINA   | CANICHE   |          0 | 2012-02-12 |
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 Riguez

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

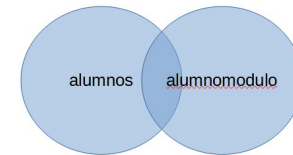
5 rows in set (0.00 sec)

5.8.2.-Consultas multitaslas SQL2

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

1/ JOIN INTERNA

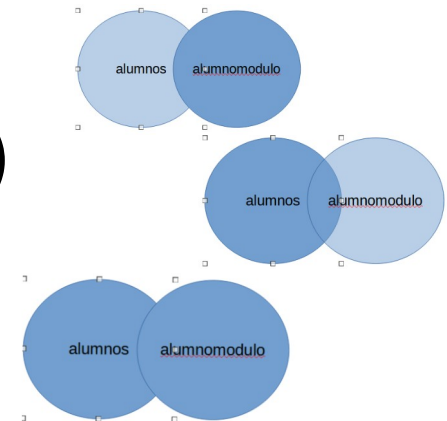
- De Equivalencia (INNER JOIN)
- Natural (NATURAL JOIN)



2/ PRODUCTO CARTESIANO (CROSS JOIN) “ERROR”

3/ JOIN EXTERNA

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



5.8.2.-Consultas multitas SQL2

- **1/ JOIN INTERNA**
 - De Equivalencia (INNER JOIN)
 - Natural (NATURAL JOIN)

INNER JOIN

– Visualizar los datos de los alumnos y cada una de sus notas:

```
select alumnos.*,notafinal from alumnos INNER JOIN alumnomodulo  
on alumnos.id_alumno=idalumno;
```

1/ JOIN INTERNA

NATURAL JOIN

Se utiliza cuando el nombre del campo por el que se realiza el filtro de ambas tablas es el mismo, entonces el filtro es automático.

Para probarlo, cambiamos el nombre de campo idalumno en alumnomodulo por id_alumno:

```
alter table alumnomodulo change idalumno id_alumno int not null;
```

```
select alumnos.*,idciclo, idmodulo, notafinal from alumnos  
natural join alumnomodulo;
```

2/ PRODUCTO CARTESIANO (CROSS JOIN) “ERROR”

`select alumnos.*,idciclo, idmodulo, notafinal from alumnos cross join alumnomodulo;`

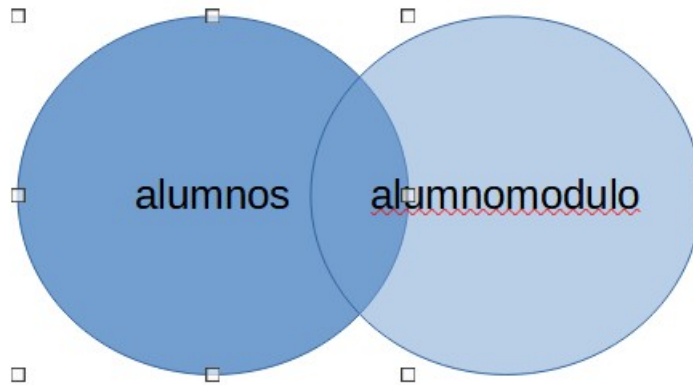
El número de registro que obtendríamos sería el número de alumnos por el número de registros de alumnomodulo.

```
mysql> select (select count(*) from alumnos)* (select count(*) from alumnomodulo);
+-----+
| (select count(*) from alumnos)* (select count(*) from alumnomodulo) |
+-----+
| 53700 |
+-----+
1 row in set (0.00 sec)
```

97	Autumn	Tyler	3	34	5.0
98	Griffin	Landry	3	34	5.0
99	Rina	Campos	3	34	5.0
100	Gage	Vaughan	3	34	5.0

```
+-----+
53700 rows in set (0.03 sec)
```

3/ JOIN EXTERNA



LEFT OUTER JOIN

– Visualizar los alumnos sin nota

```
select alumnos.* from alumnos where id_alumno not in (select idalumno from alumnomodulo);
```

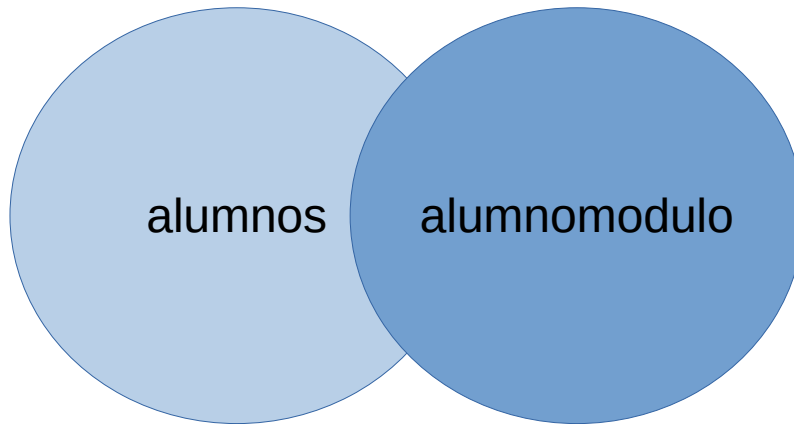
– Visualizar los datos de los alumnos, ciclo, modulo y cada una de sus notas

```
select alumnos.*,idciclo, idmodulo, notafinal from alumnos inner join alumnomodulo on alumnos.id_alumno=idalumno;
```

– Visualizar los datos de los alumnos, ciclo, modulo y cada una de sus notas, **incluso de los alumnos sin notas**.

```
select alumnos.*,idciclo, idmodulo, notafinal from alumnos left outer join alumnomodulo on alumnos.id_alumno=idalumno;
```


3/ JOIN EXTERNA

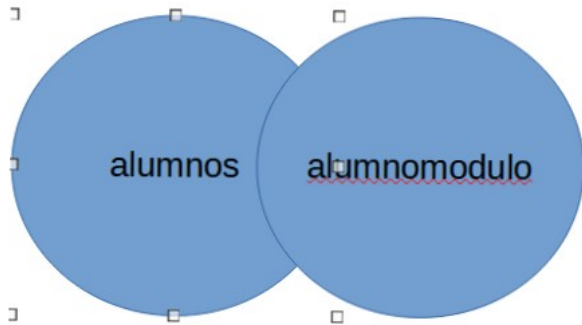


RIGHT OUTER JOIN

```
select alumnos.*,idciclo, idmodulo, notafinal from alumnos RIGHT outer join  
alumnomodulo on alumnos.id_alumno=idalumno;
```

Como resultado daría lo mismo que el INNER JOIN ya que no pueden haber notas que no pertenezcan a un alumno determinado

3/ JOIN EXTERNA



FULL OUTER JOIN

En mysql no funciona FULL OUTER JOIN, habría que unir las dos:
`select alumnos.*,idciclo, idmodulo, notafinal from alumnos right outer join
alumnomodulo on alumnos.id_alumno=alumnomodulo.id_alumno
UNION`

`select alumnos.*,idciclo,
idmodulo, notafinal from alumnos LEFT outer join
alumnomodulo on alumnos.id_alumno=alumnomodulo.id_alumno order by 6;`

Como resultado daría lo mismo que el LEFT OUTER JOIN ya que no pueden haber notas que no pertenezcan a un alumno determinado.

`select alumnos.*,idciclo, idmodulo, notafinal from alumnos left outer join
alumnomodulo on alumnos.id_alumno=alumnomodulo.id_alumno order by 6;`

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> 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	

rows in set (0.01 sec)

```
mysql> show tables;
```

Tables_in_jardineria
clientes
detallepedidos
empleados
gamasproductos
oficinas
pagos
pedidos
productos

8 rows in set (0.00 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 |
+-----+-----+-----+-----+-----+-----+
| 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 |
```

Se observa que se incluye laica que no tiene propietario.

```

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