

# TEMA 4

# REALIZACIÓN DE CONSULTAS

# CON SQL

BASE DE DATOS



*Daniel Rodríguez Fernández*

# Contenidos del Tema

1. Consultas de selección básicas
2. Filtros
3. Ordenación
4. Consultas resumen
5. Subconsultas
6. Funciones
7. Subconsultas multitable
8. Consultas reflexivas derivadas

# 1. Consultas de Selección Básicas

TEMA 4: REALIZACIÓN DE CONSULTAS  
CON SQL



Daniel Rodríguez Fernández

```
SELECT [ DISTINCT ] colum | * | expr FROM tabla ;
```

## Especificaciones:

- **column:** establece la columna o columnas que se quieren seleccionar de una tabla. Pueden seleccionarse varias columnas separándolas con comas, o todas mediante \*.

Puede crearse un alias, añadiendo después del nombre de la columna: AS alias

- **expr:** permite realizar una operación, indicando operadores, operandos y funciones.
- **DISTINCT:** fuerza que sólo se muestren los registros con valores distintos, eliminando las repeticiones.

## Ejemplos (BBDD nba):

1. Mostrar todos los campos de la tabla equipos (nombre, ciudad, conferencia, división):

```
SELECT * FROM equipos;
```

2. Mostrar los campos nombre y ciudad de la tabla equipos:

```
SELECT nombre, ciudad FROM equipos;
```

3. Mostrar las ciudades que tiene equipo en la NBA:

```
SELECT DISTINCT ciudad FROM equipos;
```

4. Mostrar nombre completo de los equipos y llamarlo franquicias:

```
SELECT concat (ciudad, ' ', nombre) as Franquicias FROM equipos;
```

# 2. Filtros

TEMA 4: REALIZACIÓN DE CONSULTAS  
CON SQL



Daniel Rodríguez Fernández

Los filtros son condiciones que interpreta el SGBD para seleccionar registros y mostrarlos como resultado de la consulta, para ello se añade la cláusula WHERE.

```
SELECT [ DISTINCT ] colum | * | expr FROM tabla  
[ WHERE filtro ];
```

### Ejemplo:

1. Mostrar los equipos de la conferencia oeste:

```
SELECT * FROM equipos  
WHERE conferencia='west';
```

2. Mostrar los nombres de los jugadores de los lakers:

```
SELECT nombre FROM jugadores  
WHERE nombre_equipo='lakers';
```

Los filtros pueden contener expresiones formadas por:

### Operandos:

- Constantes: 20, 5.7, 'España', '2020-01-02'.
- Variables: campo edad, campo nombre, etc.
- Expresiones.

### Paréntesis: ()

- Los operadores tienen una prioridad: ^, \*, /, %, +, -
- Para alterar la prioridad pueden utilizarse paréntesis.

Ejemplo:  $3+4*2 \neq (3+4)*2$

### Operadores relacionales: >, <, <>, >=, <=, =

- Sirven para comparar dos operandos, devolviendo el valor 1 si la expresión es cierta y 0 si es falsa.



## Operadores lógicos: AND, OR, NOT

**AND:** el resultado es cierto si todos los operandos son ciertos.

**OR:** el resultado es cierto si alguno de los operandos es cierto.

**NOT:** el resultado es el opuesto al del operando.

Operador	Condición 1	Condición 2	Evaluación
Y	V	V	V
	F	V	F
	V	F	F
	F	F	F
O	V	V	V
	F	V	V
	V	F	V
	F	F	F
No		V	F
		F	V

**Funciones:** permiten realizar determinados cálculos sin tener que especificar todas las operaciones: redondear, media, convertir a mayúsculas, etc.

## Ejemplos (BBDD nba):

1. Mostrar nombre y altura de los jugadores españoles de los lakers:

```
SELECT nombre, altura FROM jugadores  
WHERE nombre_equipo='lakers' AND procedencia='spain';
```

1. Mostrar nombre y altura de los jugadores españoles o eslovenos de los lakers:

```
SELECT nombre, altura FROM jugadores  
WHERE nombre_equipo='lakers'  
AND (procedencia='spain' OR procedencia='slovenia');
```

1. Mostrar nombre y altura de los jugadores españoles con altura superior a 6-5:

```
SELECT nombre, altura FROM jugadores  
WHERE procedencia='spain' AND altura>'6-5';
```

## Operador de pertenencia: IN

Permite comprobar si una columna tiene un valor igual a alguno de los incluidos dentro del paréntesis.

Nombre\_columna IN (valor1, valor2, ... )

**Ejemplo:** Mostrar jugadores españoles, eslovenos y serbios de los lakers:

### **1ª opción:**

```
SELECT * FROM jugadores WHERE nombre_equipo='lakers'  
AND (procedencia='spain' OR procedencia='slovenia'  
OR procedencia='serbia montenegro');
```

### **2ª opción:**

```
SELECT * FROM jugadores WHERE nombre_equipo='lakers'  
AND procedencia IN ('spain', 'slovenia', 'serbia montenegro');
```

## Operador de rango: BETWEEN

Permite seleccionar los registros comprendidos entre los dos valores establecidos.

Nombre\_columna BETWEEN valor1 AND valor2

**Ejemplo:** Mostrar jugadores de los lakers con un peso entre 260 y 300:

### ***1ª opción:***

```
SELECT * FROM jugadores WHERE nombre_equipo='lakers'  
AND (peso>=260 AND peso<=300);
```

### ***2ª opción:***

```
SELECT * FROM jugadores WHERE nombre_equipo='lakers'  
AND peso BETWEEN 260 AND 300;
```

## Test de valor nulo: IS NULL / IS NOT NULL

Permiten verificar si un campo tiene o no valor nulo.

Nombre\_columna {IS | IS NOT} NULL

### Ejemplo:

1. Mostrar jugadores cuya procedencia es desconocida:

```
SELECT * FROM jugadores WHERE procedencia IS NULL;
```

2. Mostrar jugadores cuya procedencia es conocida:

```
SELECT * FROM jugadores WHERE procedencia IS NOT NULL;
```

## Test de patrón: LIKE

Nombre\_columna LIKE 'patron'

Obtienen registros que cumplen un patrón. Pueden utilizarse comodines:

% cadena de cualquier tamaño

\_ cualquier carácter.

### Ejemplo:

1. Mostrar jugadores cuyo nombre es Anthony:

```
SELECT * FROM jugadores WHERE nombre LIKE 'Anthony%';
```

2. Mostrar equipos cuyo nombre empiece por R, termine por S y tenga 7 caracteres:

```
SELECT * FROM equipos WHERE nombre LIKE 'R____S';
```



## Limite de registros: LIMIT

LIMIT [desplazamiento] nfilas  
desplazamiento -> primer registro que se mostrara.  
Nfilas > numero de registros que se mostraran.

### Ejemplo:

1. Mostrar 3 equipos a partir del decimo:

```
SELECT * FROM equipos LIMIT 10,3;
```

2. Mostrar 5 jugadores cuyo nombre empiece por Ant:

```
SELECT * FROM jugadores  
WHERE nombre LIKE 'ant%'  
LIMIT 5;
```

# 3. Ordenación

TEMA 4: REALIZACIÓN DE CONSULTAS  
CON SQL



*Daniel Rodríguez Fernández*



**SELECT [DISTINCT] expresion [FROM tabla] [WHERE filtro]**

**[ORDER BY {columna | expr | posicion} [ ASC | DESC ]**

- La cláusula ORDER BY permite ordenar los registros resultado de la consulta en función del nombre de un campo, una expresión o posición numérica del campo.
- El orden puede ser ascendente (ASC) o descendente (DESC). El comportamiento por defecto es ascendente.

**Ejemplo:** Mostrar los equipos de la conferencia oeste en orden alfabético:

```
SELECT nombre, division FROM equipos WHERE conferencia='west'  
ORDER BY nombre;    (o bien -> ORDER BY 1)
```

# 4. Consultas Resumen

TEMA 4: REALIZACIÓN DE CONSULTAS  
CON SQL



*Daniel Rodríguez Fernández*

Las consultas pueden ofrecer información que resuma el contenido de las tablas, para ello se utilizan funciones:

- SUM (expresion) #Suma los valores
- AVG (expresion) #Calcula la media
- MIN (expresion) #Calcula el valor menor
- MAX (expresion) #Calcula el valor mayor
- COUNT (columna) #Cuenta el número de valores excluyendo nulos
- COUNT (\*) # Cuenta el número de valores incluyendo nulos

### Ejemplos:

1. ¿Cuánto pesa el jugador más pesado de la NBA?

```
SELECT max(peso) FROM jugadores;
```

2. ¿Cuántos jugadores tienen los Lakers?

```
SELECT count(*) FROM jugadores WHERE nombre_equipo='Lakers';
```

Se pueden realizar agrupaciones de registros que cumplan una determinada condición o tengan el mismo valor en alguna columna.

```
SELECT [DISTINCT] expresion FROM tabla [WHERE filtro]  
[GROUP BY expr [,expr] ...]  
[ORDER BY {columna | expr | posicion} [ ASC | DESC ] ...]
```

### Ejemplos:

1. ¿Cuánto pesa el jugador más pesado de cada equipo de la NBA?

```
SELECT nombre_equipo, max(peso) FROM jugadores  
GROUP BY nombre_equipo;
```

2. ¿Cuántos jugadores tienen cada equipo de la NBA?

```
SELECT nombre_equipo, count(*) AS 'Nº_jugadores' FROM jugadores  
GROUP BY nombre_equipo;
```

## Filtros de grupos: HAVING

Permiten aplicar los mismos filtros que la cláusula WHERE pero sobre resultados calculados mediante agrupaciones.

```
SELECT [DISTINCT] expresion [FROM tabla] [WHERE filtro]  
[GROUP BY expr [,expr] ...]  
[HAVING filtro_grupos]  
[ORDER BY {columna | expr | posicion} [ ASC | DESC ] ...]
```

**Ejemplo:** Equipos cuyos jugadores pesen de media más de 228 libras

```
SELECT nombre_equipo, avg(peso) FROM jugadores  
GROUP BY nombre_equipo  
HAVING avg(peso)>228  
ORDER BY avg(peso) DESC;
```

# 5. Subconsultas

TEMA 4: REALIZACIÓN DE CONSULTAS  
CON SQL



Daniel Rodríguez Fernández

Se utilizan para realizar filtrados con los datos de otra consulta. Estos filtros se pueden aplicar en:

- La clausula WHERE → filtrar registros.
- La clausula HAVING → filtrar grupos.

### Ejemplo:

```
SELECT nombre FROM jugadores WHERE nombre_equipo IN  
(SELECT nombre FROM equipos WHERE division = 'southwest');
```

→ Esta consulta es equivalente a hacer:

```
SELECT nombre FROM equipos WHERE division = 'southwest';
```

Resultado:: Hornets, Spurs, Rockets, Mavericks, Grizzlies.

→ Y tomar el resultado para la siguiente consulta:

```
SELECT nombre FROM jugadores WHERE nombre_equipo IN  
( 'Hornets', 'Spurs', 'Rockets', 'Mavericks','Grizzlies')
```

# TEST DE COMPARACIÓN

Consiste en utilizar los **operadores relacionales** (=, >, <, ...) para comparar el valor producido con un valor único generado por una subconsulta. La subconsulta debe:

- Generar un único valor.
- Ir a la derecha del operador relacional.

**Ejemplo:** Nombre del jugador mas alto de la NBA:

```
SELECT nombre FROM jugadores  
WHERE altura = (SELECT max(altura) FROM jugadores);
```



# TETS DE PERTENENCIA A CONJUNTO IN:

Consiste en utilizar el operador operador IN para filtrar los registros cuya expresión coincida con algún valor producido por la subconsulta

**Ejemplo: Divisiones en las que juega algún jugador español**

SELECT division FROM equipos WHERE nombre IN

(SELECT nombre\_equipo FROM jugadores WHERE procedencia='Spain');

```
mysql> describe equipos;
```

Field	Type	Null	Key	Default	Extra
Nombre	varchar(20)	NO	PRI		
Ciudad	varchar(20)	YES		NULL	
Conferencia	varchar(4)	YES		NULL	
Division	varchar(9)	YES		NULL	

4 rows in set (0.26 sec)

← EQUIPOS

JUGADORES →

```
mysql> describe jugadores;
```

Field	Type	Null	Key	Default	Extra
codigo	int(11)	NO	PRI		
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 (0.04 sec)

# TEST DE EXISTENCIA: EXISTS, NOT EXISTS:

Permite filtrar los resultados de una consulta si existen filas en la subconsulta asociada, es decir si la subconsulta genera al menos una fila.

SELECT expresion FROM Tabla

WHERE {EXISTS | NOT EXISTS} (subconsulta)

Ejemplo: Equipos que no tengan jugadores españoles:

1º. Equipos con  
jugadores  
españoles



```
mysql> select nombre_equipo from jugadores where procedencia='spain';  
+-----+  
| nombre_equipo |  
+-----+  
| Grizzlies     |  
| Lakers        |  
| Raptors       |  
| Raptors       |  
| Trail Blazers  |  
+-----+  
5 rows in set (0.00 sec)
```

2º. Los equipos  
restantes



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

## TEST CUANTIFICADOS: ALL y ANY

Sirven para calcular la relación entre una expresión y todos los registros de la consulta (ALL) o algunos de ellos (ANY).

### Ejemplo:

1. Jugadores que pesan mas que todos los jugadores españoles:

```
SELECT nombre, peso FROM jugadores WHERE peso>ALL  
(SELECT peso FROM jugadores WHERE procedencia='Spain');
```

2. Bases que pesan mas que algún pivot:

```
SELECT nombre, peso FROM jugadores  
WHERE posicion='G' AND peso>ANY  
(SELECT peso FROM jugadores WHERE posicion='C');
```

# SUBCONSULTAS ANIDADAS

Los resultados de una consulta se pueden utilizar para filtrar otra consulta, de esta forma podemos estructurar consultas.

**Ejemplo:** Ciudad en la que juega el jugador mas alto de la NBA:

1. Obtenemos la altura  
del jugador mas alto:

```
mysql> select max(altura) from jugadores;
+-----+
| max(altura) |
+-----+
| 7-6         |
+-----+
1 row in set (0.01 sec)
```

2. Obtenemos el nombre  
del equipo a través de  
la altura del jugador:

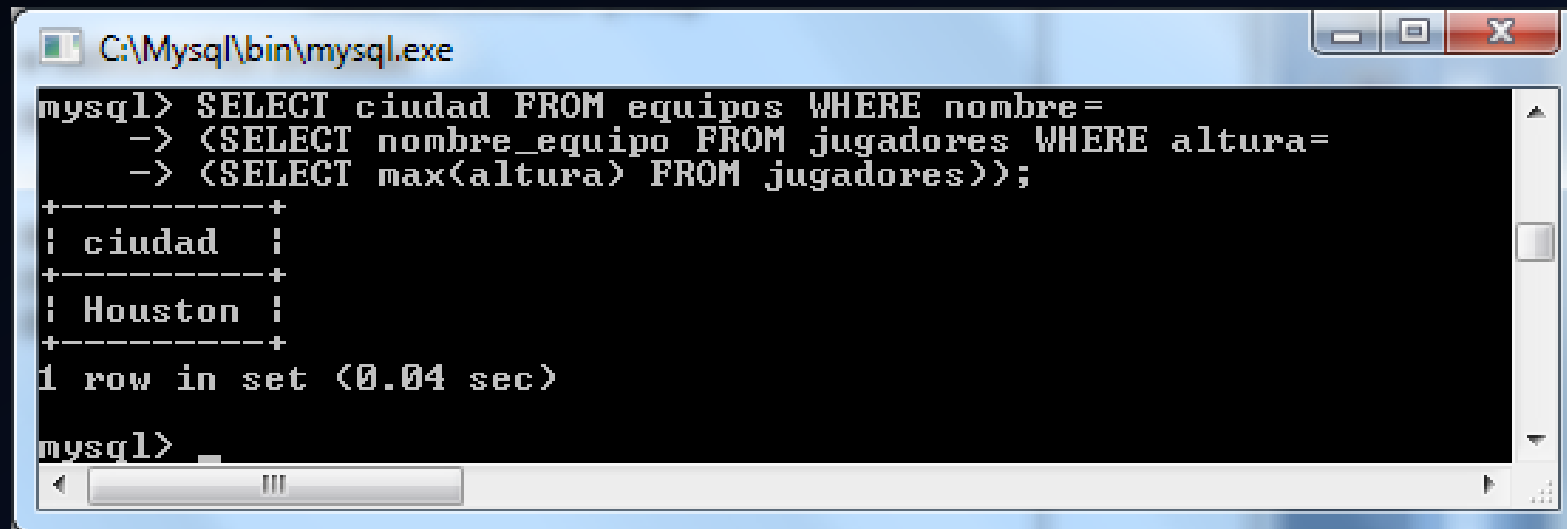
```
mysql> select nombre_equipo FROM jugadores WHERE altura='7-6';
+-----+
| nombre_equipo |
+-----+
| Rockets       |
+-----+
1 row in set (0.00 sec)
```

3. Obtenemos el nombre  
de la ciudad a partir  
del equipo

```
mysql> select ciudad from equipos where nombre='Rockets';
+-----+
| ciudad |
+-----+
| Houston |
+-----+
1 row in set (0.01 sec)
```

**Ejemplo:** Ciudad en la que juega el jugador mas alto de la NBA:

```
SELECT ciudad FROM equipos WHERE nombre=  
(SELECT nombre_equipo FROM jugadores WHERE altura=  
(SELECT max(altura) FROM jugadores));
```



A screenshot of a MySQL command prompt window. The title bar shows the path 'C:\Mysql\bin\mysql.exe'. The command prompt displays the following SQL query and its result:

```
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 (0.04 sec)

mysql>

# 6. Funciones

TEMA 4: REALIZACIÓN DE CONSULTAS  
CON SQL



*Daniel Rodríguez Fernández*

# FUNCIONES PARA GRUPOS DE VALORES

Se aplican a un conjunto de filas para obtener un único valor:

- **AVG(expr)**: calcula la media de expr ignorando nulos.
- **COUNT(\* | expr)**: cuenta el numero de valores.
- **MAX(expr)**: calcula el valor maximo de expr.
- **MIN(expr)**: calcula el valor minimo de expr.
- **SUM(expr)**: calcula la suma de expr

# FUNCIONES PARA VALORES SIMPLES

Se aplican a un único valor que puede ser un número, una variable o un campo de una tabla.

- **ABS(n)**: devuelve el valor absoluto del número n.
- **SIGN(n)**: devuelve el signo de n, positivo (1) ó negativo (-1).
- **CEIL(n)**: redondea por exceso el número n.
- **FLOOR (n)**: redondea por defecto el número n.
- **ROUND (n,[m])**: redondea n utilizando m decimales.
- **TRUNCATE (n,[m])**: corta n utilizando m decimales.
- **MOD(n,m)**: devuelve el resto de dividir n entre m.
- **POWER (n,m)**: devuelve la potencia n elevado a m.
- **SQRT (n)**: devuelve la raíz cuadrada de n.



# FUNCIONES PARA CADENAS

Permite manipular cadenas de caracteres:

- **CHAR(n)**: devuelve el carácter cuyo valor binario coincide con n según la tabla ASCII

```
mysql> select CHAR(65);
+-----+
| CHAR(65) |
+-----+
| A        |
+-----+
1 row in set (0.00 sec)
```

- **CONCAT (cad1,cad2, ...)**: devuelve una única cadena resultado de concatenar todas las cadenas.

```
mysql> select concat('Hola',' ','que',' ','tal');
+-----+
| concat('Hola',' ','que',' ','tal') |
+-----+
| Hola que tal                       |
+-----+
1 row in set (0.00 sec)
```

- **LOWER (cad)**: devuelve la cadena cad en minúsculas.
- **UPPER (cad)**: devuelve la cadena cad en mayúsculas.
- **LENGTH (cad)**: devuelve la longitud de la cadena cad.

# FUNCIONES PARA FECHAS

- Permite manipular fechas:
- **SYSDATE ()**: devuelve la fecha del sistema.
- **DATE\_ADD (fecha, intervalo p)**: devuelve fecha modificada en el intervalo especificado con p (2 month , -5 day, ...).

```
MySQL 5.7 Command Line Client
+-----+
| Fecha actual |
+-----+
| 2022-02-01 12:01:26 |
+-----+
```

- **LAST\_DAY (fecha)**: devuelve el último día del mes de la fecha.
- **DATE | DAY | MONTH | YEAR | TIME | HOUR | MINUTE (fecha)** : devuelven la parte correspondiente de la fecha: día, mes o año.

```
mysql> select DATE(sysdate());
+-----+
| DATE(sysdate()) |
+-----+
| 2013-02-25      |
+-----+
1 row in set (0.00 sec)
```

```
mysql> select TIME(sysdate());
+-----+
| TIME(sysdate()) |
+-----+
| 21:20:56        |
+-----+
1 row in set (0.00 sec)
```

# 7. Subconsultas multitable

TEMA 4: REALIZACIÓN DE CONSULTAS  
CON SQL



*Daniel Rodríguez Fernández*

# SUBCONSULTAS MULTITABLA

Una consulta multitabla es aquella en la que se utilizan datos de varias tablas, para ello se utilizan campos relacionados de las tablas (join).

- **Sintaxis:**

- `SELECT [DISTINCT] expresión`
- `FROM referencias tablas`
- `[WHERE filtro]`
- `[GROUP BY expr ]`
- `[ORDER BY {columna | expr | posicion} [ ASC| DESC]`

La diferencia con las consultas sencillas la encontramos en la cláusula FROM, y la forma de utilizarlo va a depender de la versión de SQL.

# CONSULTA MULTITABLA SQL1

Sintaxis: `SELECT * FROM tabla1, tabla2;`

Produce como resultado un conjunto de registros con todas las posibles combinaciones entre las filas de las dos tablas, a esto se le denomina producto cartesiano.

Ejemplo:

```
SELECT * FROM equipos, jugadores limit 40;
```

¿Qué resultado muestra esta consulta?

Mostrara los 30 equipos y todos los jugadores de la NBA con cada equipo.



Si a la consulta que genera producto cartesiano le añadimos un filtro que iguale los campos que comparten las tablas se obtendrán solo los registros de ambas tablas relacionados:

**Sintaxis:**    `SELECT * FROM tabla1, tabla2`  
                  `WHERE tabla1.campoN= tabla2.campoM`

**Ejemplo:**

```
SELECT * FROM equipos, jugadores
WHERE equipos.nombre=jugadores.nombre_equipo;
```

**¿Qué resultado muestra esta consulta?**

*Mostrara los 30 equipos y sus jugadores correspondientes.*

# CONSULTA MULTITABLA SQL2

SQL 2 ofrece mayor variedad de tipos para consultas multitabla:

## PRODUCTO CARTESIANO: CROSS JOIN

Muestra todas las posibles combinaciones de registros entre ambas tablas, incluyendo valores nulos.

$$T_1 \times T_2 = R$$

T1			T2			R			
A	B		C	D		A	B	C	D
1	X		iv	10		1	X	iv	10
2	Y		v	10		1	X	v	10
			v	20		1	X	v	20
						2	Y	iv	10
						2	Y	v	10
						2	Y	v	20

### Sintaxis:

```
SELECT * FROM tabla1 CROSS JOIN tabla2;  
SELECT * FROM tabla1, tabla2;
```

**Ejemplo:** Mostrar el producto cartesiano entre equipos y jugadores:

```
SELECT * FROM jugadores CROSS JOIN equipos;
```

## Composiciones internas: INNER JOIN

Selecciona los registros que satisfagan la condición de emparejamiento, suprimiendo nulos.

### Ejemplo:

Mostrar el nombre y puntos por partido de los jugadores de los wizards en la temporada 05/06.

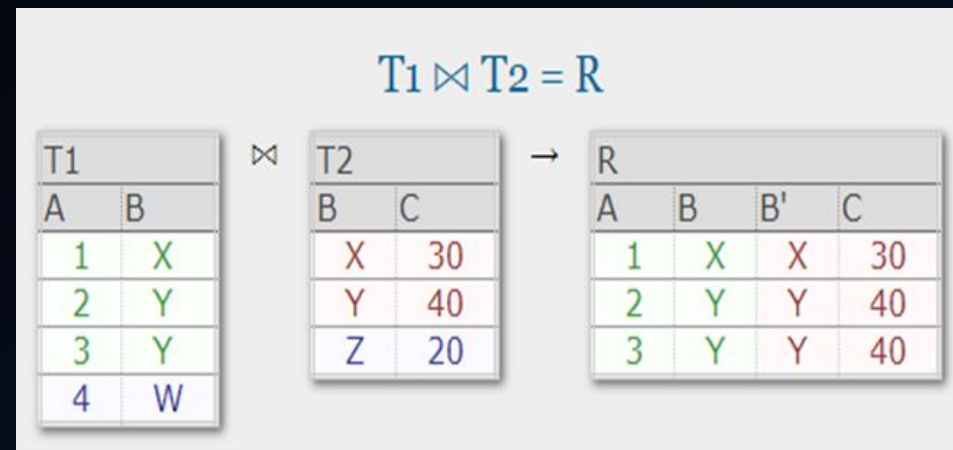
```
SELECT nombre, puntos_por_partido,temporada,nombre_equipo  
FROM jugadores INNER JOIN estadisticas  
ON jugadores.codigo=estadisticas.jugador  
WHERE nombre_equipo='wizards' AND temporada='05/06'  
ORDER BY puntos_por_partido DESC;
```



# Composiciones naturales: NATURAL JOIN

Especialización de INNER JOIN

que compara todas las columnas que tengan el mismo nombre en las dos tablas y sólo muestra una en el resultado de la consulta.



**Ejemplo:** Suponiendo que el campo nombre en la tabla equipos fuera nombre\_equipo. Mostrar los jugadores de cada equipo:

```
SELECT * FROM jugadores NATURAL JOIN equipos;
```

## Composiciones externas: OUTER JOIN

Esta operación no requiere que cada registro tenga un registro equivalente en la otra tabla. Existen 3 maneras de realizar outer join:

**LEFT JOIN** : Los registros que se admiten sin tener correspondencia son los de la tabla izquierda.

T1		□	T2		→	R			
A	B		B	C		A	B	B'	C
1	X		X	30		1	X	X	30
2	Y		Y	40		2	Y	Y	40
3	Y		Z	20		3	Y	Y	40
4	W					4	W	-	-

**RIGHT JOIN** : Los registros que se admiten sin tener correspondencia son los de la tabla derecha

T1		□	T2		→	R			
A	B		B	C		A	B	B'	C
1	X		X	30		1	X	X	30
2	Y		Y	40		2	Y	Y	40
3	Y		Z	20		3	Y	Y	40
4	W					-	-	Z	20

**FULL JOIN** : Se admiten todos los registros.

T1		□	T2		→	R			
A	B		B	C		A	B	B'	C
1	X		X	30		1	X	X	30
2	Y		Y	40		2	Y	Y	40
3	Y		Z	20		3	Y	Y	40
4	W					4	W	-	-
						-	-	Z	20

# Composiciones externas: OUTER JOIN

```
SELECT * FROM tabla1 LEFT JOIN | RIGHT JOIN tabla2  
ON tabla1.campo1=tabla2.campo2;
```

## EJEMPLO:

Imagina que en la BDD de NBA hubiera jugadores sin equipo y se deseara mostrar un listado de todos los jugadores con su equipo (incluidos los que no tienen equipo):

```
SELECT jugadores.nombre, equipos.nombre  
FROM jugadores LEFT JOIN equipos  
ON jugadores.nombre_equipo=equipos.nombre;
```

Esta consulta mostrar un listado con todos los jugadores y su equipo, para los jugadores que no tienen equipo se mostraría NULL .



# 8. Consultas reflexivas derivadas

TEMA 4: REALIZACIÓN DE CONSULTAS  
CON SQL



*Daniel Rodríguez Fernández*

# CONSULTAS REFLEXIVAS

Para realizar una consulta con datos de una relación reflexiva se utiliza JOIN en el que las dos tablas son la misma y por tanto es necesario utilizar un alias para cada uno de los roles de la entidad en la relación.

## Sintaxis:

```
SELECT lista_campos  
FROM tabla1 rol1 INNER JOIN tabla1 rol2  
ON rol1.campo1=rol2.campo2;
```

## Ejemplo:

Dada una tabla “Empleados” en la que existe una relacion reflexiva para reflejar el jefe de cada empleado, mostrar una lista de empleados con sus jefes correspondientes

```
SELECT emp.nombre as Empleado, jefe.nombre as Jefe  
FROM empleados emp INNER JOIN empleados jefe  
ON emp.codigojefe=jefe.codigoempleado;
```



# CONSULTAS CON TABLAS DERIVADAS

Las consultas con tablas derivadas utilizan sentencias SELECT en la clausula FROM en lugar de nombre de tabla.

```
SELECT lista_campos FROM  
(SELECT lista_campos FROM lista_tablas WHERE filtros)  
[AS TablaDerivada];
```

## Ejemplo:

Calcular la mayor valoracion de la temporada 2000/01. Valoracion=1 punto por cada punto y rebote y 2 puntos por cada tapon.

```
SELECT max(valoracion) FROM  
(SELECT jugador,  
(puntos_por_partido+rebotes_por_partido+2*tapones_por_partido) AS valoracion  
FROM estadisticas WHERE temporada='00/01') AS TablaValoraciones ;
```

# OPERADORES DE CONJUNTOS

Los operadores de conjuntos permiten combinar los resultados de varias sentencias SELECT

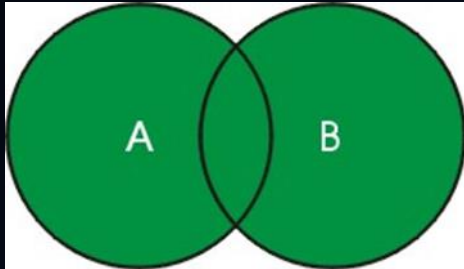
para obtener un único resultado.

`SELECT ... FROM ... WHERE ...`

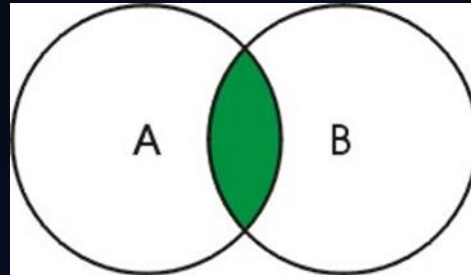
`Operador_de_conjunto`

`SELECT ... FROM ... WHERE ...`

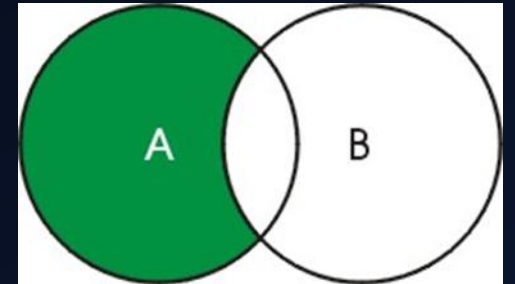
**Operador de conjunto:**



**UNIÓN**



**INTERSECCIÓN**



**DERECHA**

# EJEMPLO UNION: (OR)

Jugadores que se encuentran entre los 5 mejores reboteadores o taponadores en la temporada 2000/01.

```
(SELECT nombre FROM jugadores J, estadísticas E
WHERE J.codigo=E.jugador AND temporada='00/01'
ORDER BY rebotes_por_partido DESC LIMIT 5)
```

## UNION

```
(SELECT nombre FROM jugadores J, estadísticas E
WHERE J.codigo=E.jugador AND temporada='00/01'
ORDER BY tapones_por_partido DESC LIMIT 5);
```

nombre
Dikembre Mutombo
Ben Wallace
Tim Duncan
Antonio McDyess
Shaquille O'Neal

U

nombre
Corey Maggette
Theo Ratliff
Jermaine O'Neal
Shaquille O'Neal
Dikembre Mutombo

=

nombre
Dikembre Mutombo
Ben Wallace
Tim Duncan
Antonio McDyess
Shaquille O'Neal
Corey Maggette
Theo Ratliff
Jermaine O'Neal



## EJEMPLO INTERSECCION (AND):

Jugadores que se encuentran entre los 5 mejores reboteadores y entre los 5 mejores taponadores simultaneamente en la temporada 2000/01.

```
SELECT R.Nombre FROM
```

```
(SELECT nombre FROM jugadores J, estadísticas E  
WHERE J.codigo=E.jugador AND temporada='00/01'  
ORDER BY rebotes_por_partido DESC LIMIT 5) as R
```

```
INNER JOIN
```

```
(SELECT nombre FROM jugadores, estadísticas  
WHERE jugadores.codigo=estadísticas.jugador AND temporada='00/01'  
ORDER BY tapones_por_partido DESC LIMIT 5) as T
```

```
ON R.nombre=T.nombre ;
```

nombre
Dikembre Mutombo
Ben Wallace
Tim Duncan
Antonio McDyess
Shaquille O'Neal

∩

nombre
Corey Maggette
Theo Ratliff
Jermaine O'Neal
Shaquille O'Neal
Dikembre Mutombo

=

Nombre
Shaquille O'Neal
Dikembre Mutombo

# EJEMPLO DIFERENCIA:

Jugadores que se encuentran entre los 5 mejores reboteadores pero no entre los 5 mejores taponadores en la temporada 2000/01.

```
SELECT R.Nombre, T.nombre FROM
(SELECT nombre FROM jugadores J, estadísticas E
WHERE J.codigo=E.jugador AND temporada='00/01'
ORDER BY rebotes_por_partido DESC LIMIT 5) as R
LEFT JOIN
(SELECT nombre FROM jugadores J, estadísticas E
WHERE J.codigo=E.jugador AND temporada='00/01'
ORDER BY tapones_por_partido DESC LIMIT 5) as T
ON R.nombre=T.nombre WHERE T.nombre IS NULL ;
```

RESULTADO DE LEFT JOIN  
SIN FILTRO WHERE

Nombre	nombre
Dikembre Mutombo	Dikembre Mutombo
Ben Wallace	NULL
Tim Duncan	NULL
Antonio McDyess	NULL
Shaquille O'Neal	Shaquille O'Neal

5 rows in set (0.01 sec)

nombre
Dikembre Mutombo
Ben Wallace
Tim Duncan
Antonio McDyess
Shaquille O'Neal

-

nombre
Corey Maggette
Theo Ratliff
Jermaine O'Neal
Shaquille O'Neal
Dikembre Mutombo

=

Nombre	nombre
Ben Wallace	NULL
Tim Duncan	NULL
Antonio McDyess	NULL