



Bases de datos

U.T.4

Realización de consultas con SQL

El lenguaje DML

- ❑ Las sentencias DML del lenguaje SQL son consultas (query) o peticiones a la base de datos del tipo:

- ❑ ***SELECT***

- Extrae información de la base de datos

- ❑ ***INSERT***

- Insertar uno o varios registros en alguna tabla.

- ❑ ***DELETE***

- Borra registros de una tabla

- ❑ ***UPDATE***

- Modifica registros de una tabla.

La sentencia SELECT

- ❑ Es la sentencia más versátil de SQL y por tanto la más compleja.
- ❑ Se utiliza para obtener información de la base de datos.
- ❑ Es posible ejecutar sentencias muy sencillas que muestran todos los registros de una tabla:

#Consulta 1: Obtener todos los campos de todos los registros de la tabla empleados

```
SELECT * FROM empleados
```

La sentencia SELECT

- ❑ Y también consultas que devuelven información filtrada de múltiples tablas, utilizando incluso tablas virtuales:

#consulta 2: Obtener el total de los pedidos de los clientes de una tienda ordenados por dicha cantidad total

```
SELECT NombreCliente, Tot.Cantidad
```

```
FROM Clientes INNER JOIN Pedidos
```

```
    ON Clientes.NumeroCliente=Pedidos.NumeroCliente
```

```
INNER JOIN (SELECT sum(Cantidad*PrecioUnidad) as
```

```
    Cantidad, NumeroPedido FROM DetallePedidos
```

```
    GROUP BY NumeroPedido) Tot
```

```
ON Pedidos.numeroPedido=Tot.NumeroPedido
```

```
ORDER BY Tot.Cantidad.
```

SELECT – Sentencia básica

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

select_expr:

nombre_columna [AS alias]

| *

| expresion

❑ ***DISTINCT*** fuerza a suprimir las repeticiones

SELECT - ejemplos

SELECT * FROM vehiculos;

matricula	modelo	marca
1129FGT	ibiza gt	seat
1132GHT	leon tdi 105cv	seat
M6836YX	corolla g6	toyota
7423FZY	coupe	hyundai
3447BYD	a3 tdi 130cv	audi

SELECT - ejemplos

```
SELECT matricula,  
concat(marca,modelo) as coche  
FROM vehiculos;
```

matricula	coche
1129FGT	seatibiza gt
1132GHT	seateleoneon tdi 105cv
M6836YX	toyotacorolla g6
7423FZY	hyundaicoupe
3447BYD	audia3 tdi 130cv

SELECT - ejemplos

SELECT 1+6;

+-----+
1+6
+-----+
7
+-----+

***SELECT matricula, modelo,1+5
FROM vehiculos;***

+-----+	+-----+	+-----+
matricula	modelo	1+5
+-----+	+-----+	+-----+
1129FGT	ibiza gt	6
1132GHT	leon tdi 105cv	6
M6836YX	corolla g6	6
7423FZY	coupe	6
3447BYD	a3 tdi 130cv	6
+-----+	+-----+	+-----+

SELECT – ejemplos

SELECT *marca* ***FROM***
vehiculos;

+	-----	+
	marca	
+	-----	+
	seat	
	seat	
	toyota	
	hyundai	
	audi	
+	-----	+

SELECT ***DISTINCT*** *marca*
FROM *vehiculos;*

+	-----	+
	marca	
+	-----	+
	seat	
	toyota	
	hyundai	
	audi	
+	-----	+

► Realizar actividad 4.1

SELECT - Filtros

```
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.
- ❑ La palabra clave para realizar filtros es ***WHERE***.

Filtros – Ejemplos

```
SELECT * FROM vehiculos  
WHERE marca='seat';
```

matricula	modelo	marca
1129FGT	ibiza gt	seat
1132GHT	leon tdi 105cv	seat

Expresiones para filtros

- ❑ Los filtros se construyen mediante expresiones
- ❑ Una expresión es una combinación de ***operadores***, ***operandos*** y ***funciones***.
- ❑ Operandos:
 - ❑ Constantes: 3, 3.2, 'España', '2011-11-22'
 - ❑ Variables
 - ❑ Otras expresiones...
- ❑ Operadores aritméticos:
 - ❑ +, -, *, /, %

Expresiones para filtros

- ❑ Operadores relacionales:

- ❑ $>$, $<$, $<>$, $>=$, $<=$, $=$

- ❑ Devuelven **1** para cierto y **0** para falso

- ❑ Operadores lógicos:

- ❑ AND, OR, NOT

- ❑ Toman como operandos los valores lógicos cierto (1) y falso (0), además del valor nulo (null) (tabla pág 139)

Expresiones para filtros

- ❑ Paréntesis: ()
 - ❑ Alteran la prioridad de los operadores.
- ❑ Funciones: <https://dev.mysql.com/doc/refman/5.0/en/functions.html>
 - ❑ date_add ()
 - ❑ date_sub ()
 - ❑ now()
 - ❑ concat ()
 - ❑ ...

Filtros - Expresiones

#expresión 1 (oracle): $(2+3)*7$

SELECT $(2+3)*7$ from dual;

$(2+3)*7$

35

#expresión 2 (mysql): $(2+3)>(6*2)$

SELECT $(2+3)>(6*2)$;

+-----+

| $(2+3)>(6*2)$ |

+-----+

| 0 | #0 = falso, es falso que $5>12$

+-----+

Filtros - Expresiones

#(mysql) : la fecha de hoy -31 años;

```
SELECT date_sub(now(), interval 31 year);
```

```
+-----+
| date_sub(now(), interval 31 year) |
+-----+
| 1977-10-30 13:41:40                |
+-----+
```

► Leer expresiones página 137-138

Construcción de filtros

#selecciona los nombres de los jugadores de los Lakers:

```
SELECT Nombre FROM jugadores WHERE  
Nombre_equipo='Lakers';
```

#selecciona los nombres de los jugadores españoles de los Lakers:

```
SELECT codigo,Nombre,Altura  
FROM jugadores WHERE Nombre_equipo='Lakers'  
and Procedencia='Spain'
```

Construcción de Filtros

#Selecciona los jugadores españoles y eslovenos de los lakers

```
SELECT Nombre, Altura,Procedencia FROM jugadores
WHERE Nombre_equipo='Lakers'
AND (Procedencia='Spain' OR
Procedencia='Slovenia');
```

Nombre	Altura	Procedencia
Pau Gasol	7-0	Spain
Sasha Vujacic	6-7	Slovenia

Filtros – Otros operadores

- ❑ Además de los operadores aritméticos, lógicos, etc, una expresión de filtro puede estar formada por otros operadores de SQL:
 - ❑ Operador de pertenencia a conjuntos: ***IN***
 - ❑ Operador de rango: ***BETWEEN... AND***
 - ❑ Test de valor nulo: ***IS null*** , ***IS NOT null***
 - ❑ Test de patrón: ***LIKE***
 - ❑ Límite de número de registros: ***LIMIT***

Filtros – Operador IN

- ❑ Operador de pertenencia a conjuntos

- ❑ Sintaxis:

 - nombre_columna **IN** (valor1, valor2, ...)

 - nombre_columna **NOT IN** (valor1, valor2, ...)

- ❑ Permite comprobar si una columna tiene un valor igual que cualquiera de los valores de una lista.

Filtros – Operador IN

Operador de pertenencia a conjuntos

#Selecciona los jugadores españoles, eslovenos y serbios de los lakers

```
SELECT Nombre, Altura, Procedencia  
FROM jugadores  
WHERE Nombre_equipo='Lakers' AND  
Procedencia IN  
    ('Spain', 'Slovenia', 'Serbia &  
    Montenegro');
```

Filtros – Operador IN

Operador de pertenencia a conjuntos

#Selecciona los jugadores españoles, eslovenos y serbios de los lakers, versión larga

```
SELECT Nombre, Altura, Procedencia  
FROM jugadores  
WHERE Nombre_equipo='Lakers' AND  
(Procedencia = 'Spain'  
OR Procedencia='Slovenia'  
OR Procedencia='Serbia & Montenegro');
```

Filtros con operador de rango

- ❑ Permite seleccionar los registros que estén incluidos en un rango.
- ❑ Su sintaxis es:
nombre_columna **BETWEEN** valor1 **AND** valor2

Filtros con operador de rango

```
SELECT Nombre,Nombre_equipo,Peso FROM jugadores  
WHERE Peso BETWEEN 270 AND 300;
```

Nombre	Nombre_equipo	Peso
Chris Richard	Timberwolves	270
Paul Davis	Clippers	275
....
David Harrison	Pacers	280

► Realizar actividad 4.2

Filtros con test de valor nulo

- ❑ Los operadores IS e IS NOT permiten verificar si un campo es o no nulo, respectivamente.

- ❑ Sintaxis:

nombre_columna ***IS null***

nombre_columna ***IS NOT null***

Filtros con test de valor nulo

#seleccionar jugadores cuya procedencia es desconocida

```
SELECT nombre,Nombre_equipo
```

```
FROM jugadores WHERE Procedencia IS null;
```

+	-----	+	-----	+
	nombre		Nombre_equipo	
+	-----	+	-----	+
	Anthony Carter		Nuggets	
+	-----	+	-----	+

#la query contraria obtiene el resto de jugadores

```
SELECT nombre,Nombre_equipo
```

```
FROM jugadores WHERE Procedencia IS NOT null;
```

Filtros con test de patrón

- ❑ Los filtros con test de patrón seleccionan los registros que cumplan una serie de características.
- ❑ Se pueden usar los comodines % y _ para buscar una cadena de caracteres.

El **carácter comodín %** busca coincidencias de cualquier número de caracteres, incluso cero caracteres.

El **carácter comodín _** busca coincidencias de exactamente un carácter.

- ❑ Sintaxis: nombre_columna *LIKE* expresion

Filtros con test de patrón

```
SELECT * FROM vehiculos where modelo  
    like '%tdi%';
```

matricula	modelo	marca	
1132GHT	leon tdi 105cv	seat	
3447BYD	a3 tdi 130cv	audi	

Filtros con test de patrón

- ❑ Sacar los equipos que empiecen por R, que terminen por S y que tengan 7 caracteres.

```
SELECT * FROM equipos where Nombre like 'R_ _  
_ _ _S';
```

- ❑ Sacar los equipos que aparezca en su nombre como segunda letra la 'o'

```
SELECT * FROM equipos where Nombre like '_o  
%';
```

Filtros por límite de registros

- ❑ Limita el número de registros devuelto por una consulta.
- ❑ No es un operador estándar en todos los SGBD.
- ❑ Sintaxis (MySQL):
LIMIT [desplazamiento,] nfilas]
- ❑ Sintaxis (ORACLE):
SELECT * FROM tabla WHERE rownum <= nfilas

Filtros por límite de registros

#devuelve los 4 primeros jugadores

```
SELECT nombre,Nombre_equipo  
FROM jugadores limit 4;
```

#devuelve 3 filas a partir de la sexta

```
SELECT nombre,Nombre_equipo  
FROM jugadores LIMIT 5,3;
```

Filtros por límite de registros

(ORACLE) Saca los 25 primeros jugadores
SELECT *
FROM jugadores
WHERE ***rownum*** <= 25;

Ordenación con ORDER BY

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

- ❑ Esta cláusula especifica el criterio de clasificación de la consulta.

Ordenación con ORDER BY

- ❑ Puede contener expresiones con valores de columnas:

```
SELECT * FROM ALUMNOS ORDER BY NOTA/25
```

- ❑ Es posible anidar los criterios de ordenación. El situado más a la izquierda será el principal.
- ❑ Se puede utilizar un número, que indica la posición de la columna en el SELECT, como criterio de ordenación:

```
SELECT DEPT_NO, DNOMBRE, LOC FROM DEPART  
ORDER BY 2
```

Ordena por la segunda columna, DNOMBRE

Ordenación - Ejemplo

```
SELECT Division,Nombre
FROM equipos
WHERE
    Conferencia='West'
ORDER BY
    Division ASC,
    Nombre DESC;
```

Division	Nombre
NorthWest	Trail Blazers
NorthWest	Timberwolves
NorthWest	Supersonics
NorthWest	Nuggets
NorthWest	Jazz
Pacific	Warriors
Pacific	Suns
Pacific	Lakers
Pacific	Kings
Pacific	Clippers
SouthWest	Spurs
SouthWest	Rockets
SouthWest	Mavericks
SouthWest	Hornets
SouthWest	Grizzlies

Consultas resumen

- ❑ Generan información resumida de los registros de una tabla.
- ❑ Hay que hacer uso de las funciones de columna para generar la información calculada sobre el conjunto de registros.

Consultas resumen

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

```
+-----+  
| count(*) |  
+-----+  
|         5 |  
+-----+
```

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.

Consultas resumen - Ejemplos

#consulta 1

#¿Cuánto pesa el jugador más pesado de la nba?

SELECT max(peso) FROM jugadores;

#consulta 2

#¿Cuánto mide el jugador más bajito de la nba?

SELECT min(altura) FROM jugadores;

#consulta 3

#¿Cuántos jugadores tienen los Lakers?

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

#consulta 4

#¿Cuánto pesan de media los jugadores de los Blazers?

SELECT avg(peso) FROM jugadores WHERE Nombre_equipo='Blazers';

Consultas resumen

- ❑ Con las consultas resumen se pueden realizar agrupaciones de registros.
- ❑ El agrupamiento se lleva a cabo mediante la cláusula **GROUP BY**.
- ❑ Las columnas seleccionadas en un SELECT con agrupación deben ser: una constante, una función de grupo, una columna expresada en el GROUP BY.
- ❑ Se puede establecer una condición de búsqueda o filtro para los grupos, mediante la cláusula **HAVING**.
- ❑ **HAVING** controla cual de los grupos se visualiza => Filtra los resultados calculados mediante agrupaciones.

Agrupaciones

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


Agrupaciones

```
SELECT * FROM vehiculos;
```

matricula	modelo	marca
1129FGT	ibiza gt	seat
1132GHT	leon tdi 105cv	seat
M6836YX	corolla g6	toyota
7423FZY	coupe	hyundai
3447BYD	a3 tdi 130cv	audi

Agrupaciones

```
SELECT marca, count(*) FROM  
vehiculos GROUP BY marca;
```

```
+-----+-----+  
| marca  | count(*) |  
+-----+-----+  
| audi   |         1 |  
| hyundai |         1 |  
| seat   |         2 |  
| toyota |         1 |  
+-----+-----+
```

Agrupaciones - Ejemplos

#¿Cuánto pesa el jugador más pesado de cada equipo?

#¿Cuántos equipos tiene cada conferencia en la nba?

#¿Cuánto pesan de media los jugadores de españa, francia e italia?

Agrupaciones – error típico

```
SELECT count(*), conferencia FROM equipos;
```

```
ERROR 1140 (42000): Mixing of GROUP columns  
(MIN(),MAX(),COUNT(),...) with no GROUP  
columns is illegal  
if there is no GROUP BY clause
```

Filtros de Grupos

```
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] , ...]
```

- ❑ **HAVING** aplica los mismos filtros que la cláusula WHERE.

Filtros de grupos - Ejemplos

#Seleccionar los equipos de la nba
#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) ;
```

Filtros de grupos - ejemplos

#seleccionar qué equipos de la nba tienen

más de 1 jugador español

```
SELECT Nombre_equipo,count(*)
```

```
FROM jugadores
```

```
WHERE procedencia='Spain'
```

```
GROUP BY Nombre_equipo
```

```
HAVING count(*)>1;
```

Nombre_equipo	count(*)
Raptors	2

Evaluación de las agrupaciones

□ La evaluación de las cláusulas en tiempo de ejecución se efectúa en el siguiente orden:

1. **WHERE** Selecciona las filas
2. **GROUP BY** Agrupa estas filas
3. **HAVING** Filtra los grupos. Selecciona y elimina los grupos.
4. **ORDER BY** Clasifica la salida. Ordena los grupos.

Subconsultas

- ❑ Las subconsultas se utilizan para realizar filtrados con los datos de otra consulta.

```
SELECT nombre FROM jugadores
WHERE Nombre_equipo IN
(SELECT Nombre FROM equipos WHERE division='SouthWest');
```

+-----+	
nombre	
+-----+	
Andre Brown	
Kwame Brown	
Brian Cardinal	
Jason Collins	
...	
+-----+	

Subconsultas

- ❑ Son consultas que aparecen dentro de la cláusula WHERE o HAVING de otra sentencia SQL.
- ❑ Presentan algunas diferencias con las consultas externas:
 - ❑ Una subconsulta siempre tiene un único elemento de selección en la cláusula SELECT.
 - ❑ La cláusula ORDER BY no puede ser especificada en una subconsulta.
 - ❑ Los nombres de columna que aparecen en una subconsulta pueden referirse a columnas de tablas en la consulta principal.

Operadores para subconsultas

❑ Test de Comparación

```
SELECT nombre FROM jugadores  
WHERE altura =
```

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

```
+-----+  
| nombre |  
+-----+  
| Yao Ming |  
+-----+
```

Test de comparación

❑ Errores típicos

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

ERROR 1241 (21000): Operand should contain 1 column(s)

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

ERROR 1242 (21000): Subquery returns more than 1 row

Operadores para subconsultas

❑ Test de pertenencia a conjunto

```
SELECT division FROM equipos WHERE nombre IN
(SELECT Nombre_equipo FROM jugadores WHERE
procedencia='Spain');
```

```
+-----+
| division |
+-----+
| Atlantic |
| NorthWest |
| Pacific  |
| SouthWest |
+-----+
```

Operadores para subconsultas

❑ Test de existencia

#equipos sin españoles

```
SELECT Nombre FROM equipos WHERE EXISTS
  (SELECT Nombre FROM jugadores
   WHERE equipos.Nombre = jugadores.Nombre_Equipo
   AND procedencia='Spain');
```

+-----+
Nombre
+-----+
76ers
Bobcats
Bucks
...
+-----+

- ❑ En la práctica la subconsulta en un test EXISTS siempre se escribe utilizando la notación SELECT *.

Operadores para subconsultas

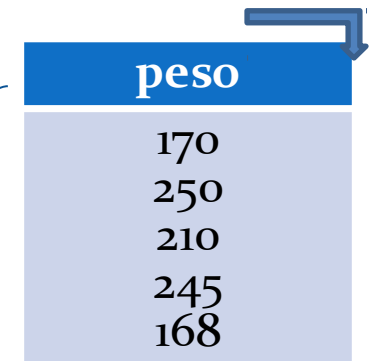
❑ *Test cuantificados ALL y ANY*

jugadores de la nba que pesan más que todos
#los jugadores españoles

```
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
...	...
+-----+-----+	

>




peso
170
250
210
245
168

Operadores para subconsultas

❑ Test cuantificados *ALL* y *ANY*

#bases que pesen más que algún pivot

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

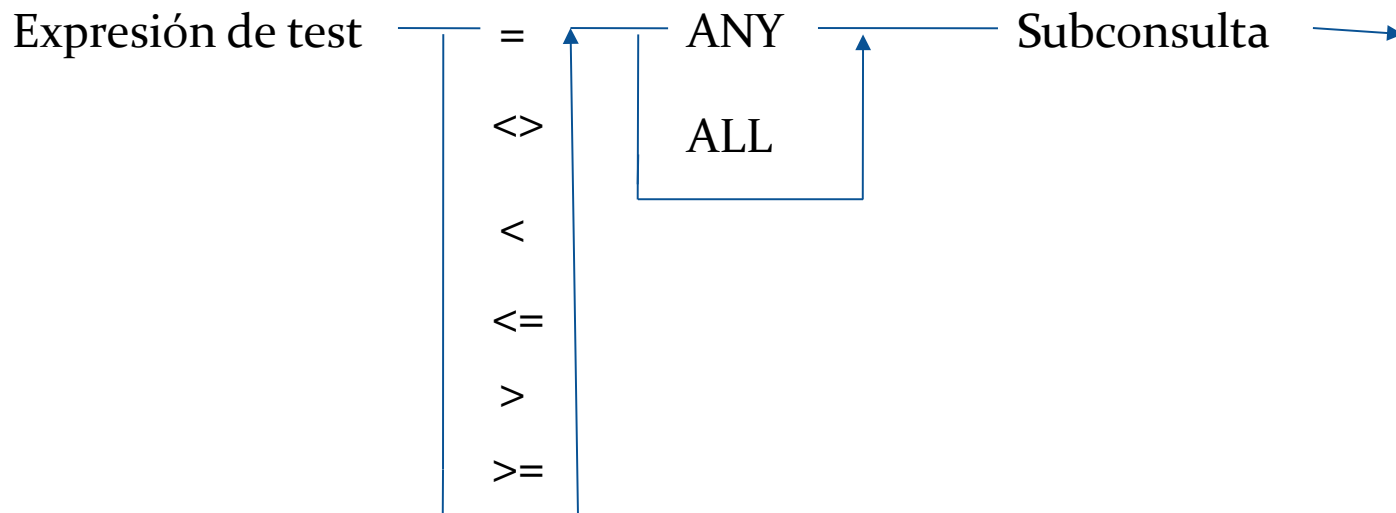


+	-----	+	-----	+
	nombre		peso	
+	-----	+	-----	+
	Joe Johnson		235	
+	-----	+	-----	+

peso
262
270
265
270
285
255
265
230
235
...
240

Test cuantificados ALL y ANY

- Se utilizan conjuntamente con uno de los seis operadores de comparación (=, <>, <, <=, >, >=) para comparar un único valor de test con una columna de valores producidos por la subconsulta.



Test cuantificado ANY

- ❑ SQL compara el valor de test con ***cada*** valor de datos en la columna resultado de la subconsulta, uno cada vez.
- ❑ Si ***alguna*** de las comparaciones individuales producen un resultado TRUE, el test ANY devuelve un resultado TRUE.

Test cuantificado ALL

- ❑ SQL compara el valor de test con **cada** valor de datos en la columna resultado de la subconsulta, uno cada vez.
- ❑ Si ***todas*** las comparaciones individuales producen un resultado TRUE, el test ALL devuelve un resultado TRUE.

Subconsultas en la cláusula Having

- Una subconsulta en la cláusula HAVING funciona como parte de la selección de grupo de filas efectuada por dicha cláusula.

```
SELECT jugador, avg(Puntos_por_partido) FROM estadisticas
WHERE temporada like '05/06' and temporada like '06/07'
GROUP BY jugador
HAVING avg(Puntos_por_partido) > (
    SELECT avg(Puntos_por_partido) from
estadisticas);
```

Subconsultas anidadas

- ❑ Se puede hacer una subconsulta para filtrar los resultados de otra subconsulta => anidar subconsultas.
- ❑ Permite generar consultas de manera sencilla.
- ❑ Permite explorar la información de la bd de forma estructurada.

Subconsultas anidadas

Obtener el nombre de la ciudad donde juega el jugador más alto de la nba.

```
SELECT ciudad FROM equipos WHERE nombre =  
  (SELECT Nombre_equipo FROM jugadores  
    WHERE altura =  
      (SELECT MAX(altura) FROM  
        jugadores));
```

```
+-----+  
| ciudad |  
+-----+  
| Houston |  
+-----+
```

Consultas multitabla

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

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]
```

Producto cartesiano

```
SELECT * FROM propietarios;
```

dni	nombre
51993482Y	José Pérez
2883477X	Matías Fernández
37276317Z	Francisco Martínez

```
SELECT * FROM animales;
```

codigo	nombre	tipo	propietario
1	Cloncho	gato	51993482Y
2	Yoda	gato	51993482Y
3	Sprocket	perro	37276317Z

Producto cartesiano

```
SELECT * FROM animales, propietarios;
```

codigo	nombre	tipo	propietario	dni	nombre
1	Cloncho	gato	51993482Y	51993482Y	José Pérez
2	Yoda	gato	51993482Y	51993482Y	José Pérez
3	Sprocket	perro	37276317Z	51993482Y	José Pérez
1	Cloncho	gato	51993482Y	2883477X	Matías Fernández
2	Yoda	gato	51993482Y	2883477X	Matías Fernández
3	Sprocket	perro	37276317Z	2883477X	Matías Fernández
1	Cloncho	gato	51993482Y	37276317Z	Francisco Martínez
2	Yoda	gato	51993482Y	37276317Z	Francisco Martínez
3	Sprocket	perro	37276317Z	37276317Z	Francisco Martínez

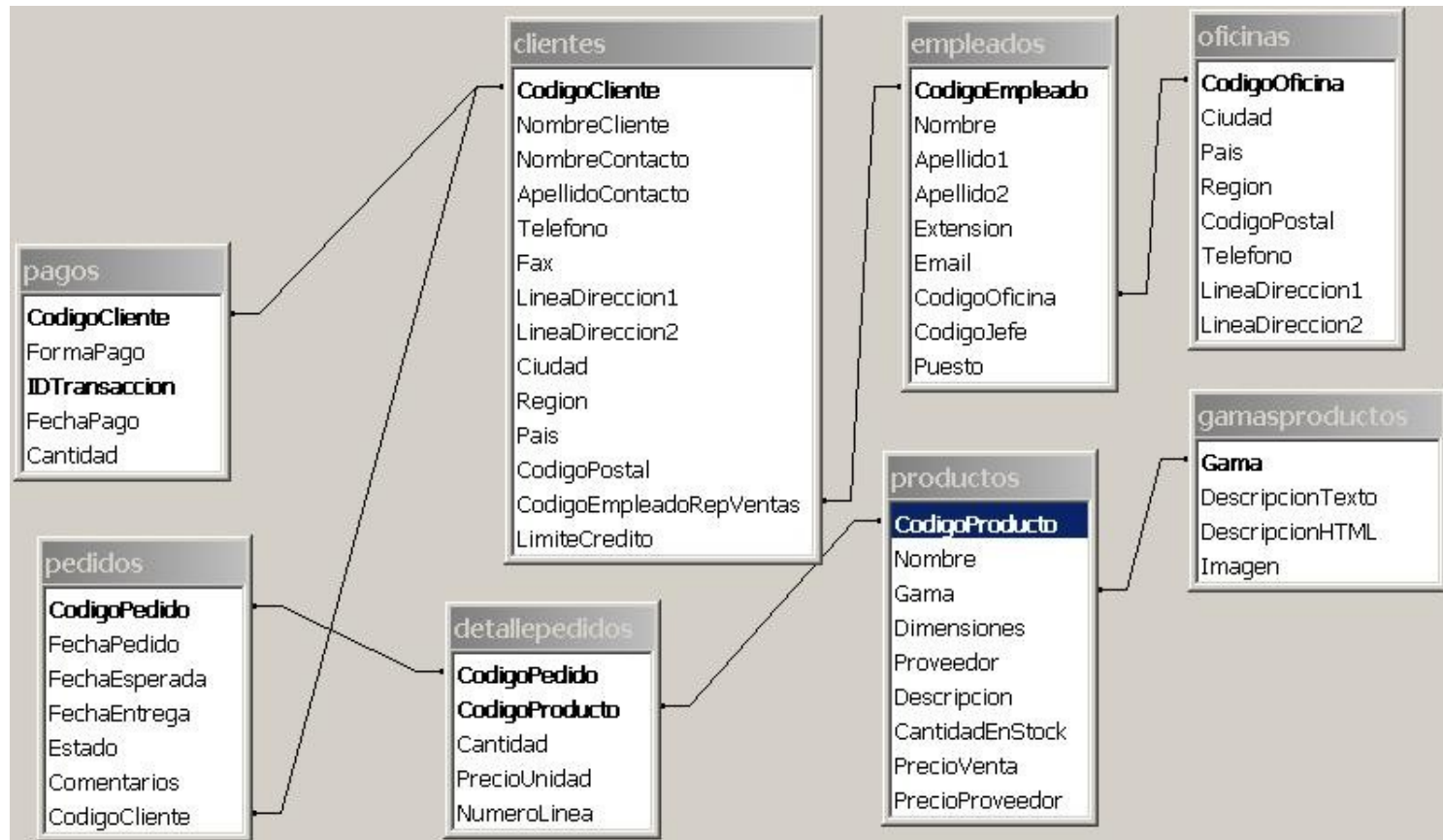
Producto cartesiano + filtro

❑ *Join*

```
SELECT * FROM animales,propietarios
        WHERE propietarios.dni=animales.propietario;
```

codigo	nombre	tipo	propietario	dni	nombre
1	Cloncho	gato	51993482Y	51993482Y	José Pérez
2	Yoda	gato	51993482Y	51993482Y	José Pérez
3	Sprocket	perro	37276317Z	37276317Z	Francisco Martínez

BBDD Jardineria



Join - Ejemplos

```
SELECT Empleados.Nombre, Clientes.NombreCliente, Pedidos.CodigoPedido
FROM Clientes, Pedidos, Empleados
WHERE Clientes.CodigoCliente=Pedidos.CodigoCliente
AND Empleados.CodigoEmpleado = Clientes.CodigoEmpleadoRepVentas
ORDER BY Empleados.Nombre;
```

Nombre	NombreCliente	CodigoPedido
Emmanuel	Naturagua	18
Emmanuel	Beragua	16
.....		
Walter Santiago	DGPRODUCTIONS GARDEN	65
Walter Santiago	Gardening & Associates	58

Join - Ejemplos

```
SELECT Empleados.Nombre, COUNT(Pedidos.CodigoPedido) as
NumeroDePedidos
FROM Clientes, Pedidos, Empleados
WHERE Clientes.CodigoCliente=Pedidos.CodigoCliente
AND Empleados.CodigoEmpleado=Clientes.CodigoEmpleadoRepVentas
GROUP BY Empleados.Nombre
ORDER BY NumeroDePedidos;
```

Nombre	NumeroDePedidos
Michael	5
Lorena	10
...	..
Walter Santiago	20

Consultas multitable SQL2

- ❑ Join Interna
 - ❑ De equivalencia (INNER)
 - ❑ Natural (NATURAL)
- ❑ Producto Cartesiano (CROSS JOIN)
- ❑ Join Externa
 - ❑ De tabla izquierda (LEFT JOIN)
 - ❑ De tabla derecha (RIGHT JOIN)
 - ❑ Completa (FULL JOIN)

Composiciones internas (INNER JOIN)

```
SELECT * FROM animales INNER JOIN propietarios  
ON animales.propietario = propietarios.dni;
```

codigo	nombre	tipo	propietario	dni	nombre
1	Cloncho	gato	51993482Y	51993482Y	José Pérez
2	Yoda	gato	51993482Y	51993482Y	José Pérez
3	Sprocket	perro	37276317Z	37276317Z	Francisco Martínez

OJO: Los animales sin propietarios no salen.

Composiciones naturales (NATURAL JOIN)

```
SELECT CodigoEmpleado, Empleados.Nombre, Oficinas.CodigoOficina,  
       Oficinas.Ciudad  
FROM Empleados NATURAL JOIN Oficinas;
```

CodigoEmpleado	Nombre	CodigoOficina	Ciudad
1	Marcos	TAL-ES	Talavera de la Reina
2	Ruben	TAL-ES	Talavera de la Reina
...			
31	Mariko	SYD-AU	Sydney

Producto Cartesiano (CROSS JOIN)

#equivalente a `SELECT * FROM animales,propietarios;`

`SELECT * FROM animales CROSS JOIN propietarios;`

codigo	nombre	tipo	propietario	dni	nombre
1	Cloncho	gato	51993482Y	51993482Y	José Pérez
1	Cloncho	gato	51993482Y	2883477X	Matías Fernández
1	Cloncho	gato	51993482Y	37276317Z	Francisco Martínez
2	Yoda	gato	51993482Y	51993482Y	José Pérez
2	Yoda	gato	51993482Y	2883477X	Matías Fernández
2	Yoda	gato	51993482Y	37276317Z	Francisco Martínez
3	Sprocket	perro	37276317Z	51993482Y	José Pérez
3	Sprocket	perro	37276317Z	2883477X	Matías Fernández
3	Sprocket	perro	37276317Z	37276317Z	Francisco Martínez
4	Arco	perro	NULL	51993482Y	José Pérez
4	Arco	perro	NULL	2883477X	Matías Fernández
4	Arco	perro	NULL	37276317Z	Francisco Martínez

Composiciones externas (OUTER JOIN)

❑ LEFT vs RIGHT

- ❑ Incluir todos los registros de la tabla que aparece a la IZQUIERDA (LEFT), aunque no tengan coincidencia con los registros de la tabla derecha
- ❑ Incluir todos los registros de la tabla que aparece a la DERECHA (RIGHT) aunque no tenga coincidencia con los registros de la tabla izquierda

❑ FULL

- ❑ Incluye todos los registros sin coincidencias, tanto los de la tabla IZQUIERDA como los de la DERECHA

Composiciones externas (LEFT OUTER JOIN)

```
#animales LEFT OUTER JOIN propietarios
```

```
#animales está a la izquierda
```

```
#propietarios está a la derecha
```

```
SELECT * FROM animales LEFT OUTER JOIN propietarios  
    ON animales.propietario = propietarios.dni;
```

codigo	nombre	tipo	propietario	dni	nombre
1	Cloncho	gato	51993482Y	51993482Y	José Pérez
2	Yoda	gato	51993482Y	51993482Y	José Pérez
3	Sprocket	perro	37276317Z	37276317Z	Francisco Martínez
4	Arco	perro	NULL	NULL	NULL

Composiciones externas (RIGHT OUTER JOIN)

#ejemplo de **RIGHT** OUTER JOIN

#animales RIGHT OUTER JOIN propietarios

#animales está a la izquierda

#propietarios está a la derecha

```
SELECT * FROM animales RIGHT OUTER JOIN propietarios
      ON animales.propietario = propietarios.dni;
```

codigo	nombre	tipo	propietario	dni	nombre
1	Cloncho	gato	51993482Y	51993482Y	José Pérez
2	Yoda	gato	51993482Y	51993482Y	José Pérez
NULL	NULL	NULL	NULL	2883477X	Matías Fernández
3	Sprocket	perro	37276317Z	37276317Z	Francisco Martínez

Composiciones Externas (FULL OUTER JOIN)

#ejemplo de FULL OUTER JOIN

#animales FULL OUTER JOIN propietarios

#animales está a la izquierda

#propietarios está a la derecha

```
SELECT * FROM animales FULL OUTER JOIN propietarios
ON animales.propietario = propietarios.dni;
```

codigo	nombre	tipo	propietario	dni	nombre
1	Cloncho	gato	51993482Y	51993482Y	José Pérez
2	Yoda	gato	51993482Y	51993482Y	José Pérez
3	Sprocket	perro	37276317Z	37276317Z	Francisco Martínez
4	Arco	perro	NULL	NULL	NULL
NULL	NULL	NULL	NULL	2883477X	Matías Fernández

UNIONES

SELECT FROM

UNION [ALL]

SELECT FROM

Consultas reflexivas

```
SELECT concat(emp.Nombre, ' ', emp.Apellido1) as  
    Empleado,  
    concat(jefe.Nombre, ' ', jefe.Apellido1) as jefe  
FROM Empleados emp INNER JOIN Empleados jefe  
    ON emp.CodigoJefe=jefe.CodigoEmpleado;
```

```
+-----+-----+  
  
| Empleado          | jefe          |  
+-----+-----+  
| Marcos Magaña     | Ruben López   |  
| Ruben López       | Alberto Soria |  
.....  
| Alberto Soria     | Kevin Fallmer |  
| Kevin Fallmer     | Julian Bellinelli |  
| Kevin Fallmer     | Mariko Kishi  |  
+-----+-----+
```

Subconsultas correlacionadas

- ❑ Subconsulta que contiene una *referencia externa*, es decir, una referencia en la cláusula FROM a alguna tabla de cualquiera de las subconsultas contenedoras de dicha subconsulta correlacionada.
- ❑ Los resultados de la subconsulta están correlacionados con cada fila individual de la consulta principal.

Subconsultas correlacionadas

Lista todas las oficinas cuyos objetivos exceden a la suma de las cuotas de los vendedores que trabajan en ellas.

```
SELECT Ciudad
FROM Oficinas
WHERE Objetivo > (SELECT SUM(Cuota)
                  FROM RepVentas
                  WHERE Oficina_rep=Oficina)
```

Consultas con tablas derivadas

- ❑ Son aquellas que utilizan sentencias SELECT en la cláusula FROM en lugar de nombres de tablas.
- ❑ Especie de tabla temporal cuyo contenido es el resultado de ejecutar una consulta.
- ❑ Ayudan a obtener información relacionada de forma mucho más avanzada.
- ❑ Las tablas derivadas no tienen limitación, es decir, se pueden unir a otras tablas, filtrar, agrupar, etc. Para facilitar su uso se suelen identificar con un alias mediante la cláusula **AS**.

Consultas con tablas derivadas

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

Tablas derivadas - ejemplo

- Se desea sacar el importe del pedido de menor coste de todos los pedidos
 - 1º: sacar el total de todos los pedidos
 - 2º: sacar el pedido con menor coste con la función de columna MIN

Tablas derivadas - ejemplo

#1° calcular el total de cada pedido

```
SELECT (SUM(Cantidad*PrecioUnidad)) as total, CodigoPedido
FROM DetallePedidos
GROUP BY CodigoPedido;
```

total	CodigoPedido
1567	1
7113	2
10850	3
...	
154	117
51	128

Tablas derivadas - ejemplo

#2: Para calcular el menor pedido, se puede hacer una tabla
derivada de la consulta anterior y con la función MIN
obtener el menor de ellos:

```
SELECT MIN(total) FROM (  
    SELECT SUM(Cantidad*PrecioUnidad) as total, CodigoPedido  
    FROM DetallePedidos  
    GROUP BY CodigoPedido) AS TotalPedidos;
```

```
+-----+  
| MIN(total) |  
+-----+  
|          4 |  
+-----+
```

#TotalPedidos es la tabla derivada formada
#por el resultado de la consulta entre paréntesis

Otra forma de resolverlo

```
SELECT (SUM(Cantidad*PrecioUnidad)) AS Total  
FROM DetallePedidos  
GROUP BY CodigoPedido  
ORDER BY Total ASC  
LIMIT 1
```