

MÓDULO II

PROGRAMACIÓN DE BASES DE DATOS RELACIONALES

Unidad Formativa 2

Definición y Manipulación de Datos

Lenguajes Relacionales

LENGUAJES RELACIONALES

MÓDULOS	UNIDADES FORMATIVAS
PROGRAMACIÓN DE BASES DE DATOS RELACIONALES	DISEÑO DE BASES DE DATOS RELACIONALES Introducción a las bases de datos Modelos conceptuales de bases de datos El modelo relacional El ciclo de vida de un proyecto Creación y diseño de bases de datos
	DEFINICIÓN Y MANIPULACIÓN DE DATOS Lenguajes relacionales El lenguaje de manipulación de la base de datos
	DESARROLLO DE PROGRAMAS EN EL ENTORNO DE LA BASE DE DATOS Lenguajes de programación de bases de datos

EL LENGUAJE DE MANIPULACIÓN DE DATOS

MÓDULOS	UNIDADES FORMATIVAS
PROGRAMACIÓN DE BASES DE DATOS RELACIONALES	DISEÑO DE BASES DE DATOS RELACIONALES Introducción a las bases de datos Modelos conceptuales de bases de datos El modelo relacional El ciclo de vida de un proyecto Creación y diseño de bases de datos
	DEFINICIÓN Y MANIPULACIÓN DE DATOS Lenguajes relacionales El lenguaje de manipulación de la base de datos
	DESARROLLO DE PROGRAMAS EN EL ENTORNO DE LA BASE DE DATOS Lenguajes de programación de bases de datos

SQL - Structured Query Language

- SQL es un lenguaje “no procedimental”
 - Se indica lo que se quiere obtener y no el “cómo”
 - El “cómo” lo decide el “optimizer” (optimizador de sentencias)

SQL (Structured Query Language)

DDL

CREATE
DROP
ALTER
RENAME
TRUNCATE

DML

SELECT
INSERT
UPDATE
DELETE

DCL

GRANT
REVOKE

TCL

START
TRANSACTION
COMMIT
ROLLBACK
SAVEPOINT

Tipos de Datos – MySQL – nros. enteros

- BIT [(M)] M indica el nro de bits. Entre 1 y 64. Por defecto 1.
(En el resto de tipos enteros, tanto M como ZEROFILL, están marcados como deprecated)
- TINYINT [UNSIGNED]. Entre (-128 a 127) ó (0 a 255)
- BOOL, BOOLEAN. Sinónimo de TINYINT (probar en consola)
- SMALLINT [UNSIGNED]. Entre (-32768 a 32767) ó (0 a 65535)
- MEDIUMINT [UNSIGNED]. Entre (-8388608 a ...)
- INT [UNSIGNED]. Entre (-2.147.483.648 a ...)
- INTEGER [UNSIGNED]. Sinónimo de INT
- BIGINT [UNSIGNED]. Entre (-9223372036854775808 a ...)

Tipos de Datos – MySQL – nros. reales

(En los tipos reales tanto `UNSIGNED` como `ZEROFILL`, están marcados como deprecated)

- `DECIMAL [(M [,D])]`.
 - `M` indica el nro total de dígitos y `D` los decimales.
- `DEC...` Sinónimo de `DECIMAL`

No estándares en SQL

- `FLOAT [(M ,D)]`.
- `FLOAT [(p)]`
- `DOUBLE [(M ,D)]`
- `DOUBLE PRECISION [(M ,D)]`

Tipos ENTEROS

Tipo	Bytes	Valor mínimo	Valor máximo cs	UNSIGNED	
				Valor mínimo	Valor máximo
TINYINT	1	-128	127	0	255
SMALLINT	2	-32768	32767	0	65535
MEDIUMINT	3	-8388608	8388607	0	16777215
INT	4	-2147483648	2147483647	0	4294967295
BIGINT	8	-2^{63}	$2^{63}-1$	0	$2^{64}-1$

Tipos de Datos – MySQL – temporales

- DATE. De '1000-01-01' a '9999-12-31'.
- DATETIME. De '1000-01-01 00:00:00' a '9999-12-31 23:59:59'
- TIMESTAMP. '1970-01-01 00:00:00' hasta el año 2037.
- TIME. de '-838:59:59' a '838:59:59'.
- YEAR [(2 | 4)]. Por defecto 4 dígitos. Para 2: 70 a 69 (1970 a 2069).

Cadenas y binarios

- CHAR [(M)]. Cadena de largo fijo. Siempre M caracteres. Por def. 1.
- VARCHAR [(M)]. Cadena de largo variable, máximo M caracteres.
- TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT
- BLOB, TINYBLOB, MEDIUMBLOB, LONGBLOB

DDL – Data Definition Language

Definición de Bases de Datos/Esquemas

CREATE DATABASE | SCHEMA [IF NOT EXISTS] <nombre_bbdd>
[**CHARACTER SET** <charset>] [**COLLATE** <collate>]

CREATE DATABASE ejemplo **CHARACTER SET** utf8mb4;

DROP DATABASE | SCHEMA [IF EXISTS] <nombre_bbdd>

ALTER DATABASE | SCHEMA <nombre_bbdd> <cambios_especificaciones>

Seleccionar una Base de Datos

USE <nombre_bbdd>

Mostrar sentencia de creación

SHOW CREATE DATABASE <nombre_bbdd>

DDL – Data Definition Language

CREATE TABLE

CREATE TABLE <nombre_tabla> (<definiciones de columnas>)

- ```
CREATE TABLE `productos` (
 `id_producto` int unsigned NOT NULL,
 `producto` varchar(100) NOT NULL,
 `precio` decimal(10,2) NOT NULL
);
```

# CREATE TABLE

**CREATE TABLE** <nombre\_tabla> (<definiciones de columnas>, **PRIMARY KEY** (<columnas>))

```
• CREATE TABLE `productos` (
 `id_producto` int unsigned NOT NULL AUTO_INCREMENT,
 `producto` varchar(100) NOT NULL,
 `precio` decimal(10,2) NOT NULL,
 PRIMARY KEY(id_producto)
);
```

```
• CREATE TABLE `productos` (
 `id_producto` int unsigned PRIMARY KEY NOT NULL AUTO_INCREMENT,
 `producto` varchar(100) NOT NULL,
 `precio` decimal(10,2) NOT NULL
);
```

# CREATE TABLE- REFERENCES

```
• CREATE TABLE `productos` (
 `id_producto` int unsigned NOT NULL AUTO_INCREMENT,
 `producto` varchar(100) NOT NULL,
 `precio` decimal(10,2) NOT NULL,
 `fk_fabricante` int unsigned DEFAULT NULL,
 PRIMARY KEY(id_producto),
 CONSTRAINT `productos_ibfk_1` FOREIGN KEY (`fk_fabricante`) REFERENCES `fabricantes` (`id_fabricante`)
);
```

**CREATE** [**TEMPORARY**] **TABLE** [**IF NOT EXISTS**] nombre\_tabla  
(**definicion**,...)

**definicion**:

col\_nom **col\_def**

| [**CONSTRAINT** [nombre\_c]] **PRIMARY KEY** (col\_nom,...)

| [**CONSTRAINT** [nombre\_c]] **FOREIGN KEY** (col\_nom,...) **reference\_def**

| **CHECK** (expr)

**col\_def** : tipo\_dato [**NOT NULL** | **NULL**] [**DEFAULT** valor] [**AUTO\_INCREMENT**]  
[**UNIQUE** [**KEY**] | [**PRIMARY KEY**] ]

**reference\_def**: **REFERENCES** nombre\_tabla (col\_nom,...)

[**ON DELETE** **opciones\_ref**]

[**ON UPDATE** **opciones\_ref**]

**opciones\_ref** : **RESTRICT** | **CASCADE** | **SET NULL** | **NO ACTION**

# Opciones Foreign Key

**ON DELETE | ON UPDATE**

**RESTRICT | CASCADE | SET NULL | NO ACTION**

**RESTRICT** Impide que se puedan eliminar las filas referenciadas o cambiar su pk

**CASCADE** Permite que se puedan actualizar o eliminar las filas referenciadas

**SET NULL** Asigna NULL a las fK que apuntan a filas eliminadas o cambiada su pk

**NO ACTION** Equivalente a RESTRICT.



# Modificación y Eliminación Tablas

**DROP [TEMPORARY] TABLE [IF EXISTS]** nombre\_tabla [, nombre\_tabla];

**ALTER TABLE** nombre\_tabla **MODIFY** Permite modificar el tipo de dato y restricciones de una columna.

**ALTER TABLE** nombre\_tabla **CHANGE** Permite renombrar una columna, modificar el tipo de dato y sus restricciones.

**ALTER TABLE** nombre\_tabla **ADD** Permite añadir una nueva columna. Se puede agregar al principio **FIRST** o después de alguna columna **AFTER** nombre\_columna

**ALTER TABLE** nombre\_tabla **DROP** Permite eliminar una columna

Mostrar listado de tablas

**SHOW TABLES**

Mostrar información de la estructura

**DESC|DESCRIBE** nombre\_tabla

Mostrar sentencia de creación

**SHOW CREATE TABLE** nombre\_tabla

# SELECT

**SELECT** <lista atributos>  
**FROM** <nombre\_tabla>  
**WHERE** <condiciones>

```
SELECT ingrediente, calorias FROM ingredientes;
```

```
SELECT ingrediente, calorias FROM ingredientes
WHERE calorias < 50;
```

**SELECT** <lista atributos>  
**FROM** <lista nombre\_tablas>  
**WHERE** <condiciones>

```
SELECT plato, ingrediente FROM platos, ingredientes
WHERE platos.id_ingrediente_principal = ingredientes.id_ingrediente;
```

```
SELECT plato, ingrediente as 'Ingrediente principal' FROM platos, ingredientes
WHERE platos.id_ingrediente_principal = ingredientes.id_ingrediente;
```

# SELECT

- Campos calculados
- Alias de campos
- Modificadores **ALL** (opcional), **DISTINCT**, **DISTINCTROW** (sinónimo)
- **[ORDER BY {nombre\_col | expr} [ASC | DESC], ...]**
- **[LIMIT {[offset,] row\_COUNT | row\_COUNT OFFSET offset}]**

# Algunas funciones

| CADENAS             |                            |
|---------------------|----------------------------|
| <b>CONCAT</b>       | Concatena                  |
| <b>CONCAT_WS</b>    | Concatena con separador    |
| <b>LOWER</b>        | Minúsculas                 |
| <b>UPPER</b>        | Mayúsculas                 |
| <b>SUBSTR</b>       | Subcadena                  |
| <b>LEFT - RIGHT</b> | Subcadenas                 |
| <b>CHAR_LENGTH</b>  | Cantidad de caracteres     |
| <b>LTRIM</b>        | Elimina espacios izq.      |
| <b>RTRIM</b>        | Elimina espacios der.      |
| <b>TRIM</b>         | Elimina espacios izq y der |
| <b>REPLACE</b>      | Reemplazar caracteres      |

| MATEMÁTICAS       |                |
|-------------------|----------------|
| <b>ABS()</b>      | Valor absoluto |
| <b>POW(x,y)</b>   | x elevado a y  |
| <b>SQRT()</b>     | Raíz cuadrada  |
| <b>PI()</b>       | $\pi$          |
| <b>ROUND()</b>    | Redondeo       |
| <b>TRUNCATE()</b> | Trunca         |
| <b>CEIL</b>       | Redondeo der.  |
| <b>FLOOR</b>      | Redondeo izq.  |
| <b>MOD</b>        | Módulo         |

| FECHA Y HORA     |                     |
|------------------|---------------------|
| <b>NOW()</b>     | Fecha y hora actual |
| <b>CURTIME()</b> | Hora actual         |
| <b>ADDDATE</b>   | Agrega días a fecha |
| <b>DATEDIFF</b>  | Dif. entre 2 fechas |
| <b>YEAR</b>      | Año de una fecha    |
| <b>MONTH</b>     | Mes...              |
| <b>MONTHNAME</b> | Nombre del mes      |
| <b>DAY</b>       | Día de ...          |
| <b>DAYNAME</b>   | Nombre del día      |
| <b>HOUR</b>      | Horas               |
| <b>MINUTE</b>    | Minutos             |
| <b>SECOND</b>    | Segundos            |

# SELECT ... WHERE...

- Condiciones para comparar valores o expresiones.
- Condiciones para comprobar si un valor está dentro de un rango de valores.
- Condiciones para comprobar si un valor está dentro de un conjunto de valores.
- Condiciones para comparar cadenas con patrones.
- Condiciones para comprobar si una columna tiene valores a NULL.

# Operadores más utilizados

| Aritméticos |          |
|-------------|----------|
| +           | Suma     |
| -           | Resta    |
| *           | Producto |
| /           | División |
| %           | Módulo   |

| Relacionales     |               |
|------------------|---------------|
| <                | Menor         |
| <=               | Menor o igual |
| >                | Mayor         |
| >=               | Mayor o igual |
| =                | Igual         |
| <> ó !=          | Distinto      |
| BETWEEN...AND... | Entre         |
| IN               | Incluido en   |
| LIKE             | Patrón (% _)  |
| IS [NOT] NULL    | Es o no nulo? |

| Lógicos  |             |
|----------|-------------|
| AND ó && | Y lógico    |
| OR ó     | O lógico    |
| NOT ó !  | Neg. lógica |

# SELECT ... WHERE...

- Condiciones para comparar valores o expresiones. **>, >=, <, ...**
- Condiciones para comprobar si un valor está dentro de un rango de valores. **BETWEEN**
- Condiciones para comprobar si un valor está dentro de un conjunto de valores. **IN ()**
- Condiciones para comparar cadenas con patrones. **LIKE**
- Condiciones para comprobar si una columna tiene valores a NULL. **IS NULL**

# SELECT resumido para una Tabla

**SELECT** [**DISTINCT**] expr [, expr ...]

[**FROM** tabla]

[**WHERE** condicion]

[**GROUP BY** {columna | expr | posición} [**ASC** | **DESC**], ... [**WITH ROLLUP**]]

[**HAVING** having\_condition]

[**ORDER BY** {col\_name | expr | position} [**ASC** | **DESC**], ...]

[**LIMIT** {[offset,] row\_COUNT | row\_COUNT OFFSET offset}]



# GROUP BY

**SELECT** expr [, expr ...]

**FROM** tabla

[**WHERE** condicion]

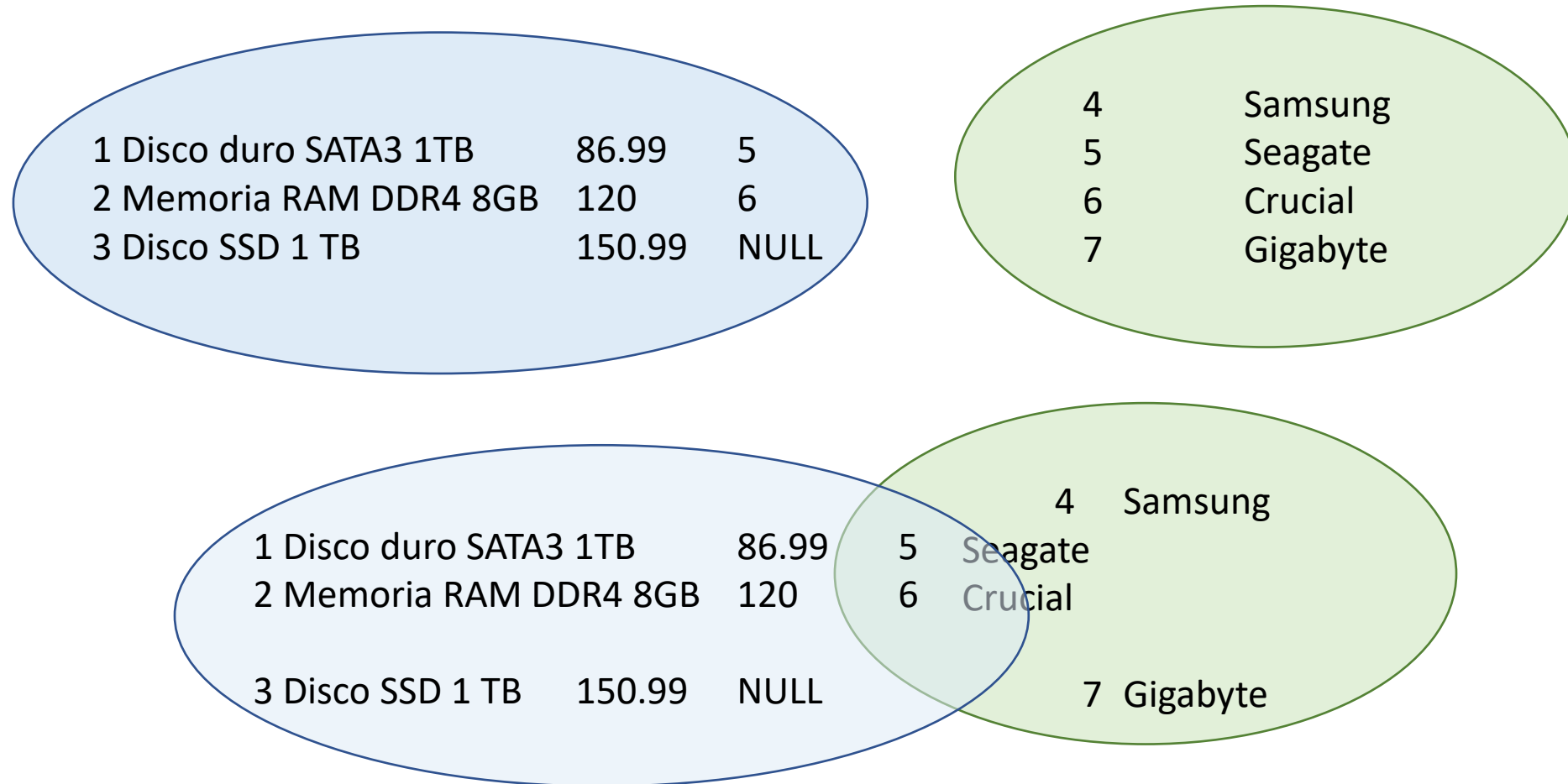
[**GROUP BY** {columna | expr | posición} [**ASC** | **DESC**], ...]

| Funciones de agregación |                             |
|-------------------------|-----------------------------|
| <b>AVG()</b>            | Promedio / Media aritmética |
| <b>SUM()</b>            | Sumatorio                   |
| <b>COUNT()</b>          | Cantidad de elementos       |
| <b>MAX()</b>            | Máximo valor                |
| <b>MIN()</b>            | Mínimo valor                |

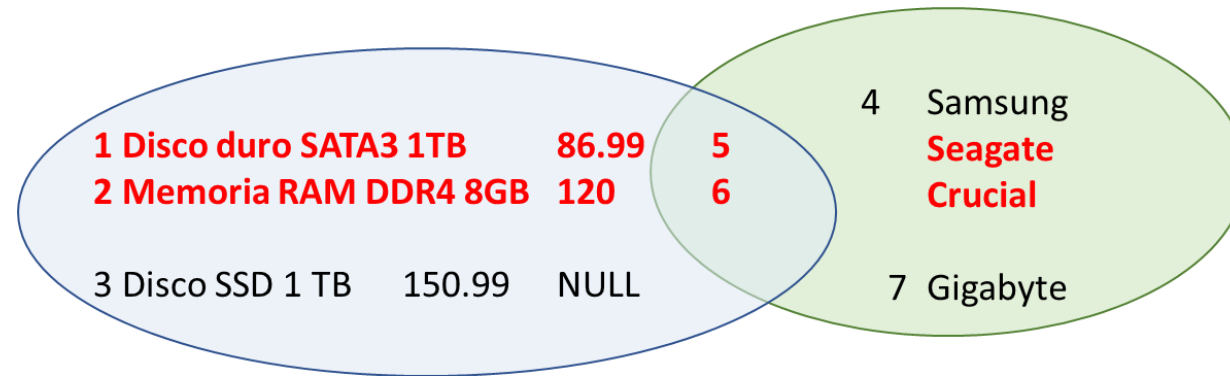
# SELECT varias tablas

- Producto cartesiano
  - ... **FROM** tabla1, tabla2; **SQL 1 (SQL-86)**
  - ... **FROM** tabla1 **JOIN** tabla2; (**INNER JOIN = JOIN**) **SQL 2 (SQL-92)**
- Filtrando resultados
  - ... **FROM** tabla1, tabla2 **WHERE** fk\_1 = pk\_2;
  - ... **FROM** tabla1 **JOIN** tabla2 **ON** fk\_1 = pk\_2;

# JOIN



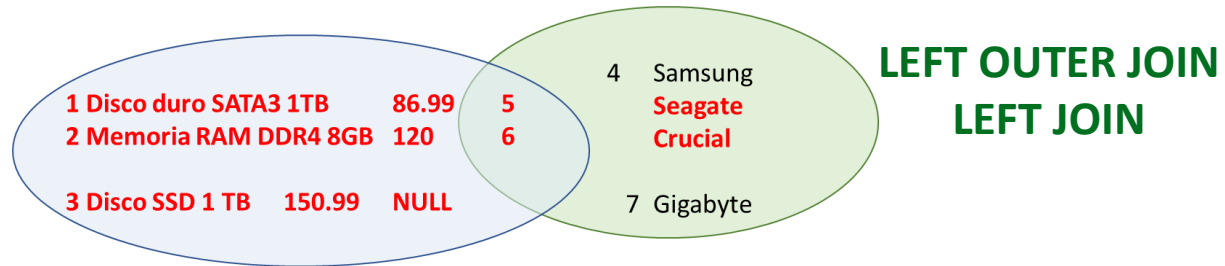
# JOIN INTERNO



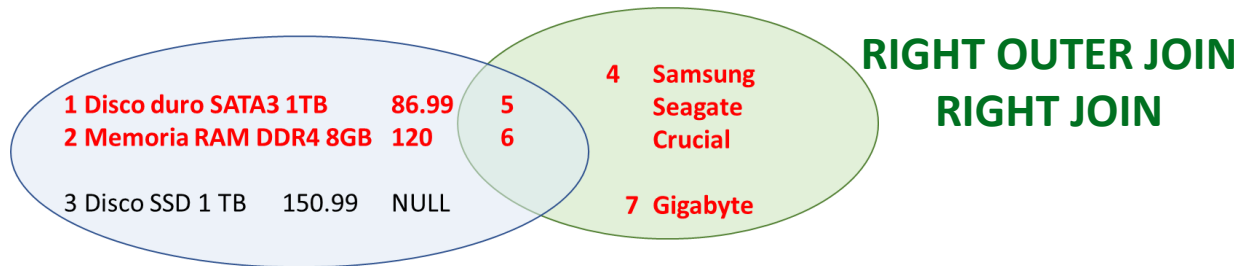
**SELECT** ... **FROM** productos **INNER JOIN** fabricantes  
**ON** productos.fk\_fabricante = fabricante.id\_fabricante;

| Id_prod | producto      | precio | Id_fabr | fabricante |
|---------|---------------|--------|---------|------------|
| 1       | Disco duro... | 86.99  | 5       | Seagate    |
| 2       | Memoria ...   | 120    | 6       | Crucial    |

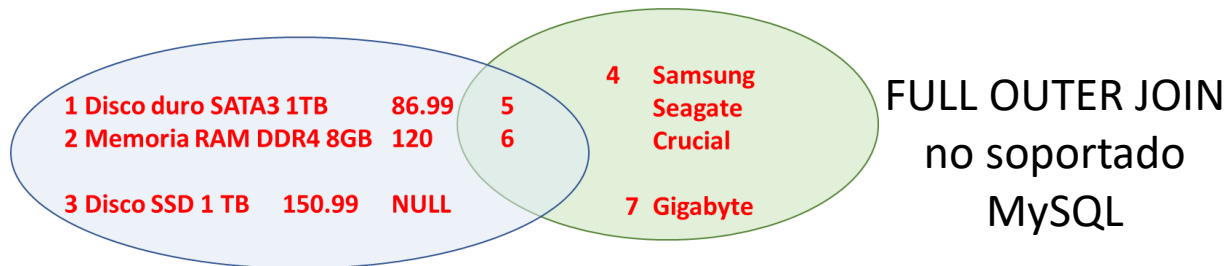
# JOIN EXTERNOS



| Id_prod | producto      | precio | Id_fabr | fabricante |
|---------|---------------|--------|---------|------------|
| 1       | Disco duro... | 86.99  | 5       | Seagate    |
| 2       | Memoria ...   | 120    | 6       | Crucial    |
| 3       | Disco SSD ... | 150.99 | NULL    | NULL       |

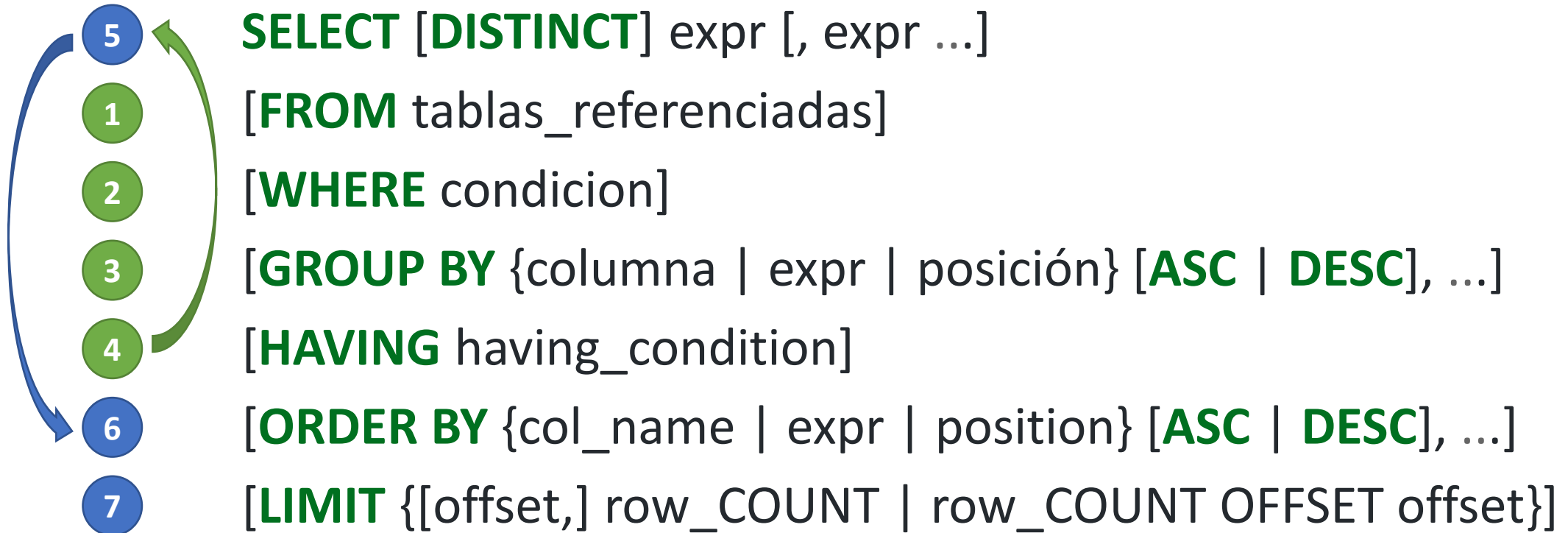


| Id_prod | producto      | precio | Id_fabr | fabricante |
|---------|---------------|--------|---------|------------|
| 1       | Disco duro... | 86.99  | 5       | Seagate    |
| 2       | Memoria ...   | 120    | 6       | Crucial    |
| NULL    | NULL          | NULL   | 4       | Samsung    |
| NULL    | NULL          | NULL   | 7       | Gigabyte   |



| Id_prod | producto      | Precio | Id_fabr | fabricante |
|---------|---------------|--------|---------|------------|
| 1       | Disco duro... | 86.99  | 5       | Seagate    |
| 2       | Memoria ...   | 120    | 6       | Crucial    |
| NULL    | NULL          | NULL   | 4       | Samsung    |
| NULL    | NULL          | NULL   | 7       | Gigabyte   |
| 3       | Disco SSD ... | 150.99 | NULL    | NULL       |

# SELECT resumido



**¿En qué orden se ejecuta el SELECT?**

# SUBCONSULTAS

**Una subconsulta es una consulta anidada dentro de otra consulta.**

- Subconsultas de columna

- Devuelve una columna con varias filas

| Col 1 |
|-------|
|       |
|       |
|       |

- Subconsultas de fila

Devuelven más de una columna pero una única fila

| Col 1 | Col 2 | Col 3 | Col 4 | Col n |
|-------|-------|-------|-------|-------|
|       |       |       |       |       |

- Subconsultas de tabla

Devuelven una o varias columnas y 0 o varias filas

| Col 1 | Col 2 | Col 3 | Col 4 | Col n |
|-------|-------|-------|-------|-------|
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |

- Subconsultas escalares

Devuelven una columna y una fila (un solo valor)

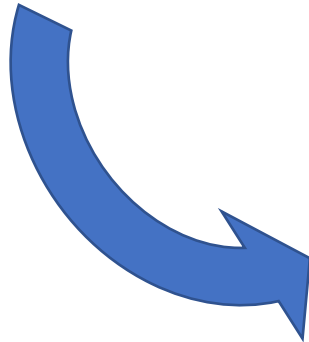
| Col 1 |
|-------|
|       |

# Subconsultas en la cláusula WHERE

Ej.: Productos cuyo precio es mayor a la media de todos los precios.

Si la media fuera: 120

```
select *
from productos
where precio > 120;
```



Obtener la media

```
select avg(precio)
from productos;
```



Combinando las dos

```
select * from productos
where precio > (select avg(precio)
from productos);
```



# Operadores en subconsultas en WHERE o HAVING

(**>**, **>=**, **<**, **<=**, **!=**, **<>**, **=**) Operadores básicos de comparación

- ... **where** coste **<** (**select** ...) Será 'V' si coste es menor al valor devuelto por la consulta.

(**ALL**, **ANY**) Se utilizan junto con los operadores de comparación.

- ... **where** coste **>= ALL** (**select** ...) Será 'V' si coste es mayor o igual a todos los valores devueltos por la consulta.
- ... **where** coste **= ANY** (**select** ...) Será 'V' si coste es igual a alguno de los valores devueltos por la consulta.

(**IN**, **NOT IN**)

- ... **where** id\_empleado **IN** (**select** ...) Será 'V' si id\_empleado se encuentra entre los valores devueltos por la consulta.

(**EXISTS**, **NOT EXISTS**)

- ... **where EXISTS** (**select** ...) Será 'V' si la consulta devuelve algo.

# Operadores y tipos de subconsultas

(**>**, **>=**, **<**, **<=**, **!=**, **<>**, **=**) La subconsulta debe ser “escalar”

(**ALL**, **ANY**) La subconsulta será de tipo “columna”

(**IN**, **NOT IN**) La subconsulta será una de tipo “tabla”  
con la cantidad de columnas iguales a la cantidad de valores buscados

(**EXISTS**, **NOT EXISTS**) La subconsulta será de tipo “tabla”

# Subconsultas en la cláusula FROM / JOIN


Ej.: Productos cuyo precio es mayor a la media de los precios de su fabricante.

Obtener media por fabricante

```
select id_fabricante, avg(precio) media
from productos join fabricantes
on fk_fabricante = id_fabricante
group by id_fabricante;
```

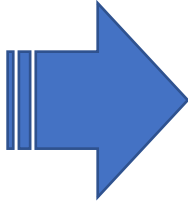
Si hacemos un join de productos con medias

```
select *
from productos join medias
on fk_fabricante = medias.id_fabricante;
```




| id_producto | producto                | precio | fk_fabricante | id_fabricante | media |
|-------------|-------------------------|--------|---------------|---------------|-------|
| 1           | Disco duro SATA3 1TB    | 86.99  | 5             | 5             | 86.99 |
| 2           | Memoria RAM DDR4 8GB    | 120    | 6             | 6             | 437.5 |
| 4           | GeForce GTX 1050Ti      | 180    | 7             | 7             | 180   |
| 5           | GeForce GTX 1080 Xtreme | 755    | 6             | 6             | 437.5 |
| ...         |                         |        |               |               |       |

Imaginemos que se llama “medias”



| Id_fabricante | media   |
|---------------|---------|
| 1             | 223.995 |
| 2             | 501.5   |
| 3             | 119.995 |
| 5             | 86.99   |
| 6             | 437.5   |
| 7             | 180     |



Falta el where precio >= media

## Obtener la tabla medias

```
select id_fabricante, avg(precio) media
from productos join fabricantes
on fk_fabricante = id_fabricante
group by id_fabricante;
```



## El join con productos y filtramos

```
select *
from productos join medias
on fk_fabricante = medias.id_fabricante
where precio >= medias.media;
```



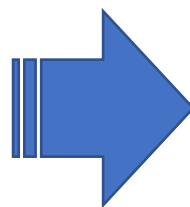
```
select * from productos
join (select id_fabricante, avg(precio) media
 from productos join fabricantes
 on fk_fabricante = id_fabricante
 group by id_fabricante) medias
on fk_fabricante = medias.id_fabricante
where precio >= medias.media;
```

# Subconsultas en la cláusula HAVING

Ej.: Fabricantes que venden el mismo nro de productos que “Asus”

Si Asus vendiera 5...

```
select fabricante, count(*) cant
from productos join fabricantes
on fk_fabricante = id_fabricante
group by fabricante
having cant = 5;
```



Cantidad de productos de Asus

```
select count(*)
from productos join fabricantes
on fk_fabricante = id_fabricante
where fabricante = 'Asus';
```

```
select fabricante, count(*) cant
from productos join fabricantes
on fk_fabricante = id_fabricante
group by fabricante
having cant = (select count(*)
from productos join fabricantes
on fk_fabricante = id_fabricante
where fabricante = 'Asus');
```

# Subconsultas en la cláusula SELECT

Ej.: Comparar el precio de cada producto con la media de los precios del resto

Media de los precios  
excluyendo al id\_producto 10

```
select avg(precio)
from productos
where id_producto <> 10;
```

Estructura de la tabla que queremos obtener

```
select p.producto, p.precio, 'media otros'
from productos p;
```

Subconsulta correlacionada

Combinando las dos

```
select p.producto, p.precio, (select avg(precio) from productos
where id_producto <> p.id_producto)
from productos p;
```

# Algunos comandos

```
> mysql -u<usuario> -p

> show databases;
> use <bdd>;
> show tables;
> describe <tabla>;
> source <ruta\fichero.sql>;
> show session variables;
> show global variables;
```

```
> set lc_time_names='es_ES'
> set global lc_time_names='es_ES'
```



# INSERT, UPDATE, DELETE

**INSERT INTO** <nombre\_tabla>  
[(<campo> (, <campo>)\* )]  
**VALUES** (<valor> (,<valor>)\*);

**INSERT INTO** <nombre\_tabla>  
[(<campo> (, <campo>)\* )]  
**SELECT ...;**

**UPDATE** <nombre\_tabla>  
**SET** <nombre\_campo> = <valor> (,<nombre\_campo> = <valor>)\*  
[**WHERE** <condición>];

**DELETE FROM** <nombre\_tabla>  
[**WHERE** <condición>];

# Transacciones – Propiedades ACID

## **ATOMICIDAD (*Atomicity*)**

Las transacciones están compuestas de una o más actividades agrupadas como una sola unidad de trabajo indivisible.

## **CONSISTENCIA (*Consistency*)**

Finalizada una transacción, de forma correcta o no, el sistema debe quedar en un estado consistente.

## **AISLAMIENTO (*Isolation*)**

Las transacciones deben soportar acceso a múltiples usuarios, de forma que las transacciones deben aislarse una de otras y evitar el acceso concurrente.

## **DURABILIDAD (*Durability*)**

Una vez finalizada una transacción, el resultado debe persistir y tolerar cualquier tipo de fallo del sistema.

# AUTOCOMMIT

- MySQL tiene activado por defecto el “modo AUTOCOMMIT”, es decir, cualquier cambio en los datos son permanentes y no se pueden deshacer.
- Se configura a través de la variable “AUTOCOMMIT”

Para conocer el valor actual:

```
SELECT @@AUTOCOMMIT;
```

Para desactivar el modo:

```
SET AUTOCOMMIT = 0;
```

Para activar el modo:

```
SET AUTOCOMMIT = 1;
```

# Crear Transacciones

**START TRANSACTION**

**COMMIT**

**ROLLBACK**

# Problemas en el acceso concurrente

- **Dirty Read (Lectura sucia)**
- **Non-Repeatable Read (Lectura No Repetible)**
- **Phantom Read (Lectura fantasma)**

# Dirty read (lectura sucia)

Sucede cuando una segunda transacción lee datos que están siendo modificados por una transacción antes de que haga COMMIT. Las lecturas pueden no ser consistentes dentro de una transacción.

| Transacción 1                                                          | Transacción 2                                                         |
|------------------------------------------------------------------------|-----------------------------------------------------------------------|
| update productos set precio = precio * 1.1<br>where fk_fabricante = 1; |                                                                       |
|                                                                        | select id_producto, precio from productos<br>where fk_fabricante = 1; |
| rollback;                                                              |                                                                       |

# Non-Repeatable Read (Lectura No Repetible)

Se produce cuando una transacción consulta el mismo dato dos veces durante la ejecución de la transacción y la segunda vez encuentra que el valor del dato ha sido modificado por otra transacción.

| Transacción 1                                                                       | Transacción 2                                                                                    |
|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| <code>select id_producto, precio from productos<br/>where fk_fabricante = 1;</code> |                                                                                                  |
|                                                                                     | <code>update productos set precio = precio * 1.1<br/>where fk_fabricante = 1;<br/>commit;</code> |
| <code>select id_producto, precio from productos<br/>where fk_fabricante = 1;</code> |                                                                                                  |

# Phantom Read (Lectura fantasma)

Ocurre cuando una transacción ejecuta dos veces una consulta que devuelve un conjunto de filas y en la segunda ejecución de la consulta aparecen nuevas filas en el conjunto que no existían cuando se inició la transacción.

| Transacción 1                                                    | Transacción 2                                                    |
|------------------------------------------------------------------|------------------------------------------------------------------|
| select cant(*) cant, avg(precio) precio_medio<br>from productos; |                                                                  |
|                                                                  | Insert into productos values (null, 'Nuevo', 800, 1);<br>commit; |
| select cant(*) cant, avg(precio) precio_medio<br>from productos; |                                                                  |



# Problemas en el acceso concurrente

- **Dirty Read (Lectura sucia)**

Sucede cuando una segunda transacción lee datos que están siendo modificados por una transacción antes de que haga COMMIT. Las lecturas pueden no ser consistentes dentro de una transacción.

- **Non-Repeatable Read (Lectura No Repetible)**

Se produce cuando una transacción consulta el mismo dato dos veces durante la ejecución de la transacción y la segunda vez encuentra que el valor del dato ha sido modificado por otra transacción.

- **Phantom Read (Lectura fantasma)**

Ocurre cuando una transacción ejecuta dos veces una consulta que devuelve un conjunto de filas y en la segunda ejecución de la consulta aparecen nuevas filas en el conjunto que no existían cuando se inició la transacción.

# Niveles de Aislamiento (Transaction Isolation)

- **Read Uncommitted**

En este nivel no se realiza ningún bloqueo, por lo tanto, permite que sucedan los tres problemas

- **Read Committed**

En este caso, si los datos leídos por una transacción son modificados (commit) por otras transacciones, se pueden dar los problemas Non-Repeatable Read y Phantom Read.

- **Repeatable Read** (Modo por defecto en MySQL)

Todas las lecturas dentro de la misma transacción leen de la captura de la base de datos tomada por la primer lectura. Según la especificación ANSI SQL, sólo puede suceder el problema del Phantom Read. (En MySQL no se produce el problema de Phantom Read con este nivel de aislamiento)

- **Serializable**

En este caso las transacciones se ejecutan unas detrás de otras, sin que exista la posibilidad de concurrencia. Todas las sentencias SELECT son convertidas implícitamente a SELECT ... LOCK IN SHARE MODE.

# Modos de aislamiento y los problemas (MySQL)

| Nivel            | Dirty Read<br>(Lectura sucia)                                                    | Non-Repeatable Read<br>(Lectura no repetible) | Phantom Read<br>(Lectura fantasma) |
|------------------|----------------------------------------------------------------------------------|-----------------------------------------------|------------------------------------|
| Read Uncommitted | Se puede dar<br><b>SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;</b> | Se puede dar                                  | Se puede dar                       |
| Read Committed   | --<br><b>SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;</b>             | Se puede dar                                  | Se puede dar                       |
| Repeatable Read  | --<br><b>SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;</b>            | --                                            | -- (en MySQL)                      |
| Serializable     | --<br><b>SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE;</b>               | --                                            | --                                 |

# Optimización de consultas (con índices)

**CREATE INDEX** nombre\_idx **ON** tabla(columna, ...)

**CREATE UNIQUE INDEX** nombre\_idx **ON** tabla(columna)

**CREATE FULL TEXT INDEX** nombre\_idx **ON** tabla(columna, ...)

**ALTER TABLE** tabla **ADD INDEX** [nombre\_idx] (columna, ...)

**SHOW INDEX FROM** tabla    o    **DESCRIBE** tabla

**DROP INDEX** nombre\_idx **ON** tabla

# Analizar consultas

## EXPLAIN | DESCRIBE SELECT ...

```
explain select asignatura, avg(nota) media
from notas join asignaturas on fk_asignatura = id_asignatura
group by fk_asignatura
having media > (select avg(nota) from notas join alumnos
 on fk_alumno = id_alumno
 where dni = '55630078R');
```

|   | id | select_type | table       | partitions | type   | possible_keys         | key     | key_len | ref                               | rows | filtered | Extra                           |
|---|----|-------------|-------------|------------|--------|-----------------------|---------|---------|-----------------------------------|------|----------|---------------------------------|
| ▶ | 1  | PRIMARY     | notas       | NULL       | ALL    | PRIMARY               | NULL    | NULL    | NULL                              | 33   | 100.00   | Using temporary; Using filesort |
|   | 1  | PRIMARY     | asignaturas | NULL       | eq_ref | PRIMARY,id_asignatura | PRIMARY | 4       | 3_universidad.notas.fk_asignatura | 1    | 100.00   | NULL                            |
|   | 2  | SUBQUERY    | alumnos     | NULL       | const  | PRIMARY,id_alumno,dni | dni     | 38      | const                             | 1    | 100.00   | Using index                     |
|   | 2  | SUBQUERY    | notas       | NULL       | ref    | PRIMARY               | PRIMARY | 4       | const                             | 1    | 100.00   | NULL                            |

# Copias de Seguridad de la BBDD

- En frío – En caliente
- Físicas/binarias – Lógicas/textuales
- Completas – Incrementales
- Manuales - Automáticas

# A consultar

- Triggers
- Procedimientos
- Funciones