CONSULTAS RESUMEN

```
SELECT [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}]
```

Es muy importante conocer en qué orden se ejecuta cada una de las cláusulas que forman la sentencia SELECT.

El orden de ejecución es el siguiente:

- Cláusula FROM.
- Cláusula WHERE (Es opcional, puede ser que no aparezca).
- Cláusula GROUP BY (Es opcional, puede ser que no aparezca).
- Cláusula HAVING (Es opcional, puede ser que no aparezca).
- Cláusula SELECT.
- Cláusula ORDER BY (Es opcional, puede ser que no aparezca).
- Cláusula LIMIT (Es opcional, puede ser que no aparezca). Se usa para limitar el número de filas devueltas de un conjunto de resultados.

LIMIT 4 (devuelve 4 registros) mientras LIMIT 4,3 (devuelve 3 registros a partir del 4ºregistro)

OFFSET nos permite especificar qué fila comenzar a partir de la recuperación de datos

*Obtiene datos comenzando en la segunda fila y limita los resultados a 2.

```
SELECT * FROM tabla LIMIT 1, 2;
```

. LIKE (Es opcional, puede ser que no aparezca).

. AND (Es opcional, puede ser que no aparezca).

. ON (INNER JOIN)

FUNCIONES DE AGREGACIÓN

Realizan una operación específica sobre todas las filas de un grupo.

Función	Descripción
`MAX(expr)`	Valor máximo del grupo
`MIN(expr)`	Valor mínimo del grupo
`AVG(expr)`	Valor medio del grupo
`SUM(expr)`	Suma de todos los valores del grupo
`COUNT(*)`	Número de filas que tiene el resultado de la consulta
`COUNT(columna)`	Número de valores no nulos que hay en esa columna

¿Diferencia entre COUNT(*) y COUNT(columna)?

- COUNT(*): Calcula el número de filas que tiene el resultado de la consulta.
- COUNT(columna): Cuenta el número de valores no nulos que hay en esa columna.

Supongamos que tenemos los siguientes valores en la tabla *alumno*:

id	nombre	apellido1	apellido2	fecha_nacimiento	es_repetidor	teléfono
1	María	Sánchez	Pérez	1990/12/01	no	NULL
2	Juan	Sáez	Vega	1998/04/02	no	618253876
3	Pepe	Ramírez	Gea	1988/01/03	no	NULL
4	Lucía	López	Ruiz	1993/06/13	sí	678516294

SELECT COUNT(teléfono) FROM alumno;	SELECT COUNT(*) FROM alumno;
COUNT(teléfono) 2	COUNT(*) 4

Contar valores distintos COUNT(DISTINCT columna)

Supongamos que tenemos los siguientes valores en la tabla *producto*:

id	nombre	precio	código_fabricante
1	Disco duro SATA3 1TB	86	5
2	Memoria RAM DDR4 8GB	120	4
3	Disco SSD 1 TB	150	5
4	GeForce GTX 1050Ti	185	5

Calcular el número de valores distintos de código de fabricante que aparecen en la tabla producto:

```
SELECT COUNT(DISTINCT código_fabricante) FROM producto;
```

COUNT(DISTINCT código_fabricante)
2

OPERACIONES:

LÓGICAS: AND OR NOT

RELACIONALES: < > = !

ARITMÉTICAS: + - * / %

DE RANGO: BETWEEN (entre)

VALOR NULO: NULL (IS IS NOT)

DE PATRÓN: LIKE ('%' '_')

GROUP BY / HAVING

La cláusula GROUP BY nos permite crear grupos de filas que tienen los mismos valores en las columnas por las que se desea agrupar.

La cláusula HAVING nos permite crear filtros sobre los grupos de filas que tienen los mismos valores en las columnas por las que se desea agrupar.

Ejemplo de agrupamiento de filas (GROUP BY) con condición de agrupamiento (HAVING)

```
SELECT fabricante.nombre, AVG(producto.precio)
FROM producto INNER JOIN fabricante
ON producto.codigo_fabricante = fabricante.codigo
WHERE fabricante.nombre != 'Seagate'
GROUP BY fabricante.codigo
HAVING AVG(producto.precio) >= 150
```

Tabla: producto

codigo	nombre	precio	codigo_fabricante
1	Portátil A		1
2	Monitor 24		2
3	Disco SSD 1 TB		3
4	Impresora		4
5	Monitor 27		2
6	Portátil B		1

Tabla: fabricante

codigo	nombre
1	Lenovo
2	Asus
3	Seagate
4	HP

Resultado del INNER JOIN:

producto.codigo	producto.nombre	producto.precio	producto.codigo_fabricante	fabricante.codigo	fabricante.nombre
1	Portátil A	599	1	1	Lenovo
2	Monitor 24	203	2	2	Asus
3	Disco SSD 1 TB	115	3	3	Seagate
4	Impresora	49	4	4	HP
5	Monitor 27	242	2	2	Asus
6	Portátil B	320	1	1	Lenovo

```
SELECT fabricante.nombre, AVG(producto.precio)
FROM producto INNER JOIN fabricante
ON producto.codigo_fabricante = fabricante.codigo
WHERE fabricante.nombre != 'Seagate'
GROUP BY fabricante.codigo
HAVING AVG(producto.precio) >= 150
```

producto.codigo	producto.nombre	producto.precio	producto.codigo_fabricante	fabricante.codigo	fabricante.nombre	
1	Portátil A	599	1	1	Lenovo	✓
2	Monitor 24	203	2	2	Asus	✓
~~3~~	~~Disco SSD 1 TB~~	~~115~~	~~3~~	~~3~~	~~Seagate~~	X
4	Impresora	49	4	4	HP	✓
5	Monitor 27	242	2	2	Asus	✓
6	Portátil B	320	1	1	Lenovo	✓

Resultado de aplicar el filtro WHERE:

producto.codigo	producto.nombre	producto.precio	producto.codigo_fabricante	fabricante.codigo	fabricante.nombre
1	Portátil A	599	1	1	Lenovo
2	Monitor 24	203	2	2	Asus
4	Impresora	49	4	4	HP
5	Monitor 27	242	2	2	Asus
6	Portátil B	320	1	1	Lenovo

```
SELECT fabricante.nombre, AVG(producto.precio)
FROM producto INNER JOIN fabricante
ON producto.codigo_fabricante = fabricante.codigo
WHERE fabricante.nombre != 'Seagate'
GROUP BY fabricante.codigo
HAVING AVG(producto.precio) >= 150
```

producto.codigo	producto.nombre	producto.precio	producto.codigo_fabricante	fabricante.codigo	fabricante.nombre
1	Portátil A	599	**1**	1	Lenovo
6	Portátil B	320	**1**	1	Lenovo
2	Monitor 24	203	**2**	2	Asus
5	Monitor 27	242	**2**	2	Asus
4	Impresora	49	**4**	4	HP

```
SELECT fabricante.nombre, AVG(producto.precio)
FROM producto INNER JOIN fabricante
ON producto.codigo_fabricante = fabricante.codigo
WHERE fabricante.nombre != 'Seagate'
GROUP BY fabricante.codigo
HAVING AVG(producto.precio) >= 150
```

producto.codigo	producto.nombre	producto.precio	producto.codigo_fabricante	fabricante.codigo	fabricante.nombre
1	Portátil A	599	**1**	1	Lenovo
6	Portátil B	320	**1**	1	Lenovo

459.5

AVG(producto.precio) >= 150 ✓

producto.codigo	producto.nombre	producto.precio	producto.codigo_fabricante	fabricante.codigo	fabricante.nombre
2	Monitor 24	203	**2**	2	Asus
5	Monitor 27	242	**2**	2	Asus

222.5

AVG(producto.precio) >= 150 ✓

producto.codigo	producto.nombre	producto.precio	producto.codigo_fabricante	fabricante.codigo	fabricante.nombre
4	Impresora	49	**4**	4	HP

49

AVG(producto.precio) >= 150 X

Resultado después de aplicar el filtro HAVING:

producto.codigo	producto.nombre	producto.precio	producto.codigo_fabricante	fabricante.codigo	fabricante.nombre
1	Portátil A	599	**1**	1	Lenovo
6	Portátil B	320	**1**	1	Lenovo
2	Monitor 24	203	**2**	2	Asus
5	Monitor 27	242	**2**	2	Asus

```
SELECT fabricante.nombre, AVG(producto.precio)
FROM producto INNER JOIN fabricante
ON producto.codigo_fabricante = fabricante.codigo
WHERE fabricante.nombre != 'Seagate'
GROUP BY fabricante.codigo
HAVING AVG(producto.precio) >= 150
```

fabricante.nombre	AVG(producto.precio)
Lenovo	459.5
Asus	222.5

CONSULTAS SOBRE VARIAS TABLAS

INNER JOIN

```
/* SQL 1 */
SELECT *
FROM empleado, departamento
WHERE empleado.id_departamento = departamento.id
```

```
/* SQL 2 */
SELECT *
FROM empleado INNER JOIN departamento
ON empleado.id_departamento = departamento.id
```

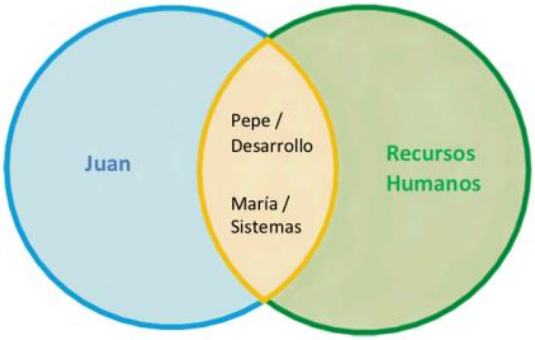
Tabla: empleado

id	nombre	id_departamento
1	Pepe	1
2	María	2
3	Juan	NULL

Tabla: departamento

id	nombre
1	Desarrollo
2	Sistemas
3	Recursos Humanos

Estas filas quedan **fuera de la intersección**



El resultado de INNER JOIN es:

empleado.id	empleado.nombre	empleado.id_departamento	departamento.id	departamento.nombre
1	Pepe	1	1	Desarrollo
2	María	2	2	Sistemas

NATURAL JOIN: no es necesario utilizar la cláusula ON para indicar sobre qué columna vamos a relacionar las dos tablas. Se van a **relacionar sobre aquellas columnas que tengan el mismo nombre.** Sólo cuando estemos seguros que los nombres de las columnas sobre las que quiero relacionar las dos tablas se llaman igual en las dos tablas. Lo normal es que no suelan tener el mismo nombre y que debamos usar una composición de tipo INNER JOIN.

Inner J > SELECT * FROM 1ºtabla INNER JOIN 2º tabla ON 1.CodComun = 2.Cod.Comun;
Natural J > SELECT * FROM 1ºtabla NATURAL JOIN 2ºtabla;

LEFT JOIN

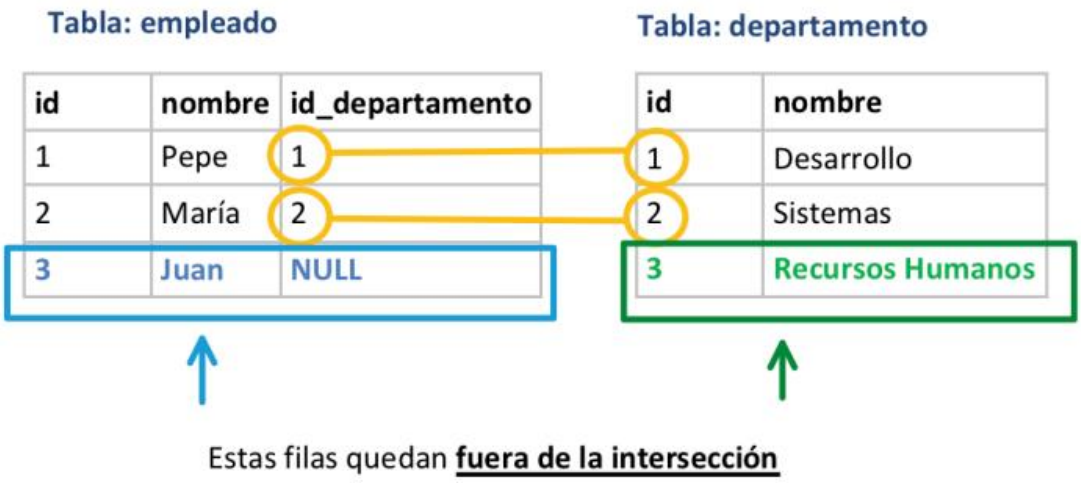
Devuelve todos los registros relacionados en ambas tablas, seleccionando SOLO los que cumplan las restricciones.

Los valores Nulos no se combinan.

SELECT * FROM 1ªtabla (letra) INNER JOIN 2ªtabla (letra)
ON (letra1ªt).CodComún = (letra2ªt).CodComún
WHERE (letra).Lo que pida LIKE '%xxx_'

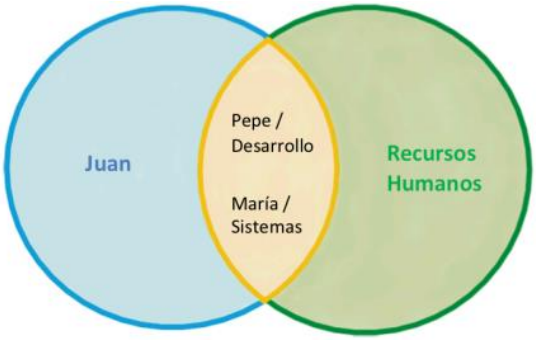
Todas las filas de la 1ª columna + las asociadas de la 2ªcolumna (la intersección entre ambas

```
/* SQL 2 */
SELECT *    1ª                    2ª
FROM empleado LEFT JOIN departamento
ON empleado.id_departamento = departamento.id
```



El resultado de LEFT JOIN:

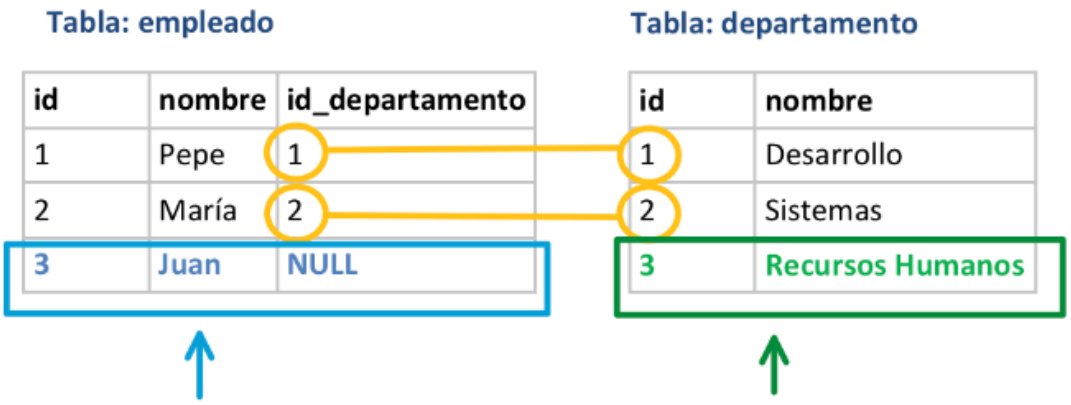
empleado.id	empleado.nombre	empleado.id_departamento	departamento.id	departamento.nombre
1	Pepe	1	1	Desarrollo
2	María	2	2	Sistemas
3	Juan	NULL	NULL	NULL



RIGHT JOIN

Todas las Filas de la 2ªcolumna + Asociadas de la 1ªColumna (la intersección entre ambas)

```
/* SQL 2 */
SELECT *
FROM empleado RIGHT JOIN departamento
ON empleado.id_departamento = departamento.id
```



Estas filas quedan **fuera de la intersección**

SELECT * FROM 2ªtabla (letra) INNER JOIN 1ªtabla (letra)
ON (letra2ªt).CodComún = (letra1ªt).CodComún
WHERE (letra).Lo que pida LIKE '%xxx_'

El resultado de RIGHT JOIN:

empleado.id	empleado.nombre	empleado.id_departamento	departamento.id	departamento.nombre
1	Pepe	1	1	Desarrollo
2	María	2	2	Sistemas
NULL	NULL	NULL	3	Recursos Humanos

SUBCONSULTAS: ANY (si algún regísto coincide) ALL (todos coinciden)
IN (si está en) NOT IN (si no está)

El resultado de un **FULL OUTER JOIN** (que en mysql no existe) es obtener la intersección de las dos tablas, junto las filas de ambas tablas que no se puedan combinar.

CROSS JOIN Muestra todas las combinaciones posibles entre filas

SELECT * FROM Empleados CROSS JOIN Sucursales; (muestra todos los empleados y relaciónalos con las sucursales)

OUTER JOIN Útil para averiguar los campos NULL.

```sql
SELECT *
FROM empleado LEFT JOIN departamento
ON empleado.codigo_departamento = departamento.codigo

UNION

SELECT
FROM empleado RIGHT JOIN departamento
ON empleado.codigo_departamento = departamento.codigo
```

UNION: Permite añadir el resultado de una SELECT a otra SELECT.

También se pueden **unir varias tablas**:

```sql
SELECT *
FROM cliente INNER JOIN empleado
ON cliente.codigo_empleado_rep_ventas = empleado.codigo_empleado
INNER JOIN pago
ON cliente.codigo_cliente = pago.codigo_cliente;
```

Unir una tabla consigo misma:

```sql
SELECT empleado.nombre, empleado.apellido1, empleado.apellido2, jefe.nombre, jefe.
    apellido1, jefe.apellido2
FROM empleado INNER JOIN empleado AS jefe
ON empleado.codigo_jefe = jefe.codigo_empleado
```

INTERSECT: Permite unir dos consultas cuyo resultado sean filas que estén en las dos tablas
*Muestra modelos presentes en ambos almacenes AL MISMO TIEMPO:

MINUS: Une dos consultas cuyo resultado son las filas que están el la 1ª tabla y pero NO en la segunda
NO SE USAN EN MySQL

CASE: (permite una estructura parecida a Switch)

```
SELECT
     CASE
        WHEN xxxx   ><=! (numero) THEN 'xxx'
        WHEN xxxx   ><=! (numero) THEN 'ddd'
        ELSE 'ccc'
     END categoría_xxxx
FROM  tabla;
```