BBDD

Bases de Datos – Consultas multitabla - Composiciones

IES Ciudad Escolar
Desarrollo de Aplicaciones Multiplataforma

Objetivos

- Reconocer y diferenciar las distintas composiciones de que disponemos durante las consultas multitablas en MySQL.
- Utilizar correctamente dichas composiciones.

Lenguaje SQL

El lenguaje de programación **SQL es el lenguaje fundamental de los SGBD** relacionales y los elementos que lo componen son:

a) **DML (*Data Manipulation Language*)**: es el lenguaje que consulta o manipula los datos ya existentes de nuestra BD.

b) **DDL (*Data Definition Language*)**: permite la **definición, modificación y eliminación de las estructuras básicas** (BD, tablas, etc.) en un SGBD.

c) **DCL (*Data Control Language*)**: administra a los usuarios de la BD, concediendo o denegando los permisos oportunos.

d) **TCL (*Transaction Control Language*)**: lenguaje que controla el procesamiento de las transacciones de la BD.

DML (Data Manipulation Language)

DML (Data Manipulation Language) o Lenguaje de Manipulación de Datos es la parte de SQL dedicada a la manipulación de los datos.

Las sentencias DML son las siguientes:

- **SELECT**: se utiliza para realizar consultas y extraer información de la base de datos.
- **INSERT**: se utiliza para insertar registros en las tablas de la base de datos.
- **UPDATE**: se utiliza para actualizar los registros de una tabla.
- **DELETE**: se utiliza para eliminar registros de una tabla.

En esta Unidad seguimos centrándonos en el uso de la sentencia **SELECT**.

https://dev.mysql.com/doc/refman/8.0/en/select.html

Composiciones

- Las **consultas multitabla** que hemos visto permiten consultar información en más de una tabla.
- La única diferencia respecto a las consultas sencillas es que especificamos en la cláusula FROM cuáles son las tablas que vamos a usar y cómo las vamos a relacionar entre sí.
- Para realizar este tipo de consultas podemos usar dos alternativas:

1. **La sintaxis de SQL 1** (SQL-86), que consiste en realizar el producto cartesiano de las tablas y añadir un filtro para relacionar los datos que tienen en común. (la que hemos venido usando)

    1. Composición cruzada (producto cartesiano)
    2. Composición interna (intersección): Utilizamos la cláusula WHERE para indicar la(s) columna(s) que queremos relacionar entre tablas.

2. **La sintaxis de SQL 2** (SQL-92 y SQL-2003) que incluye todas las cláusulas de tipo JOIN.

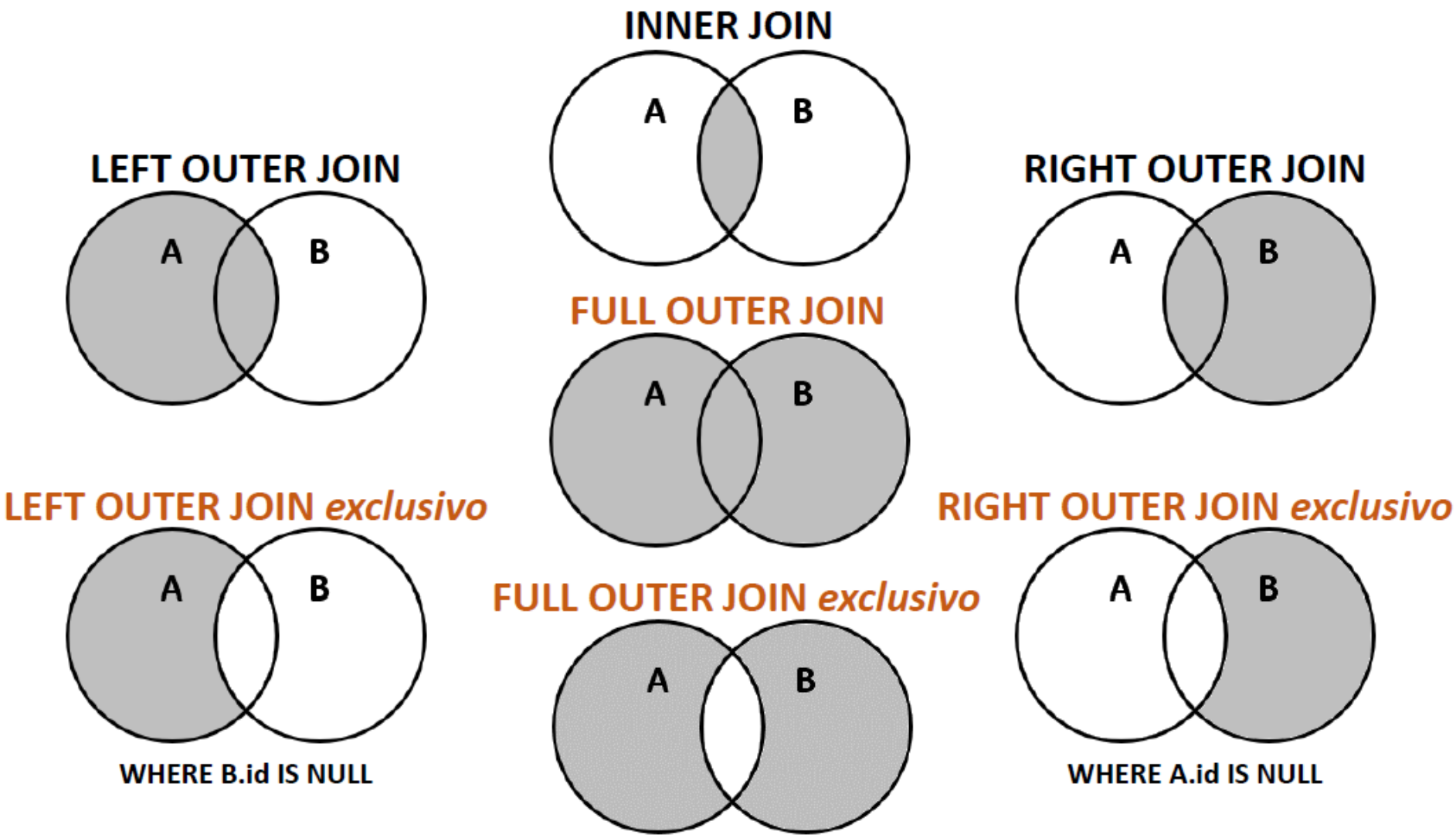
    1. Composición cruzada (producto cartesiano) CROSS JOIN
    2. Composición interna (intersección) INNER JOIN
    3. Composición externa OUTER JOIN:

        3.1. LEFT OUTER JOIN
        3.2. RIGHT OUTER JOIN
        3.3. FULL OUTER JOIN (no implementada en MySQL)

- Otras operaciones sobre conjuntos disponibles en MySQL: UNION y UNION ALL

Composiciones: JOIN

- La siguiente imagen muestra gráficamente el resultado de cada operación JOIN en las **consultas multitabla:**



Composiciones cruzadas: CROSS JOIN

- El resultado de una consulta multitabla en el que no se establezca la relación entre campos de todas las tablas involucradas, nos devolverá el **producto cartesiano** (todas las combinaciones de las filas de una tabla con las del resto de tablas).

Tabla: empleado

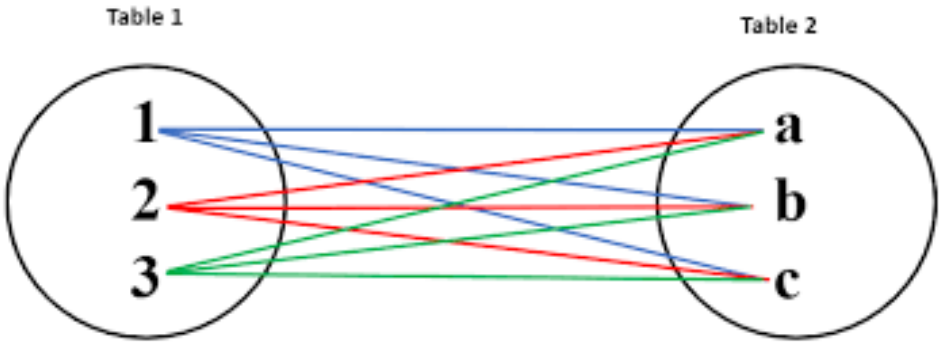
id	nombre	Id_departamento
1	Pepe	1
2	María	2
3	Juan	NULL

Tabla: departamento

id	nombre
1	Desarrollo
2	Sistemas
3	Recursos Humanos

SQL-1:
```
SELECT * FROM empleado, departamento;
```

SQL-2:
```
SELECT * FROM empleado CROSS JOIN departamento;
```



id	nombre	Id_departamento	Id	nombre
1	Pepe	1	1	Desarrollo
1	Pepe	1	2	Sistemas
1	Pepe	1	3	Recursos Humanos
2	María	2	1	Desarrollo
2	María	2	2	Sistemas
2	María	2	3	Recursos Humanos
3	Juan	NULL	1	Desarrollo
3	Juan	NULL	2	Sistemas
3	Juan	NULL	3	Recursos Humanos

Composiciones internas: INNER JOIN

- El resultado de una consulta multitabla en el que se relacionan las tablas mediante INNER JOIN, nos devolverá la **intersección** de ambas tablas (filas en las que se comparta un mismo valor en los campos relacionados).

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**

SQL-1:

```
SELECT *
FROM empleado, departamento
WHERE empleado.id_departamento = departamento.id;
```

SQL-2:

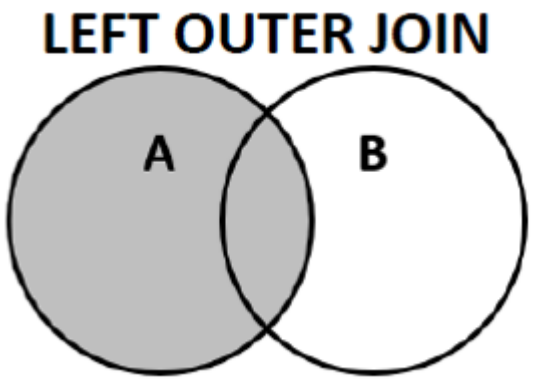
```
SELECT *
FROM empleado INNER JOIN departamento
ON empleado.id_departamento = departamento.id;

SELECT *
FROM a INNER JOIN b ON a.idb = b.id
INNER JOIN c ON a.idc= c.id;
```

El **resultado de la operación INNER JOIN** es:

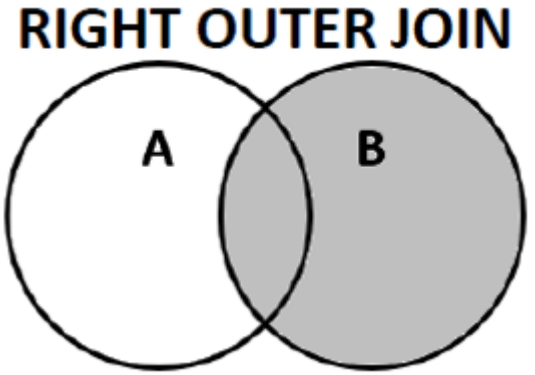
empleado.id	empleado.nombre	empleado.id_departamento	departamento.id	departamento.nombre
1	Pepe	1	1	Desarrollo
2	María	2	2	Sistemas

Composiciones externas: OUTER JOIN

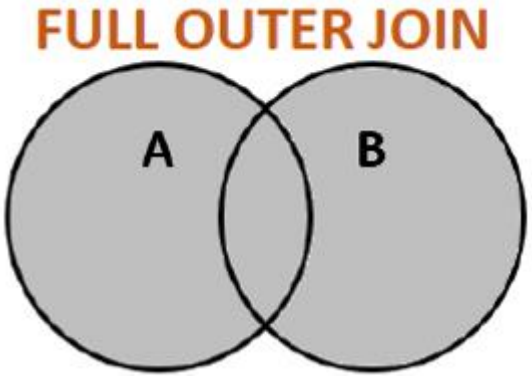
- El resultado de una consulta multitabla en el que se relacionan las tablas mediante **OUTER JOIN**, nos devolverá TANTO la intersección de ambas tablas (filas en las que se comparta un mismo valor en los campos relacionados). COMO también **las tuplas restantes de una u otra tabla o de ambas**.



Si, en una consulta, además de los registros que cumplen la condición de selección, también desea obtener todos los registros de la tabla izquierda.



Si, en una consulta, además de los registros que cumplen la condición de selección, también desea obtener todos los registros de la tabla derecha.



Si, en una consulta, además de los registros que cumplen la condición de selección, también desea obtener todos los registros de ambas tablas.

Comp. externa: LEFT OUTER JOIN

- El resultado de una consulta multitabla en el que se relacionan las tablas mediante **LEFT OUTER JOIN**, nos devolverá TANTO la intersección de ambas tablas (filas en las que se comparta un mismo valor en los campos relacionados). COMO también **las tuplas restantes de la tabla izquierda**.

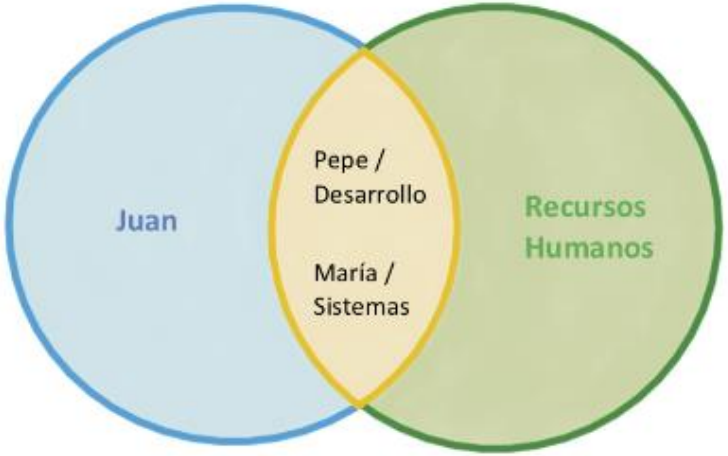
Tabla: empleado

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**



SQL-1: N/A

SQL-2:

```
SELECT *
FROM empleado LEFT JOIN departamento
ON empleado.id_departamento = departamento.id;
```

El **resultado de la operación LEFT JOIN** es:

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

Comp. externa: RIGHT OUTER JOIN

- El resultado de una consulta multitabla en el que se relacionan las tablas mediante **RIGHT OUTER JOIN**, nos devolverá TANTO la intersección de ambas tablas (filas en las que se comparta un mismo valor en los campos relacionados). COMO también **las tuplas restantes de la tabla derecha**.

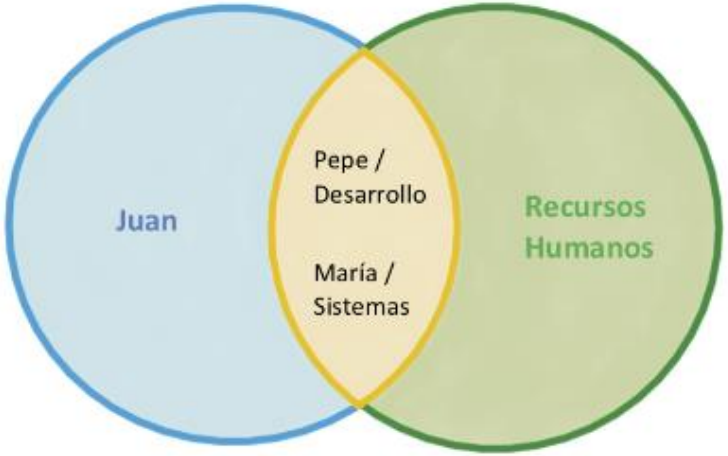
Tabla: empleado

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**



SQL-1: N/A

SQL-2:

```
SELECT *
FROM empleado RIGHT JOIN departamento
ON empleado.id_departamento = departamento.id;
```

El **resultado de la operación RIGHT JOIN** es:

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

Comp. externa: FULL OUTER JOIN

- El resultado de una consulta multitabla en el que se relacionan las tablas mediante **FULL OUTER JOIN**, nos devolverá TANTO la intersección de ambas tablas (filas en las que se comparta un mismo valor en los campos relacionados). COMO también **las tuplas restantes de la tabla izquierda** y **las tuplas resultantes de la tabla derecha**.

- La composición **FULL OUTER JOIN no está implementada en MySQL**, por lo tanto para poder simular esta operación será necesario hacer uso del operador **UNION**, que nos realiza la unión del resultado de dos consultas: `LEFT JOIN` UNION `RIGHT JOIN`

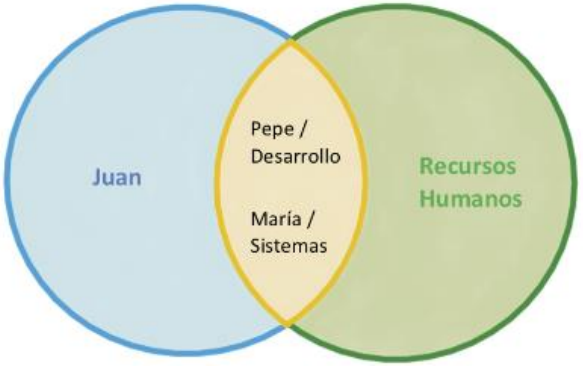
Tabla: empleado

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**



SQL-1: `N/A`

SQL-2:

```
SELECT *
FROM empleado LEFT JOIN departamento
ON empleado.id_departamento =
departamento.id

UNION

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

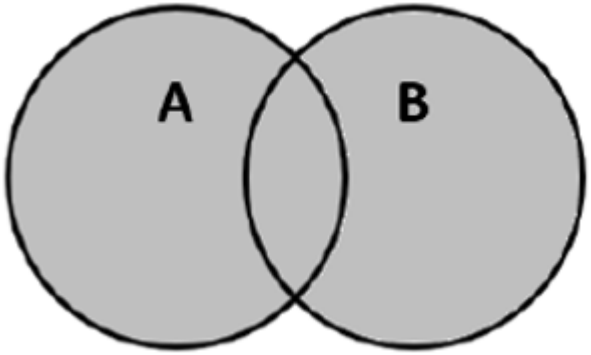
id	nombre	Id_departamento	Id	nombre
1	Pepe	1	1	Desarrollo
2	María	2	2	Sistemas
3	Juan	NULL	NULL	NULL
NULL	NULL	NULL	3	Recursos humanos

Composiciones: UNION / UNION ALL

- **UNION** es un operador binario de conjuntos usado para combinar el resultado de dos consultas SELECT.
- El resultado obtenido después de la operación de UNION es una colección de resultados de ambas tablas.
- Las siguientes condiciones son obligatorias:

    1. Ambas sentencias SELECT deben devolver el mismo número de campos y en el mismo orden.
    2. Los tipos de datos de dichos campos deben ser los mismo (o compatibles)

- La operación de **UNION** devuelve valores distintos.
- Si queremos recuperar también los valores repetidos, debemos usar la operación **UNION ALL**




```
SELECT * FROM A
UNION
SELECT * FROM B;
```

```
SELECT * FROM A
UNION ALL
SELECT * FROM B;
```