MySQL

Clausulas JOIN

Clausulas JOIN | SQL

Vamos a crear una nueva tabla en nuestra base de datos “Empresa”. En esta ocasión, vamos a crearla en una nueva pestaña, para no perder el trabajo anterior.

Para ello, botón derecho sobre la pestaña ‘Query1’ y pulsamos ***new tab***.

Clausulas JOIN | SQL

Desde allí crearemos la tabla “Productos” donde podremos asignar como llave foránea el ID de la tabla “Empleados”:

```
CREATE TABLE Articulos(
ID INT AUTO_INCREMENT PRIMARY KEY,
Marca VARCHAR(255),
Vendedor INT,
Modelo VARCHAR(255),
Stock INT,
Precio DECIMAL(10, 2),
Foreign key(Vendedor) references Empleados(ID)
);
```

Clausulas JOIN | SQL

Aprovechemos para conocer otra alternativa para la introducción de datos en MySQL:

```
INSERT INTO Articulos (Marca, Vendedor, Modelo, Stock, Precio)
VALUES
('Samsung', 1, 'Galaxy S21', 10, 999.99),
('Apple', 1, 'iPhone 12', 5, 1299.99),
('Sony', 2, 'PlayStation 5', 15, 499.99),
('LG', 3, 'OLED TV', 8, 1999.99),
('Nike', 2, 'Air Max 270', 12, 129.99),
('Adidas', 1, 'Ultraboost', 3, 159.99);
```

Clausulas JOIN | SQL

Hacemos una comprobación de los datos:

```
SELECT * from Articulos;
```

El LEFT JOIN es una operación que permite combinar filas de dos tablas basándose en una condición de igualdad especificada, tomando todas las filas de una tabla y las filas correspondientes de otra tabla.

Es decir, El LEFT JOIN nos permite combinar las filas de la tabla "empleados" con las filas correspondientes de la tabla “articulos" basándose en una condición de igualdad, es decir, que exista un valor común en una o más filas.

Clausulas JOIN | SQL

Vamos a realizar un ejemplo paso a paso para entender mejor cómo funciona el LEFT JOIN utilizando las tablas "empleados" y “articulos".

```
SELECT e.ID, e.Nombre, p.Marca, p.Modelo
FROM empleados e
LEFT JOIN articulos p ON e.ID = p.Vendedor;
```

En este caso, "e" y "p" son alias o abreviaturas que se utilizan para referirse a las tablas "empleados" y "productos", respectivamente. Los alias se utilizan para simplificar y hacer más legible la escritura de consultas SQL.

Clausulas JOIN | SQL

En una cláusula **LEFT JOIN**, la **tabla principal** se refiere a la tabla que aparece antes de la palabra clave "**LEFT JOIN**", y la **tabla secundaria** es la tabla que aparece después de la palabra clave "**LEFT JOIN**".

La **tabla principal** es aquella de la cual se conservan todas las filas, independientemente de si hay coincidencias en la tabla secundaria. En otras palabras, todas las filas de la **tabla principal** estarán presentes en el resultado, incluso si no hay filas coincidentes en la tabla secundaria.

La **tabla secundaria** es la tabla que se une a la **tabla principal**. Las filas de la **tabla secundaria** se combinan con las filas de la **tabla principal** basándose en una condición de unión específica. Si una fila de la **tabla secundaria** coincide con una fila de la **tabla principal** según la condición de unión, se incluirá en el resultado. Si no hay coincidencias, se mostrarán valores nulos para las columnas de la **tabla secundaria**.

Clausulas JOIN | SQL

Cuando se utiliza un alias, se coloca un nombre o letra seguida de un punto (por ejemplo, "e." o "p.") antes del nombre de la columna para indicar a qué tabla se está haciendo referencia.

"e.ID" se refiere a la columna "ID" de la tabla "empleados".
"e.Nombre" se refiere a la columna "Nombre" de la tabla "empleados".
"p.Marca" se refiere a la columna "Marca" de la tabla “articulos".
"p.Modelo" se refiere a la columna "Modelo" de la tabla “articulos".

El uso de alias ayuda a simplificar y clarificar la escritura de consultas SQL cuando se trabaja con múltiples tablas en una consulta JOIN.

Es importante tener en cuenta que el alcance del alias está limitado a la consulta actual.

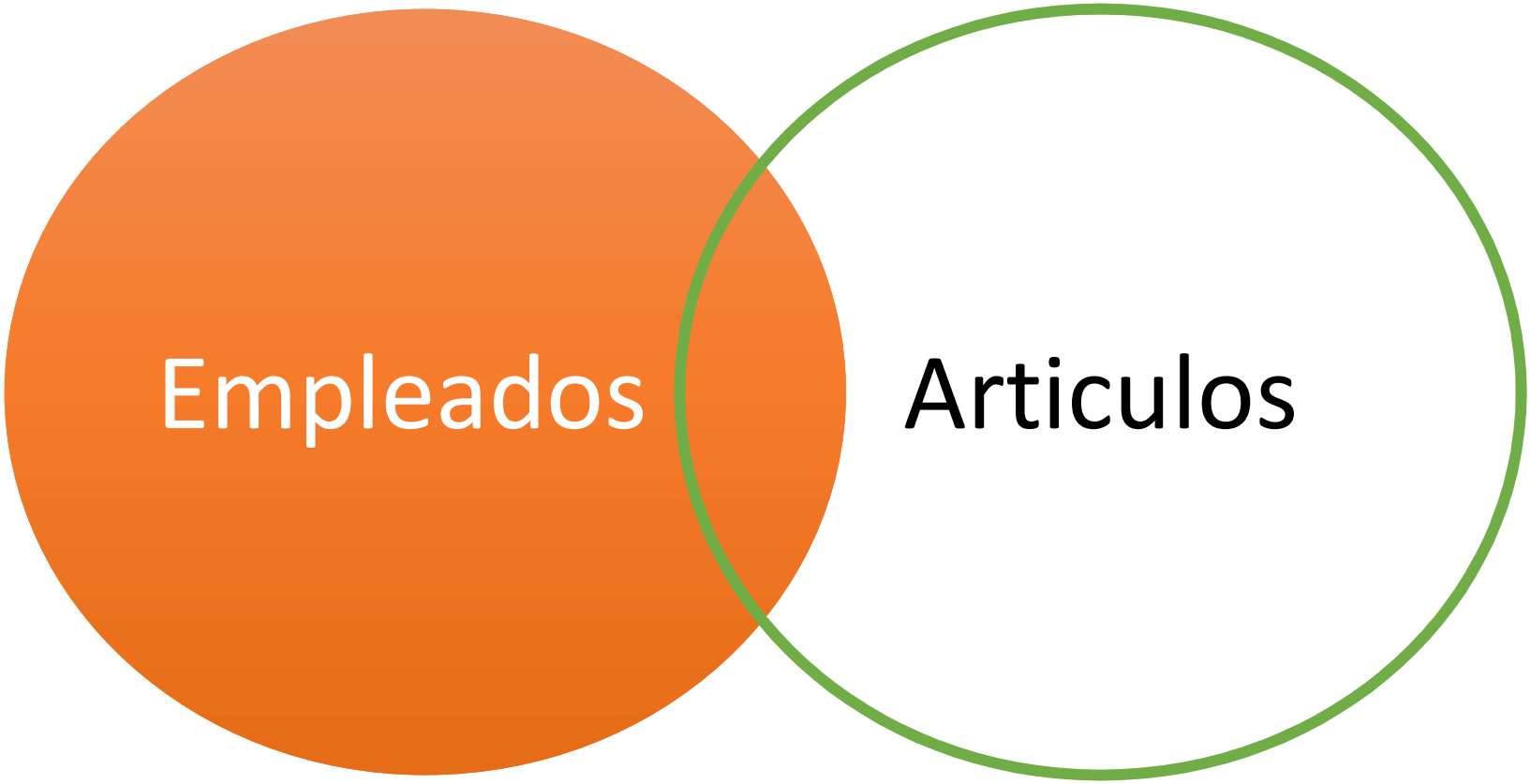
Clausulas JOIN | SQL

```
-- Primero seleccionamos los datos que deseamos obtener
SELECT e.ID, e.Nombre, p.Marca, p.Modelo
-- Indicamos la tabla principal y su alias
FROM empleados e
-- Por último, establecemos la condición de igualdad
LEFT JOIN articulos p ON e.ID = p.Vendedor;
```

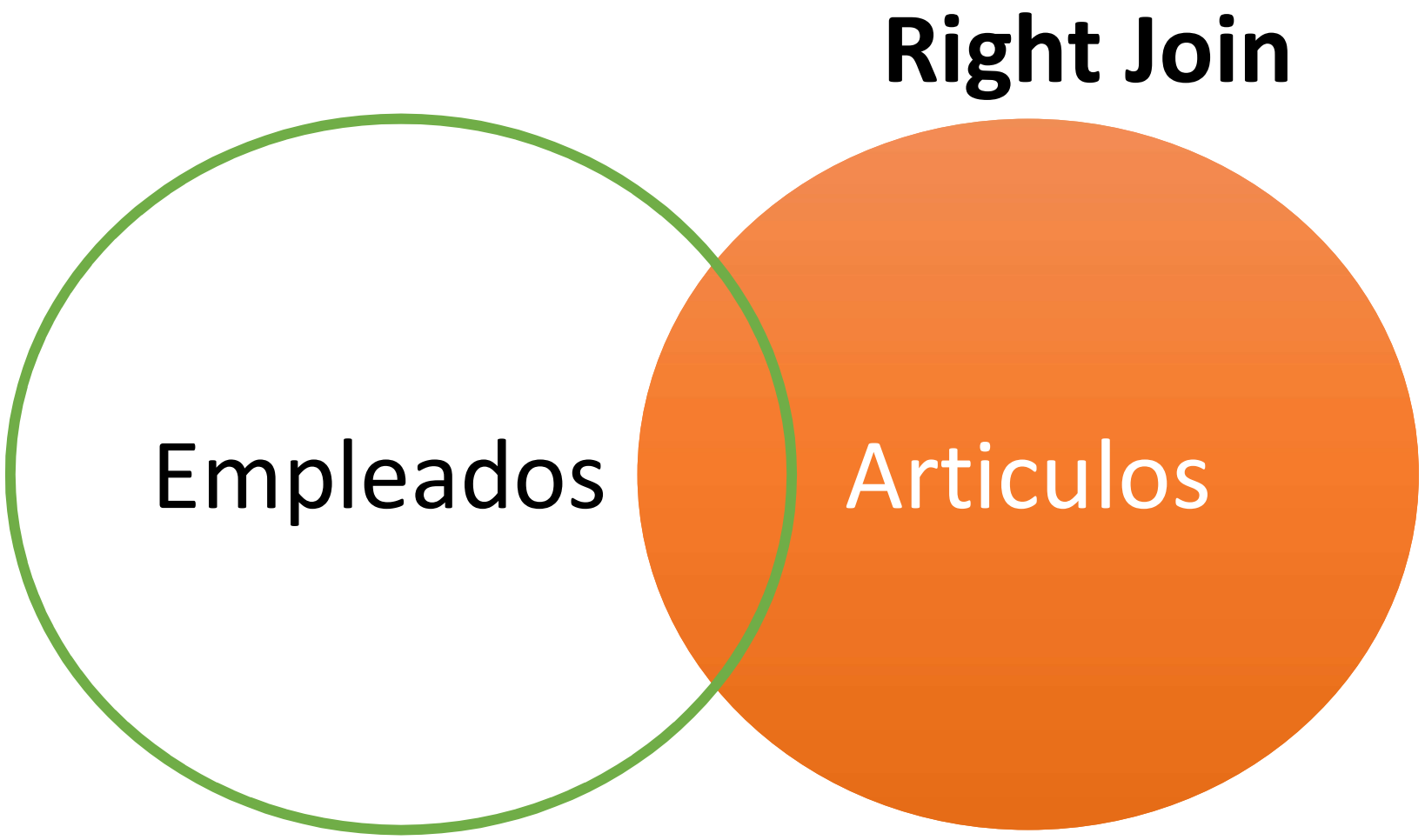
cuando se realiza el **LEFT JOIN**, se buscarán las coincidencias entre los valores de las columnas especificadas en la condición de igualdad, y se combinarán las filas correspondientes de ambas tablas en el resultado final. Si no hay coincidencias, se mostrará una fila con valores nulos para las columnas de la tabla que no tenga correspondencia.

Clausulas JOIN | SQL

Left Join



Clausulas JOIN | SQL



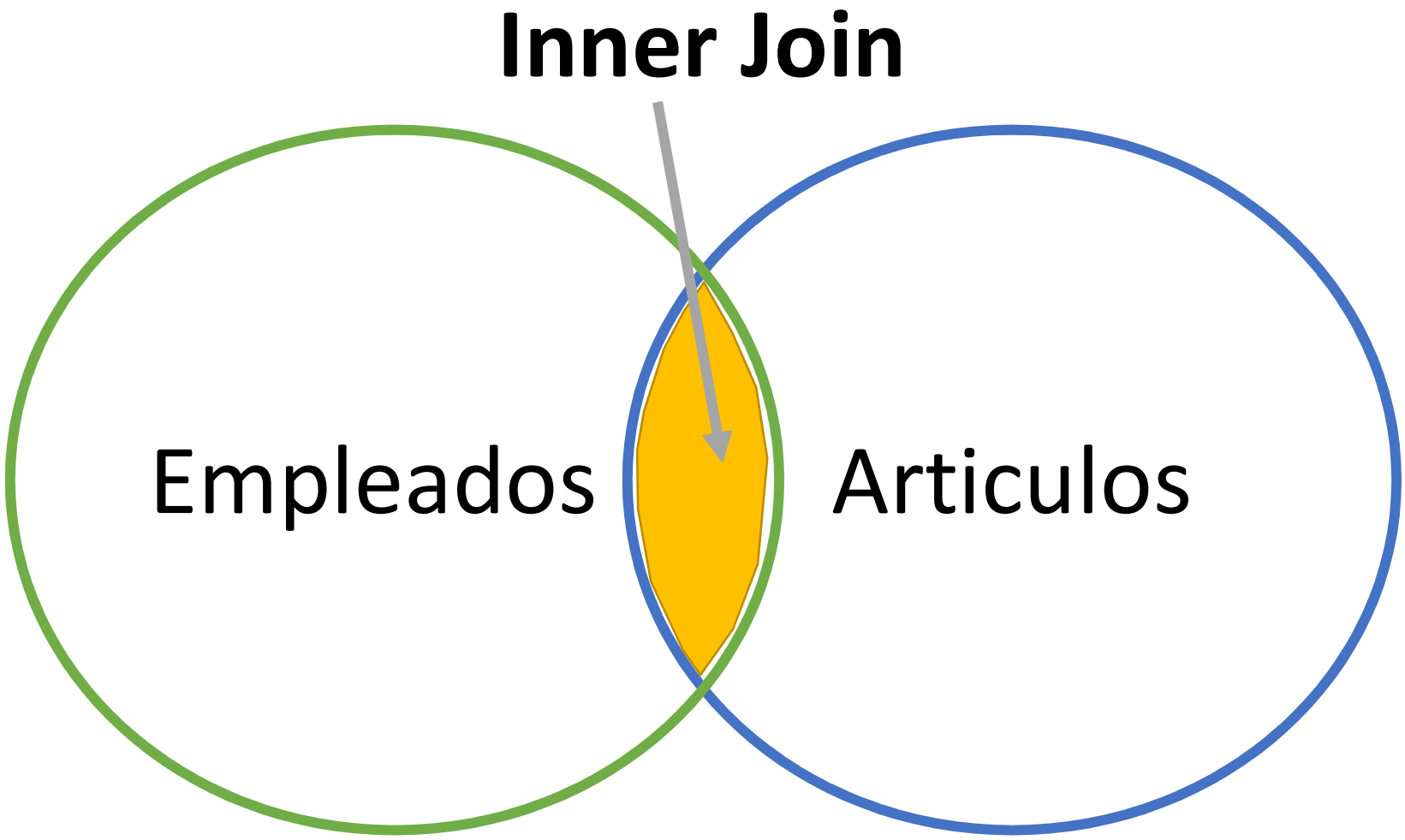
Clausulas JOIN | SQL

Al utilizar un Right Join estamos cambiando la tabla principal de nuestra relación, es decir, en este caso la tabla principal sería “Productos”, por lo que el resultado ahora sería diferente:

```
SELECT e.ID, e.Nombre, p.Marca, p.Modelo
FROM empleados e
RIGHT JOIN articulos p ON e.ID = p.Vendedor;
```

Es decir, la tabla principal es aquella en la que todos los datos son mostrados, pero solo se muestran los de la tabla secundaria que tengan alguna relación con la principal.

Clausulas JOIN | SQL



Clausulas JOIN | SQL

En el caso de Inner Join solo se mostrarán los datos que están directamente relacionados:

```
SELECT e.ID, e.Nombre, p.Marca, p.Modelo
FROM empleados e
INNER JOIN articulos p ON e.ID = p.Vendedor;
```

Es decir, en este caso se actúa como si no hubiera una tabla principal.

Clausulas JOIN | SQL

Por último veremos el caso del CROSS JOIN, que devuelve todas las posibles combinaciones de datos que se pueden generar entre los campos seleccionados de las tablas.

```
SELECT e.ID, e.Nombre, p.Marca, p.Modelo
FROM empleados e
CROSS JOIN articulos p ON e.ID = p.Vendedor;
```

En esta consulta, se seleccionan las columnas "ID" y "Nombre" de la tabla "empleados" y las columnas "Marca" y "Modelo" de la tabla “articulos". Al utilizar el CROSS JOIN, se generarán todas las combinaciones posibles de filas entre ambas tablas, sin importar si hay una relación lógica o condición de unión entre ellas.

Clausulas JOIN | SQL

El **CROSS JOIN** puede ser útil en ciertos escenarios, como cuando se requiere generar todas las posibles combinaciones de datos entre dos conjuntos de datos.

Sin embargo, es importante tener en cuenta que puede producir un número muy grande de filas resultantes, especialmente si las tablas tienen muchas filas.