Guía teórico practica de Consultas multitabla

¿Qué son los JOINs?

Los JOINs son una forma de combinar filas de dos o más tablas en una base de datos relacional, con base en una columna o columnas relacionadas entre ellas. Esto es muy útil cuando la información que necesitamos está distribuida en varias tablas y queremos obtener un conjunto de resultados que integre los datos de todas esas tablas.

Tipos de JOINs

Se debe hacer las bases de datos , copiar y mostrar evidencia de resultados de las consultas

Existen diferentes tipos de JOINs en MySQL:

INNER JOIN: Devuelve las filas cuando hay una coincidencia en ambas tablas.
LEFT JOIN: Devuelve todas las filas de la tabla izquierda y las filas coincidentes de la tabla derecha.
RIGHT JOIN: Devuelve todas las filas de la tabla derecha y las filas coincidentes de la tabla izquierda.
FULL OUTER JOIN: Devuelve todas las filas de ambas tablas, independientemente de si hay coincidencias.

Ejemplo práctico

Supongamos que tenemos dos tablas en nuestra base de datos: clientes y pedidos.

```sql
-- Creación de la tabla clientes
CREATE TABLE clientes (
    cliente_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL
);
```

```sql
-- Creación de la tabla pedidos
CREATE TABLE pedidos (
    pedido_id INT AUTO_INCREMENT PRIMARY KEY,
    cliente_id INT NOT NULL,
    fecha DATE NOT NULL,
    FOREIGN KEY (cliente_id) REFERENCES clientes(cliente_id)
);
```

```sql
-- Inserción de 10 registros en la tabla clientes
INSERT INTO clientes (nombre) VALUES
    ('Juan Pérez'),
    ('Ana López'),
    ('Pedro Ramón'),
    ('María García'),
    ('Carlos Torres'),
    ('Laura Martínez'),
    ('Andrés Guzmán'),
    ('Sofía Rodríguez'),
    ('Javier Sánchez'),
    ('Lucía Fernández');
```

```sql
-- Inserción de 10 registros en la tabla pedidos
INSERT INTO pedidos (cliente_id, fecha) VALUES
    (1, '2023-04-15'),
    (1, '2023-04-25'),
    (3, '2023-05-01'),
    (4, '2023-05-05'),
    (5, '2023-05-10'),
    (6, '2023-05-15'),
    (7, '2023-05-20'),
    (8, '2023-05-25'),
    (9, '2023-05-30'),
    (10, '2023-06-01');
```

INNER JOIN

```sql
SELECT c.nombre, p.pedido_id, p.fecha
FROM clientes c
INNER JOIN pedidos p ON c.cliente_id = p.cliente_id;
```

nombre	pedido_id	fecha
Juan Pérez	1	2023-04-15
Juan Pérez	2	2023-04-25
Pedro Ramón	3	2023-05-01
María García	4	2023-05-05
Carlos Torres	5	2023-05-10
Laura Martínez	6	2023-05-15
Andrés Guzmán	7	2023-05-20
Sofía Rodríguez	8	2023-05-25
Javier Sánchez	9	2023-05-30
Lucía Fernández	10	2023-06-01

1. Estamos seleccionando el nombre de la tabla clientes, el pedido_id y la fecha de la tabla pedidos.
2. INNER JOIN pedidos p: Aquí estamos indicando que queremos realizar un INNER JOIN con la tabla pedidos, y le asignamos el alias p.
3. ON c.cliente_id = p.cliente_id: Esta es la condición de unión (join condition) que especifica cómo se relacionan las dos tablas. En este caso, estamos uniendo las filas de ambas tablas donde el valor de cliente_id en la tabla clientes coincide con el valor de cliente_id en la tabla pedidos.

El INNER JOIN devuelve las filas que tienen coincidencias en ambas tablas. Es decir, solo se incluirán en el resultado aquellos clientes que tengan al menos un pedido y, para cada cliente, se mostrarán sus pedidos correspondientes.

LEFT JOIN

```sql
SELECT c.nombre, p.pedido_id, p.fecha
FROM clientes c
LEFT JOIN pedidos p ON c.cliente_id = p.cliente_id;
```

nombre	pedido_id	fecha
Juan Pérez	1	2023-04-15
Juan Pérez	2	2023-04-25
Ana López	*NULL*	*NULL*
Pedro Ramón	3	2023-05-01
María García	4	2023-05-05
Carlos Torres	5	2023-05-10
Laura Martínez	6	2023-05-15
Andrés Guzmán	7	2023-05-20
Sofía Rodríguez	8	2023-05-25
Javier Sánchez	9	2023-05-30
Lucía Fernández	10	2023-06-01

El LEFT JOIN es un tipo de JOIN que devuelve todas las filas de la tabla izquierda (en este caso, la tabla clientes) y las filas coincidentes de la tabla derecha (la tabla pedidos). Incluso si no hay coincidencias en la tabla derecha, se incluirán las filas de la tabla izquierda con valores NULL en las columnas de la tabla derecha.

El LEFT JOIN garantiza que se devuelvan todas las filas de la tabla clientes, incluso si no hay pedidos correspondientes para algún cliente en la tabla pedidos. En ese caso, las columnas de la tabla pedidos (pedido_id y fecha) tendrán valores NULL para esas filas.

RIGHT JOIN

```sql
SELECT c.nombre, p.pedido_id, p.fecha
FROM clientes c
RIGHT JOIN pedidos p ON c.cliente_id = p.cliente_id;
```

nombre	pedido_id	fecha
Juan Pérez	1	2023-04-15
Juan Pérez	2	2023-04-25
Pedro Ramón	3	2023-05-01
María García	4	2023-05-05
Carlos Torres	5	2023-05-10
Laura Martínez	6	2023-05-15
Andrés Guzmán	7	2023-05-20
Sofía Rodríguez	8	2023-05-25
Javier Sánchez	9	2023-05-30
Lucía Fernández	10	2023-06-01

El RIGHT JOIN devuelve todas las filas de la tabla pedidos (tabla derecha) y las filas coincidentes de la tabla clientes (tabla izquierda). En este caso, no hay pedidos sin cliente, por lo que no hay filas adicionales.

FULL JOIN

```sql
SELECT c.nombre, p.pedido_id, p.fecha
FROM clientes c
LEFT JOIN pedidos p ON c.cliente_id = p.cliente_id
UNION
SELECT c.nombre, p.pedido_id, p.fecha
FROM clientes c
RIGHT JOIN pedidos p ON c.cliente_id = p.cliente_id;
```

nombre	pedido_id	fecha
Juan Pérez	1	2023-04-15
Juan Pérez	2	2023-04-25
Ana López	*NULL*	*NULL*
Pedro Ramón	3	2023-05-01
María García	4	2023-05-05
Carlos Torres	5	2023-05-10
Laura Martínez	6	2023-05-15
Andrés Guzmán	7	2023-05-20
Sofía Rodríguez	8	2023-05-25
Javier Sánchez	9	2023-05-30
Lucía Fernández	10	2023-06-01

Al combinar los resultados del LEFT JOIN y del RIGHT JOIN mediante UNION, obtenemos un resultado similar al FULL OUTER JOIN, que incluye todas las filas de ambas tablas, independientemente de si hay coincidencias o no

```sql
CREATE TABLE empleado (
    id_empleado INT PRIMARY KEY,
    nombre VARCHAR(50),
    apellido VARCHAR(50),
    id_puesto INT
);
```

```sql
-- Crear tabla puesto_trabajo
CREATE TABLE puesto_trabajo (
    id_puesto INT PRIMARY KEY,
    nombre_puesto VARCHAR(50),
    salario DECIMAL(10, 2)
);
```

```sql
-- Crear tabla cargo
CREATE TABLE cargo (
    id_cargo INT PRIMARY KEY,
    nombre_cargo VARCHAR(50),
    id_puesto INT
);
```

```sql
ertar algunos datos de ejemplo
 INTO empleado (id_empleado, nombre, apellido, id_puesto)
VALUES
    (1, 'Juan', 'Pérez', 1),
    (2, 'María', 'García', 2),
    (3, 'Pedro', 'Rodríguez', 3),
    (4, 'Ana', 'Martínez', NULL);
```

```sql
INSERT INTO puesto_trabajo (id_puesto, nombre_puesto, salario)
VALUES
    (1, 'Gerente', 5000.00),
    (2, 'Supervisor', 3500.00),
    (3, 'Analista', 4200.00);

INSERT INTO cargo (id_cargo, nombre_cargo, id_puesto)
VALUES
    (1, 'Director General', 1),
    (2, 'Jefe de Departamento', 2),
    (3, 'Asistente', NULL);
```

Esta consulta devolverá los empleados que tienen un puesto de trabajo y un cargo asignado, mostrando el nombre, apellido, puesto, salario y cargo.

```sql
SELECT e.nombre, e.apellido, pt.nombre_puesto, pt.salario, c.nombre_cargo
FROM empleado e
INNER JOIN puesto_trabajo pt ON e.id_puesto = pt.id_puesto
INNER JOIN cargo c ON pt.id_puesto = c.id_puesto;
```

Esta consulta devolverá todos los empleados, incluso aquellos que no tienen un puesto de trabajo o un cargo asignado. Los valores nulos se mostrarán para los puestos o cargos no asignados.

```sql
SELECT e.nombre, e.apellido, pt.nombre_puesto, pt.salario, c.nombre_cargo
FROM empleado e
LEFT JOIN puesto_trabajo pt ON e.id_puesto = pt.id_puesto
LEFT JOIN cargo c ON pt.id_puesto = c.id_puesto;
```

Esta consulta devolverá todos los puestos de trabajo y cargos, incluso aquellos que no tienen un empleado asignado. Los valores nulos se mostrarán para los empleados no asignados.

```sql
SELECT e.nombre, e.apellido, pt.nombre_puesto, pt.salario, c.nombre_cargo
FROM empleado e
RIGHT JOIN puesto_trabajo pt ON e.id_puesto = pt.id_puesto
RIGHT JOIN cargo c ON pt.id_puesto = c.id_puesto;
```

¿Qué hace esta consulta?

```sql
SELECT pt.nombre_puesto, AVG(pt.salario) AS salario_promedio, COUNT(e.id_empleado) AS cantidad_empleados
FROM empleado e
INNER JOIN puesto_trabajo pt ON e.id_puesto = pt.id_puesto
GROUP BY pt.nombre_puesto;
```

¿Qué hace esta consulta?

```sql
SELECT e.nombre, e.apellido, pt.nombre_puesto, pt.salario, c.nombre_cargo
FROM empleado e
INNER JOIN puesto_trabajo pt ON e.id_puesto = pt.id_puesto
LEFT JOIN cargo c ON pt.id_puesto = c.id_puesto
WHERE pt.salario BETWEEN 3000 AND 5000
   AND c.nombre_cargo IN ('Director General', 'Jefe de Departamento');
```

Para la base de datos de ejemplo hacer:

1. Obtener el nombre de los empleados, el nombre de sus jefes y la ciudad donde trabajan, agrupados por ciudad.

```sql
SELECT e.nombre, j.nombre AS jefe, e.ciudad
FROM empleado e
LEFT JOIN empleado j ON e.jefe = j.idempleado
GROUP BY e.ciudad;
```

2. Obtener el nombre de los clientes, el nombre de la ciudad donde viven y el nombre del país, para los clientes de EE.UU. y Canadá:

```sql
SELECT c.nombrecompania, c.ciudad, p.nombrepais
FROM cliente c
JOIN pais p ON c.pais = p.nombrepais
WHERE p.nombrepais IN ('USA', 'Canada')
ORDER BY c.nombrecompania;

```

3. Obtener los productos vendidos, la cantidad total vendida y el precio promedio de venta, para los productos con un precio promedio entre $10 y $20:

```sql
SELECT p.nombreproducto, SUM(fv.cantidad) AS cantidad_total, AVG(fv.precioUnidad) AS precio_promedio
FROM factventa fv
JOIN producto p ON fv.idproducto = p.idproducto
GROUP BY p.nombreproducto
HAVING AVG(fv.precioUnidad) BETWEEN 10 AND 20
ORDER BY cantidad_total DESC;
```

4. Obtener el nombre de los empleados que han realizado ventas, el número de pedidos que han atendido y el total de ventas:

```sql
SELECT e.nombre, COUNT(fv.idpedido) AS num_pedidos, SUM(fv.precioUnidad * fv.cantidad) AS total_ventas
FROM empleado e
JOIN factventa fv ON e.idempleado = fv.idempleado
GROUP BY e.nombre
ORDER BY total_ventas DESC;
```

5. Obtener el nombre de los clientes y el número de pedidos que han realizado, solo para aquellos clientes que han realizado más de 5 pedidos:

```sql
SELECT c.nombrecompania, COUNT(fv.idpedido) AS num_pedidos
FROM cliente c
JOIN factventa fv ON c.idcliente = fv.idcliente
GROUP BY c.nombrecompania
HAVING COUNT(fv.idpedido) > 5
ORDER BY num_pedidos DESC;
```

6. Obtener el nombre de los productos y la categoría a la que pertenecen, junto con el precio máximo y mínimo de venta:

```sql
SELECT p.nombreproducto, c.nombrecategoria, MAX(fv.precioUnidad) AS precio_maximo, MIN(fv.precioUnidad) AS
precio_minimo
FROM producto p
JOIN categoria c ON p.idcategoria = c.idcategoria
JOIN factventa fv ON p.idproducto = fv.idproducto
GROUP BY p.nombreproducto, c.nombrecategoria;
```

7. Obtener el nombre de los empleados y el número de ciudades diferentes a las que han enviado pedidos:

```sql
SELECT e.nombre, COUNT(DISTINCT fv.idciudaddestino) AS num_ciudades
FROM empleado e
JOIN factventa fv ON e.idempleado = fv.idempleado
GROUP BY e.nombre
ORDER BY num_ciudades DESC;
```

8. Obtener el nombre de los productos, el nombre de la categoría y el nombre del proveedor, ordenados por categoría y luego por proveedor:

```sql
SELECT p.nombreproducto, c.nombrecategoria, pr.nombrecompania
FROM producto p
JOIN categoria c ON p.idcategoria = c.idcategoria
JOIN proveedor pr ON p.idproveedor = pr.idproveedor
ORDER BY c.nombrecategoria, pr.nombrecompania;
```

9. Obtener el nombre de los empleados que no son jefes, el nombre de su jefe y la ciudad donde trabajan:

```sql
SELECT e.nombre AS empleado, j.nombre AS jefe, e.ciudad
FROM empleado e
LEFT JOIN empleado j ON e.jefe = j.idempleado
WHERE e.jefe IS NOT NULL;
```

10. Obtener el nombre de los empleados, el nombre de su jefe y el total de ventas que han realizado:

```sql
SELECT e.nombre AS empleado, j.nombre AS jefe, SUM(fv.precioUnidad * fv.cantidad) AS total_ventas
FROM empleado e
LEFT JOIN empleado j ON e.jefe = j.idempleado
LEFT JOIN factventa fv ON e.idempleado = fv.idempleado
GROUP BY e.nombre, j.nombre
ORDER BY total_ventas DESC;
```

11. Obtener el nombre de los productos, el nombre de la categoría y la cantidad total vendida de cada producto

```sql
SELECT p.nombreproducto, c.nombrecategoria, SUM(fv.cantidad) AS cantidad_total
FROM producto p
JOIN categoria c ON p.idcategoria = c.idcategoria
LEFT JOIN factventa fv ON p.idproducto = fv.idproducto
GROUP BY p.nombreproducto, c.nombrecategoria
ORDER BY cantidad_total DESC;
```

12. Obtener el nombre de los productos y el nombre de los proveedores que los suministran, pero solo para los productos que tienen un stock disponible inferior a 6:

```sql
SELECT p.nombreproducto, pr.nombrecompania
FROM producto p
JOIN proveedor pr ON p.idproveedor = pr.idproveedor
WHERE p.stockenunidades < 6
```

13. Obtener el nombre de los productos, el nombre de la categoría y el total de ventas de cada producto, pero solo para los productos que han generado ventas superiores al promedio:

```sql
SELECT p.nombreproducto, c.nombrecategoria, SUM(fv.precioUnidad * fv.cantidad) AS total_ventas
FROM producto p
JOIN categoria c ON p.idcategoria = c.idcategoria
JOIN factventa fv ON p.idproducto = fv.idproducto
GROUP BY p.nombreproducto, c.nombrecategoria
HAVING SUM(fv.precioUnidad * fv.cantidad) > (
    SELECT AVG(fv2.precioUnidad * fv2.cantidad)
    FROM factventa fv2
)
ORDER BY total_ventas DESC;
```

14. Obtener el nombre de los clientes, el total de pedidos que han realizado y el valor promedio de sus pedidos:

```sql
SELECT c.nombrecompania, COUNT(fv.idpedido) AS total_pedidos, AVG(fv.precioUnidad * fv.cantidad) AS valor_promedio
FROM cliente c
JOIN factventa fv ON c.idcliente = fv.idcliente
GROUP BY c.nombrecompania
ORDER BY valor_promedio DESC;
```

15. Obtener el nombre del producto con el nombre de su categoría.
16. Nombre del producto con mayor descuento.

Haga el siguiente ejercicio de practica de consulta multitabla.

```sql
-- Creación de la tabla Inmueble
CREATE TABLE Inmueble (
    idInmueble INT PRIMARY KEY,
    tipo VARCHAR(50),
    superficie DECIMAL(10,2),
    numHabitaciones INT
);
```

```sql
-- Creación de la tabla Dirección
CREATE TABLE Direccion (
    idDireccion INT PRIMARY KEY,
    calle VARCHAR(100),
    ciudad VARCHAR(50),
    codigoPostal VARCHAR(10),
    idInmueble INT,
    FOREIGN KEY (idInmueble) REFERENCES Inmueble(idInmueble)
);
```

```sql
-- Creación de la tabla Vendedor
CREATE TABLE Vendedor (
    idVendedor INT PRIMARY KEY,
    nombre VARCHAR(50),
    apellidos VARCHAR(50),
    email VARCHAR(100)
);
```

```sql
-- Creación de la tabla DatosVenta
CREATE TABLE DatosVenta (
    idVenta INT PRIMARY KEY,
    fechaVenta DATE,
    precioVenta DECIMAL(15,2),
    idInmueble INT,
    idVendedor INT,
    FOREIGN KEY (idInmueble) REFERENCES Inmueble(idInmueble),
    FOREIGN KEY (idVendedor) REFERENCES Vendedor(idVendedor)
);
```

-- 1. Obtener todos los inmuebles con su dirección y vendedor

-- 2. Obtener los inmuebles vendidos en un rango de fechas

-- 3. Obtener los inmuebles de un vendedor específico

-- 4. Obtener la dirección de un inmueble específico

-- 5. Obtener el precio más alto y bajo de venta

-- 6. Obtener la cantidad de inmuebles por tipo

-- 7. Obtener los inmuebles ordenados por superficie descendente

-- 8. Obtener los inmuebles con 3 o más habitaciones

-- 9. Obtener el total de ventas por vendedor

-- 10. Obtener los inmuebles sin dirección registrada