COMBINACIÓN DE TABLAS

Técnica utilizada para combinar dos o más tablas en una sola tabla, basada en una o varias columnas que tienen en común. Esto se logra utilizando la cláusula **JOIN** en una consulta SQL. El objetivo de la combinación de tablas es recuperar datos de varias tablas relacionadas en una sola consulta.

- **INNER JOIN**: El más común. Se seleccionan las filas de ambas tablas que cumplen con una condición de combinación.
- **LEFT JOIN**: Devuelve todas las filas de la tabla de la izquierda (la primera tabla que se menciona en la consulta) y las filas de la tabla de la derecha que cumplen con la condición de combinación. Si una fila de la tabla de la izquierda no tiene correspondencia en la tabla de la derecha, se rellenan los valores de las columnas de la tabla de la derecha con NULL.
- **RIGHT JOIN**: Similar al left join, pero devuelve todas las filas de la tabla de la derecha y las filas de la tabla de la izquierda que cumplen con la condición de combinación. Si una fila de la tabla de la derecha no tiene correspondencia en la tabla de la izquierda, se rellenan los valores de las columnas de la tabla de la izquierda con NULL.
- **FULL OUTER JOIN**: Devuelve todas las filas de ambas tablas, incluyendo las filas que no tienen correspondencia en la otra tabla. Si una fila no tiene correspondencia en la otra tabla, se rellenan los valores de las columnas correspondientes con NULL. Sin embargo, no todos los sistemas de bases de datos soportan el full outer join de manera nativa. En lugar de ello, se puede utilizar la combinación de un left join y un right join.

EJEMPLO

```
CREATE DATABASE ventas21;
CREATE TABLE clientes (
    id_cliente INT PRIMARY KEY,
    nombre VARCHAR(50),
    email VARCHAR(50)
);

CREATE TABLE pedidos (
    id_pedido INT PRIMARY KEY,
    id_cliente INT,
    fecha_pedido DATE,
    cantidad FLOAT,
    FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente)
);

INSERT INTO clientes VALUES (1, 'Juan', 'juan@gmail.com');
INSERT INTO clientes VALUES (2, 'Pedro', 'pedro@gmail.com');
INSERT INTO clientes VALUES (3, 'Maria', 'maria@gmail.com');

INSERT INTO pedidos VALUES (1, 1, '2022-01-01', 10.0);
```

```
INSERT INTO pedidos VALUES (2, 1, '2022-02-01', 20.0);
INSERT INTO pedidos VALUES (3, 2, '2022-02-15', 15.0);
```

INNER JOIN

Por ejemplo, si queremos obtener la información del nombre de los clientes y la fecha de sus pedidos, podemos utilizar un inner join de la siguiente manera

```
SELECT clientes.nombre, pedidos.fecha_pedido
FROM clientes
INNER JOIN pedidos ON clientes.id_cliente = pedidos.id_cliente;
```

En esta consulta, estamos seleccionando el nombre de los clientes y la fecha de sus pedidos. Utilizamos la cláusula INNER JOIN para combinar las tablas clientes y pedidos, utilizando la columna id_cliente como condición de combinación

RESULTADO

nombre	fecha_pedido
Juan	2022-01-01
Juan	2022-02-01
Pedro	2022-02-15

LEFT JOIN

Por ejemplo, si queremos obtener información del nombre de los clientes y la fecha de sus pedidos (incluyendo a los clientes que no han realizado pedidos aún), podemos utilizar un left join de la siguiente manera:

```
SELECT clientes.nombre, pedidos.fecha_pedido
FROM clientes
LEFT JOIN pedidos ON clientes.id_cliente = pedidos.id_cliente;
```

En esta consulta, estamos seleccionando el nombre de los clientes y la fecha de sus pedidos, utilizando un left join para combinar las tablas clientes y pedidos. Como resultado, obtenemos todas las filas de la tabla clientes, y las filas de la tabla pedidos que cumplen con la condición de combinación. Si un cliente no ha realizado ningún pedido aún, la columna fecha_pedido será NULL.

RESULTADO

nombre	fecha_pedido
Juan	2022-01-01
Juan	2022-02-01

nombre	fecha_pedido
Pedro	2022-02-15
Maria	NULL

RIGHT JOIN

Por ejemplo, si queremos obtener información del nombre de los clientes y la fecha de sus pedidos (incluyendo los pedidos sin cliente correspondiente), podemos utilizar un right join de la siguiente manera:

```
SELECT clientes.nombre, pedidos.fecha_pedido
FROM clientes
RIGHT JOIN pedidos ON clientes.id_cliente = pedidos.id_cliente;
```

En esta consulta, estamos seleccionando el nombre de los clientes y la fecha de sus pedidos, utilizando un right join para combinar las tablas clientes y pedidos. Como resultado, obtenemos todas las filas de la tabla pedidos, y las filas de la tabla clientes que cumplen con la condición de combinación. Si un pedido no tiene un cliente correspondiente, la columna nombre será NULL

RESULTADO

nombre	fecha_pedido
Juan	2022-01-01
Juan	2022-02-01
Pedro	2022-02-15
NULL	2022-03-01

FULL OUTER JOIN

Por ejemplo, si queremos obtener todas las filas de ambas tablas, incluyendo las filas que no tienen correspondencia en la otra tabla.:

```
SELECT clientes.nombre, pedidos.fecha_pedido
FROM clientes
FULL OUTER JOIN pedidos
ON clientes.id_cliente = pedidos.id_cliente;
```

En este caso, estamos utilizando la cláusula FULL OUTER JOIN para obtener todas las filas de ambas tablas, incluyendo las filas que no tienen correspondencia en la otra tabla.

RESULTADO

nombre	fecha_pedido

nombre	fecha_pedido
Juan	2022-01-01
Juan	2022-02-01
Pedro	2022-02-15
Maria	NULL
NULL	2022-03-01
Ana	NULL

Como se puede ver, el resultado incluye todas las filas de ambas tablas. En los casos en que no hay correspondencia entre las filas de ambas tablas, se incluyen valores nulos. En este ejemplo, la fila correspondiente a Ana de la tabla clientes no tiene correspondencia en la tabla pedidos, por lo que se incluye con el valor de NULL en la columna fecha_pedido. De manera similar, la fila correspondiente a Maria en la tabla pedidos no tiene correspondencia en la tabla clientes, por lo que se incluye con el valor de NULL en la columna nombre.

EN RESUMEN

En resumen, las diferentes combinaciones de tablas en SQL nos permiten realizar consultas complejas que nos permiten obtener información de diferentes fuentes de datos de manera más eficiente y flexible. El inner join nos permite obtener sólo las filas que tienen correspondencia en ambas tablas, el left join nos permite obtener todas las filas de la tabla de la izquierda y las filas correspondientes de la tabla de la derecha, el right join nos permite obtener todas las filas de la tabla de la derecha y las filas correspondientes de la tabla de la izquierda, y el full outer join nos permite obtener todas las filas de ambas tablas, incluyendo las filas que no tienen correspondencia en la otra tabla.

UNIÓN NATURAL

Es una operación que combina dos tablas en una sola tabla utilizando todas las columnas con nombres idénticos como condiciones de combinación. En otras palabras, la unión natural se utiliza cuando se desea combinar dos tablas que tienen columnas en común, pero no se necesita especificar las columnas de combinación de forma explícita.

EJEMPLO

Considere las tablas empleados y departamentos que tienen una columna en común llamada departamento_id. Podemos unir estas dos tablas utilizando la cláusula NATURAL JOIN:

```
SELECT *
SELECT id_cliente, nombre, NULL AS fecha_pedido
FROM clientes
UNION NATURAL
SELECT id_cliente, NULL AS nombre, fecha_pedido
FROM pedidos;
```

EXPLICACIÓN

En esta consulta, estamos utilizando UNION NATURAL para combinar dos consultas que seleccionan diferentes columnas de dos tablas diferentes. La primera consulta selecciona las columnas id_cliente y nombre de la tabla clientes, y añade una columna adicional NULL llamada fecha_pedido. La segunda consulta selecciona las columnas id_cliente y fecha_pedido de la tabla pedidos, y añade una columna adicional NULL llamada nombre. Al utilizar UNION NATURAL, la consulta combina los resultados de ambas consultas, eliminando las filas duplicadas y manteniendo las columnas en común (id_cliente en este caso).

RESULTADO

id_cliente	nombre	fecha_pedido
1	Juan	NULL
2	Pedro	NULL
3	Maria	NULL
4	Ana	NULL
1	NULL	2022-01-01
1	NULL	2022-02-01
2	NULL	2022-02-15
NULL	NULL	2022-03-01

omo se puede ver, el resultado combina las filas de ambas tablas, manteniendo las columnas en común (id_cliente en este caso) y eliminando las filas duplicadas. Las columnas adicionales (nombre y fecha_pedido) se incluyen en las filas correspondientes, y se rellenan con valores nulos en caso de no tener correspondencia en la otra tabla.

EN RESUMEN

En resumen, UNION NATURAL nos permite combinar los resultados de dos consultas que tienen las mismas columnas en común, eliminando las filas duplicadas y manteniendo las columnas en común en el resultado combinado. Esto es útil en casos en los que necesitamos obtener información de dos fuentes de datos diferentes que tienen información relacionada pero no necesariamente idéntica.