INSTITUTO NACIONAL DE MEXICALI

INGENIERIA EN SISTEMAS COMPUTACIONALES

FUNDAMENTOS EN BASE DE DATOS

COMO HACER UN CROSS SIN JOIN

ALUMNO:

ELIAN QUIÑONEZ MORFIN

MAESTRO:

JOSE RAMON BOGARIN VALENZUELA

SEMESTRE: 4

¿Qué es el CROSS JOIN en SQL?

El CROSS JOIN es un tipo de JOIN en SQL que devuelve el **producto cartesiano** de dos tablas. Es decir, **combina cada fila de la primera tabla con cada fila de la segunda tabla**, generando un conjunto de resultados donde todas las combinaciones posibles aparecen en la salida.

¿Cómo funciona CROSS JOIN en SQL?

Cuando se realiza un CROSS JOIN entre dos tablas:

- Si la primera tabla tiene m filas y la segunda tiene n filas, el resultado final tendrá m * n filas.
- No hay una condición explícita de unión (ON), lo que significa que todas las combinaciones posibles de las filas de ambas tablas serán devueltas.

¿Qué métodos usar como alternativa del CROSS JOIN?

1. **INNER JOIN con una condición de cruce**

Si las tablas tienen una columna en común, podemos usar un INNER JOIN en lugar de un CROSS JOIN, lo que reducirá la cantidad de combinaciones a solo aquellas que tienen coincidencias.

Ejemplo de código:

```
SELECT c.nombre, p.pedido_id, p.producto
FROM Clientes c
INNER JOIN Pedidos p ON c.cliente_id = p.cliente_id;
```

Ventajas:

- Evita registros sin coincidencias, mostrando solo relaciones válidas.
- Optimiza la consulta al filtrar datos desde el principio.

Desventajas:

- Si hay registros en una tabla que no tienen coincidencia en la otra, estos se pierden en el resultado.

2. **OUTER JOIN (LEFT JOIN, RIGHT JOIN, FULL JOIN)**

El OUTER JOIN nos permite unir tablas pero manteniendo las filas aunque no haya coincidencia.

2.1 LEFT JOIN (Unión externa izquierda)

Devuelve todos los registros de la tabla izquierda y, si hay coincidencias en la tabla derecha, también las incluye. Si no hay coincidencia, los valores de la tabla derecha aparecerán como NULL.

Ejemplo

```
SELECT c.nombre, p.pedido_id, p.producto
FROM Clientes c
LEFT JOIN Pedidos p ON c.cliente_id = p.cliente_id;
```

Ventajas:

- Mantiene todos los datos de la tabla izquierda, incluso si no hay coincidencias.

Desventajas:

- Puede generar valores NULL, lo que requiere tratamientos adicionales en los informes.

2.2 RIGHT JOIN (Unión externa derecha)

Funciona igual que el LEFT JOIN, pero manteniendo todos los registros de la tabla derecha.

Ejemplo:

```
SELECT c.nombre, p.pedido_id, p.producto
FROM Clientes c
RIGHT JOIN Pedidos p ON c.cliente_id = p.cliente_id;
```

Ventajas:

- Se asegura de que todos los registros de la tabla derecha estén en el resultado.

Desventajas:

- Como en LEFT JOIN, aparecen valores NULL cuando no hay coincidencias.

2.3 FULL JOIN (Unión externa completa)

Devuelve todos los registros de ambas tablas, llenando los valores faltantes con NULL cuando no hay coincidencia.

Ejemplo:

```
SELECT c.nombre, p.pedido_id, p.producto
FROM Clientes c
FULL JOIN Pedidos p ON c.cliente_id = p.cliente_id;
```

Ventajas:

- Mantiene todos los datos de ambas tablas.

Desventajas:

- Puede generar muchos valores NULL, lo que dificulta su uso.

3. UNION (Concatenación de resultados sin combinaciones)

Se usa cuando queremos unir los resultados de dos consultas en una sola tabla sin hacer un cruce de datos.

Ejemplo:

```
SELECT c.nombre, p.*
FROM Clientes c
CROSS APPLY (SELECT * FROM Pedidos p WHERE c.cliente_id = p.cliente_id) p;
```

Ventajas:

- Es útil cuando queremos unir datos de diferente tipo sin hacer combinaciones.

Desventajas:

- Solo funciona si las columnas de ambas consultas tienen el mismo número y tipo de datos.

4. **CROSS APPLY y OUTER APPLY (SQL Server)**

Se usa para aplicar una subconsulta dependiente a cada fila de la tabla principal.

Ejemplo:

SELECT c.nombre, p.*

FROM Clientes c

CROSS APPLY (SELECT * FROM Pedidos p WHERE c.cliente_id = p.cliente_id) p;

Ventajas:

- Más eficiente en algunos casos que CROSS JOIN.

Desventajas:

- Solo está disponible en SQL Server y no en MySQL o PostgreSQL.

5. **Producto cartesiano sin CROSS JOIN**

Si simplemente mencionamos dos tablas sin JOIN, SQL genera un producto cartesiano.

Ejemplo:

SELECT c.nombre, p.producto

FROM Clientes c, Pedidos p;

Ventajas:

- Es simple de escribir.

Desventajas:

- Puede generar millones de combinaciones innecesarias y ralentizar el sistema

Conclusión

Existen varias alternativas al CROSS JOIN en SQL, como INNER JOIN, OUTER JOIN, UNION, CROSS APPLY y el uso de producto cartesiano sin JOIN. Cada método tiene su propósito y optimiza el rendimiento según el caso de uso. La mejor opción dependerá de la relación entre las tablas y la cantidad de datos a procesar, evitando combinaciones innecesarias y mejorando la eficiencia de las consultas.