JOIN

SISTEMAS DE INFORMACION 1
PROCESAMIENTO DE DATOS

INVESTIGACION

EQUIPO:
CARLOS ALEXIS VIDAL MARQUEZ
ROBERTO LEOS PEREZ
DIANA AYLIN BAUTISTA MORALES

19 DE JULIO DE 2024

JOIN

SQL JOIN es una operación de base de datos que combina filas de dos o más tablas basándose en una columna relacionada entre ellas. Join es el proceso de tomar datos de varias tablas y colocarlos en una vista generada. Por tanto, una instrucción de “SQL JOIN” en un comando Select combina las columnas entre una o más tablas en una base de datos relacional y retorna a un conjunto de datos.

Su utilidad principal como lo dice el significado es que nos permite recopilar datos de diferentes tablas y crear una relación de ellos para después crear un único conjunto de datos.

Tipos principales de JOIN:

INNER JOIN:
Devuelve solo las filas donde hay una coincidencia en ambas tablas.

LEFT JOIN:
Devuelve todas las filas de la tabla izquierda y las filas coincidentes de la tabla derecha. Si no hay coincidencia en la tabla derecha, se mostrarán valores NULL para las columnas de esa tabla.

RIGHT JOIN:
Devuelve todas las filas de la tabla derecha y las filas coincidentes de la tabla izquierda. Si no hay coincidencia en la tabla izquierda, se mostrarán valores NULL para las columnas de esa tabla.

FULL OUTER JOIN:
Devuelve todas las filas de ambas tablas, incluso si no hay coincidencia en la tabla opuesta. Se mostrarán valores NULL para las columnas de la tabla que no tenga coincidencia.

CROSS JOIN:
Devuelve el producto cartesiano de las dos tablas, es decir, cada fila de la primera tabla se combina con cada fila de la segunda tabla.

INNER JOIN

INNER JOIN, también conocido como combinación equi, es el tipo de combinación más utilizado. Esta combinación se utiliza para recuperar filas de varias tablas comparando el valor de un campo que coincide entre las tablas. Los campos que se combinan deben tener tipos de datos similares, y no se pueden combinar los tipos de datos MEMO y OLEOBJECT.

Para generar una instrucción INNER JOIN, utilice las palabras clave INNER JOIN en la cláusula FROM de una instrucción SELECT.

Sintaxis general:

SELECT tabla1.columna, tabla2.columna FROM tabla1 INNER JOIN tabla2 ON tabla1.columna_comun = tabla2.columna_comun;

Ejemplo:

Supongamos que tienes dos tablas: Clientes y Pedidos. La tabla Clientes tiene información sobre los clientes (ID, nombre, etc.), y la tabla Pedidos contiene información sobre los pedidos realizados (ID, ID_Cliente, fecha, etc.). Quieres obtener una lista de clientes que han realizado pedidos, junto con los detalles de esos pedidos. Un INNER JOIN entre estas tablas, basado en el ID_Cliente, te permitiría lograr esto.

Código

```
SELECT
    Clientes.Nombre,
    Pedidos.Fecha
FROM
    Clientes
INNER JOIN
    Pedidos ON Clientes.ID = Pedidos.ID_Cliente;
```

Este código devolverá una tabla con el nombre del cliente y la fecha de su pedido, pero solo para los clientes que tienen pedidos registrados en la tabla Pedidos. Los clientes sin pedidos no aparecerán en el resultado.

¿Qué hace exactamente?

1. Compara filas:

El INNER JOIN compara cada fila de la tabla izquierda con cada fila de la tabla derecha, basándose en la condición definida en la cláusula ON.

2. Identifica coincidencias:

Si los valores de las columnas especificadas en la condición coinciden, las filas se combinan y se incluyen en el resultado.

3. Excluye no coincidencias:
Si no hay coincidencia para una fila en particular, esa fila no se incluye en el resultado final.

LEFT JOIN

El LEFT JOIN (o LEFT OUTER JOIN) devuelve todas las filas de la tabla izquierda y las que coinciden de la derecha.
Si no hay coincidencia en la tabla derecha, mostrará NULL.
Diferencia frente a INNER JOIN:
• El INNER JOIN solo muestra coincidencias.
• El LEFT JOIN muestra todas las filas de la tabla izquierda, aunque no tengan coincidencia.

Sintaxis:
SELECT tabla1.columna, tabla2.columna FROM tabla1 LEFT JOIN tabla2 ON tabla1.columna_comun = tabla2.columna_comun;

Ejemplo:
Supongamos que tenemos dos tablas: Clientes y Pedidos
Código

```
SELECT Clientes.Nombre, Pedidos.Producto
FROM Clientes
LEFT JOIN Pedidos ON Clientes.ClienteID = Pedidos.ClienteID;
```

En este caso, se muestran todos los clientes, incluso el cliente, que no tiene pedidos asociados. Para el cliente, la columna "Producto" aparece como NULL.

RIGHT JOIN

Un RIGHT JOIN en SQL, o RIGHT OUTER JOIN , combina datos de dos o más tablas, dando prioridad a la tabla de la derecha. Devuelve todas las filas de la tabla de la derecha y las filas coincidentes de la tabla de la izquierda. Si no hay coincidencia en la tabla izquierda, se mostrarán valores nulos para las columnas de esa tabla.

Es útil cuando se necesita garantizar que todos los registros de una tabla específica (la tabla de la derecha) estén incluidos en el resultado, incluso si no tienen correspondencia en la otra tabla.

Casos de uso

- **Integración de datos:** Cuando necesitas combinar datos de diferentes fuentes, asegurando que todos los datos de una fuente (la tabla de la derecha) estén presentes en el resultado.
- **Generación de informes:** Para crear informes que incluyan todos los datos de una tabla específica, como mostrar todos los productos y las ventas asociadas, incluso si no hay ventas para algunos productos.
- **Análisis de datos:** Garantizar que todos los registros de una tabla se incluyan en el análisis, incluso si faltan datos relacionados en otra tabla.
- **Escenarios donde la integridad de la tabla de la derecha es crítica:** Si necesitas ver todos los registros de una tabla, independientemente de si tienen o no una coincidencia en la otra tabla, un RIGHT JOIN es una opción.

Sintaxis:

Código

```
SELECT columnas
FROM tabla_izquierda
RIGHT JOIN tabla_derecha
ON tabla_izquierda.columna_comun = tabla_derecha.columna_comun;
```

Ejemplo: Si tienes una tabla Restaurantes y una tabla Calificaciones, un RIGHT JOIN podría mostrar todos los restaurantes, incluso aquellos que no tienen calificaciones, mostrando valores NULL en las columnas de calificación para esos restaurantes.

FULL OUTER JOIN

Un FULL OUTER JOIN en SQL recupera todos los registros de ambas tablas involucradas en la unión, incluyendo aquellos que no tienen coincidencias en la otra tabla. En los casos donde no hay coincidencia, se devuelven valores nulos para las columnas de la tabla que no tiene el registro.

En esencia, un FULL OUTER JOIN combina los resultados de un LEFT OUTER JOIN y un RIGHT OUTER JOIN, proporcionando un conjunto de resultados completo que incluye todos los registros de ambas tablas.

Casos de uso:

- **Análisis de datos:**

Cuando se necesita un conjunto de datos completo de dos tablas, incluyendo aquellos registros que no tienen correspondencia en la otra tabla, un FULL OUTER JOIN es útil para obtener una visión completa de la información.

- **Reconciliación de datos:**

Permite identificar discrepancias entre dos conjuntos de datos, ya que muestra todos los registros, incluso aquellos que no tienen coincidencias.

- **Informes y análisis:**

En escenarios donde se requiere un informe que incluya todos los registros posibles, incluso aquellos que no tienen una relación directa en la otra tabla, un FULL OUTER JOIN es la opción adecuada.

- **Casos donde se requiere conocer todos los registros de ambas tablas:**

Si necesitas saber qué clientes no han realizado pedidos y qué pedidos corresponden a clientes que no están en la base de datos de clientes, un FULL OUTER JOIN te permite obtener esa información.

- **Combinación de datos de diferentes fuentes:**

Cuando se combinan datos de diferentes sistemas o tablas, un FULL OUTER JOIN asegura que no se pierdan datos, incluso si existen desajustes.

Sintaxis:

```
SELECT column_list
FROM table1
FULL OUTER JOIN table2
ON table1.column = table2.column;
```

Ejemplo:

Imagina que tienes dos tablas: empleados y departamentos. Si usas un FULL OUTER JOIN entre ellas, el resultado mostrará:

- Todos los empleados, incluso aquellos que no están asignados a ningún departamento.
- Todos los departamentos, incluso aquellos que no tienen empleados asignados.
- Los empleados que están asignados a departamentos, con la información de ambos.

CROSS JOIN

Un CROSS JOIN en SQL combina cada fila de una tabla con cada fila de otra tabla, produciendo todas las combinaciones posibles, lo que se conoce como un producto cartesiano. En esencia, crea una nueva tabla donde cada registro de la primera tabla se une a cada registro de la segunda.

¿Por qué se le dice producto cartesiano?

El término "producto cartesiano" proviene de las matemáticas y se refiere a la operación que crea un nuevo conjunto a partir de dos conjuntos existentes, donde cada elemento del primer conjunto se combina con cada elemento del segundo conjunto. En SQL, la operación CROSS JOIN realiza exactamente esta misma operación al combinar cada fila de una tabla con cada fila de otra tabla.

Sintaxis

Código

```
SELECT columnas
FROM tabla1
CROSS JOIN tabla2;
```

Por eso:

Si una tabla tiene 5 registros y la otra 3, el resultado serán 15 combinaciones (5×3).

Ejemplo:

SELECT doctor.nombre, medicamento.nombre FROM doctor CROSS JOIN medicamento;

Esto muestra todos los posibles pares entre doctores y medicamentos, sin importar si están relacionados.

En SQL, un JOIN es una operación fundamental que permite combinar registros de dos o más tablas en función de una relación entre ellas, generalmente a través de una columna común. Es esencial para trabajar con bases de datos relacionales, ya que permite obtener información más completa y coherente uniendo datos dispersos en varias tablas.

Los tipos principales de JOIN son:

1. **INNER JOIN**:
   Devuelve solo las filas que tienen coincidencias en ambas tablas. Es el tipo de JOIN más común.
2. **LEFT JOIN (o LEFT OUTER JOIN)**:
   Devuelve todas las filas de la tabla de la izquierda y las que coinciden de la derecha. Si no hay coincidencia, se muestran valores nulos.
3. **RIGHT JOIN (o RIGHT OUTER JOIN)**:
   Es similar al LEFT JOIN, pero devuelve todas las filas de la tabla de la derecha y las coincidentes de la izquierda.
4. **FULL JOIN (o FULL OUTER JOIN)**:
   Devuelve todas las filas de ambas tablas. Si no hay coincidencias, se completa con valores nulos en las columnas que no aplican.
5. **CROSS JOIN**:
   Devuelve el producto cartesiano entre dos tablas, es decir, combina cada fila de la primera tabla con todas las filas de la segunda.

SQL JOIN: Aprenda INTERIOR, IZQUIERDA, DERECHA, COMPLETA y CRUZADA . (s/f). Alura. Recuperado el 18 de julio de 2025, de https://www.aluracursos.com/blog/sql-join-aprenda-inner-left-right-full-e-cross

Datademia. (2024, 16 de octubre). *¿Qué es un JOIN en SQL?* Datademia. https://datademia.es/blog/que-es-un-join-en-sql

du Mortier, G. (2024, 16 de diciembre). *UNIÓN INTERNA vs. UNIÓN EXTERNA: ¿Cuál es la diferencia?* LearnSQL.es; LearnSQL.com. https://learnsql.es/blog/inner-join-vs-outer-join-cual-es-la-diferencia/

Ejemplos de SQL INNER JOIN . (2023, 24 de noviembre). La Escuela del SQL. https://www.laescueladelsql.com/sql-inner-join-ejemplos/

o365devx. (s/f). *Realizar combinaciones mediante Access SQL* . Microsoft.com. Recuperado el 18 de julio de 2025, de https://learn.microsoft.com/es-es/office/vba/access/concepts/structured-query-language/perform-joins-using-access-sql

Sławińska, M. (28 de noviembre de 2023). *Explicación de SQL JOIN: 5 ejemplos claros de INNER JOIN SQL para principiantes* . LearnSQL.es; LearnSQL.com. https://learnsql.es/blog/explicacion-de-sql-joins-5-ejemplos-claros-de-inner-join-sql-para-principiantes/

Tipos de unión SQL: Elección entre unión derecha e izquierda . (19 de julio de 2024). TiDB; PingCAP. https://www.pingcap.com/article/sql-join-types-choosing-between-right-and-left-join/

Unión externa completa de SQL . (s/f). Hightouch.com. Recuperado el 18 de julio de 2025, de https://hightouch.com/sql-dictionary/sql-full-outer-join

Wdzięczna, D. (2022, 5 de diciembre). *Guía ilustrada del CROSS JOIN de SQL* . LearnSQL.es; LearnSQL.com. https://learnsql.es/blog/guia-ilustrada-del-cross-join-de-sql/