Consultas *Join* y variantes

Francisco Martín Seguí Camejo.

Curso de Tecnólogo en Análisis y Desarrollo de Sistemas – Instituto Federal de Educación, Ciencia y Tecnología Sul-Rio-Grandense (IFSUL) – Campus Sant'Ana do Livramento.

francisco.segui@estudiantes.utec.edu.uy

Abstract. *This article describes the Join from relational algebra and three of its variations. The concepts of each operation and the differences between them are explained followed by its implementation using SQL language.*

1. Introducción:

Una operación *Join* combina dos o más tablas generando un conjunto resultante de la información almacenada en tales relaciones. Para lograr una operación J*oin* las tablas deben tener columnas relacionadas y/o *foreign keys* en común, las cuales son usadas para vincularlas entre sí.
En el presente artículo se comienza describiendo la operación denominada *Inner Join* para luego explicar *left join* y *right join*, explayándose en sus características y presentando un breve ejemplo en cada caso.

2. Desarrollo:

3.1. Inner Join

Comando denominado *Inner Join* o solamente *Join.*
La operación *Inner Join* devuelve como resultado todas las filas en común de las relaciones. Cada vez que la condición es cumplida, la fila será incluida en el conjunto resultante. Debemos recordar que las columnas que usamos para vincular ambas tablas que tengan valores *Null* no serán incluidas en la relación resultante, ya que ellas no combinan con ningún valor. *Null* no es igual a nada, *Null* no es igual a *Null*.
Las columnas usadas para la condición *Join* no necesariamente deben ser del mismo tipo de datos, aunque si deben ser compatibles, entendiéndose por ello que si son de dos tipos de datos diferentes, tienen que ser pasibles de ser convertidos al tipo de dato de la otra columna. En principio el Servidor *SQL* intenta llevar a cabo una conversión implícita, aunque si eso falla pueden ser usados comandos como *Cast* o *Convert*.
Una operación *Join* puede hacerse en más de dos tablas para generar un resultado. Para hacerlo debemos especificar dos tablas primero y luego las restantes una por una.

Pongamos como ejemplo lo siguiente: Tenemos cuatro relaciones: Empleado, Empleadoterritorios, Región y Territorio. Si queremos saber todas las regiones asociadas a empleados, primero debemos recuperar el territorio de cada empleado para luego recuperar la región de cada territorio.
Un *Join* que involucra más de tres tablas funciona de la siguiente manera:
Un conjunto de resultados es generado uniendo las primeras dos tablas, luego él es unido a la tercera tabla y así en sucesivamente.
El comando para realizarlo es el siguiente:

```
SELECT * FROM Empleados as E JOIN Empleadosterritorios as ET
ON E.empleadoid = ET.empleadoid
JOIN Territorios T
ON ET.territorioid = T.territorioid
JOIN Region R
ON T.regionid = R.regionid
```

Una operación *Join* también puede ser usada en una declaración *Update* o en un *Delete*. Esto es útil cuando queremos actualizar o borrar filas, y necesitamos información que está guardada en varias tablas. Supongamos que queremos actualizar el precio de un producto suministrado por determinado proveedor, pero el nombre del proveedor está guardado en la tabla Proveedores. Para ellos necesitaríamos hacer un *Join* entre las tablas Productos y Proveedores.

3.1. Outer Joins:

Dentro de *Outer Joins* se ubican: *Left Join* y *Right Join*.
Las operaciones *Outer Join* retornan todas las filas que coinciden con la condición del *Join* y además algunas más, que no coinciden.
En estas consultas el orden de las tablas importa, a diferencia del *Inner Join* donde no es significativa.
Por lo que *Outer Join* puede ser definido como la unión de un *Inner Join* y las filas sin par , o sea que no están vinculadas a otras por su id o los atributos indicados para hacerlas coincidir.

3.2. Right Join:

Retorna las filas que se corresponden entre sí en ambas tablas, igual al *Inner Join* , y además retorna filas de la tabla de la derecha que no tienen una correspondiente en la tabla opuesta.
Todas las filas que no tienen su correspondiente (*match*) contienen un valor *Null* en todas las columnas de la tabla izquierda.
A continuación un ejemplo:

Tenemos dos Relaciones: Customers y City. Queremos llevar a cabo una operación *Right Join*, siendo Customers la tabla de la izquierda y City la derecha.
El atributo que usaremos para crear un *"match"* entre ellas es Cityid.
El comando SQL que debemos ejecutar es el siguente:

SELECT * from Customer RIGHT JOIN City ON Customer.Cityid = City.Cityid

Así todos las filas de Customer serán exhibidas y de la tabla City solo las que coincidan mediante Cityid.

3.3. Left Join:
Es opuesta a *Right Join.*
Las filas que coinciden con la condición del *Join* son retornadas, en conjunto con las filas de la tabla izquierda que no tienen correspondencia con filas de la tabla derecha.
Las filas que no tienen *"match"* , o que no tienen coincidencia, de la tabla derecha, pasan a tener el valor *Null.*
Una *left join* puede ser transformado en una *right join* si el orden de sus tablas es cambiado o invertido. La tabla derecha pasa a ser la izquierda y vice versa. Esto hace que el orden de las tablas pase a cobrar vital importancia. Se podría decir en puridad que nos encontramos frente a la misma operación.
El comando *SQL Left Outer Join y left Join* son intercambiables, por lo que ambos pueden ser utlizados.

Ejemplo comando SQL:
Tablas Customer y City.
Queremos llevar a cabo una operación left Join, siendo Customers la tabla de la izquierda y City la derecha. El atributo que usaremos para crear un *"match"* entre ellas es Cityid.

SELECT * from Customer LEFT JOIN City ON Customer.Cityid = City.Cityid

Conclusión

Debe señalarse que las operaciones *Join* en *SQL* son de vital importancia en un mundo donde los datos abundan en cantidad y están dispersos, ya que permiten la creación de nuevas tablas combinando los datos de diversas fuentes y así obteniendo información trascendental para los individuos, empresas y hasta entidades políticas.
Así contribuyendo positivamente al trabajo del programador como también a las entidades que necesitan optimizar su tiempo y productividad.

Referencias

Rojas Carlos, Guerrero Fernando, Microsoft SQL Server 2000 Programming by Example, United States, 2001.

W3Schools Disponible: https://www.w3schools.com/sql/sql_join.asp. Acceso octubre/2020.