30-1-2026

Reporte de lectura - UNIONES SQL

Ezequiel Gutiérrez Torres
TUXTLA GUTIÉRREZ, CHIAPAS

1. Definición de JOIN en SQL

Una unión (JOIN) en SQL es una operación que permite combinar filas de dos o más tablas basándose en una relación común entre ellas, generalmente mediante columnas relacionadas. Esta operación es fundamental en bases de datos relacionales, ya que permite extraer información significativa de múltiples tablas en una sola consulta.

Las uniones resuelven el problema de la normalización de datos, permitiendo:

- Relacionar información almacenada en diferentes tablas
- Evitar la duplicación de datos
- Crear vistas consolidadas de información
- Optimizar el almacenamiento y mantenimiento de la base de datos

2. Tipos de Uniones (JOIN)

SQL proporciona varios tipos de uniones, cada una con un propósito específico según los datos que se deseen obtener:

2.1. INNER JOIN

Funcionamiento

INNER JOIN devuelve únicamente las filas que tienen coincidencias en ambas tablas. Es el tipo de unión más común y restrictivo, ya que excluye cualquier fila que no tenga correspondencia en la otra tabla.

Sintaxis General

```
SELECT columnas
FROM tabla1
INNER JOIN tabla2
ON tabla1.columna = tabla2.columna;
```

Ejemplo Práctico

Consideremos dos tablas: Estudiantes **y** Cursos

Tabla: Estudiantes

id_estudiante	nombre	id_curso
1	Ana García	101
2	Luis Pérez	102

3	María López	NULL

Tabla: Cursos

id_curso	nombre_curso
101	Bases de Datos
102	Programación Web
103	Redes de Computadoras

Consulta:

```
SELECT Estudiantes.nombre, Cursos.nombre_curso
FROM Estudiantes
INNER JOIN Cursos
ON Estudiantes.id_curso = Cursos.id_curso;
```

Resultado:

nombre	nombre_curso
Ana García	Bases de Datos
Luis Pérez	Programación Web

Nota: *María López no aparece en el resultado porque su id_curso es NULL y no tiene coincidencia en la tabla Cursos. El curso "Redes de Computadoras" tampoco aparece porque ningún estudiante está inscrito en él.*

2.2. LEFT JOIN (LEFT OUTER JOIN)

Funcionamiento

LEFT JOIN devuelve todas las filas de la tabla de la izquierda (primera tabla mencionada), junto con las filas coincidentes de la tabla de la derecha. Si no existe coincidencia, los valores de la tabla derecha serán NULL. Este tipo de unión garantiza que no se pierda información de la tabla principal.

Sintaxis General

```
SELECT columnas
FROM tabla1
LEFT JOIN tabla2
ON tabla1.columna = tabla2.columna;
```

Ejemplo Práctico

Usando las mismas tablas anteriores:

Consulta:

```
SELECT Estudiantes.nombre, Cursos.nombre_curso
FROM Estudiantes
LEFT JOIN Cursos
ON Estudiantes.id_curso = Cursos.id_curso;
```

Resultado:

nombre	nombre_curso
Ana García	Bases de Datos
Luis Pérez	Programación Web
María López	NULL

Nota: *Ahora María López aparece en el resultado aunque no tenga un curso asignado. El valor de nombre_curso es NULL para ella, indicando que no hay coincidencia en la tabla Cursos.*

2.3. RIGHT JOIN (RIGHT OUTER JOIN)

Funcionamiento

RIGHT JOIN es el complemento de LEFT JOIN. Devuelve todas las filas de la tabla de la derecha (segunda tabla mencionada), junto con las filas coincidentes de la tabla izquierda. Si no existe coincidencia, los valores de la tabla izquierda serán NULL.

Sintaxis General

```
SELECT columnas
FROM tabla1
RIGHT JOIN tabla2
ON tabla1.columna = tabla2.columna;
```

Ejemplo Práctico

Usando las mismas tablas:

Consulta:

```
SELECT Estudiantes.nombre, Cursos.nombre_curso
FROM Estudiantes
RIGHT JOIN Cursos
ON Estudiantes.id_curso = Cursos.id_curso;
```

Resultado:

nombre	nombre_curso
Ana García	Bases de Datos
Luis Pérez	Programación Web
NULL	Redes de Computadoras

Nota: *Ahora aparecen todos los cursos, incluyendo "Redes de Computadoras" que no tiene estudiantes inscritos. El valor de nombre es NULL para este curso, indicando que no hay estudiantes relacionados.*

2.4. FULL JOIN (FULL OUTER JOIN)

Funcionamiento

FULL JOIN combina los resultados de LEFT JOIN y RIGHT JOIN. Devuelve todas las filas de ambas tablas, mostrando NULL en las columnas donde no hay coincidencia. Es útil cuando se necesita ver toda la información disponible de ambas tablas, independientemente de si existe o no una relación.

Sintaxis General

```
SELECT columnas
FROM tabla1
FULL JOIN tabla2
ON tabla1.columna = tabla2.columna;
```

Ejemplo Práctico

Usando las mismas tablas:

Consulta:

```
SELECT Estudiantes.nombre, Cursos.nombre_curso
FROM Estudiantes
FULL JOIN Cursos
ON Estudiantes.id_curso = Cursos.id_curso;
```

Resultado:

nombre	nombre_curso
Ana García	Bases de Datos
Luis Pérez	Programación Web
María López	NULL
NULL	Redes de Computadoras

Nota: *El resultado incluye todos los estudiantes y todos los cursos. María López aparece sin curso asignado (NULL en nombre_curso), y "Redes de Computadoras" aparece sin estudiantes (NULL en nombre). Es la unión más completa de todas.*

2.5. CROSS JOIN

Funcionamiento

CROSS JOIN realiza un producto cartesiano entre dos tablas, combinando cada fila de la primera tabla con todas las filas de la segunda tabla. No requiere una condición de unión. El resultado tiene un número de filas igual al producto del número de filas de ambas tablas. Se utiliza raramente en la práctica, principalmente para generar combinaciones exhaustivas.

Sintaxis General

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

Ejemplo Práctico

Para este ejemplo, usaremos tablas más pequeñas para demostrar el concepto:

Tabla: Colores

color
Rojo
Azul

Tabla: Tamaños

tamaño
Pequeño
Mediano
Grande

Consulta:

```
SELECT Colores.color, Tamaños.tamaño
FROM Colores
CROSS JOIN Tamaños;
```

Resultado:

color	tamaño
Rojo	Pequeño
Rojo	Mediano

Rojo	Grande
Azul	Pequeño
Azul	Mediano
Azul	Grande

Nota: *El resultado contiene 6 filas (2 colores × 3 tamaños = 6 combinaciones). Cada color se combina con cada tamaño posible. CROSS JOIN es útil para generar catálogos de productos, combinaciones de opciones, o escenarios de prueba.*

3. Conclusiones

Las uniones SQL son herramientas fundamentales para trabajar con bases de datos relacionales. La elección del tipo de unión adecuado depende de los requisitos específicos de la consulta:

INNER JOIN: Se utiliza cuando solo se necesitan los registros que tienen coincidencias en ambas tablas. Es la opción más restrictiva y comúnmente usada.

LEFT JOIN: Útil cuando se necesita conservar todos los registros de la tabla principal, incluso si no tienen correspondencia en la segunda tabla.

RIGHT JOIN: Similar a LEFT JOIN pero conservando todos los registros de la segunda tabla. Menos común en la práctica.

FULL JOIN: Proporciona la vista más completa al incluir todos los registros de ambas tablas, mostrando NULL donde no hay coincidencias.

CROSS JOIN: Genera todas las combinaciones posibles entre las filas de dos tablas. Se usa principalmente para casos especiales como generación de combinaciones.

Comprender estos tipos de uniones permite diseñar consultas más eficientes y precisas, optimizando el rendimiento de las bases de datos y facilitando la obtención de información relevante para el análisis y toma de decisiones.

4. Referencias

• SQL JOIN Documentation - W3Schools

• Database System Concepts - Silberschatz, Korth, Sudarshan

• MySQL Official Documentation - JOIN Syntax

• PostgreSQL Documentation - Queries