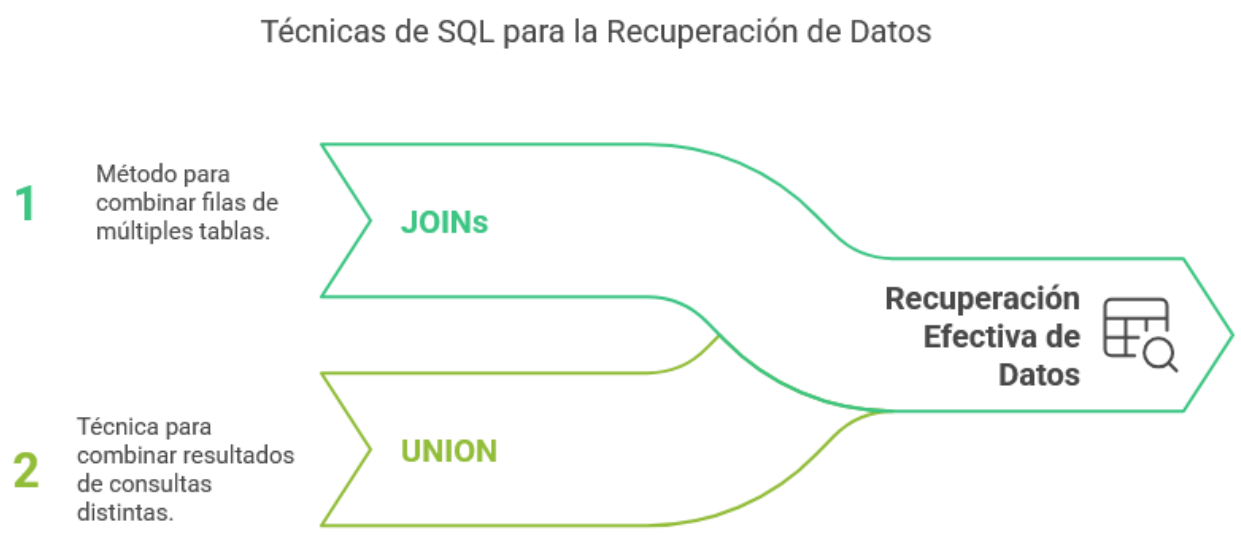
Relacionando Datos desde Múltiples Tablas en SQL

Propósito: Este documento presenta un resumen de los principales temas e ideas discutidos en el documento fuente, centrándose en las técnicas para recuperar información relacionando dos o más tablas en SQL, específicamente a través del uso de JOINs y el operador UNION.



Temas Principales

1. **Recuperación de Datos Relacionados mediante JOINS:**

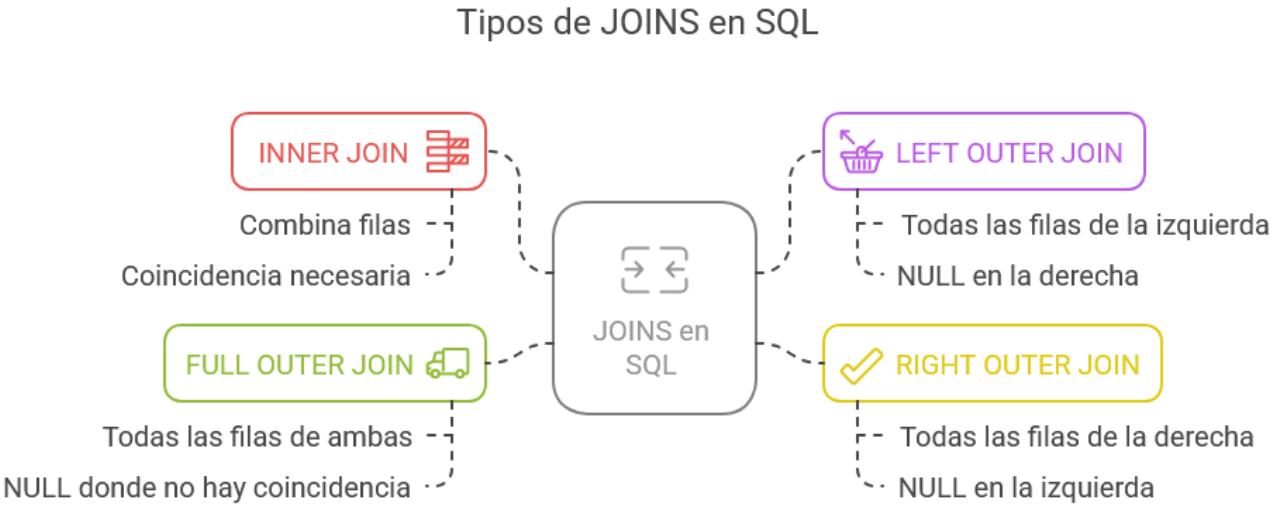
- El documento introduce el concepto fundamental de utilizar JOINs para combinar datos de dos o más tablas basándose en relaciones lógicas definidas entre ellas.
- Se destaca que las condiciones de combinación pueden especificarse tanto en la cláusula FROM como en la cláusula WHERE, aunque se recomienda la cláusula FROM para una mejor organización y separación de las condiciones.
- Se subraya la importancia de la **no ambigüedad** al referenciar columnas cuando múltiples tablas están involucradas en una consulta, requiriendo la calificación de los nombres de columna duplicados con el nombre de la tabla.
- La lista de selección en un JOIN puede incluir cualquier subconjunto de las columnas de las tablas combinadas, e incluso una tabla puede actuar como "puente" sin que sus columnas se incluyan en el resultado final.
- Aunque las condiciones de JOIN comúnmente utilizan el operador de igualdad (=), se pueden emplear otros operadores relacionales y predicados.
- El motor de consultas de SQL Server optimiza el proceso de JOIN eligiendo el método más eficiente.

- Se clasifican las combinaciones en INNER JOIN y OUTER JOIN, diferenciando su comportamiento en relación con las filas coincidentes y no coincidentes entre las tablas.

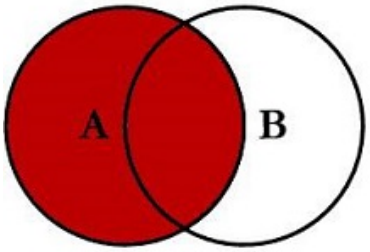
1. **Tipos Específicos de JOINS:**

- **INNER JOIN:**Combina filas de dos tablas **solo cuando existe una coincidencia** de valores en un campo común especificado en la condición ON.
- Sintaxis: SELECT campos FROM tb1 INNER JOIN tb2 ON tb1.campo1 comp tb2.campo2
- Se considera el tipo de combinación **predeterminado** si no se especifica explícitamente.
- El ejemplo proporcionado ilustra cómo un INNER JOIN entre las tablas "Personas" y "Puestos" devuelve solo las personas cuyo puesto existe en la tabla "Puestos".
- **Cita:** "Las vinculaciones entre tablas se realiza mediante la cláusula INNER que combina registros de dos tablas siempre que haya concordancia de valores en un campo común."
- **Cita:** "Especifica que se devuelvan todos los pares de filas coincidentes. Rechaza las filas no coincidentes de las dos tablas. Si no se especifica ningún tipo de combinación, éste es el valor predeterminado."
- **OUTER JOINS (LEFT, RIGHT, FULL):**Se utilizan para incluir todas las filas de al menos una de las tablas involucradas en la combinación, incluso si no hay una coincidencia en la otra tabla.
- **LEFT OUTER JOIN:**Retorna **todas las filas de la tabla de la izquierda** especificada en el FROM, y las filas coincidentes de la tabla de la derecha. Si no hay coincidencia, se insertan valores NULL para las columnas de la tabla de la derecha.
- Sintaxis: SELECT campos FROM tb1 LEFT [OUTER] JOIN tb2 ON tb1.campo1 comp tb2.campo2

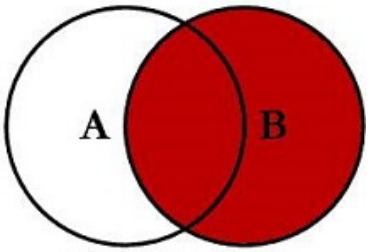
- El ejemplo muestra cómo un LEFT OUTER JOIN entre "Personas" y "Puestos" incluye a todas las personas, mostrando su puesto si existe en la tabla "Puestos" o NULL si no.
- **Cita:** "LEFT [OUTER] JOIN toma todos los registros de la tabla de la izquierda aunque no tengan ningún registro en la tabla de la derecha."
- **RIGHT OUTER JOIN:**Realiza la operación inversa al LEFT OUTER JOIN. Retorna **todas las filas de la tabla de la derecha**, y las filas coincidentes de la tabla de la izquierda. Los valores NULL se insertan para las columnas de la tabla de la izquierda si no hay coincidencia.
- Sintaxis: SELECT campos FROM tb1 RIGHT [OUTER] JOIN tb2 ON tb1.campo1 comp tb2.campo2
- El ejemplo ilustra cómo un RIGHT OUTER JOIN entre "Personas" y "Puestos" muestra todos los puestos, y la persona asociada si existe en la tabla "Personas" o NULL si no.
- **Cita:** "RIGHT [OUTER] realiza la misma operación pero al contrario, toma todos los registros de la tabla de la derecha..."
- **FULL OUTER JOIN:**Combina los resultados de un LEFT OUTER JOIN y un RIGHT OUTER JOIN. Retorna **todas las filas de ambas tablas**. Si no hay coincidencia entre las filas, se insertan valores NULL para las columnas de la tabla que no tiene la fila coincidente.
- Sintaxis: FULL [OUTER] JOIN tb2 ON tb1.campo1 comp tb2.campo2
- El ejemplo muestra cómo un FULL OUTER JOIN entre "Personas" y "Puestos" incluye todas las personas y todos los puestos, con valores NULL donde no hay correspondencia.
- **Cita:** "FULL [OUTER] Es una combinación de LEFT y RIGHT JOIN. En este caso devuelve todos los registros de ambas tablas y en los casos donde no puede hacer el JOIN devuelve valores NULL."



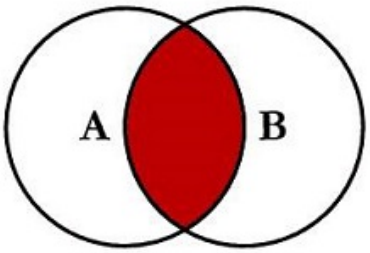
SQL JOINS



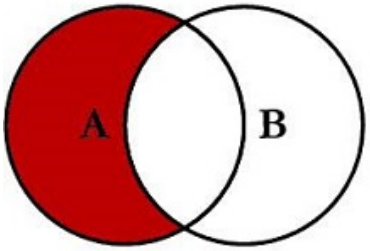

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



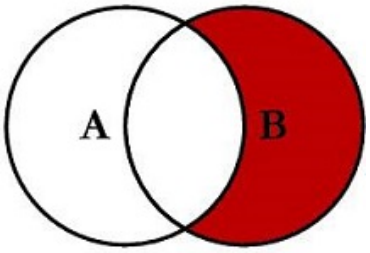

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



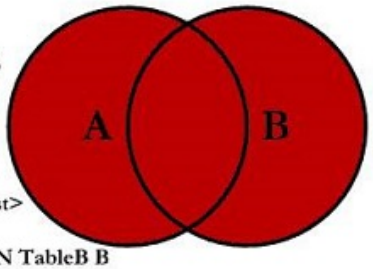

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



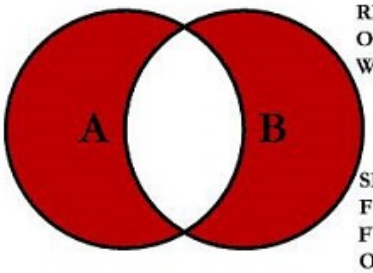

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

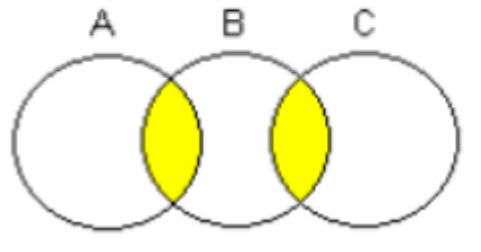
1. **JOINS Sin Restricciones (Producto Cartesiano):**

- Se menciona el uso de JOINs sin una cláusula ON (o condiciones en el WHERE que las restrinjan), lo que resulta en un **producto cartesiano** entre las tablas involucradas.
- Cada fila de la primera tabla se combina con cada fila de la segunda tabla.
- El ejemplo con 5 registros en "Personas" y 5 en "Puestos" resulta en 25 filas (5 x 5).
- **Cita:** "Se utiliza en los casos que se quiere hacer el producto cartesiano entre dos tablas."
- **Cita:** "Especifica el producto resultante de dos tablas. Devuelve las mismas filas que se devolverían si no se especificara la cláusula WHERE."

1. **JOINS con Más de Dos Tablas:**

- SQL Server permite combinar más de dos tablas en una sola consulta.
- Es crucial incluir una cláusula ON (o una condición equivalente en el WHERE) **por cada tabla adicional** que se une.
- El ejemplo con las tablas "Personas", "Puestos" y "Área" ilustra cómo se pueden encadenar múltiples INNER JOINs para obtener información relacionada entre las tres tablas.
- **Cita:** "Se debe tener en cuenta que por cada tabla que intervenga en el JOIN debe existir su clausula en el WHERE o en el ON."

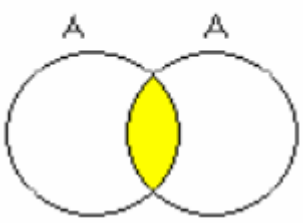
JOINS con mas de dos tablas



1. **SELF JOINS:**

- Un SELF JOIN implica **unir una tabla consigo misma**.
- Es útil para consultar datos jerárquicos o para comparar filas dentro de la misma tabla.
- Se requiere el uso de **alias de tabla** para distinguir entre las diferentes "instancias" de la misma tabla que se están utilizando en la combinación.
- El ejemplo muestra cómo un SELF JOIN en la tabla "Personas" (con alias pe1 y pe2) puede utilizarse para obtener el nombre de cada empleado y el nombre de su supervisor, asumiendo que la tabla tiene una columna que referencia al supervisor (por ejemplo, supervisor) que también es un codigo de la misma tabla "Personas".
- **Cita:** "El SEFT JOIN correlaciona diferentes registros que se encuentran en la misma tabla."
- **Cita:** "Cuando se utiliza más de una vez la misma tabla para construir un JOIN es necesario utilizar ALIAS para poder identificarlas."
- **Cita:** "Esto es puramente conceptual ya que lo que se esta aplicando es un INNER JOIN entre la misma tabla “personas”."

SELF JOINS



1. **Unión de Resultados con el Operador UNION:**

- El operador UNION se utiliza para **combinar los conjuntos de resultados de dos o más consultas SELECT en un único conjunto de resultados**.
- Es diferente de los JOINs, ya que UNION **apila las filas** en lugar de combinar columnas de diferentes tablas.
- **Requisitos para usar UNION:**Las consultas deben tener el **mismo número de columnas**.
- Las columnas correspondientes deben tener **tipos de datos compatibles**.
- Las columnas correspondientes deben estar en el **mismo orden**.
- Sintaxis: consulta1 UNION [ALL] consulta2
- Por defecto, UNION **elimina las filas duplicadas** del conjunto de resultados final.
- Si se utiliza UNION ALL, **todas las filas** de las consultas combinadas se incluyen en el resultado, incluso las duplicadas.
- **Cita:** "Combina los resultados de dos o más consultas en un solo conjunto de resultados que incluye todas las filas que pertenecen a las consultas de la unión."
- **Cita:** "La operación UNION es distinta de la utilización de combinaciones de columnas de dos tablas."
- **Cita:** "Por defecto UNION remueve las filas duplicadas"

1. **Otros Elementos Mencionados:**

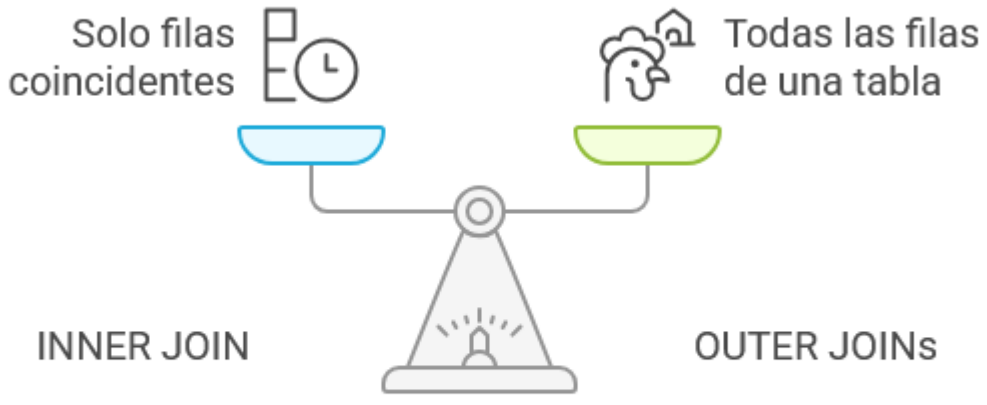
- Se menciona brevemente la cláusula TOP, aunque no se proporciona detalle sobre su funcionalidad.
- Se incluyen secciones para "Comentario sobre Contenido de la Clase" y "Guardar Comentario", lo que sugiere que este documento forma parte de un material de aprendizaje más amplio.

Ideas o Hechos Importantes

- Los JOINs son esenciales para consultar datos relacionados almacenados en múltiples tablas.
- La elección del tipo de JOIN (INNER, LEFT OUTER, RIGHT OUTER, FULL OUTER) determina qué filas se incluyen en el resultado en función de la existencia o no de coincidencias entre las tablas.
- Es fundamental comprender la diferencia entre INNER JOIN (solo filas coincidentes) y los OUTER JOINs (incluyen todas las filas de al menos una tabla).
- El uso de alias de tabla es necesario cuando se trabaja con SELF JOINs o cuando se referencian columnas con nombres idénticos en múltiples tablas.
- El operador UNION permite combinar los resultados de diferentes consultas, apilando las filas, y requiere consistencia en el número, orden y tipo de datos de las columnas.
- UNION elimina duplicados por defecto, mientras que UNION ALL los conserva.

Conclusiones

Este documento proporciona una introducción clara y concisa a las técnicas fundamentales para relacionar datos en SQL utilizando JOINs y el operador UNION. Se explican los diferentes tipos de JOINs con ejemplos prácticos, destacando su comportamiento en diversas situaciones de relación entre tablas. Además, se introduce el concepto de SELF JOIN para trabajar con datos dentro de la misma tabla y el operador UNION para combinar resultados de múltiples consultas. La comprensión de estos conceptos es crucial para la construcción de consultas SQL eficientes y efectivas que involucren datos distribuidos en múltiples tablas.



Elige el tipo de JOIN adecuado para tus necesidades de datos.