CONSULTAS AVANZADAS Y VISTAS

Diseño y Programación de una BBDD

ÍNDICE

CONSULTAS AVANZADAS Y VISTAS

1. **Explica para qué se utiliza la cláusula JOIN, e indica cuándo es necesario utilizarla.**

   La Cláusula **JOIN** se usa para combinar datos de 2 o más tablas relacionadas en una consulta. Es necesario utilizarla cuando queremos recuperar información de múltiples tablas relacionadas con una sola consulta y con ella unir los datos mediante una "condición" especifica.

   Ejemplo:

```
SELECT E.EMPLOYEE_ID,
       E.LAST_NAME,
       E.DEPARTMENT_ID,
       D.DEPARTMENT_ID,
       D.LOCATION_ID
  FROM EMPLOYEES E
       INNER JOIN DEPARTMENTS D
               ON D.DEPARTMENT_ID = E.DEPARTMENT_ID
 WHERE E.EMPLOYEE_ID = 200;
```

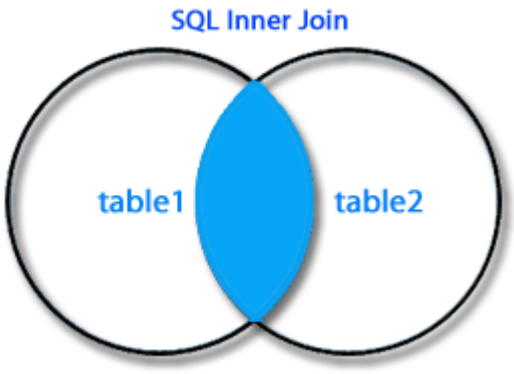
En este ejemplo combina 2 tablas, "employees" y la tabla "department" para recuperar información específica sobre el empleado cuya "employee_id" sea igual al valor 200.

2. **¿A qué se llama combinación externa? ¿Cuántos tipos de JOINs hay en MySQL? Agrega una ilustración a la respuesta para aclarar los conceptos.**

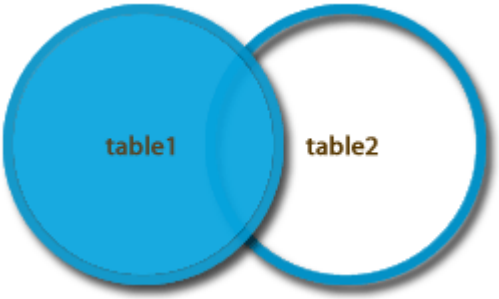
   En SQL una combinación externa es un "tipo" de **JOIN** el cual incluye registros que no tienen correspondencia en ambas tablas de la unión.

   En SQL hay 3 tipos de JOIN (Interno, Externo y Cruzado) a su vez se dividen en 5º tipos de JOIN en SQL según el estándar ANSI:

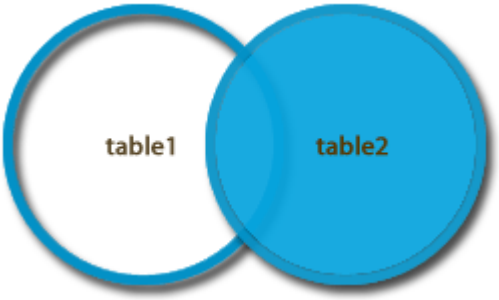
- **INNER JOIN (JOIN Interno):** Devuelve solo los registros los cuales coincidan en las tablas consultadas.



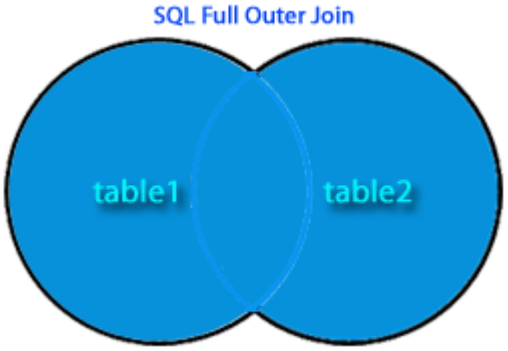
- **LEFT JOIN (JOIN Externo):** Devuelve todos los registros de la tabla de la izquierda (**1º tabla que se menciona en la consulta**), independientemente si existe un registro correspondiente en la tabla de la derecha.



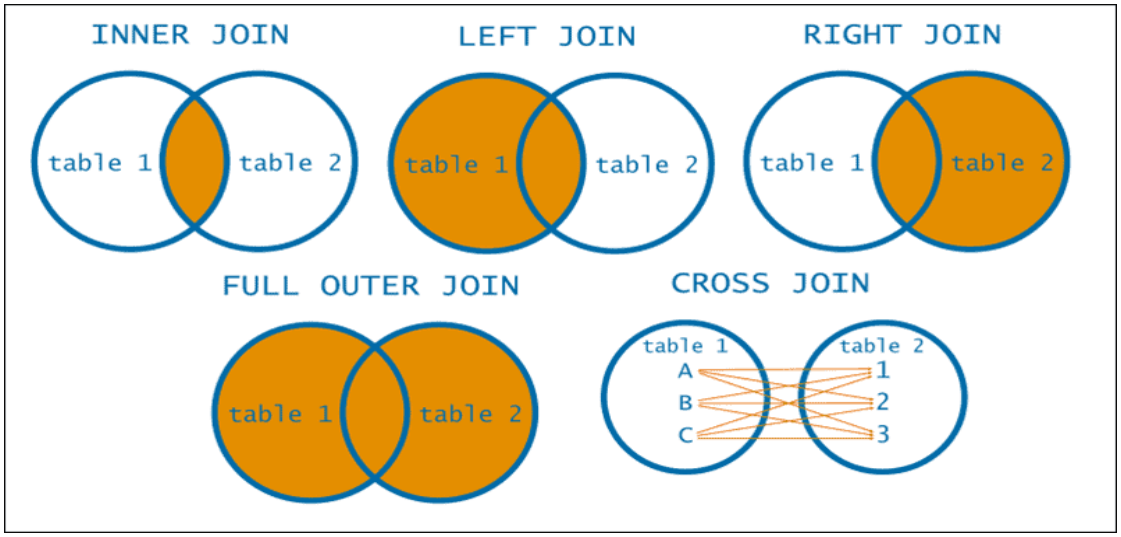
- **RIGHT JOIN (JOIN Externo):** Devuelve todos los registros de la tabla de la derecha (**2º tabla que se menciona en la consulta**), independientemente si existe un registro correspondiente en la tabla de la izquierda.



- **FULL JOIN (JOIN Externo):** Devuelve el resultado de la tabla de la izquierda y tabla derecha, aunque no tengan correspondencia en la otra tabla. La tabla combinada contendrá todos los registros de ambas tablas y presentará valores **NULLS** para registros sin pareja.



- **CROSS JOIN (JOIN Cruzado):** Devuelve todas las líneas de las tablas por cruzamiento, es decir, para cada línea de la tabla izquierda queremos todas las líneas de la tabla de la derecha y viceversa.



3. ¿Cómo se combinan más de dos tablas?

Para combinar más de dos tablas se puede encadenar múltiples cláusulas “JOIN” cada una especificando como se relacionan las tablas adicionales. El proceso es esencialmente el mismo que unir dos tablas, pero repites la cláusula “JOIN” para cada nueva tabla que quieras incluir en la consulta.

```
SELECT
    A.column1,
    B.column2,
    C.column3
FROM
    TableA A
INNER JOIN TableB B ON A.common_column = B.common_column
INNER JOIN TableC C ON B.another_common_column = C.another_common_column;
```

4. ¿La condición JOIN debe utilizar siempre el operador igual? ¿Se puede utilizar otro operador de comparación? ¿Cómo se hace?

En la condición JOIN no siempre se utiliza el operador igual "=", también se utilizan los operadores mayor ">", menor "<", mayor igual ">=", menor igual "<=", desigualdad "<>" y diferente "!=".

En el ejemplo utilizamos mayor que con INNER JOIN para la condición.

```
SELECT *

FROM empleados AS e

INNER JOIN departamentos AS d ON e.salario > d.presupuesto_maximo;
```

Hay que indicar que, aunque la sintaxis te permite usar diferentes operadores, los JOIN que no utilizan el operador igual no siempre son eficientes en términos de rendimiento, porque es posible que no puedan aprovechar los índices de la misma manera que los JOIN basados en igualdad.

5. ¿Qué es una subconsulta?

Las subconsultas son consultadas anidadas dentro de una consulta. Se pueden utilizar en diversas partes de la consulta principal, como en la cláusula 'Select', 'From', 'Where' o 'Having'. Por lo general retorna un conjunto de resultados que la consulta principal puede utilizar para realizar una operación adicional, como un filtrado, seleccionar o calcular.

6. ¿Dónde se incluyen las subconsultas dentro de las cláusulas SQL? Indica un ejemplo.

Las subconsultas se puedes incluir como:

- **Subconsultas escalares**: estas retornan un único valor y se utilizan en lugares donde se espera un solo valor, como en la cláusula 'SELECT' o en la clausula 'WHERE'. Por ejemplo:

```
SELECT nombre, salario

FROM mecánicos

WHERE salario > ( SELECT AVG(salario) FROM mecanicos);
```

- **Subconsultas de Correlación:** hacen referencia a las columnas de la consulta externa y se ejecutan repetidamente, una vez por cada fila procesada por la consulta externa. Por ejemplo:

  SELECT m1.nombre, m1.salario

  FROM mecanicos m1

  WHERE m1.salario > (SELECT AVG(salario) FROM mecanicos m2 WHERE m1.departamento_id= m2.departamento_id);

- **Subconsultas de Tabla o también llamada Subconsultas de FROM:** devuelven una o más columnas y posiblemente varias filas, y se utilizan en la cláusula 'FROM' para definir una tabla temporal sobre la cual se ejecuta la consulta externa. Por ejemplo:

  SELECT tmp.nombre, tmp.salario

  FROM (SELECT nombre, salario FROM mecanicos WHERE salria > 1000) AS tmp;

  Las subconsultas es una herramienta muy interesante que nos ayuda a crear consultas complejas y realizar operaciones que de otro modo requerirían múltiples consultas SQL.

7. ¿Qué es una consulta de UNION y para qué sirve? Indica un ejemplo.

Las consultas que incluyen el operador UNION se utilizan para combinar los resultados de dos o más consultas en un único conjunto de resultados. Cada consulta de "SELECT" dentro del "UNION" debe de tener el mismo número de columnas en el resultado, con tipos de datos compatibles y en el mismo orden. Con la peculiaridad de que las columnas en cada consulta no necesitan ser los mismos. Ejemplo:

SELECT orden_trabajo_id, 'Servicio' AS Tipo, descripcion, precio_servicio AS Precio

FROM servicios

WHERE fecha = '2023-07-15'

UNION

SELECT orden_trabajo_id, 'Repuesto', descripcion, Precio_repuesto

FROM repuestos

WHERE fecha = '2023-07-15';

En la **primera consulta** consultamos los servicios realizados en la fecha indicada.

En la **segunda consulta** los respuestos utilizados en la misma fecha.

8. ¿Qué requisitos se deben cumplir para poder crear una consulta de UNION?

Los requisitos para utilizar UNION son:

- **Numero de columnas:** cada consulta “SELECT” dentro del UNION debe tener el mismo número de columnas en su conjunto de resultados.
- **Tipos de datos compatibles:** las columnas correspondientes en cada consulta “SELECT” deben tener tipo de datos que sean compatibles entre sí.
- **Orden de columnas:** el orden de las columnas debe ser el mismo en todas las consultas.
- **Ordenamiento:** si se utiliza ORDER BY en los resultados combinados, solo puedes especificar la cláusula al final de la última consulta.
- **Filtrado:** del mismo modo, si quieres limitar los resultados deberes aplicar la cláusula LIMIT.

9. ¿Qué diferencia existe entre las cláusulas UNION vs. UNION ALL?

DIFERENCIAS	
CLÁUSULA UNION	**CLÁUSULA UNION ALL**
Elimina automáticamente las filas duplicadas.	Incluye todas las filas, incluyendo las duplicadas.
Muestra solo una instancia de una fila si existe una duplicada en los resultados combinados.	Muestra todas las instancias de las filas, incluso si existen duplicados en los resultados combinados.
Realiza un proceso de eliminación de duplicados para garantizar que no haya filas repetidas en el resultado final.	No realiza ningún proceso de eliminación de duplicados.

10. ¿Qué diferencia existe entre las cláusulas UNION vs. JOIN?

DIFERENCIAS	
CLÁUSULAS UNION	**CLÁUSULAS JOIN**
Se usa para combinar resultados de 2 o más consultas en un solo conjunto de resultados.	Se usa para combinar filas de diferentes tablas en función de una condición de igualdad entre columnas.
Es útil cuando deseamos juntas datos de diferentes fuentes que tienen una estructura similar, como lista de nombres o valores.	Es útil cuando deseamos relacionar y combinar datos de tablas distintas, como unir información de empleados con sus respectivos departamentos.
Elimina automáticamente las filas duplicadas en el resultado final a no ser que usemos “UNION ALL”	Podemos realizar diferentes tipos de combinaciones (Internas, externas y cruzadas), según nuestras necesidades.

11. ¿Qué es una vista?

Una vista es una representación virtual de una o más tablas que se guarda dentro de la base de datos. No es una tabla física, sino un objeto lógico que actúa como una tabla virtual.

Se emplean para simplificar consultas, mejorar la seguridad, reutilizar consultas comunes y modularizar la BBDD.

12. Explica el uso de las sentencias CREATE OR REPLACE VIEW

La sentencia “CREATE OR REPALCE VIEW” se utiliza para crear una nueva vista o para reemplazar una existente si ya tiene el mismo nombre. Esta función es útil para actualizar o modificar la definición de una vista sin tener que eliminar primero con “DROP VIEW” y luego recrearla con “CREATE VIEW”.

Sintaxis:

CREATE OR REPLACE VIEW nombre_vista AS

SELECT columnas

FROM tabla

WHERE condiciones;

Nombre_vista: es el nombre que queremos dar a la vista.

SELECT columnas, es la consulta que define que columnas y datos mostrará la vista.

FROM tabla, especifica las tablas de onde se obtendrán los datos.

WHERE condiciones, son las condiciones que filtraran y definirán los datos que estarán presentes en la vista.

13. ¿Cómo se otorgan privilegios al usuario sobre las vistas?

Se puede otorgar privilegios a usuarios sobre vistas de la misma manera que lo harías con las tablas. La gestión de los privilegios es una parte de la administración de la base de datos, proporciona que usuarios tienen permisos para hacer que acciones en las distintas estructuras de la base de datos.

Para otorgar permisos de lectura:

```
GRANT SELECT ON nombre_base_datos.nombre_vista TO 'nombre_usuario'@'hostname';
```

Para otorgar permisos de escritura (INSERT, UPDATE, DELETE)

```
GRANT INSERT, UPDATE, DELETE ON nombre_base_datos.nombre_vista TO 'nombre_usuario'@'hostname';
```

Para otorgar todos los permisos:

```
GRANT ALL PRIVILEGES ON nombre_base_datos.nombre_vista TO 'nombre_usuario'@'hostname';
```

Revocar y Otorgar privilegios complejos:

Revoca todos los privilegios existentes

```
REVOKE ALL PRIVILEGES ON nombre_base_datos.nombre_vista FROM 'nombre_usuario'@'hostname';
```

```
GRANT SELECT, INSERT ON nombre_base_datos.nombre_vista TO 'nombre_usuario'@'hostname';
```

Privilegios con opción de concesión:

```
GRANT SELECT ON nombre_base_datos.nombre_vista TO 'nombre_usuario'@'hostname' WITH GRANT OPTION;
```