TAREA 14

MORENO SANTOYO MARIANA

1 Comando SELECT

El comando `SELECT` se utiliza para recuperar datos de una base de datos SQL. Es esencial para cualquier consulta que busca mostrar información específica de las tablas de la base de datos.

1.1 Casos de Uso

- Seleccionar datos específicos de una tabla.
- Filtrar registros con condiciones específicas usando la cláusula `WHERE`.
- Ordenar los resultados con `ORDER BY`.

1.2 Restricciones

- El uso de `*` puede afectar el rendimiento en tablas grandes.
- Riesgo de inyección SQL si no se maneja adecuadamente la entrada de los usuarios.

1.3 Ejemplos

```
SELECT * FROM Usuarios WHERE edad > 30;
SELECT nombre FROM Usuarios ORDER BY nombre ASC;
```

2 Comando FROM

El comando `FROM` especifica la tabla de la cual se deben extraer los datos. Es parte fundamental de toda consulta que involucre `SELECT`.

2.1 Casos de Uso

- Especificar la tabla desde donde se seleccionan las columnas.
- Combinar múltiples tablas usando cláusulas de unión.

2.2 Restricciones

- Solo puede referenciar tablas existentes en la base de datos.
- Requiere un diseño eficiente de la base de datos para un buen rendimiento de las consultas.

2.3 Ejemplos

```
SELECT nombre, edad FROM Usuarios;
SELECT u.nombre, p.producto FROM Usuarios u JOIN Productos p ON u.id_usuario = p.id_usua
```

3 Comando JOIN

El comando `JOIN` en SQL se utiliza para combinar filas de dos o más tablas basadas en una columna relacionada entre ellas, siendo un componente esencial para consultas que integran datos de diversas tablas.

3.1 Casos de Uso

- Combinar tablas que tienen relaciones, como usuarios y pedidos, donde cada pedido está ligado a un usuario específico.
- Obtener reportes detallados que requieren la combinación de información de varias tablas, como clientes, productos y compras.

3.2 Restricciones

- Es crucial que las tablas que se unen tengan al menos una columna en común.
- El rendimiento puede degradarse si las tablas son grandes y no están bien indexadas.

3.3 Ejemplos

```
-- INNER JOIN que devuelve filas con coincidencias en ambas tablas:
SELECT empleados.nombre, departamentos.nombre
FROM empleados
JOIN departamentos ON empleados.dep_id = departamentos.id;

-- LEFT JOIN incluye todas las filas de la primera tabla y las coincidentes de la segunda
SELECT empleados.nombre, departamentos.nombre
FROM empleados
LEFT JOIN departamentos ON empleados.dep_id = departamentos.id;
```

3.4 Comando WHERE

El comando `WHERE` se utiliza en SQL para filtrar registros, permitiendo especificar qué registros deben ser incluidos en el resultado de una consulta basándose en condiciones específicas.

3.4.1 Casos de Uso

- Filtrar registros que cumplen una condición específica.
- Usarse en declaraciones de `SELECT`, `UPDATE`, y `DELETE` para limitar los datos sobre los cuales operar.

3.4.2 Restricciones

- No se puede usar con funciones agregadas directamente en las condiciones.
- Solo afecta las filas antes de que cualquier agrupación (`GROUP BY`) se aplique.

3.4.3 Ejemplo

```
SELECT nombre FROM usuarios WHERE edad >= 18;
```

Este ejemplo selecciona los nombres de los usuarios que son mayores de edad.

3.5 Comando HAVING

El comando `HAVING` se utiliza para filtrar registros agrupados en base a una condición, especialmente útil con funciones de agregado, como se ve en combinación con `GROUP BY`.

3.5.1 Casos de Uso

- Filtrar datos agrupados basándose en un criterio específico después de aplicar funciones de agregación.
- Usarse exclusivamente en consultas que también utilizan la cláusula `GROUP BY`.

3.5.2 Restricciones

- Requiere que las columnas usadas en la condición también estén incluidas en la cláusula `GROUP BY` o que sean usadas en una función de agregado.
- Se aplica después de que los datos han sido agrupados, por lo que no afecta la selección inicial de filas antes del agrupamiento.

3.5.3 Ejemplo

```
SELECT departamento, SUM(salario) FROM empleados
GROUP BY departamento HAVING SUM(salario) > 100000;
```

Este ejemplo muestra los departamentos cuya suma total de salarios excede los 100,000.

3.6 Subconsultas Correlacionadas

Las subconsultas correlacionadas en SQL son una técnica avanzada que permite referenciar elementos de la consulta externa dentro de la subconsulta para cada fila procesada. Esto las hace ideales para condiciones complejas que dependen de cada fila evaluada.

3.6.1 Casos de Uso

- Comparar cada fila con un valor agregado calculado a partir de la misma tabla o una relacionada.
- Filtrar registros según condiciones que involucran otros datos de la misma tabla o de tablas relacionadas.
- Actualizar o eliminar registros de una tabla basándose en condiciones que dependen de otros registros de la misma o de otras tablas.

3.6.2 Restricciones y Consideraciones de Rendimiento

- Las subconsultas correlacionadas pueden ser menos eficientes que otras construcciones SQL como los `JOIN`, especialmente si la consulta externa maneja muchos registros, debido a que la subconsulta se evalúa múltiples veces.
- Se recomienda evaluar alternativas más eficientes siempre que el rendimiento sea una preocupación.

3.6.3 Ejemplos Prácticos

```
-- Ejemplo usando EXISTS para encontrar empleados sin premios
SELECT last_name, first_name
FROM employee e1
WHERE NOT EXISTS (
    SELECT * FROM payment_history ph
    WHERE ph.employee_id = e1.employee_id AND ph.payment_type = 'award'
);

-- Ejemplo en la cláusula SELECT para calcular el salario promedio del departamento
SELECT employee_id, first_name, last_name, department_id,
```

```
       (SELECT AVG(salary) FROM employees
        WHERE department_id = e.department_id) AS avg_department_salary
FROM employees e;
```

4 Referencias

References

[1] LearnSQL.com, "Correlated Subquery in SQL: A Beginner's Guide," [Online]. Available: https://learnsql.com/blog/correlated-subquery-in-sql-beginners-guide/. [Accedido: 10- Abr- 2024].

[2] SQLTutorial.org, "SQL Correlated Subqueries," [Online]. Available: https://www.sqltutorial.org/sql-subquery/correlated-subquery/. [Accedido: 10- Abr- 2024].

[3] LearnSQL.com, "WHERE and HAVING: Simple Examples," [Online]. Available: https://learnsql.com/blog/having-vs-where/. [Accedido: 10- Abr- 2024].

[4] W3Schools, "SQL HAVING Clause," [Online]. Available: https://www.w3schools.com/sql/sql$_h$*aving.asp.*[*Accedido* : 10 − *Abr* − 2024].*SQLTutorial.org*, "*SQLHAVING* : *TheUltimateGuide*|*HAVINGvs.WHERE*," [*Online*].*Available* : *https* : *//www.sqltutorial.org/sql* − *having/*.[*Accedido* : 10 − *Abr* − 2024].

[5] LearnSQL.com, "How To Learn The SELECT Statement in SQL," [Online]. Available: https://learnsql.com/. [Accedido: 10- Abr- 2024].

[6] freeCodeCamp, "SQL Select – Statement and Query Examples," [Online]. Available: https://www.freecodecamp.org/. [Accedido: 10- Abr- 2024].

[7] SQLShack, "SQL examples for beginners: SQL SELECT statement usage," [Online]. Available: https://www.sqlshack.com/. [Accedido: 10- Abr- 2024].

[8] Devart Blog, "SQL SELECT Statement (Basics and Practical Examples)," [Online]. Available: https://blog.devart.com/. [Accedido: 10- Abr- 2024].

[9] SQL Tutorial, "FROM Clause in SQL," [Online]. Available: https://www.sqltutorial.org/sql-from/. [Accedido: 10- Abr- 2024].

[10] W3Schools, "SQL FROM Clause," [Online]. Available: https://www.w3schools.com/sql/sql$_f$*rom.asp.*[*Ac* 10 − *Abr* − 2024].