Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO. 3

ÁLGEBRA RELACIONAL Y SQL

ALUMNO: David Arael Vargas Alarcon

CATEDRÁTICO: Dr. Eduardo Cornejo Velázquez

1. Introducción

En el contexto de las bases de datos distribuidas, es esencial comprender y aplicar los conceptos fundamentales de álgebra relacional y SQL (Structured Query Language), ya que estos son los pilares que permiten interactuar y manipular bases de datos de manera eficiente y efectiva. El álgebra relacional proporciona el marco teórico que sustenta las operaciones sobre conjuntos de datos, definiendo cómo se pueden combinar, filtrar y transformar las tablas de una base de datos. SQL, por su parte, es el lenguaje práctico que implementa estos conceptos, permitiendo a los usuarios realizar consultas complejas, administrar datos, y mantener la integridad y seguridad de la información.
En esta práctica, se realizaron diversos ejercicios utilizando MySQL, un sistema de gestión de bases de datos relacional (RDBMS) ampliamente utilizado en la industria. MySQL es conocido por su fiabilidad, flexibilidad, y capacidad para manejar grandes volúmenes de datos, lo que lo convierte en una herramienta ideal para reforzar la comprensión y habilidades en álgebra relacional y SQL. A través de estos ejercicios, los participantes pudieron aplicar conceptos teóricos en situaciones prácticas, desde la creación y modificación de tablas hasta la realización de consultas avanzadas que implican manipulación de cadenas de texto, operaciones con fechas, y la ordenación de resultados. Además, se exploró la importancia de la seguridad en las bases de datos, destacando las vulnerabilidades como la inyección SQL y las mejores prácticas para mitigarlas. Esta práctica no solo fortaleció las competencias técnicas en el manejo de bases de datos, sino que también subrayó la importancia de una gestión cuidadosa y segura de los datos en entornos distribuidos y colaborativos.

2. Marco teórico

Álgebra Relacional

El álgebra relacional es un lenguaje procedural que proporciona las operaciones fundamentales necesarias para consultar y manipular datos en una base de datos relacional. Las operaciones básicas incluyen selección, proyección, unión, intersección, diferencia, producto cartesiano, y operaciones de conjunto como unión y diferencia.

1. Selección (): Permite extraer filas que cumplen una determinada condición.
2. Proyección (): Se utiliza para seleccionar columnas específicas de una tabla.
3. Unión (): Combina los resultados de dos consultas, eliminando duplicados.
4. Intersección (): Retorna las filas que son comunes a dos tablas.
5. Diferencia (-): Muestra las filas que están en una tabla pero no en la otra.
6. Producto cartesiano (×): Combina todas las filas de dos tablas.

SQL (Structured Query Language)

SQL es un lenguaje estándar utilizado para gestionar y manipular bases de datos relacionales. SQL permite realizar consultas, insertar, actualizar y eliminar datos, así como definir esquemas de bases de datos y gestionar transacciones.

1. SELECT: Se utiliza para consultar datos de una base de datos.
2. INSERT: Añade nuevos registros a una tabla.
3. UPDATE: Modifica los datos existentes en una tabla.
4. DELETE: Elimina registros de una tabla.
5. CREATE TABLE: Define la estructura de una tabla.
6. DROP TABLE: Elimina una tabla de la base de datos.
7. JOIN: Combina filas de dos o más tablas basadas en una relación común entre ellas.

MySQL

MySQL es un RDBMS que utiliza SQL para gestionar y manipular bases de datos. Es conocido por su rendimiento, flexibilidad y facilidad de uso, lo que lo hace ideal para una amplia gama de aplicaciones, desde pequeñas aplicaciones hasta grandes sistemas empresariales.

3. Herramientas empleadas

MySQL Workbench

ERD Plus es una herramienta en línea utilizada para crear diagramas de entidad-relación (ERD) y diagramas relacionados con bases de datos. ERD Plus permite a los usuarios diseñar visualmente la estructura de una base de datos, incluyendo entidades (tablas), relaciones, y atributos. Es especialmente útil para modelar bases de datos antes de implementarlas en un sistema de gestión de bases de datos (DBMS).

MySQL Server

MySQL Server es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto y ampliamente utilizado. MySQL utiliza el lenguaje SQL (Structured Query Language) para gestionar y manipular datos. Es conocido por su alta velocidad, fiabilidad y facilidad de uso, y es compatible con diferentes plataformas, incluyendo Windows, Linux, y macOS.

1. Utlización

   (a) Gestión de Bases de Datos: Se utiliza para almacenar, organizar y gestionar grandes volúmenes de datos en aplicaciones web y empresariales.

   (b) Aplicaciones Web: Es comúnmente utilizado en combinación con lenguajes de programación como PHP para desarrollar aplicaciones web dinámicas y sitios web que requieren bases de datos.

   (c) Análisis de Datos: MySQL Server permite realizar consultas complejas para extraer, analizar y reportar datos, lo que es útil en áreas como el análisis de negocio y la inteligencia empresarial.

   (d) Escalabilidad: Es adecuado tanto para pequeñas aplicaciones como para grandes sistemas que requieren manejo de grandes volúmenes de datos y alta concurrencia.

Termnial/Command line

Para ejecutar comandos SQL directamente en el servidor MySQL.

4. Desarrollo

Employee Table

En la siguiente imagen, se presentan las siguientes tablas, las cuales sirvieron de apoyo para realizar los ejercicios que se solicitan

Employee table

Employee_id	First_name	Last_name	Salary	Joining_date	Departement
1	Bob	Kinto	1000000	2019-01-20	Finance
2	Jerry	Kansxo	6000000	2019-01-15	IT
3	Philip	Jose	8900000	2019-02-05	Banking
4	John	Abraham	2000000	2019-02-25	Insurance
5	Michael	Mathew	2200000	2019-02-28	Finance
6	Alex	chreketo	4000000	2019-05-10	IT
7	Yohan	Soso	1230000	2019-06-20	Banking

Reward table

Employee_ref_id	date_reward	amount
1	2019-05-11	1000
2	2019-02-15	5000
3	2019-04-22	2000
1	2019-06-20	8000

Ejercicios a resolver

1. Obtener el tamaño del texto en todos los valores de la columna “First name”.
   (a) Algebra relacional:
   $$PI_{LENGTH(First_name)}(Employee)$$
   (b) SQL:
   ```
   SELECT LENGTH(First_name) AS Length_of_First_name FROM Employee;
   ```
2. Obtener el nombre de todos los empleados después de reemplazar ‘o’ con ‘gato’.
   (a) Algebra relacional:
   $$_{REPLACE(First_name,'o','')}(Employee)$$
   (b) SQL:
   ```
   SELECT REPLACE(First_name, 'o', '#') AS Modified_First_name FROM Employee;
   ```

3. Obtener el nombre y apellido de todos los empleados en una sola columna separados por “Guion bajo”..

   (a) Algebra relacional:

   $$_{CONCAT(First_name,'_',Last_name)}(Employee)$$

   (b) SQL:

```
SELECT CONCAT(First_name, '_', Last_name) AS Full_Name FROM
Employee;
```

4. Obtener el año, mes y día de la columna “Joining date”.

   (a) Algebra relacional:

   $$_{YEAR(Joining_date),MONTH(Joining_date),DAY(Joining_date)}(Employee)$$

   (b) SQL:

```
SELECT YEAR(Joining_date) AS Year, MONTH(Joining_date) AS Month,
DAY(Joining_date) AS Day FROM Employee;
```

5. Obtener todos los empleados en orden ascendente por nombre.

   (a) Algebra relacional:

   $$_{First_name}(Employee)(ORDERBY First_name ASC)$$

   (b) SQL:

```
SELECT * FROM Employee ORDER BY First_name ASC;
```

6. Obtener todos los empleados en orden descendente por nombre.

   (a) Algebra relacional:

   $$_{First_name}(Employee)(ORDERBY First_name DESC$$

   (b) SQL:

```
SELECT * FROM Employee ORDER BY First_name DESC;
```

7. Obtener todos los empleados en orden ascendente por nombre y en orden descendente por salario.

   (a) Algebra relacional:

   $$_{First_name,Salary}(Employee)(ORDERBY First_name ASC, Salary DESC)$$

   (b) SQL:

```
SELECT * FROM Employee ORDER BY First_name ASC, Salary DESC;
```

8. Obtener todos los empleados con el nombre “Bob”.

   (a) Algebra relacional:

   $$_{First_name='Bob'}(Employee)$$

   (b) SQL:

```
SELECT * FROM Employee WHERE First_name = 'Bob';
```

9. Obtener todos los empleados con el nombre “Bob” o “Alex”.

   (a) Algebra relacional:

   $$_{First_name='Bob' First_name='Alex'}(Employee)$$

   (b) SQL:

   ```
   SELECT * FROM Employee WHERE First_name IN ('Bob', 'Alex');
   ```

10. Obtener todos los empleados que no tengan el nombre “Bob” o “Alex”.

    (a) Algebra relacional:

    $$_{First_name'Bob' First_name'Alex'}(Employee))$$

    (b) SQL:

    ```
    SELECT * FROM Employee WHERE First_name NOT IN ('Bob', 'Alex');
    ```

11. ¿Qué es una inyección SQL?

    (a) Definición: Una inyección SQL es un tipo de ataque de seguridad en el cual un atacante puede interferir en las consultas que una aplicación hace a su base de datos. Esto se hace mediante la inserción maliciosa de código SQL en un campo de entrada, lo que puede resultar en la manipulación de la base de datos subyacente. Esto puede permitir al atacante realizar acciones no autorizadas como leer datos sensibles, modificar o eliminar datos, o ejecutar comandos en la base de datos.

    (b) Ejemplo de Inyección SQL: Si un formulario web pide un nombre de usuario y lo inserta directamente en una consulta SQL sin validación, un atacante podría ingresar algo como ’; DROP TABLE Employee; –, que eliminaría la tabla de empleados si no se maneja correctamente.

5. Conclusión

Esta actividad aborda la manipulación de bases de datos utilizando SQL, destacando cómo aplicar operaciones básicas y avanzadas para extraer y modificar datos. Al trabajar con funciones como LENGTH, REPLACE, CONCAT, y métodos de ordenamiento, se demuestra la versatilidad de SQL para gestionar datos de manera eficiente. La creación de tablas y la inserción de datos en ellas proporcionan una base sólida para realizar consultas específicas. Además, se resalta la importancia de la seguridad en las bases de datos a través de la explicación de las inyecciones SQL, subrayando la necesidad de validar entradas para evitar vulnerabilidades. Este ejercicio no solo fortalece las habilidades técnicas en SQL, sino que también fomenta una comprensión crítica de la protección de datos y la integridad de las bases de datos en aplicaciones reales.