Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO. 2

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). Estos conceptos son la base para interactuar y manipular bases de datos de manera eficiente. En esta práctica, se realizaron diversos ejercicios utilizando MySQL, un sistema de gestión de bases de datos relacional (RDBMS) popular, para reforzar la comprensión y habilidades en estas áreas.

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. Escribe la sintaxis para crear la tabla “Employee”.

```
CREATE TABLE Employee (
EmployeeID INT PRIMARY KEY,
First_name VARCHAR(50),
Last_name VARCHAR(50),
Salary DECIMAL(10, 2),
Joining_date DATE NOT NULL,
Department VARCHAR(50)
);
```

2. Escribe la sintaxis para insertar 7 registros (de la imagen) a la tabla “Employee”.

```
INSERT INTO Employee (EmployeeID, First_name, Last_name, Salary,
Joining_date, Department)
VALUES
(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');
```

3. Escribe la sintaxis para crear la tabla “Reward”.

```
    CREATE TABLE Employee_ref (
    Employee_ref_id INT PRIMARY KEY,
    date_reward DATE,
    amount DECIMAL(10, 2)
);
```

4. Escribe la sintaxis para insertar 4 registros (en la imagen) a la tabla “Reward”.

```
INSERT INTO Employee_ref (Employee_ref_id, date_reward, amount)
VALUES
(1, '2019-05-11', 1000),
(2, '2019-02-15', 5000),
(3, '2019-04-22', 2000),
(4, '2019-06-20', 8000);
```

5. Obtener todos los empleados.

   (a) Algebra relacional: True(Employee)

   (b) SQL:

   ```
   SELECT * FROM Employee;
   ```

6. Obtener el primer nombre y apellido de todos los empleados.

   (a) Algebra relacional: First name,Last name (Employee)

   (b) SQL:

   ```
   SELECT First_name, Last_name FROM Employee;
   ```

7. Obtener todos los valores de la columna “First-name” usando el alias “Nombre de empleado”.

   (a) Algebra relacional: First n ame→Nombre de Empleado (Employee)

   (b) SQL:

   ```
   SELECT First_name AS 'Nombre de empleado' FROM Employee;
   ```

8. Obtener todos los valores de la columna "Last$_name$" *enminúsculas*.

9. Algebra Relacional: NO hay operación

10. SQL:

```
SELECT LOWER(Last_name) FROM Employee;
```

Obtener todos los valores de la columna "Last$_name$" *enmayusculas*.

Algebra Relacional: NO hay operación

SQL:

```
SELECT UPPER(Last_name) FROM Employee;
```

Obtener los nombre únicos de la columna "Departament".

1. Algebra Relacional: N Departament (Employee)
2. SQL:

```
SELECT DISTINCT Department FROM Employee;
```

Obtener los primeros 4 caracteres de todos los valors de la columna "First$_name$".

Algebra relacional: No hay operación

SQL:

```
SELECT LEFT(First_name, 4) FROM Employee;
```

Obtener la posición de la letra "h" en el nombre del empleado con First$_name = \mathit{Jhon}$".

Algebra relacional: No hay operación

SQL:

```
SELECT INSTR(First_name, 'h') FROM Employee WHERE First_name = 'Jhon';
```

Obtener todos los valores de la columna "First$_name$" *despuésremovertodoslosespaciosenblancoaladerecha*.

SQL:

```
SELECT RTRIM(First_name) FROM Employee;
```

Obtener todos los valores de la columna "First$_name$" *despuésderemoverlosespaciosenblancodelaizquie*

SQL:

```
SELECT LTRIM(First_name) FROM Employee;
```