

REPORTE DE PRÁCTICA NO. 3

ALGEBRA RELACIONAL Y SQL

ALUMNO: Perez Orta Braulio Adrian
Dr. Eduardo Cornejo-Velázquez



1. Introducción

El álgebra relacional es la base matemática que subyace en la gestión de bases de datos relacionales. Imagina que las bases de datos son como grandes almacenes llenos de información organizada en tablas. El álgebra relacional nos proporciona un conjunto de operaciones para trabajar con estas tablas de manera lógica y coherente.

Algunas de las operaciones más importantes del álgebra relacional incluyen:

El álgebra relacional constituye la base matemática fundamental para la gestión de bases de datos relacionales. Para visualizarlo mejor, imagina que las bases de datos son enormes almacenes repletos de información organizada en tablas. El álgebra relacional nos ofrece un conjunto de operaciones que nos permiten manipular y consultar estas tablas de manera lógica y estructurada.

Entre las operaciones más destacadas del álgebra relacional se encuentran:

Selección: Esta operación nos permite filtrar las filas de una tabla según ciertos criterios específicos. Por ejemplo, podríamos seleccionar todas las filas de una tabla de empleados donde el salario sea superior a un valor determinado.

Proyección: Con esta operación, podemos elegir columnas específicas de una tabla. Imagina que tienes una tabla con información sobre clientes y solo te interesa obtener sus nombres y direcciones. La proyección te permite extraer únicamente esas dos columnas.

Unión: Esta operación combina dos tablas en una sola tabla más grande. Por ejemplo, si tienes una tabla con información de productos y otra con información de pedidos, puedes unirlos para obtener una vista completa de los productos que se han pedido.

Intersección: Nos proporciona la parte común entre dos tablas. Si tienes una tabla con empleados de tiempo completo y otra con empleados de medio tiempo, la intersección te mostrará aquellos empleados que están en ambas categorías.

Diferencia: Esta operación nos permite identificar las filas que están presentes en una tabla pero no en otra. Por ejemplo, podríamos usar la diferencia para encontrar los productos que están en stock pero que aún no se han vendido.

2. Marco teórico

Bases de Datos y su Importancia

Las bases de datos son fundamentales para el funcionamiento de la mayoría de las aplicaciones informáticas actuales. Puedes imaginar que son como vastos almacenes digitales donde se organiza y guarda información estructurada. Estas bases de datos nos ofrecen varias capacidades esenciales:

Almacenamiento de Datos: Desde registros de clientes hasta detalles de productos, las bases de datos permiten almacenar información de manera ordenada y eficiente.

Recuperación de Datos: A través de consultas, podemos extraer información específica de las bases de datos.

Actualización de Datos: Las bases de datos nos permiten modificar registros existentes.

Eliminar Datos: Eliminación de Datos: Cuando cierta información ya no es necesaria, puede borrarse.

3. Herramientas empleadas

Este Informe utiliza el formato latex para su redaccion, MySQL command line para las bases de datos y las actividades requeridas asi como github para subir adecuadamente el trabajo

4. Desarrollo

Sentencias SQL

```
create table Employee ( EmployeeID INT AUTO INCREMENT PRIMARY KEY, First-name VARCHAR(50),  
Last-name VARCHAR(50), Joining-date DATE, Salary DECIMAL(10, 2),  
);  
CREATE TABLE Reward ( RewardID INT AUTO INCREMENT PRIMARY KEY, EmployeeID INT,  
Reward-type VARCHAR(50), Reward-date DATE, FOREIGN KEY (EmployeeID) REFERENCES Em-  
ployee(EmployeeID) );  
); y sql INSERT INTO Employee (First-name, Last-name, Joining-date, Salary) VALUES ('Bob', 'Smith',  
'2020-01-15', 50000.00), ('Alex', 'Johnson', '2019-03-22', 60000.00), ('Alice', 'Williams', '2021-07-30', 55000.00);  
INSERT INTO Reward (EmployeeID, Reward-type, Reward-date) VALUES (1, 'Employee of the Month',  
'2021-02-01'), (2, 'Best Performance', '2021-05-15');
```

1. 

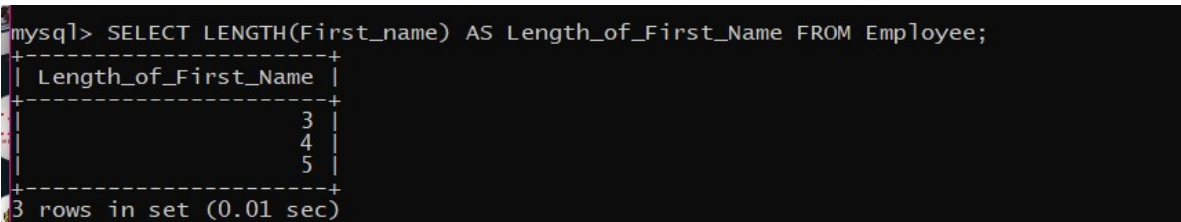
```
mysql> use practica2  
Database changed  
mysql> CREATE TABLE Employee (EmployeeID INT AUTO_INCREMENT PRIMARY KEY, First_name VARCHAR(50), Last_name VARCHAR(50), Joining_date DATE, Salary DECIMAL(10, 2));  
Query OK, 0 rows affected (0.03 sec)
```
2. 

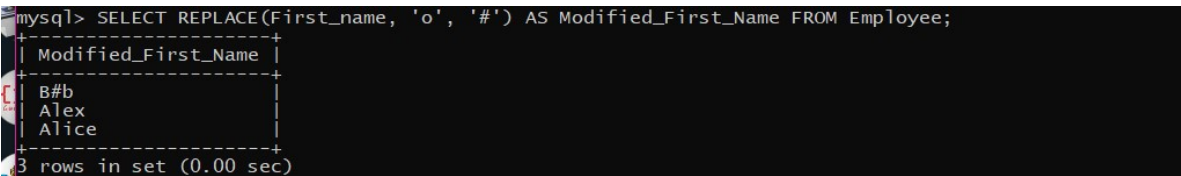
```
mysql> CREATE TABLE Reward (RewardID INT AUTO_INCREMENT PRIMARY KEY, EmployeeID INT, Reward_type VARCHAR(50), Reward_date DATE, FOREIGN KEY (EmployeeID) RE  
REFERENCES Employee(EmployeeID));  
Query OK, 0 rows affected (0.03 sec)
```
3. 

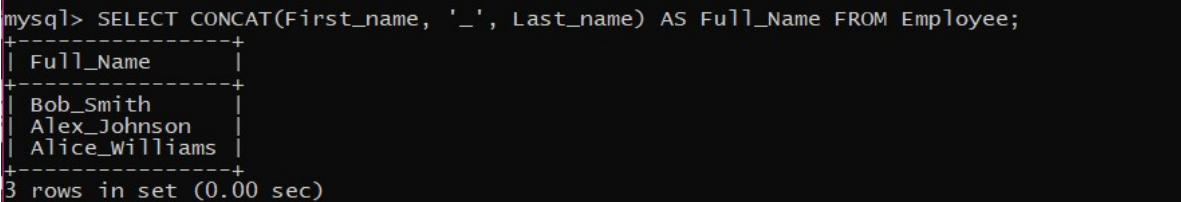
```
mysql> INSERT INTO Employee (First_name, Last_name, Joining_date, Salary) VALUES('Bob', 'Smith', '2020-01-15', 50000.00),('Alex', 'Johnson', '2019-03-22', 60000.00),('Alic  
e', 'Williams', '2021-07-30', 55000.00);  
Query OK, 3 rows affected (0.01 sec)  
Records: 3 Duplicates: 0 Warnings: 0
```
4. 

```
mysql> INSERT INTO Reward (EmployeeID, Reward_type, Reward_date) VALUES(1, 'Employee of the Month', '2021-02-01'),(2, 'Best Performance', '2021-05-15');  
Query OK, 2 rows affected (0.01 sec)  
Records: 2 Duplicates: 0 Warnings: 0
```

PROYECCIONES

5. 

```
mysql> SELECT LENGTH(First_name) AS Length_of_First_Name FROM Employee;  
+-----+  
| Length_of_First_Name |  
+-----+  
| 3 |  
| 4 |  
| 5 |  
+-----+  
3 rows in set (0.01 sec)
```
6. 

```
mysql> SELECT REPLACE(First_name, 'o', '#') AS Modified_First_Name FROM Employee;  
+-----+  
| Modified_First_Name |  
+-----+  
| B#b |  
| Alex |  
| Alice |  
+-----+  
3 rows in set (0.00 sec)
```
7. 

```
mysql> SELECT CONCAT(First_name, '_', Last_name) AS Full_Name FROM Employee;  
+-----+  
| Full_Name |  
+-----+  
| Bob_Smith |  
| Alex_Johnson |  
| Alice_Williams |  
+-----+  
3 rows in set (0.00 sec)
```

8.

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

Year	Month	Day
2020	1	15
2019	3	22
2021	7	30

3 rows in set (0.00 sec)

9.

```
mysql> SELECT * FROM Employee ORDER BY First_name ASC  
-> ;
```

EmployeeID	First_name	Last_name	Joining_date	Salary
2	Alex	Johnson	2019-03-22	60000.00
3	Alice	Williams	2021-07-30	55000.00
1	Bob	Smith	2020-01-15	50000.00

3 rows in set (0.00 sec)

10.

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

EmployeeID	First_name	Last_name	Joining_date	Salary
1	Bob	Smith	2020-01-15	50000.00
3	Alice	Williams	2021-07-30	55000.00
2	Alex	Johnson	2019-03-22	60000.00

3 rows in set (0.00 sec)

11.

```
mysql> ypSELECT * FROM Employee ORDER BY First_name ASC, Salary DESC;  
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'ypSELECT * FROM  
Employee ORDER BY First_name ASC, Salary DESC' at line 1  
mysql> SELECT * FROM Employee ORDER BY First_name ASC, Salary DESC;
```

EmployeeID	First_name	Last_name	Joining_date	Salary
2	Alex	Johnson	2019-03-22	60000.00
3	Alice	Williams	2021-07-30	55000.00
1	Bob	Smith	2020-01-15	50000.00

3 rows in set (0.00 sec)

12.

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

EmployeeID	First_name	Last_name	Joining_date	Salary
1	Bob	Smith	2020-01-15	50000.00

1 row in set (0.00 sec)

13.

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

EmployeeID	First_name	Last_name	Joining_date	Salary
1	Bob	Smith	2020-01-15	50000.00
2	Alex	Johnson	2019-03-22	60000.00

2 rows in set (0.00 sec)

14.

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

EmployeeID	First_name	Last_name	Joining_date	Salary
3	Alice	Williams	2021-07-30	55000.00

1 row in set (0.00 sec)

15. La inyección SQL es una técnica de ataque en la que un atacante inserta o "inyecta" código SQL malicioso en una consulta SQL a través de la entrada de datos de un usuario. Esto puede permitir al atacante manipular la base de datos, acceder a datos sensibles, o incluso ejecutar comandos en el servidor. Es importante protegerse contra inyecciones SQL utilizando prácticas como la validación de entradas y el uso de consultas preparadas.:

5. Conclusiones

En conclusión, este informe nos ha permitido comprender mejor cómo funcionan las bases de datos y las herramientas que utilizamos para gestionarlas. El álgebra relacional nos proporciona una base teórica sólida para trabajar con datos, mientras que SQL nos ofrece un lenguaje práctico y poderoso para interactuar con esos datos. MySQL, por su parte, nos demuestra cómo estos conceptos se aplican en un sistema de gestión de bases de datos real y ampliamente utilizado. Esta combinación de teoría y práctica es crucial para cualquier profesional que desee especializarse en el campo de la gestión de bases de datos.

Referencias Bibliográficas

Referencias: : Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, 13(6), 377-387. : Date, C. J. (2003). An Introduction to Database Systems (8th ed.). Addison-Wesley. DuBois, P. (2020). MySQL: The definitive guide to using, programming, and administering MySQL 8.0. O'Reilly Media. Silberschatz, A., Korth, H. F., Sudarshan, S. (2019). Database system concepts (7th ed.). McGraw-Hill Education.