REPORTE DE PRÁCTICA NO 1.3

Practica 1.3 - Álgebra relacional y SQL (1)

ALUMNO: Esperilla Mendoza Luis Erick
Dr. Eduardo Cornejo-Velázquez

1. Introducción

El diseño y gestión de bases de datos distribuidas requieren un enfoque sistemático para garantizar una manipulación eficiente de la información.
En esta práctica, se utilizará MySQL para crear y consultar las tablas Employee y Reward, aplicando sentencias SQL clave. Esto permitirá reforzar habilidades en manipulación de datos, uso de alias y optimización de consultas.

2. Marco teórico

Álgebra relacional

El álgebra relacional es un conjunto de operaciones formales utilizadas en bases de datos relacionales para manipular y consultar datos. Se basa en operadores matemáticos que permiten combinar, filtrar y transformar datos almacenados en tablas. Entre sus principales operaciones se encuentran:
Selección: Filtra filas que cumplen una condición.
Proyección: Selecciona columnas especficas.
Unión: Combina dos conjuntos de datos.
Diferencia: Devuelve los elementos de una tabla que no están en otra.
Producto cartesiano: Combina todas las filas de dos tablas.
Join: Une tablas según una condición común.

SQL

SQL es un lenguaje estándar para gestionar bases de datos relacionales.
Permite definir, manipular y consultar datos mediante diversas sentencias.
Se divide en tres categorías principales:

1. **DDL (Data Definition Language)** – Define la estructura de la base de datos.
   - `CREATE TABLE` – Crea tablas.
   - `ALTER TABLE` – Modifica estructuras de tablas.
   - `DROP TABLE` – Elimina tablas.
2. **DML (Data Manipulation Language)** – Manipula los datos dentro de la base.
   - `INSERT INTO` – Agrega registros.
   - `UPDATE` – Modifica registros existentes.
   - `DELETE` – Elimina registros.
3. **DQL (Data Query Language)** – Consulta los datos almacenados. `SELECT` – Recupera información de las tablas.

MySQL

SENTENCIAS UTILIZADAS

Listing 1: Crear base de datos

```
CREATE DATABASE;
```

Se usa para crear una nueva base de datos.

Listing 2: Usar base de datos

```
USE;
```

Permite seleccionar una base de datos para trabajar con ella.

Listing 3: Crear una tabla en la base de datos

```
CREATE TABLE;
```

Define una nueva tabla dentro de la base de datos con sus columnas y tipos de datos.

Listing 4: Agregar registros en las tablas

```
INSERT INTO;
```

Agrega registros a una tabla existente.

Listing 5: Agregar registros en las tablas

```
INSERT INTO;
```

Agrega registros a una tabla existente.

Listing 6: Seleccionar algo de una tabla

```
SELECT;
```

Se utiliza para recuperar datos de una tabla.

Listing 7: Especificar la tabla de la que queremos obtener informacion

```
FROM;
```

Especifica la tabla de la cual se obtendrán los datos en una consulta.

Listing 8: Convertir a minusculas

```
SELECT LOWER();
```

Convierte los valores de una columna en minúsculas.

Listing 9: Convertir a mayusculas

```
SELECT UPPER();
```

Convierte los valores de una columna en mayúsculas.

3. Herramientas empleadas

- **DataGrip**
  - **Tipo**: IDE para bases de datos desarrollado por JetBrains.
  - **Uso**: Se utiliza para gestionar y administrar bases de datos, escribir y ejecutar consultas SQL, diseñar esquemas de bases de datos, y conectarse a múltiples servidores de bases de datos, todo en una sola interfaz.

4. Desarrollo

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

```
CREATE TABLE employee (
    Employee_id INT AUTO_INCREMENT PRIMARY KEY,
    First_name VARCHAR(80) NOT NULL,
    Last_name VARCHAR(80) NOT NULL,
    Salary FLOAT NOT NULL,
    Joining_date DATE NOT NULL,
    Department VARCHAR(80) NOT NULL
);
```

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

```
INSERT INTO employee (First_name, Last_name, Salary, Joining_date, Department)
VALUES
        ('Bob', 'Kinto', 1000000, '2019-01-20', 'Finance'),
        ('Jerry', 'Kansxo', 6000000, '2019-01-15', 'IT'),
        ('Philip', 'Jose', 8900000, '2019-02-05', 'Banking'),
        ('John', 'Abraham', 2000000, '2019-02-25', 'Insurance'),
        ('Michael', 'Mathew', 2200000, '2019-02-28', 'Finance'),
        ('Alex', 'Chreketo', 4000000, '2019-05-10', 'IT'),
        ('Yohan', 'Soso', 1230000, '2019-06-20', 'Banking');
```

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

```
CREATE TABLE reward (
    date_reward DATE NOT NULL,
    amount FLOAT NOT NULL,
    employee_ref_id INT,
    FOREIGN KEY (employee_ref_id) REFERENCES employee(employee_id)
);
```

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

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

5. Obtener todos los empleados.

SELECT * FROM employee;

$$\sigma_{\text{true}}(\text{employee}) \tag{1}$$

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

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

```
SELECT employee.First_name, employee.Last_name FROM employee;
```

$$\pi_{\text{First\_name},\text{Last\_name}}(\text{employee})$$

	First_name	Last_name
1	Bob	Kinto
2	Jerry	Kansxo
3	Philip	Jose
4	John	Abraham
5	Michael	Mathew
6	Alex	Chreketo
7	Yohan	Soso

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

```
SELECT employee.First_name AS Nombre_de_empleado FROM employee;
```

$$\rho_{\text{Nombre\_de\_empleado}/\text{First\_name}}(\pi_{\text{First\_name}}(\text{employee}))$$

	Nombre_de_empleado
1	Bob
2	Jerry
3	Philip
4	John
5	Michael
6	Alex
7	Yohan

8. Obtener todos los valores de la columna “LastName” en minúsculas.

```
SELECT LOWER(employee.Last_name) FROM employee;
```

	`LOWER(employee.Last_name)`
1	kinto
2	kansxo
3	jose
4	abraham
5	mathew
6	chreketo
7	soso

$$\pi_{\text{LOWER(Last\_name)}}(\text{employee})$$

9. Obtener todos los valores de la columna “LastName” en mayúsculas.

```
SELECT UPPER(employee.Last_name) FROM employee;
```

$$\pi_{\text{UPPER(Last\_name)}}(\text{employee})$$

	`UPPER(employee.Last_name)`
1	KINTO
2	KANSXO
3	JOSE
4	ABRAHAM
5	MATHEW
6	CHREKETO
7	SOSO

10. Obtener los nombre únicos de la columna “Departament”.

```
SELECT DISTINCT employee.Department FROM employee;
```

$$\pi_{\text{Department}}(\text{employee})$$

	Department
1	Finance
2	IT
3	Banking
4	Insurance

11. Obtener los primeros 4 caracteres de todos los valores de la columna “FirstName”.

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

$$\pi_{\text{LEFT(First\_name, 4)}}(\text{employee})$$

	`LEFT(First_name, 4)`
1	Bob
2	Jerr
3	Phil
4	John
5	Mich
6	Alex
7	Yoha

12. Obtener la posición de la letra “h” en el nombre del empleado con FirstName = “John”.

```
SELECT POSITION('h' IN First_name)
FROM employee
WHERE First_name = 'John';
```

$$\pi_{\text{POSITION('h' IN First\_name)}}(\sigma_{\text{First\_name}='Jhon'}(\text{employee}))$$

	`POSITION('h' IN First_name)`
1	3

13. Obtener todos los valores de la columna “FirstName” después de remover los espacios en blanco de la derecha.

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

$$\pi_{\text{RTRIM(First\_name)}}(\text{employee})$$

	`RTRIM(First_name)`
1	Bob
2	Jerry
3	Philip
4	John
5	Michael
6	Alex
7	Yohan

14. Obtener todos los valores de la columna “FirstName” después de remover los espacios en blanco de la izquierda.

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

$$\pi_{\text{LTRIM(First\_name)}}(\text{employee})$$

	`LTRIM(First_name)`
1	Bob
2	Jerry
3	Philip
4	John
5	Michael
6	Alex
7	Yohan

5. Conclusiones

Durante esta práctica, fortalecí mis habilidades tanto en álgebra relacional como en SQL. El álgebra relacional me ayudó a comprender las operaciones fundamentales como selección y proyección, mientras que SQL me permitió manipular datos de manera práctica.

Lo más sorprendente fue ver cómo ambas herramientas se complementan: el álgebra relacional para el análisis estructural y SQL para la manipulación directa de datos.

Un desafío que enfrenté fue la necesidad de tener en cuenta el tipo de datos al aplicar ciertas operaciones, lo que resaltó la importancia de entender bien el esquema de la base de datos.

Referencias Bibliográficas

References

[1] Gómez, F. J. (2016). *SQL y álgebra relacional: Teoría y práctica* (2daed.). Editorial Académica Española.

[2] Martínez, A. (2018). *SQL y diseño de bases de datos relacionales* (3ra ed.). Ediciones Técnicas

[3] García, J. A., & Rodríguez, L. (2017). *Bases de datos: Teoría y práctica* (4ta ed.). Editorial Universitaria.