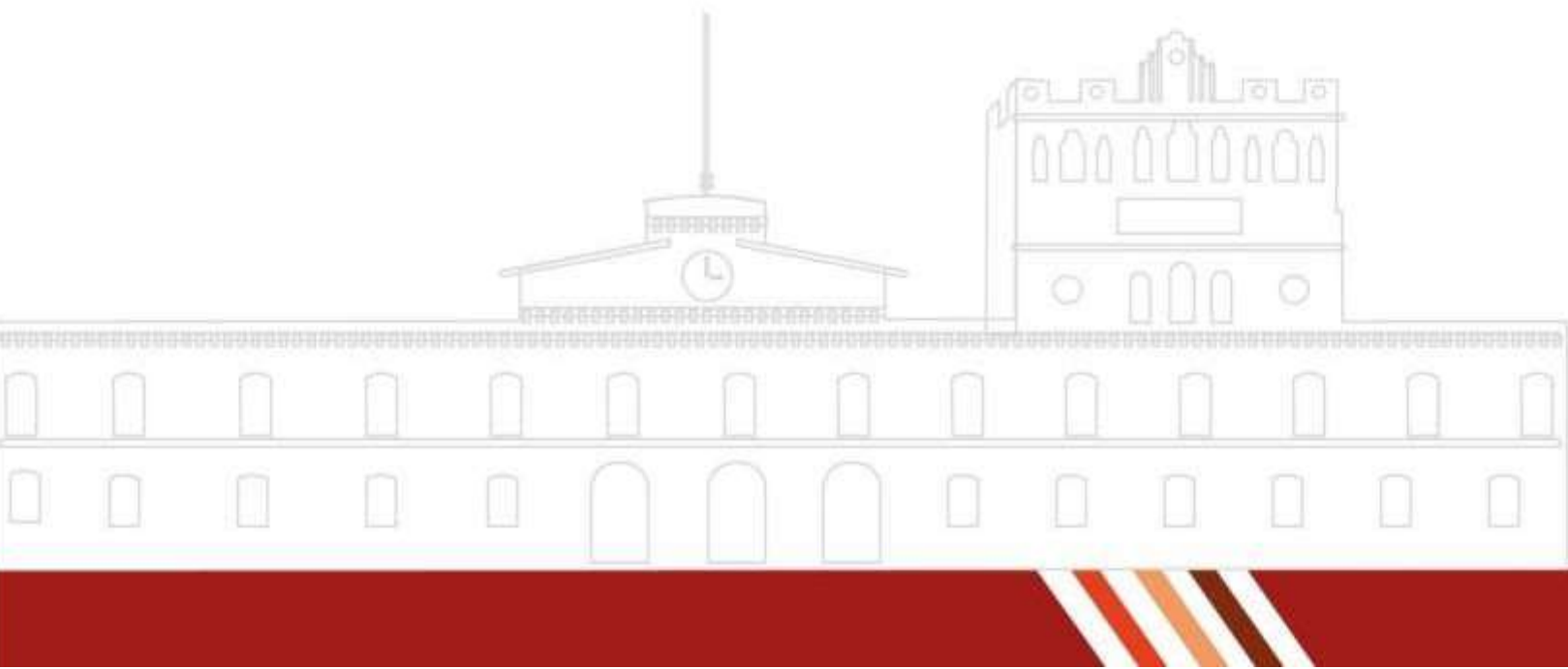




***Reporte de Practica No. 1.3 Nombre de  
la Practica: EJERCICIOS 1  
Alumno: Kevin Badillo Olmos  
Dr. Eduardo Cornejo Velázquez***



# Introducción

En el reporte se desarrollan ejercicios prácticos utilizando álgebra relacional y su implementación en SQL con MySQL, aplicados a las tablas *Employee* y *Reward*. El objetivo es reforzar los conceptos teóricos y la práctica de consultas básicas, creación e inserción de datos, y manipulación de cadenas.

A diferencia de SQL, que es un lenguaje de implementación, el álgebra relacional se enfoca en operaciones conceptuales y precisas sobre las relaciones (tablas) que componen la base de datos. Estas operaciones permiten seleccionar, proyectar, unir, combinar o eliminar información de manera rigurosa y predecible. El álgebra relacional es fundamental por varias razones: Base teórica de SQL: todas las consultas SQL se pueden expresar mediante operaciones de álgebra relacional. Optimización de consultas: los motores de bases de datos utilizan principios del álgebra relacional para mejorar el rendimiento de las consultas. Rigor matemático: asegura que los resultados sean consistentes y que las operaciones sobre las tablas respeten las reglas de integridad. Educación y análisis: facilita la comprensión de cómo se combinan y transforman los datos dentro de un RDBMS. Entre los operadores principales del álgebra relacional se encuentran la selección, proyección, unión, diferencia, producto cartesiano, join y renombrado, cada uno de ellos con un papel específico en la manipulación de los datos. En síntesis, el álgebra relacional constituye el fundamento lógico y conceptual de las bases de datos relacionales, proporcionando un marco para expresar consultas de manera precisa y estructurada antes de traducirlas a SQL u otro lenguaje de implementación.

## 2. Marco teórico

### Álgebra Relacional

- **Definición:** El álgebra relacional es un modelo teórico que define un conjunto de operaciones matemáticas aplicadas a las relaciones (tablas) en bases de datos. Estas operaciones permiten consultar, combinar y transformar la información de manera estructurada.
- **Operaciones fundamentales:**
  - **Selección ( $\sigma$ ):** Filtra tuplas que cumplen una condición específica.
  - **Proyección ( $\pi$ ):** Selecciona columnas de interés en una relación.
  - **Unión ( $\cup$ ):** Combina tuplas de dos relaciones con la misma estructura.
  - **Diferencia ( $-$ ):** Devuelve las tuplas que están en una relación pero no en otra.
  - **Producto cartesiano ( $\times$ ):** Combina todas las tuplas de dos relaciones.
  - **Renombramiento ( $\rho$ ):** Cambia el nombre de una relación o de sus atributos.
  - **Join ( $\bowtie$ ):** Une dos relaciones según una condición de igualdad o correspondencia.
- **Ejemplo:**
  - Seleccionar el nombre y apellido de los empleados:
    - Álgebra relacional:  $\pi$  nombre, apellido (Empleado).
    - SQL equivalente: `SELECT nombre, apellido FROM Empleado;`

### SQL (Structured Query Language)

- **Definición:** SQL es el lenguaje estándar utilizado para el manejo y consulta de bases de datos relacionales. Permite crear, modificar y consultar datos de manera estructurada.
- **Tipos de sentencias:**

- **DDL (Data Definition Language):** Incluye operaciones como CREATE, ALTER, DROP para definir y modificar la estructura de las bases de datos.
- **DML (Data Manipulation Language):** Operaciones sobre los datos como INSERT, SELECT, UPDATE, DELETE.
- **DCL (Data Control Language):** Permite gestionar permisos y seguridad con sentencias como GRANT y REVOKE.
- **TCL (Transaction Control Language):** Controla transacciones con comandos como COMMIT, ROLLBACK y SAVEPOINT.
- **Características:** Lenguaje declarativo, portabilidad, estandarización (ANSI SQL), y soporte para consultas complejas, subconsultas, funciones de agregación y procedimientos almacenados.

## MySQL

- **Definición:** MySQL es un sistema de gestión de bases de datos relacional (SGBD) de código abierto, basado en SQL, ampliamente utilizado en aplicaciones web y empresariales por su eficiencia, velocidad y confiabilidad.
- **Características principales:**
  - Compatibilidad con SQL estándar.
  - Soporte para integridad referencial con llaves primarias y foráneas.
  - Optimización de consultas mediante índices.
  - Manejo de transacciones con motores de almacenamiento como InnoDB.
  - Escalabilidad, replicación y alta disponibilidad.
  - Soporte para funciones, vistas, triggers y procedimientos almacenados.
- **Ejemplo de aplicación:** sitios web dinámicos (WordPress, Drupal), sistemas de gestión empresarial (ERP, CRM), y plataformas de comercio electrónico.

## Herramientas utilizadas

- **MySQL Server:** Motor principal que administra la base de datos y procesa las consultas SQL.
- **MySQL Workbench:** Entorno gráfico para diseñar esquemas, ejecutar consultas, modelar datos y administrar usuarios.
- **phpMyAdmin (opcional):** Herramienta web para gestionar bases de datos MySQL de forma visual.
- **Conectores de programación:** Integración con lenguajes como PHP, Python, Java o C++ para desarrollar aplicaciones que interactúan con la base de datos.

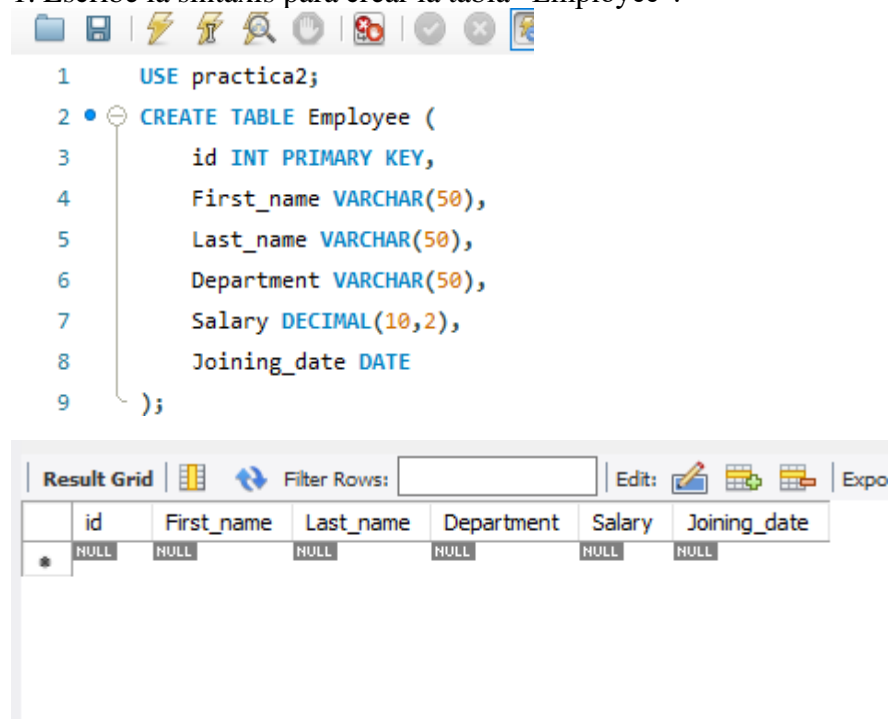
## 3. Metodología de diseño e implementación

Las herramientas que se utilizó en esta práctica fue el Sistema Gestor de Bases de Datos (SGBD) MySQL Workbench, en el cual ya hemos estado familiarizados y vamos a realizar el uso de algebra relacional.

## 4. Desarrollo (MySQL)

### EJERCICIOS.

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



```
1  USE practica2;
2  CREATE TABLE Employee (
3      id INT PRIMARY KEY,
4      First_name VARCHAR(50),
5      Last_name VARCHAR(50),
6      Department VARCHAR(50),
7      Salary DECIMAL(10,2),
8      Joining_date DATE
9  );
```

The Result Grid shows the structure of the newly created table 'Employee' with the following columns: id, First\_name, Last\_name, Department, Salary, and Joining\_date. All cells in the first row are NULL.

	id	First_name	Last_name	Department	Salary	Joining_date
*	NULL	NULL	NULL	NULL	NULL	NULL

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

```
INSERT INTO Employee (id, First_name, Last_name, Department, Salary, Joining_date) VALUES
(1, 'Jhon', 'Doe', 'HR', 50000, '2020-01-10'),
(2, 'Jane', 'Smith', 'IT', 60000, '2019-03-15'),
(3, 'Michael', 'Johnson', 'Finance', 70000, '2021-06-20'),
(4, 'Emily', 'Davis', 'IT', 55000, '2018-11-25'),
(5, 'William', 'Brown', 'HR', 52000, '2022-05-12'),
(6, 'Sophia', 'Wilson', 'Finance', 75000, '2017-08-30'),
(7, 'Daniel', 'Miller', 'IT', 58000, '2019-09-01');
```

The Result Grid displays the 7 records inserted into the 'Employee' table.

	id	First_name	Last_name	Department	Salary	Joining_date
▶	1	Jhon	Doe	HR	5000...	2020-01-10
	2	Jane	Smith	IT	6000...	2019-03-15
	3	Michael	Johnson	Finance	7000...	2021-06-20
	4	Emily	Davis	IT	5500...	2018-11-25
	5	William	Brown	HR	5200...	2022-05-12
	6	Sophia	Wilson	Finance	7500...	2017-08-30
	7	Daniel	Miller	IT	5800...	2019-09-01

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

```
CREATE TABLE Reward (  
    Employee_id INT,  
    Date_reward DATE,  
    Amount DECIMAL(10,2),  
    FOREIGN KEY (Employee_id) REFERENCES Employee(id)  
);
```

Employee_id	Date_reward	Amount
-------------	-------------	--------

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

```
INSERT INTO Reward VALUES(1, '2019-05-11', 1000), (3, '2022-01-20', 1500),  
(5, '2022-03-10', 1200),  
(7, '2021-07-05', 800);
```

	Employee_id	Date_reward	Amount
▶	1	2019-05-11	1000.00
	3	2022-01-20	1500.00
	5	2022-03-10	1200.00
	7	2021-07-05	800.00

5. Obtener todos los empleados.

```
28 • SELECT * FROM Employee;
```

29

Result Grid						
		Filter Rows:		Edit:		Export/I
	id	First_name	Last_name	Department	Salary	Joining_date
▶	1	Jhon	Doe	HR	50000.00	2020-01-10
	2	Jane	Smith	IT	60000.00	2019-03-15
	3	Michael	Johnson	Finance	70000.00	2021-06-20
	4	Emily	Davis	IT	55000.00	2018-11-25
	5	William	Brown	HR	52000.00	2022-05-12
	6	Sophia	Wilson	Finance	75000.00	2017-08-30
	7	Daniel	Miller	IT	58000.00	2019-09-01

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

```
28 • SELECT First_name, Last_name FROM Employee;  
29
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
First_name	Last_name			
Jhon	Doe			
Jane	Smith			
Michael	Johnson			
Emily	Davis			
William	Brown			
Sophia	Wilson			
Daniel	Miller			

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

```
29 • SELECT First_name AS "Nombre de empleado" FROM Employee;  
30
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Nombre de empleado				
Jhon				
Jane				
Michael				
Emily				
William				
Sophia				

8. Obtener todos los valores de la columna “Last\_name” en minúsculas.

```
29 • SELECT LOWER(Last_name) FROM Employee;  
30
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
LOWER(Last_name)				
doe				
smith				
johnson				
davis				
brown				
wilson				
miller				

9. Obtener todos los valores de la columna “Last\_name” en mayúsculas.

```
30 • SELECT upper(Last_name) FROM Employee;
31
```

<

Result Grid | Filter Rows:  | Export: | Wrap Cell Content

upper(Last_name)
DOE
SMITH
JOHNSON
DAVIS
BROWN
WILSON
MILLER

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

```
31 • SELECT DISTINCT Department FROM Employee;
32
```

<

Result Grid | Filter Rows:  | Export: | Wi

Department
HR
IT
Finance

11. Obtener los primeros 4 caracteres de todos los valores de la columna “First\_name”.

```
32 • SELECT SUBSTRING(First_name, 1, 4) FROM Employee;
33
34
```

<

Result Grid | Filter Rows:  | Export: | Wrap Cell Content:

SUBSTRING(First_name, 1, 4)
Jhon
Jane
Mich
Emil
Will
Soph
Dani

12. Obtener la posición de la letra “h” en el nombre del empleado con First\_name = “Jhon”.

```
33 • SELECT POSITION('h' IN First_name)
34 FROM Employee
35 WHERE First_name = 'Jhon';
36
37
```

Result Grid		Filter Rows:	Export:
POSITION('h' IN First_name)			
▶	2		

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

```
36 • SELECT RTRIM(First_name) FROM Employee;
37
38
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
RTRIM(First_name)				
▶	Jhon			
	Jane			
	Michael			
	Emily			
	William			
	Sophia			
	Daniel			

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

```
37 • SELECT LTRIM(First_name) FROM Employee;
38
39
```

Result Grid		Filter Rows:	Export:	Wrap Cell
LTRIM(First_name)				
▶	Jhon			
	Jane			
	Michael			
	Emily			
	William			
	Sophia			
	Daniel			



## 5. Conclusiones

La tarea permitió aplicar los fundamentos del álgebra relacional para representar consultas sobre bases de datos de manera formal y estructurada. A través de operaciones como proyección ( $\pi$ ), selección ( $\sigma$ ), renombramiento ( $\rho$ ) y eliminación de duplicados ( $\delta$ ), se logró traducir requerimientos comunes en expresiones precisas que reflejan cómo se manipulan los datos en un entorno relacional. Además, se identificaron limitaciones del álgebra relacional clásica frente a funciones más avanzadas como el manejo de cadenas o formatos de texto, lo que abre la puerta a explorar lenguajes más expresivos como SQL o extensiones del álgebra relacional. En resumen, esta práctica fortalece la comprensión de cómo se estructuran las consultas en bases de datos y prepara el terreno para diseñar sistemas más eficientes y robustos en el manejo de la información.

## Bibliografía

- | El manual de MySQL: <https://dev.mysql.com/doc/>
- | El libro: Elmasri, R. & Navathe, S. (2015). *Fundamentals of Database Systems*. Pearson.