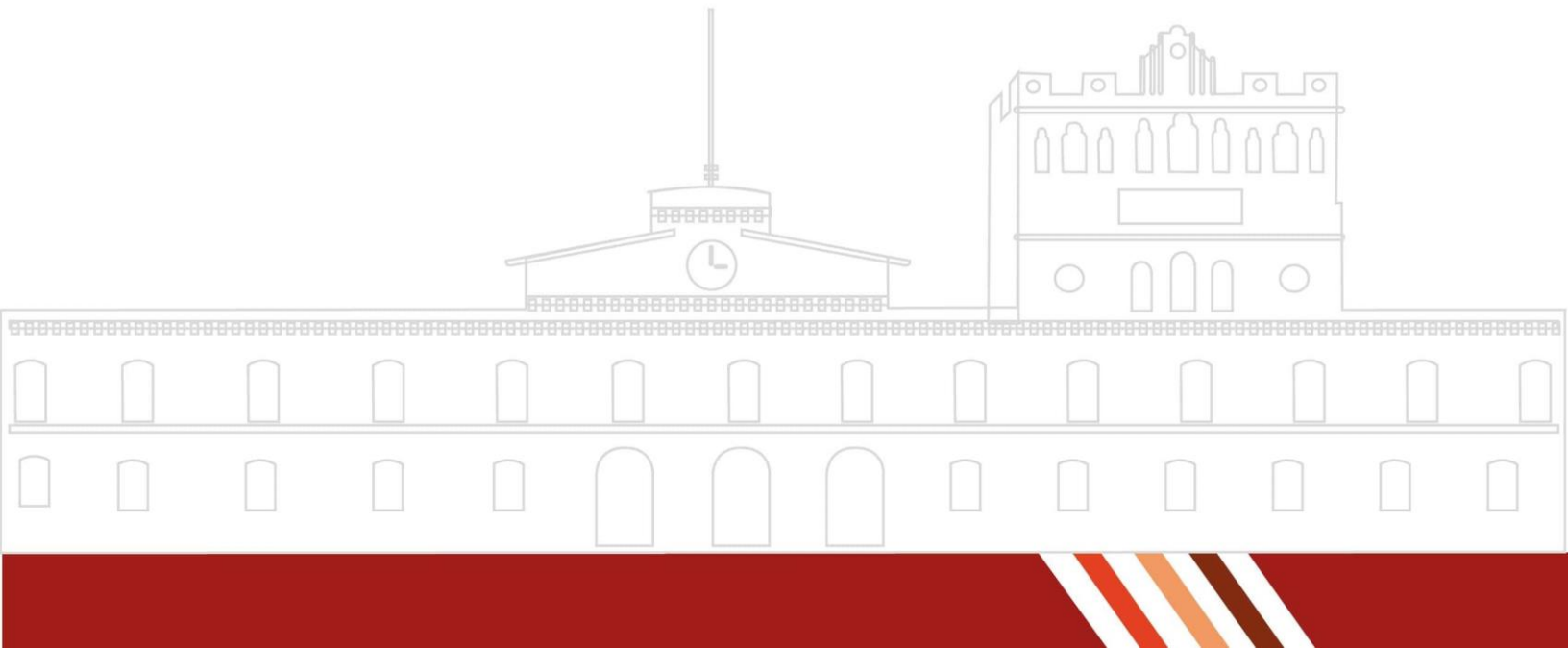


# REPORTE DE PRÁCTICA

## ÁLGEBRA RELACIONAL Y BASES DE DATOS DISTRIBUIDAS

**ALUMNO: Amuchategui Flores Braulio**

**Dr. Eduardo Cornejo-Velázquez**



## Introducción

El álgebra relacional es una piedra angular en el campo de las bases de datos relacionales, proporcionando un conjunto de operaciones fundamentales que permiten la manipulación y consulta de datos estructurados en tablas. Desarrollado como un marco teórico por Edgar F. Codd en la década de 1970, el álgebra relacional se basa en conceptos matemáticos que aseguran la precisión, consistencia y eficiencia en la gestión de datos. A través de operaciones como la selección, proyección, unión, intersección, diferencia, y producto cartesiano, es posible realizar diversas transformaciones y combinaciones de datos para obtener la información deseada de una base de datos. Estas operaciones son la base sobre la cual se construyen lenguajes de consulta como SQL, lo que permite a los usuarios interactuar con los datos de manera intuitiva y efectiva. La relevancia del álgebra relacional radica en su capacidad para representar consultas complejas de forma precisa y optimizable por los sistemas de gestión de bases de datos (SGBD). Además, al ser un modelo declarativo, el álgebra relacional permite a los usuarios especificar *qué* quieren obtener sin necesidad de detallar *cómo* se debe ejecutar la consulta, delegando la optimización y ejecución a los SGBD. Esto no solo facilita la creación de consultas eficientes, sino que también asegura que las operaciones se realicen de manera consistente y predecible, independientemente del tamaño o la complejidad de la base de datos. El álgebra relacional es un componente esencial en el diseño y operación de bases de datos relacionales, proporcionando un lenguaje universal y estandarizado para la manipulación de datos. Su importancia trasciende el ámbito teórico, impactando directamente en la forma en que las organizaciones almacenan, consultan y utilizan la información, lo que lo convierte en un tema de estudio crucial para cualquier persona interesada en el campo de las ciencias de la computación y la gestión de datos.

## Marco Teórico

La practica junto con sus ejercicios se enfoca en operaciones más avanzadas de SQL, como la manipulación de cadenas, el uso de funciones de agregación, y la ordenación de resultados.

- Manipulación de Cadenas en SQL: SQL proporciona varias funciones para manipular cadenas de texto, como LENGTH, REPLACE, CONCAT, UPPER, y LOWER, que permiten modificar y analizar los datos textuales de las tablas.
- Ordenación y Filtrado: El uso de cláusulas como ORDER BY y WHERE permite ordenar y filtrar los resultados de una consulta, lo cual es crucial para obtener información relevante y específica de la base de datos.
- Funciones de Fecha y Hora: Las funciones como YEAR, MONTH, y DAY permiten extraer partes específicas de una fecha

## Herramientas Empleadas

Para esta práctica, se utilizó **MySQL Workbench**, una herramienta de interfaz gráfica de usuario (GUI) diseñada específicamente para MySQL. Esta herramienta proporciona un entorno integrado para el diseño, modelado, generación y administración de bases de datos MySQL.

## Desarrollo de la Práctica

### Creación de Tablas e Inserción de Datos

#### 4.1.1 Tabla "Employee"

```
CREATE TABLE Employee (  
    Employee id INT PRIMARY KEY,  
    First name VARCHAR(50),  
    Last name VARCHAR(50),  
    Salary DECIMAL(10, 2),  
    Joining_date DATE,  
    Department VARCHAR(50)  
);
```

```
INSERT INTO Employee (Employee id, 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');
```

#### 4.1.2 Tabla "Reward"

```
CREATE TABLE Reward (  
    Employee_ref_id INT,  
    date_reward DATE,  
    amount DECIMAL(10 , 2),  
    FOREIGN KEY ( Employee_ref_id ) REFERENCES Employee ( Employee_id )  
);
```

## Consultas SQL

### 4.2.1 Consultas Básicas

-- Obtener el tamaño del texto en todos los valores de la columna First\_name

**SELECT** LENGTH(First\_name) **FROM** Employee;

lgebra Relacional:  $\pi_{\{LENGTH(First\_name)\}}(Employee)$

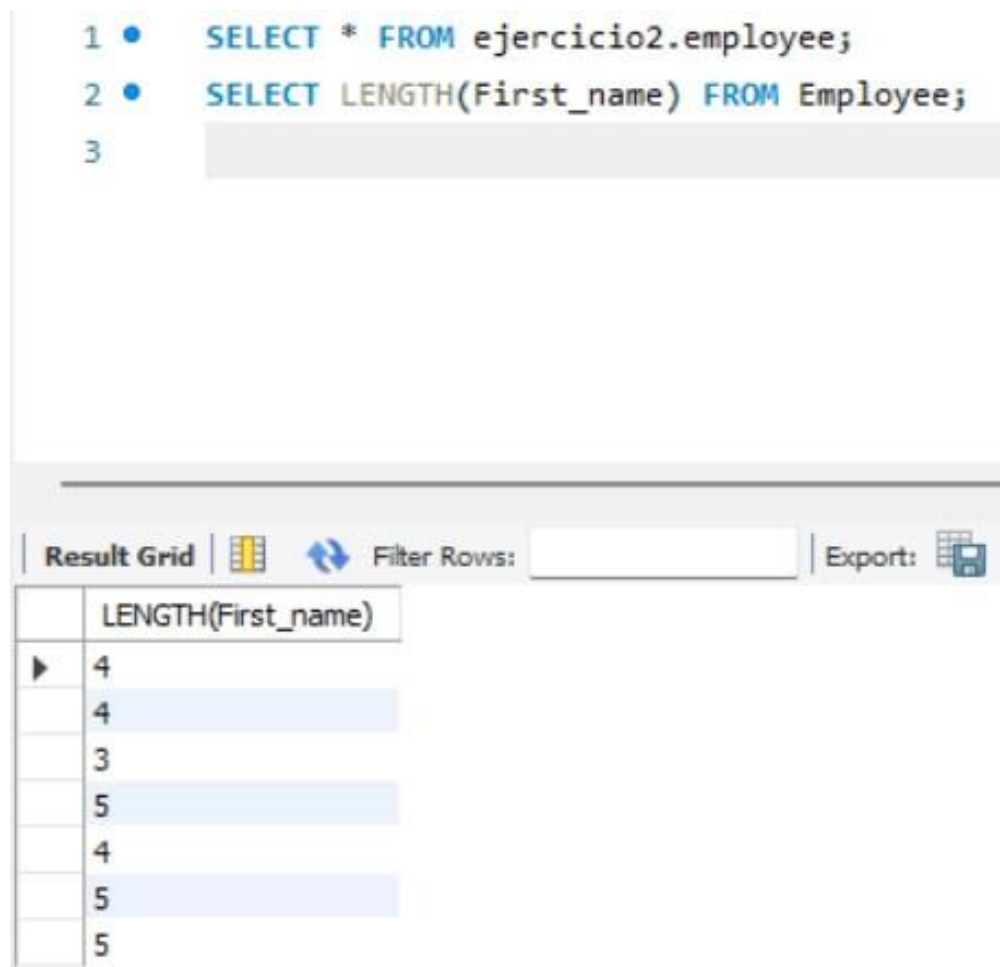


Figure 1: Se Obtiene el tamaño del texto en todos los valores de la columna "First\_name"

-- Obtener el nombre de todos los empleados después de reemplazar o con \#  
**SELECT** REPLACE(First\_name , 'o' , '#') **FROM** Employee;  
 lgebra Relacional : \_ {LENGTH(First\_name)}(Employee)

```

1 • SELECT * FROM ejercicio2.employee;
2 • SELECT LENGTH(First_name) FROM Employee;
3 • SELECT REPLACE(First_name, 'o', '#') FROM Employee;

```

	REPLACE(First_name, 'o', '#')
▶	Jh#n
	Jane
	B#b
	Alice
	Alex
	Maria
	Chris

Figure 2: Resultado de la consulta para obtener el nombre de todos los empleados después de reemplazar 'o' con '#'



-- Obtener el nombre y apellido de todos los empleados en una sola columna separados por '  
**SELECT** CONCAT(First\_name, ' ', Last\_name) **AS** Full\_name **FROM** Employee;  
 lgebra Relacional:  $\rho_{\{CONCAT(First\_name, ' ', Last\_name)\}}(Employee)$

```

1 • SELECT * FROM ejercicio2.employee;
2 • SELECT LENGTH(First_name) FROM Employee;
3 • SELECT REPLACE(First_name, 'o', '#') FROM Employee;
4 • SELECT CONCAT(First_name, ' ', Last_name) AS Full_name FROM Employee;
5

```

---

Result Grid Filter Rows:  Export: Wrap Cell Content:

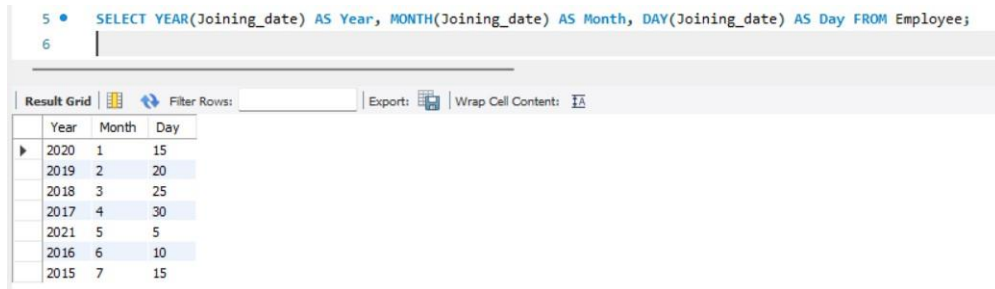
	Full_name
▶	Jhon_Doe
	Jane_Smith
	Bob_Johnson
	Alice_Williams
	Alex_Brown
	Maria_Davis
	Chris_Miller

Figure 3: Resultado de la consulta nombre y apellido de todos los empleados en una sola columna separados por ' '.

#### 4.2.2 Manipulación de Cadenas

-- Obtener el año, mes y día de la columna `Joining_date`

**SELECT YEAR(Joining\_date) AS Year, MONTH(Joining\_date) AS Month, DAY(Joining\_date) AS Day FROM Employee;**  
Algebra Relacional:  $\{ \text{YEAR}(\text{Joining\_date}), \text{MONTH}(\text{Joining\_date}), \text{DAY}(\text{Joining\_date}) \}$  (Empleado)



	Year	Month	Day
▶	2020	1	15
	2019	2	20
	2018	3	25
	2017	4	30
	2021	5	5
	2016	6	10
	2015	7	15

Figure 4: Resultado de la consulta para obtener el año, mes y día de la columna

-- Obtener todos los empleados en orden ascendente por nombre  
**SELECT \* FROM Employee ORDER BY First\_name ASC;**  
 lgebra Relacional :  $\_ \{ASC(First\_name)\}(Employee)$

6 • **SELECT \* FROM Employee ORDER BY First\_name ASC;**

Result Grid | | Filter Rows:  | Edit: | Export/I

	EmployeeID	First_name	Last_name	Department	Joining_date	Salary
▶	5	Alex	Brown	IT	2021-05-05	5200.00
	4	Alice	Williams	Marketing	2017-04-30	5500.00
	3	Bob	Johnson	Finance	2018-03-25	6000.00
	7	Chris	Miller	IT	2015-07-15	6100.00
	2	Jane	Smith	HR	2019-02-20	4500.00
	1	Jhon	Doe	IT	2020-01-15	5000.00
	6	Maria	Davis	Sales	2016-06-10	4800.00

Figure 5: Resultado de la consulta para orden ascendente por nombre

-- Obtener todos los empleados en orden descendente por nombre

**SELECT \* FROM Employee ORDER BY First\_name DESC;**

lgebra Relacional :  $\{DESC(First\_name)\}(Employee)$



7 • **SELECT \* FROM Employee ORDER BY First\_name DESC;**

Result Grid | Filter Rows: | Edit: | Export/1

	EmployeeID	First_name	Last_name	Department	Joining_date	Salary
▶	6	Maria	Davis	Sales	2016-06-10	4800.00
	1	Jhon	Doe	IT	2020-01-15	5000.00
	2	Jane	Smith	HR	2019-02-20	4500.00
	7	Chris	Miller	IT	2015-07-15	6100.00
	3	Bob	Johnson	Finance	2018-03-25	6000.00
	4	Alice	Williams	Marketing	2017-04-30	5500.00
	5	Alex	Brown	IT	2021-05-05	5200.00

Figure 6: Resultado de la consulta para orden descendente por nombre

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

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

lgebra Relacional : - {ASC(First\_name), DESC(Salary)}(Employee)

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

EmployeeID	First_name	Last_name	Department	Joining_date	Salary
5	Alex	Brown	IT	2021-05-05	5200.00
4	Alice	Williams	Marketing	2017-04-30	5500.00
3	Bob	Johnson	Finance	2018-03-25	6000.00
7	Chris	Miller	IT	2015-07-15	6100.00
2	Jane	Smith	HR	2019-02-20	4500.00
1	Jhon	Doe	IT	2020-01-15	5000.00
6	Maria	Davis	Sales	2016-06-10	4800.00

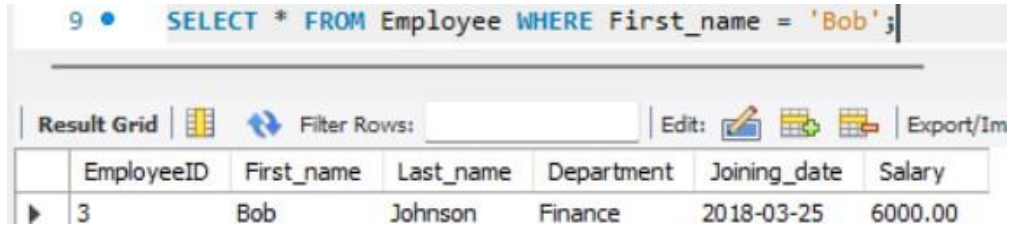
Figure 7: Resultado de la consulta para orden ascendente por nombre y en orden descendente por salario

### 4.2.3 Consultas Avanzadas

-- Obtener todos los empleados con el nombre Bob

**SELECT \* FROM Employee WHERE First\_name = 'Bob';**

lgebra Relacional :  $\sigma_{\text{First\_name} = \text{'Bob'}}(\text{Employee})$



The screenshot shows a database query interface. At the top, a text box contains the SQL query: `SELECT * FROM Employee WHERE First_name = 'Bob';`. Below the text box is a toolbar with icons for 'Result Grid', 'Filter Rows', 'Edit', and 'Export/Import'. The 'Result Grid' icon is selected. Below the toolbar is a table with the following data:

	EmployeeID	First_name	Last_name	Department	Joining_date	Salary
▶	3	Bob	Johnson	Finance	2018-03-25	6000.00

Figure 8: Resultado de la consulta para empleados con el nombre "Bob"

-- Obtener todos los empleados con el nombre Bob o Alex  
**SELECT \* FROM Employee WHERE First\_name IN ('Bob', 'Alex');**  
 lgebra Relacional:  $\{ \text{First\_name} = \text{'Bob'} \} \cup \{ \text{First\_name} = \text{'Alex'} \} (\text{Employee})$

10 • **SELECT \* FROM Employee WHERE First\_name IN ('Bob', 'Alex');**

Result Grid | Filter Rows: | Edit: | Export/Import:

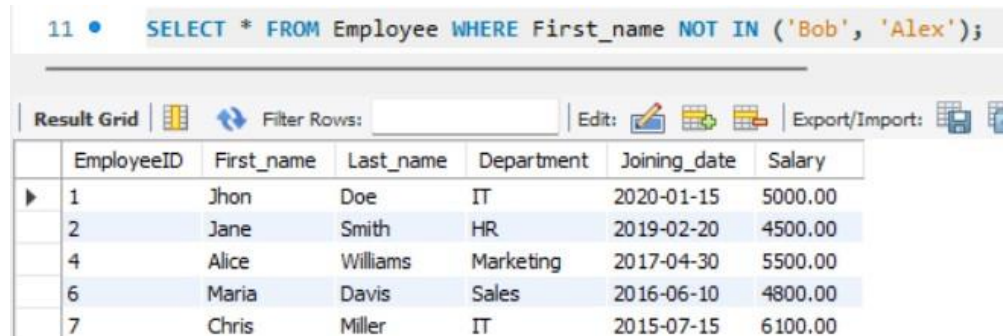
	EmployeeID	First_name	Last_name	Department	Joining_date	Salary
▶	3	Bob	Johnson	Finance	2018-03-25	6000.00
	5	Alex	Brown	IT	2021-05-05	5200.00

Figure 9: Resultado de la consulta empleados con el nombre “Bob” o “Alex”

-- Obtener todos los empleados que no tengan el nombre Bob o Alex

**SELECT \* FROM Employee WHERE First\_name NOT IN ('Bob', 'Alex');**

lgebra Relacional: - { (First\_name = 'Bob' First\_name = 'Alex') }(Employee)



11 • SELECT \* FROM Employee WHERE First\_name NOT IN ('Bob', 'Alex');

Result Grid | Filter Rows: | Edit: | Export/Import:

	EmployeeID	First_name	Last_name	Department	Joining_date	Salary
▶	1	Jhon	Doe	IT	2020-01-15	5000.00
	2	Jane	Smith	HR	2019-02-20	4500.00
	4	Alice	Williams	Marketing	2017-04-30	5500.00
	6	Maria	Davis	Sales	2016-06-10	4800.00
	7	Chris	Miller	IT	2015-07-15	6100.00

Figure 10: Resultado de la consulta con empleados que no tengan el nombre “Bob” o “Alex”

**¿Qué es una inyección SQL? Descripción:** Una inyección SQL es una vulnerabilidad de seguridad que permite a un atacante interferir con las consultas a una base de datos. Este tipo de ataque se produce cuando se inserta código SQL malicioso en una consulta, lo que puede permitir a los atacantes acceder, modificar o eliminar datos sin autorización. Para prevenir las inyecciones SQL, es esencial validar y escapar correctamente las entradas del usuario, y utilizar consultas preparadas con parámetros.



## Conclusiones

A través de estos ejercicios, se refuerza la capacidad de realizar manipulaciones más complejas en las bases de datos. El manejo adecuado de las funciones de manipulación de cadenas, fechas, y la ordenación avanzada permiten obtener información más precisa y útil de la base de datos, lo cual es esencial para la toma de decisiones basada en datos.

## Referencias Bibliográficas

- Costal Costa, D. (2007). *El modelo relacional y el álgebra relacional*.
- Definición Wiki. (2023). *Definición de Álgebra Relacional en MySQL: según Autor, Ejemplos, qué es, Concepto, Significado*.
- Universitat Jaume I. (2018). *BASES DE DATOS (IG18 Semipresencial) El Modelo Relacional Álgebra*.
- Platzi. (2023). *Álgebra relacional y Bases de Datos*.