

# REPORTE DE PRÁCTICA NO. ?

## ÁLGEBRA RELACIONAL Y BASES DE DATOS DISTRIBUIDAS

**ALUMNO: Alvarado Moreno Juan Diego**

**Dr. Eduardo Cornejo-Velázquez**



## Introducción

La presente práctica se centra en la aplicación del **álgebra relacional** en el contexto de **bases de datos distribuidas**. El álgebra relacional, un conjunto de operaciones matemáticas fundamentales, es esencial para manipular y consultar datos en **bases de datos relacionales**. Estas operaciones son la base teórica de lenguajes de consulta como **SQL** y son cruciales para la **fragmentación y distribución de datos** en sistemas distribuidos, asegurando la **consistencia y disponibilidad** de la información.

Las operaciones del álgebra relacional, incluyendo **selección, proyección, unión, intersección y diferencia**, permiten a los sistemas distribuidos ejecutar consultas de manera **paralela y optimizada**, mejorando así el rendimiento y la escalabilidad del sistema.

## 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;
```

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

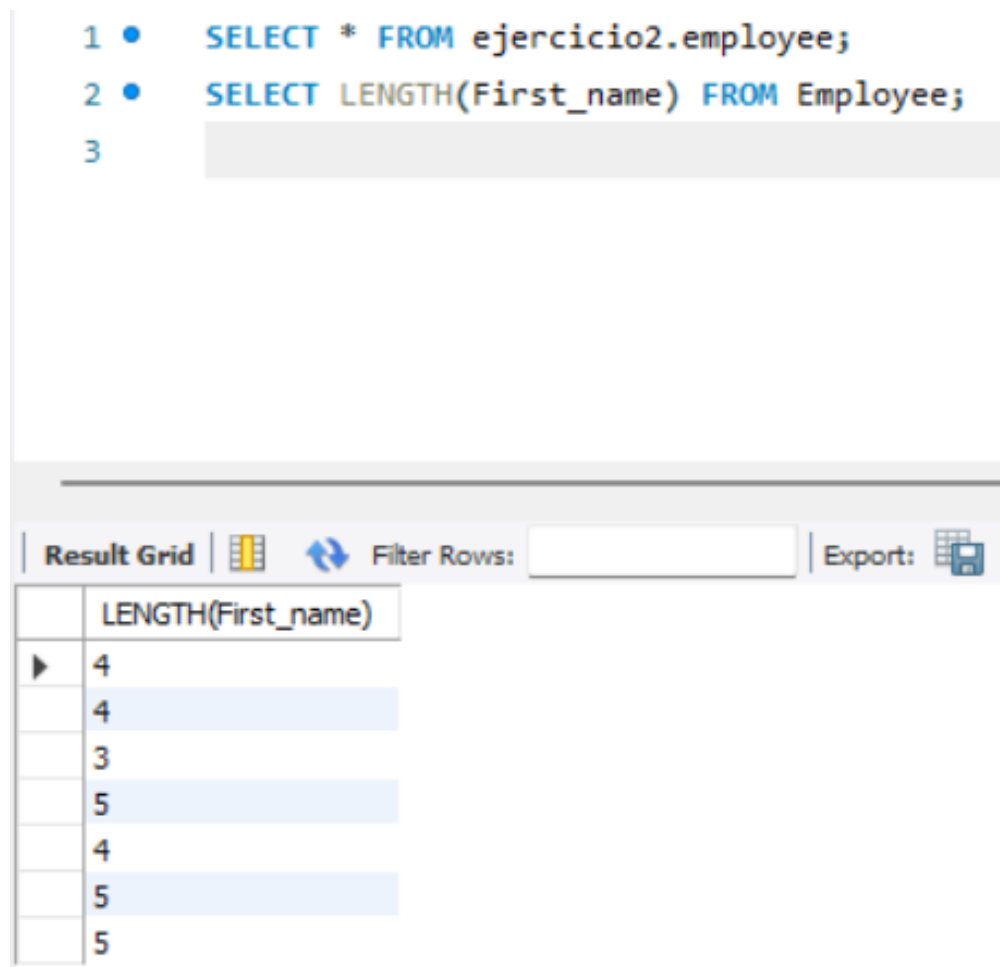


Figure 1: Se Obtiene el tamaño del texto en todos los valores de la columna “First\_name”

— Obtener el nombre de todos los empleados después de reemplazar 'o' con '#'

**SELECT** REPLACE(First\_name, 'o', '#') **FROM** Employee;

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

```

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

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content
	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: - {CONCAT(First\_name, ' ', Last\_name)}(Employee)

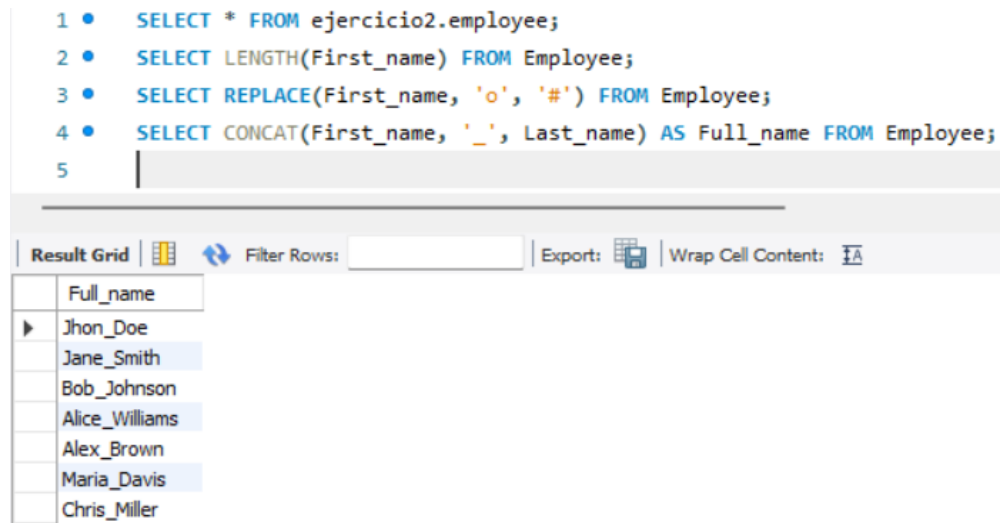
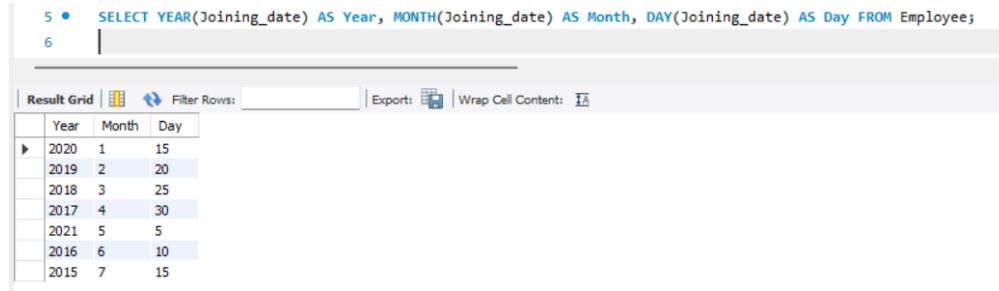


Figure 3: Resultado de la consulta nombre y apellido de todos los empleados en una sola columna separados por ' '.

#### 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;**  
álgebra Relacional:  $\pi_{\{YEAR(Joining\_date), MONTH(Joining\_date), DAY(Joining\_date)\}}(Employee)$



The screenshot shows a SQL query editor with the following query:

```
5 • SELECT YEAR(Joining_date) AS Year, MONTH(Joining_date) AS Month, DAY(Joining_date) AS Day FROM Employee;
6 |
```

Below the query editor, there is a "Result Grid" section with a table of results:

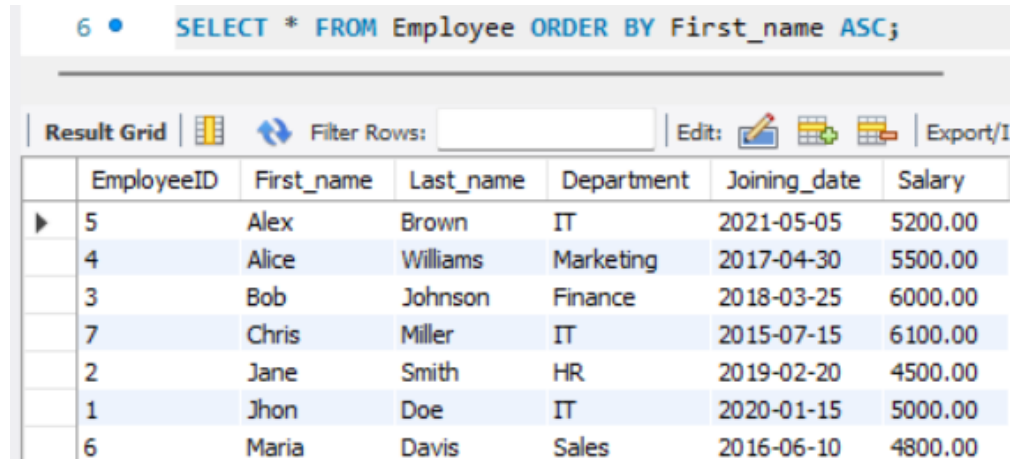
	Year	Month	Day
▶	2020	1	15
	2019	2	20
	2018	3	25
	2017	4	30
	2021	5	5
	2016	6	10
	2015	7	15

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)



The screenshot shows a database interface with a query editor at the top containing the SQL statement: `SELECT * FROM Employee ORDER BY First_name ASC;`. Below the editor is a toolbar with options like 'Result Grid', 'Filter Rows', 'Edit', and 'Export'. The main area displays a table of employee data sorted by first name in ascending order. The table has columns for EmployeeID, First\_name, Last\_name, Department, Joining\_date, and Salary. The rows are: Alex Brown (IT), Alice Williams (Marketing), Bob Johnson (Finance), Chris Miller (IT), Jane Smith (HR), Jhon Doe (IT), and Maria Davis (Sales).

	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;**

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

	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

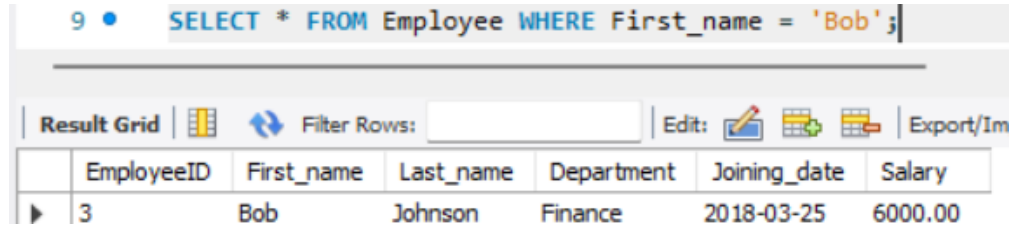
Figure 7: Resultado de la consulta para orden ascendente por nombre y en orden descendente por salario

### 4.2.3 Consultas Avanzadas

— *Obtener todos los empleados con el nombre Bob*

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

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



9 • `SELECT * FROM Employee WHERE First_name = 'Bob';`

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

	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:  $\pi_{\{First\_name = 'Bob' \vee First\_name = 'Alex'\}}(Employee)$

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

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

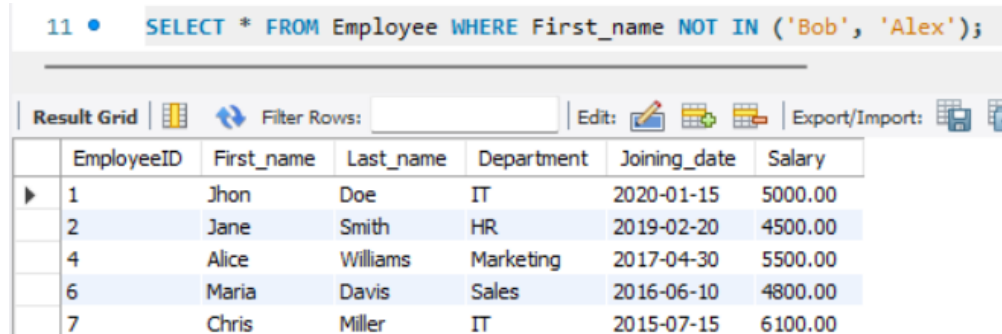
	EmployeeID	First_name	Last_name	Department	Joining_date	Salary
▶	3	Bob	Johnson	Finance	2018-03-25	6000.00
	5	Alex	Brown	IT	2021-05-05	5200.00

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' );**

álgebra Relacional:  $- \{ (First\_name = 'Bob' \vee First\_name = 'Alex') \} (Employee)$



	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*.