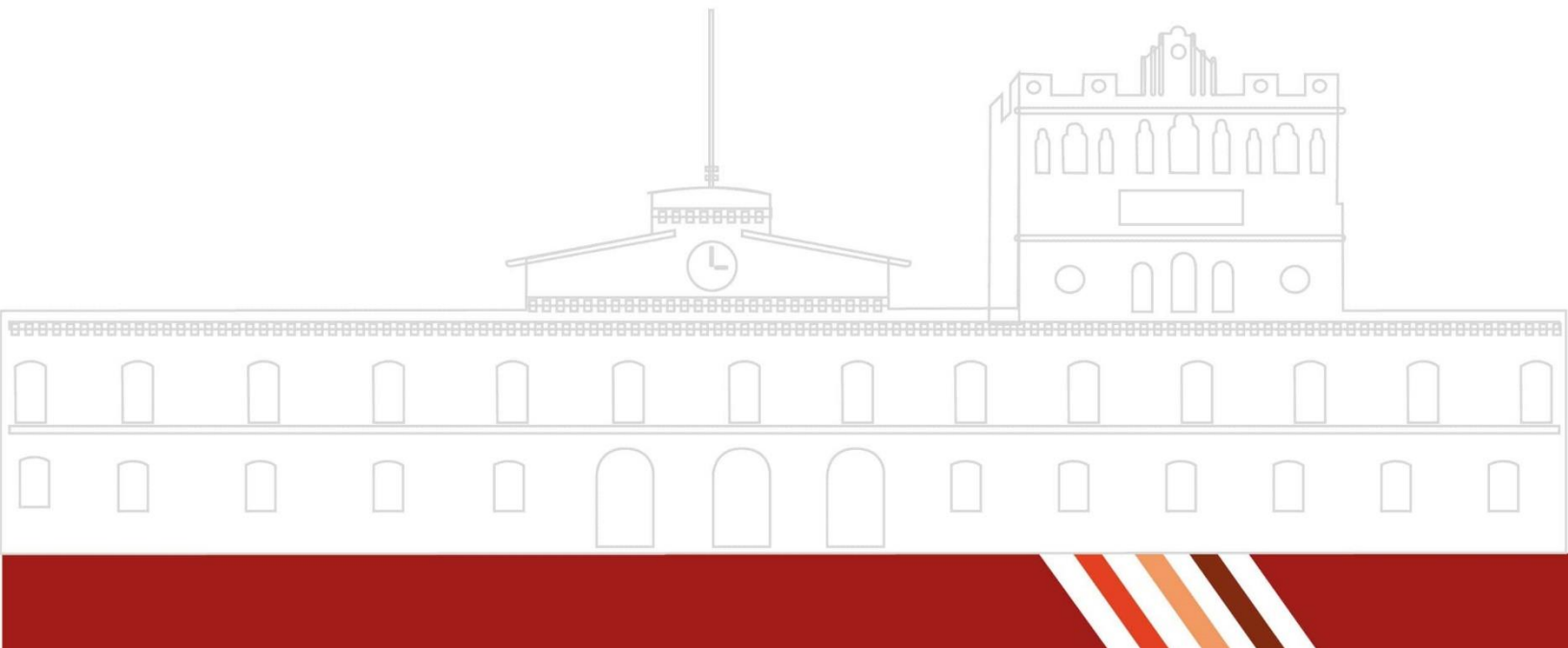


REPORTE DE PRÁCTICA NO. 1.3

1.3. Práctica. Álgebra relacional y SQL (2)

ALUMNO: Sebastian Trejo Muñoz

Dr. Eduardo Cornejo-Velázquez



1. Introducción

En esta práctica, exploraremos el manejo y manipulación de bases de datos a través de ejercicios prácticos centrados en las tablas **Employee** y **Reward**. El objetivo es aplicar conocimientos de **SQL** para resolver una serie de ejercicios que ponen a prueba la capacidad de consultar, filtrar y modificar datos en estas tablas.

Cada ejercicio requerirá la redacción de una sentencia **SQL** que resuelva el problema planteado. Para demostrar la comprensión y correcta aplicación de los conceptos, se deberá incluir una captura de pantalla que muestre tanto la sentencia **SQL** utilizada como el resultado de su ejecución. Esta documentación permitirá verificar que la consulta fue ejecutada con éxito y que el resultado cumple con los requerimientos del ejercicio.

2. Marco teórico

Tablas en Bases de Datos Relacionales:

En un sistema de bases de datos relacional, la información se organiza en tablas, que consisten en filas y columnas. Cada tabla representa una entidad específica, y cada fila en la tabla representa un registro único de esa entidad. Las columnas, por su parte, definen los atributos o características de la entidad. Por ejemplo, la tabla `Employee` podría contener columnas como `EmployeeID`, `Name`, y `Position`, mientras que la tabla `Reward` podría incluir `RewardID`, `EmployeeID`, y `RewardDate`.

Consultas SQL:

SQL permite a los usuarios interactuar con la base de datos mediante comandos que pueden consultar (`SELECT`), insertar (`INSERT`), actualizar (`UPDATE`), y eliminar (`DELETE`) datos. Las consultas SQL son el pilar de la interacción con los datos, permitiendo filtrar, ordenar, agrupar, y sumarizar la información almacenada. En el contexto de las tablas `Employee` y `Reward`, las consultas SQL se utilizan para extraer información específica, como la relación entre empleados y las recompensas recibidas.

Relaciones entre Tablas:

En un esquema relacional, las tablas pueden estar vinculadas entre sí mediante claves foráneas (`Foreign Keys`), lo que permite establecer relaciones entre diferentes conjuntos de datos. Por ejemplo, la relación entre la tabla `Employee` y `Reward` puede estar definida a través de la columna `EmployeeID`, donde cada recompensa está asociada con un empleado específico. Estas relaciones son fundamentales para realizar consultas que combinen información de múltiples tablas.

Filtrado y Manipulación de Datos:

A través de las consultas SQL, es posible aplicar filtros y condiciones que permiten extraer conjuntos de datos específicos basados en ciertos criterios. Operadores como `WHERE`, `AND`, `OR`, y `LIKE` son comunes en el filtrado de datos, permitiendo la selección de registros que cumplan con ciertas condiciones.

3. Herramientas utilizadas

1. MySQL Server

MySQL Server es un sistema de gestión de bases de datos relacional de código abierto, usado para almacenar y manejar datos. Permite realizar consultas **SQL**, gestionar la seguridad y control de acceso, y es conocido por su velocidad y escalabilidad. Es popular en el desarrollo web para manejar bases de datos en aplicaciones como sitios de comercio electrónico y sistemas de gestión de contenido.

2. Overleaf

Overleaf es una plataforma en línea para crear y colaborar en documentos **LaTeX**. Permite la edición colaborativa en tiempo real y la compilación automática del documento. Ofrece plantillas y se integra con herramientas de referencias bibliográficas y control de versiones. Es ampliamente usado en entornos académicos y científicos para escribir artículos, tesis y presentaciones de alta calidad.



4. Desarrollo

Sentencias SQL

1. Obtener el tamaño del texto en todos los valores de la columna “First_name”.

```
SELECT First_name, LENGTH(First_name) FROM Employee;
```

22 • `SELECT First_name, LENGTH(First_name) FROM Employee;`



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

First_name	LENGTH(First_name)
Bob	3
Jerry	5
Philip	6
John	4
Michael	7
Alex	4
Yohan	5

2. Obtener el nombre de todos los empleados después de reemplazar ‘o’ con ‘#’.

```
SELECT REPLACE(First_name, 'o', '#') FROM Employee;
```

23 • `SELECT REPLACE(First_name, 'o', '#') FROM Employee;`

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

REPLACE(First_name, 'o', '#')
B#b
Jerry
Philip
J#hn
Michael
Alex
Y#han

3. Obtener el nombre y apellido de todos los empleados en una sola columna separados por “_”.

```
SELECT CONCAT(First_name, '_', Last_name) FROM Employee;
```

24 • `SELECT CONCAT(First_name, '_', Last_name) FROM Employee;`

CONCAT(First_name, '_', Last_name)
Bob_Kinto
Jerry_Kansxo
Philip_Jose
John_Abraham
Michael_Mathew
Alex_chreketo
Yohan_Soso

4. Obtener el año, mes y día de la columna "Joining_date".

`SELECT YEAR(Joining_date), MONTH(Joining_date), DAY(Joining_date) FROM Employee;`

25 • `SELECT YEAR(Joining_date), MONTH(Joining_date), DAY(Joining_date) FROM Employee;`

YEAR(Joining_date)	MONTH(Joining_date)	DAY(Joining_date)
2019	1	20
2019	1	15
2019	2	5
2019	2	25
2019	2	18
2019	5	10
2019	6	20

5. Obtener todos los empleados en orden ascendente por nombre.

`SELECT * FROM Employee ORDER BY First_name ASC;`

26 • `SELECT * FROM Employee ORDER BY First_name ASC;`

Employee_id	First_name	Last_name	Salary	Joining_date	Departament
6	Alex	chreketo	4000000	2019-05-10	IT
1	Bob	Kinto	1000000	2019-01-20	Finance
2	Jerry	Kansxo	6000000	2019-01-15	IT
4	John	Abraham	2000000	2019-02-25	Insurance
5	Michael	Mathew	2200000	2019-02-18	Finance
3	Philip	Jose	8900000	2019-02-05	Banking
7	Yohan	Soso	1230000	2019-06-20	Banking
NULL	NULL	NULL	NULL	NULL	NULL

6. Obtener todos los empleados en orden descendente por nombre.

SELECT * FROM Employee ORDER BY First_name DESC;

27 • SELECT * FROM Employee ORDER BY First_name DESC;

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

	Employee_id	First_name	Last_name	Salary	Joining_date	Departament
▶	7	Yohan	Soso	1230000	2019-06-20	Banking
	3	Philip	Jose	8900000	2019-02-05	Banking
	5	Michael	Mathew	2200000	2019-02-18	Finance
	4	John	Abraham	2000000	2019-02-25	Insurance
	2	Jerry	Kansxo	6000000	2019-01-15	IT
	1	Bob	Kinto	1000000	2019-01-20	Finance
	6	Alex	chreketo	4000000	2019-05-10	IT
*	NULL	NULL	NULL	NULL	NULL	NULL

7. 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;

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

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

	Employee_id	First_name	Last_name	Salary	Joining_date	Departament
	6	Alex	chreketo	4000000	2019-05-10	IT
	1	Bob	Kinto	1000000	2019-01-20	Finance
	2	Jerry	Kansxo	6000000	2019-01-15	IT
	4	John	Abraham	2000000	2019-02-25	Insurance
	5	Michael	Mathew	2200000	2019-02-18	Finance
	3	Philip	Jose	8900000	2019-02-05	Banking
	7	Yohan	Soso	1230000	2019-06-20	Banking
*	NULL	NULL	NULL	NULL	NULL	NULL

8. Obtener todos los empleados con el nombre "Bob".

SELECT * FROM Employee WHERE First_name = 'Bob';

29 • SELECT * FROM Employee WHERE First_name = 'Bob';







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

	Employee_id	First_name	Last_name	Salary	Joining_date	Departament
▶	1	Bob	Kinto	1000000	2019-01-20	Finance
*	NULL	NULL	NULL	NULL	NULL	NULL

9. Obtener todos los empleados con el nombre “Bob” o “Alex”.

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

30 • `SELECT * FROM Employee WHERE First_name IN ('Bob','Alex');`

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

	Employee_id	First_name	Last_name	Salary	Joining_date	Departament
▶	1	Bob	Kinto	1000000	2019-01-20	Finance
	6	Alex	chreketo	4000000	2019-05-10	IT
•	NULL	NULL	NULL	NULL	NULL	NULL




10. Obtener todos los empleados que no tengan el nombre “Bob” o “Alex”.



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

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

Result Grid

Filter Rows:

Edit:   

Export/Import:  

Employee_id	First_name	Last_name	Salary	Joining_date	Department
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	2019-02-18	Finance
7	Yohan	Soso	1230000	2019-06-20	Banking
NULL	NULL	NULL	NULL	NULL	NULL

11. ¿Qué es una inyección SQL?

Es un tipo de vulnerabilidad de seguridad que permite a un atacante interferir con las consultas que una aplicación hace a su base de datos. Ocurre cuando un atacante inserta o “inyecta” código SQL malicioso en una entrada de la aplicación, como un campo de formulario, con el objetivo de manipular la base de datos y acceder a información no autorizada

Algebra relacional

5. Obtener todos los empleados en orden ascendente por nombre.

$\tau_{First_name}(Employee)$

6. Obtener todos los empleados en orden descendente por nombre.

$\tau_{First_name\downarrow}(Employee)$

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

$\tau_{First_name,Salary\downarrow}(Employee)$

8. Obtener todos los empleados con el nombre “Bob”.

$\sigma_{First_name='Bob'}(Employee)$

9. Obtener todos los empleados con el nombre “Bob” o “Alex”.

$\sigma_{First_name='Bob'\vee First_name='Alex'}(Employee)$

10. Obtener todos los empleados que no tengan el nombre “Bob” o “Alex”.

$\sigma_{First_name\neq'Bob'\wedge First_name\neq'Alex'}(Employee)$

5. Conclusión

En esta práctica, se llevó a cabo una serie de ejercicios utilizando SQL para la creación y manipulación de bases de datos relacionales. A través de la creación de las tablas Employee y Reward, se estableció una estructura básica para almacenar información relevante sobre empleados y sus recompensas. Posteriormente, se realizaron varias consultas que ilustraron cómo extraer y manipular datos de estas tablas, desde la simple recuperación de todos los registros hasta operaciones más específicas como la obtención de valores únicos y la transformación de texto.

Esta práctica no solo reforzó los conceptos fundamentales del manejo de bases de datos relacionales, sino que también demostró la flexibilidad y el poder de SQL como herramienta para manejar grandes volúmenes de datos de manera eficiente. Al comprender y aplicar estas técnicas, se establecen las bases para desarrollar habilidades avanzadas en la gestión de datos, esenciales en entornos de trabajo que requieren un análisis de información preciso y confiable.