REPORTE DE PRÁCTICA NO. 2

Práctica. Álgebra relacional y SQL (1)

ALUMNO: Jose Angel Castro Paredes
Dr. Eduardo Cornejo-Velázquez

1. Introducción

En la era de la información, el manejo eficiente de datos es fundamental para el desarrollo de cualquier sistema informático. Las bases de datos relacionales han demostrado ser una solución sólida para almacenar, organizar y recuperar información de manera estructurada. Para comprender su funcionamiento, es esencial conocer dos pilares fundamentales: el Álgebra Relacional y SQL.
El álgebra relacional proporciona la base teórica para la manipulación de datos en bases de datos relacionales, permitiendo realizar operaciones como selección, proyección y unión. Por otro lado, SQL, es el lenguaje estándar utilizado para interactuar con bases de datos, mientras que MySQL es uno de los sistemas de gestión más utilizados para implementar estas operaciones en la práctica.

2. Marco teórico

Algebra relacional

El álgebra relacional es un conjunto de operaciones matemáticas que permiten la manipulación y consulta de bases de datos relacionales. Fue propuesta por Edgar F. Codd en 1970 como la base teórica de los sistemas de bases de datos relacionales.

Sql

SQL es un lenguaje de consulta estructurado diseñado para la gestión de bases de datos relacionales. Permite la manipulación y recuperación de datos de manera eficiente. SQL es el estándar para la interacción con bases de datos relacionales, y su uso es fundamental en la gestión de la información en sistemas informáticos modernos.

Sql

MySQL es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto basado en SQL. Fue desarrollado por MySQL AB y actualmente es mantenido por Oracle Corporation. Es una de las bases de datos más utilizadas en aplicaciones web y empresariales debido a su rendimiento, fiabilidad y facilidad de uso. MySQL es ampliamente utilizado en aplicaciones como WordPress, Facebook y Twitter debido a su eficiencia y escalabilidad.

3. Herramientas empleadas

1. MySQL Workbench.
2. Latex Overleaf

4. Desarrollo

Ejercicios

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

```
CREATE table employee(idEmpoyee INT NOT NULL, firstName varchar (80) not null, lasName
    varchar(80) not null, salary float not null, joniningDate date not null, department
    varchar(50)not null);
```

Listing 1: Crear tabla employee

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

Employee table

Employee_id	First_name	Last_name	Salary	Joining_date	Departement
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

Figure 1: Tabla employee

```
INSERT INTO practica1.employee(idEmpoyee, firstName, lasName, salary, joniningDate,
    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'); varchar(50)not null);
```

Listing 2: Insertar registros en tabla reward

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

```
CREATE TABLE practica1.reward(employeeRefId int not null, dateReward date not null, amount
    float not null);
```

Listing 3: Crear tabla reward.

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

Reward table

Employee_ref_id	date_reward	amount
1	2019-05-11	1000
2	2019-02-15	5000
3	2019-04-22	2000
1	2019-06-20	8000

Figure 2: Tabla reward

```
INSERT INTO practica1.reward(employeeRefId, dateReward, amount) values
(1, '2019-05-11', 1000),
(2, '2019-02-15', 5000),
(3, '2019-04-22', 2000),
(1, '2019-06-20', 8000);
```

Listing 4: Insertar registros en tabla reward.

5. Obtener todos los empleados

```
select concat(firstName,' ', lasName) as Empleados from employee;
```

Listing 5: Seleccionar nombre de empleados con concat.

Empleados
Bob Kinto
Jerry Kansxo
Philip Jose
John Abraham
Michael Mathew
Alex Chreketo
Yohan Soso

Figure 3: Resultados sentencia

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

```
1 select firstName, lasName from employee;
```

Listing 6: Seleccionar nombre y apellido.

	firstName	lasName
▶	Bob	Kinto
	Jerry	Kansxo
	Philip	Jose
	John	Abraham
	Michael	Mathew
	Alex	Chreketo
	Yohan	Soso

Figure 4: Resultados sentencia

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

```
1 select firstName as 'Nombre de empleado' from employee;
```

Listing 7: Sentencia seleccionar firstname como Nombre de empleado.

	Nombre de empleado
▶	Bob
	Jerry
	Philip
	John
	Michael
	Alex
	Yohan

Figure 5: Resultados sentencia

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

```
1 select lower(lasName) as 'Apellido minusculas' from employee;
```

Listing 8: Sentencia seleccionar firstname en minusculas.

	Apellido minusculas
▶	kinto
	kansxo
	jose
	abraham
	mathew
	chreketo
	soso

Figure 6: Resultados sentencia

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

```
1 select upper(lasName) as 'Apellido mayusculas' from employee;
```

Listing 9: Sentencia seleccionar firstname en mayusculas.

	Apellido mayusculas
▶	KINTO
	KANSXO
	JOSE
	ABRAHAM
	MATHEW
	CHREKETO
	SOSO

Figure 7: Resultados sentencia

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

```
1 select distinct department from employee;
```

Listing 10: Sentencia seleccionar departamentos sin repetirse.

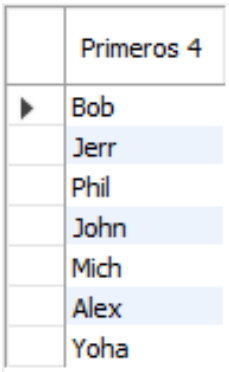
	department
▶	Finance
	IT
	Banking
	Insurance

Figure 8: Resultados sentencia

11. Obtener los primeros 4 caracteres de todos los valors de la columna “Firstname”.

```
1 select left(firstName, 4) as 'Primeros 4' from employee;
```

Listing 11: Sentencia seleccionar primeros 4 caracteres.



	Primeros 4
▶	Bob
	Jerr
	Phil
	John
	Mich
	Alex
	Yoha

Figure 9: Resultados sentencia

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

```
1 select instr(firstName, 'h')as 'Posicion Letra h' FROM employee where firstName='John';
```

Listing 12: Sentencia seleccionar la posicion de la letra H.

	Posicion Letra h
▶	3

Figure 10: Resultados sentencia

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

```
1 select ltrim(firstName)as 'Izquierda' FROM employee;
```

Listing 13: Sentencia borrar espacios a la izquierda.

	Izquierda
▶	Bob
	Jerry
	Philip
	John
	Michael
	Alex
	Yohan

Figure 11: Resultados sentencia

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

```
1 select rtrim(firstName)as 'Derecha' FROM employee;
```

Listing 14: Sentencia borrar espacios a la derecha.

	Derecha
▶	Bob
	Jerry
	Philip
	John
	Michael
	Alex
	Yohan

Figure 12: Resultados sentencia

6. Notaciones algebra relacional

5. Obtener todos los empleados:

$$\sigma_{true}(Empleados)$$

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

$$\pi_{First\_name,Last\_name}(Empleados)$$

7. Obtener todos los valores de la columna "First_name" usando el alias "Nombre de empleado":

$$\rho_{Nombre\_de\_empleado \leftarrow First\_name}(\pi_{First\_name}(Empleados))$$

8. Obtener todos los valores de la columna "Last_name" en minúsculas:

$$\pi_{lower(Last\_name)}(Empleados)$$

9. Obtener todos los valores de la columna "Last_name" en mayúsculas:

$$\pi_{upper(Last\_name)}(Empleados)$$

10. Obtener los nombres únicos de la columna "Departament":

$$\delta(\pi_{Departament}(Empleados))$$

11. Obtener los primeros 4 caracteres de todos los valores de la columna "First_name":

$$\pi_{substr(First\_name,1,4)}(Empleados)$$

12. Obtener la posición de la letra "h" en el nombre del empleado con First_name = "Jhon":

$$\pi_{position('h' \text{ in } First\_name)}(\sigma_{First\_name='Jhon'}(Empleados))$$

13. Obtener todos los valores de la columna "First_name" después de remover los espacios en blanco de la derecha:

$$\pi_{rtrim(First\_name)}(Empleados)$$

14. Obtener todos los valores de la columna "First_name" después de remover los espacios en blanco de la izquierda:

$$\pi_{ltrim(First\_name)}(Empleados)$$

5. Conclusiones

A lo largo de este documento, exploramos los conceptos fundamentales del Álgebra Relacional, SQL y MySQL, elementos clave en la gestión de bases de datos relacionales. Vimos cómo el álgebra relacional proporciona una base teórica sólida para la manipulación de datos, permitiendo realizar consultas estructuradas a través de operaciones como selección, proyección y renombramiento.
Además, analizamos distintas consultas en álgebra relacional para extraer y transformar datos, desde obtener todos los empleados hasta aplicar funciones como conversión de mayúsculas y minúsculas, extracción de caracteres y eliminación de espacios en blanco. Estas operaciones no solo demuestran la flexibilidad del modelo relacional, sino que también reflejan su importancia en la construcción de bases de datos eficientes.

Referencias Bibliográficas

References

[1] Codd, E. F. (1970). "A Relational Model of Data for Large Shared Data Banks." Communications of the ACM.

[2] Silberschatz, A., Korth, H. F., Sudarshan, S. (2020). Database System Concepts (7th ed.). McGraw-Hill.

[3] Date, C. J. (2004). An Introduction to Database Systems (8th ed.). Pearson.

[4] McFadden, F. R., Hoffer, J. A. (**2013**). Modern Database Management *11th ed Pearson*