Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO. 1.3.

álgebra Relacional y SQL

ALUMNO: Armando Oswaldo Pérez Romero
Dr. Eduardo Cornejo-Velázquez

1. Introducción

En esta práctica se estudió la aplicación del Álgebra Relacional como instrumento esencial para la gestión y consulta de bases de datos. Se realizaron consultas a través de esta formalización teórica y, luego, se incorporaron en MySQL Workbench para verificar su operación en un ambiente real de administración de bases de datos.

La finalidad principal del ejercicio fue fortalecer la conexión entre la teoría del Álgebra Relacional y su aplicación práctica en SQL, entendiendo cómo las consultas teóricas se convierten en sentencias ejecutables en MySQL. Esto permitio fortalecer los conocimientos acerca de la organización y recuperación de datos en bases de datos relacionales, otorgando un entendimiento más profundo de su operación en aplicaciones reales.

2. Marco teórico

Álgebra Relacional

Álgebra relacional es un lenguaje de consulta procedural. Consta de un conjunto de operaciones que toman como entrada una o dos relaciones y producen como resultado una nueva relación, por lo tanto, es posible anidar y combinar operadores.

1. Selección → Filtra filas de una relación según una condición.
2. Proyección → Extrae columnas específicas de una relación.
3. Unión → Combina los registros de dos relaciones eliminando duplicados.
4. Intersección → Devuelve los registros que aparecen en ambas relaciones.
5. Diferencia → Muestra los registros de una relación que no están en la otra.
6. Producto cartesiano → Combina todas las filas de una relación con todas las filas de otra.
7. Unión natural → Une dos relaciones por atributos comunes eliminando redundancias.
8. División → Encuentra registros que están relacionados con todos los valores de otra tabla.

Date C. J. Introducción a los Sistemas de Bases de Datos. Prentice Hall. Séptima edición 2001.

SQL

El lenguaje de consulta estructurado (SQL, por sus siglas en ingl´ es) es un lenguaje est´andar empleado para administrar y manejar bases de datos relacionales. Mediante SQL, los usuarios tienen la posibilidad de hacer varias operaciones en las bases de datos, tales como a˜nadir, actualizar, eliminar y consultar informaci´on. Este lenguaje es fundamental para interactuar con sistemas de gesti´on de bases de datos (SGBD) relacionales, como MySQL, PostgreSQL, SQL Server y Oracle, entre otros.

Escofet, C. M. (2002). El lenguaje SQL. UOC, la universidad virtual

MySQL

MySQL es un SGBD de código abierto ampliamente utilizado en el desarrollo de aplicaciones web, sistemas empresariales y diversas plataformas que requieren almacenamiento y manejo eficiente de datos. Se destaca por su rapidez, fiabilidad y facilidad de uso, además de ser compatible con múltiples sistemas operativos, como Windows, Linux y macOS.

Santillán, L. A. C., Ginestà, M. G., Mora, Ó. P. (2014). Bases de datos en MySQL. Universitat oberta de Catalunya.

Sentencias Utilizadas

1. CREATE TABLE: Para definir la estructura de las tablas.
2. INSERT INTO: Para agregar registros a las tablas.
3. SELECT: Para consultar informaci´on almacenada.
4. Funciones de cadena y agregación: Como LOWER(), UPPER(), DISTINCT(), LEFT(), RTRIM(), LTRIM(), entre otras.

3. Herramientas empleadas

Describir qué herramientas se han utilizado...

1. MySQL Workbench.

   MySQL Workbench es una herramienta visual de diseño y administración de bases de datos desarrollada por Oracle. Está específicamente diseñada para trabajar con bases de datos MySQL

4. Desarrollo

Ejercicio

El conjunto de ejercicios está basado en las tablas Employee y Reward. Se pide que para cada ejercicio se incluya una captura de pantalla con la sentencia SQL de solución y del resultado de su ejecución.

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

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

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

Listing 1: Crear tabla Employee.

```
CREATE TABLE Employee
(
Employee_id INT NOT NULL,
First_name VARCHAR(80) NOT NULL,
Last_name VARCHAR(80) NOT NULL,
Salary INT NOT NULL,
Joining_date DATE NOT NULL,
Departement VARCHAR(80) NOT NULL

);
```

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

```
INSERT INTO Employee(Employee_id,First_name,Last_name,Salary,Joining_date,Departement)values(1,"Bob","Kinto",1000000,"2019-01-20","Finance");
INSERT INTO Employee(Employee_id,First_name,Last_name,Salary,Joining_date,Departement)values(2,"Jerry","Kansxo",6000000,"2019-01-15","IT");
INSERT INTO Employee(Employee_id,First_name,Last_name,Salary,Joining_date,Departement)values(3,"Philip","Jose",8900000,"2019-02-05","Banking");
INSERT INTO Employee(Employee_id,First_name,Last_name,Salary,Joining_date,Departement)values(4,"John","Abraham",2000000,"2019-02-25","Insurance");
INSERT INTO Employee(Employee_id,First_name,Last_name,Salary,Joining_date,Departement)values(5,"Michael","Mathew",2200000,"2019-02-28","Finance");
INSERT INTO Employee(Employee_id,First_name,Last_name,Salary,Joining_date,Departement)values(6,"Alex","Chreketo",4000000,"2019-05-10","IT");
INSERT INTO Employee(Employee_id,First_name,Last_name,Salary,Joining_date,Departement)values(7,"Yohan","Soso",1230000,"2019-06-20","Banking");
```

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

Listing 2: Crear los 7 registros

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

Listing 3: Crear tabla Reward.

```
CREATE TABLE Reward
(
Employee_ref_id INT NOT NULL,
date_reward DATE NOT NULL,
amount INT NOT NULL

);
```

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

```
INSERT INTO Reward(Employee_ref_id,Date_reward,Amount)values(1,"2019-05-11",1000);
INSERT INTO Reward(Employee_ref_id,Date_reward,Amount)values(2,"2019-02-15",5000);
INSERT INTO Reward(Employee_ref_id,Date_reward,Amount)values(3,"2019-04-22",2000);
INSERT INTO Reward(Employee_ref_id,Date_reward,Amount)values(1,"2019-06-20",8000);
```

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

Listing 4: Crear los 4 registros

5. Obtener todos los empleados

$$\pi_{*}(\text{Employee})$$

Listing 5: Obtención de todos los empleados.

```
SELECT * FROM dbone.Employee;
```

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

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

$$\pi_{\text{First\_name, Last\_name}}(\text{Employee})$$

Listing 6: Obtención del nombre y apellido.

```
SELECT First_name, Last_name FROM dbone.Employee;
```

	First_name	Last_name
▶	Bob	Kinto
	Jerry	Kansxo
	Philip	Jose
	John	Abraham
	Michael	Mathew
	Alex	Chreketo
	Yohan	Soso

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

$$\rho_{\text{"Nombre de empleado"}/\text{First}}(\pi \text{First\_name}(\text{Employee}))$$

Listing 7: Obtención del nombre y apellido.

```
SELECT First_name AS Nombre_de_empleado FROM dbone.Employee;
```

	Nombre_de_empleado
▶	Bob
	Jerry
	Philip
	John
	Michael
	Alex
	Yohan

8. Obtener todos los valores de la columna “Last-name” en minúsculas.

$$\pi_{\text{LOWER(Last\_name)}}(\text{Employee})$$

Listing 8: Obtención en minusculas.

```
SELECT LOWER(Last_name) FROM dbone.Employee;
```

	LOWER(Last_name)
▶	kinto
	kansxo
	jose
	abraham
	mathew
	chreketo
	soso

9. Obtener todos los valores de la columna “Last-name” en mayusculas

$$\pi_{\mathrm{UPPER(Last\_name)}}(\mathrm{Employee})$$

Listing 9: Obtención en mayusculas.

```
SELECT UPPER(Last_name) FROM dbone.Employee;
```

	UPPER(Last_name)
▶	KINTO
	KANSXO
	JOSE
	ABRAHAM
	MATHEW
	CHREKETO
	SOSO

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

$$\delta(\pi_{\text{Departement}}(\text{Employee}))$$

Listing 10: Obtención de nombres unicos.

```
SELECT DISTINCT Departement FROM dbone.Employee;
```

	Departement
▶	Finance
	IT
	Banking
	Insurance

11. Obtener los primeros 4 caracteres de todos los valors de la columna “First-name”.

$$\pi_{\text{LEFT(First\_name, 4)}}(\text{Employee})$$

Listing 11: Obtención de los 4 caracteres.

```
SELECT LEFT (First_name,4) FROM dbone.Employee;
```

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

$$\pi_{\text{INSTR(First, 'h')}}(\sigma \text{First\_name} =' John'(\text{Employee}))$$

Listing 12: Obtención de la posición.

```
SELECT LOCATE("h",First_name) FROM dbone.Employee WHERE
First_name = "John";
```

	LEFT (First_name,4)
▶	Bob
	Jerr
	Phil
	John
	Mich
	Alex
	Yoha

	LOCATE("h",First_name)
▶	3

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

$$\pi_{\text{RTRIM(First\_name)}}(\text{Employee})$$

Listing 13: Obtención de los valores.

```
SELECT RTRIM(First_name) FROM dbone.Employee;
```

	RTRIM(First_name)
▶	Bob
	Jerry
	Philip
	John
	Michael
	Alex
	Yohan

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

$$\pi_{\text{LTRIM(First\_name)}}(\text{Employee})$$

Listing 14: Obtención de los valores.

```
SELECT LTRIM(First_name) FROM dbone.Employee;
```

	LTRIM(First_name)
▶	Bob
	Jerry
	Philip
	John
	Michael
	Alex
	Yohan

5. Conclusiones

En conclusión en esta práctica se reforzó el conocimiento sobre el Álgebra Relacional y su aplicación en bases de datos relacionales mediante MySQL Workbench. Se comprendió la importancia de las operaciones fundamentales como selección, proyección, unión, intersección, diferencia, producto cartesiano y unión natural, así como su traducción a sentencias SQL.
Además, se aprendieron nuevas consultas en SQL, lo que permitió ampliar el dominio sobre la manipulación de datos en bases de datos relacionales. La práctica también ayudó a consolidar la relación entre la teoría del Álgebra Relacional y su implementación en un entorno de gestión de bases de datos, fortaleciendo así la capacidad de análisis y estructuración de consultas más eficientes.

Referencias Bibliográficas

References

[1] Documentación oficial de MySQL: https://dev.mysql.com/doc

[2] Santillán, L. A. C., Ginestà, M. G., Mora, Ó. P. (2014). Bases de datos en MySQL. Universitat oberta de Catalunya.

[3] Escofet, C. M. (2002). El lenguaje SQL. UOC, la universidad virtual