

REPORTE DE PRÁCTICA NO. 3

NOMBRE DE LA PRÁCTICA —2.1 Práctica. Distribuidora de herramientas

Dr. Eduardo Cornejo-Velázquez

ALUMNO:Cesar Martinez Andrade



1. Introducción

En esta práctica, exploraremos el fascinante mundo de las funciones en el contexto de MySQL. Las funciones son herramientas poderosas que nos permiten realizar operaciones específicas dentro de nuestras consultas SQL. Al utilizarlas correctamente, podemos simplificar nuestro código, mejorar la eficiencia y obtener resultados más precisos.

Para llevar a cabo esta práctica, utilizaremos el software MySQL Command Client, una herramienta que nos permite interactuar con bases de datos MySQL mediante líneas de código. Aquí, escribiremos nuestras consultas y las compilaremos para obtener los resultados deseados.

Además, aprovecharemos los conceptos del álgebra relacional que hemos estudiado previamente en clase. El álgebra relacional es una teoría matemática que nos proporciona un conjunto de operaciones para manipular relaciones (tablas) en bases de datos. Al combinar estas operaciones con las funciones de MySQL, podremos resolver problemas complejos de manera elegante y estructurada.

Objetivos

1. Comprender el propósito y la sintaxis de las funciones en MySQL.
2. Familiarizarnos con el uso del software MySQL Command Client.
3. Aplicar el álgebra relacional para resolver problemas específicos.

2. Marco teórico

Álgebra Relacional

El álgebra relacional de manera breve es un conjunto de operaciones matemáticas que se aplican a relaciones (tablas) en una base de datos. Fue propuesto por Edgar F. Codd[4], el padre de las bases de datos relacionales. Algunas de las operaciones más importantes del álgebra relacional son:

- Selección[3]: Permite filtrar filas de una tabla según una condición específica. Por ejemplo, podemos seleccionar todos los estudiantes que tienen una calificación superior a 8.
- Proyección: Nos permite elegir columnas específicas de una tabla. Por ejemplo, podemos proyectar solo los nombres y apellidos de los profesores.
- Unión: Combina dos tablas para obtener un conjunto de filas sin duplicados. Por ejemplo, podemos unir las tablas de estudiantes y profesores para obtener una lista completa de personas relacionadas con la institución.

Funciones de SQL

Las funciones en SQL son operaciones predefinidas que se utilizan para procesar o transformar los datos de una base de datos. Estas funciones se combinan con la sentencia `SELECT` y otras sentencias para realizar diversas tareas. Aquí tienes una breve descripción de algunas funciones comunes:

1. Funciones de Agregado: Estas funciones realizan cálculos sobre grupos de filas y devuelven un solo valor. Ejemplos incluyen `SUM`, `AVG`, `COUNT`, `MIN` y `MAX`.
2. Funciones de Cadena: Manipulan cadenas de texto. Algunas funciones útiles son `CONCAT` (para concatenar cadenas), `SUBSTRING` (para extraer una parte de una cadena) y `UPPER/LOWER` (para convertir a mayúsculas o minúsculas).
3. Funciones de Fecha y Hora: Trabajan con fechas y horas. Ejemplos son `GETDATE()` (para obtener la fecha y hora actual), `DATEADD` (para agregar o restar días/meses/años) y `DATEDIFF` (para calcular la diferencia entre dos fechas).

- (d) Reporte mensual de ventas en la tienda del distribuidor. o
- (e) Ranking de clientes minoristas por número de herramientas vendidas y por monto de total de compras realizadas.

Sentencias SQL

En el Listado 1 se presentaran todas las lineas de código para resolver el bloque de actividades solicitado anteriormente.

Listing 1: Crear base de datos.

```
— Proveedores
CREATE TABLE Proveedores (
    proveedor_id INT PRIMARY KEY,
    nombre VARCHAR(100),
    contacto VARCHAR(100)
);

— Herramientas
CREATE TABLE Herramientas (
    herramienta_id INT PRIMARY KEY,
    nombre VARCHAR(100),
    descripcion VARCHAR(255),
    precio_compra DECIMAL(10, 2),
    proveedor_id INT,
    FOREIGN KEY (proveedor_id) REFERENCES Proveedores(proveedor_id)
);

— Inventario
CREATE TABLE Inventario (
    inventario_id INT PRIMARY KEY,
    herramienta_id INT,
    cantidad INT,
    precio_unitario DECIMAL(10, 2),
    FOREIGN KEY (herramienta_id) REFERENCES Herramientas(herramienta_id)
);

— Clientes
CREATE TABLE Clientes (
    cliente_id INT PRIMARY KEY,
    nombre VARCHAR(100),
    tipo_cliente VARCHAR(50),
    ruta_distribucion VARCHAR(100)
);

— Ventas
CREATE TABLE Ventas (
    venta_id INT PRIMARY KEY,
    cliente_id INT,
    fecha_venta DATE,
    total DECIMAL(10, 2),
    FOREIGN KEY (cliente_id) REFERENCES Clientes(cliente_id)
);

— Detalle_Ventas
```

```

CREATE TABLE Detalle_Ventas (
    detalle_id INT PRIMARY KEY,
    venta_id INT,
    herramienta_id INT,
    cantidad INT,
    precio_unitario DECIMAL(10, 2),
    FOREIGN KEY (venta_id) REFERENCES Ventas(venta_id),
    FOREIGN KEY (herramienta_id) REFERENCES Herramientas(herramienta_id)
);

```

— *Pedidos*

```

CREATE TABLE Pedidos (
    pedido_id INT PRIMARY KEY,
    cliente_id INT,
    fecha_pedido DATE,
    estado VARCHAR(50),
    FOREIGN KEY (cliente_id) REFERENCES Clientes(cliente_id)
);

```

— *Detalle_Pedidos*

```

CREATE TABLE Detalle_Pedidos (
    detalle_id INT PRIMARY KEY,
    pedido_id INT,
    herramienta_id INT,
    cantidad INT,
    precio_unitario DECIMAL(10, 2),
    FOREIGN KEY (pedido_id) REFERENCES Pedidos(pedido_id),
    FOREIGN KEY (herramienta_id) REFERENCES Herramientas(herramienta_id)
);

```

— *Proveedores*

```

INSERT INTO Proveedores (proveedor_id, nombre, contacto) VALUES
(1, 'Proveedor-Uno', 'contacto1@proveedor.com'),
(2, 'Proveedor-Dos', 'contacto2@proveedor.com');

```

— *Herramientas*

```

INSERT INTO Herramientas (herramienta_id, nombre, descripcion,
precio_compra, proveedor_id) VALUES
(1, 'Martillo', 'Martillo-de-acero', 100.00, 1),
(2, 'Destornillador', 'Destornillador-estrella', 50.00, 2),
(3, 'Llave-inglesa', 'Llave-ajustable', 150.00, 1);

```

— *Inventario*

```

INSERT INTO Inventario (inventario_id, herramienta_id,
cantidad, precio_unitario) VALUES
(1, 1, 50, 100.00),
(2, 2, 100, 50.00),
(3, 3, 30, 150.00);

```

— *Clientes*

```

INSERT INTO Clientes (cliente_id, nombre, tipo_cliente,
ruta_distribucion) VALUES
(1, 'Cliente-Uno', 'Mayorista', 'Ruta-1'),
(2, 'Cliente-Dos', 'Minorista', 'Ruta-2'),

```

```
(3, 'Cliente-Tres', 'Minorista', 'Ruta-3');
```

— *Ventas*

```
INSERT INTO Ventas (venta_id, cliente_id,
fecha_venta, total) VALUES
(1, 1, '2024-10-01', 5000.00),
(2, 2, '2024-10-02', 3000.00),
(3, 3, '2024-10-03', 4500.00);
```

— *Detalle_Ventas*

```
INSERT INTO Detalle_Ventas (detalle_id, venta_id,
herramienta_id, cantidad,
precio_unitario) VALUES
(1, 1, 1, 20, 100.00),
(2, 1, 3, 10, 150.00),
(3, 2, 2, 60, 50.00),
(4, 3, 1, 30, 100.00),
(5, 3, 3, 10, 150.00);
```

— *Pedidos*

```
INSERT INTO Pedidos (pedido_id, cliente_id, fecha_pedido,
estado) VALUES
(1, 2, '2024-10-04', 'Pagado'),
(2, 3, '2024-10-05', 'Pendiente');
```

— *Detalle_Pedidos*

```
INSERT INTO Detalle_Pedidos (detalle_id, pedido_id,
herramienta_id, cantidad, precio_unitario) VALUES
(1, 1, 2, 30, 50.00),
(2, 2, 3, 10, 150.00);
```

— *Consultas*

```
SELECT h.nombre, i.cantidad, i.precio_unitario,
(i.cantidad * i.precio_unitario) AS costo_total
FROM Inventario i
JOIN Herramientas h ON i.herramienta_id = h.herramienta_id;
```

```
SELECT p.nombre, EXTRACT(MONTH FROM v.fecha_venta)
AS mes, SUM(dv.cantidad * dv.precio_unitario) AS costo_total
FROM Proveedores p
JOIN Herramientas h ON p.proveedor_id = h.proveedor_id
JOIN Detalle_Ventas dv ON h.herramienta_id = dv.herramienta_id
JOIN Ventas v ON dv.venta_id = v.venta_id
GROUP BY p.nombre, EXTRACT(MONTH FROM v.fecha_venta);
```

```
SELECT c.nombre, c.ruta_distribucion
FROM Clientes c
WHERE c.tipo_cliente = 'Minorista';
```

```
SELECT EXTRACT(MONTH FROM v.fecha_venta) AS mes, SUM(v.total) AS total_ventas
FROM Ventas v
GROUP BY EXTRACT(MONTH FROM v.fecha_venta);
```

```
SELECT c.nombre, SUM(dv.cantidad) AS cantidad_total, SUM(dv.cantidad * dv.precio_unitario)
FROM Clientes c
JOIN Ventas v ON c.cliente_id = v.cliente_id
JOIN Detalle_Ventas dv ON v.venta_id = dv.venta_id
WHERE c.tipo_cliente = 'Minorista'
GROUP BY c.nombre
ORDER BY monto_total DESC;
```

Capturas de Pantalla

```
MySQL 9.0 Command Line Cli x + v
mysql>
mysql> -- Inventario
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Inventario (inventario_id, herramienta_id, cantidad, precio_unitario) VALUES
-> (1, 1, 50, 100.00),
-> (2, 2, 100, 50.00),
-> (3, 3, 30, 150.00);
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql>
mysql> -- Clientes
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Clientes (cliente_id, nombre, tipo_cliente, ruta_distribucion) VALUES
-> (1, 'Cliente Uno', 'Mayorista', 'Ruta 1'),
-> (2, 'Cliente Dos', 'Minorista', 'Ruta 2'),
-> (3, 'Cliente Tres', 'Minorista', 'Ruta 3');
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql>
mysql> -- Ventas
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Ventas (venta_id, cliente_id, fecha_venta, total) VALUES
-> (1, 1, '2024-10-01', 5000.00),
-> (2, 2, '2024-10-02', 3000.00),
-> (3, 3, '2024-10-03', 4500.00);
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql>
mysql> -- Detalle_Ventas
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Detalle_Ventas (detalle_id, venta_id, herramienta_id, cantidad, precio_unitario) VA
LUES
-> (1, 1, 1, 20, 100.00),
-> (2, 1, 3, 10, 150.00),
-> (3, 2, 2, 60, 50.00),
-> (4, 3, 1, 30, 100.00),
-> (5, 3, 3, 10, 150.00);
Query OK, 5 rows affected (0.00 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql>
mysql> -- Pedidos
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Pedidos (pedido_id, cliente_id, fecha_pedido, estado) VALUES
-> (1, 2, '2024-10-04', 'Pagado'),
```

Figure 2: Consultas solicitadas


```
MySQL 9.0 Command Line Cli  X + -
mysql>
mysql> -- Inventario
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Inventario (inventario_id, herramienta_id, cantidad, precio_unitario) VALUES
-> (1, 1, 50, 100.00),
-> (2, 2, 100, 50.00),
-> (3, 3, 30, 150.00);
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql>
mysql> -- Clientes
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Clientes (cliente_id, nombre, tipo_cliente, ruta_distribucion) VALUES
-> (1, 'Cliente Uno', 'Mayorista', 'Ruta 1'),
-> (2, 'Cliente Dos', 'Minorista', 'Ruta 2'),
-> (3, 'Cliente Tres', 'Minorista', 'Ruta 3');
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql>
mysql> -- Ventas
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Ventas (venta_id, cliente_id, fecha_venta, total) VALUES
-> (1, 1, '2024-10-01', 5000.00),
-> (2, 2, '2024-10-02', 3000.00),
-> (3, 3, '2024-10-03', 4500.00);
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql>
mysql> -- Detalle_Ventas
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Detalle_Ventas (detalle_id, venta_id, herramienta_id, cantidad, precio_unitario) VA
LUES
-> (1, 1, 1, 20, 100.00),
-> (2, 1, 3, 10, 150.00),
-> (3, 2, 2, 60, 50.00),
-> (4, 3, 1, 30, 100.00),
-> (5, 3, 3, 10, 150.00);
Query OK, 5 rows affected (0.00 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql>
mysql> -- Pedidos
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Pedidos (pedido_id, cliente_id, fecha_pedido, estado) VALUES
-> (1, 2, '2024-10-04', 'Pagado'),
```

Figure 3: Consultas solicitadas.

5. Conclusiones

En el contexto académico de las bases de datos, el álgebra relacional se presenta como una herramienta esencial. Durante esta práctica, he aprendido su relevancia al momento de insertar datos y he explorado algunas de las funciones que ofrece el lenguaje SQL. Entre estas funciones, encontramos elementos cruciales como WHILE, ASC, REPLACE y ORDER BY. Estas palabras reservadas me permitirán afinar y optimizar mis consultas en el futuro. Además de tener conocimiento del tema de inyección SQL que hace más interesante el tema de seguridad en base de datos.

6. Referencias

References

- [1] Algebra relacional. (n.d). <https://basesdatosdistribuidas.blogspot.com/2012/11/algebra-relacional.html>
- [2] Platzi. (n.d.). *Álgebra relacional y Bases de Datos*. <https://platzi.com/tutoriales/1566-bd/5823-algebra-relacional-y-bases-de-datos/>
- [3] VanMSFT. (n.d.). *Ejemplos de SELECT (Transact-SQL) - SQL Server*. Microsoft Learn. <https://learn.microsoft.com/es-es/sql/t-sql/queries/select-examples-transact-sql?view=sql-server-ver16>
- [4] colaboradores de Wikipedia. (2024, July 14). *Edgar Frank Codd*. Wikipedia, La Enciclopedia Libre. https://es.wikipedia.org/wiki/Edgar_Frank_Codd
- [5] Belcic, I. (2023, February 23). *¿Qué es la inyección de SQL y cómo funciona? ¿Qué Es La Inyección De SQL Y Cómo Funciona?* <https://www.avast.com/es-es/c-sql-injection>