DDL y DML en PostgreSQL

Introducción a PostgreSQL

- **PostgreSQL**: Sistema de gestión de bases de datos relacional.
- **DDL**: Lenguaje para definir estructuras de datos.
- **DML**: Lenguaje para manipular datos.

Objetivos

¿Qué aprenderemos hoy?

1. Entender la diferencia entre **DDL** y **DML** en PostgreSQL.
2. Aprender a usar **DDL** para gestionar estructuras de datos.
3. Practicar **DML** para manipular información: **SELECT**, **INSERT**, **UPDATE**, **DELETE** y **UPSERT**.
4. Implementar conceptos avanzados como **JOIN**, funciones, procedimientos y triggers.
5. Aplicar estos conceptos a un caso real con la base de datos **ClassicModels**.

Agenda

Estructura de la Clase

1. **Introducción**
    - ¿Qué es PostgreSQL?
    - Diferencias entre DDL y DML.
2. **DDL**
    - Creación, modificación y eliminación de tablas.
    - Gestión de índices.
3. **DML**
    - Operaciones: SELECT, INSERT, UPDATE, DELETE, UPSERT.
    - Ejemplo de JOIN.

4. **Funciones y Procedimientos**
    - User-Defined Functions (UDF).
    - Procedimientos almacenados (Stored Procedures).
5. **Triggers**
    - Automatización de tareas.
    - Ejemplo práctico de auditoría.
6. **Resumen y Conclusión**
    - Repaso de conceptos clave.
    - Preguntas y respuestas.

Diferencias entre DDL y DML

DDL (Data Definition Language)

- **Definición**: Lenguaje para crear y modificar estructuras de datos.
- **Operaciones comunes**:
  - `CREATE TABLE`
  - `ALTER TABLE`
  - `DROP TABLE`
  - `CREATE INDEX`

DML (Data Manipulation Language)

- **Definición**: Lenguaje para manipular los datos dentro de las estructuras creadas.
- **Operaciones comunes**:
  - `SELECT`
  - `INSERT`
  - `UPDATE`
  - `DELETE`
  - `UPSERT`

DDL: Crear Tablas

Ejemplo de creación de tablas

Crear la tabla `customers` para la base de datos **ClassicModels**:

```
CREATE TABLE customers (
    customerNumber INT PRIMARY KEY,
    customerName VARCHAR(50) NOT NULL,
    city VARCHAR(50),
    country VARCHAR(50)
);
```

DDL: Modificar Tablas

Alterar estructuras existentes

Ejemplos de operaciones con `ALTER TABLE`:

1. **Agregar una columna:**

```
ALTER TABLE customers ADD COLUMN isVIP BOOLEAN DEFAULT FALSE;
```

2. **Cambiar tipo de dato:**

```
ALTER TABLE customers ALTER COLUMN city TYPE VARCHAR(100);
```

3. **Eliminar una columna:**

```
ALTER TABLE customers DROP COLUMN country;
```

DDL: Índices

Crear y eliminar índices

1. **Crear un índice:**

```
CREATE INDEX idx_city ON customers (city);
```

2. **Eliminar un índice:**

```
DROP INDEX IF EXISTS idx_city;
```

Ventajas de los índices:

- Mejoran el rendimiento de consultas frecuentes.
- Requieren más espacio y afectan operaciones de escritura.

DDL: Eliminar Tablas

Ejemplo de `DROP TABLE`

Eliminar la tabla de prueba si existe:

```
DROP TABLE IF EXISTS test_table;
```

Precaución:

- Esta operación elimina todos los datos de la tabla.

DML: SELECT

Consulta básica de datos

Ejemplo: Obtener clientes de los Estados Unidos:

```
SELECT customerName, city, country
FROM customers
WHERE country = 'USA';
```

SELECT con JOIN

Combinar datos de múltiples tablas

Ejemplo: Obtener información de clientes y sus representantes de ventas:

```
SELECT c.customerName, c.city, e.firstName || ' ' || e.lastName AS salesRep
FROM customers c
LEFT JOIN employees e
ON c.salesRepEmployeeNumber = e.employeeNumber;
```

Explicación:

- `LEFT JOIN`: Incluye todos los clientes, incluso si no tienen representante.
- Concatena el nombre y apellido del representante.

DML: INSERT

Insertar nuevos registros

Ejemplo: Agregar un cliente nuevo:

```
INSERT INTO customers (customerNumber, customerName, city, country)
VALUES (1001, 'John Doe', 'New York', 'USA');
```

DML: UPDATE

Actualizar registros existentes

Ejemplo: Marcar clientes con crédito mayor a 50,000 como VIP:

```
UPDATE customers
SET isVIP = TRUE
WHERE creditLimit > 50000;
```

DML: DELETE

Eliminar registros

Ejemplo: Eliminar un cliente específico:

```
DELETE FROM customers
WHERE customerNumber = 103;
```

DML: UPSERT

Insertar o actualizar registros condicionalmente

Ejemplo: Agregar un cliente o actualizar su ciudad si ya existe:

```
INSERT INTO customers (customerNumber, customerName, city)
VALUES (2001, 'New Customer', 'Los Angeles')
ON CONFLICT (customerNumber)
DO UPDATE SET
    customerName = EXCLUDED.customerName,
    city = EXCLUDED.city;
```

Nota:

- `ON CONFLICT`: Maneja conflictos en claves únicas.
- `EXCLUDED`: Representa los valores nuevos que causaron el conflicto.

Resumen de DML

Operaciones vistas

1. **SELECT**: Consulta de datos con **JOIN**.
2. **INSERT**: Inserción de registros nuevos.
3. **UPDATE**: Modificación de registros existentes.
4. **DELETE**: Eliminación de registros.
5. **UPSERT**: Inserción o actualización condicional.

Funciones UDF

Crear una función

Ejemplo: Calcular un descuento:

```
CREATE OR REPLACE FUNCTION calculate_discount(amount DECIMAL, discountRate DECIMAL)
RETURNS DECIMAL AS $$
BEGIN
    RETURN amount - (amount * discountRate / 100);
END;
$$ LANGUAGE plpgsql;
```

Stored Procedures

Procedimientos Almacenados

Ejemplo: Agregar un cliente:

```
CREATE OR REPLACE PROCEDURE add_customer(
    p_customerNumber INT,
    p_customerName VARCHAR,
    p_city VARCHAR
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO customers (customerNumber, customerName, city)
    VALUES (p_customerNumber, p_customerName, p_city);
EXCEPTION WHEN unique_violation THEN
    RAISE NOTICE 'El cliente ya existe.';
END;
$$;

CALL add_customer(2002, 'Jane Doe', 'Chicago');
```

Triggers

Automatización de tareas

1. **Tabla de auditoría:**

```sql
CREATE TABLE customer_audit (
    audit_id SERIAL PRIMARY KEY,
    customerNumber INT,
    operation VARCHAR(10),
    changed_at TIMESTAMP DEFAULT NOW()
);
```

2. **Función del trigger:**

```
CREATE OR REPLACE FUNCTION audit_customers()
RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'UPDATE') THEN
        INSERT INTO customer_audit (customerNumber, operation)
        VALUES (NEW.customerNumber, 'UPDATE');
    ELSIF (TG_OP = 'DELETE') THEN
        INSERT INTO customer_audit (customerNumber, operation)
        VALUES (OLD.customerNumber, 'DELETE');
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

3. **Crear el trigger:**

```
CREATE TRIGGER trg_audit_customers
AFTER UPDATE OR DELETE ON customers
FOR EACH ROW
EXECUTE FUNCTION audit_customers();
```

Resumen

¿Qué aprendimos hoy?

1. **DDL**: Crear, modificar y eliminar estructuras de datos.
2. **DML**:
    - SELECT, INSERT, UPDATE, DELETE y UPSERT.
    - Uso de JOIN.
3. **Funciones y Procedimientos**: Reutilización de lógica.
4. **Triggers**: Automatización de auditorías.

¡Gracias!

Preguntas y Respuestas