UACM
Universidad Autónoma de la Ciudad de México
NADA HUMANO ME ES AJENO

Lic. Daniel Leal Laris
Ramírez Angeles Rafael
11-011-0654
Especialidad de Datos I
Práctica Normalización

Introducción

La normalización de tablas en una base de datos es un proceso crucial para diseñar bases de datos eficientes y sin redundancias. Este método se utiliza para organizar la información de manera estructurada y reducir la redundancia de datos, lo que a su vez mejora la integridad y la eficiencia de la base de datos.

Se basa en un conjunto de reglas que ayudan a eliminar la redundancia y las dependencias entre los datos. Estas reglas están diseñadas para dividir las tablas en estructuras más pequeñas y relacionadas, evitando así problemas como la pérdida de coherencia de datos y la actualización inconsistente de la información.

El proceso sigue una serie de niveles o formas (1NF, 2NF, 3NF, etc.), cada uno de los cuales aborda diferentes aspectos de la redundancia y las dependencias en los datos. La idea principal es que cada tabla cumpla con ciertas condiciones para minimizar la duplicación de información y garantizar la integridad referencial.

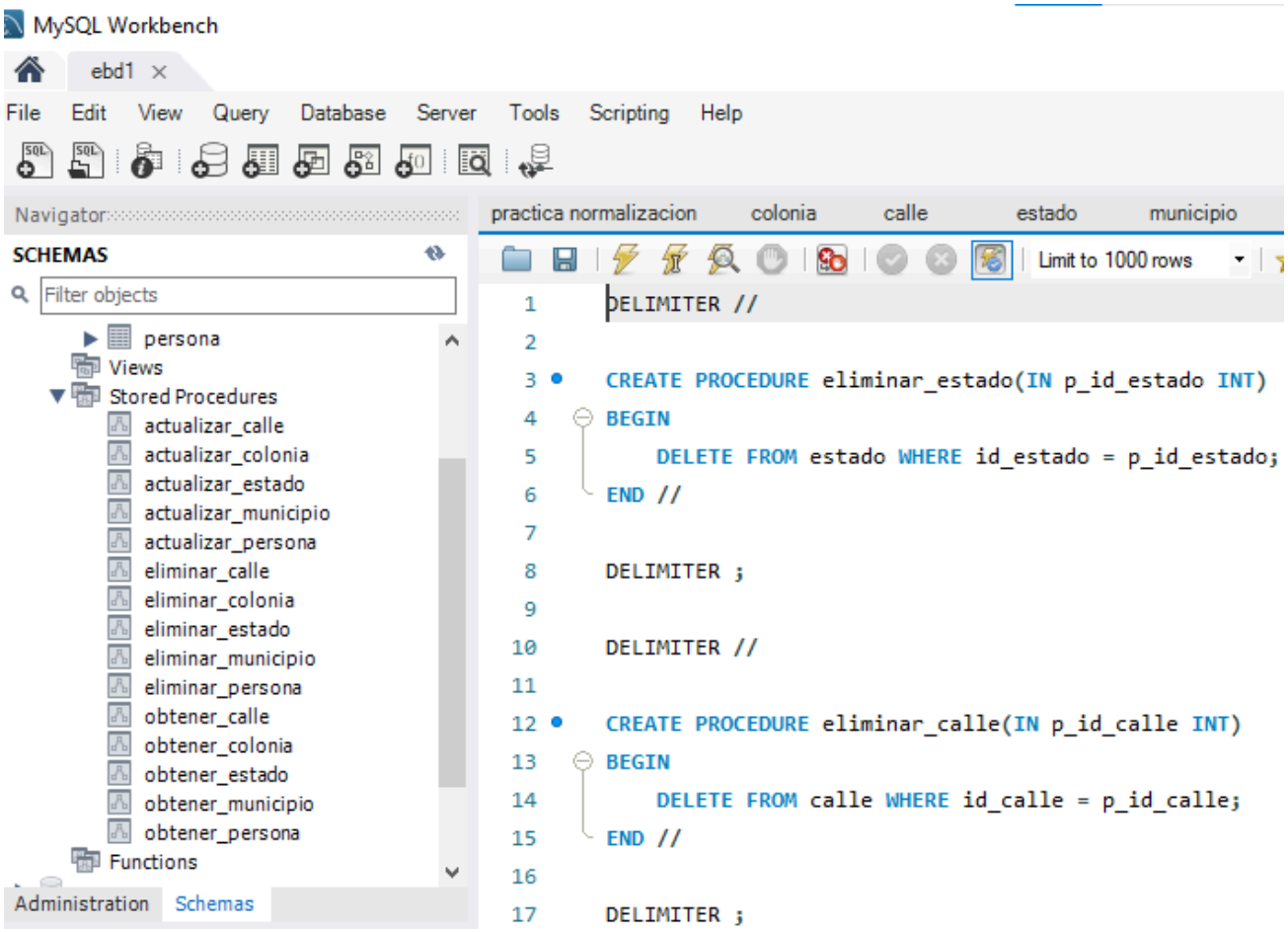
Desarrollo

En el laboratorio se adquirieron los conceptos de las sentencias Procedure la cual define un procedimiento de SQL en un servidor.

Esta sentencia se puede incorporar a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico.

Se elaboró un CRUD con el lenguaje de programación Python que actuó sobre las tablas creadas en MySQL.

Lic. Daniel Leal Laris
Ramírez Angeles Rafael
11-011-0654
Especialidad de Datos I
Práctica Normalización



Script SQL donde se puede observar el Stored Procedure para eliminar un registro en la tabla estado y eliminar un registro en la tabla calle. Se observan también todos los Stored Procedure utilizados en la práctica

C: > Users > Rafael > Documents > Especialidad base datos I > practica direcciones > procedure_insert.py > ...

Click here to ask Blackbox to help you code faster

```python
#Insertar estado
import mysql.connector

# Datos de ejemplo para el estado a insertar
id_estado = 1
descripcion = 'Guerrero'

try:
    # Conectar a la base de datos
    cnx = mysql.connector.connect(user='root', password='root', database='direcciones')
    cursor = cnx.cursor()

    # Llamar al procedimiento almacenado directamente
    cursor.execute("""
        INSERT INTO estado (id_estado, descripcion)
        VALUES (%s, %s)
    """, (id_estado, descripcion))

    # Confirmar los cambios
    cnx.commit()

    print("Inserción de estado realizada con éxito.")

except mysql.connector.Error as err:
    print(f"Error: {err}")

finally:
    # Cerrar la conexión
    cursor.close()
    cnx.close()
```

Script de python que inserta un nuevo registro en la tabla “estado”

```python
# Actualizar calle
import mysql.connector

# Conexión a la base de datos
conexion = mysql.connector.connect(
    user="root",
    password="root",
    database="direcciones"
)

# Crear un objeto cursor
cursor = conexion.cursor()

# Procedimiento para actualizar registros en la tabla calle
def actualizar_calle(id_calle, nueva_descripcion):
    try:
        cursor.callproc('actualizar_calle', (id_calle, nueva_descripcion))
        conexion.commit()
        print("Registro de calle actualizado exitosamente.")
    except Exception as e:
        print(f"Error al actualizar el registro de calle: {str(e)}")
    finally:
        cursor.close()
        conexion.close()

# Ejemplo de uso
actualizar_calle(2, 'Calle Vieja')
```

Script de python que actualiza un registro en la tabla “calle”

C: > Users > Rafael > Documents > Especialidad base datos I > practica direcciones > procedure_eliminar.py > ...

```python
# Eliminar persona
import mysql.connector

def eliminar_persona(id_persona):
    try:
        # Conectar a la base de datos
        cnx = mysql.connector.connect(user='root', password='root', database='direcciones')
        cursor = cnx.cursor()

        # Sentencia SQL para eliminar un registro en la tabla persona
        delete_query = "DELETE FROM persona WHERE id_persona = %s"

        # Ejecutar la sentencia SQL con el parámetro
        cursor.execute(delete_query, (id_persona,))

        # Hacer commit para aplicar los cambios
        cnx.commit()

        print(f"Registro con id_persona {id_persona} eliminado exitosamente.")

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        # Cerrar la conexión
        cursor.close()
        cnx.close()

# Llamar a la función eliminar_persona con el id_persona deseado
eliminar_persona(1)
```

Script de python que elimina un registro en la tabla “persona”

Lic. Daniel Leal Laris
Ramírez Angeles Rafael
11-011-0654
Especialidad de Datos I
Práctica Normalización

C: > Users > Rafael > Documents > Especialidad base datos I > practica direcciones > procedure_seleccionar.py > ...

```python
#Seleccionar colonia
import mysql.connector

def obtener_colonia(id_colonia):
    try:
        # Conectar a la base de datos
        cnx = mysql.connector.connect(user='root', password='root', database='direcciones')
        cursor = cnx.cursor()

        # Llamar al procedimiento almacenado
        cursor.callproc('obtener_colonia', (id_colonia,))

        # Recuperar los resultados
        for result in cursor.stored_results():
            rows = result.fetchall()
            for row in rows:
                print(row)

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        # Cerrar la conexión
        cursor.close()
        cnx.close()

# Llamar a la función obtener_colonia con el valor deseado
obtener_colonia(1)
```

Script de python que selecciona un registro en la tabla “colonia”

Result Grid | Filter Rows:

id_estado	descripcion
1	Guerrero
2	Oaxaca
3	Ciudad de México
4	Sonora
5	San Luis Potosí
NULL	NULL

Tabla Estado modificada con el script de Python

Conclusiones

Los stored procedures en SQL son bloques de código SQL que se almacenan en el sistema de gestión de bases de datos y se pueden ejecutar mediante una llamada desde una aplicación, un programa o incluso desde otras consultas SQL.

Permiten encapsular lógica de negocio compleja en un solo lugar facilitando la reutilización del código, ya que se puede llamar al procedimiento desde múltiples ubicaciones en la aplicación, evitando la duplicación de código y mejorando la consistencia.

Ofrecen un rendimiento mejorado en comparación con enviar múltiples consultas desde la aplicación. Esto se debe a que se reduce la cantidad de datos que se deben transferir entre la base de datos y la aplicación, y también se pueden aprovechar optimizaciones del servidor.

Ayudan a mejorar la seguridad al limitar el acceso directo a las tablas de la base de datos. Los usuarios pueden tener permisos para ejecutar el procedimiento, pero no necesariamente para acceder a las tablas subyacentes, lo que ayuda a controlar el acceso a los datos.

Cualquier cambio necesario se realiza en un único lugar. Esto simplifica el mantenimiento y reduce el riesgo de errores al actualizar la lógica de la aplicación.

Gestionan transacciones de manera más efectiva. Pueden incluir múltiples operaciones SQL dentro de una transacción, lo que garantiza que todas las operaciones se completen correctamente o se deshagan en caso de error.

Es más fácil para los desarrolladores trabajar con la base de datos sin tener que comprender completamente la lógica interna de cada consulta.