📌 Manejo de transacciones en PostgreSQL usando `with` en Python

✨ Introducción

En esta guía aprenderás cómo manejar **transacciones de manera automática y segura** utilizando el manejador de contexto **`with`** en Python al trabajar con PostgreSQL. Este enfoque garantiza que, si todo sale bien, se hace **commit** automáticamente, y si ocurre algún error, se realiza **rollback** sin necesidad de escribirlo manualmente.

🔥 Paso 1: Conexión y apertura de cursor usando `with`

📂 **Archivo:** `transacciones.py`

Descripción:
Establecemos la conexión y el cursor utilizando **`with`** para delegar el manejo de transacciones al contexto.

```
import psycopg
```

```
with psycopg.connect(
    user="postgres", password="admin",
    host="localhost", port="5432", dbname="test_db"
) as conexion:
    with conexion.cursor() as cursor:
```

👉 **Explicación:**
✅ Usando **`with`**, al salir del bloque:

- Si no hubo errores → se hace **commit** automáticamente.
- Si ocurrió un error → se hace **rollback** automáticamente.

🔥 Paso 2: Ejecutar la sentencia `INSERT`

Descripción:
Definimos y ejecutamos la sentencia SQL para insertar un nuevo registro en la tabla **persona**.

```
sentencia_insert = (
    "INSERT INTO persona (nombre, apellido, email) "
    "VALUES (%s, %s, %s)"
)
valores_insert = ("Alex", "Rojas", "arojas@mail.com")
cursor.execute(sentencia_insert, valores_insert)
```

👉 **Explicación:**
Agregamos el registro con nombre **Alex**, apellido **Rojas**, y su email.

🔥 Paso 3: Ejecutar la sentencia `UPDATE`

Descripción:
Definimos y ejecutamos una segunda sentencia SQL para actualizar un registro existente.

```
sentencia_update = (
    "UPDATE persona "
    "SET nombre=%s, apellido=%s, email=%s "
    "WHERE id_persona=%s"
)
valores_update = ("Juan", "Perez", "jperez@mail.com", 1)
cursor.execute(sentencia_update, valores_update)
```

👉 **Explicación:**
Actualizamos el registro con **id_persona = 1**, restableciendo su nombre a **Juan**, apellido a **Perez** y email a **jperez@mail.com**.

🔥 Paso 4: Confirmación automática de la transacción

Descripción:
Al salir del bloque `with`, se ejecuta automáticamente **commit** si todo fue exitoso, o **rollback** si hubo errores.

```
print("Termina la transacción, se hizo commit")
```

👉 **Explicación:**
Imprimimos un mensaje indicando que la transacción se completó.

✅ Si ocurre un error, `with` hace automáticamente **rollback** y la excepción puede capturarse con **try-except** externo si se desea para saber el detalle exacto del error, pero es opcional.

✅ Código completo por archivo

📂 **Archivo completo:** `transacciones.py`
➡️ Aquí tienes el **código completo actualizado**:

```
import psycopg

with psycopg.connect(
    user="postgres", password="admin",
    host="localhost", port="5432", dbname="test_db"
) as conexion:
    with conexion.cursor() as cursor:
        sentencia_insert = (
            "INSERT INTO persona (nombre, apellido, email) "
            "VALUES (%s, %s, %s)"
        )
        valores_insert = ("Alex", "Rojas", "arojas@mail.com")
        cursor.execute(sentencia_insert, valores_insert)
        sentencia_update = (
            "UPDATE persona "
            "SET nombre=%s, apellido=%s, email=%s "
            "WHERE id_persona=%s"
        )
```

```
        valores_update = ("Juan", "Perez", "jperez@mail.com", 1)
        cursor.execute(sentencia_update, valores_update)
print("Termina la transacción, se hizo commit")
```

🎉 Conclusión

En esta guía aprendiste cómo manejar **transacciones automáticas** utilizando `with` al trabajar con PostgreSQL en Python. Esta práctica facilita el control de la transacción, reduce errores y asegura que no sea necesario invocar manualmente **commit** o **rollback**.

Este enfoque es más limpio, seguro y recomendado para mantener la integridad de los datos. 🚀

✨ Sigue adelante con tu aprendizaje 🚀, ¡el esfuerzo vale la pena!

¡Saludos! 👋

Ing. Marcela Gamiño e Ing. Ubaldo Acosta

Fundadores de GlobalMentoring.com.mx