



✓ Guía: Conclusión del uso de Pool de Conexiones en la clase DAO

🎯 ¿Qué se aprendió en esta lección?

Finalizamos el proyecto aplicando el uso de *Pool de Conexiones* en nuestra clase `PersonaDAO`. Sin embargo, en esta etapa ya **no fue necesario modificar más código**, porque:

Ya estamos utilizando `with` para abrir automáticamente la conexión y el cursor, lo cual `psycopg_pool` maneja perfectamente.

✓ ¿Qué significa esto?

Con la nueva versión de `psycopg_pool`:

- Ya **no necesitamos obtener ni liberar manualmente las conexiones**.
- Ya **no usamos clases intermedias como `CursorDelPool`**.
- Cada vez que usamos `with`, se ejecuta:
 - `__enter__`: obtiene una conexión y un cursor.
 - `__exit__`: hace `commit` o `rollback`, cierra el cursor y regresa la conexión al pool automáticamente.

✚ ¿Se modificó algo en `PersonaDAO`?

No. Toda la clase ya funcionaba correctamente gracias al uso de `with`.

Lo único que puedes agregar como *buena práctica* es **cerrar el pool al final del script**, si estás haciendo pruebas directas:

```
from conexion import Conexion
```

```
Conexion._pool.close()
```

📄 Esto no es obligatorio si el programa ya termina, pero se recomienda si quieres liberar recursos antes del final.

💡 Qué se hacía antes (solo como documentación)

Antes, al no tener manejo automático:

- Había que obtener manualmente la conexión del pool.
- Obtener el cursor desde la conexión.
- Hacer `commit()` o `rollback()` dependiendo del resultado.
- Liberar la conexión con `putconn()`.

Ese manejo lo hacía una clase como `CursorDelPool`, que ahora **ya no es necesaria** ✅

🔧 Pruebas realizadas en esta lección

Se probaron los 4 métodos CRUD de la clase `PersonaDAO`:

1. `seleccionar()`: Muestra todos los registros como objetos `Persona`.
2. `insertar()`: Agrega un nuevo registro (por ejemplo, "Alejandra Téllez").
3. `actualizar()`: Cambia valores de un registro existente.
4. `eliminar()`: Elimina un registro (por ejemplo, el id 15).

🔍 En todos los casos se observaron correctamente los mensajes de log:

- Inicio y fin del bloque `with`
- Commit exitoso o rollback en caso de errores
- Reutilización automática de conexiones desde el pool

✦ Código completo de la clase PersonaDAO:

```
from conexion import Conexion
from persona import Persona
from logger_base import log

class PersonaDAO:
    """
    DAO (Data Access Object)
    CRUD (Create-Read-Update-Delete)
    """
    _SELECCIONAR = 'SELECT * FROM persona ORDER BY id_persona'
    _INSERTAR = 'INSERT INTO persona(nombre, apellido, email) VALUES(%s, %s, %s)'
    _ACTUALIZAR = 'UPDATE persona SET nombre=%s, apellido=%s, email=%s WHERE id_persona=%s'
    _ELIMINAR = 'DELETE FROM persona WHERE id_persona=%s'

    @classmethod
    def seleccionar(cls):
        with Conexion.obtener_conexion() as conexion:
            with conexion.cursor() as cursor:
                cursor.execute(cls._SELECCIONAR)
                registros = cursor.fetchall()
                personas = []
                for registro in registros:
                    persona = Persona(registro[0], registro[1], registro[2], registro[3])
                    personas.append(persona)
                return personas

    @classmethod
    def insertar(cls, persona):
        with Conexion.obtener_conexion() as conexion:
            with conexion.cursor() as cursor:
                valores = (persona.nombre, persona.apellido, persona.email)
                cursor.execute(cls._INSERTAR, valores)
                log.debug(f'Persona insertada: {persona}')
                return cursor.rowcount

    @classmethod
    def actualizar(cls, persona):
        with Conexion.obtener_conexion() as conexion:
            with conexion.cursor() as cursor:
                valores = (persona.nombre, persona.apellido, persona.email, persona.id_persona)
                cursor.execute(cls._ACTUALIZAR, valores)
```

```

        log.debug(f'Persona actualizada: {persona}')
        return cursor.rowcount

    @classmethod
    def eliminar(cls, persona):
        with Conexion.obtener_conexion() as conexion:
            with conexion.cursor() as cursor:
                valores = (persona.id_persona,)
                cursor.execute(cls._ELIMINAR, valores)
                log.debug(f'Objeto eliminado: {persona}')
                return cursor.rowcount

if __name__ == '__main__':
    # Insertar un registro
    # persona1 = Persona(nombre='Pedro', apellido='Najera', email='pnajera@mail.com')
    # personas_insertadas = PersonaDAO.insertar(persona1)
    # log.debug(f'Personas insertadas: {personas_insertadas}')




    # Actualizar un registro
    # persona1 = Persona(1, 'Juan Carlos', 'Juarez', 'cjjuarez@mail.com')
    # personas_actualizadas = PersonaDAO.actualizar(persona1)
    # log.debug(f'Personas actualizadas: {personas_actualizadas}')

    # Eliminar un registro
    # persona1 = Persona(id_persona=7)
    # personas_eliminadas = PersonaDAO.eliminar(persona1)
    # log.debug(f'Personas eliminadas: {personas_eliminadas}')

    # Seleccionar objetos
    personas = PersonaDAO.seleccionar()
    for persona in personas:
        log.debug(persona)

    # Cerramos el pool de conexiones
    Conexion._pool.close()
    log.debug('Pool de conexiones cerrado correctamente.')
```

Conclusión

-  Gracias a `psycopg_pool` y el uso de `with`, **la clase DAO no requiere cambios**.
-  El pool de conexiones se gestiona de forma automática y eficiente.
-  Solo se recomienda agregar la línea final para cerrar el pool si haces pruebas:

```
Conexion._pool.close()
```

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

¡Saludos! 🙌

Ing. Marcela Gamiño e Ing. Ubaldo Acosta

Fundadores de [GlobalMentoring.com.mx](https://www.globalmentoring.com.mx)