Patrón de diseño Serialized LOB

Alumno: Pedro Arias

¿Qué es Serialized LOB?

Serialized LOB se refiere a la serialización de objetos grandes (Large Object Binary). En esencia, es un proceso que convierte objetos complejos o grandes en una secuencia de bytes para su almacenamiento o transmisión.

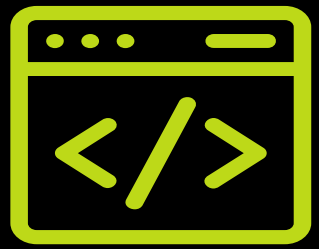
Uso de Serialized LOB

Este patrón se utiliza principalmente para manejar objetos de gran tamaño, como imágenes, archivos multimedia o incluso datos complejos en bases de datos

Funcionamiento Interno:

01 La serialización implica convertir un objeto, que puede ser complejo o grande, en una secuencia de bytes que puede ser almacenada en una base de datos, un sistema de archivos o transmitida a través de la red.

02 Un proceso de serialización toma el objeto y lo convierte en una secuencia de bytes que representan la estructura y el estado del objeto.



03 Posteriormente, estos bytes pueden ser deserializados para reconstruir el objeto original.

Ejemplo con serialized LOB

```python
import pickle

# Clase de ejemplo
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

# Objeto a serializar
persona = Persona("Juan", 30)

# Serialización: Convertir el objeto en una secuencia de bytes
datos_serializados = pickle.dumps(persona)

# Almacenar los datos serializados en un archivo
with open("persona.pickle", "wb") as archivo:
    archivo.write(datos_serializados)
```

```python
# Leer los datos serializados desde el archivo
with open("persona.pickle", "rb") as archivo:
    datos_leidos = archivo.read()

# Deserialización: Convertir los datos en un objeto
persona_recuperada = pickle.loads(datos_leidos)

# Imprimir los datos de la persona recuperada
print("Nombre:", persona_recuperada.nombre)
print("Edad:", persona_recuperada.edad)
```

Ejemplo sin Serialized LOB

```python
# Crear un diccionario con los datos de un empleado
empleado = {
    'nombre': "Juan Pérez",
    'salario': 50000.0
}

# Recuperar los datos del empleado desde el diccionario
nombre_recuperado = empleado['nombre']
salario_recuperado = empleado['salario']

# Imprimir los datos del empleado recuperado
print("Nombre del Empleado:", nombre_recuperado)
print("Salario del Empleado:", salario_recuperado)
```

Ejemplo ilustrativo

Sistemas de Mensajería: En aplicaciones de mensajería y comunicación en tiempo real, los objetos complejos, como mensajes multimedia o datos personalizados, a menudo se serializan antes de enviarse a través de la red y luego se deserializan en el extremo receptor.

Otras aplicaciones

Bases de datos, Sistemas de almacenamiento en la nube, sistemas de mensajeria, sistemas de cola de mensajes, Persistencia de Estado en Aplicaciones web, Sistemas de informes, Juegos y multimedia, entre otros.

Ventajas 

-Facilita el almacenamiento, la transferencia y la recuperación de objetos grandes y complejos

-Permite la persistencia de objetos grandes en bases de datos.

-Simplifica la gestión de objetos en la memoria.

Desventajas 

-Puede aumentar el tamaño de la base de datos.

-La serialización puede ser costosa en términos de rendimiento.

-Compatibilidad entre versiones puede ser un desafío.

Muchas gracias por la atención

https://www.youtube.com/watch?v=EX0_pIIjumM