Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2024

Proyecto 1 – Procesamiento de Tuplas

Carnet 201907177 – Byron Miguel Galicia

Resumen

Este ensayo aborda la problemática del alojamiento óptimo de objetos de bases de datos en sitios distribuidos, enfocándose en la minimización del costo de transmisión de datos para aplicaciones. Se exploran los métodos para diseñar esquemas de distribución que reduzcan estos costos, especialmente en problemas combinatorios NP-Hard. El análisis incluye la implementación de la Programación Orientada a Objetos (POO) en Python, la utilización de estructuras secuenciales y cíclicas, y la visualización mediante Graphviz. Se discuten diversas posturas sobre la eficiencia de diferentes métodos, incluyendo el agrupamiento de patrones de acceso. Las conclusiones destacan la relevancia de una adecuada implementación técnica y la visualización de datos para la optimización en contextos nacionales e internacionales.

Palabras clave

Alojamiento de datos, Distribución distribuida, Optimización NP-Hard, Programación Orientada a Objetos, Visualización Graphviz.

Abstract

This essay addresses the problem of optimal placement of database objects in distributed sites, focusing on minimizing data transmission costs for applications. It explores methods for designing distribution schemes that reduce these costs, particularly in NP-Hard combinatorial problems. The analysis includes the implementation of Object-Oriented Programming (OOP) in Python, the use of sequential and cyclic structures, and visualization through Graphviz. Various positions on the efficiency of different methods are discussed, including access pattern clustering. Conclusions highlight the relevance of proper technical implementation and data visualization for optimization in both national and international contexts.

Keywords

Data placement, Distributed distribution, NP-Hard optimization, Object-Oriented Programming, Graphviz visualization.

Introducción

La gestión eficiente del alojamiento de objetos de bases de datos en entornos distribuidos es crucial para la minimización de costos de transmisión y el rendimiento óptimo de aplicaciones. Este ensayo examina cómo la Programación Orientada a Objetos (POO) en Python y la visualización con Graphviz pueden abordar el problema de diseño de distribución, el cual se presenta como un desafío combinatorio NP-Hard. Se exploran las bases teóricas y prácticas del problema, incluyendo la necesidad de optimización y los métodos actuales para la resolución, proporcionando una perspectiva integral sobre su impacto técnico y económico.

Desarrollo del tema

1. Definición de Clases y Estructuras

Matriz y ValorMatriz

- **Matriz**: Representa una matriz con un nombre, número de filas y columnas, y una lista de valores.
- **ValorMatriz**: Representa un valor en una posición específica de la matriz, con fila, columna y el valor asociado.

Estructuras de Datos

- **Nodo y NodoValor**: Definen nodos para listas, donde Nodo contiene una referencia a una Matriz y un siguiente nodo, y NodoValor contiene un ValorMatriz y un siguiente nodo.
- **ListaCircular y ListaSimple**: Implementan listas circulares y simples para almacenar matrices y valores, respectivamente. La lista circular es útil para mantener una referencia continua a través de los nodos, mientras que la lista simple es más directa para operaciones lineales.

2. Operaciones y Métodos

Agregar y Recorrer Listas

- **ListaCircular.agregar()**: Agrega un nodo a la lista circular, creando un enlace continuo.
- **ListaCircular.recorre()**: Recorre la lista circular e imprime detalles de las matrices almacenadas, aunque el código para recorrer los valores no está completo.
- **ListaSimple.insertar()**: Inserta un valor en una lista simple.
- **ListaSimple.recorrer()**: Recorre y muestra los valores de una lista simple.

Comparación y Análisis de Filas

- **MatrizPatrones.comparar_filas()**: Compara las filas de la matriz para identificar filas repetidas. Utiliza listas simples para almacenar filas repetidas y analizadas.
- **MatrizPatrones.obtener_valores_fila()**: Extrae valores de una fila específica de la matriz.
- **MatrizPatrones.comparar_valores()**: Compara dos listas de valores para determinar si son iguales.

Suma y Reducción de Filas

- **sumar_filas()**: Suma los valores de las filas especificadas en la matriz. Sin embargo, el código para realizar la suma y gestionar los valores parece estar comentado y necesita implementación.
- **crear_matriz_reducida()**: Crea una nueva matriz reducida que suma las filas especificadas y reorganiza las filas restantes.

Combinación de Valores de Matrices

- **Prueba()**: Combina valores de una matriz y filas repetidas para crear una nueva matriz combinada. El método incluye la creación de una lista combinada de valores y la determinación del tamaño de la nueva matriz.

3. Generación de Representaciones Gráficas

Generación de DOT

- **generar_dot()**: Crea un archivo DOT que representa las matrices en formato gráfico. Se generan tablas para mostrar los valores de las matrices y sus patrones, proporcionando una representación visual útil para el análisis.

4. Consideraciones y Mejoras

1. **Completar Métodos**: Algunos métodos, como ListaCircular.recorre() y sumar_filas(), tienen fragmentos de código comentados o incompletos. Estos deben ser completados para que la funcionalidad sea completa.
2. **Optimización**: El código para verificar si un valor está en una lista combinada (Prueba()) podría optimizarse. Considerar el uso de estructuras más eficientes si se permite en el futuro.
3. **Documentación y Validación**: Asegúrate de que cada método tenga comentarios claros y realiza pruebas exhaustivas para validar que el código funciona correctamente en todos los casos.
4. **Manejo de Errores**: Implementar manejo de errores para asegurar que el código sea robusto ante entradas inválidas o casos inesperados.

Conclusiones

El ensayo concluye que la combinación de POO en Python y la visualización con Graphviz ofrece una solución efectiva para el problema de alojamiento de datos en sitios distribuidos. La implementación adecuada de estas herramientas no solo optimiza el proceso de distribución, sino que también mejora la comprensión del patrón de acceso a los datos. Se recomienda profundizar en la aplicación de metodologías de agrupamiento y la integración de técnicas avanzadas de visualización para futuros desarrollos en el área.

Referencias bibliográficas

C. J. Date, (1991). *An introduction to Database Systems.* Addison-Wesley Publishing Company, Inc.