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

ENSAYO DE PROYECTO 1

202302220 – Enner Esaí Mendizabal Castro

Resumen

En la actualidad, el almacenamiento de información en diversas bases de datos se ha vuelto muy popular por las ventajas que este brinda. No obstante, esto conlleva un incremento en el costo de transmisión debido a la necesidad de comunicación continua entre servidores. El proyecto desarrollado aborda esta problemática utilizando listas circulares simplemente enlazadas, implementadas a través de la Programación Orientada a Objetos (POO), para almacenar temporalmente la información de un archivo de entrada, procesarla y, con estos datos, generar un archivo de salida en formato .xml, manteniendo la estructura del archivo original .xml, así como una representación gráfica de la matriz reducida con ayuda de las bibliotecas ElementTree y graphviz. Las listas generadas manualmente mediante POO facilitan un código más claro, estructurado y seguro, al permitir la creación de funciones específicas para cada una de las necesidades del programa, que pueden ejecutarse desde el backend y retornar únicamente lo requerido.

Palabras clave

Programación Orientada a Objetos POO), listas circulares simplemente enlazadas, base de datos, matriz.

Abstract

Nowadays, storing information in various databases has become extremely popular due to the advantages it provides. However, this leads to an increase in the transmission cost due to the need for continuous communication between servers. The developed project addresses this problem by using simply linked circular lists, implemented through Object Oriented Programming (OOP), to temporarily store the information from an input file, process it and, with this data, generate an output file in .xml format, maintaining the structure of the original .xml file, as well as a graphical representation of the reduced matrix with the help of the ElementTree and graphviz libraries. The lists generated manually through OOP facilitate a clearer, more structured and secure code, by allowing the creation of specific functions for each of the program's needs, which can be executed from the backend and return only what is required.

Keywords

Programación Orientada a Objetos POO), listas circulares simplemente enlazadas, base de datos, matriz.

Introducción

El alojamiento de objetos en distintas bases de datos tiene un costo inherente en su transmisión, por tal motivo es necesario que se pueda usar una metodología de agrupamiento para reducir la cantidad de datos que se transmite y minimizar los costos de transmisión

En el presente trabajo se pretende mostrar la manera en la cual se resolvió este problema mediante el uso del lenguaje de programación de alto nivel *phyton*, el cual, teniendo de entrada un documento con extensión *.xml*, logra leerlo, guardarlo mediante listas creadas manualmente con POO y apuntadores, consiguiendo de esta forma que se pueda realizar el proceso de agrupamiento, y crear el nuevo archivo de salida ya agrupado en pdf utilizando nodos con graphviz.

A continuación, se verán todos los procesos y problemas que se realizaron y surgieron durante la creación del programa y la manera en la que se resolvieron.

Desarrollo del tema

El proyecto 1 de Introducción a la Programación y Computación 2 trataba sobre la transmisión de información en distintas bases de datos en sitios distribuidos.

Almacenamiento distribuido

Este tipo de almacenamiento, el almacenamiento distribuido, es un tipo de almacenamiento en donde la información se guarda en varios en distintos lugares gracias a la red de maquinas conectadas o mecanismos de repositorios. (HostZealot Team, 2024)

Con este mecanismo, los datos se dividen en distintas partes, las cuales están divididas dentro cada uno de los dispositivos en la red, consiguiente que con esta descentralización que:

- Se pueda acceder simultáneamente a las distintas partes de la información que se busca.
- Mayor velocidad para cargar la información
- Mejores mecanismos de recuperación y copias de seguridad.

Actualmente es de los sistemas que más se están usando debido a que son más baratos cuando se tiene un bajo presupuesto, debido a que requieren de un hardware más básico, son más escalables y pueden trabajar con más cantidad de datos. (HostZealot Team, 2024)

A pesar de todas las ventajas que poseen estos sistemas, tienen el inconveniente de tener algunos problemas de seguridad, debido a que los dispositivos necesitan estar en constante comunicación mediante la red y, principalmente, existe costo de accesos, los cuales dependen del tipo y la cantidad de información que se transmita.

Figura 1. Computadoras transmitiendo información a un tipo de almacenamiento distribuido.

Fuente: Cristian.j. 24/7 tecno, 2019, 247tecno.com

Sobre el problema de la cantidad de información a transmitir es para el cual se trabaja en el proyecto 1.

Se pretendía lograr la manera en la cual la información que se transmita sea la mínima para que, de esta forma, se logren minimizar los costos de transmisión entre los dispositivos conectados en la red.

Para lograr minimizar los costos se ideó una manera de obtener un nuevo esquema replicado adaptado a partir del de la base de datos, pero con menor cantidad de información.

Para esta minimización de la información, se aplicó una metodología de agrupamiento, para una cierta cantidad de tuplas y una cierta cantidad de sitios, es decir, se obtendría la matriz de frecuencias de acceso, la cual podría tener cualquier cantidad de filas y columnas, esta se transformaría en una matriz de patrones de acceso, con la cual se agruparían aquellas duplas con el mismo patrón para obtener así una matriz con una menor cantidad de datos.

Solución del problema de almacenamiento distribuido

Para solucionar el problema del almacenamiento distribuido y la disminución de costos de transmisión, se creó un programa con el lenguaje de programación de alto nivel *phyton* con el cual se realizaría esa reducción.

Primeramente, se comenzó con la creación de las clases, específicamente una lista circular simplemente enlazada, con esta se guardó temporalmente cada uno de los valores que se encontraban en el archivo de entrada, es decir, no se usarían en esta ocasión las listas previamente creadas y las cuales ofrece *phyton* de manera nativa, se usarían unas listas creadas usando la memoria dinámica y la programación orientada a objetos, con conceptos como los apuntadores y los nodos.

Listas circulares simplemente enlazadas

Para la creación de la lista circular enlazada se comenzó creando un nodo, este poseería como atributos el dato que contendría y el siguiente y anterior datos, de tal forma que se fueran enlazando.

```
1  class nodo:  # El nodito
2      def __init__(self, dato, siguiente=None, anterior=None):
3          self.dato = dato  # Datos
4          self.siguiente = siguiente  # apuntadores
5          self.anterior = anterior
```

Figura 2. Código para crear una clase de tipo nodo para una lista circular.

Fuente: Elaboración propia.

Ahora bien, una vez teniendo el nodo sobre el cual se manejaría la lista, puesto a que este nodo sería de cierta forma y ejemplificando demasiado una “celda” en la cual se contendría la información y la cual sabría cuáles serían su “celda” adyacente.

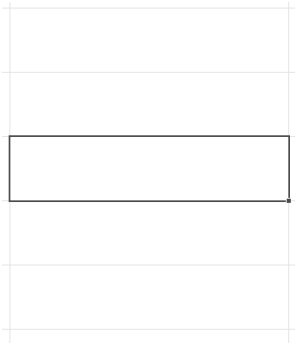

Figura 3. Visualización de cómo sería un nodo mediante el programa de Excel.

Fuente: Elaboración propia.

Una vez creada esta, clase sobre la cual se guardaría la información, se procedió con la creación de la una clase la cual contendría todos los nodos, es decir, continuando con la analogía de las celdas, esta sería la fila en la cual están contenidas las celdas

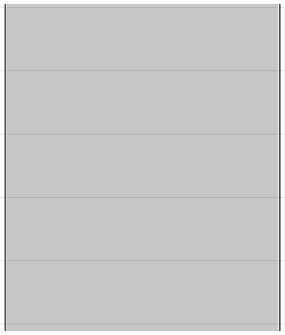

Figura 4. Visualización de cómo sería una lista mediante el programa de Excel.

Fuente: Elaboración propia.

Por último, dado a que se trabajaría con matrices, se creó otra clase, la cual contendría listas, de tal manera que se terminara añadiendo una nueva "dimensión" a la matriz que estamos creando con nodos y listas, quedando de tal forma que se pueda tener de manera abstracta una matriz, recordando siempre que se utilizó una lista circular simplemente enlazada, por lo tanto, el último nodo sería seguido por el primero de la lista, lo cual haría que siempre estuviera conectada.

Finalmente, se le añadió a cada una sus distintas funciones para que se pudieran realizar las acciones que se requirieran como eliminar o añadir valores, filas, buscar valores dependiendo de la posición en la que se encuentre cada uno de los valores, etc.

Creación del menú

Claramente, este programa requiere que se pueda interactuar en cierta medida con el usuario, de tal forma que esta pueda indicar lo que se quiere realizar y cuándo, para eso se creó un simple ciclo el cual mostraría las opciones y solicitaría que se ingrese una opción.

Reducción de matriz

Una vez con el menú, finalmente se llega con la reducción de la matriz.

Primeramente, se tomó el valor de entrada en la dirección que indicara el usuario para que se leyera mediante un ciclo que crearía una matriz con valores vacíos dentro de sus nodos para que, a posterior, en el proceso de llenado de la matriz, se rellenen con todos los valores de la matriz de entrada.

Es conveniente mencionar que para la ayuda de este proceso se usó la librería llamada *ElementTree* para trabajar más fácilmente con los archivos de entrada *.xml* y también se usó la librería *copy* para clonar los valores que contienen las matrices, dado a que *phyton*, por la manera en que trabaja con las variables, no permite esto de manera simple como lo harían otros lenguajes de programación como puede ser java.

Una vez que se crearon todas las matrices con los valores del archivo de entrada, se trabajó con los valores para transformar una de estas copias en una matriz con patrones de accesos mediante un ciclo que detectaba los accesos para modificar los valores con "unos" y "ceros" dependiendo si posee entradas o no.

Terminada la matriz de valores de entrada, se realizó la matriz reducida utilizando los valores de la matriz de entrada mediante una comparación de cada una de las filas y, si detectaba igualdad en los patrones de acceso, sumaba estas filas y las eliminaba, guardando información de las repeticiones que se hicieron para hacer este proceso y las filas en las cuales se encontraban las igualdades.

Creación del archivo de salida

Si se procesa el archivo, es necesario que se genere una salida, puesto a que debe entregar resultados, por tal motivo, usando la misma librería que se utilizó para la entrada, se creó la salida, generando un archivo *.xml*, con la matriz reducida.

Creación de gráfica

Para la generación de gráfica, se utilizó la librería *graphviz* esta permitiría que se pudieran crear diagrama con nodos y exportarlos a pdf para que se pudieran visualizar más claramente. Simplemente se creó un ciclo el cual recorrería las matrices ya guardadas de manera temporal dentro de las matrices creadas con POO y se asignarían los valores en los nodos que se irían creando y conectando en sus posiciones respectivas, finalizando de esta forma todo lo requerido para este proyecto realizado.

Conclusiones

El almacenamiento distribuido es un tipo de almacenamiento de información que ha adquirido popularidad en la actualidad debido a las grandes ventajas que presenta, pero tiene la desventaja de que se debe transmitir mucha información entre los dispositivos de la red, lo que genera costos de transmisión muy elevados.

Las listas que poseen cierto dinamismo y que vienen incluidas en ciertos lenguajes de programación como podrían ser phyton, se crean gracias a la lógica de POO usando nodos con apuntadores, programándolas manualmente se pueden crear funciones para estas dependiendo de las necesidades del programa, consiguiendo un código más limpio y ordenado.

Referencias bibliográficas

HostZealot Team. (04 de abril de 2024). *Desvelando el mundo del almacenamiento distribuido: Tipos y ejemplos reales*. Obtenido de hostzealot: https://es.hostzealot.com/blog/about-servers/desvelando-el-mundo-del-almacenamiento-distribuido-tipos-y-ejemplos-reales#:~:text=El%20almacenamiento%20distribuido%20es%20un%20tipo%20de%20marco,red%20de%20m%C3%A1quinas%20conectadas%20o%20mecanismos%20de%20r

J., C. (17 de febrero de 2019). *Qué es una unidad de almacenamiento*. Obtenido de 24/7 tecno: https://247tecno.com/wp-content/uploads/2017/09/almacenar.png

Apéndice

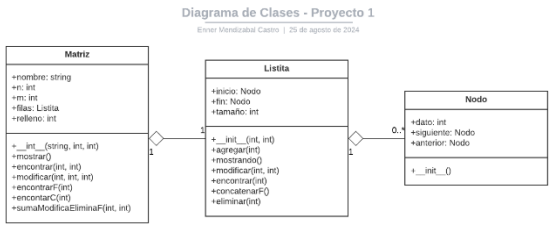

Figura 5. Diagrama de clases del proyecto

Fuente: Elaboración propia.

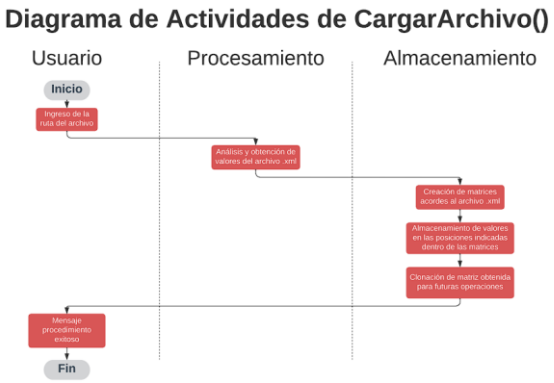

Figura 6. Diagrama de clases de la función dedicada a la cargar del archivo de entrada

Fuente: Elaboración propia.

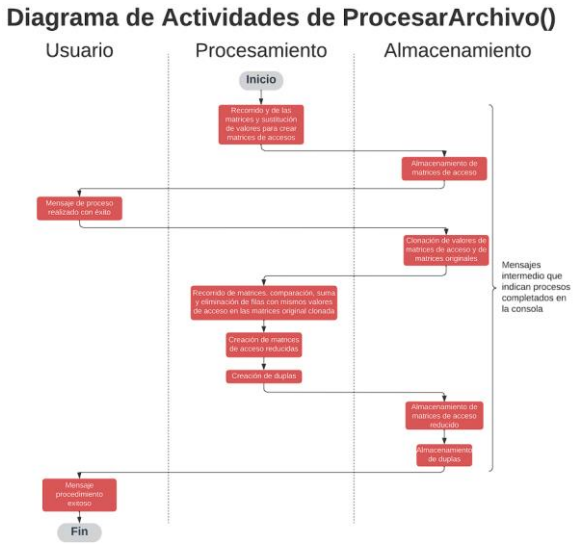

Figura 7. Diagrama de clases de la función dedicada a procesar archivos el archivo de entrada

Fuente: Elaboración propia.

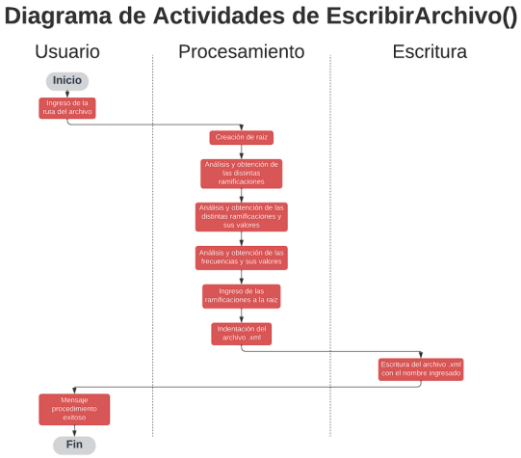

Figura 8. Diagrama de clases de la función dedicada a escribir el archivo ya procesado en formato xml

Fuente: Elaboración propia.

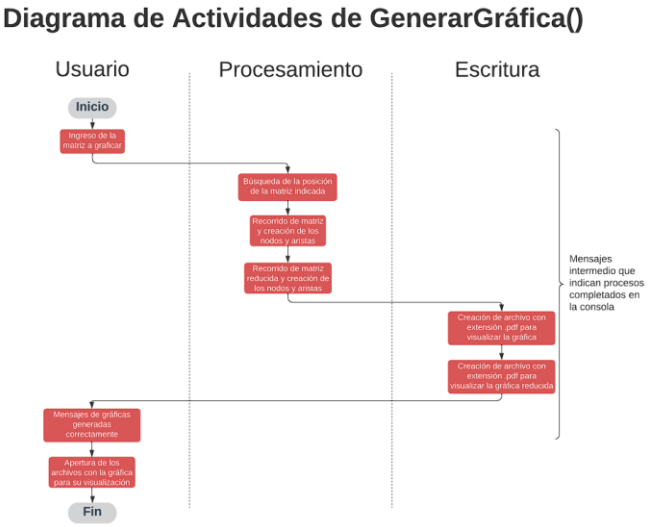

Figura 9. Diagrama de clases de la función dedicada a generar las gráficas de la gráfica ya reducida.

Fuente: Elaboración propia.