Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2021.

REDUCCION DE MATRICES POR FREQUENCIA

201612174 – Alberto Gabriel Reyes Ning

Resumen

Escribe un programa de texto que se calcula en forma matriz la forma de alojar objetos de base de datos en sitios distribuidos de manera que el costo total de la transmisión de datos para el procesamiento de todas las aplicaciones sea minimizado.
Si los bases de datos se acceden con la misma frecuencia binaria, se puede reducir el costo utilizando solo uno de los varios bases de datos solo si tienen la misma frecuencia binaria. El programa se escribe un nuevo esquema en forma de una matriz reducida de acceso replicando de alojamiento que se adapte a un nuevo patrón de uso de la base de datos y minimice los costos de transmisión.
Este programa se ayuda en minimizar costos y tiempo utilizado en ejecutar consultas a los bases de datos de forma exponencial debido al tamaño de la red de computadores. El programa también se puede mostrar en forma gráfica los consultas hecho a los bases de datos o sitios hecho.

Palabras clave
Listas Circulares, Listas Simples, Dinámica, Python, Matriz

Abstract

Write a text-based program that calculates through matrixes a way of accommodating objects from databases from distributed sites in such a manner that the total cost of the transmission of data for the procedures of all the applications are minimized.

If the databases are accessed with the same frequency in binary format, the total cost can be reduced by utilizing only one of the databases given that it has the same binary frequency. The program writes a new scheme in the form of a reduced access matrix replicating the accommodation that the patron of usage adapts for the database while minimizing the costs of transmission.

This program helps in minimizing the costs and time utilized in executing queries to the databases in an exponential format based upon the size of the network. The program can also help in displaying in graphical format the queries that are transmitted to the databases or distributed sites.

Keywords
Circular Linked Lists, Simple Linked Lists, Dynamic, Python, Matrix

Introducción

Crear un software el cual se debe ser de interfaz de línea de comandos y fácil de utilizar. El programa debe ser capaz de cargar un archivo XML y procesarlo en forma matriz.

Dicha aplicación se recibe un archivo XML que contiene información sobre la frecuencia de acceso a un base de datos en sitios distribuidos. El usuario se puede elegir la ruta donde está el archivo XML para cargar y se puede elegir a procesarlo, sacar un archivo XML después de procesar los datos o generar una gráfica sobre los datos en forma matriz utilizando GraphViz.

La aplicación se entregará al inicio del mes de marzo y se realizará en Visual Studio Code 1.54.1 en lenguaje Python 3.8.1.

Desarrollo del tema

A. Funciones del Sistema

1. Cargar Archivo
2. Procesar Archivo
3. Escribir Archivo Salida
4. Mostrar datos del estudiante
5. Generar Grafica
6. Salida

B. Explicación del Programa

El programa al iniciar se muestra un menú principal al usuario donde el usuario se puede entrar un numero entre 1 a 6. Si el usuario se entra un numero afuera del rango, el programa se muestra un error y regresa al menú principal.

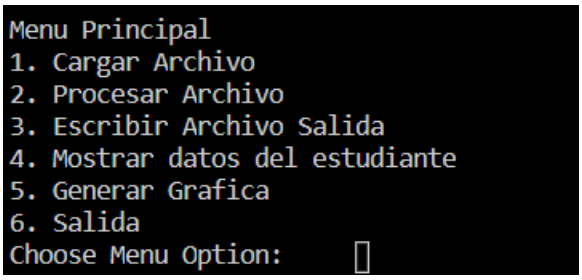

Figura 1. Opción 1

Fuente: elaboración propia

-Si el usuario elige la opción 1: Cargar Archivo

El programa se pide la ruta donde esta el archivo de entrada. El usuario ingrese la ruta y el programa se cargar el archivo en memoria en listas circulares.

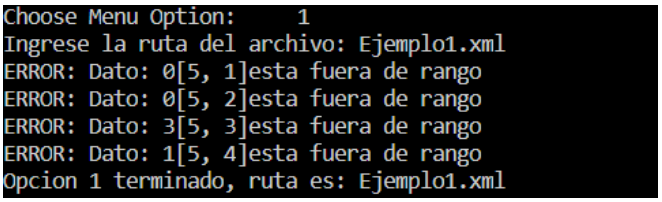

Figura 1. Opción 2

Fuente: elaboración propia

-Si el usuario elige la opción 2: Procesar Archivo

El programa se procesa los datos cargados en memoria. Si no hay datos cargados, la función ejecuta y regresa al menú principal regresando un matriz nulo. Si hay datos cargados, la función ejecuta comparando las filas y reduciendo el matriz regresando un matriz de acceso reducido.

```
Choose Menu Option:     2

REDUCIENDO: Ejemplo

Comparando la fila: 1 con las otras filas.
Sumando Fila: 1 == Fila: 3
Comparando la fila: 2 con las otras filas.
Comparando la fila: 3 con las otras filas.
Comparando la fila: 4 con las otras filas.

REDUCIENDO: Ejemplo2
```

Figura 3. Opción 1

Fuente: elaboración propia

-Si el usuario elige la opción 3: Escribir Archivo Salida

El programa se utiliza los datos de la matriz procesada y utilizando ElementTree, lo pasa a un archivo XML. Si los datos no están procesados, la función regresa los datos de entrada. Si no hay datos cargados, la función regresa un XML nulo. Si hay datos cargados y procesados, la función regresa un XML con los datos procesados a la ruta escrita y regresa al menú principal.

```
Choose Menu Option:     3
Escribir una ruta especifica: file.xml
Se escribio el archivo satisfactorio
```

Figura 4. Opción 1

Fuente: elaboración propia

-Si el usuario elige la opción 4: Mostrar datos del estudiante

El programa se muestra los datos del estudiante y regresa al menú principal.

```
Choose Menu Option:     4

Alberto Gabriel Reyes Ning
201612174
IPC2-A
Ingenieria en Ciencias y Sistemas
4to Semestre
```

Figura 5. Opción 1

Fuente: elaboración propia

-Si el usuario elige la opción 5: Generar Grafica

El programa se muestra una lista de las matrices de entrada donde el usuario se puede elegir uno para graficar. Utilizando GraphViz, el programa se genera una gráfica de Nodos en forma matriz y regresa al menú principal.

```
Choose Menu Option:     5
1: Ejemplo
2: Ejemplo2

Choose List: 2
```

Figura 6. Opción 1

Fuente: elaboración propia

-Si el usuario elige la opción 6: Salida

El programa se termina de ejecutar.

```
Choose Menu Option:     6
Saliendo del applicacion
```

Figura 7. Opción 1

Fuente: elaboración propia

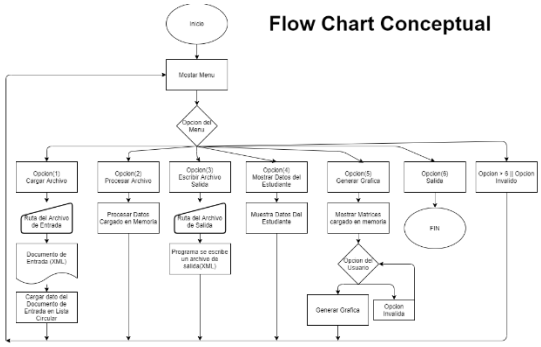

Figura 8. Flow Chart del Programa Escrito

Fuente: elaboración propia

C. Atributos del Sistema

-Aplicación de Texto desarrollada en Visual Studio Code

-Cargar Archivos XML por Elementtree

-Graficar Graficas por GraphViz

-Programación Orientado a Objetos

-Utilización de Listas Dinámicas (Listas Circulares y Listas Simples)

*Listas Circulares

*Listas Simples

*POO

D. Relación entre Entidades

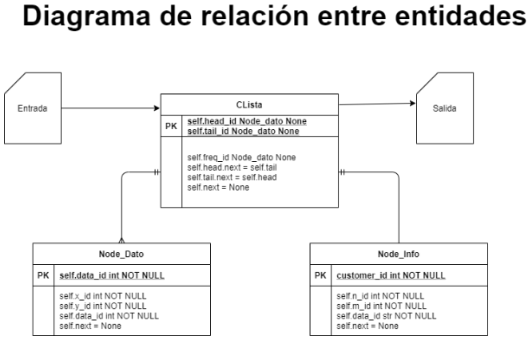

Figura 9. Diagrama de relación entre entidades en el programa.

Fuente: elaboración propia.

Cuando el programa se carga un archivo en memoria, los datos del archivo están Guardado en un Clase CLista que contiene dos Nodos que son respectivamente Nodo_Dato y Nodo_Info.

La clase CLista contiene al inicio un head y tail que son de la clase Nodo_Dato. Al agregar información al CLista, se crea un nuevo nodo Nodo_Dato y lo agrega a la lista circular basado en su posición (x, y). Después de agregar todos los datos, se agrega un nuevo Nodo_Info al inicio de la lista circular en el posición self.head que contiene información del nombre del matriz y su tamaño.

La clase CLista es solo uno a varios con la clase Nodo_Dato por que una lista circular puede contener varios Nodos de Nodo_dato y esos Nodos solo se puede relacionar con una lista circular. La clase CLista es solo uno a uno con Nodo_Info porque cada lista circular solo puede contener un Nodo_Info que contiene información de la matriz y ese Nodo solo se puede relacionar con una lista circular.

E. TDA Utilizado

Lista Simple

Una lista simple es una lista linear donde los elementos no están guardados en ubicaciones de memoria contiguas. Estas listas son dinámicas debido que se guarda los elementos en memoria utilizando apuntes donde cada nodo se apunta al siguiente nodo o a None. Listas simples son utilizados por su eficiencia en agregando y borrando información cuando él información se agrega/borra al inicio/fin de la lista.

Lista Circular

Una lista circular es una variación de una lista donde el fin de la lista siempre se apunta al inicio de la lista. Las ventajas de utilizar una lista circular en vez de una lista simple aparecen en operaciones que necesitan el uso repetido de una lista en forma circular.

F. TDA Explicación

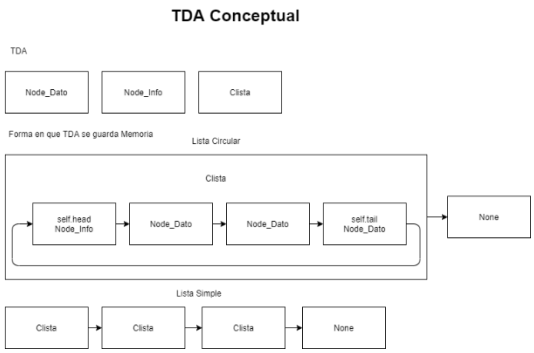

Figura 10. Concepto de cómo se guardo los datos en el programa.

Fuente: elaboración propia.

El Nodo Nodo_Dato y Nodo_Info contiene cuatro variables donde tres son utilizados para guardar información y el variable next se utiliza para apuntar al siguiente Nodo.

La clase CLista contiene cuatro variables donde dos head y tail, son utilizados para anotar las posiciones de los nodos para crear un Lista Circular donde al inicio head se apunta al tail y tail se apunta a head. La agregación de next fue para crear una Lista Simple sin el uso de más Nodos donde casa Lista Circular se apunta al siguiente Lista Circular hasta que termina el ciclo. La variable freq se utiliza para guardar información de la frecuencia de cada fila y en forma de Lista Simple donde su next siempre apunta a la siguiente frecuencia o a None.

Conclusiones

El uso de listas circulares ayudó con la carga de memoria y facilito borrar datos ya que se podría agregar y borrar dinámicamente la lista. Las listas simples ayudaron en guardar los varios matrices dinámicamente.

En este problema, aunque la lista circular si ayudo en cargar los archivos a memoria, no fue eficiente por que se tenía que atravesar la lista cada vez para verificar los datos o buscar un nodo. El programa se podría ser más eficiente con la utilización de las listas incluidos en Python que te deja apuntar donde quieres verificar la información utilizando [x][y].

Referencias bibliográficas

Howson, S. (2018, 6 marzo). *Python XML with ElementTree: Beginner's Guide*. www.datacamp.com. https://www.datacamp.com/community/tutorials/python-xml-elementtree

User Guide — GraphViz 0.16 documentación. (s. f.). graphviz.readthedocs.io. Recuperado 5 de marzo de 2021, de https://graphviz.readthedocs.io/en/stable/manual.html

ANEXO

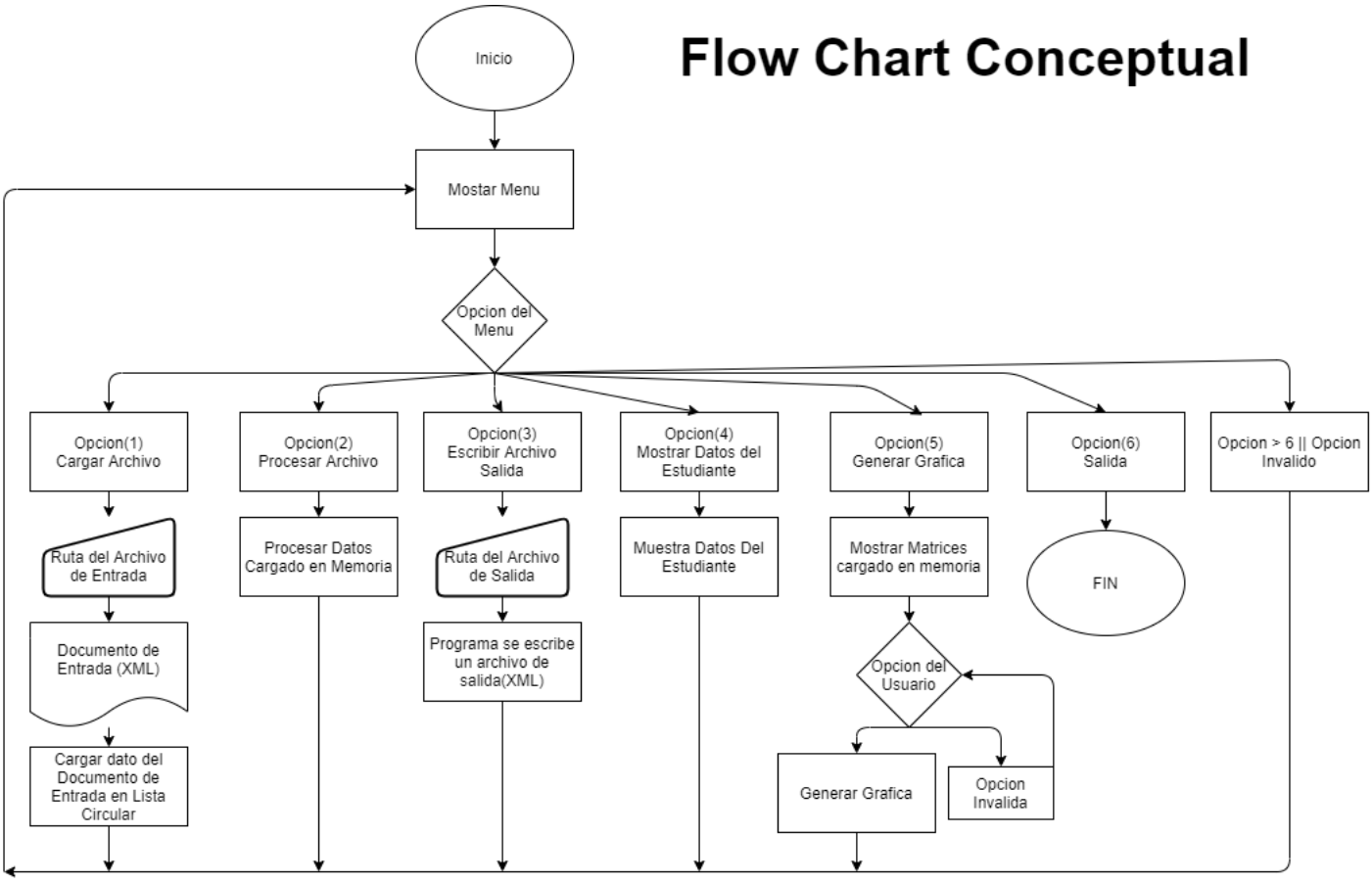

Figura 8. Flow Chart del Programa Escrito

Fuente: elaboración propia

Diagrama de relación entre entidades

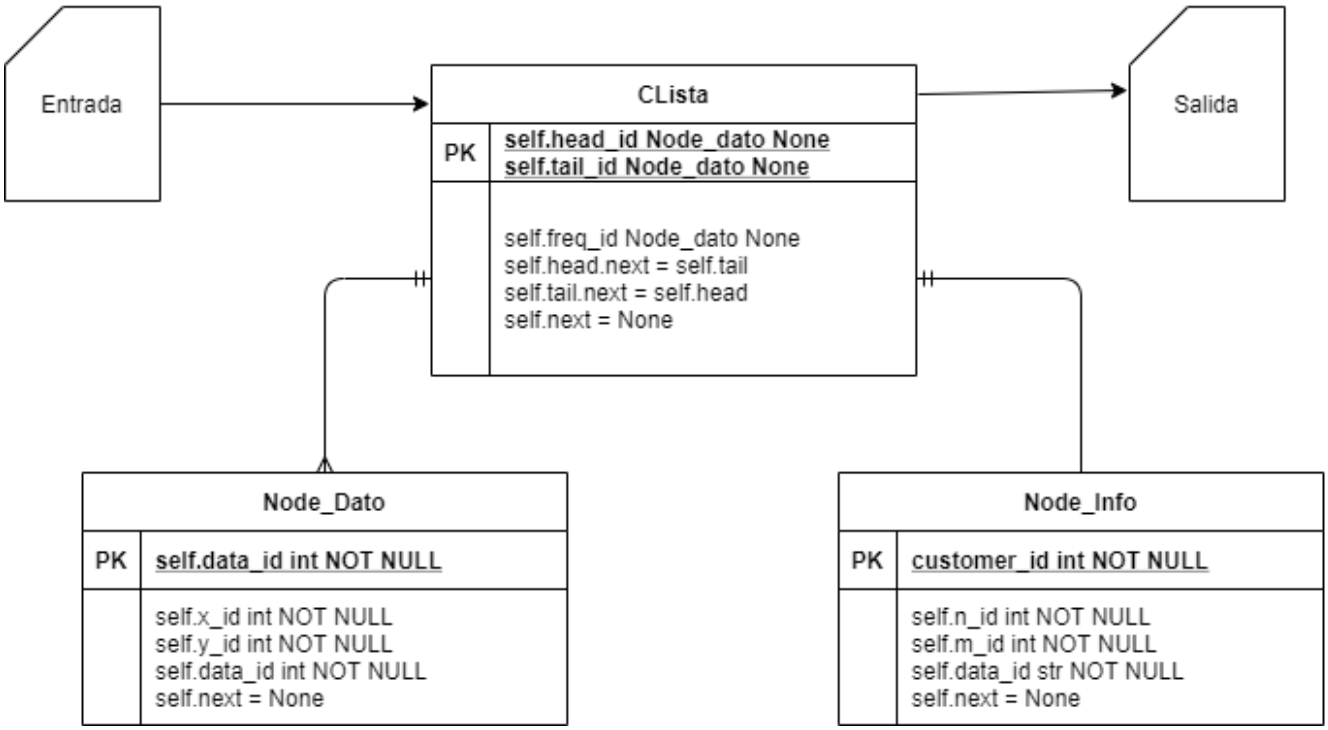

Figura 9. Diagrama de relación entre entidades en el programa.

Fuente: elaboración propia.

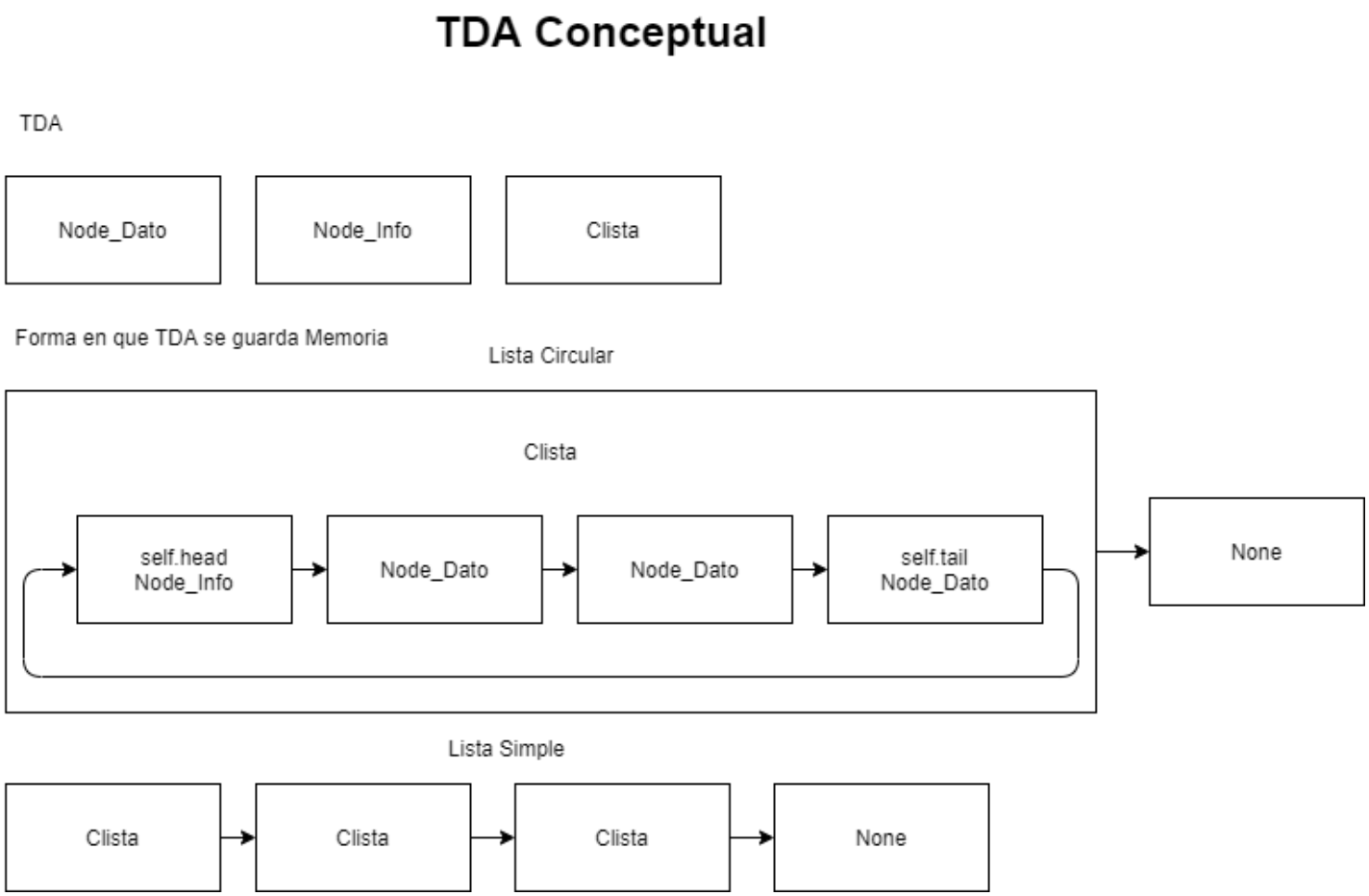

Figura 10. Concepto de cómo se guardó los datos en el programa.

Fuente: elaboración propia.