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.

PROYECTO 1

201903974 – Steven Josue González Monroy

Resumen

El proyecto consistió en dar una solución integral, implementando tipos de datos abstractos, a el alojamiento de objetos de bases de datos en sitios, de manera que los costos totales de transmisión de datos para el procesamiento de todas las comunicaciones cliente-servidor sean mínimos.

El programa desarrollado para dar solución al problema fue desarrollado en Python, y consistió en la lectura y procesamiento de archivos en formato XML que contienen matrices con los datos a minimizar, una vez procesado el archivo los datos son almacenados en memoria y procesados, de igual manera permite generar un archivo XML con las matrices de datos ya reducidas, así como permite su graficación por medio de la herramienta Graphviz para visualizar los datos finales con más facilidad.

Se utilizaron listas enlazadas como estructuras de datos abstractas para que el consumo de memoria sea dinámico, por lo tanto, el programa resulta útil para minimizar la carga en las bases de datos, disminuyendo la cantidad de memoria dinámica utilizada y agilizando la comunicación con los usuarios.

Palabras clave

-TDA
-Listas
-Nodos
-Matrices
-Estructura

Abstract

The project consists of providing a comprehensive solution, implementing abstract data types, to the hosting of database objects in sites, so that the total data transmission costs for the processing of all client-server communications are minimal.

The program developed to solve the problem was developed in Python and consisted of reading and processing files in XML format that contain matrices with the data to be minimized, once the file is processed the data is stored in memory and processed, in the same way In this way, it allows generating an XML file with the data matrices already reduced, as well as allowing their graphing using the Graphviz tool to visualize the final data more easily.

Linked lists were used as abstract data structures to make memory consumption dynamic, therefore, the resulting program to minimize the load on databases, reducing the amount of dynamic memory used and speeding up communication with users.

Keywords

-ADT
-Lists
-Nodes
-Matrix
-Estructure

Introducción

Un servidor de base de datos es un programa que provee servicios de base de datos a otros programas u otras computadoras, como es definido por el modelo cliente-servidor. Las bases de datos están situadas en un servidor y se puede acceder a ellas desde terminales o equipos con un programa -llamado cliente- que permita el acceso a la base o bases de datos.

Un objeto de base de datos es una entidad de una base de datos, esta entidad puede ser un atributo, un set de tuplas, una relación o un archivo. Los objetos de base de datos son unidades independientes que deben ser alojadas en los sitios de una red.

El programa desarrollado tiene como objetivo reducir la cantidad de objetos de base de datos que son enviados sin perder información, para disminuir la carga del servidor, agilizando y minimizando la transmisión de datos.

Desarrollo del tema

a). Problema:

Este problema consiste en alojar objetos de bases 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. Un objeto de base de datos es una entidad de una base de datos, esta entidad puede ser un atributo, un set de tuplas, una relación o un archivo. Los objetos de base de datos son unidades independientes que deben ser alojadas en los sitios de una red.

Una definición formal del problema se presenta en la figura No. 1.

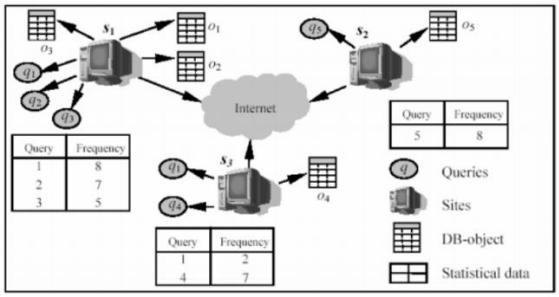

Figura 1. Problema de diseño de distribución en una base de datos.

Fuente: Enunciado Proyecto 1, 2021, pág. 2.

La figura No. 1 muestra un set de objetos de bases de datos O = {o 1 , o 2 , …, o no }, una red de computadoras que consiste en un set de sitios S = {s 1 , s 2 , …, s ns }, donde un set de consultas Q = {q 1 , q 2 , … q nq } son ejecutadas, los objetos de base de datos requeridos por cada consulta, un esquema inicial de alojamiento de objetos de bases de datos, y las frecuencias de acceso de cada consulta desde cada sitio en un período de tiempo. El problema consiste en obtener un nuevo esquema replicado de alojamiento que se adapte a un nuevo patrón de uso de base de datos y minimice los costos de transmisión.

Se debe diseñar un programa que acepte “n” matrices de frecuencia de accesos y por cada una obtener los grupos formados por las tuplas con el mismo patrón de acceso. Finalmente, se debe obtener la matriz de frecuencia de acceso reducida.

b) Codificación:

El software fue codificado en lenguaje Python, aplicando paradigmas de programación orientada a objetos, paradigma funcional, descriptivo y estructurado.

Consta de 9 clases las cuales son:

- Main.py
- Carga.py

- Procesamiento.py
- nodoDato.py
- nodoFrecuencias.py
- nodoListaMatrices.py
- listaDatos.py
- listaFrecuencias.py
- listaMatrices.py

b.1) Main:

Esta clase contiene el código con el cual arranca el programa, consiste en un ciclo while que muestra al usuario el menú principal de la aplicación. Desde aquí el usuario elegirá que desea hacer y el programa por medio del método input() obtendrá su elección, y ejecutará los métodos provenientes de otras clases de acuerdo a la elección del usuario.

b.2) Carga:

Esta clase contiene un único método el cual obtiene la dirección del archivo ingresada manualmente por el usuario. Verificará que no se tenga cargado ningún archivo XML, si esto es verdadero se creará un elementTree con los datos del XML introducido, de lo contrario si las matrices contienen datos de un archivo previamente cargado procede a ejecutar un método propio de las listas que las vacía para luego crear el elemento y trabajar con el archivo actual, una vez creado el árbol se procede a retornar el elemento para su análisis.

b.3) Procesamiento:

Esta clase contiene métodos que procesan los datos que se cargan en los árboles XML:

1. validarDimensiones:
   Consiste en verificar que la cantidad de elementos pertenecientes a cada matriz concuerde con las dimensiones establecidas en sus atributos.

2. crearLista:
   Una vez validadas las dimensiones con la ayuda de los métodos propios de elementTree se procede a agregar datos a una lista enlazada y automáticamente se procede a generar una lista con las matrices reducidas.

3. generarXML:
   Una vez procesado el archivo de entrada XML se podrá generar la grafica por medio de este método, consiste en concatenar la estructura básica con los datos de la matriz que se desee para finalmente tener una única variable string que contendrá el código dot para que graphviz pueda generar una gráfica y finalmente mostrarla con el método view(), el cual de manera automática abrirá la imagen con la gráfica creada.

4. generarXML:
   este método únicamente invoca al método generarXML() propio de la lista de matrices reducidas.

b.4) nodoDato:

Esta clase contiene la codificación del objeto nodo para las listas de datos, a partir de esta clase se inicia a trabajar con el paradigma de Programación Orientada a Objetos (POO). Únicamente consiste en un constructor con sus atributos, para este objeto

nodo los atributos son dato, x (posición x), y(posición y), identidad y por ultimo siguiente que será otro objeto del mismo tipo nodo

b.5) nodoFrecuencias:

Esta clase es similar a la clase anterior sin embargo esta esta diseñada exclusivamente para listas que almacenaran las frecuencias con que se repiten los patrones de filas en las matrices. Esta clase únicamente contiene un constructor con los siguientes dos atributos: frecuencia y grupo

Donde frecuencia será el numero de veces que se repitió el patrón, y grupo será el grupo de filas al que le corresponde la frecuencia.

b.6) nodoListaMatrices:

Al igual que las otras dos clases nodo esta consiste en un constructor y sus atributos, los atributos de este objeto son: nombre, n (dimensión n), m(dimensión m), datos (lista), frecuencias(lista).

A diferencia de los otros dos nodos, este nodo contendrá como atributos otras listas.

Asi mismo este nodo contendrá un el método generarTabla(), el cual será utilizado cuando se requiera graficar la matriz, esta concatenara los datos de la lista del atributo datos y los concatenara con las etiquetas <td> y <tr> de html para finalmente retornar un string con la estructura de la tabla para graficarla.

b.7) listaDatos:

Esta clase contiene los métodos para la formación de listas enlazadas simples, los métodos utilizados son:

- **add()**:este método se encarga de crear un nuevo nodo, verifica que la lista este vacía o si contiene nodos para así ubicar el nuevo nodo al final.
- **length():** consiste en un contador que aumenta cada vez que se encuentra un nuevo nodo, y se detiene hasta que el siguiente nodo sea nulo, finalmente retorna el valor del contador.
- **toString():** este método únicamente fue creado para llevar un registro de los datos que se iban agregando a las listas, sin embargo el usuario final no lo utiliza en ningún momento, consiste en imprimir en consola el valor de cada dato que contenga la lista.
- **getDato():** este método consiste en realizar una búsqueda, para ello se recibe como parámetro la posición x y y del dato que se desea buscar, luego este método realiza comparaciones entra cada nodo hasta encontrar una coincidencia, finalmente si hay coincidencia retorna retorna el atributo dato del nodo actual, de lo contrario retorna nulo.
- **getIdentidad():** este método es similar al anterior la diferencia consiste en que este método al encontrar una coincidencia retorna el valor de la identidad que le corresponde al dato, ya sea 0 o 1, si no encuentra coincidencias retorna nulo.
- **eliminar():** este método consiste en recorrer los nodos de la lista hasta encontrar el nodo que coincida con los valores x y y buscados. Al realizarlo los valores de los nodos anteriores se corren una posición y se establece al nodo de la coincidencia como nulo.
- **setDato():** Este método consiste en buscar un nodo comparándolos por su posición x y y, cuando encuentra una coincidencia reasigna un nuevo valor al atributo dato del nodo encontrado.
- **XML():** este método recibe como parámetro un árbol, proveniente de la lista de matrices y consiste en agregar subelementos a dicho árbol con los datos y demás atributos del nodo para

finalmente finalizar la generación del archivo xml con los datos reducidos.

b.8) listaMatrices:

Al igual que la clase anterior esta clase contiene cada uno de los métodos para la creación de un nuevo elemento para una lista, así como los métodos para el procesamiento y reducciones de matrices.

- **Add ():** este método se encarga de agregar un nuevo nodo con cada uno de sus atributos.
- **Length ():** crea un contador y aumenta su valor en 1 cada vez que se encuentra un nodo en la lista, finalmente retorna el valor del contador.
- **Imprimir ():** por cada nodo en la lista hace un llamado al método imprimir() propio de cada lista de datos que el nodo tiene como atributo.
- **Reducir ():** este es el método mas complejo del programa, consistió en generar una nueva lista de matrices. Posteriormente se procede a iterar con varios contadores tanto filas y columnas, para comparar entre filas fue necesario el uso de dos contadores, uno para la fila actual y otro contador para la fila con la que se desea comparar. Por cada iteración en la fila actual se procedía a comparar cada valor en la siguiente fila, si se encontraban coincidencias entonces se sumaba a un contador de coincidencias, si al terminar de iterar la fila, el numero de coincidencias era equivalente al numero de columnas de la matriz esto indicaba que las dos filas tenían identidades idénticas, por lo tanto se procedía a iterar desde el inicio sumando las filas. Finalmente se suma en 1 a un contador de frecuencias y se procede a agregar este a la nueva lista de matrices en conjunto con su nueva fila ya sumada, este proceso se repite hasta que ya no existan filas en la matriz, finalmente se retorna la lista con las matrices ya reducidas.

Conclusiones

-Las Listas enlazadas son útiles para minimizar el uso de memoria a la hora de guardar grandes cantidades de datos, los cuales no se conoce exactamente su cantidad.

-La creación de listas enlazadas en Python es innecesaria ya que las listas en este lenguaje ya son dinámicas y sencillas de utilizar, por lo tanto, esto ahorra muchas líneas de código al elaborar una.

-La programación orientada a objetos resulta ser de gran ayuda en casi cualquier escenario, en las listas enlazadas toma un gran protagonismo a la hora de generar nodos.

Referencias bibliográficas

- Luis Joyanes Aguilar, (1998). Estructuras de Datos.
  McGraw-Hill Interamericana. Colección:
  1ª Edición / 888 págs.

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