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.

Distribuidor de Datos

201901055 – Angel Geovany Aragón Pérez

Resumen

Es un proyecto que consiste en alojar objetos de base de datos en sitios distribuidos, de manera que el costo total de la trasmisión de datos para el procesamiento de la aplicación sea reducido y para esto se busca resolver así el problema combinatorio NP-Hard que principalmente es por la fuerte demanda de recursos.

El set de objetos de una base de datos es una red de computadoras que consiste en un set de sitios donde un set de consultas son ejecutadas en donde los objetos de base de datos requeridos por cada consulta es 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.

Para resolver este problema se realiza un nuevo esquema replicando de alojamiento que se adapte a un nuevo patrón de uso de la base de datos y minimice los costos de transmisión.

Palabras clave
Trasmisión, Alojamiento, Distribución, procesamiento, computadoras

Abstract

It is a project that consists of hosting database objects in distributed sites, so that the total cost of data transmission for the processing of the application is reduced and for this it is sought to solve the NP-Hard combinatorial problem that mainly It is due to the strong demand for resources.

The set of objects of a database is a computer network that consists of a set of sites where a set of queries are executed where the database objects required for each query is an initial scheme for hosting database objects of data, and the frequencies of access of each query from each site in a period of time.

To solve this problem, a new replicating scheme of accommodation is made that adapts to a new pattern of use of the database and minimizes transmission costs.

Keywords
Transmission, Hosting, Distribution, Processing, Computers

Introducción

En la actualidad se sufre muchos de los problemas combinatorios NP-Hard por la fuerte demanda de recursos, para reducir estos problemas se realiza una aplicación que hace reducir los costos totales de la trasmisión de datos por medio de obtener la matriz de frecuencia de acceso en los sitios y transformarla en patrones de acceso y agrupar las tuplas con el mismo patrón utilizando tipos de datos abstractos para realizar las operaciones y así reducir los costos de trasmisión de datos en las base de datos

Desarrollo del tema

Para el desarrollo del proyecto de distribución de datos se utilizó estructuras de datos de tipo abstracto (TDA) para reorganizar los datos de manera que se reduzcan las operaciones.

Para comprender los datos de tipo abstracto primero debes conocer la abstracción, la abstracción es una herramienta que más nos ayuda a la hora de solucionar un problema, es un mecanismo fundamental para la comprensión de problemas que poseen una gran cantidad de detalles.

El proceso de abstracción presenta dos aspectos complementarios:

- Destacar los aspectos relevantes del objeto.
- Ignorar los aspectos irrelevantes del mismo

De modo general podemos decir que la abstracción permite establecer un nivel jerárquico en el estudio del problema

Por ejemplo: los lenguajes de programación de alto nivel permiten al programador abstraerse del sin fin de detalles de los lenguajes ensambladores. Otro ejemplo, la memoria de la computadora es una estructura unidimensional formada por celdas y sin embargo trabajamos como si fuera única.

La abstracción nos brinda la posibilidad de ir definiendo una serie de refinamientos sucesivos a nuestro TDA y entiéndase bien que cuando decimos refinamientos sucesivos nos estamos refiriendo a la estrategia que se utiliza para descomponer un problema en subproblemas.

Un Tipo de Dato Abstracto (TDA) es un modelo que define valores y las operaciones que se pueden realizan sobre ellos. Y se denomina abstracto ya que la intención es que quien lo utiliza, no necesita conocer los detalles de la representación interna o bien el cómo están implementadas las operaciones.

Hay una gran variación de estructuras abstractas, algunos ejemplos de su uso en la programación son:

- **Conjuntos**: implementación de conjuntos con sus operaciones), operaciones de inserción, borrado, búsqueda.
- **Árboles** Binarios de Búsqueda: Implementación de árboles de elementos, utilizados para la representación interna de datos complejos. Aunque siempre se los toma como un TDA separado son parte de la familia de los grafos.
- **Pilas y Colas:** Implementación de los algoritmos FIFO y LIFO.
- **Grafos**: Implementación de grafos; una serie de vértices unidos mediante una serie de arcos o aristas.

Ahora que ya conocemos que son los datos de tipo abstracto hablaremos del proceso realizado para disminuir las operaciones que se realza en la base de datos y sobre la utilización de la aplicación.

Al iniciar la aplicación saldrá un menú con las siguientes funciones que puede realizar:

- Cargar Archivo: Lee y guarda un archivo XML por medio la ruta absoluta del archivo.

- Procesar Archivo: Realiza la disminución de las tuplas creando una nueva matriz reducida.

- Escribir Archivo: Escribe un nuevo archivo XML con las nuevas matrices reducidas.

- Mostrar datos del estudiante: Muestra los datos del estudiante encargado de la elaboración del proyecto

- Generar gráfica: Genera una gráfica de la matriz que desea.

- Salida: Sale de la aplicación.

Para hacer el menú se utilizó una clase llamada Menu donde contiene dos métodos estáticos:

- mostrar(): Este proceso se encarga de mostrar en pantalla las funciones que puede realizar la aplicación.
- menu(): Este proceso se encarga de re direccionar según la opción seleccionada por el usuario.

Para la opción de cargar el archivo, Procesar archivo y escribir archivo se utilizó una clase llamada Procesos que contiene todas las funciones necesarias para realizar las operaciones para crear la matriz reducida.

- hacer_matriz(): Este proceso recibe dos parámetros que representan el tamaño que se desea crear la matriz y crea dos lista ortogonal con el tamaño seleccionado, donde una será la matriz con los valores y otra que será con 0 y 1 para realizar las operaciones.

- procesar(): Este proceso llama la función de recorrer de la lista circular donde se almacena las matrices con sus valores.

- cargar_Archivo(): Este proceso se encarga de leer un archivo XML utilizando la librería minidom y de la ruta absoluta del archivo que el usuario ingrese.

- Leer_Archivo(): Este proceso se encarga de leer y guarda los datos en sus respectiva posición y lista.

- escribir(): Este proceso se encarga de escribir un nuevo archivo XML con las nuevas matrices reducidas utilizando la librería xml.etree.ElementTree.

- graficar(): Este proceso se encarga de escribir un archivo .dot mostrando una matriz seleccionada por el usuario por su ubicación.

Para la realización de las operaciones de la clase Procesos se utiliza dos listas ortogonales dentro de una lista circular y esa lista circular dentro de otra lista circular.

Para la creación de las listas necesarias se utiliza las siguientes clases:

- Lista_matriz: Esta clase contiene todas las funciones de la lista que contiene la matriz y está contenida en una lista circular.

- Nodo_lista_matriz: Esta clase contiene el valor de esa posición de la lista y el nodo siguiente

- Matriz: Esta clase contiene todas las funciones de las operaciones que puede realizar la lista ortogonal.

- Nodo_Matriz: Esta clase contiene el valor en esa posición de la lista y los nodos de arriba, abajo, siguiente, anterior.

- Nodo: Esta clase contiene el valor en esa posición de la lista y el nodo siguiente:

- Lista_Circular: Esta clase contiene todas las funciones de lista circular que almacena a las demás listas.

Estas clases por ser lista y como se busca casi el mismo objetivo por ese motivo comparten algunas funciones, las funciones que contienen estas clases son las siguientes:

- esta_vacia(): Esta funcion indica que la lista esta vacia.

- tamaño(): Esta funcion retorna el tamaño de la lista

- agregar_ultimo(): Este proceso crea un nuevo nodo y le da el valor que recibió

- recorrer(): Recorre la lista

- modificar(): Cambia el valor de un nodo seleccionado

- crear(): Recibe dos parámetros que son n y m. Crear una n nodos hacia la derecha y m nodos hacia abajo en la lista ortogonal.

- eliminar(): Recibe un dato, busca dentro de la lista un dato que sea igual al dato recibido y si lo hay lo elimina.
- buscar(): Busca un dato por su posición o por su valor y lo retorna.

- son_iguales(): Recorre la lista y retorna las posiciones en donde el valor sea igual.

Conclusiones

Como resultado de este proyecto se pudo crear un nuevo archivo xml mas pequeño reduciendo el costo de la trasmisión de datos.
Con el uso de tipo de datos abstracto se obtuvo un manejo de datos ordenados sin pérdidas de datos a la hora de agrupar las tuplas.

Referencias bibliográficas

M. Careño, A. Sandoval, I. Estrada (2012) México, UABCS