**UNIVERSIDAD NACIONAL DE SAN AGUSTIN**

FACULTAD DE PRODUCCION Y SERVICIOS

ESCUELA PROFESIONAL DE CIENCIA DE LA

COMPUTACIÓN

Gestor de Datos usando

Google cluster-usage traces

Autores:

Nicolas Jimenez Artica

Nicoll del Rosario Aza Mamani

Williams Nifla Catasi

Denis Ccallo Valeriano

Arequipa – Perú

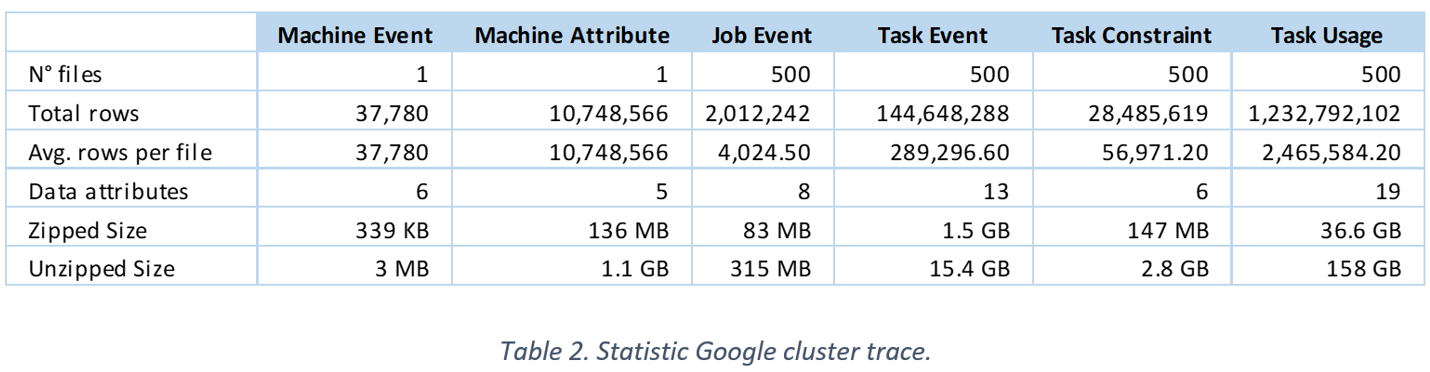
2018

1. INTRODUCCIÓN

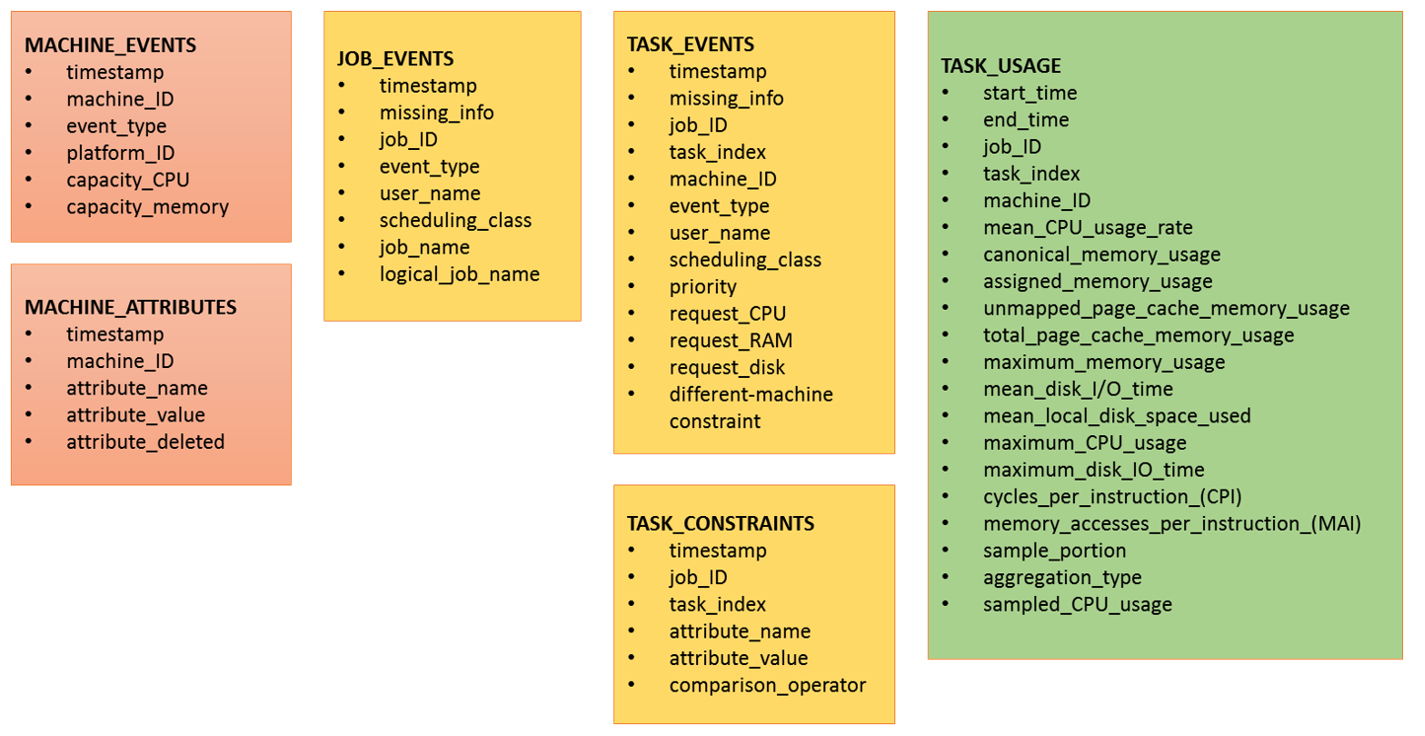
Este documento presenta la documentación del código de la implementación de Gestor de Datos para los archivos de Google cluster-usage traces además de un esquema para su mejor comprensión de cómo funciona. Implementaremos funciones de Indexacion, Transacciones, un simulador de Control de concurrencia para varios usuarios.

**Google cluster-usage traces**

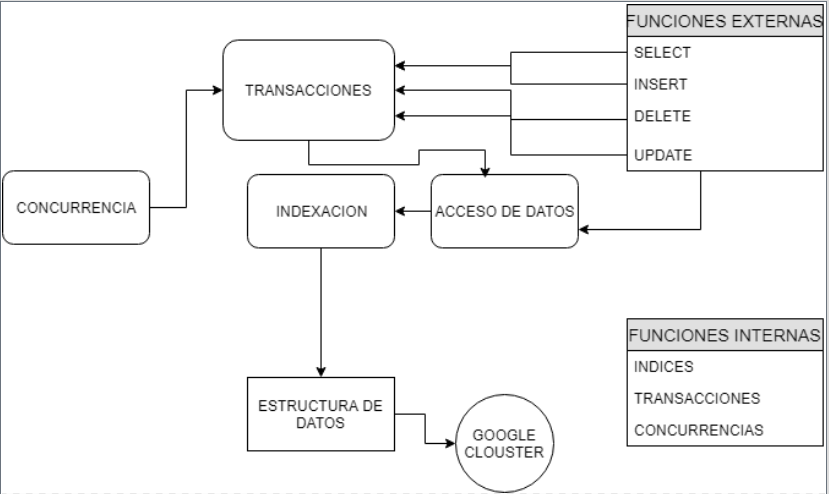
Contiene 6 tablas con pesos variantes entre 3 mb hasta uno con 158 gb



Cada **tabla** está dentro de una **carpeta,** tiene su propio Id como identificador las cuales usaremos para las funciones de Insert, Delete ,Update y Select



1. ARQUITECTURA



¿Como funciona?

**Main.cpp**

int main()

Un bucle con operaciones de Indexar, Seleccionar, Actualizar, Eliminar y simular una Concurrencia

Void indices(string carpeta)  
Seleccionara una carpeta donde está alojada una tabla y lo indexara, en el momento de indexar bloqueara el archivo y cuando termine lo desbloqueara, la indexación lo hará con un árbol B

string select(string ID,string carpeta)

Selecciona un ID y una carpeta para buscar lo solicitado, cada tabla tendrá su función para seleccionar

string update(string ID,string carpeta)

Selecciona un ID y una carpeta para actualizar lo solicitado, cada tabla tendrá su función para actualizar

string Delete(string ID,string carpeta)

Selecciona un ID y una carpeta para eliminar el registro solicitado, cada tabla tendrá su función para eliminar

string Insertar(string ID,string carpeta)

Selecciona un ID y una carpeta para Insertar al final, cada tabla tendrá su función para Insertar

void Concurrecia()

Función que pone las operaciones de select, insert, delete, update en una cola de prioridad añadiéndole un valor de prioridad, después ordenara todas las operaciones que ingresamos para después ser ejecutadas según la cola de prioridad, simulando así a varios usuarios realizando operaciones.

**Indices.h**

void task\_usage::indicesTU()

Lee cada archivo dentro de la carpeta, crea archivos hashkey.txt, indi.txt, Clusterkey.txt, añadirá los id a estos txt, ordenara los IDs para finalmente insertara en forma ordenada

void task\_events::indicesTE()

Lee cada archivo dentro de la carpeta, crea archivos indi.txt, Clusterkey.txt, añadirá los id a estos txt, ordenara los IDs para finalmente insertara en forma ordenada

void task\_constraints::indicesTC()

Lee cada archivo dentro de la carpeta, Ordena los IDs en un arreglo, crea archivos hashkey.txt, indi.txt, añadirá los id a estos txt

void machine\_events::indicesME()

Lee cada archivo dentro de la carpeta, Ordena los IDs en un arreglo, crea archivos hashkey.txt, inid.txt, añadirá los id a estos txt

void machine\_attributes::indicesMA()

Lee cada archivo dentro de la carpeta, Ordena los IDs en un arreglo, crea archivos hashkey.txt, inid.txt, añadirá los id a estos txt

void job\_events::indicesJE()

Lee cada archivo dentro de la carpeta, Ordena los IDs en un arreglo, crea archivos hashkey.txt, indi.txt, añadirá los id a estos txt

**Earbol.h**

BTreeNode \*BTreeNode::search(long long int k)

Busca en el árbol B

void BTree::insert(long long int k)

Inserta en el arbol

void BTreeNode::insertNonFull(long long int k)

Inserta en el árbol

void BTreeNode::splitChild(int i, BTreeNode \*y)

Dividira el árbol

**Hash.h**

Guarda IDd no repetidos junto con sus respectivas posiciones junto a los índices globales siendo su función de dispersión no tomar sus repetidos

**Tablas.h**

Aquí están las clases de las tablas con sus respectivos atributos y funciones

**Bloqueos.h**

Funciones donde un boleano pasara de true=bloqueado a false=desbloqueado de todas las tablas

**Estructuras.h**

Cola de prioridad de Clase funciones con funciones boleanas de “ > ” y ” < ”

Clase Datos con ID, posición, Parchivo y funciones boleanas de “>” y ”<”

**Fbasicas.h**

string numero(int n)

retornara “num” según la cantidad de “n” que ingreses a entra función, lo usaremos para saber en cuantas partes se dividió el archivo de una tabla dentro de su carpeta

bool esNumero(char\* num)

comprobara si un char es numero o no

**Select.h**

char \*job\_events::selectJE(string ID)

Abre el hashkey.txt de Job\_events y busca el ID, si lo encuentra entonces abrirá el índice para buscar donde esta ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta de Job\_events , si lo encuentra retornara “selección hecha correctamente” caso contrario “esta línea de registro fue eliminada”.

char \*machine\_attributes::selectMA(string ID)

Abre el hashkey.txt de machine\_attributes y busca el ID, si lo encuentra entonces abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta de machine\_attributes , si lo encuentra retornara “selección hecha correctamente” caso contrario “esta línea de registro fue eliminada”.

string machine\_events::selectME(string ID)

Abre el hashkey.txt de machine\_events y busca el ID, si lo encuentra entonces abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta de machine\_events , si lo encuentra retornara “selección hecha correctamente” caso contrario “esta línea de registro fue eliminada”.

char \*task\_constraints::selectTC(string ID)

Abre el hashkey.txt de task\_constraints y busca el ID, si lo encuentra entonces abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta de task\_constraints , si lo encuentra retornara “selección hecha correctamente” caso contrario “esta línea de registro fue eliminada”.

char \*task\_events::selectTE(string ID)

Abre el Clusterkey de task\_events y busca el ID, si lo encuentra entonces abrirá el índice para buscar donde esta ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta task\_events, si lo encuentra retornara “selección hecha correctamente” caso contrario “esta línea de registro fue eliminada”.

char\* task\_usage::selectTU(string ID)

Abre el Clusterkey .txt de task\_usage y busca el ID, si lo encuentra entonces abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta de task\_usage, si lo encuentra retornara “selección hecha correctamente” caso contrario “esta línea de registro fue eliminada”.

**Delete.h**

string machine\_events::Delete(string ID)

Abre el Haskey.txt de machine\_events y busca el ID, si lo encuentra abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta machine-events, si lo encuentra eliminara la línea y devolverá “línea eliminada” caso contrario “no se pudo realizar la operación”

string machine\_attributes::Delete(string ID)

Abre el Haskey.txt de machine\_events y busca el ID, si lo encuentra abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta machine-events, si lo encuentra eliminara la línea y devolverá “línea eliminada” caso contrario “no se pudo realizar la operación”

string job\_events::Delete(string ID)

Abre el Haskey.txt de job\_events y busca el ID, si lo encuentra abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta job\_events, si lo encuentra eliminara la línea y devolverá “línea eliminada” caso contrario “no se pudo realizar la operación”

string task\_constraints::Delete(string ID)

Abre el Haskey.txt de machine\_events y busca el ID, si lo encuentra abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta machine-events, si lo encuentra eliminara la línea y devolverá “línea eliminada” caso contrario “no se pudo realizar la operación”

string task\_events::Delete(string ID)

Abre el Clusterkey.txt de task\_events y busca el ID, si lo encuentra abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta task-events, si lo encuentra eliminara la línea y devolverá “línea eliminada” caso contrario “no se pudo realizar la operación”

string task\_usage::Delete(string ID)

Abre el Clusterkey.txt de task\_usage y busca el ID, si lo encuentra abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta task\_usage, si lo encuentra eliminara la línea y devolverá “línea eliminada” caso contrario “no se pudo realizar la operación”

**Update.h**

string job\_events::update(string ID,string NewLine)

Abre el Hashkey.txt de job\_events para buscar el ID, si lo encuentra abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta job\_events, si lo encuentra entonces reemplazara en esa línea llamando a la función Newline para actualizar los datos.

string machine\_attributes::update(string ID,string NewLine)

Abre el Hashkey.txt de machine\_attributes para buscar el ID, si lo encuentra abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta machine\_attributes, si lo encuentra entonces reemplazara en esa línea llamando a la función Newline para actualizar los datos.

string machine\_events::update(string ID,string NewLine)

Abre el Hashkey.txt de machine\_events para buscar el ID, si lo encuentra abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta machine\_events, si lo encuentra entonces reemplazara en esa línea llamando a la función Newline para actualizar los datos.

string task\_constraints::update(string ID,string NewLine)

Abre el Hashkey.txt de task\_constraints para buscar el ID, si lo encuentra abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta task\_constraints, si lo encuentra entonces reemplazara en esa línea llamando a la función Newline para actualizar los datos.

string task\_events::update(string ID,string NewLine)

Abre el Hashkey.txt de job\_events para buscar el ID, si lo encuentra abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta job\_events, si lo encuentra entonces reemplazara en esa línea llamando a la función Newline para actualizar los datos.

string task\_usage::update(string ID,string NewLine)

Abre el Hashkey.txt de job\_events para buscar el ID, si lo encuentra abrirá el índice para buscar donde está ubicado el registro del ID para posteriormente buscarlo en el archivo de la carpeta job\_events, si lo encuentra entonces reemplazara en esa línea llamando a la función Newline para actualizar los datos.

**Newline.h**

string machine\_attributes::NewLine(string ID)

Funcion que recibe un ID y agrega nuevos registros de los atributos de machine\_atritubes, retorna un string con esos atributos

string machine\_events::NewLine(string ID)

Funcion que recibe un ID y agrega nuevos registros de los atributos de machine\_events, retorna un string con esos atributos

string job\_events::NewLine(string ID)

Funcion que recibe un ID y agrega nuevos registros de los atributos de jo\_events, retorna un string con esos atributos

string task\_constraints::NewLine(string ID)

Funcion que recibe un ID y agrega nuevos registros de los atributos de task\_constraints, retorna un string con esos atributos

string task\_events::NewLine(string ID)

Funcion que recibe un ID y agrega nuevos registros de los atributos de task\_events, retorna un string con esos atributos

string task\_usage::NewLine(string ID)

Funcion que recibe un ID y agrega nuevos registros de los atributos de task\_usage, retorna un string con esos atributos

**Insert.h**

string machine\_attributes::insertMA(string ID,string newline)

Abrimos el archivo donde está la tabla machine\_attributes donde insertaremos al final del archivo los datos que se obtendrán llamando a la función Newline, luego abrimos el archivo de índices, sacamos la posición del ID del archivo de la tabla, luego insertamos 3 campos: ID, posición, Nro del archivo donde se está insertando, luego sacamos la posición del ID del archivo de indices para el archivo Hashkey con la diferencia de que insertamos 2 campos los cuales son ID, posición.

string machine\_events::insertME(string ID,string newline)

Abrimos el archivo donde está la tabla machine\_events donde insertaremos al final del archivo los datos que se obtendrán llamando a la función Newline, luego abrimos el archivo de índices, sacamos la posición del ID del archivo de la tabla, luego insertamos 3 campos: ID, posición, Nro del archivo donde se está insertando, luego sacamos la posición del ID del archivo de indices para el archivo Hashkey con la diferencia de que insertamos 2 campos los cuales son ID, posición.

string job\_events::insertJE(string ID,string newline)

Abrimos el archivo donde está la tabla job-events donde insertaremos al final del archivo los datos que se obtendrán llamando a la función Newline, luego abrimos el archivo de índices, sacamos la posición del ID del archivo de la tabla, luego insertamos 3 campos: ID, posición, Nro del archivo donde se está insertando, luego sacamos la posición del ID del archivo de indices para el archivo Hashkey con la diferencia de que insertamos 2 campos los cuales son ID, posición.

string task\_constraints::insertTC(string ID,string newline)

Abrimos el archivo donde está la tabla task\_constraints donde insertaremos al final del archivo los datos que se obtendrán llamando a la función Newline, luego abrimos el archivo de índices, sacamos la posición del ID del archivo de la tabla, luego insertamos 3 campos: ID, posición, Nro del archivo donde se está insertando, luego sacamos la posición del ID del archivo de indices para el archivo Hashkey con la diferencia de que insertamos 2 campos los cuales son ID, posición.

string task\_events::insertTE(string ID,string newline)

Abrimos el archivo donde está la tabla task\_events donde insertaremos al final del archivo los datos que se obtendrán llamando a la función Newline, luego abrimos el archivo de índices, sacamos la posición del ID del archivo de la tabla, luego insertamos 3 campos: ID, posición, Nro del archivo donde se está insertando, luego sacamos la posición del ID del archivo de indices para el archivo Hashkey con la diferencia de que insertamos 2 campos los cuales son ID, posición.

string task\_usage::insertTU(string ID,string newline)

Abrimos el archivo donde está la tabla task\_usage donde insertaremos al final del archivo los datos que se obtendrán llamando a la función Newline, luego abrimos el archivo de índices, sacamos la posición del ID del archivo de la tabla, luego insertamos 3 campos: ID, posición, Nro del archivo donde se está insertando, luego sacamos la posición del ID del archivo de indices para el archivo Hashkey con la diferencia de que insertamos 2 campos los cuales son ID, posición.

1. CODIGO

Main.cpp

|  |
| --- |
|  |
|  | #include "insert.h"  #include "EArbol.h"  machine\_attributes MA;  machine\_events ME;  job\_events JE;  task\_constraints TC;  task\_events TE;  task\_usage TU;  void indices(string carpeta)  { |
|  | string respuesta; |
|  | if(carpeta=="job\_events") |
|  | { |
|  | while(JE.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | JE.bloquear(); |
|  | JE.indicesJE(); |
|  | JE.desbloquear(); |
|  | } |
|  | else if(carpeta=="task\_usage") |
|  | { |
|  | while(TU.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | TU.bloquear(); |
|  | TU.indicesTU(); |
|  | TU.desbloquear(); |
|  | } |
|  | else if(carpeta=="machine\_attributes") |
|  | { |
|  | while(MA.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | MA.bloquear(); |
|  | MA.indicesMA(); |
|  | MA.desbloquear(); |
|  | } |
|  | else if(carpeta=="machine\_events") |
|  | { |
|  | while(ME.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | ME.bloquear(); |
|  | ME.indicesME(); |
|  | ME.desbloquear(); |
|  | } |
|  | else if(carpeta=="task\_constraints") |
|  | { |
|  | while(TC.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | TC.bloquear(); |
|  | TC.indicesTC(); |
|  | TC.desbloquear(); |
|  | } |
|  | else if(carpeta=="task\_events") |
|  | { |
|  | while(TE.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | TE.bloquear(); |
|  | TE.indicesTE(); |
|  | TE.desbloquear(); |
|  | } |
|  | else |
|  | cout<<"No se tiene la tabla a buscar."<<endl; |
|  | } |
|  | string select(string ID,string carpeta) |
|  | { |
|  | string respuesta; |
|  | if(carpeta=="job\_events") |
|  | { |
|  | while(JE.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | respuesta=JE.selectJE(ID); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="machine\_attributes") |
|  | { |
|  | while(MA.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | respuesta=MA.selectMA(ID); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="machine\_events") |
|  | { |
|  | while(ME.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | respuesta=ME.selectME(ID); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="task\_constraints") |
|  | { |
|  | while(TC.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | respuesta=TC.selectTC(ID); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="task\_events") |
|  | { |
|  | while(TE.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | respuesta=TE.selectTE(ID); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="task\_usage") |
|  | { |
|  | while(TE.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | respuesta=TU.selectTU(ID); |
|  | return respuesta; |
|  | } |
|  | else |
|  | return "No se tiene la tabla a buscar."; |
|  | } |
|  | string update(string ID,string carpeta) |
|  | { |
|  | string respuesta; |
|  | string newline; |
|  | if(carpeta=="job\_events") |
|  | { |
|  | while(JE.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | JE.bloquear(); |
|  | newline=JE.NewLine(ID); |
|  | respuesta=JE.update(ID,newline); |
|  | JE.desbloquear(); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="machine\_attributes") |
|  | { |
|  | while(MA.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | MA.bloquear(); |
|  | newline=MA.NewLine(ID); |
|  | respuesta=MA.update(ID,newline); |
|  | MA.desbloquear(); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="machine\_events") |
|  | { |
|  | while(ME.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | ME.bloquear(); |
|  | newline=ME.NewLine(ID); |
|  | respuesta=ME.update(ID,newline); |
|  | ME.desbloquear(); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="task\_constraints") |
|  | { |
|  | while(TC.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | TC.bloquear(); |
|  | newline=TC.NewLine(ID); |
|  | respuesta=TC.update(ID,newline); |
|  | TC.desbloquear(); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="task\_events") |
|  | { |
|  | while(TE.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | TE.bloquear(); |
|  | newline=TE.NewLine(ID); |
|  | respuesta=TE.update(ID,newline); |
|  | TE.desbloquear(); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="task\_usage") |
|  | { |
|  | while(TU.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | TU.bloquear(); |
|  | newline=TU.NewLine(ID); |
|  | respuesta=TU.update(ID,newline); |
|  | TU.desbloquear(); |
|  | return respuesta; |
|  | } |
|  | else |
|  | return "No se tiene la tabla a buscar."; |
|  | } |
|  | string Delete(string ID,string carpeta) |
|  | { |
|  | string respuesta; |
|  | if(carpeta=="job\_events") |
|  | { |
|  | while(JE.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | JE.bloquear(); |
|  | respuesta=JE.Delete(ID); |
|  | JE.desbloquear(); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="machine\_attributes") |
|  | { |
|  | while(MA.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | MA.bloquear(); |
|  | respuesta=MA.Delete(ID); |
|  | MA.desbloquear(); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="machine\_events") |
|  | { |
|  | while(ME.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | ME.bloquear(); |
|  | respuesta=ME.Delete(ID); |
|  | ME.desbloquear(); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="task\_constraints") |
|  | { |
|  | while(TC.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | TC.bloquear(); |
|  | respuesta=TC.Delete(ID); |
|  | TC.desbloquear(); |
|  | return respuesta; |
|  | } |
|  | else if(carpeta=="task\_events") |
|  | { |
|  | while(TE.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | TE.bloquear(); |
|  | respuesta=TE.Delete(ID); |
|  | TE.desbloquear(); |
|  | return respuesta; |
|  | } |
|  | else |
|  | return "No se tiene la tabla a buscar."; |
|  | } |
|  | string Insertar(string ID,string carpeta) |
|  | { |
|  | string newline; |
|  | string resultado; |
|  | if(carpeta=="job\_events") |
|  | { |
|  | while(JE.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | JE.bloquear(); |
|  | newline=JE.NewLine(ID); |
|  | resultado=JE.insertJE(ID,newline); |
|  | JE.desbloquear(); |
|  | return resultado; |
|  | } |
|  | else if(carpeta=="task\_usage") |
|  | { |
|  | while(TU.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | TU.bloquear(); |
|  | newline=TU.NewLine(ID); |
|  | resultado=TU.insertTU(ID,newline); |
|  | TU.desbloquear(); |
|  | return resultado; |
|  | } |
|  | else if(carpeta=="machine\_attributes") |
|  | { |
|  | while(MA.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | MA.bloquear(); |
|  | newline=MA.NewLine(ID); |
|  | resultado=MA.insertMA(ID,newline); |
|  | MA.desbloquear(); |
|  | return resultado; |
|  | } |
|  | else if(carpeta=="machine\_events") |
|  | { |
|  | while(ME.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | ME.bloquear(); |
|  | newline=ME.NewLine(ID); |
|  | resultado=ME.insertME(ID,newline); |
|  | ME.desbloquear(); |
|  | return resultado; |
|  | } |
|  | else if(carpeta=="task\_constraints") |
|  | { |
|  | while(TC.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | TC.bloquear(); |
|  | newline=TC.NewLine(ID); |
|  | resultado=TC.insertTC(ID,newline); |
|  | TC.desbloquear(); |
|  | return resultado; |
|  | } |
|  | else if(carpeta=="task\_events") |
|  | { |
|  | while(TE.Estado()) |
|  | { |
|  | Sleep(5000); |
|  | } |
|  | TE.bloquear(); |
|  | newline=TE.NewLine(ID); |
|  | resultado=TE.insertTE(ID,newline); |
|  | TE.desbloquear(); |
|  | return resultado; |
|  | } |
|  | else |
|  | cout<<"No se tiene la tabla a buscar."<<endl; |
|  | } |
|  | void Concurrecia() |
|  | { |
|  | string opc,IDX,carpetaX,Newline; |
|  |  |
|  | vector<funciones> Funciones; |
|  | while(true) |
|  | { |
|  | cout<<endl; |
|  | cout<<"Que operacion desea realizar"<<endl; |
|  | cout<<"1.- Seleccionar por indice de una tabla:"<<endl; |
|  | cout<<"2.- Insertar registro en la BD:"<<endl; |
|  | cout<<"3.- Actualizar registro de la BD:"<<endl; |
|  | cout<<"4.- Eliminar registro de la BD:"<<endl; |
|  | cout<<"5.- Terminar y Ejecutar"<<endl; |
|  | cin>>opc; |
|  | if(opc=="1") |
|  | { |
|  | funciones AUX; |
|  | cout<<"Ingrese indice seguido de la tabla que desea seleccionar"<<endl; |
|  | cin>>IDX>>carpetaX; |
|  | AUX.carpeta=carpetaX; |
|  | AUX.funcion=1; |
|  | AUX.ID=IDX; |
|  | Funciones.push\_back(AUX); |
|  | } |
|  | else if(opc=="2") |
|  | { |
|  | funciones AUX; |
|  | cout<<"Ingrese indice seguido de la tabla que desea insertar"<<endl; |
|  | cin>>IDX>>carpetaX; |
|  | if(carpetaX=="job\_events") |
|  | { |
|  | Newline=JE.NewLine(IDX); |
|  | } |
|  | else if(carpetaX=="task\_usage") |
|  | { |
|  | Newline=TU.NewLine(IDX); |
|  | } |
|  | else if(carpetaX=="machine\_attributes") |
|  | { |
|  | Newline=MA.NewLine(IDX); |
|  | } |
|  | else if(carpetaX=="machine\_events") |
|  | { |
|  | Newline=ME.NewLine(IDX); |
|  | } |
|  | else if(carpetaX=="task\_constraints") |
|  | { |
|  | Newline=TC.NewLine(IDX); |
|  | } |
|  | else if(carpetaX=="task\_events") |
|  | { |
|  | Newline=TE.NewLine(IDX); |
|  | } |
|  | else return; |
|  | if(Newline!="Error") |
|  | { |
|  | AUX.carpeta=carpetaX; |
|  | AUX.funcion=2; |
|  | AUX.ID=IDX; |
|  | AUX.newline=Newline; |
|  | Funciones.push\_back(AUX); |
|  | } |
|  | else return; |
|  | } |
|  | else if(opc=="3") |
|  | { |
|  | funciones AUX; |
|  | cout<<"Ingrese indice seguido de la tabla que desea actualizar"<<endl; |
|  | cin>>IDX>>carpetaX; |
|  | if(carpetaX=="job\_events") |
|  | { |
|  | Newline=JE.NewLine(IDX); |
|  | } |
|  | else if(carpetaX=="task\_usage") |
|  | { |
|  | Newline=TU.NewLine(IDX); |
|  | } |
|  | else if(carpetaX=="machine\_attributes") |
|  | { |
|  | Newline=MA.NewLine(IDX); |
|  | } |
|  | else if(carpetaX=="machine\_events") |
|  | { |
|  | Newline=ME.NewLine(IDX); |
|  | } |
|  | else if(carpetaX=="task\_constraints") |
|  | { |
|  | Newline=TC.NewLine(IDX); |
|  | } |
|  | else if(carpetaX=="task\_events") |
|  | { |
|  | Newline=TE.NewLine(IDX); |
|  | } |
|  | else return; |
|  | if(Newline!="Error") |
|  | { |
|  | AUX.carpeta=carpetaX; |
|  | AUX.funcion=3; |
|  | AUX.ID=IDX; |
|  | AUX.newline=Newline; |
|  | Funciones.push\_back(AUX); |
|  | } |
|  | else return; |
|  | } |
|  | else if(opc=="4") |
|  | { |
|  | funciones AUX; |
|  | cout<<"Ingrese indice seguido de la tabla que desea eliminar"<<endl; |
|  | cin>>IDX>>carpetaX; |
|  | AUX.carpeta=carpetaX; |
|  | AUX.funcion=4; |
|  | AUX.ID=IDX; |
|  | Funciones.push\_back(AUX); |
|  | } |
|  | else if(opc=="5") |
|  | { |
|  | break; |
|  | } |
|  | else |
|  | { |
|  | cout<<"Opcion incorrecta :("<<endl; |
|  | } |
|  | } |
|  | sort(Funciones.begin(),Funciones.end(),[](const funciones& fun1,const funciones& fun2){return (fun1.funcion<fun2.funcion);}); |
|  | for(int i=0;i<Funciones.size();i++) |
|  | { |
|  | if(Funciones[i].newline=="") |
|  | { |
|  | if(Funciones[i].funcion==1) |
|  | { |
|  | if(Funciones[i].carpeta=="job\_events") |
|  | { |
|  | cout<<JE.selectJE(Funciones[i].ID)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="task\_usage") |
|  | { |
|  | cout<<TU.selectTU(Funciones[i].ID)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="machine\_attributes") |
|  | { |
|  | cout<<MA.selectMA(Funciones[i].ID)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="machine\_events") |
|  | { |
|  | cout<<ME.selectME(Funciones[i].ID)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="task\_constraints") |
|  | { |
|  | cout<<TC.selectTC(Funciones[i].ID)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="task\_events") |
|  | { |
|  | cout<<TE.selectTE(Funciones[i].ID)<<endl<<endl; |
|  | } |
|  | else return; |
|  | } |
|  | else if(Funciones[i].funcion==4) |
|  | { |
|  | if(Funciones[i].carpeta=="job\_events") |
|  | { |
|  | cout<<JE.Delete(Funciones[i].ID)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="task\_usage") |
|  | { |
|  | cout<<TU.Delete(Funciones[i].ID)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="machine\_attributes") |
|  | { |
|  | cout<<MA.Delete(Funciones[i].ID)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="machine\_events") |
|  | { |
|  | cout<<ME.Delete(Funciones[i].ID)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="task\_constraints") |
|  | { |
|  | cout<<TC.Delete(Funciones[i].ID)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="task\_events") |
|  | { |
|  | cout<<TE.Delete(Funciones[i].ID)<<endl<<endl; |
|  | } |
|  | else return; |
|  | } |
|  | } |
|  | else |
|  | { |
|  | if(Funciones[i].funcion==2) |
|  | { |
|  | if(Funciones[i].carpeta=="job\_events") |
|  | { |
|  | cout<<JE.insertJE(Funciones[i].ID,Funciones[i].newline)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="task\_usage") |
|  | { |
|  | cout<<TU.insertTU(Funciones[i].ID,Funciones[i].newline)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="machine\_attributes") |
|  | { |
|  | cout<<MA.insertMA(Funciones[i].ID,Funciones[i].newline)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="machine\_events") |
|  | { |
|  | cout<<ME.insertME(Funciones[i].ID,Funciones[i].newline)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="task\_constraints") |
|  | { |
|  | cout<<TC.insertTC(Funciones[i].ID,Funciones[i].newline)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="task\_events") |
|  | { |
|  | cout<<TE.insertTE(Funciones[i].ID,Funciones[i].newline)<<endl<<endl; |
|  | } |
|  | else return; |
|  | } |
|  | else if(Funciones[i].funcion==3) |
|  | { |
|  | if(Funciones[i].carpeta=="job\_events") |
|  | { |
|  | cout<<JE.update(Funciones[i].ID,Funciones[i].newline)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="task\_usage") |
|  | { |
|  | cout<<TU.update(Funciones[i].ID,Funciones[i].newline)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="machine\_attributes") |
|  | { |
|  | cout<<MA.update(Funciones[i].ID,Funciones[i].newline)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="machine\_events") |
|  | { |
|  | cout<<ME.update(Funciones[i].ID,Funciones[i].newline)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="task\_constraints") |
|  | { |
|  | cout<<TC.update(Funciones[i].ID,Funciones[i].newline)<<endl<<endl; |
|  | } |
|  | else if(Funciones[i].carpeta=="task\_events") |
|  | { |
|  | cout<<TE.update(Funciones[i].ID,Funciones[i].newline)<<endl<<endl; |
|  | } |
|  | else return; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | int main() |
|  | { |
|  | string opc; |
|  | cout<<"---BIENVENIDOS A NUESTRO MOTOR DE BASE DE DATOS---"<<endl; |
|  | while(true) |
|  | { |
|  | cout<<"1.- Indexar una tabla:"<<endl; |
|  | cout<<"2.- Seleccionar por indice de una tabla:"<<endl; |
|  | cout<<"3.- Insertar registro en la BD:"<<endl; |
|  | cout<<"4.- Actualizar registro de la BD:"<<endl; |
|  | cout<<"5.- Eliminar registro de la BD:"<<endl; |
|  | cout<<"6.- Crear una concurrecia:"<<endl; |
|  | cout<<"7.- Exit:"<<endl; |
|  | cin>>opc; |
|  | if(opc=="1") |
|  | { |
|  | string archivo; |
|  | cout<<"Ingrese tabla a indexar"<<endl; |
|  | cin>>archivo; |
|  | indices(archivo); |
|  | } |
|  | else if(opc=="2") |
|  | { |
|  | string ID; |
|  | string archivo; |
|  | cout<<"Ingrese indice seguido de la tabla que desea seleccionar"<<endl; |
|  | cin>>ID>>archivo; |
|  | cout<<select(ID,archivo)<<endl; |
|  | } |
|  | else if(opc=="3") |
|  | { |
|  | string ID; |
|  | string archivo; |
|  | cout<<"Ingrese indice seguido de la tabla que desea seleccionar"<<endl; |
|  | cin>>ID>>archivo; |
|  | Insertar(ID,archivo); |
|  | } |
|  | else if(opc=="4") |
|  | { |
|  | string ID,archivo; |
|  | cout<<"Ingrese indice y despues la tabla que desea actualizar"<<endl; |
|  | cin>>ID>>archivo; |
|  | cout<<update(ID,archivo)<<endl; |
|  | } |
|  | else if(opc=="5") |
|  | { |
|  | string ID,archivo; |
|  | cout<<"Ingrese indice y despues la tabla que desea eliminar"<<endl; |
|  | cin>>ID>>archivo; |
|  | cout<<Delete(ID,archivo)<<endl; |
|  | } |
|  | else if(opc=="6") |
|  | { |
|  | Concurrecia(); |
|  | cout<<"Concurrecia Finalizada"<<endl; |
|  | } |
|  | else if(opc=="7") |
|  | { |
|  | break; |
|  | } |
|  | else |
|  | { |
|  | cout<<"Opcion incorrecta :("<<endl; |
|  | } |
|  | } |
|  | return 0; |
|  | } |

Indices.h

|  |
| --- |
|  |
|  | #include "NewLine.h"  void task\_usage::indicesTU() |
|  | { |
|  | char \*linea=new char[1000]; |
|  | long int posi=0,posicion,posiHash,temp=-1,cont=0; |
|  | int c=0; |
|  | vector<Datos> ID; |
|  | ID.shrink\_to\_fit(); |
|  | for(int i=0;i<this->cantidad;i++) |
|  | { |
|  | string num=numero(i); |
|  | string url=("google/"+this->archivo+"/part-"+num+"-of-"+numero(this->cantidad)+".csv"); |
|  | ifstream archivo (url,std::fstream::binary); |
|  | if (archivo == NULL) |
|  | { |
|  | cout<<"Error al abrir el archivo"<<endl; |
|  | return; |
|  | } |
|  | else |
|  | { |
|  | posicion=archivo.tellg(); |
|  | while(!archivo.eof() ) |
|  | { |
|  | if(posi!=3) |
|  | { |
|  | archivo.getline(linea,1000,','); |
|  | posi++; |
|  | } |
|  | else |
|  | { |
|  | long long int dato=atoll(linea); |
|  | Datos neew(dato,posicion,i); |
|  | ID.push\_back(neew); |
|  | if(ID.size()>33461309) |
|  | { |
|  | cont=0; |
|  | sort(ID.begin(),ID.end(),[](const Datos& datos1,const Datos& datos2){return (datos1.ID<datos2.ID);}); |
|  | ofstream HashKey("indices/"+this->archivo+"/HashKey-"+this->archivo+"-"+numero(c)+".txt"); |
|  | ofstream indi("indices/"+this->archivo+"/"+this->archivo+"-"+numero(c)+".txt"); |
|  | ofstream ClusterKeys ("indices/"+this->archivo+"/ClusterKey-"+this->archivo+"-"+numero(c)+".txt"); |
|  | for(int i=0;i<ID.size();i++) |
|  | { |
|  |  |
|  | if(temp!=ID[i].ID &&temp<ID[i].ID) |
|  | { |
|  | if(cont==5000) |
|  | { |
|  | posicion=indi.tellp(); |
|  | ClusterKeys<<ID[i].ID<<','<<posicion<<endl; |
|  | cont=1; |
|  | } |
|  | HashKey<<ID[i].ID<<','<<indi.tellp()<<endl; |
|  | cont++; |
|  | temp=ID[i].ID; |
|  | } |
|  | indi<< ID[i].ID <<','<< ID[i].posicion <<','<< ID[i].Parchivo <<endl; |
|  | } |
|  | HashKey.close(); |
|  | indi.close(); |
|  | ClusterKeys.close(); |
|  | ID.clear(); |
|  | c++; |
|  | } |
|  |  |
|  | archivo.getline(linea,1000); |
|  | posicion=archivo.tellg(); |
|  | posi=0; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | sort(ID.begin(),ID.end(),[](const Datos& datos1,const Datos& datos2){return (datos1.ID<datos2.ID);}); |
|  | ofstream indi("indices/"+this->archivo+"/"+this->archivo+"-"+numero(c)+".txt"); |
|  | ofstream ClusterKeys ("indices/"+this->archivo+"/ClusterKey-"+this->archivo+"-"+numero(c)+".txt"); |
|  | for(int i=0;i<ID.size();i++) |
|  | { |
|  | if(temp!=ID[i].ID &&temp<ID[i].ID) |
|  | { |
|  | posicion=indi.tellp(); |
|  | cont++; |
|  | if(cont==5000) |
|  | { |
|  | ClusterKeys<<ID[i].ID<<','<<posicion<<endl; |
|  | cont=1; |
|  | } |
|  | temp=ID[i].ID; |
|  | } |
|  | indi<< ID[i].ID <<','<< ID[i].posicion <<','<< ID[i].Parchivo <<endl; |
|  | } |
|  | ClusterKeys<<ID[ID.size()-1].ID<<','<<posicion; |
|  | indi.close(); |
|  | ClusterKeys.close(); |
|  | ID.clear(); |
|  | } |
|  | void task\_events::indicesTE() |
|  | { |
|  | char \*linea=new char[1000]; |
|  | long int posi=0,posicion,posiHash,temp=-1,cont=0; |
|  | int c=0; |
|  | vector<Datos> ID; |
|  | ID.shrink\_to\_fit(); |
|  | for(int i=0;i<this->cantidad;i++) |
|  | { |
|  | string num=numero(i); |
|  | string url=("google/"+this->archivo+"/part-"+num+"-of-"+numero(this->cantidad)+".csv"); |
|  | ifstream archivo (url,std::fstream::binary); |
|  | if (archivo == NULL) |
|  | { |
|  | cout<<"Error al abrir el archivo"<<endl; |
|  | return; |
|  | } |
|  | else |
|  | { |
|  | posicion=archivo.tellg(); |
|  | while(!archivo.eof() ) |
|  | { |
|  | if(posi!=3) |
|  | { |
|  | archivo.getline(linea,1000,','); |
|  | posi++; |
|  | } |
|  | else |
|  | { |
|  | long long int dato=atoll(linea); |
|  | Datos neew(dato,posicion,i); |
|  | ID.push\_back(neew); |
|  | if(ID.size()>33461309) |
|  | { |
|  | cont=0; |
|  | sort(ID.begin(),ID.end(),[](const Datos& datos1,const Datos& datos2){return (datos1.ID<datos2.ID);}); |
|  | ofstream indi("indices/"+this->archivo+"/"+this->archivo+"-"+numero(c)+".txt"); |
|  | ofstream ClusterKeys ("indices/"+this->archivo+"/ClusterKey-"+this->archivo+"-"+numero(c)+".txt"); |
|  | for(int i=0;i<ID.size();i++) |
|  | { |
|  |  |
|  | if(temp!=ID[i].ID &&temp<ID[i].ID) |
|  | { |
|  | if(cont==5000) |
|  | { |
|  | posicion=indi.tellp(); |
|  | ClusterKeys<<ID[i].ID<<','<<posicion<<endl; |
|  | cont=1; |
|  | } |
|  | cont++; |
|  | temp=ID[i].ID; |
|  | } |
|  | indi<< ID[i].ID <<','<< ID[i].posicion <<','<< ID[i].Parchivo <<endl; |
|  | } |
|  |  |
|  | indi.close(); |
|  | ClusterKeys.close(); |
|  | ID.clear(); |
|  | c++; |
|  | } |
|  |  |
|  | archivo.getline(linea,1000); |
|  | posicion=archivo.tellg(); |
|  | posi=0; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | sort(ID.begin(),ID.end(),[](const Datos& datos1,const Datos& datos2){return (datos1.ID<datos2.ID);}); |
|  | ofstream indi("indices/"+this->archivo+"/"+this->archivo+"-"+numero(c)+".txt"); |
|  | ofstream ClusterKeys ("indices/"+this->archivo+"/ClusterKey-"+this->archivo+"-"+numero(c)+".txt"); |
|  | for(int i=0;i<ID.size();i++) |
|  | { |
|  | if(temp!=ID[i].ID &&temp<ID[i].ID) |
|  | { |
|  | posicion=indi.tellp(); |
|  | cont++; |
|  | if(cont==5000) |
|  | { |
|  | ClusterKeys<<ID[i].ID<<','<<posicion<<endl; |
|  | cont=1; |
|  | } |
|  | temp=ID[i].ID; |
|  | } |
|  | indi<< ID[i].ID <<','<< ID[i].posicion <<','<< ID[i].Parchivo <<endl; |
|  | } |
|  | ClusterKeys<<ID[ID.size()-1].ID<<','<<posicion; |
|  | indi.close(); |
|  | ClusterKeys.close(); |
|  | ID.clear(); |
|  | } |
|  | void task\_constraints::indicesTC() |
|  | { |
|  | char \*linea=new char[1000]; |
|  | long int posi=0,posicion; |
|  | long long int dato,temp=-1; |
|  | vector<Datos> ID; |
|  | for(int i=0;i<this->cantidad;i++) |
|  | { |
|  | string num=numero(i); |
|  | string url=("google/"+this->archivo+"/part-"+num+"-of-"+numero(this->cantidad)+".csv"); |
|  | ifstream archivo (url,std::fstream::binary); |
|  | if (archivo == NULL) |
|  | { |
|  | cout<<"Error al abrir el archivo"<<endl; |
|  | return; |
|  | } |
|  | else |
|  | { |
|  | posicion=archivo.tellg(); |
|  | while(!archivo.eof() ) |
|  | { |
|  | if(posi!=this->indice) |
|  | { |
|  | archivo.getline(linea,1000,','); |
|  | posi++; |
|  | } |
|  | else |
|  | { |
|  | dato=atoll(linea); |
|  | Datos neew(dato,posicion,i); |
|  | ID.push\_back(neew); |
|  | archivo.getline(linea,1000); |
|  | posicion=archivo.tellg(); |
|  | posi=0; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | sort(ID.begin(),ID.end(),[](const Datos& datos1,const Datos& datos2){return (datos1.ID<datos2.ID);}); |
|  | ofstream indi("indices/"+this->archivo+".txt"); |
|  | ofstream HashKeys ("indices/HashKey-"+this->archivo+".txt"); |
|  | for(int i=0;i<ID.size();i++) |
|  | { |
|  | if(temp!=ID[i].ID &&temp<ID[i].ID) |
|  | { |
|  | temp=ID[i].ID; |
|  | posicion=indi.tellp(); |
|  | HashKeys<<temp<<','<<posicion<<endl; |
|  | } |
|  | indi<< ID[i].ID <<','<< ID[i].posicion <<','<< ID[i].Parchivo <<endl; |
|  | } |
|  | indi.close(); |
|  | HashKeys.close(); |
|  | cout<<"Indexacion Realizada Correctamente :3"<<endl; |
|  | } |
|  | void machine\_events::indicesME() |
|  | { |
|  | char \*linea=new char[1000]; |
|  | long int posi=0,posicion; |
|  | long long int dato,temp=-1; |
|  | vector<Datos> ID; |
|  | for(int i=0;i<this->cantidad;i++) |
|  | { |
|  | string num=numero(i); |
|  | string url=("google/"+this->archivo+"/part-"+num+"-of-"+numero(this->cantidad)+".csv"); |
|  | ifstream archivo (url,std::fstream::binary); |
|  | if (archivo == NULL) |
|  | { |
|  | cout<<"Error al abrir el archivo"<<endl; |
|  | return; |
|  | } |
|  | else |
|  | { |
|  | posicion=archivo.tellg(); |
|  | while(!archivo.eof() ) |
|  | { |
|  | if(posi!=this->indice) |
|  | { |
|  | archivo.getline(linea,1000,','); |
|  | posi++; |
|  | } |
|  | else |
|  | { |
|  | dato=atoll(linea); |
|  | Datos neew(dato,posicion,i); |
|  | ID.push\_back(neew); |
|  | archivo.getline(linea,1000); |
|  | posicion=archivo.tellg(); |
|  | posi=0; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | sort(ID.begin(),ID.end(),[](const Datos& datos1,const Datos& datos2){return (datos1.ID<datos2.ID);}); |
|  | ofstream indi("indices/"+this->archivo+".txt"); |
|  | ofstream HashKeys ("indices/HashKey-"+this->archivo+".txt"); |
|  | for(int i=0;i<ID.size();i++) |
|  | { |
|  | if(temp!=ID[i].ID &&temp<ID[i].ID) |
|  | { |
|  | temp=ID[i].ID; |
|  | posicion=indi.tellp(); |
|  | HashKeys<<temp<<','<<posicion<<endl; |
|  | } |
|  | indi<< ID[i].ID <<','<< ID[i].posicion <<','<< ID[i].Parchivo <<endl; |
|  | } |
|  | indi.close(); |
|  | HashKeys.close(); |
|  | cout<<"Indexacion Realizada Correctamente :3"<<endl; |
|  | } |
|  | void machine\_attributes::indicesMA() |
|  | { |
|  | char \*linea=new char[1000]; |
|  | long int posi=0,posicion; |
|  | long long int dato,temp=-1; |
|  | vector<Datos> ID; |
|  | for(int i=0;i<this->cantidad;i++) |
|  | { |
|  | string num=numero(i); |
|  | string url=("google/"+this->archivo+"/part-"+num+"-of-"+numero(this->cantidad)+".csv"); |
|  | ifstream archivo (url,std::fstream::binary); |
|  | if (archivo == NULL) |
|  | { |
|  | cout<<"Error al abrir el archivo"<<endl; |
|  | return; |
|  | } |
|  | else |
|  | { |
|  | posicion=archivo.tellg(); |
|  | while(!archivo.eof() ) |
|  | { |
|  | if(posi!=this->indice) |
|  | { |
|  | archivo.getline(linea,1000,','); |
|  | posi++; |
|  | } |
|  | else |
|  | { |
|  | dato=atoll(linea); |
|  | Datos neew(dato,posicion,i); |
|  | ID.push\_back(neew); |
|  | archivo.getline(linea,1000); |
|  | posicion=archivo.tellg(); |
|  | posi=0; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | sort(ID.begin(),ID.end(),[](const Datos& datos1,const Datos& datos2){return (datos1.ID<datos2.ID);}); |
|  | ofstream indi("indices/"+this->archivo+".txt"); |
|  | ofstream HashKeys ("indices/HashKey-"+this->archivo+".txt"); |
|  | for(int i=0;i<ID.size();i++) |
|  | { |
|  | if(temp!=ID[i].ID &&temp<ID[i].ID) |
|  | { |
|  | temp=ID[i].ID; |
|  | posicion=indi.tellp(); |
|  | HashKeys<<temp<<','<<posicion<<endl; |
|  | } |
|  | indi<< ID[i].ID <<','<< ID[i].posicion <<','<< ID[i].Parchivo <<endl; |
|  | } |
|  | indi.close(); |
|  | HashKeys.close(); |
|  | cout<<"Indexacion Realizada Correctamente :3"<<endl; |
|  | } |
|  | void job\_events::indicesJE() |
|  | { |
|  | char \*linea=new char[1000]; |
|  | long int posi=0,posicion; |
|  | long long int dato,temp=-1; |
|  | vector<Datos> ID; |
|  | for(int i=0;i<this->cantidad;i++) |
|  | { |
|  | string num=numero(i); |
|  | string url=("google/"+this->archivo+"/part-"+num+"-of-"+numero(this->cantidad)+".csv"); |
|  | ifstream archivo (url,std::fstream::binary); |
|  | if (archivo == NULL) |
|  | { |
|  | cout<<"Error al abrir el archivo"<<endl; |
|  | return; |
|  | } |
|  | else |
|  | { |
|  | posicion=archivo.tellg(); |
|  | while(!archivo.eof() ) |
|  | { |
|  | if(posi!=this->indice) |
|  | { |
|  | archivo.getline(linea,1000,','); |
|  | posi++; |
|  | } |
|  | else |
|  | { |
|  | dato=atoll(linea); |
|  | Datos neew(dato,posicion,i); |
|  | ID.push\_back(neew); |
|  | archivo.getline(linea,1000); |
|  | posicion=archivo.tellg(); |
|  | posi=0; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | sort(ID.begin(),ID.end(),[](const Datos& datos1,const Datos& datos2){return (datos1.ID<datos2.ID);}); |
|  | ofstream indi("indices/"+this->archivo+".txt"); |
|  | ofstream HashKeys ("indices/HashKey-"+this->archivo+".txt"); |
|  | for(int i=0;i<ID.size();i++) |
|  | { |
|  | if(temp!=ID[i].ID &&temp<ID[i].ID) |
|  | { |
|  | temp=ID[i].ID; |
|  | posicion=indi.tellp(); |
|  | HashKeys<<temp<<','<<posicion<<endl; |
|  | } |
|  | indi<< ID[i].ID <<','<< ID[i].posicion <<','<< ID[i].Parchivo <<endl; |
|  | } |
|  | indi.close(); |
|  | HashKeys.close(); |
|  | cout<<"Indexacion Realizada Correctamente :3"<<endl; |
|  | } |

**Earbol.h**

|  |
| --- |
|  |
|  |  |
|  | #include "includes.h"  int ComparisonCount = 0; |
|  |  |
|  | class BTreeNode |
|  | { |
|  | long long int \*keys; |
|  | long long int t; |
|  | BTreeNode \*\*C; |
|  | int n; |
|  | bool leaf; |
|  | public: |
|  | BTreeNode(int \_t, bool \_leaf); |
|  | void insertNonFull(long long int k); |
|  | void splitChild(int i, BTreeNode \*y); |
|  | void traverse(); |
|  | BTreeNode \*search(long long int k); |
|  | friend class BTree; |
|  | }; |
|  |  |
|  | // A BTree |
|  | class BTree |
|  | { |
|  | BTreeNode \*root; |
|  | int t; |
|  | public: |
|  |  |
|  | BTree(int \_t) |
|  | { |
|  | root = NULL; t = \_t; |
|  | } |
|  | void traverse() |
|  | { |
|  | if (root != NULL) root->traverse(); |
|  | } |
|  | BTreeNode\* search(long long int k) |
|  | { |
|  | return (root == NULL) ? NULL : root->search(k); |
|  | } |
|  | void insert(long long int k); |
|  | }; |
|  |  |
|  | BTreeNode::BTreeNode(int t1, bool leaf1) |
|  | { |
|  | t = t1; |
|  | leaf = leaf1; |
|  | keys = new long long int[2 \* t - 1]; |
|  | C = new BTreeNode \*[2 \* t]; |
|  | n = 0; |
|  | } |
|  |  |
|  | BTreeNode \*BTreeNode::search(long long int k) |
|  | { |
|  | int i = 0; |
|  | while (i < n && k > keys[i]) |
|  | i++; |
|  | if (keys[i] == k) |
|  | { |
|  | ComparisonCount++; |
|  | return this; |
|  | } |
|  | if (leaf == true) |
|  | { |
|  | ComparisonCount++; |
|  | return NULL; |
|  | } |
|  |  |
|  | return C[i]->search(k); |
|  | } |
|  |  |
|  |  |
|  | void BTree::insert(long long int k) |
|  | { |
|  | if (root == NULL) |
|  | { |
|  | ComparisonCount++; |
|  | root = new BTreeNode(t, true); |
|  | root->keys[0] = k; |
|  | root->n = 1; |
|  | } |
|  | else |
|  | { |
|  | if (root->n == 2 \* t - 1) |
|  | { |
|  | ComparisonCount++; |
|  | BTreeNode \*s = new BTreeNode(t, false); |
|  | s->C[0] = root; |
|  | s->splitChild(0, root); |
|  | int i = 0; |
|  | if (s->keys[0] < k) |
|  | { |
|  | ComparisonCount++; |
|  | i++; |
|  | }s->C[i]->insertNonFull(k); |
|  |  |
|  | root = s; |
|  | } |
|  | else |
|  | root->insertNonFull(k); |
|  | } |
|  | } |
|  |  |
|  | void BTreeNode::insertNonFull(long long int k) |
|  | { |
|  | int i = n - 1; |
|  |  |
|  | if (leaf == true) |
|  | { |
|  | ComparisonCount++; |
|  | while (i >= 0 && keys[i] > k) |
|  | { |
|  | keys[i + 1] = keys[i]; |
|  | i--; |
|  | } |
|  |  |
|  | keys[i + 1] = k; |
|  | n = n + 1; |
|  | } |
|  | else // If this node is not leaf |
|  | { |
|  | while (i >= 0 && keys[i] > k) |
|  | i--; |
|  | if (C[i + 1]->n == 2 \* t - 1) |
|  | { |
|  | ComparisonCount++; |
|  | splitChild(i + 1, C[i + 1]); |
|  | if (keys[i + 1] < k) |
|  | i++; |
|  | } |
|  | C[i + 1]->insertNonFull(k); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | void BTreeNode::splitChild(int i, BTreeNode \*y){ |
|  |  |
|  | BTreeNode \*z = new BTreeNode(y->t, y->leaf); |
|  | z->n = t - 1; |
|  |  |
|  | for (int j = 0; j < t - 1; j++) |
|  | z->keys[j] = y->keys[j + t]; |
|  |  |
|  | if (y->leaf == false) |
|  | { |
|  | ComparisonCount++; |
|  | for (int j = 0; j < t; j++) |
|  | z->C[j] = y->C[j + t]; |
|  | } |
|  |  |
|  | y->n = t - 1; |
|  | for (int j = n; j >= i + 1; j--) |
|  | C[j + 1] = C[j]; |
|  | C[i + 1] = z; |
|  | for (int j = n - 1; j >= i; j--) |
|  | keys[j + 1] = keys[j]; |
|  | keys[i] = y->keys[t - 1]; |
|  |  |
|  | n = n + 1; |
|  | } |
|  | void BTreeNode::traverse() |
|  | { |
|  | std::queue<BTreeNode\*> queue; |
|  | queue.push(this); |
|  | while (!queue.empty()) |
|  | { |
|  | BTreeNode\* current = queue.front(); |
|  | queue.pop(); |
|  | int i; |
|  | for (i = 0; i < current->n; i++) //\* |
|  | { |
|  | if (current->leaf == false) //\* |
|  | { |
|  | ComparisonCount++; |
|  | queue.push(current->C[i]); |
|  | }cout << " " << current->keys[i] << endl; |
|  | } |
|  | if (current->leaf == false) //\* |
|  | { |
|  | ComparisonCount++; |
|  | queue.push(current->C[i]); |
|  | } |
|  | } |
|  | } |

**Hash.h**

|  |
| --- |
|  |
|  | #include "includes.h"  const int TABLE\_SIZE = 100000; |
|  | class LinkedHashEntry { |
|  | private: |
|  | LinkedHashEntry \*next; |
|  | public: |
|  | int ID; |
|  | int posicion; |
|  | int PArchivo; |
|  | LinkedHashEntry(int ID,int posicion,int PArchivo) { |
|  | this->ID = ID; |
|  | this->posicion=posicion; |
|  | this->next = NULL; |
|  | } |
|  | int getValue() { return ID; } |
|  | int getposicion(){return posicion;} |
|  | LinkedHashEntry \*getNext() { return next; } |
|  | void setNext(LinkedHashEntry \*next) { this->next = next; } |
|  | }; |
|  |  |
|  | class HashMap { |
|  | private: |
|  | LinkedHashEntry \*\*table; |
|  |  |
|  | public: |
|  | int tam=0; |
|  | void AmpliTam(){tam++;} |
|  | HashMap() |
|  | { |
|  | table = new LinkedHashEntry\*[TABLE\_SIZE]; |
|  | for (int i = 0; i < TABLE\_SIZE; i++) |
|  | table[i] = NULL; |
|  | } |
|  | int get(int value) |
|  | { |
|  | LinkedHashEntry \*A; |
|  | for(int i=0;i<tam;i++) |
|  | { |
|  | A=table[i]; |
|  | while(A) |
|  | { |
|  | if(A->ID==value) |
|  | { |
|  | return i; |
|  | } |
|  | A=A->getNext(); |
|  | } |
|  | } |
|  | return -1; |
|  | } |
|  | void CountAdelan(int value); |
|  | void put(int value,string destino); |
|  | void despegar(int value); |
|  | void mostrar(); |
|  | ~HashMap() |
|  | { |
|  | for (int i = 0; i < TABLE\_SIZE; i++) |
|  | if (table[i] != NULL) |
|  | { |
|  | LinkedHashEntry \*prevEntry = NULL; |
|  | LinkedHashEntry \*entry = table[i]; |
|  | while (entry != NULL) { |
|  | prevEntry = entry; |
|  | entry = entry->getNext(); |
|  | delete prevEntry; |
|  | } |
|  | } |
|  | delete[] table; |
|  | } |
|  | }; |

**Tablas.h**

|  |
| --- |
|  |
|  | #include "Hash.h"  #include "FBasicas.h" |
|  | #include "PrymariQueue.h" |
|  | class job\_events |
|  | { |
|  | public: |
|  | int timespamp; |
|  | string missing\_info; |
|  | int job\_ID; |
|  | string event\_type; |
|  | string user\_name; |
|  | string scheduling\_class; |
|  | string job\_name; |
|  | string logical\_job\_name; |
|  |  |
|  | int cantidad=500; |
|  | string archivo="job\_events"; |
|  | int indice=3; |
|  | int datos=2012242; |
|  |  |
|  | bool bloqueada=false; |
|  | void bloquear(); |
|  | void desbloquear(); |
|  | bool Estado(); |
|  | void indicesJE(); |
|  | string insertJE(string ID,string newline); |
|  | char \*selectJE(string ID); |
|  | string update(string ID,string NewLine); |
|  | string Delete(string ID); |
|  | string NewLine(string ID); |
|  | }; |
|  | class machine\_attributes |
|  | { |
|  | public: |
|  | int timestamp; |
|  | int machine\_ID; |
|  | string attribute\_name; |
|  | string attribute\_value; |
|  | bool attribute\_deleted; |
|  |  |
|  | int cantidad=1; |
|  | string archivo="machine\_attributes"; |
|  | int indice=2; |
|  | bool bloqueada=false; |
|  |  |
|  | void bloquear(); |
|  | void desbloquear(); |
|  | bool Estado(); |
|  | void indicesMA(); |
|  | string insertMA(string ID,string newline); |
|  | char \*selectMA(string ID); |
|  | string update(string ID,string NewLine); |
|  | string Delete(string ID); |
|  | string NewLine(string ID); |
|  | }; |
|  | class machine\_events |
|  | { |
|  | public: |
|  | int timestamp; |
|  | int machine\_ID; |
|  | string event\_type; |
|  | string platform\_ID; |
|  | string capacity\_CPU; |
|  | string capacity\_memory; |
|  |  |
|  | int cantidad=1; |
|  | string archivo="machine\_events"; |
|  | int indice=2; |
|  | bool bloqueada=false; |
|  |  |
|  | void bloquear(); |
|  | void desbloquear(); |
|  | bool Estado(); |
|  | void indicesME(); |
|  | string insertME(string ID,string newline); |
|  | string selectME(string ID); |
|  | string update(string ID,string NewLine); |
|  | string Delete(string ID); |
|  | string NewLine(string ID); |
|  | }; |
|  | class task\_constraints |
|  | { |
|  | public: |
|  | int timestamp; |
|  | int job\_ID; |
|  | int task\_index; |
|  | string attribute\_name; |
|  | string comparison\_operator; |
|  | string attribute\_value; |
|  |  |
|  | int cantidad=500; |
|  | string archivo="task\_constraints"; |
|  | int indice=2; |
|  | bool bloqueada=false; |
|  |  |
|  | void bloquear(); |
|  | void desbloquear(); |
|  | bool Estado(); |
|  | void indicesTC(); |
|  | string insertTC(string ID,string newline); |
|  | char \*selectTC(string ID); |
|  | string update(string ID,string NewLine); |
|  | string Delete(string ID); |
|  | string NewLine(string ID); |
|  | }; |
|  | class task\_events |
|  | { |
|  | public: |
|  | int timestamp; |
|  | string missing\_info; |
|  | int job\_ID; |
|  | string task\_index; |
|  | int machine\_ID; |
|  | string event\_type; |
|  | string user\_name; |
|  | string scheduling\_class; |
|  | int priority; |
|  | string resource\_request\_for\_CPU\_cores; |
|  | string resource\_request\_for\_RAM; |
|  | string resource\_request\_for\_local\_disk\_space; |
|  | string different\_machine\_constraint; |
|  |  |
|  | int cantidad=500; |
|  | string archivo="task\_events"; |
|  | int indice=3; |
|  | bool bloqueada=false; |
|  | void bloquear(); |
|  | void desbloquear(); |
|  | bool Estado(); |
|  | void indicesTE(); |
|  | void indices2TE(); |
|  | string insertTE(string ID,string newline); |
|  | char \*selectTE(string ID); |
|  | string update(string ID,string NewLine); |
|  | string Delete(string ID); |
|  | string NewLine(string ID); |
|  | }; |
|  | class task\_usage |
|  | { |
|  | public: |
|  | int start\_time; |
|  | int end\_time; |
|  | int task\_index; |
|  | string machine\_ID; |
|  | string mean\_CPU\_usage\_rate; |
|  | string canonical\_memory\_usage; |
|  | string assigned\_memory\_usage; |
|  | string unmapped\_page\_cache\_memory\_usage; |
|  | string total\_page\_cache\_memory\_usage; |
|  | string maximun\_memory\_usage; |
|  | string mean\_disk\_IO\_time; |
|  | string mean\_local\_disk\_space\_used; |
|  | string maximun\_CPU\_usage; |
|  | string maximun\_disk\_IO\_time; |
|  | string cycles\_per\_instruction\_CPI; |
|  | string memory\_accesses\_per\_instruction\_MAI; |
|  | string sample\_portion; |
|  | string aggregation\_type; |
|  | string sampled\_CPU\_usage; |
|  |  |
|  | int cantidad=500; |
|  | string archivo="task\_usage"; |
|  | int indice=4; |
|  | bool bloqueada=false; |
|  | void bloquear(); |
|  | void desbloquear(); |
|  | bool Estado(); |
|  | void indicesTU(); |
|  | string insertTU(string ID,string newline); |
|  | char \*selectTU(string ID); |
|  | string update(string ID,string NewLine); |
|  | string Delete(string ID); |
|  | string NewLine(string ID); |
|  | }; |

**Bloqueos.h**

|  |
| --- |
|  |
|  |  |
|  | #include "update.h"  void machine\_attributes::bloquear() |
|  | { |
|  | this->bloqueada=true; |
|  | } |
|  | void machine\_attributes::desbloquear() |
|  | { |
|  | this->bloqueada=false; |
|  | } |
|  | void task\_usage::bloquear() |
|  | { |
|  | this->bloqueada=true; |
|  | } |
|  | void task\_usage::desbloquear() |
|  | { |
|  | this->bloqueada=false; |
|  | } |
|  | void machine\_events::bloquear() |
|  | { |
|  | this->bloqueada=true; |
|  | } |
|  | void machine\_events::desbloquear() |
|  | { |
|  | this->bloqueada=false; |
|  | } |
|  | void task\_constraints::bloquear() |
|  | { |
|  | this->bloqueada=true; |
|  | } |
|  | void task\_constraints::desbloquear() |
|  | { |
|  | this->bloqueada=false; |
|  | } |
|  | void task\_events::bloquear() |
|  | { |
|  | this->bloqueada=true; |
|  | } |
|  | void task\_events::desbloquear() |
|  | { |
|  | this->bloqueada=false; |
|  | } |
|  | void job\_events::bloquear() |
|  | { |
|  | this->bloqueada=true; |
|  | } |
|  | void job\_events::desbloquear() |
|  | { |
|  | this->bloqueada=false; |
|  | } |
|  |  |
|  | bool machine\_attributes::Estado() |
|  | { |
|  | return this->bloqueada; |
|  | } |
|  | bool machine\_events::Estado() |
|  | { |
|  | return this->bloqueada; |
|  | } |
|  | bool task\_constraints::Estado() |
|  | { |
|  | return this->bloqueada; |
|  | } |
|  | bool task\_events::Estado() |
|  | { |
|  | return this->bloqueada; |
|  | } |
|  | bool job\_events::Estado() |
|  | { |
|  | return this->bloqueada; |
|  | } |
|  | bool task\_usage::Estado() |
|  | { |
|  | return this->bloqueada; |
|  | } |

**Estructuras.h**

|  |
| --- |
|  |
|  |  |
|  | #include <queue>  using namespace std; |
|  |  |
|  |  |
|  | class funciones |
|  | { |
|  | public: |
|  | string ID; |
|  | string carpeta; |
|  | string newline=NULL; |
|  | int funcion; |
|  | funciones(): ID(""),carpeta(""),newline(""),funcion(0){} |
|  | funciones(string ID,string carpeta,int funcion):ID(ID),carpeta(carpeta),newline(""),funcion(funcion){} |
|  | funciones(string ID,string carpeta,string newline,int funcion):ID(ID),carpeta(carpeta),newline(newline),funcion(funcion){} |
|  | }; |
|  | bool operator< (const funciones& structfun1, const funciones &structfun2) |
|  | { |
|  | return structfun1.funcion > structfun2.funcion; |
|  | } |
|  |  |
|  | bool operator> (const funciones& structfun1, const funciones &structfun2) |
|  | { |
|  | return structfun1.funcion > structfun2.funcion; |
|  | } |
|  |  |
|  | class Datos { |
|  | public: |
|  | long long int ID; |
|  | int posicion; |
|  | int Parchivo; |
|  | Datos(): ID(0),posicion(0),Parchivo(0){} |
|  | Datos( long long int ID, int posicion,int Parchivo ):ID(ID), posicion(posicion),Parchivo(Parchivo){} |
|  | }; |
|  |  |
|  | bool operator< (const Datos& structdatos1, const Datos &structdatos2) |
|  | { |
|  | return structdatos1.ID > structdatos2.ID; |
|  | } |
|  |  |
|  | bool operator> (const Datos& structdatos1, const Datos &structdatos2) |
|  | { |
|  | return structdatos1.ID < structdatos2.ID; |
|  | } |

Fbasicas.h

|  |
| --- |
|  |
|  |  |
|  | #include "includes.h"  string numero(int n) |
|  | { |
|  | vector<char> hola={'0','0','0','0','0'}; |
|  | vector<int> nums; |
|  | string num=""; |
|  | char nchar; |
|  | if(n<=9) |
|  | { |
|  | nchar=n+48; |
|  | hola[hola.size()-1]=nchar; |
|  | } |
|  | else |
|  | { |
|  | if(n<=99) |
|  | { |
|  | for(int i=0;i<2;i++) |
|  | { |
|  | int aux=n%10; |
|  | hola[hola.size()-1-i]=aux+48; |
|  | n=n/10; |
|  | } |
|  | } |
|  | else if(n<=999) |
|  | { |
|  | for(int i=0;i<3;i++) |
|  | { |
|  | int aux=n%10; |
|  | hola[hola.size()-1-i]=aux+48; |
|  | n=n/10; |
|  | } |
|  | } |
|  | else if(n<=9999) |
|  | { |
|  | for(int i=0;i<4;i++) |
|  | { |
|  | int aux=n%10; |
|  | hola[hola.size()-1-i]=aux+48; |
|  | n=n/10; |
|  | } |
|  | } |
|  | else if(n<=99999) |
|  | { |
|  | for(int i=0;i<5;i++) |
|  | { |
|  | int aux=n%10; |
|  | hola[hola.size()-1-i]=aux+48; |
|  | n=n/10; |
|  | } |
|  | } |
|  | } |
|  | for(auto n:hola) num+=n; |
|  | return num; |
|  | } |
|  | bool esNumero(char\* num) |
|  | { |
|  |  |
|  | for(int i=0;i<strlen(num);i++) |
|  | { |
|  | if (!isdigit(num[i])) |
|  | return false; |
|  | } |
|  | return true; |
|  | } |

**Select.h**

|  |
| --- |
|  |
|  |  |
|  |  |
|  | #include "indices.h"  char \*job\_events::selectJE(string ID) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | int divi=0; |
|  | string ids; |
|  | bool encontrado=false; |
|  | ifstream HashKey("indices/HashKey-"+this->archivo+".txt"); |
|  | if (!HashKey) |
|  | { |
|  |  |
|  | return ("No se encuentra la tabla a buscar"); |
|  | } |
|  | else |
|  | { |
|  | while(!HashKey.eof()) |
|  | { |
|  | HashKey.getline(linea,10000,','); |
|  | ids=linea; |
|  | if(ID==ids) |
|  | { |
|  | HashKey.getline(linea,10000); |
|  | encontrado=true; |
|  | break; |
|  | } |
|  | else |
|  | HashKey.getline(linea,10000); |
|  | } |
|  | if(encontrado) |
|  | { |
|  | string res; |
|  | long int n=atoi(linea),s; |
|  | ifstream csv("indices/"+this->archivo+".txt"); |
|  | csv.seekg(n,ios::beg); |
|  | csv.getline(linea,10000,','); |
|  | while(ID==linea) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | csv.getline(seg,1000); |
|  | n=atoi(linea); |
|  | s=atoi(seg); |
|  | ifstream is("google/"+this->archivo+"/part-"+numero(s)+"-of-"+numero(this->cantidad)+".csv",std::fstream::binary); |
|  | is.seekg(n,ios::beg); |
|  | is.getline(linea,1000); |
|  | if(linea[0]==' ') |
|  | cout<<"Esta linea de registro fue eliminada"<<endl; |
|  | else |
|  | cout<<linea<<endl; |
|  | csv.getline(linea,1000,','); |
|  | } |
|  | return "Seleccion hecha correctamente :3"; |
|  | } |
|  | else |
|  | { |
|  | return ("NO SE ENCONTRO INDICE A BUSCAR!!!!GG"); |
|  | } |
|  |  |
|  | } |
|  | } |
|  | char \*machine\_attributes::selectMA(string ID) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | int divi=0; |
|  | string ids; |
|  | bool encontrado=false; |
|  | ifstream HashKey("indices/HashKey-"+this->archivo+".txt"); |
|  | if (!HashKey) |
|  | { |
|  |  |
|  | return ("No se encuentra la tabla a buscar"); |
|  | } |
|  | else |
|  | { |
|  | while(!HashKey.eof()) |
|  | { |
|  | HashKey.getline(linea,10000,','); |
|  | ids=linea; |
|  | if(ID==ids) |
|  | { |
|  | HashKey.getline(linea,10000); |
|  | encontrado=true; |
|  | break; |
|  | } |
|  | else |
|  | HashKey.getline(linea,10000); |
|  | } |
|  | if(encontrado) |
|  | { |
|  | string res; |
|  | long int n=atoi(linea),s; |
|  | ifstream csv("indices/"+this->archivo+".txt"); |
|  | csv.seekg(n,ios::beg); |
|  | csv.getline(linea,10000,','); |
|  | while(ID==linea) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | csv.getline(seg,1000); |
|  | n=atoi(linea); |
|  | s=atoi(seg); |
|  | ifstream is("google/"+this->archivo+"/part-"+numero(s)+"-of-"+numero(this->cantidad)+".csv",std::fstream::binary); |
|  | is.seekg(n,ios::beg); |
|  | is.getline(linea,1000); |
|  | if(linea[0]==' ') |
|  | cout<<"Esta linea de registro fue eliminada"<<endl; |
|  | else |
|  | cout<<linea<<endl; |
|  | csv.getline(linea,1000,','); |
|  | } |
|  | return "Seleccion hecha correctamente :3"; |
|  | } |
|  | else |
|  | { |
|  | return ("NO SE ENCONTRO INDICE A BUSCAR!!!!GG"); |
|  | } |
|  |  |
|  | } |
|  | } |
|  | string machine\_events::selectME(string ID) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | int divi=0; |
|  | string ids; |
|  | bool encontrado=false; |
|  | ifstream HashKey("indices/HashKey-"+this->archivo+".txt"); |
|  | if (!HashKey) |
|  | { |
|  |  |
|  | return ("No se encuentra la tabla a buscar"); |
|  | } |
|  | else |
|  | { |
|  | while(!HashKey.eof()) |
|  | { |
|  | HashKey.getline(linea,10000,','); |
|  | ids=linea; |
|  | if(ID==ids) |
|  | { |
|  | HashKey.getline(linea,10000); |
|  | encontrado=true; |
|  | break; |
|  | } |
|  | else |
|  | HashKey.getline(linea,10000); |
|  | } |
|  | if(encontrado) |
|  | { |
|  | string res; |
|  | long int n=atoi(linea),s; |
|  | ifstream csv("indices/"+this->archivo+".txt"); |
|  | csv.seekg(n,ios::beg); |
|  | csv.getline(linea,10000,','); |
|  | while(ID==linea) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | csv.getline(seg,1000); |
|  | n=atoi(linea); |
|  | s=atoi(seg); |
|  | ifstream is("google/"+this->archivo+"/part-"+numero(s)+"-of-"+numero(this->cantidad)+".csv",std::fstream::binary); |
|  | is.seekg(n,ios::beg); |
|  | is.getline(linea,1000); |
|  | if(linea[0]==' ') |
|  | cout<<"Esta linea de registro fue eliminada"<<endl; |
|  | else |
|  | cout<<linea<<endl; |
|  | csv.getline(linea,1000,','); |
|  | } |
|  | return "Seleccion hecha correctamente :3"; |
|  | } |
|  | else |
|  | { |
|  | return ("NO SE ENCONTRO INDICE A BUSCAR!!!!GG"); |
|  | } |
|  |  |
|  | } |
|  | } |
|  | char \*task\_constraints::selectTC(string ID) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | int divi=0; |
|  | string ids; |
|  | bool encontrado=false; |
|  | ifstream HashKey("indices/HashKey-"+this->archivo+".txt"); |
|  | if (!HashKey) |
|  | { |
|  |  |
|  | return ("No se encuentra la tabla a buscar"); |
|  | } |
|  | else |
|  | { |
|  | while(!HashKey.eof()) |
|  | { |
|  | HashKey.getline(linea,10000,','); |
|  | ids=linea; |
|  | if(ID==ids) |
|  | { |
|  | HashKey.getline(linea,10000); |
|  | encontrado=true; |
|  | break; |
|  | } |
|  | else |
|  | HashKey.getline(linea,10000); |
|  | } |
|  | if(encontrado) |
|  | { |
|  | string res; |
|  | long int n=atoi(linea),s; |
|  | ifstream csv("indices/"+this->archivo+".txt"); |
|  | csv.seekg(n,ios::beg); |
|  | csv.getline(linea,10000,','); |
|  | while(ID==linea) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | csv.getline(seg,1000); |
|  | n=atoi(linea); |
|  | s=atoi(seg); |
|  | ifstream is("google/"+this->archivo+"/part-"+numero(s)+"-of-"+numero(this->cantidad)+".csv",std::fstream::binary); |
|  | is.seekg(n,ios::beg); |
|  | is.getline(linea,1000); |
|  | if(linea[0]==' ') |
|  | cout<<"Esta linea de registro fue eliminada"<<endl; |
|  | else |
|  | cout<<linea<<endl; |
|  | csv.getline(linea,1000,','); |
|  | } |
|  | return "Seleccion hecha correctamente :3"; |
|  | } |
|  | else |
|  | { |
|  | return ("NO SE ENCONTRO INDICE A BUSCAR!!!!GG"); |
|  | } |
|  |  |
|  | } |
|  | } |
|  | char \*task\_events::selectTE(string ID) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | int divi=0; |
|  | char \*a=new char[ID.length()+1]; |
|  | strcpy(a, ID.c\_str()); |
|  | long long int IDn=atoll(a),campo1,campo2=0,campo3; |
|  | char\* ids=new char[1000]; |
|  | bool encontrado=false; |
|  | for(int i=0;i<5;i++) |
|  | { |
|  | ifstream csv("indices/"+this->archivo+"/ClusterKey-"+this->archivo+"-"+numero(i)+".txt",std::fstream::binary); |
|  | if (!csv) |
|  | { |
|  | return "No se encuentra la tabla a buscar"; |
|  | } |
|  | else |
|  | { |
|  | while(!csv.eof()) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | campo1=atoll(linea); |
|  | if(IDn>campo1) |
|  | { |
|  | csv.getline(linea,1000); |
|  | campo2=atoll(linea); |
|  | } |
|  | else |
|  | { |
|  | encontrado=true; |
|  | csv.close(); |
|  | break; |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | ifstream is("indices/"+this->archivo+"/"+this->archivo+"-"+numero(i)+".txt",std::fstream::binary); |
|  | is.seekg(campo2,ios::beg); |
|  | while(!is.eof()) |
|  | { |
|  | is.getline(linea,1000,','); |
|  | if(ID==linea) |
|  | { |
|  |  |
|  | while(ID==linea) |
|  | { |
|  | is.getline(linea,1000,','); |
|  | campo2=atoll(linea); |
|  | is.getline(linea,1000); |
|  | campo3=atoll(linea); |
|  | ifstream iss("google/"+this->archivo+"/part-"+numero(campo3)+"-of-"+numero(this->cantidad)+".csv",std::fstream::binary); |
|  | iss.seekg(campo2,ios::beg); |
|  | iss.getline(linea,1000); |
|  | if(linea[0]==' ') |
|  | cout<<"Esta linea de registro fue eliminada"<<endl; |
|  | else |
|  | cout<<linea<<endl; |
|  | iss.close(); |
|  | is.getline(linea,1000,','); |
|  | } |
|  | return "Seleccion hecha exitosamente"; |
|  | } |
|  | else |
|  | { |
|  | is.getline(linea,1000); |
|  | } |
|  | } |
|  | return "No se encontro los indices"; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | char\* task\_usage::selectTU(string ID) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | int divi=0; |
|  | char \*a=new char[ID.length()+1]; |
|  | strcpy(a, ID.c\_str()); |
|  | long long int IDn=atoll(a),campo1,campo2=0,campo3; |
|  | char\* ids=new char[1000]; |
|  | bool encontrado=false; |
|  | for(int i=0;i<37;i++) |
|  | { |
|  | ifstream csv("indices/"+this->archivo+"/ClusterKey-"+this->archivo+"-"+numero(i)+".txt",std::fstream::binary); |
|  | if (!csv) |
|  | { |
|  | return "No se encuentra la tabla a buscar"; |
|  | } |
|  | else |
|  | { |
|  | while(!csv.eof()) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | campo1=atoll(linea); |
|  | if(IDn>campo1) |
|  | { |
|  | csv.getline(linea,1000); |
|  | campo2=atoll(linea); |
|  | } |
|  | else |
|  | { |
|  | encontrado=true; |
|  | csv.close(); |
|  | break; |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | ifstream is("indices/"+this->archivo+"/"+this->archivo+"-"+numero(i)+".txt",std::fstream::binary); |
|  | is.seekg(campo2,ios::beg); |
|  | while(!is.eof()) |
|  | { |
|  | is.getline(linea,1000,','); |
|  | if(ID==linea) |
|  | { |
|  |  |
|  | while(ID==linea) |
|  | { |
|  | is.getline(linea,1000,','); |
|  | campo2=atoll(linea); |
|  | is.getline(linea,1000); |
|  | campo3=atoll(linea); |
|  | ifstream iss("google/"+this->archivo+"/part-"+numero(campo3)+"-of-"+numero(this->cantidad)+".csv",std::fstream::binary); |
|  | iss.seekg(campo2,ios::beg); |
|  | iss.getline(linea,1000); |
|  | if(linea[0]==' ') |
|  | cout<<"Esta linea de registro fue eliminada"<<endl; |
|  | else |
|  | cout<<linea<<endl; |
|  | iss.close(); |
|  | is.getline(linea,1000,','); |
|  | } |
|  | return "Seleccion hecha exitosamente"; |
|  | } |
|  | else |
|  | { |
|  | is.getline(linea,1000); |
|  | } |
|  | } |
|  | return "No se encontro los indices"; |
|  | } |
|  | } |
|  | } |
|  | } |

**Delete.h**

|  |
| --- |
|  |
|  |  |
|  | #include "Bloqueos.h"  string machine\_events::Delete(string ID) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | string ids,url; |
|  | bool encontrado=false; |
|  | ifstream HashKey("indices/HashKey-"+this->archivo+".txt"); |
|  | if (!HashKey) |
|  | { |
|  |  |
|  | return ("No se encuentra la tabla a buscar"); |
|  | } |
|  | else |
|  | { |
|  | while(!HashKey.eof()) |
|  | { |
|  | HashKey.getline(linea,10000,','); |
|  | ids=linea; |
|  | if(ID==ids) |
|  | { |
|  | HashKey.getline(linea,10000); |
|  | encontrado=true; |
|  | break; |
|  | } |
|  | else |
|  | HashKey.getline(linea,10000); |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | long long int n=atoi(linea),s; |
|  | url="indices/"+this->archivo+".txt"; |
|  | ifstream csv(url); |
|  | csv.seekg(n,ios::beg); |
|  | csv.getline(linea,10000,','); |
|  | if(ID==linea) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | csv.getline(seg,1000); |
|  | n=atoll(linea); |
|  | s=atoi(seg); |
|  | fstream is("google/"+this->archivo+"/part-"+numero(s)+"-of-"+numero(this->cantidad)+".csv",ios::in|ios::out); |
|  | if(is!=NULL) |
|  | { |
|  | is.seekg(n,ios::beg); |
|  | is.getline(linea,1000); |
|  | int t=strlen(linea); |
|  | is.seekg(n,ios::beg); |
|  | for(int i=0;i<t;i++) |
|  | { |
|  | is<<' '; |
|  | } |
|  | return "Linea eliminada"; |
|  | } |
|  | else |
|  | { |
|  | return "no se pudo realizar la operacion"; |
|  | } |
|  | } |
|  | } |
|  | else |
|  | { |
|  | return "Indice no encontrado en la tabla"; |
|  | } |
|  | } |
|  | string machine\_attributes::Delete(string ID) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | string ids,url; |
|  | bool encontrado=false; |
|  | ifstream HashKey("indices/HashKey-"+this->archivo+".txt"); |
|  | if (!HashKey) |
|  | { |
|  |  |
|  | return ("No se encuentra la tabla a buscar"); |
|  | } |
|  | else |
|  | { |
|  | while(!HashKey.eof()) |
|  | { |
|  | HashKey.getline(linea,10000,','); |
|  | ids=linea; |
|  | if(ID==ids) |
|  | { |
|  | HashKey.getline(linea,10000); |
|  | encontrado=true; |
|  | break; |
|  | } |
|  | else |
|  | HashKey.getline(linea,10000); |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | long long int n=atoi(linea),s; |
|  | url="indices/"+this->archivo+".txt"; |
|  | ifstream csv(url); |
|  | csv.seekg(n,ios::beg); |
|  | csv.getline(linea,10000,','); |
|  | if(ID==linea) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | csv.getline(seg,1000); |
|  | n=atoll(linea); |
|  | s=atoi(seg); |
|  | fstream is("google/"+this->archivo+"/part-"+numero(s)+"-of-"+numero(this->cantidad)+".csv",ios::in|ios::out); |
|  | if(is!=NULL) |
|  | { |
|  | is.seekg(n,ios::beg); |
|  | is.getline(linea,1000); |
|  | int t=strlen(linea); |
|  | is.seekg(n,ios::beg); |
|  | for(int i=0;i<t;i++) |
|  | { |
|  | is<<' '; |
|  | } |
|  | return "Linea eliminada"; |
|  | } |
|  | else |
|  | { |
|  | return "no se pudo realizar la operacion"; |
|  | } |
|  | } |
|  | } |
|  | else |
|  | { |
|  | return "Indice no encontrado en la tabla"; |
|  | } |
|  | } |
|  | string job\_events::Delete(string ID) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | string ids,url; |
|  | bool encontrado=false; |
|  | ifstream HashKey("indices/HashKey-"+this->archivo+".txt"); |
|  | if (!HashKey) |
|  | { |
|  |  |
|  | return ("No se encuentra la tabla a buscar"); |
|  | } |
|  | else |
|  | { |
|  | while(!HashKey.eof()) |
|  | { |
|  | HashKey.getline(linea,10000,','); |
|  | ids=linea; |
|  | if(ID==ids) |
|  | { |
|  | HashKey.getline(linea,10000); |
|  | encontrado=true; |
|  | break; |
|  | } |
|  | else |
|  | HashKey.getline(linea,10000); |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | long long int n=atoi(linea),s; |
|  | url="indices/"+this->archivo+".txt"; |
|  | ifstream csv(url); |
|  | csv.seekg(n,ios::beg); |
|  | csv.getline(linea,10000,','); |
|  | if(ID==linea) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | csv.getline(seg,1000); |
|  | n=atoll(linea); |
|  | s=atoi(seg); |
|  | fstream is("google/"+this->archivo+"/part-"+numero(s)+"-of-"+numero(this->cantidad)+".csv",ios::in|ios::out); |
|  | if(is!=NULL) |
|  | { |
|  | is.seekg(n,ios::beg); |
|  | is.getline(linea,1000); |
|  | int t=strlen(linea); |
|  | is.seekg(n,ios::beg); |
|  | for(int i=0;i<t;i++) |
|  | { |
|  | is<<' '; |
|  | } |
|  | return "Linea eliminada"; |
|  | } |
|  | else |
|  | { |
|  | return "no se pudo realizar la operacion"; |
|  | } |
|  | } |
|  | } |
|  | else |
|  | { |
|  | return "Indice no encontrado en la tabla"; |
|  | } |
|  | } |
|  | string task\_constraints::Delete(string ID) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | string ids,url; |
|  | bool encontrado=false; |
|  | ifstream HashKey("indices/HashKey-"+this->archivo+".txt"); |
|  | if (!HashKey) |
|  | { |
|  |  |
|  | return ("No se encuentra la tabla a buscar"); |
|  | } |
|  | else |
|  | { |
|  | while(!HashKey.eof()) |
|  | { |
|  | HashKey.getline(linea,10000,','); |
|  | ids=linea; |
|  | if(ID==ids) |
|  | { |
|  | HashKey.getline(linea,10000); |
|  | encontrado=true; |
|  | break; |
|  | } |
|  | else |
|  | HashKey.getline(linea,10000); |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | long long int n=atoi(linea),s; |
|  | url="indices/"+this->archivo+".txt"; |
|  | ifstream csv(url); |
|  | csv.seekg(n,ios::beg); |
|  | csv.getline(linea,10000,','); |
|  | if(ID==linea) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | csv.getline(seg,1000); |
|  | n=atoll(linea); |
|  | s=atoi(seg); |
|  | fstream is("google/"+this->archivo+"/part-"+numero(s)+"-of-"+numero(this->cantidad)+".csv",ios::in|ios::out); |
|  | if(is!=NULL) |
|  | { |
|  | is.seekg(n,ios::beg); |
|  | is.getline(linea,1000); |
|  | int t=strlen(linea); |
|  | is.seekg(n,ios::beg); |
|  | for(int i=0;i<t;i++) |
|  | { |
|  | is<<' '; |
|  | } |
|  | return "Linea eliminada"; |
|  | } |
|  | else |
|  | { |
|  | return "no se pudo realizar la operacion"; |
|  | } |
|  | } |
|  | } |
|  | else |
|  | { |
|  | return "Indice no encontrado en la tabla"; |
|  | } |
|  | } |
|  | string task\_events::Delete(string ID) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | int divi=0; |
|  | char \*a=new char[ID.length()+1]; |
|  | strcpy(a, ID.c\_str()); |
|  | long long int IDn=atoll(a),campo1,campo2=0,campo3; |
|  | char\* ids=new char[1000]; |
|  | bool encontrado=false; |
|  | for(int i=0;i<5;i++) |
|  | { |
|  | ifstream csv("indices/"+this->archivo+"/ClusterKey-"+this->archivo+"-"+numero(i)+".txt",std::fstream::binary); |
|  | if (!csv) |
|  | { |
|  | return "No se encuentra la tabla a buscar"; |
|  | } |
|  | else |
|  | { |
|  | while(!csv.eof()) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | campo1=atoll(linea); |
|  | if(IDn>campo1) |
|  | { |
|  | csv.getline(linea,1000); |
|  | campo2=atoll(linea); |
|  | } |
|  | else |
|  | { |
|  | encontrado=true; |
|  | csv.close(); |
|  | break; |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | ifstream is("indices/"+this->archivo+"/"+this->archivo+"-"+numero(i)+".txt",std::fstream::binary); |
|  | is.seekg(campo2,ios::beg); |
|  | while(!is.eof()) |
|  | { |
|  | is.getline(linea,1000,','); |
|  | if(ID==linea) |
|  | { |
|  | is.getline(linea,1000,','); |
|  | campo2=atoll(linea); |
|  | is.getline(linea,1000); |
|  | campo3=atoll(linea); |
|  | fstream iss("google/"+this->archivo+"/part-"+numero(campo3)+"-of-"+numero(this->cantidad)+".csv",ios::in|ios::out); |
|  | iss.seekg(campo2,ios::beg); |
|  | iss.getline(linea,1000); |
|  | int t=strlen(linea); |
|  | iss.seekg(campo2,ios::beg); |
|  | for(int i=0;i<t;i++) |
|  | { |
|  | iss<<' '; |
|  | } |
|  | return "Linea eliminada"; |
|  | } |
|  | else |
|  | { |
|  | is.getline(linea,1000); |
|  | } |
|  | } |
|  | return "No se encontro los indices"; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | string task\_usage::Delete(string ID) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | int divi=0; |
|  | char \*a=new char[ID.length()+1]; |
|  | strcpy(a, ID.c\_str()); |
|  | long long int IDn=atoll(a),campo1,campo2=0,campo3; |
|  | char\* ids=new char[1000]; |
|  | bool encontrado=false; |
|  | for(int i=0;i<5;i++) |
|  | { |
|  | ifstream csv("indices/"+this->archivo+"/ClusterKey-"+this->archivo+"-"+numero(i)+".txt",std::fstream::binary); |
|  | if (!csv) |
|  | { |
|  | return "No se encuentra la tabla a buscar"; |
|  | } |
|  | else |
|  | { |
|  | while(!csv.eof()) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | campo1=atoll(linea); |
|  | if(IDn>campo1) |
|  | { |
|  | csv.getline(linea,1000); |
|  | campo2=atoll(linea); |
|  | } |
|  | else |
|  | { |
|  | encontrado=true; |
|  | csv.close(); |
|  | break; |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | ifstream is("indices/"+this->archivo+"/"+this->archivo+"-"+numero(i)+".txt",std::fstream::binary); |
|  | is.seekg(campo2,ios::beg); |
|  | while(!is.eof()) |
|  | { |
|  | is.getline(linea,1000,','); |
|  | if(ID==linea) |
|  | { |
|  | is.getline(linea,1000,','); |
|  | campo2=atoll(linea); |
|  | is.getline(linea,1000); |
|  | campo3=atoll(linea); |
|  | fstream iss("google/"+this->archivo+"/part-"+numero(campo3)+"-of-"+numero(this->cantidad)+".csv",ios::in|ios::out); |
|  | iss.seekg(campo2,ios::beg); |
|  | iss.getline(linea,1000); |
|  | int t=strlen(linea); |
|  | iss.seekg(campo2,ios::beg); |
|  | for(int i=0;i<t;i++) |
|  | { |
|  | iss<<' '; |
|  | } |
|  | return "Linea eliminada"; |
|  | } |
|  | else |
|  | { |
|  | is.getline(linea,1000); |
|  | } |
|  | } |
|  | return "No se encontro los indices"; |
|  | } |
|  | } |
|  | } |
|  | } |

**Update.h**

|  |
| --- |
|  |
|  |  |
|  | #include "Select.h"  string job\_events::update(string ID,string NewLine) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | char\* aux=new char[1000]; |
|  | long long int divi=0,variacion,posicionI; |
|  | string ids,Newline,url; |
|  | bool encontrado=false; |
|  |  |
|  | ifstream HashKey("indices/HashKey-"+this->archivo+".txt"); |
|  | if (!HashKey) |
|  | { |
|  |  |
|  | return ("No se encuentra la tabla a buscar"); |
|  | } |
|  | else |
|  | { |
|  | while(!HashKey.eof()) |
|  | { |
|  | HashKey.getline(linea,10000,','); |
|  | ids=linea; |
|  | if(ID==ids) |
|  | { |
|  | HashKey.getline(linea,10000); |
|  | encontrado=true; |
|  | break; |
|  | } |
|  | else |
|  | HashKey.getline(linea,10000); |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | long long int n=atoi(linea),s; |
|  | url="indices/"+this->archivo+".txt"; |
|  | ifstream csv(url); |
|  | csv.seekg(n,ios::beg); |
|  | posicionI=n; |
|  | csv.getline(linea,10000,','); |
|  | if(ID==linea) |
|  | { |
|  |  |
|  | csv.getline(linea,1000,','); |
|  | csv.getline(seg,1000); |
|  | n=atoll(linea); |
|  | s=atoi(seg); |
|  | fstream is("google/"+this->archivo+"/part-"+numero(s)+"-of-"+numero(this->cantidad)+".csv",ios::in|ios::out); |
|  | if(is!=NULL) |
|  | { |
|  |  |
|  | is.seekg(n,ios::beg); |
|  | is.getline(linea,1000); |
|  | char\* p1 = new char[Newline.length() + 1]; |
|  | strcpy(p1,Newline.c\_str()); |
|  | int t=strlen(linea),tn=strlen(p1),tam=t-tn; |
|  | is.seekg(n,ios::beg); |
|  | for(int i=0;i<tn;i++) |
|  | { |
|  | is<<p1[i]; |
|  | } |
|  | for(int i=0;i<tam;i++) |
|  | { |
|  | is<<' '; |
|  | } |
|  | return "Linea modificada"; |
|  | } |
|  | else |
|  | { |
|  | return "no se pudo realizar la operacion"; |
|  | } |
|  | } |
|  | } |
|  | else |
|  | { |
|  | return "Indice no encontrado en la tabla"; |
|  | } |
|  | } |
|  | string machine\_attributes::update(string ID,string NewLine) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | char\* aux=new char[1000]; |
|  | vector<string> auxV; |
|  | long long int divi=0,variacion,posicionI; |
|  | string ids,Newline,url; |
|  | bool encontrado=false; |
|  |  |
|  | ifstream HashKey("indices/HashKey-"+this->archivo+".txt"); |
|  | if (!HashKey) |
|  | { |
|  |  |
|  | return ("No se encuentra la tabla a buscar"); |
|  | } |
|  | else |
|  | { |
|  | while(!HashKey.eof()) |
|  | { |
|  | HashKey.getline(linea,10000,','); |
|  | ids=linea; |
|  | if(ID==ids) |
|  | { |
|  | HashKey.getline(linea,10000); |
|  | encontrado=true; |
|  | break; |
|  | } |
|  | else |
|  | HashKey.getline(linea,10000); |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | long long int n=atoi(linea),s; |
|  | url="indices/"+this->archivo+".txt"; |
|  | ifstream csv(url); |
|  | csv.seekg(n,ios::beg); |
|  | posicionI=n; |
|  | csv.getline(linea,10000,','); |
|  | if(ID==linea) |
|  | { |
|  |  |
|  | csv.getline(linea,1000,','); |
|  | csv.getline(seg,1000); |
|  | n=atoll(linea); |
|  | s=atoi(seg); |
|  | fstream is("google/"+this->archivo+"/part-"+numero(s)+"-of-"+numero(this->cantidad)+".csv",ios::in|ios::out); |
|  | if(is!=NULL) |
|  | { |
|  |  |
|  | is.seekg(n,ios::beg); |
|  | is.getline(linea,1000); |
|  | char\* p1 = new char[Newline.length() + 1]; |
|  | strcpy(p1,Newline.c\_str()); |
|  | int t=strlen(linea),tn=strlen(p1),tam=t-tn; |
|  | is.seekg(n,ios::beg); |
|  | for(int i=0;i<tn;i++) |
|  | { |
|  | is<<p1[i]; |
|  | } |
|  | for(int i=0;i<tam;i++) |
|  | { |
|  | is<<' '; |
|  | } |
|  | return "Linea modificada"; |
|  | } |
|  | else |
|  | { |
|  | return "no se pudo realizar la operacion"; |
|  | } |
|  | } |
|  | } |
|  | else |
|  | { |
|  | return "Indice no encontrado en la tabla"; |
|  | } |
|  | } |
|  | string machine\_events::update(string ID,string NewLine) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | char\* aux=new char[1000]; |
|  | vector<string> auxV; |
|  | long long int divi=0,variacion,posicionI; |
|  | string ids,Newline,url; |
|  | bool encontrado=false; |
|  |  |
|  | ifstream HashKey("indices/HashKey-"+this->archivo+".txt"); |
|  | if (!HashKey) |
|  | { |
|  |  |
|  | return ("No se encuentra la tabla a buscar"); |
|  | } |
|  | else |
|  | { |
|  | while(!HashKey.eof()) |
|  | { |
|  | HashKey.getline(linea,10000,','); |
|  | ids=linea; |
|  | if(ID==ids) |
|  | { |
|  | HashKey.getline(linea,10000); |
|  | encontrado=true; |
|  | break; |
|  | } |
|  | else |
|  | HashKey.getline(linea,10000); |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | long long int n=atoi(linea),s; |
|  | url="indices/"+this->archivo+".txt"; |
|  | ifstream csv(url); |
|  | csv.seekg(n,ios::beg); |
|  | posicionI=n; |
|  | csv.getline(linea,10000,','); |
|  | if(ID==linea) |
|  | { |
|  |  |
|  | csv.getline(linea,1000,','); |
|  | csv.getline(seg,1000); |
|  | n=atoll(linea); |
|  | s=atoi(seg); |
|  | fstream is("google/"+this->archivo+"/part-"+numero(s)+"-of-"+numero(this->cantidad)+".csv",ios::in|ios::out); |
|  | if(is!=NULL) |
|  | { |
|  |  |
|  | is.seekg(n,ios::beg); |
|  | is.getline(linea,1000); |
|  | char\* p1 = new char[Newline.length() + 1]; |
|  | strcpy(p1,Newline.c\_str()); |
|  | int t=strlen(linea),tn=strlen(p1),tam=t-tn; |
|  | is.seekg(n,ios::beg); |
|  | for(int i=0;i<tn;i++) |
|  | { |
|  | is<<p1[i]; |
|  | } |
|  | for(int i=0;i<tam;i++) |
|  | { |
|  | is<<' '; |
|  | } |
|  | return "Linea modificada"; |
|  | } |
|  | else |
|  | { |
|  | return "no se pudo realizar la operacion"; |
|  | } |
|  | } |
|  | } |
|  | else |
|  | { |
|  | return "Indice no encontrado en la tabla"; |
|  | } |
|  | } |
|  | string task\_constraints::update(string ID,string NewLine) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | char\* aux=new char[1000]; |
|  | long long int posicionI; |
|  | string ids,Newline,url; |
|  | bool encontrado=false; |
|  |  |
|  | ifstream HashKey("indices/HashKey-"+this->archivo+".txt"); |
|  | if (!HashKey) |
|  | { |
|  |  |
|  | return ("No se encuentra la tabla a buscar"); |
|  | } |
|  | else |
|  | { |
|  | while(!HashKey.eof()) |
|  | { |
|  | HashKey.getline(linea,10000,','); |
|  | ids=linea; |
|  | if(ID==ids) |
|  | { |
|  | HashKey.getline(linea,10000); |
|  | encontrado=true; |
|  | break; |
|  | } |
|  | else |
|  | HashKey.getline(linea,10000); |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | long long int n=atoll(linea),s; |
|  | url="indices/"+this->archivo+".txt"; |
|  | ifstream csv(url); |
|  | csv.seekg(n,ios::beg); |
|  | posicionI=n; |
|  | csv.getline(linea,10000,','); |
|  | if(ID==linea) |
|  | { |
|  |  |
|  | csv.getline(linea,1000,','); |
|  | csv.getline(seg,1000); |
|  | n=atoll(linea); |
|  | s=atoi(seg); |
|  | fstream is("google/"+this->archivo+"/part-"+numero(s)+"-of-"+numero(this->cantidad)+".csv",ios::in|ios::out); |
|  | if(is!=NULL) |
|  | { |
|  |  |
|  | is.seekg(n,ios::beg); |
|  | is.getline(linea,1000); |
|  | char\* p1 = new char[Newline.length() + 1]; |
|  | strcpy(p1,Newline.c\_str()); |
|  | int t=strlen(linea),tn=strlen(p1),tam=t-tn; |
|  | is.seekg(n,ios::beg); |
|  | for(int i=0;i<tn;i++) |
|  | { |
|  | is<<p1[i]; |
|  | } |
|  | for(int i=0;i<tam;i++) |
|  | { |
|  | is<<' '; |
|  | } |
|  | return "Linea modificada"; |
|  | } |
|  | else |
|  | { |
|  | return "no se pudo realizar la operacion"; |
|  | } |
|  | } |
|  | } |
|  | else |
|  | { |
|  | return "Indice no encontrado en la tabla"; |
|  | } |
|  | } |
|  | string task\_events::update(string ID,string NewLine) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | char\* aux=new char[1000]; |
|  | vector<string> auxV; |
|  | char \*a=new char[ID.length()+1]; |
|  | strcpy(a, ID.c\_str()); |
|  | long long int divi=0,variacion,posicionI,IDn=atoll(a),campo1,campo2=0,campo3; |
|  | string ids,Newline,url; |
|  | bool encontrado=false; |
|  |  |
|  | for(int i=0;i<5;i++) |
|  | { |
|  | ifstream csv("indices/"+this->archivo+"/ClusterKey-"+this->archivo+"-"+numero(i)+".txt",std::fstream::binary); |
|  | if (!csv) |
|  | { |
|  | return "No se encuentra la tabla a buscar"; |
|  | } |
|  | else |
|  | { |
|  | while(!csv.eof()) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | campo1=atoll(linea); |
|  | if(IDn>campo1) |
|  | { |
|  | csv.getline(linea,1000); |
|  | campo2=atoll(linea); |
|  | } |
|  | else |
|  | { |
|  | encontrado=true; |
|  | csv.close(); |
|  | break; |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | ifstream is("indices/"+this->archivo+"/"+this->archivo+"-"+numero(i)+".txt",std::fstream::binary); |
|  | is.seekg(campo2,ios::beg); |
|  | while(!is.eof()) |
|  | { |
|  | is.getline(linea,1000,','); |
|  | campo1=atoll(linea); |
|  | if(IDn==campo1) |
|  | { |
|  | is.getline(linea,1000,','); |
|  | campo2=atoll(linea); |
|  | is.getline(linea,1000); |
|  | campo3=atoll(linea); |
|  | fstream iss("google/"+this->archivo+"/part-"+numero(campo3)+"-of-"+numero(this->cantidad)+".csv",ios::in|ios::out); |
|  | iss.seekg(campo2,ios::beg); |
|  | iss.getline(linea,1000); |
|  | char\* p1 = new char[Newline.length() + 1]; |
|  | strcpy(p1,Newline.c\_str()); |
|  | int t=strlen(linea),tn=strlen(p1),tam=t-tn; |
|  | iss.seekg(campo2,ios::beg); |
|  | for(int i=0;i<tn;i++) |
|  | { |
|  | iss<<p1[i]; |
|  | } |
|  | for(int i=0;i<tam;i++) |
|  | { |
|  | iss<<' '; |
|  | } |
|  | return "Linea modificada"; |
|  | } |
|  | else if(IDn<campo1) |
|  | { |
|  | break; |
|  | } |
|  | else |
|  | { |
|  | is.getline(linea,1000); |
|  | } |
|  | } |
|  | return "No se encontro los indices"; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | string task\_usage::update(string ID,string NewLine) |
|  | { |
|  | char \*linea=new char[10000]; |
|  | char \*seg=new char[10000]; |
|  | char\* aux=new char[1000]; |
|  | vector<string> auxV; |
|  | char \*a=new char[ID.length()+1]; |
|  | strcpy(a, ID.c\_str()); |
|  | long long int divi=0,variacion,posicionI,IDn=atoll(a),campo1,campo2=0,campo3; |
|  | string ids,Newline,url; |
|  | bool encontrado=false; |
|  |  |
|  | for(int i=0;i<5;i++) |
|  | { |
|  | ifstream csv("indices/"+this->archivo+"/ClusterKey-"+this->archivo+"-"+numero(i)+".txt",std::fstream::binary); |
|  | if (!csv) |
|  | { |
|  | return "No se encuentra la tabla a buscar"; |
|  | } |
|  | else |
|  | { |
|  | while(!csv.eof()) |
|  | { |
|  | csv.getline(linea,1000,','); |
|  | campo1=atoll(linea); |
|  | if(IDn>campo1) |
|  | { |
|  | csv.getline(linea,1000); |
|  | campo2=atoll(linea); |
|  | } |
|  | else |
|  | { |
|  | encontrado=true; |
|  | csv.close(); |
|  | break; |
|  | } |
|  | } |
|  | if(encontrado) |
|  | { |
|  | ifstream is("indices/"+this->archivo+"/"+this->archivo+"-"+numero(i)+".txt",std::fstream::binary); |
|  | is.seekg(campo2,ios::beg); |
|  | while(!is.eof()) |
|  | { |
|  | is.getline(linea,1000,','); |
|  | campo1=atoll(linea); |
|  | if(IDn==campo1) |
|  | { |
|  | is.getline(linea,1000,','); |
|  | campo2=atoll(linea); |
|  | is.getline(linea,1000); |
|  | campo3=atoll(linea); |
|  | fstream iss("google/"+this->archivo+"/part-"+numero(campo3)+"-of-"+numero(this->cantidad)+".csv",ios::in|ios::out); |
|  | iss.seekg(campo2,ios::beg); |
|  | iss.getline(linea,1000); |
|  | char\* p1 = new char[Newline.length() + 1]; |
|  | strcpy(p1,Newline.c\_str()); |
|  | int t=strlen(linea),tn=strlen(p1),tam=t-tn; |
|  | iss.seekg(campo2,ios::beg); |
|  | for(int i=0;i<tn;i++) |
|  | { |
|  | iss<<p1[i]; |
|  | } |
|  | for(int i=0;i<tam;i++) |
|  | { |
|  | iss<<' '; |
|  | } |
|  | return "Linea modificada"; |
|  | } |
|  | else if(IDn<campo1) |
|  | { |
|  | break; |
|  | } |
|  | else |
|  | { |
|  | is.getline(linea,1000); |
|  | } |
|  | } |
|  | return "No se encontro los indices"; |
|  | } |
|  | } |
|  | } |
|  | } |

**Newline.h**

|  |
| --- |
|  |
|  |  |
|  | #include "Tablas.h"  string machine\_attributes::NewLine(string ID) |
|  | { |
|  | char\* aux=new char[1000]; |
|  | cout<<"Introducir el nuevo valor para el campo TimeSpamp"<<endl;cin>>aux; |
|  | if(!esNumero(aux))return "Error"; |
|  | string str(aux); |
|  | cout<<"Introducir el nuevo valor para el campo Attribute\_Name"<<endl;cin>>this->attribute\_name; |
|  | cout<<"Introducir el nuevo valor para el campo Attribute\_Value"<<endl;cin>>this->attribute\_value; |
|  | cout<<"Introducir el nuevo valor para el campo Attribute\_Deleted"<<endl;cin>>aux; |
|  | if(!esNumero(aux)) return "Error"; |
|  | string str2(aux); |
|  | string Newline=str+','+','+ID+','+this->attribute\_name+','+this->attribute\_value+','+str2; |
|  | return Newline; |
|  | } |
|  | string machine\_events::NewLine(string ID) |
|  | { |
|  | char\* aux=new char[1000]; |
|  | cout<<"Introducir el nuevo valor para el campo TimeSpamp"<<endl;cin>>aux; |
|  | if(!esNumero(aux))return "Error"; |
|  | cout<<"Introducir el nuevo valor para el campo Event\_Type"<<endl;cin>>this->event\_type; |
|  | cout<<"Introducir el nuevo valor para el campo Platform\_ID"<<endl;cin>>this->platform\_ID; |
|  | cout<<"Introducir el nuevo valor para el campo Capacity\_CPU"<<endl;cin>>this->capacity\_CPU; |
|  | cout<<"Introducir el nuevo valor para el campo Capacity\_Memory"<<endl;cin>>this->capacity\_memory; |
|  | string str(aux); |
|  | string Newline=str+','+','+ID+','+this->event\_type+','+this->platform\_ID+','+capacity\_CPU+','+capacity\_memory; |
|  | return Newline; |
|  | } |
|  | string job\_events::NewLine(string ID) |
|  | { |
|  | char\* aux=new char[1000]; |
|  | cout<<"Introducir el nuevo valor para el campo TimeSpamp"<<endl;cin>>aux; |
|  | if(!esNumero(aux))return "Error"; |
|  | cout<<"Introducir el nuevo valor para el campo Missing\_Info"<<endl;cin>>this->missing\_info; |
|  | cout<<"Introducir el nuevo valor para el campo Event\_Type"<<endl;cin>>this->event\_type; |
|  | cout<<"Introducir el nuevo valor para el campo User\_Name"<<endl;cin>>this->user\_name; |
|  | cout<<"Introducir el nuevo valor para el campo Scheduling\_Class"<<endl;cin>>this->scheduling\_class; |
|  | cout<<"Introducir el nuevo valor para el campo Logical\_Job\_Name"<<endl;cin>>this->logical\_job\_name; |
|  | string str(aux); |
|  | string Newline=str+','+','+ID+','+this->missing\_info+','+this->event\_type+','+this->user\_name+','+this->scheduling\_class+','+this->logical\_job\_name; |
|  | return Newline; |
|  | } |
|  | string task\_constraints::NewLine(string ID) |
|  | { |
|  | char\* aux=new char[1000]; |
|  | cout<<"Introducir el nuevo valor para el campo TimeSpamp"<<endl;cin>>aux; |
|  | if(!esNumero(aux))return "Error"; |
|  | string str(aux); |
|  | cout<<"Introducir el nuevo valor para el campo task\_index"<<endl;cin>>aux; |
|  | if(!esNumero(aux)) return "Error"; |
|  | string str2(aux); |
|  | cout<<"Introducir el nuevo valor para el campo Attribute\_Name"<<endl;cin>>this->attribute\_name; |
|  | cout<<"Introducir el nuevo valor para el campo Comparison\_Operator"<<endl;cin>>this->comparison\_operator; |
|  | cout<<"Introducir el nuevo valor para el campo Attribute\_Value"<<endl;cin>>this->attribute\_value; |
|  |  |
|  | string Newline=str+','+','+ID+','+str2+','+this->attribute\_name+','+this->comparison\_operator+','+attribute\_value; |
|  | return Newline; |
|  | } |
|  | string task\_events::NewLine(string ID) |
|  | { |
|  | char\* aux=new char[1000]; |
|  | cout<<"Introducir el nuevo valor para el campo TimeSpamp"<<endl;cin>>aux; |
|  | if(!esNumero(aux))return "Error"; |
|  | string str(aux); |
|  | cout<<"Introducir el nuevo valor para el campo Missing\_Info"<<endl;cin>>this->missing\_info; |
|  | cout<<"Introducir el nuevo valor para el campo Machine\_ID"<<endl;cin>>aux; |
|  | if(!esNumero(aux)) return "Error"; |
|  | string str2(aux); |
|  | cout<<"Introducir el nuevo valor para el campo Event\_Type"<<endl;cin>>this->event\_type; |
|  | cout<<"Introducir el nuevo valor para el campo user\_name"<<endl;cin>>this->user\_name; |
|  | cout<<"Introducir el nuevo valor para el campo scheduling\_class"<<endl;cin>>this->scheduling\_class; |
|  | cout<<"Introducir el nuevo valor para el campo priority"<<endl;cin>>aux; |
|  | if(!esNumero(aux)) return "Error"; |
|  | string str3(aux); |
|  | cout<<"Introducir el nuevo valor para el campo resource\_request\_for\_CPU\_cores"<<endl;cin>>this->resource\_request\_for\_CPU\_cores; |
|  | cout<<"Introducir el nuevo valor para el campo resource\_request\_for\_RAM"<<endl;cin>>this->resource\_request\_for\_RAM; |
|  | cout<<"Introducir el nuevo valor para el campo resource\_request\_for\_local\_disk\_space"<<endl;cin>>this->resource\_request\_for\_local\_disk\_space; |
|  | cout<<"Introducir el nuevo valor para el campo different\_machine\_constraint"<<endl;cin>>this->different\_machine\_constraint; |
|  |  |
|  | string Newline=str+','+this->missing\_info+','+ID+','+str2+','+this->event\_type+','+this->user\_name+','+this->scheduling\_class+','+str3+this->resource\_request\_for\_CPU\_cores+','+this->resource\_request\_for\_RAM+','+this->resource\_request\_for\_local\_disk\_space+','+this->different\_machine\_constraint; |
|  | return Newline; |
|  | } |
|  | string task\_usage::NewLine(string ID) |
|  | { |
|  | char\* aux=new char[1000]; |
|  | cout<<"Introducir el nuevo valor para el campo Start Time"<<endl;cin>>aux; |
|  | if(!esNumero(aux))return "Error"; |
|  | string str(aux); |
|  | cout<<"Introducir el nuevo valor para el campo End Time"<<endl;cin>>aux; |
|  | if(!esNumero(aux))return "Error"; |
|  | string str2(aux); |
|  | cout<<"Introducir el nuevo valor para el campo task\_index"<<endl;cin>>aux; |
|  | if(!esNumero(aux))return "Error"; |
|  | string str3(aux); |
|  | cout<<"Introducir el nuevo valor para el campo Machine\_ID"<<endl;cin>>this->machine\_ID; |
|  | cout<<"Introducir el nuevo valor para el campo mean\_CPU\_usage\_rate"<<endl;cin>>this->mean\_CPU\_usage\_rate; |
|  | cout<<"Introducir el nuevo valor para el campo canonical\_memory\_usage"<<endl;cin>>this->canonical\_memory\_usage; |
|  | cout<<"Introducir el nuevo valor para el campo assigned\_memory\_usage"<<endl;cin>>this->assigned\_memory\_usage; |
|  | cout<<"Introducir el nuevo valor para el campo unmapped\_page\_cache\_memory\_usage"<<endl;cin>>this->unmapped\_page\_cache\_memory\_usage; |
|  | cout<<"Introducir el nuevo valor para el campo total\_page\_cache\_memory\_usage"<<endl;cin>>this->total\_page\_cache\_memory\_usage; |
|  | cout<<"Introducir el nuevo valor para el campo maximun\_memory\_usage"<<endl;cin>>this->maximun\_memory\_usage; |
|  | cout<<"Introducir el nuevo valor para el campo mean\_disk\_IO\_time"<<endl;cin>>this->mean\_disk\_IO\_time; |
|  | cout<<"Introducir el nuevo valor para el campo mean\_local\_disk\_space\_used"<<endl;cin>>this->mean\_local\_disk\_space\_used; |
|  | cout<<"Introducir el nuevo valor para el campo maximun\_CPU\_usage"<<endl;cin>>this->maximun\_CPU\_usage; |
|  | cout<<"Introducir el nuevo valor para el campo maximun\_disk\_IO\_time"<<endl;cin>>this->maximun\_disk\_IO\_time; |
|  | cout<<"Introducir el nuevo valor para el campo cycles\_per\_instruction\_CPI"<<endl;cin>>this->cycles\_per\_instruction\_CPI; |
|  | cout<<"Introducir el nuevo valor para el campo memory\_accesses\_per\_instruction\_MAI"<<endl;cin>>this->memory\_accesses\_per\_instruction\_MAI; |
|  | cout<<"Introducir el nuevo valor para el campo sample\_portion"<<endl;cin>>this->sample\_portion; |
|  | cout<<"Introducir el nuevo valor para el campo aggregation\_type"<<endl;cin>>this->aggregation\_type; |
|  | cout<<"Introducir el nuevo valor para el campo sampled\_CPU\_usage"<<endl;cin>>this->sampled\_CPU\_usage; |
|  | string Newline=str+','+str2+','+ID+','+str3+','+this->machine\_ID+','+this->mean\_CPU\_usage\_rate+','+this->canonical\_memory\_usage+','+this->assigned\_memory\_usage+','+this->unmapped\_page\_cache\_memory\_usage+','+this->total\_page\_cache\_memory\_usage+','+this->maximun\_memory\_usage+','+this->mean\_disk\_IO\_time+','+this->mean\_local\_disk\_space\_used+','+this->maximun\_CPU\_usage+','+this->maximun\_disk\_IO\_time+','+this->cycles\_per\_instruction\_CPI+','+this->memory\_accesses\_per\_instruction\_MAI+','+this->sample\_portion+','+this->aggregation\_type+','+this->sampled\_CPU\_usage; |
|  | return Newline; |
|  | } |

**Insert.h**

|  |
| --- |
|  |
|  |  |
|  | #include "Delete.h"  string machine\_attributes::insertMA(string ID,string newline) |
|  | { |
|  | char\* linea=new char[1000]; |
|  | int tamNewline=newline.length(),tamNewIndice,tamNewHash; |
|  | char \*a=new char[ID.length()+1]; |
|  | strcpy(a, ID.c\_str()); |
|  | long long int Posicion,IDn=atol(a); |
|  | string newIndice,newHash,Result; |
|  | fstream BD; |
|  | BD.open("google/"+this->archivo+"/part-00000-of-"+numero(this->cantidad)+".csv",ios::in|ios::out|ios::app); |
|  | if(BD!=NULL) |
|  | { |
|  | BD<<newline<<endl; |
|  | Posicion=BD.tellg()-tamNewline-2; |
|  | fstream Indices; |
|  | Indices.open("indices/"+this->archivo+".txt",ios::in|ios::out|ios::app); |
|  | if(Indices!=NULL) |
|  | { |
|  |  |
|  | ostringstream convert; |
|  | convert << Posicion; |
|  | Result = convert.str(); |
|  | newIndice=ID+','+Result+','+'0'; |
|  | tamNewIndice=newIndice.length(); |
|  | Indices<<newIndice<<endl; |
|  | Posicion=Indices.tellg()-tamNewIndice-2; |
|  | ostringstream convert2; |
|  | convert2 << Posicion; |
|  | Result = convert2.str(); |
|  | newHash=ID+','+Result; |
|  | fstream HashIndices; |
|  | HashIndices.open("indices/HashKey-"+this->archivo+".txt",ios::in|ios::out|ios::app); |
|  | HashIndices<<newHash<<endl; |
|  | return "Insercion hecha Correctamente"; |
|  | } |
|  | Else |
|  | { |
|  | return "Error"; |
|  | } |
|  | } |
|  | Else |
|  | { |
|  | return "Error"; |
|  | } |
|  | } |
|  | string machine\_events::insertME(string ID,string newline) |
|  | { |
|  | char\* linea=new char[1000]; |
|  | int tamNewline=newline.length(),tamNewIndice,tamNewHash; |
|  | char \*a=new char[ID.length()+1]; |
|  | strcpy(a, ID.c\_str()); |
|  | long long int Posicion,IDn=atol(a); |
|  | string newIndice,newHash,Result; |
|  | fstream BD; |
|  | BD.open("google/"+this->archivo+"/part-00000-of-"+numero(this->cantidad)+".csv",ios::in|ios::out|ios::app); |
|  | if(BD!=NULL) |
|  | { |
|  | BD<<newline<<endl; |
|  | Posicion=BD.tellg()-tamNewline-2; |
|  | fstream Indices; |
|  | Indices.open("indices/"+this->archivo+".txt",ios::in|ios::out|ios::app); |
|  | if(Indices!=NULL) |
|  | { |
|  |  |
|  | ostringstream convert; |
|  | convert << Posicion; |
|  | Result = convert.str(); |
|  | newIndice=ID+','+Result+','+'0'; |
|  | tamNewIndice=newIndice.length(); |
|  | Indices<<newIndice<<endl; |
|  | Posicion=Indices.tellg()-tamNewIndice-2; |
|  | ostringstream convert2; |
|  | convert2 << Posicion; |
|  | Result = convert2.str(); |
|  | newHash=ID+','+Result; |
|  | fstream HashIndices; |
|  | HashIndices.open("indices/HashKey-"+this->archivo+".txt",ios::in|ios::out|ios::app); |
|  | HashIndices<<newHash<<endl; |
|  | return "Insercion hecha Correctamente"; |
|  | } |
|  | Else |
|  | { |
|  | return "Error"; |
|  | } |
|  | } |
|  | Else |
|  | { |
|  | return "Error"; |
|  | } |
|  | } |
|  | string job\_events::insertJE(string ID,string newline) |
|  | { |
|  | char\* linea=new char[1000]; |
|  | int tamNewline=newline.length(),tamNewIndice,tamNewHash; |
|  | char \*a=new char[ID.length()+1]; |
|  | strcpy(a, ID.c\_str()); |
|  | long long int Posicion,IDn=atol(a); |
|  | string newIndice,newHash,Result; |
|  | fstream BD; |
|  | BD.open("google/"+this->archivo+"/part-00000-of-"+numero(this->cantidad)+".csv",ios::in|ios::out|ios::app); |
|  | if(BD!=NULL) |
|  | { |
|  | BD<<newline<<endl; |
|  | Posicion=BD.tellg()-tamNewline-2; |
|  | fstream Indices; |
|  | Indices.open("indices/"+this->archivo+".txt",ios::in|ios::out|ios::app); |
|  | if(Indices!=NULL) |
|  | { |
|  |  |
|  | ostringstream convert; |
|  | convert << Posicion; |
|  | Result = convert.str(); |
|  | newIndice=ID+','+Result+','+'0'; |
|  | tamNewIndice=newIndice.length(); |
|  | Indices<<newIndice<<endl; |
|  | Posicion=Indices.tellg()-tamNewIndice-2; |
|  | ostringstream convert2; |
|  | convert2 << Posicion; |
|  | Result = convert2.str(); |
|  | newHash=ID+','+Result; |
|  | fstream HashIndices; |
|  | HashIndices.open("indices/HashKey-"+this->archivo+".txt",ios::in|ios::out|ios::app); |
|  | HashIndices<<newHash<<endl; |
|  | return "Insercion hecha Correctamente"; |
|  | } |
|  | else |
|  | { |
|  | return "Error"; |
|  | } |
|  | } |
|  | else |
|  | { |
|  | return "Error"; |
|  | } |
|  | } |
|  | string task\_constraints::insertTC(string ID,string newline) |
|  | { |
|  | char\* linea=new char[1000]; |
|  | int tamNewline=newline.length(),tamNewIndice,tamNewHash; |
|  | char \*a=new char[ID.length()+1]; |
|  | strcpy(a, ID.c\_str()); |
|  | long long int Posicion,IDn=atol(a); |
|  | string newIndice,newHash,Result; |
|  | fstream BD; |
|  | BD.open("google/"+this->archivo+"/part-00000-of-"+numero(this->cantidad)+".csv",ios::in|ios::out|ios::app); |
|  | if(BD!=NULL) |
|  | { |
|  | BD<<newline<<endl; |
|  | Posicion=BD.tellg()-tamNewline-2; |
|  | fstream Indices; |
|  | Indices.open("indices/"+this->archivo+".txt",ios::in|ios::out|ios::app); |
|  | if(Indices!=NULL) |
|  | { |
|  |  |
|  | ostringstream convert; |
|  | convert << Posicion; |
|  | Result = convert.str(); |
|  | newIndice=ID+','+Result+','+'0'; |
|  | tamNewIndice=newIndice.length(); |
|  | Indices<<newIndice<<endl; |
|  | Posicion=Indices.tellg()-tamNewIndice-2; |
|  | ostringstream convert2; |
|  | convert2 << Posicion; |
|  | Result = convert2.str(); |
|  | newHash=ID+','+Result; |
|  | fstream HashIndices; |
|  | HashIndices.open("indices/HashKey-"+this->archivo+".txt",ios::in|ios::out|ios::app); |
|  | HashIndices<<newHash<<endl; |
|  | return "Insercion hecha Correctamente"; |
|  | } |
|  | else |
|  | { |
|  | return "Error"; |
|  | } |
|  | } |
|  | else |
|  | { |
|  | return "Error"; |
|  | } |
|  | } |
|  | string task\_events::insertTE(string ID,string newline) |
|  | { |
|  | char\* linea=new char[1000]; |
|  | int tamNewline=newline.length(),tamNewIndice,tamnewCluster; |
|  | char \*a=new char[ID.length()+1]; |
|  | strcpy(a, ID.c\_str()); |
|  | long long int Posicion,IDn=atol(a); |
|  | string newIndice,newCluster,Result; |
|  | fstream BD; |
|  | BD.open("google/"+this->archivo+"/part-00000-of-"+numero(this->cantidad)+".csv",ios::in|ios::out|ios::app); |
|  | if(BD!=NULL) |
|  | { |
|  | BD<<newline<<endl; |
|  | Posicion=BD.tellg()-tamNewline-2; |
|  | fstream Indices; |
|  | Indices.open("indices/"+this->archivo+"/"+this->archivo+"-00000.txt",ios::in|ios::out|ios::app); |
|  | if(Indices!=NULL) |
|  | { |
|  |  |
|  | ostringstream convert; |
|  | convert << Posicion; |
|  | Result = convert.str(); |
|  | newIndice=ID+','+Result+','+'0'; |
|  | tamNewIndice=newIndice.length(); |
|  | Indices<<newIndice<<endl; |
|  | Posicion=Indices.tellg()-tamNewIndice-2; |
|  | ostringstream convert2; |
|  | convert2 << Posicion; |
|  | Result = convert2.str(); |
|  | newCluster=ID+','+Result; |
|  | fstream HashIndices; |
|  | HashIndices.open("indices/"+this->archivo+"/ClusterKey-"+this->archivo+"-00000.txt",ios::in|ios::out|ios::app); |
|  | HashIndices<<newCluster<<endl; |
|  | return "Insercion hecha Correctamente"; |
|  | } |
|  | else |
|  | { |
|  | return "Error"; |
|  | } |
|  | } |
|  | else |
|  | { |
|  | return "Error"; |
|  | } |
|  | } |
|  | string task\_usage::insertTU(string ID,string newline) |
|  | { |
|  | char\* linea=new char[1000]; |
|  | int tamNewline=newline.length(),tamNewIndice,tamnewCluster; |
|  | char \*a=new char[ID.length()+1]; |
|  | strcpy(a, ID.c\_str()); |
|  | long long int Posicion,IDn=atol(a); |
|  | string newIndice,newCluster,Result; |
|  | fstream BD; |
|  | BD.open("google/"+this->archivo+"/part-00000-of-"+numero(this->cantidad)+".csv",ios::in|ios::out|ios::app); |
|  | if(BD!=NULL) |
|  | { |
|  | BD<<newline<<endl; |
|  | Posicion=BD.tellg()-tamNewline-2; |
|  | fstream Indices; |
|  | Indices.open("indices/"+this->archivo+"/"+this->archivo+"-00000.txt",ios::in|ios::out|ios::app); |
|  | if(Indices!=NULL) |
|  | { |
|  |  |
|  | ostringstream convert; |
|  | convert << Posicion; |
|  | Result = convert.str(); |
|  | newIndice=ID+','+Result+','+'0'; |
|  | tamNewIndice=newIndice.length(); |
|  | Indices<<newIndice<<endl; |
|  | Posicion=Indices.tellg()-tamNewIndice-2; |
|  | ostringstream convert2; |
|  | convert2 << Posicion; |
|  | Result = convert2.str(); |
|  | newCluster=ID+','+Result; |
|  | fstream HashIndices; |
|  | HashIndices.open("indices/"+this->archivo+"/ClusterKey-"+this->archivo+"-00000.txt",ios::in|ios::out|ios::app); |
|  | HashIndices<<newCluster<<endl; |
|  | return "Insercion hecha Correctamente"; |
|  | } |
|  | else |
|  | { |
|  | return "Error"; |
|  | } |
|  | } |
|  | else |
|  | { |
|  | return "Error"; |
|  | } |
|  | } |