//功能实现版/home/web/ztedatabase/input1.csv

#include<iostream>

#include <vector>

#include<algorithm>

#include<ctime>

#include<map>

#include<set>

#include<fstream>

#include<sstream>

#include<unordered\_map>

using namespace std;

struct CSVDATA1 {

int id1, id2,id3;

};

struct CSVDATA2 {

int id1, id2,id3;

};

bool compid2(struct CSVDATA1 & a, struct CSVDATA1 & b)

{

return a.id2 < b.id2;

}

bool compid3(struct CSVDATA2 & a, struct CSVDATA2 & b)

{

return a.id3 < b.id3;

}

struct JOIN {

int t1\_id1,t1\_id2, t2\_id1,t2\_id2;

};

struct RESULT1{

int max,min,t2\_id2,t1\_id2;

bool operator < (const RESULT1 &a) const

{

//按id3由小到大排列，如果要由大到小排列，使用“>”号即可

if(max<a.max) return true ;

else if(max==a.max&&t2\_id2<a.t2\_id2) return true;

else if(max==a.max&&t2\_id2==a.t2\_id2&&t1\_id2<a.t1\_id2) return true;

else return false;

}

};

//读取csv文件，数据保存在vector<struct CSVDATA>中

vector<CSVDATA1> READCSV1(char const \*file)

{

vector<CSVDATA1> incsv;

CSVDATA1 intp;

FILE \*fp;

fp=fopen(file,"r");

while(!feof(fp))

{

fscanf(fp,"%d,%d,%d",&intp.id1,&intp.id2,&intp.id3);

incsv.push\_back(intp);

if(feof(fp))

{

break;

}

}

fclose(fp);

return incsv;

}

vector<CSVDATA2> READCSV2(char const \*file)

{

vector<CSVDATA2> incsv;

CSVDATA2 intp;

FILE \*fp;

fp=fopen(file,"r");

while(!feof(fp))

{

fscanf(fp,"%d,%d,%d",&intp.id1,&intp.id2,&intp.id3);

incsv.push\_back(intp);

if(feof(fp))

{

break;

}

}

fclose(fp);

return incsv;

}

void Bucket\_Sort1(vector<CSVDATA1>& A);

void Bucket\_Sort2(vector<CSVDATA2>& A);

//实现JOIN ON

vector<struct JOIN> JOINF(vector<struct CSVDATA1>& CSVdata1,vector<struct CSVDATA2>& CSVdata2)

{

vector<struct JOIN> Join;

struct JOIN join;

unordered\_map<int,int> mp;

int len\_csv1=CSVdata1.size();

int len\_csv2=CSVdata2.size();

//cout << "The run time is:" << (double)clock() /CLOCKS\_PER\_SEC<< "s" << endl;

Bucket\_Sort1(CSVdata1);

Bucket\_Sort2(CSVdata2);

// cout << "The run time is:" << (double)clock() /CLOCKS\_PER\_SEC<< "s" << endl;

for(int i=0;i<len\_csv2;i++)

{

mp.insert(make\_pair(CSVdata2[i].id3,i));

}

for(int i=0;i<len\_csv1;i++)

{

unordered\_map<int,int>::iterator it;

it=mp.find(CSVdata1[i].id3);

if(it!=mp.end())

{

int j=it->second;

while(CSVdata1[i].id3==CSVdata2[j].id3)

{

join={CSVdata1[i].id1,CSVdata1[i].id2,CSVdata2[j].id1,CSVdata2[j].id2};

Join.push\_back(join);

j++;

if(j==len\_csv2) break;

}

}

}

return Join;

}

//实现GROUP BY

vector< RESULT1> GROUP(vector<struct JOIN>& JOINYY)

{

struct RESULT1 temp;

vector<RESULT1> result;

vector<int> res\_row;

unordered\_map<int,int> t1\_id2;

unordered\_map<int,int> t2\_id2;

unordered\_map<int,int>::iterator t1;

unordered\_map<int,int>::iterator t2;

int len\_joinyy=JOINYY.size();

for(int i=0;i<len\_joinyy;i++)

{

if(t1\_id2.find(JOINYY[i].t1\_id2)==t1\_id2.end())

{

res\_row.push\_back(i);

t2\_id2.clear();

t1\_id2.insert(make\_pair(JOINYY[i].t1\_id2,i));

t2\_id2.insert(make\_pair(JOINYY[i].t2\_id2,i));

}

else

{

if(t2\_id2.find(JOINYY[i].t2\_id2)==t2\_id2.end())

{

res\_row.push\_back(i);

t2\_id2.insert(make\_pair(JOINYY[i].t2\_id2,i));

}

else

{

int group\_min,group\_min2;

t2=t2\_id2.find(JOINYY[i].t2\_id2);

int j=t2->second;

if(JOINYY[i].t1\_id2==JOINYY[j].t1\_id2 && JOINYY[i].t2\_id2==JOINYY[j].t2\_id2)

{

if (JOINYY[i].t2\_id1< JOINYY[j].t2\_id1)

{

group\_min=JOINYY[i].t2\_id1;

group\_min2=JOINYY[i].t2\_id2;

}

else

{

group\_min=JOINYY[j].t2\_id1;

group\_min2=JOINYY[j].t2\_id2;

}

if(JOINYY[i].t1\_id1>JOINYY[j].t1\_id1)

{

JOINYY[j].t1\_id1=JOINYY[i].t1\_id1;

}

JOINYY[j].t2\_id1=group\_min;

JOINYY[j].t2\_id2=group\_min2;

}

}

}

}

for(int i=0;i<res\_row.size();i++)

{

int j=res\_row[i];

temp={JOINYY[j].t1\_id1,JOINYY[j].t2\_id1,JOINYY[j].t2\_id2,JOINYY[j].t1\_id2};

result.push\_back(temp);

}

return result;

}

//实现ORDER BY

void ORDERBY(vector<struct RESULT1> &result)

{

sort(result.begin(),result.end());

}

int main()

{

vector<struct CSVDATA1> csvdata1;

vector<struct CSVDATA2> csvdata2;

vector<struct JOIN> JOINNY; //join 后的结果

vector<RESULT1> result;

// 显示读取的数据

char const \*file1="/media/chen/F/C++/mycode/ztedatabase/input3.csv";

char const \*file2="/media/chen/F/C++/mycode/ztedatabase/input4.csv";

csvdata1=READCSV1(file1);

csvdata2=READCSV2(file2);

JOINNY=JOINF(csvdata1,csvdata2);

vector<struct CSVDATA1>().swap(csvdata1);

vector<struct CSVDATA2>().swap(csvdata2);

result=GROUP(JOINNY);

vector<struct JOIN>().swap(JOINNY);

ORDERBY(result);

for(auto it=result.begin();it!=result.end();it++)

{

printf("%d,%d\n",(\*it).max,(\*it).min);

}

vector<RESULT1>().swap(result);

cout << "The run time is:" << (double)clock() /CLOCKS\_PER\_SEC<< "s" << endl;

return 0;

}

void Bucket\_Sort1(vector<CSVDATA1>& A)

{

int len\_csv=A.size();int i=0,j=0;

vector<CSVDATA1>::iterator p = max\_element(A.begin(), A.end(),compid2);

vector<CSVDATA1>::iterator q = min\_element(A.begin(), A.end(),compid2);

int max=p->id2,min=q->id2;

int num=max-min+1;

vector<vector<CSVDATA1>> a(num,vector<CSVDATA1> (0));

CSVDATA1 temp;

for(i=0;i<len\_csv;i++)

{

temp=A[i];int t=temp.id2;

// if(t!=i) continue;

a[t-min].push\_back(temp);

}

int index=0;

for(i=0;i<num;i++)

{

if(a[i].empty()) continue;

for(j=0;j<a[i].size();j++)

{

temp=a[i][j];

A[index++]=temp;

}

}

}

void Bucket\_Sort2(vector<CSVDATA2>& A)

{

int len\_csv=A.size();int i=0,j=0;

vector<CSVDATA2>::iterator p = max\_element(A.begin(), A.end(),compid3);

vector<CSVDATA2>::iterator q = min\_element(A.begin(), A.end(),compid3);

int max=p->id3,min=q->id3;

int num=max-min+1;

vector<vector<CSVDATA2>> a(num,vector<CSVDATA2> (0));

CSVDATA2 temp;

for(i=0;i<len\_csv;i++)

{

temp=A[i];int t=temp.id3;

a[t-min].push\_back(temp);

}

int index=0;

for(i=0;i<num;i++)

{

if(a[i].empty()) continue;

for(j=0;j<a[i].size();j++)

{

temp=a[i][j];

A[index++]=temp;

}

}

}