Join on

Map版

vector<struct JOIN> JOINF(set<CSVDATA>& csvdata1,set<CSVDATA>& csvdata2)

{

// int len=0;

struct JOIN temp;

//unordered\_map<JOIN,int,hash\_name>TEMP;

vector<struct JOIN> TEMP;

auto it1=csvdata1.begin(),it2=csvdata2.begin(),it3=csvdata2.end();

it3--;

while(it1!=csvdata1.end()&&it2!=csvdata2.end())

{

if((\*it1).id3==(\*it2).id3)

{

temp={(\*it1).id1,(\*it1).id2,(\*it2).id1,(\*it2).id2};

//TEMP.insert(pair<JOIN,int> (temp,len));

TEMP.push\_back(temp);

// len++;

if(it2==it3&&it1!=csvdata1.end())

{

it1++;it2=csvdata2.begin();

}

else

{

it2++;

}

}

else if((\*it1).id3>(\*it2).id3)

{

it2++;

}

else if((\*it1).id3<(\*it2).id3)

{

it1++;it2=csvdata2.begin();

}

}

return TEMP;

}

Unordered\_map版

vector<struct JOIN> JOINF(unordered\_map<CSVDATA, int,hash\_name1> & csvdata1,unordered\_map<CSVDATA, int,hash\_name1> & csvdata2)

{

// int len=0;

struct JOIN temp;

vector<struct JOIN>TEMP;

auto it1=csvdata1.begin(),it2=csvdata2.begin(),it3=csvdata2.end();

//it3--;

while(it1!=csvdata1.end()&&it2!=csvdata2.end())

{

if((\*it1).first.id3==(\*it2).first.id3)

{

temp={(\*it1).first.id1,(\*it1).first.id2,(\*it2).first.id1,(\*it2).first.id2};

// TEMP.insert(pair<JOIN,int> (temp,len));

TEMP.push\_back(temp);

// len++;

if(next(it2)==it3&&it1!=csvdata1.end())

{

it1++;it2=csvdata2.begin();

}

else

{

it2++;

}

}

else if((\*it1).first.id3>(\*it2).first.id3)

{

it2++;

}

else if((\*it1).first.id3<(\*it2).first.id3)

{

it1++;it2=csvdata2.begin();

}

}

return TEMP;

}

功能实现版

//功能实现版

//大改动测试

#include <iostream>

#include <fstream>

#include <vector>

#include<algorithm>

#include<ctime>

#include <sstream> // 用于读写存储在内存中的string对象

using namespace std;

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

void ORDERBY(vector<struct RESULT1> &result1);

struct CSVDATA {

int id1;

int id2;

int id3;

};

struct CSVDATA1 {

int id1;

int id2;

int t2\_id2;

int t1\_id2;

};

struct JOIN {

int t1\_id1;

int t1\_id2;

int t2\_id1;

int t2\_id2;

};

struct RESULT1{

int max;

int min;

int t2\_id2;

int t1\_id2;

};

struct RESULT{

int max;

int min;

};

vector<struct RESULT1> result1;

bool compt1\_id2(struct JOIN &a, struct JOIN &b)

{

return a.t1\_id2 < b.t1\_id2;

}

bool compt1id2(struct CSVDATA &a, struct CSVDATA &b)

{

return a.id2 < b.id2;

}

bool compmax(struct RESULT1 &a, struct RESULT1 &b)

{

return a.max>b. max;

}

bool compmin(struct RESULT1 &a, struct RESULT1 &b)

{

return a.min<b. min;

}

bool comp2t1\_id2(struct RESULT1 &a, struct RESULT1 &b)

{

return a.t1\_id2<b. t1\_id2;

}

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

{

return a.id3 < b.id3;

}

bool compt2\_id2(struct CSVDATA1 &a, struct CSVDATA1 &b)

{

return a.t2\_id2 < b.t2\_id2;

}

bool compmax\_t1\_id1(struct RESULT1 &a, struct RESULT1 &b)

{

return a.max<b. max;

}

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

vector<CSVDATA> READCSV(char const \*file)

{

vector<CSVDATA> incsv;

CSVDATA 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;

}

//实现JOIN ON

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

{

vector<struct JOIN> Join;

struct JOIN join;

sort(CSVdata1.begin(),CSVdata1.end(),compid3);

sort(CSVdata2.begin(),CSVdata2.end(),compid3);

int a=CSVdata1.size(),b=CSVdata2.size();

for(int i=0,j=0;i<a && j<b; )

{

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

{

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

Join.push\_back(join);

if(j==b-1&&i<a)

{

i++;j=0;

}

else

{

j++;

}

/\*

if(Join.size()>=1000000)

{

result1=GROUP(Join);

ORDERBY(result1);

for (int i = 0; i <result1.size() ; i++ )

{

printf("%d,%d\n", result1[i].max,result1[i].min);

}

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

}

\*/

}

else if(CSVdata1[i].id3>CSVdata2[j].id3)

{

j++;

}

else if(CSVdata1[i].id3<CSVdata2[j].id3)

{

i++;j=0;

}

}

return Join;

}

//实现GROUP BY

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

{

// int len\_joinyy=JOINYY.size();

sort(JOINYY.begin(),JOINYY.end(),compt1\_id2);//先按照t1\_id2升序排序

vector<struct RESULT1> result;//保存最后的结果

vector<struct RESULT1> Result;//再分组后的数据

struct RESULT1 temp1;

struct RESULT1 temp;

CSVDATA1 Temp;//

vector<CSVDATA1> TEMP;//分组后的数据

int length\_joinyy=JOINYY.size();

int flag1=0;

int flag2=0;

for(int i=0;i<length\_joinyy;)

{

if(flag2==1) flag2=0;

if(i!=length\_joinyy-1)

{

if(JOINYY[i].t1\_id2==JOINYY[i+1].t1\_id2)

{

Temp={JOINYY[i].t1\_id1,JOINYY[i].t2\_id1,JOINYY[i].t2\_id2,JOINYY[i].t1\_id2};

TEMP.push\_back(Temp);

i++;continue;

}

else

{

Temp={JOINYY[i].t1\_id1,JOINYY[i].t2\_id1,JOINYY[i].t2\_id2,JOINYY[i].t1\_id2};

TEMP.push\_back(Temp);

i++;

}

}

else

{

Temp={JOINYY[i].t1\_id1,JOINYY[i].t2\_id1,JOINYY[i].t2\_id2,JOINYY[i].t1\_id2};

TEMP.push\_back(Temp);flag1=1;

}

int length\_TEMP=TEMP.size();

if(length\_TEMP==1)

{

temp={Temp.id1,Temp.id2,Temp.t2\_id2,Temp.t1\_id2};

result.push\_back(temp);

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

}

else if(length\_TEMP>1)

{

sort(TEMP.begin(),TEMP.end(),compt2\_id2);

for(int j=0;j<length\_TEMP;)

{

if(j!=length\_TEMP-1)

{

if(TEMP[j].t2\_id2==TEMP[j+1].t2\_id2)

{

temp1={TEMP[j].id1,TEMP[j].id2,TEMP[j].t2\_id2,TEMP[j].t1\_id2};

Result.push\_back(temp1);

j++;continue;

}

else

{

temp1={TEMP[j].id1,TEMP[j].id2,TEMP[j].t2\_id2,TEMP[j].t1\_id2};

Result.push\_back(temp1);

j++;

}

}

else

{

temp1={TEMP[j].id1,TEMP[j].id2,TEMP[j].t2\_id2,TEMP[j].t1\_id2};

Result.push\_back(temp1);

flag2=1;

}

int length\_Result=Result.size();

if(length\_Result==1)

{

temp={Result[0].max,Result[0].min,Result[0].t2\_id2,Result[0].t1\_id2};

result.push\_back(temp);

vector<struct RESULT1>().swap(Result);

}

else if(length\_Result>1)

{

sort(Result.begin(),Result.end(),compmax);

temp.max=Result[0].max;temp.t2\_id2=Result[0].t2\_id2;

sort(Result.begin(),Result.end(),compmin);

temp.min=Result[0].min;temp.t1\_id2=Result[0].t1\_id2;

result.push\_back(temp);

vector<struct RESULT1>().swap(Result);

TEMP.clear();

}

if(flag2==1) break;

}

}

if(flag1==1) break;

}

return result;

}

//实现ORDER BY

void ORDERBY(vector<struct RESULT1> &result1)

{

int len\_result11=result1.size();

for(int i=0;i<len\_result11;i++)

{

int temp=i;

for(int j=i+1;j<len\_result11;j++)

{

if(result1[temp].max>result1[j].max)

{

temp=j;

}

else if(result1[temp].max==result1[j].max&&result1[temp].t2\_id2>result1[j].t2\_id2)

{

temp=j;

}

else if(result1[temp].max==result1[j].max&&result1[temp].t2\_id2==result1[j].t2\_id2&&result1[temp].t1\_id2>=result1[j].t1\_id2)

{

temp=j;

}

}

swap(result1[i],result1[temp]);

}

// \*result1=result11;

}

int main()

{

vector<struct CSVDATA> csvdata1;

vector<struct CSVDATA> csvdata2;

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

//vector<struct RESULT1> result1; //group by之后的结果

// 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=READCSV(file1);

csvdata2=READCSV(file2);

JOINNY=JOINF(csvdata1,csvdata2);

// cout<<JOINNY.size()<<endl;

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

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

result1=GROUP(JOINNY);

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

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

// cout<<result1.size()<<endl;

ORDERBY(result1);//引用

for (int i = 0; i <result1.size() ; i++ )

{

printf("%d,%d\n", result1[i].max,result1[i].min);

}

vector<struct RESULT1>().swap(result1);

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

return 0;

}