```cpp
// SimQP.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "dbmsproj.h"
#include "semihashjoin.h"
#include <string>
#include <sstream>
#include <stdio.h>
#define GET_VARIABLE_NAME(Variable) (#Variable)
/***Using vectors represent disk to hold records block*****/

//NYC - This site has a copy of branch, account and depositor tables.
std::vector<block_t<account_t>> account_NYC;
std::vector<block_t<branch_t>> branch_NYC;
std::vector<block_t<depositor_t>> depositor_NYC;

//SFO - This site has complete branch table and a fragment of account table.
//Account table will have only records that are local to SFO.
std::vector<block_t<branch_t>> branch_SFO;
std::vector<block_t<account_t>> frag_account_SFO;

//OMA - This site has fragments of Account and Depositor tables.
//These tables store records that are local to OMA.
std::vector<block_t<account_t>> frag_account_OMA;
std::vector<block_t<depositor_t>> frag_depositor_OMA;

//HOU (Houston) – This site has a complete copy of customer table.
std::vector<block_t<customer_t>> customer_HOU;


//unsigned int MAX_RECORDS_PER_BLOCK=7;


int main(int argc, char* argv[])
{

    //initialize
    char* filename;

    //fill account table at NYC
    filename = "Account_NYC.txt";
    fillTable<account_t>(filename, account_NYC, 6);

    //fill branch table at NYC
    filename = "Branch_NYC.txt";
    fillTable<branch_t>(filename, branch_NYC, 1);

    //fill depositor table at NYC
    filename = "Depositor_NYC.txt";
    fillTable<depositor_t>(filename, depositor_NYC, 4);

    //fill customer table at HOU
    filename= "Customer_HOU.txt";
    fillTable<customer_t>(filename, customer_HOU, 8);

    //fill fragment of account table at SFO
    filename = "Frag_Account_SFO.txt";
    fillTable<account_t>(filename, frag_account_SFO, 2);
```

```
60
61      //fill fragment of depositor table at OMA
62      filename = "Frag_Depositor_OMA.txt";
63      fillTable<depositor_t>(filename, frag_depositor_OMA, 2);
64
65      //fill fragment of account table at OMA
66      filename = "Frag_Account_OMA.txt";
67      //fillTable<account_t>(filename, frag_account_OMA, 3);
68      //printf("All tables are filled.\n");
69      //printf("============================================================\n");
70      std::istringstream ss(argv[1]);
71      int query;
72      if (!(ss >> query)) std::cerr << "Invalid number" << argv[1] << std::endl;
73
74      std::vector<std::string> command;
75      std::vector<std::string> parameters;
76      std::string require_site;
77      std::string current_site;
78
79
80      switch (query)
81      {
82      case 1:
83              {
84                  std::vector<block_t<customer_t>> out;
85                  printf("Query 1: \n");
86                  printf("Require site: NYC\n");
87                  printf("Statement: Select* from customer |X depositor:\n");
88                  printf("Processing:\n");
89                  printf("The customer table is shipped from HOU to NYC site. \n");
90
91                  printf("\n");
92                  SemiHashJoin<customer_t,depositor_t>
                      (customer_HOU,depositor_NYC,"customer_name", out);
93                  printf("The customer table are SemiHashJoined with depositor table at NYC
                       site. \n");
94                  printf("Project all fields in customer table.\n");
95                  printf("\n");
96                  printf("Output: \n");
97                  printf("=========================================\n");
98                  printf("Name    Street  City \n");
99                  printf("=========================================\n");
100                 unsigned int count = 0;
101                 for each ( auto block in out)
102                 {
103
104                     for each (auto record in block.entries)
105                         {
106                             record.project(3, "customer_name",
                        "customer_city","customer_street");
107                             count++;
108                         }
109
110                 }
111                 printf("=========================================\n");
112                 printf("The query output %d records.\n",count);
113                 break;
114         }
```

```
115     case 2:
116           {
117
118               printf("Query 2: \n");
119               printf("Require site: SFO\n");
120               printf("Statement: Select name, balance from depositor |X| account where
                    branch_name='Chinatown':\n");
121               printf("Processing:\n");
122
123               printf("3) Ship the results from NYC to SFO and then do projection. \n");
124               printf("\n");
125                std::vector<block_t<account_t>> table_selec;
126               printf("Select the records with branch_name='Chinatown' using the fragment
                    of account table at SFO.\n");
127                select(account_NYC, table_selec, "branch_name", "Chinatown");
128
129                if (table_selec.size() == 0)
130                {
131                    printf("no record is selected according to the condition.\n");
132                    break;
133                }
134                else
135                {
136                    unsigned int rec_selected = (table_selec.size() - 1)*table_selec
                      [0].maxRecords + table_selec[table_selec.size()-1].nreserved;
137                    //printf("%d records are selected.\n", rec_selected);
138                }
139                printf("Ship the selection results from SFO to HOU  and hash joined with
                    depositor table. \n");
140                std::vector<join_t<depositor_t,account_t>> joinout;
141                HashJoin<depositor_t,account_t>(depositor_NYC,
                    table_selec,"account_number", joinout);
142                printf("The join results at HOU is shipped back to SFO. \n");
143                printf("\n");
144                printf("Output: \n");
145                printf("=========================================\n");
146                printf("Name Balance\n");
147                printf("=========================================\n");
148                unsigned int count=0;
149                for each (auto var in joinout)
150               {
151                        var.display(2,"customer_name","balance");
152                        count++;
153               }
154                printf("=========================================\n");
155                printf("The query output %d records.\n", count);
156                break;
157           }
158     case 3:
159     {
160
161         printf("Query 3: \n");
162         printf("Require site: SFO\n");
163         printf("Statement: Select street, city from customer|X (depositor |X account
              where account_number='A10352'):\n");
164         printf("Processing:\n");
165
166         printf("\n");
```

```
167         std::vector<block_t<account_t>> table_selec;
168
169         printf("Search the fragment of account table at SFO to check if there have
              records with account_number='A10352' \n");
170         select(frag_account_SFO, table_selec, "account_number", "A10352");
171         if (table_selec.size() == 0)
172         {
173             printf("no record is found according to the condition.\n");
174             select(account_NYC, table_selec, "account_number", "A10352");
175             printf("Search the account table at NYC.\n");
176         }
177
178         if (table_selec.size() == 0)
179         {
180             printf("no record is selected according to the condition.\n");
181             break;
182         }
183         else
184         {
185
186             unsigned int rec_selected = (table_selec.size() - 1)*table_selec
                [0].maxRecords + table_selec[table_selec.size() - 1].nreserved;
187             //printf("%d records are selected.\n", rec_selected);
188         }
189
190         std::vector<block_t<depositor_t>> semi_out;
191         printf("The depositor table is semi-joined with intermediate result from select
              operation at NYC.\n");
192         SemiHashJoin<depositor_t,account_t>(depositor_NYC,table_selec,"account_number",
              semi_out);
193
194         printf("The intermedate results from semijoin are shipped from NYC to HOU. \n");
195         printf("The customer table at HOU is hash joined with the intermedate results.
              \n");
196         std::vector<join_t<customer_t, depositor_t>> joinout;
197         HashJoin<customer_t, depositor_t>(customer_HOU, semi_out, "customer_name",
              joinout);
198
199         printf("The final result is shipped from HOU to the SFO. \n");
200
201         printf("Output: \n");
202         printf("=========================================\n");
203         printf("Street      City\n");
204         printf("=========================================\n");
205
206         unsigned int count=0;
207         for each (auto var in joinout)
208         {
209             var.display(2, "customer_street", "customer_city");
210             count++;
211         }
212         printf("=========================================\n");
213         printf("The query output %d records.\n", count);
214         break;
215     }
216     case 4:
217     {
218
219         printf("Query 4: \n");
```

```
220         printf("Require site: NYC\n");
221         printf("Statement: Select account_number, balance, branch_name branch_city from
              branch|X| acocunt where account_number='A10352'):\n");
222         printf("Processing:\n");
223
224         std::vector<block_t<account_t>> table_selec;
225         printf("Select the record from the account table at NYC where
              account_number='A10352' \n");
226         select(account_NYC, table_selec, "account_number", "A10352");
227         if (table_selec.size() == 0)
228         {
229             //printf("no record is selected according to the condition.\n");
230             break;
231         }
232         else
233         {
234             unsigned int rec_selected = (table_selec.size() - 1)*table_selec
                [0].maxRecords + table_selec[table_selec.size() - 1].nreserved;
235             //printf("%d records are selected.\n", rec_selected);
236         }
237
238         std::vector<join_t<branch_t, account_t>> joinout;
239         printf("The branch table is hash joined with intermediate result from select
              operation at NYC.\n");
240         HashJoin<branch_t, account_t>(branch_NYC, table_selec, "branch_name", joinout);
241         printf("=========================================\n");
242
243         printf("Output: \n");
244         printf("=========================================\n");
245         printf("account_nmber   balance branch_name branch_city\n");
246         printf("=========================================\n");
247         unsigned int count = 0;
248         for each (auto var in joinout)
249         {
250             var.display(4, "account_number", "balance", "branch_name", "branch_city");
251             count++;
252         }
253         printf("=========================================\n");
254         printf("The query output %d records.\n", count);
255         break;
256     }
257     default:
258         break;
259     }
260
261     printf("\n");
262
263 }
264
265
```