


 MelvinLecoy / gitcode Private[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [Settings](#) master ▾

...

[gitcode](#) / [p3-database](#) / silly.cpp

Kwan Ting Lau gitcode

 History 0 contributors

378 lines (343 sloc) | 9.9 KB

...

```
1 // Project Identifier: C0F4DFE8B340D81183C208F70F9D2D797908754D
2 #include "Table.h"
3 #include "TableEntry.h"
4 #include "getopt.h" //
5 #include <algorithm> //
6 #include <cstdlib>
7 #include <cstring> //
8 #include <exception> //
9 #include <getopt.h>
10 #include <ios>
11 #include <iostream> //
12 #include <iterator>
13 #include <map> //
14 #include <string>
15 #include <unordered_map>
16 #include <utility> //
17 #include <vector> //
18
19 using namespace std;
20
21 class Silly {
22
23 public:
24     bool quiet = false;
25     unordered_map<string, Table> map;
26
27     void QUIT() { // 地毯
28         cout << "Thanks for being silly!"
29              << "\n";
30     }
31
32     void CREATE() { //地毯
33         string tablename;
34         cin >> tablename;
35         if (map.count(tablename)) {
```

```
36     cout << "Error during CREATE: Cannot create already existing table "
37         << tablename << "\n";
38     getline(cin, tablename);
39 } else {
40     int col_num;
41     cin >> col_num;
42     Table temp;
43     temp.table_init(col_num);
44     map.emplace(tablename, temp);
45     cout << "New table " << tablename << " with column(s) ";
46     for (int i = 0; i < col_num; i++) {
47         auto ans = map[tablename].columns[i].column_name;
48         cout << ans << " ";
49     }
50     cout << "created"
51         << "\n";
52 }
53 }
54
55 void REMOVE() { // 地毯
56     string name;
57     cin >> name;
58     if (!map.count(name)) {
59         cout << "Error during REMOVE: " << name
60             << " does not name a table in the database "
61             << "\n";
62
63         getline(cin, name);
64     } else {
65         map.erase(name);
66         cout << "Table " << name << " deleted"
67             << "\n";
68     }
69 }
70
71 void PRINT() { // 地毯
72     string junk;
73     string tablename;
74     cin >> junk >> tablename;
75     if (map.count(tablename)) {
76         int num;
77         cin >> num;
78         num = map[tablename].print_all(num, tablename, quiet);
79         if (num == -1) {
80             getline(cin, tablename);
81         } else {
82             cout << "Printed " << num << " matching rows from " << tablename
83                 << "\n";
84         }
85     } else {
86         cout << "Error during PRINT: " << tablename
87             << " does not name a table in the database "
88             << "\n";
89     }
```

```
90     getline(cin, tablename);
91 }
92 }
93
94 void JOIN() { // 检查一下cin 还有最后的分类是不是正确
95     string junk;
96     string table_1;
97     string table_2;
98     string colname_1;
99     string colname_2;
100
101     int N;
102     int value_1;
103     int value_2;
104     int table_value;
105     int col_value;
106
107     cin >> table_1 >> junk >> table_2 >> junk;
108
109     if (map.count(table_1) && map.count(table_2)) {
110         cin >> colname_1 >> junk >> colname_2 >> junk >> junk >> N;
111
112         value_1 = map[table_1].find_column(colname_1);
113         value_2 = map[table_2].find_column(colname_2);
114         if (value_1 == -1) {
115             cout << "Error during JOIN: " << colname_1
116                  << " does not name a column in " << table_1 << "\n";
117             getline(cin, junk);
118             return;
119         }
120         if (value_2 == -1) {
121             cout << "Error during JOIN: " << colname_2
122                  << " does not name a column in " << table_2 << "\n";
123             cin.ignore(numeric_limits<streamsize>::max(), '\n');
124             return;
125         }
126
127         vector<string> up;
128         vector<pair<int, int>> down;
129
130         for (int i = 0; i < N; i++) {
131             cin >> junk >> col_value;
132             if (col_value == 1) {
133                 table_value = map[table_1].find_column(junk);
134             } else {
135                 table_value = map[table_2].find_column(junk);
136             }
137             if (table_value != -1) {
138                 down.emplace_back(make_pair(col_value, table_value));
139                 up.emplace_back(junk);
140             } else {
141                 if (col_value == 1) {
142                     cout << "Error during JOIN: " << junk
143                          << " does not name a column in " << table_1 << "\n";
```

```
144     getline(cin, junk);
145     return;
146 } else {
147     cout << "Error during JOIN: " << junk
148         << " does not name a column in " << table_2 << "\n";
149     getline(cin, junk);
150     return;
151 }
152 }
153 }
154 int count_2 = 0;
155 if (quiet) {
156     unordered_map<TableEntry, vector<size_t>> middle;
157
158     for (int row = 0; row < static_cast<int>(map[table_1].data.size());
159         row++) {
160         auto it = middle.find(map[table_1].data[row][value_1]);
161         if (it == middle.end()) {
162             vector<size_t> value_3(1, row);
163             middle.emplace(map[table_1].data[row][value_1], value_3);
164         } else {
165             it->second.push_back(row);
166         }
167     }
168     for (int row_2 = 0; row_2 < static_cast<int>(map[table_2].data.size());
169         row_2++) {
170         auto it = middle.find(map[table_2].data[row_2][value_2]);
171         if (it != middle.end()) {
172             count_2 += it->second.size();
173         }
174     }
175 } else {
176     for (int i = 0; i < static_cast<int>(down.size()); i++) {
177         cout << up[i] << " ";
178     }
179     cout << "\n";
180     unordered_map<TableEntry, vector<size_t>> middle_2;
181     for (int row_2 = 0; row_2 < static_cast<int>(map[table_2].data.size());
182         row_2++) {
183         auto it = middle_2.find(map[table_2].data[row_2][value_2]);
184         if (it == middle_2.end()) {
185             vector<size_t> value_4(1, row_2);
186             middle_2.emplace(map[table_2].data[row_2][value_2], value_4);
187         } else {
188             it->second.push_back(row_2);
189         }
190     }
191     for (int row_3 = 0; row_3 < static_cast<int>(map[table_1].data.size());
192         row_3++) {
193         auto it = middle_2.find(map[table_1].data[row_3][value_1]);
194         if (it != middle_2.end()) {
195             count_2 += it->second.size();
196             for (auto s : it->second) {
197                 for (pair<int, int> p : down) {
```

```
198         if (p.first == 1) {
199             cout << (map[table_1].data[row_3])[p.second] << " ";
200         } else {
201             cout << (map[table_2].data[s])[p.second] << " ";
202         }
203     }
204     cout << "\n";
205 }
206 }
207 }
208 }
209 cout << "Printed " << count_2 << " rows from joining " << table_1
210     << " to " << table_2 << "\n";
211 } else {
212     if (!map.count(table_1)) {
213         cout << "Error during JOIN: " << table_1
214             << " does not name a table in the database "
215             << "\n";
216         getline(cin, junk);
217     }
218     if (!map.count(table_2)) {
219         cout << "Error during JOIN: " << table_2
220             << " does not name a table in the database "
221             << "\n";
222         getline(cin, junk);
223     }
224 }
225 }
226
227 void INSERT() { //地毯
228     string INTO;
229     string tablename;
230     int N;
231     string junk;
232     cin >> INTO >> tablename >> N;
233     if (map.count(tablename)) {
234         cin >> junk;
235         int end = map[tablename].insert(N);
236         int start = end - N;
237         int final = end - 1;
238         cout << "Added " << N << " rows to " << tablename << " from position "
239             << start << " to " << final << "\n";
240     } else {
241         cout << "Error during INSERT: " << tablename
242             << " does not name a table in the database "
243             << "\n";
244         getline(cin, junk);
245     }
246 }
247
248 void DELETE() { // 地毯
249     string FROM;
250     string tablename;
251     cin >> FROM >> tablename;
```

```
252
253     auto it = map.find(tablename);
254     if (it == map.end()) {
255         cout << "Error udring DELETE: " << tablename
256             << " does not name a table in the database "
257             << "\n";
258         getline(cin, tablename);
259     } else {
260         int N = it->second.delete_row(tablename);
261         if (N == -1) {
262             getline(cin, tablename);
263         } else {
264             cout << "Deleted " << N << " rows from " << tablename << "\n";
265         }
266     }
267 }
268
269 void GENERATE() { // 地毯
270     string FOR;
271     string tablename;
272     string indextype;
273     string col_name;
274     cin >> FOR >> tablename;
275     auto it = map.find(tablename);
276     if (it == map.end()) {
277         cout << "Error during GENERATE: " << tablename
278             << " does not name a table in the database "
279             << "\n";
280         getline(cin, tablename);
281     } else {
282         cin >> indextype;
283         if (indextype == "hash") {
284             cin >> col_name >> col_name;
285             col_name = map[tablename].generate_hash(tablename);
286
287         } else {
288             cin >> col_name >> col_name;
289             col_name = map[tablename].generate_bst(tablename);
290         }
291         if (col_name != "ERROR") {
292             cout << "Created " << indextype << " index for table " << tablename
293                 << " on column " << col_name << "\n";
294         }
295     }
296 }
297
298 void shell() { // 地毯
299     string cmd;
300
301     cout << "% "
302         << " ";
303     while (cin >> cmd) {
304         if (cmd[0] == '#') {
305             getline(cin, cmd);
```

```
306     } else if (cmd == "QUIT") {
307         QUIT();
308         break;
309     } else if (cmd == "CREATE") {
310         CREATE();
311     } else if (cmd == "REMOVE") {
312         REMOVE();
313     } else if (cmd == "INSERT") {
314         INSERT();
315
316     } else if (cmd == "PRINT") {
317         PRINT();
318     } else if (cmd == "DELETE") {
319         DELETE();
320     } else if (cmd == "JOIN") {
321         JOIN();
322     } else if (cmd == "GENERATE") {
323         GENERATE();
324     } else {
325         cout << "Error: unrecognized command\n";
326         getline(cin, cmd);
327     }
328
329     cout << "%"
330         << " ";
331 }
332 }
333
334 void getMode(int argc, char *argv[]) { //地毯
335
336     string mode;
337     opterr = false;
338     int choice;
339     int option_index = 0;
340
341     option long_options[] = {
342         {"quiet", no_argument, nullptr, 'q'},
343         {"help", no_argument, nullptr, 'h'},
344         {nullptr, 0, nullptr, '\0'},
345     };
346
347     while ((choice = getopt_long(argc, argv, "qh", long_options,
348                                &option_index)) != -1) {
349         switch (choice) {
350             case 'h':
351                 cout << "something"
352                     << "\n";
353                 exit(1);
354                 break;
355
356             case 'q':
357                 quiet = true;
358                 break;
359
```

```
360         default:
361             cerr << "Unknown command line option" << '\n';
362             exit(1);
363         }
364     }
365     shell();
366 }
367 };
368
369 int main(int argc, char **argv) { //地毯
370     Silly s_1;
371     ios_base::sync_with_stdio(false);
372     cin >> boolalpha;
373     cout << boolalpha;
374
375     s_1.getMode(argc, argv);
376
377     return 0;
378 }
```

[Give feedback](#)