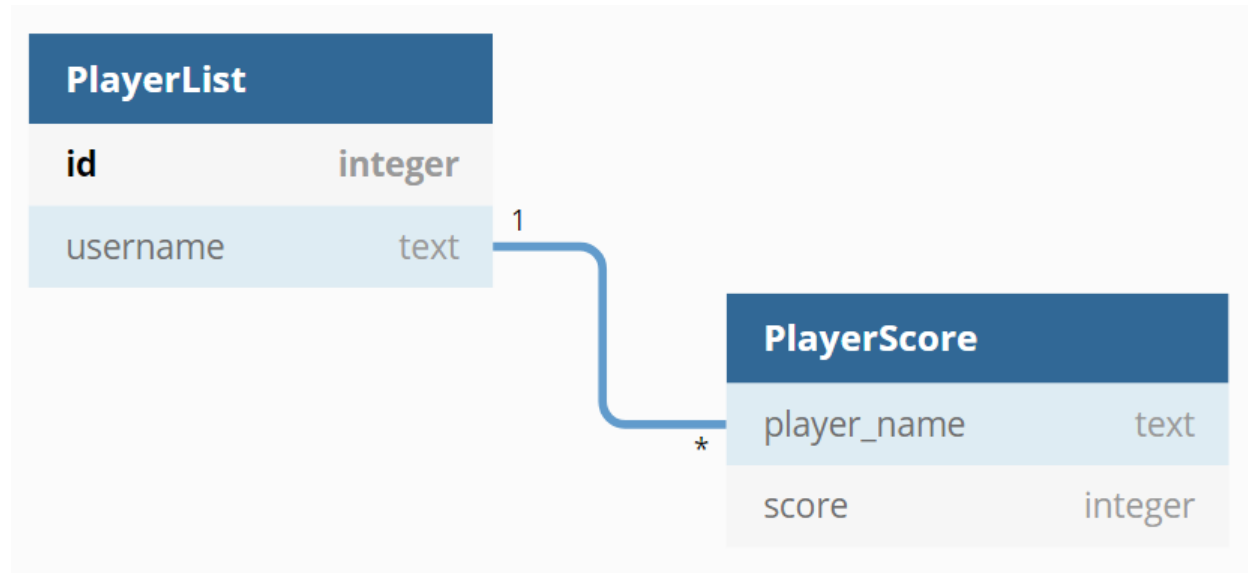


Contents

- I. Overview2
- II. Database script6

I. Overview

This database is integrated into a simple Snake Game using C++ . It's just a small game so there's not much data that need to process for database, so it's important to make sure there is no error and everything would work as intended.



As a one-to-many relationship, the **PlayerList** table holds **id** as a primary key (also auto-increment), and the **username** for each **id** is unique. Next, the **PlayerScore** table can hold multiple value of score of each player.

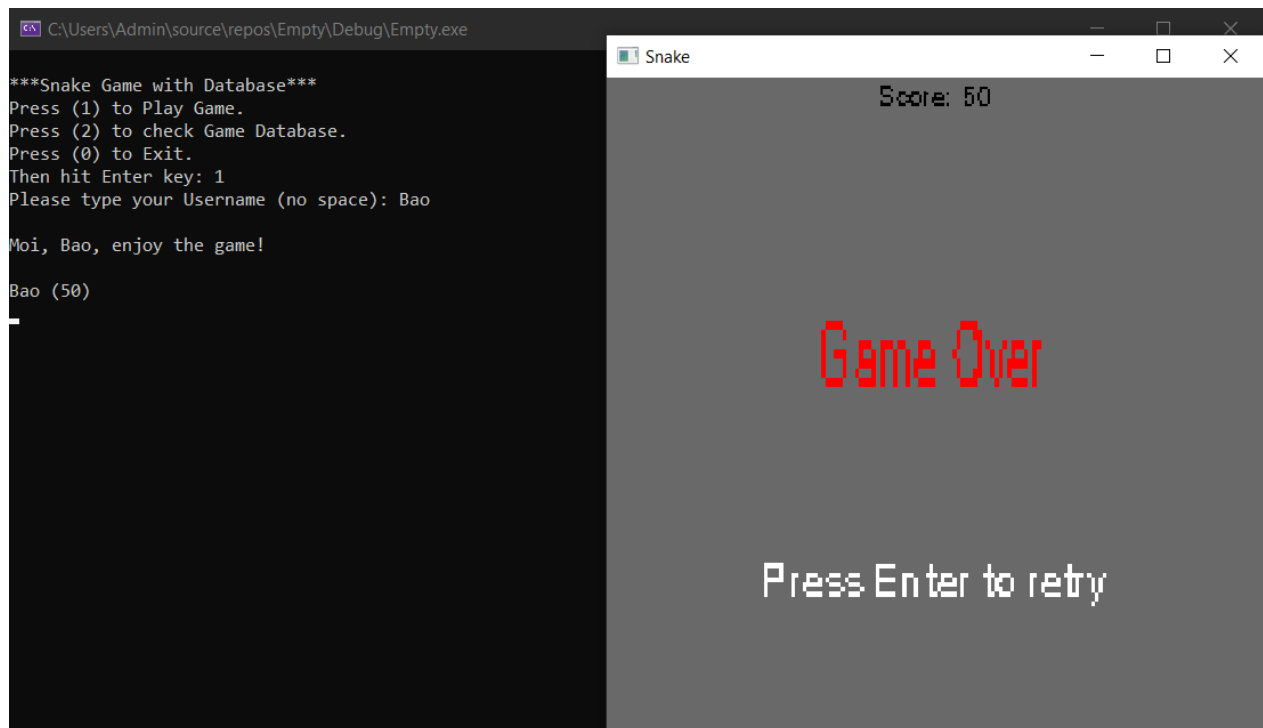
The **username** in **PlayerList** also utilizes CASCADE, so each time there's an update or delete occurs with it, the **player_name** in **PlayerScore** table will perform the same.

```
DataBase
+sqlite3* db;
+char* zErrMsg = 0;
+int rc;
+string sql;

+callback(void* NotUsed, int argc, char** argv, char** azColName)
+PrintPLcolumn()
+PrintPScolumn()
+CheckError()
+OpenDatabase()
+CloseDatabase()
+CreateTables()
+PopulatePlayerList(string name)
+PopulatePlayerScore(string name, int score)
+CheckPlayerList()
+CheckScore()
+FindSpecific(string f)
+UpdateName(string old, string newName)
+DeleteData(string ans)
+DropTable()
```

In DataBase class, there are 4 variables that mainly control the database. The sql variable contains the sql statement and uses sqlite3_exec to execute that with the callback function, which print out the table. I also modify this function combine with PrintPLcolumn and PrintPScolumn so it can print out a similar style to a data table with headers.

The OpenDatabase function will open the database called SnakeDatabase.db, and it turns on the foreign key to make sure CASCADE work properly. I call both OpenDatabase and CloseDatabase in almost every function because I usually got sqlite3 malloc error that I am not sure why, but this attempt seems to fix that. The CheckError function is also called with them to check error. The CreateTable function calls everytime the program starts.

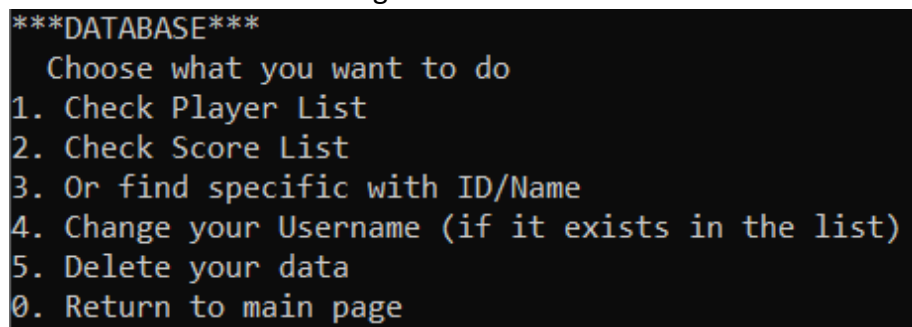


```
C:\Users\Admin\source\repos\Empty\Debug\Empty.exe
***Snake Game with Database***
Press (1) to Play Game.
Press (2) to check Game Database.
Press (0) to Exit.
Then hit Enter key: 1
Please type your Username (no space): Bao
Moi, Bao, enjoy the game!
Bao (50)
_

Snake
Score: 50
Game Over
Press Enter to retry
```

When the player starts game, it will ask the username then firstly populate it in PlayerList table. Next, it will then populate the player_name and score to PlayerScore table when game is over.

Other results when accessing database:



```
***DATABASE***
Choose what you want to do
1. Check Player List
2. Check Score List
3. Or find specific with ID/Name
4. Change your Username (if it exists in the list)
5. Delete your data
0. Return to main page
```

1. Check Player List

Your option: 1

ID	Username
1	Bao
2	thuyet

2. Check Player Score (ascending)

Your option: 2

ID	Player_Name	Score
1	Bao	0
1	Bao	50
2	thuyet	0
2	thuyet	10
2	thuyet	20

3. Find specific

What is your ID/Username: Bao

ID	Player_Name	Score
1	Bao	0
1	Bao	50

What is your ID/Username: 2

ID	Player_Name	Score
2	thuyet	0
2	thuyet	10
2	thuyet	20

4. Change Username

```
Please type your ID or your old name: Bao
And your new name is: BaoNguyen

Changed successfully! Your new name is BaoNguyen
```

ID	Player_Name	Score
1	BaoNguyen	0
1	BaoNguyen	50

```
Please type your ID or your old name: Mai
And your new name is: Bao

YOUR ID OR USERNAME IS NOT EXIST!!!
```

5. Delete data or drop table

```
Please type the ID or Username you want to delete
Or type MOIMOI to wipe out everything: thuyet

Successfully rip your data :)
```

```
Please type the ID or Username you want to delete
Or type MOIMOI to wipe out everything: MOIMOI

Nice move, see you later!

C:\Users\Admin\source\repos\Empty\Debug\Empty.exe (process 24424) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

II. Database script

```
1 #pragma once
2 #include <iostream>
3 #include <string>
4 #include <stdio.h>
5 #include <sqlite3.h>
6 #include <iomanip>
7 #include <Windows.h>
8
9 using namespace std;
10
11 class DataBase {
12
13     public:
14         // Pointer to SQLite connection
15         sqlite3* db;
16
17         // Save any error messages
18         char* zErrMsg = 0;
19
20         // Save the result of opening the file
21         int rc;
22
23         // Save any SQL
24         string sql;
25
26
27         // Create a callback function to print out
28         static int callback(void* NotUsed, int argc, char** argv, char** azColName) {
29             // int argc: holds the number of results
30             // (array) azColName: holds each column returned
31             // (array) argv: holds each value
32
33             for (int i = 0; i < argc; i++) {
34
35                 // Show column name, value, and newline
36                 cout << setw(20) << argv[i];
37
38             }
39             cout << endl;
40
41
42             return 0;
43         }
44
45         // Print column names manually
46         // Sorry I tried :(
47         void PrintPLcolumn()
48         {
49             cout << setw(20) << "ID" << setw(20) << "Username" << endl;
50             cout << "          -----" << endl;
51         }
52
53         void PrintPScolumn()
```

```

54     {
55     cout << setw(20) << "ID" << setw(20) << "Player_Name" << setw(20) <<
"Score" << endl;
56     cout << "
-----" <<
endl;
57     }
58
59     void CheckError() ↗
60     {
61     if (rc != SQLITE_OK) { ↗
62     fprintf(stderr, "SQL: %s\n", zErrMsg); ↗
63     sqlite3_free(zErrMsg);
64     }
65     else {
66     fprintf(stdout, "Successful\n");
67     }
68     }
69
70     void OpenDatabase() {
71     // Save the result of opening the file
72     rc = sqlite3_open("SnakeDatabase.db", &db);
73
74     if (rc) {
75     // Show an error message
76     cout << "DB Error: " << sqlite3_errmsg(db) << endl;
77     }
78
79     // Enable foreign key
80     sql = "PRAGMA foreign_keys = ON;";
81     rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
82
83     }
84
85     void CloseDatabase() {
86     sqlite3_close(db);
87     }
88
89     // Create tables if not exist
90     void CreateTables() {
91     OpenDatabase();
92
93     sql = "CREATE TABLE PlayerList (ID INTEGER PRIMARY KEY
AUTOINCREMENT, Username TEXT NOT NULL UNIQUE);";
94     rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg); 95
96     sql = "CREATE TABLE PlayerScore (Player_Name TEXT NOT NULL, Score
INTEGER NOT NULL, FOREIGN KEY(Player_name) REFERENCES PlayerList
(Username) ON UPDATE CASCADE ON DELETE CASCADE);";
97     rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
98
99     CloseDatabase(); ↗
100    }
101
102    // Populate when game start
103    void PopulatePlayerList(string name) {
104    cout << "\nMoi, " << name << ", enjoy the game!" <<endl;
105

```

```

106     OpenDatabase();
107
108     sql = "INSERT INTO PlayerList (Username) VALUES ('" + name + "')";
109     rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
110     //CheckError();
111
112     CloseDatabase();
113     }
114
115     // Populate when game over
116     void PopulatePlayerScore(string name, int score) {
117     cout << endl;
118     cout << name << " (" << score << ")" << endl;
119
120     OpenDatabase();
121
122     sql = "INSERT INTO PlayerScore (Player_Name, Score) VALUES ('" +
name
+ "', " + to_string(score) + ")";
123     rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
124     //CheckError();
125
126     CloseDatabase();
127     }
128
129     void CheckPlayerList() {
130     PrintPLcolumn();
131
132     OpenDatabase();
133
134     sql = "SELECT * FROM PlayerList;";
135     rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
136     //CheckError();
137
138     CloseDatabase();
139     }
140
141     void CheckScore() {
142     PrintPScolumn();
143
144     OpenDatabase();
145
146     sql = "SELECT PlayerList.ID, PlayerScore.Player_Name,
PlayerScore.Score FROM PlayerList, PlayerScore WHERE
PlayerList.Username = PlayerScore.Player_Name ORDER BY
PlayerList.ID, PlayerScore.Score;";
147     rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
148     //CheckError();
149
150     CloseDatabase();
151     }
152
153     void FindSpecific(string f) {
154     PrintPScolumn();
155
156     OpenDatabase();
157

```



```

158     sql = "SELECT PlayerList.ID, PlayerScore.Player_Name,
           PlayerScore.Score FROM PlayerList, PlayerScore WHERE PlayerList.ID
           =
           '" + f + "' AND PlayerList.Username = PlayerScore.Player_Name
           ORDER BY PlayerScore.Score;";
159     rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
160     //CheckError();
161
162     sql = "SELECT PlayerList.ID, PlayerScore.Player_Name,
           PlayerScore.Score FROM PlayerList, PlayerScore WHERE
           PlayerList.Username = '" + f + "' AND PlayerList.Username =
           PlayerScore.Player_Name ORDER BY PlayerScore.Score;";
163     rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
164     //CheckError();
165
166     CloseDatabase();
167 }
168
169 void UpdateName(string old, string newName) {
170     OpenDatabase();
171
172     // Priority for ID when an username is a number same as ID
173     sql = "UPDATE PlayerList SET Username = '" + newName + "' WHERE ID =
           '" + old + "';";
174     rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
175     //CheckError();
176
177     // If there's no row affected
178     if (sqlite3_changes(db) == 0)
179     {
180         sql = "UPDATE PlayerList SET Username = '" + newName + "'
           WHERE
           Username = '" + old + "';";
181         rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
182         //CheckError();
183
184         if (sqlite3_changes(db) == 0)
185         {
186             cout << "\nYOUR ID OR USERNAME IS NOT EXIST!!!" << endl;
187         }
188         else
189         {
190             cout << "\nChanged successfully! Your new name is " <<
           newName << endl;
191         }
192     }
193     else
194     {
195         cout << "\nChanged successfully! Your new name is " <<
           newName << endl;
196     }
197
198     CloseDatabase();
199 }
200
201 void DeleteData(string ans) {

```

```

202     OpenDatabase();
203
204     // Priority for ID, similar to Update
205     sql = "DELETE FROM PlayerList WHERE ID = '" + ans + "'";
206     rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
207     //CheckError();
208
209     if (sqlite3_changes(db) == 0)
210     {
211         sql = "DELETE FROM PlayerList WHERE Username = '" + ans + "'";
212         rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
213         //CheckError();
214
215         if (sqlite3_changes(db) == 0)
216         {
217             cout << "\nYOUR ID OR USERNAME IS NOT EXIST!!!" << endl;
218         }
219         else
220         {
221             cout << "\nSuccessfully rip your data :)" << endl;
222         }
223     }
224     else
225     {
226         cout << "\nSuccessfully rip your data :)" << endl;
227     }
228
229     CloseDatabase();
230 }
231
232 void DropTable() {
233     OpenDatabase();
234
235     sql = "DROP TABLE PlayerList; DROP TABLE PlayerScore;";
236     rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
237     //CheckError();
238
239     cout << "\nNice move, see you later!" << endl;
240
241     CloseDatabase();
242 }
243 };
244
245

```