

# 简易文件数据库系统

小组：茹奕笑（程序编写）

唐文杰（程序完善）

宋子涵（PPT制作）

2025.12.30

# 数据储存主要形式

## 结构体

```
typedef struct {
    char name[MAX_NAME];
    DataType type;
} Column;

typedef struct {
    char values[MAX_COLS][MAX_VALUE];
} Row;

typedef struct {
    char name[MAX_NAME];
    Column cols[MAX_COLS];
    int col_count;
    Row rows[MAX_ROWS];
    int row_count;
} Table;

typedef struct {
    char name[MAX_NAME];
    Table tables[MAX_TABLES];
    int table_count;
} Database;
```

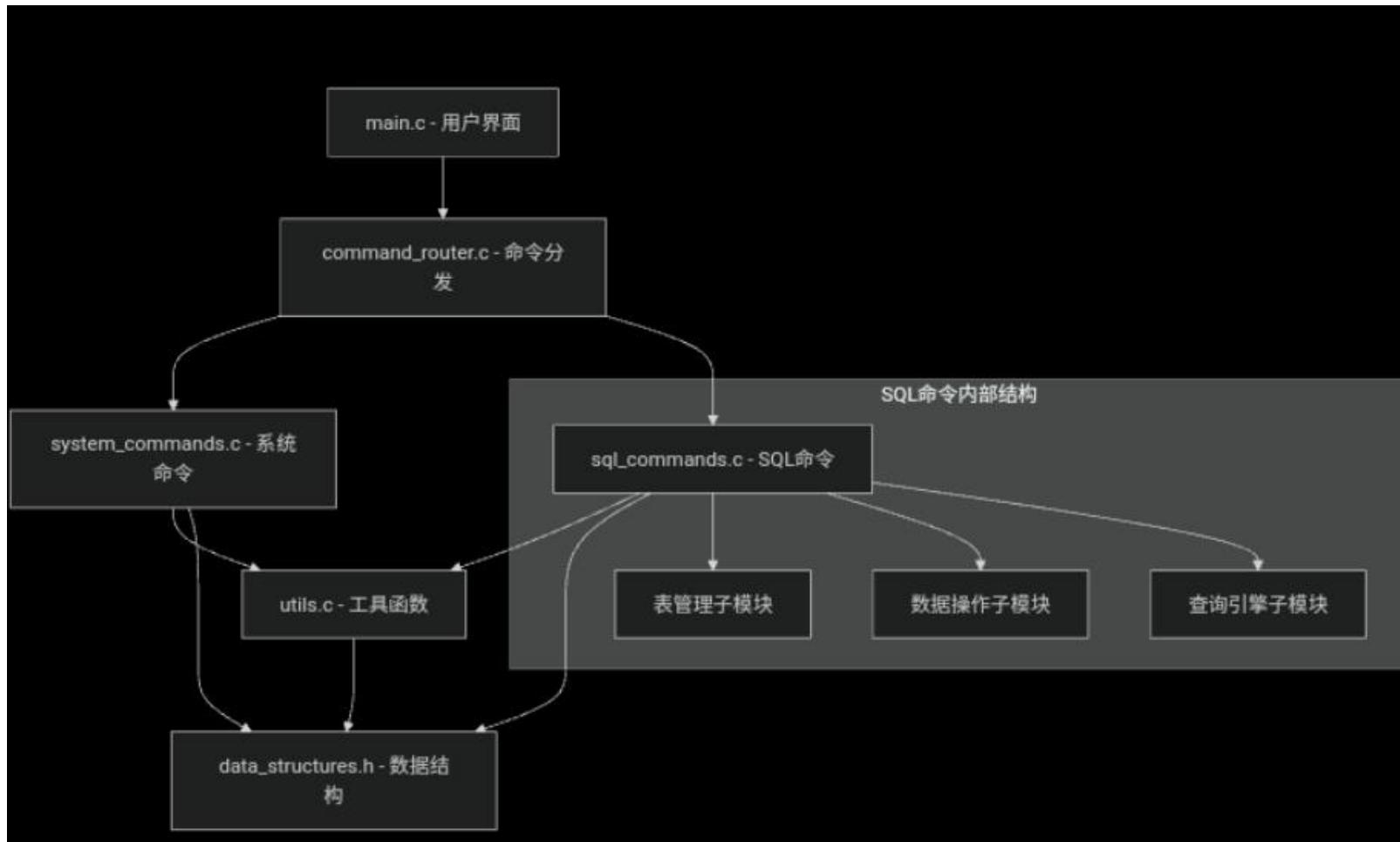
1. 内存效率：所有数据在内存中以紧凑形式存储
2. 实现简单：避免了复杂的动态内存管理
3. 文件操作简单：整个数据库可以作为一个整体保存和加载

# 模块划分

## 三、解析命令

```
void cmd_open(char *dbname);
void cmd_save(char *dbname);
void cmd_drop(char *dbname);
void cmd_tables();
void cmd_quit();
void cmd_create_table(char *input);
void cmd_drop_table(char *tablename);
void cmd_info_table(char *tablename);
void cmd_insert(char *input);
void cmd_select(char *input);
void cmd_delete(char *input);
Table* find_table(char *name);
void trim(char *str);
```

# 关键流程



# 实现功能

## 一、系统命令：

1. .open dbname – 打开/创建数据库文件
2. .save dbname – 保存数据库到文件
3. .drop dbname – 删除数据库文件
4. .tables – 列出所有表名
5. .quit – 退出系统（询问保存）
6. .help – 显示帮助信息

## 二、数据表管理：

1. create table 表名 (字段 类型,...) – 创建表
2. drop table 表名 – 删除表
3. info table 表名 – 查看表结构（字段+类型）

## 三、数据操作：

1. insert into 表名 values (值1,值2,...) – 插入数据
2. select 字段 from 表名 [where 条件] [order by 字段 asc/desc] – 查询数据
3. delete from 表名 – 删除数据（当前仅支持清空全表）

## 四、条件查询与排序

# 代码展示

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define MAX_TABLES 10
6  #define MAX_COLS 20
7  #define MAX_ROWS 1000
8  #define MAX_NAME 50
9  #define MAX_VALUE 100
10
11 typedef enum {
12     TYPE_INT,
13     TYPE_FLOAT,
14     TYPE_TEXT,
15     TYPE_DATETIME
16 } DataType;
17
18 typedef struct {
19     char name[MAX_NAME];
20     DataType type;
21 } Column;
22
23 typedef struct {
24     char values[MAX_COLS][MAX_VALUE];
25 } Row;
26
27 typedef struct {
28     char name[MAX_NAME];
29     Column cols[MAX_COLS];
30     int col_count;
31     Row rows[MAX_ROWS];
32     int row_count;
33 } Table;
34
35 typedef struct {
36     char name[MAX_NAME];
37     Table tables[MAX_TABLES];
38     int table_count;
39 } Database;
40
41 Database db;
42 int db_opened = 0;
```

```
58 int main() {
59     char input[500];
60
61     printf("简易文件数据库系统\n");
62     printf("输入 .help 查看帮助\n\n");
63
64     while (1) {
65         printf("db> ");
66         if (!fgets(input, sizeof(input), stdin)) break;
67
68         input[strcspn(input, "\n")] = 0;
69         trim(input);
70
71         if (strlen(input) == 0) continue;
72
73         if (input[0] == '.') {
74             if (strcmp(input, ".open", 6) == 0) {
75                 cmd_open(input + 6);
76             } else if (strcmp(input, ".save", 6) == 0) {
77                 cmd_save(input + 6);
78             } else if (strcmp(input, ".drop", 6) == 0) {
79                 cmd_drop(input + 6);
80             } else if (strcmp(input, ".tables") == 0) {
81                 cmd_tables();
82             } else if (strcmp(input, ".quit") == 0) {
83                 cmd_quit();
84                 break;
85             } else if (strcmp(input, ".help") == 0) {
86                 printf("系统命令: \n");
87                 printf(" .open <dbname> - 打开数据库\n");
88                 printf(" .save <dbname> - 保存数据库\n");
89                 printf(" .drop <dbname> - 删除数据库\n");
90                 printf(" .tables - 列出所有表\n");
91                 printf(" .quit - 退出\n");
92                 printf("\nSQL命令:\n");
93                 printf(" create table ... - 创建表\n");
94                 printf(" drop table ... - 删除表\n");
95                 printf(" info table ... - 显示表信息\n");
96                 printf(" insert into ... - 插入数据\n");
97                 printf(" select ... - 查询数据\n");
98                 printf(" delete from ... - 删除数据\n");
99             } else {
100                 printf("未知命令\n");
101             }
102         }
103     }
104 }
```

# 代码展示

```
102
103     } else if (strncmp(input, "create table", 12) == 0 ) {
104         cmd_create_table(input);
105     } else if (strncmp(input, "drop table", 10) == 0) {
106         cmd_drop_table(input + 11);
107     } else if (strncmp(input, "info table", 10) == 0) {
108         cmd_info_table(input + 11);
109     } else if (strncmp(input, "insert into", 11) == 0) {
110         cmd_insert(input);
111     } else if (strncmp(input, "select", 6) == 0) {
112         cmd_select(input);
113     } else if (strncmp(input, "delete from", 11) == 0) {
114         cmd_delete(input);
115     } else {
116         printf("未知SQL命令\n");
117     }
118 }
119
120 return 0;
121
122 void trim(char *str) {
123     int i, j = 0;
124     int len = strlen(str);
125
126     for (i = 0; i < len && str[i] == ' '; i++);
127
128     while (i < len) {
129         str[j++] = str[i++];
130     }
131     str[j] = '\0';
132
133     for (i = j - 1; i >= 0 && str[i] == ' '; i--) {
134         str[i] = '\0';
135     }
136 }
```

```
137
138     void cmd_open(char *dbname) {
139         trim(dbname);
140         FILE *fp = fopen(dbname, "rb");
141
142         if (fp) {
143             fread(&db, sizeof(Database), 1, fp);
144             fclose(fp);
145             printf("数据库 '%s' 以打开\n", dbname);
146         } else {
147             strcpy(db.name, dbname);
148             db.table_count = 0;
149             printf("创建新的数据库 '%s'\n", dbname);
150         }
151         db_opened = 1;
152     }
153
154     void cmd_save(char *dbname) {
155         if (!db_opened) {
156             printf("错误: 未打开数据库\n");
157             return;
158         }
159         trim(dbname);
160         FILE *fp = fopen(dbname, "wb");
161         if (fp) {
162             fwrite(&db, sizeof(Database), 1, fp);
163             fclose(fp);
164             printf("数据库已保存为 '%s'\n", dbname);
165         } else {
166             printf("错误: 无法保存数据\n");
167         }
168     }
169
170     void cmd_drop(char *dbname) {
171         trim(dbname);
172         if (remove(dbname) == 0) {
173             printf("数据库 '%s' 已删除\n", dbname);
174         } else {
175             printf("错误: 无法删除数据\n");
176         }
177     }
178 }
```

# 代码展示

```
179     void cmd_tables() {
180         if (!db_opened) {
181             printf("错误: 未打开数据库\n");
182             return;
183         }
184         printf("表列表:\n");
185         for (int i = 0; i < db.table_count; i++) {
186             printf(" %s\n", db.tables[i].name);
187         }
188     }
189
190     void cmd_quit() {
191         if (db_opened) {
192             char choice;
193             printf("是否保存数据? (y/n):");
194             scanf(" %c", &choice);
195             if (choice == 'y' || choice == 'Y') {
196                 cmd_save(db.name);
197             }
198         }
199         printf("再见! \n");
200     }
201
202     void cmd_create_table(char *input) {
203         if (!db_opened) {
204             printf("错误: 未打开数据库\n");
205             return;
206         }
207
208         char tablename[MAX_NAME];
209         char *p = strstr(input, "table") + 6;
210
211         sscanf(p, "%s", tablename);
212
213         if (find_table(tablename)) {
214             printf("错误: 表 '%s' 已存在\n", tablename);
215             return;
216         }
217
218         Table *t = &db.tables[db.table_count];
219         strcpy(t->name, tablename);
220         t->col_count = 0;
221         t->row_count = 0;
222
223         p = strchr(p, '(');
224         if (!p) {
225             printf("错误: 语法错误\n");
226             return;
227         }
228
229         p++;
230         char *end = strchr(p, ')');
231         if (!end) {
232             printf("错误: 语法错误\n");
233             return;
234         }
235         *end = '\0';
236
237         char *token = strtok(p, ",");
238         while (token && t->col_count < MAX_COLS) {
239             trim(token);
240             char colname[MAX_NAME], coltype[MAX_NAME];
241             sscanf(token, "%s %s", colname, coltype);
242
243             strcpy(t->cols[t->col_count].name, colname);
244
245             if (strcmp(coltype, "int") == 0) {
246                 t->cols[t->col_count].type = TYPE_INT;
247             } else if (strcmp(coltype, "float") == 0) {
248                 t->cols[t->col_count].type = TYPE_FLOAT;
249             } else if (strcmp(coltype, "text") == 0) {
250                 t->cols[t->col_count].type = TYPE_TEXT;
251             } else if (strcmp(coltype, "datetime") == 0) {
252                 t->cols[t->col_count].type = TYPE_DATETIME;
253             }
254
255             t->col_count++;
256             token = strtok(NULL, ",");
257
258         }
259
260         db.table_count++;
261         printf("表 '%s' 创建成功\n", tablename);
262     }
```

# 代码展示

```
263     void cmd_drop_table(char *tablename) {
264         if (!db_opened) {
265             printf("错误: 未打开数据库\n");
266             return;
267         }
268
269         trim(tablename);
270         tablename[strcspn(tablename, ";")] = 0;
271
272         for (int i = 0; i < db.table_count; i++) {
273             if (strcmp(db.tables[i].name, tablename) == 0) {
274                 for (int j = i; j < db.table_count - 1; j++) {
275                     db.tables[j] = db.tables[j + 1];
276                 }
277                 db.table_count--;
278                 printf("表 '%s' 已删除\n", tablename);
279                 return;
280             }
281         }
282         printf("错误: 表 '%s' 不存在\n", tablename);
283     }
284
285     void cmd_info_table(char *tablename) {
286         if (!db_opened) {
287             printf("错误: 未打开数据库\n");
288             return;
289         }
290
291         trim(tablename);
292         tablename[strcspn(tablename, ";")] = 0;
293
294         Table *t = find_table(tablename);
295         if (!t) {
296             printf("错误: 表 '%s' 不存在\n", tablename);
297             return;
298         }
299
300         printf("表 '%s' 信息:\n", tablename);
301         printf("字段数: %d\n", t->col_count);
302         printf("记录数: %d\n", t->row_count);
303         printf("\n字段列表:\n");
304
305         const char *type_names[] = {"int", "float", "text", "datetime"};
306         for (int i = 0; i < t->col_count; i++) {
307             printf(" %s : %s\n", t->cols[i].name, type_names[t->cols[i].type]);
308         }
309     }
```

```
311     void cmd_insert(char *input) {
312         if (!db_opened) {
313             printf("错误: 未打开数据库\n");
314             return;
315         }
316
317         char tablename[MAX_NAME];
318         char *p = strstr(input, "into") + 5;
319         sscanf(p, "%s", tablename);
320
321         Table *t = find_table(tablename);
322         if (!t) {
323             printf("错误: 表 '%s' 不存在\n", tablename);
324             return;
325         }
326
327         if (t->row_count >= MAX_ROWS) {
328             printf("错误: 表已满\n");
329             return;
330         }
331
332         p++;
333         char *end = strchr(p, ')');
334         if (!end) {
335             printf("错误: 语法错误\n");
336             return;
337         }
338         *end = '\0';
339
340         Row *row = &t->rows[t->row_count];
341         int col_idx = 0;
342 }
```

# 代码展示

```
343     while (*p && col_idx < t->col_count) {
344         trim(p);
345         if (*p == '\'') {
346             p++;
347             char *quote_end = strchr(p, '\'');
348             if (quote_end) {
349                 *quote_end = '\0';
350                 strcpy(row->values[col_idx], p);
351                 p = quote_end + 1;
352             }
353         } else {
354             char *comma = strchr(p, ',');
355             if (comma) {
356                 *comma = '\0';
357                 trim(p);
358                 strcpy(row->values[col_idx], p);
359                 p = comma + 1;
360             } else {
361                 trim(p);
362                 strcpy(row->values[col_idx], p);
363                 break;
364             }
365         }
366         col_idx++;
367         while (*p == ',' || *p == ' ') p++;
368     }
369     t->row_count++;
370     printf("插入成功\n");
371 }
372
373
374     int check_condition(Table *t, int row_idx, char *condition) {
375     if (!condition || strlen(condition) == 0) return 1;
376
377     char cond_copy[200];
378     strcpy(cond_copy, condition);
379     trim(cond_copy);
380
381     char field[MAX_NAME], op[5], value[MAX_VALUE];
382     char *p = cond_copy;
383
384 }
```

```
385     int i = 0;
386     while (*p && *p != '<' && *p != '>' && *p != '=' && *p != '!') {
387         field[i++] = *p++;
388     }
389     field[i] = '\0';
390     trim(field);
391
392     i = 0;
393     while (*p && (*p == '<' || *p == '>' || *p == '=' || *p == '!')) {
394         op[i++] = *p++;
395     }
396     op[i] = '\0';
397
398     trim(p);
399     strcpy(value, p);
400
401     if (value[0] == '\'') {
402         int len = strlen(value);
403         if (value[len-1] == '\'') {
404             value[len-1] = '\0';
405             memmove(value, value+1, len);
406         }
407     }
408
409     int col_idx = -1;
410     for (i = 0; i < t->col_count; i++) {
411         if (strcmp(t->cols[i].name, field) == 0) {
412             col_idx = i;
413             break;
414         }
415     }
416
417     if (col_idx == -1) return 0;
418     char *row_value = t->rows[row_idx].values[col_idx];
419
420     if(t->cols[col_idx].type == TYPE_INT) {
421         int v1 = atoi(row_value);
422         int v2 = atoi(value);
423
424         if (strcmp(op, "<") == 0) return v1 < v2;
425         if (strcmp(op, ">") == 0) return v1 > v2;
426         if (strcmp(op, "<=") == 0) return v1 <= v2;
427         if (strcmp(op, ">=") == 0) return v1 >= v2;
428         if (strcmp(op, "=") == 0) return v1 == v2;
429         if (strcmp(op, "!=") == 0) return v1 != v2;
430     }
```

# 代码展示

```
432     else if (t->cols[col_idx].type == TYPE_FLOAT) {
433         float v1 = atof(row_value);
434         float v2 = atof(value);
435
436         if (strcmp(op, "<") == 0) return v1 < v2;
437         if (strcmp(op, ">") == 0) return v1 > v2;
438         if (strcmp(op, "<=") == 0) return v1 <= v2;
439         if (strcmp(op, ">=") == 0) return v1 >= v2;
440         if (strcmp(op, "=") == 0) return v1 == v2;
441         if (strcmp(op, "!=") == 0) return v1 != v2;
442     }
443     else {
444         int cmp = strcmp(row_value, value);
445
446         if (strcmp(op, "<") == 0) return cmp < 0;
447         if (strcmp(op, ">") == 0) return cmp > 0;
448         if (strcmp(op, "<=") == 0) return cmp <= 0;
449         if (strcmp(op, ">=") == 0) return cmp >= 0;
450         if (strcmp(op, "=") == 0) return cmp == 0;
451         if (strcmp(op, "!=") == 0) return cmp != 0;
452     }
453
454     return 0;
455 }
456
457 int compare_rows(const void *a, const void *b, Table *t, int col_idx, int desc) {
458     Row *r1 = (Row*)a;
459     Row *r2 = (Row*)b;
460
461     int result = 0;
462
463     if (t->cols[col_idx].type == TYPE_INT) {
464         int v1 = atoi(r1->values[col_idx]);
465         int v2 = atoi(r2->values[col_idx]);
466         result = v1 - v2;
467     }
468     else if (t->cols[col_idx].type == TYPE_FLOAT) {
469         float v1 = atof(r1->values[col_idx]);
470         float v2 = atof(r2->values[col_idx]);
471         if (v1 < v2) result = -1;
472         else if (v1 > v2) result = 1;
473         else result = 0;
474     }
475     else {
476         result = strcmp(r1->values[col_idx], r2->values[col_idx]);
477     }
478 }
```

```
482     static Table *sort_table = NULL;
483     static int sort_col_idx = 0;
484     static int sort_desc = 0;
485
486     int qsort_compare(const void *a, const void *b) {
487         return compare_rows(a, b, sort_table, sort_col_idx, sort_desc);
488     }
489
490     void cmd_select(char *input) {
491         if (!db_opened) {
492             printf("错误: 未打开数据库\n");
493             return;
494         }
495
496         char tablename[MAX_NAME];
497         char condition[200] = "";
498         char order_field[MAX_NAME] = "";
499         int order_desc = 0;
500         int show_all_cols = 0;
501         char select_cols[MAX_COLS][MAX_NAME];
502         int select_col_count = 0;
503
504         char *p = input + 6;
505         trim(p);
506
507         char *from_pos = strstr(p, "from");
508         if (!from_pos) {
509             printf("错误: 语法错误, 缺少from\n");
510             return;
511         }
512
513         char fields[200];
514         int field_len = from_pos - p;
515         strncpy(fields, p, field_len);
516         fields[field_len] = '\0';
517         trim(fields);
518
519         if (strcmp(fields, "*") == 0) {
520             show_all_cols = 1;
521         } else {
522             char *token = strtok(fields, ",");
523             while (token && select_col_count < MAX_COLS) {
524                 trim(token);
525                 strcpy(select_cols[select_col_count++], token);
526                 token = strtok(NULL, ",");
527             }
528         }
529     }
530 }
```

# 代码展示

```
530     p = from_pos + 4;      569     ↓    if (show_all_cols) {      610     ↓    for (int i = 0; i < t->col_count; i++) {      611        printf("%-15s", t->cols[display_cols[i]].name);      612    }      613    }      614  } else {      615    for (int i = 0; i < select_col_count; i++) {      616      for (int j = 0; j < t->col_count; j++) {      617        if (strcmp(select_col[i], t->cols[j].name) == 0) {      618          display_cols[display_col_count] = j;      619          display_col_count++;      620          break;      621        }      622      }      623    }      624  }      625 }      626 Row temp_rows[MAX_ROWS];      627 int result_count = 0;      628 for (int i = 0; i < t->row_count; i++) {      629    if (check_condition(t, i, condition)) {      630      temp_rows[result_count++] = i;      631    }      632  }      633 result_count = temp_rows[result_count];      634 for (int i = 0; i < result_count; i++) {      635    printf("%-15s", temp_rows[i].values[display_cols[i]]);      636  }      637  printf("\n");      638 }      639 }      640 }      641 }      642 }      643 }      644 }      645 }      646 }      647 }      648 }      649 }      650 }      651 }      652 }      653 }      654 }      655 }      656 }      657 }      658 }      659 }      660 }      661 }      662 }      663 }      664 }      665 }      666 }      667 }      668 }      669 }      670 }      671 }      672 }      673 }      674 }      675 }      676 }      677 }      678 }      679 }      680 }      681 }      682 }      683 }      684 }      685 }      686 }      687 }      688 }      689 }      690 }      691 }      692 }      693 }      694 }      695 }      696 }      697 }      698 }      699 }      700 }      701 }      702 }      703 }      704 }      705 }      706 }      707 }      708 }      709 }      710 }      711 }      712 }      713 }      714 }      715 }      716 }      717 }      718 }      719 }      720 }      721 }      722 }      723 }      724 }      725 }      726 }      727 }      728 }      729 }      730 }      731 }      732 }      733 }      734 }      735 }      736 }      737 }      738 }      739 }      740 }      741 }      742 }      743 }      744 }      745 }      746 }      747 }      748 }      749 }      750 }      751 }      752 }      753 }      754 }      755 }      756 }      757 }      758 }      759 }      760 }      761 }      762 }      763 }      764 }      765 }      766 }      767 }      768 }      769 }      770 }      771 }      772 }      773 }      774 }      775 }      776 }      777 }      778 }      779 }      780 }      781 }      782 }      783 }      784 }      785 }      786 }      787 }      788 }      789 }      790 }      791 }      792 }      793 }      794 }      795 }      796 }      797 }      798 }      799 }      800 }      801 }      802 }      803 }      804 }      805 }      806 }      807 }      808 }      809 }      810 }      811 }      812 }      813 }      814 }      815 }      816 }      817 }      818 }      819 }      820 }      821 }      822 }      823 }      824 }      825 }      826 }      827 }      828 }      829 }      830 }      831 }      832 }      833 }      834 }      835 }      836 }      837 }      838 }      839 }      840 }      841 }      842 }      843 }      844 }      845 }      846 }      847 }      848 }      849 }      850 }      851 }      852 }      853 }      854 }      855 }      856 }      857 }      858 }      859 }      860 }      861 }      862 }      863 }      864 }      865 }      866 }      867 }      868 }      869 }      870 }      871 }      872 }      873 }      874 }      875 }      876 }      877 }      878 }      879 }      880 }      881 }      882 }      883 }      884 }      885 }      886 }      887 }      888 }      889 }      890 }      891 }      892 }      893 }      894 }      895 }      896 }      897 }      898 }      899 }      900 }
```

## 总结分析

本次的代码以结构体为主要形式储存数据，通过解析输入的命令以调用工具函数来对数据库文件进行相应的操作。

## 性能分析

优

劣

感谢观看