C言語によるデータベース管理プログラムの制作

東京都立南多摩中等教育学校　4年　小島 柊

動機

情報の授業内で「データベース」を扱った際、「射影」・「選択」といった関係演算について学習した。理解を深めるために、テーブルの作成と射影・選択演算を実装した簡易的なプログラムを制作した。仕様に示すコマンドは全て一から自作した。

仕様

コマンドラインから次のようなSQLコマンドを入力する。

新規テーブルの作成
`new table [テーブル名] [カラム数] [カラム名1] …`

ファイルに保存されたテーブルを開く
`open [テーブル名]`

レコードの追加
`append [フィールド1] [フィールド2] …`

レコードの削除
`delete [カラム名]==[値]`

射影　`project [カラム名] [カラム名] …`
選択　`select [カラム名]==[値]`

射影・選択のリセット　`reset`
レコードの全削除　`delete all`
データの一覧表示　`show`
プログラムの終了　`exit`

プログラムを終了するか、新しいテーブルを作成すると変更内容をファイルに保存する。
（ファイル名は「[テーブル名].shuDB」）

実行例

書籍データベース「Library」を作成し、レコードの追加・射影・選択演算を行う例

```
shuDB >> new table Library 3 Title Author Year
Table "Library" created!

shuDB >> append Bocchan Soseki_Natsume 1906
shuDB >> append Wagahai_ha_neko_dearu Soseki_Natsume 1906
shuDB >> append Cello_hiki_no_Gauthe Kenji_Miyazawa 1934
shuDB >> show

*** Library ***
+---------------+---------------+---------------+
| Title         | Author        | Year          |
+---------------+---------------+---------------+
| Bocchan       | Soseki_Natsume | 1906          |
| Wagahai_ha_n...| Soseki_Natsume | 1906          |
| Cello_hiki_n...| Kenji_Miyazawa | 1934          |
+---------------+---------------+---------------+

shuDB >> project Title Year

*** Library ***
+---------------+---------------+
| Title         | Year          |
+---------------+---------------+
| Bocchan       | 1906          |
| Wagahai_ha_n...| 1906          |
| Cello_hiki_n...| 1934          |
+---------------+---------------+

shuDB >> select Year==1906

*** Library ***
+---------------+---------------+
| Title         | Year          |
+---------------+---------------+
| Bocchan       | 1906          |
| Wagahai_ha_n...| 1906          |
+---------------+---------------+

shuDB >> exit
File saved.
exit.
```

データの追加

一覧表示

射影

選択

実装

データ構造

レコードの数は変動するため、**リスト構造**を用いた。各レコードのフィールドの値は配列で管理しているために、カラム数は変更できない。当初はカラム数も可変にするため二重のリスト構造で制作していたが、メモリ管理が複雑化したのでとりやめた。

クエリ処理

クエリを標準入力から受け取り `sscanf()` で引数を解析した。

射影・選択

テーブルの構造体に `COL_HIDE` というメンバがある。射影によるカラムの表示非表示を、**ビットフラグ** で管理している。
また、レコードの構造体に `hide` というメンバがある。選択演算によって表示対象となるかどうかの情報が格納される。
データの一覧表示の際に、これらの値を調べている。

```
if (sscanf(q, "new table "FORMAT_STR"%d %[^\n]",
name, &n, args) == 3) {…
if (sscanf(q, "open "FORMAT_STR, name) == 1) {…
if (sscanf(q, "append "FORMAT_REST, args) == 1) {…
if (sscanf(q, "delete "FORMAT_REST, args) == 1) {…
if (sscanf(q, "project "FORMAT_REST, args) == 1) {…
if (sscanf(q, "select "FORMAT_REST, args) == 1) {…
if (strcmp(q, "reset") == 0) {…
if (strcmp(q, "show") == 0) {…
if (strcmp(q, "exit") == 0) {…
```

`sscanf()`, `strcmp()` でクエリを解析

```
for (int i = 0; i < tab.COL; i++) {
    if (tab.COL_HIDE & 1<<i) {
        // 隠蔽されていた場合
        continue;
    }
    if (strlen(rec->value[i]) > 20) {…
    } else {
        printf(" %-20s|", rec->value[i]);
    }
}
```

レコードを表示する関数の一部。`i` 列目を表示するかどうかを、ビットフラグ `COL_HIDE` を用いて判定している。

```
typedef struct Rec {
    // 配列へのポインタ
    // i列目のフィールドにvalue[i]
    でアクセスできる。
    char (*value)[STRING_MAX];
    struct Rec *next;
    u_int8_t hide;
} Rec;
typedef struct {
    char name[STRING_MAX];
    uint32_t ROW;
    uint32_t COL;
    uint64_t COL_HIDE;
    Rec *colname;
    Rec *begin;
    Rec *end;

} Tab;
Tab tab;
```

`Rec`: レコードの構造体。`next` は次のレコードへのポインタ
`Tab`: テーブルの構造体。

今後の課題

あくまでデータベースの基本的な操作について理解するために製作した簡易的なものであるため、実用上の様々な問題がある。

- **データを全て文字列として扱っている**。データが肥大化するうえ、不正な値の入力を防げない。
  【解決案】テーブルの作成時、引数として各カラムのデータ型とバイト数を受け取るようにする。
  例）new table T 2 int(4) a; string(50) b;
- 選択演算の条件式として「[カラム名]==[値]」（値の一致）にしか対応していない
  【解決案】上述のようにデータ型の設定を行い、比較可能な型（整数など）については**大小比較も条件式にできる**ようにする。
- 結合演算が行えない
  【解決案】複数のテーブルをまとめてデータベースとして扱えるようにする。
- データベースに不可欠である主キーを設定できない
  【解決案】 テーブル作成時に設定できるようにする

今後、上記のような改善を行う予定である。

参考文献

リスト処理については、柴田望洋『新・明解C言語 実践編』（SBクリエイティブ, 2015）を参考にした。
SQLコマンドについては MySQL およびデータベース学習支援ツールの sAccess の記法を参考にした。

実行環境

MacBook Air (M1, 2020)　　gcc 12.0.5