Tema 1

Responsabili: PÎRLEA Cătălin-Alexandru, NICULESCU Rareș-Cosmin

Data publicare: 16.03.2020

Deadline: 1.04.2020 ora 23:59 (se acceptă teme trimise pană la data de 4.04.2020, ora 23:59 cu penalizare de 1 punct/zi din maxim 10 puncte)

1. Introducere:

O bază de date, uneori numită și bancă de date (abreviat BD), reprezintă o modalitate de stocare a unor informații și date pe un suport extern (un dispozitiv de stocare), cu posibilitatea extinderii ușoare și a regăsirii rapide a acestora.

2. Cerință:

Tema presupune să realizați un program care să simuleze funcționalitatea unei baze de date. Funcționalitățile pe care le va avea baza de date și detaliile de implementare sunt descrise in secțiunile următoare.

3. **Implementare:** structura de reprezentare a bazei de date este prezentată in Figura 1, care este compusă din:

- Structura principală (*t_db*) conține lista de tabele și numele său.

```
typedef struct db {
    char name[MAX_DB_NAME_LEN];
    t_table* tables;
} t_db;
```

- Un tabel conține numele său, tipul de date din celule, o listă de coloane, o listă de linii și o legătură către următorul tabel.

```
typedef struct table {
    char name[MAX_TABLE_NAME_LEN];
    t_cellType    type;
    t_column*     columns;
    void*         lines;  // t_intLine* | t_floatLine* | t_stringLine*
    struct table* next;
} t_table;
```

- coloană conține numele său și o legătură către următoarea coloană.

```
typedef struct column {
    char name[MAX_COLUMN_NAME_LEN];
    struct column* next;
} t_column;
```

- linie din tabel care conține o listă de celule și o legătură către următoarea linie (de exemplu linie cu valori șiruri de caractere).

```
typedef struct stringLine {
    t_stringCell*      cells;
    struct stringLine* next;
} t_stringLine;
```

- celulă din tabel, care conține valoarea sa și o legătură către următoarea celulă (de exemplu valoare: șir de caractere).

```
typedef struct stringCell {
    char*  value;
    struct stringCell* next;
} t_stringCell;
```

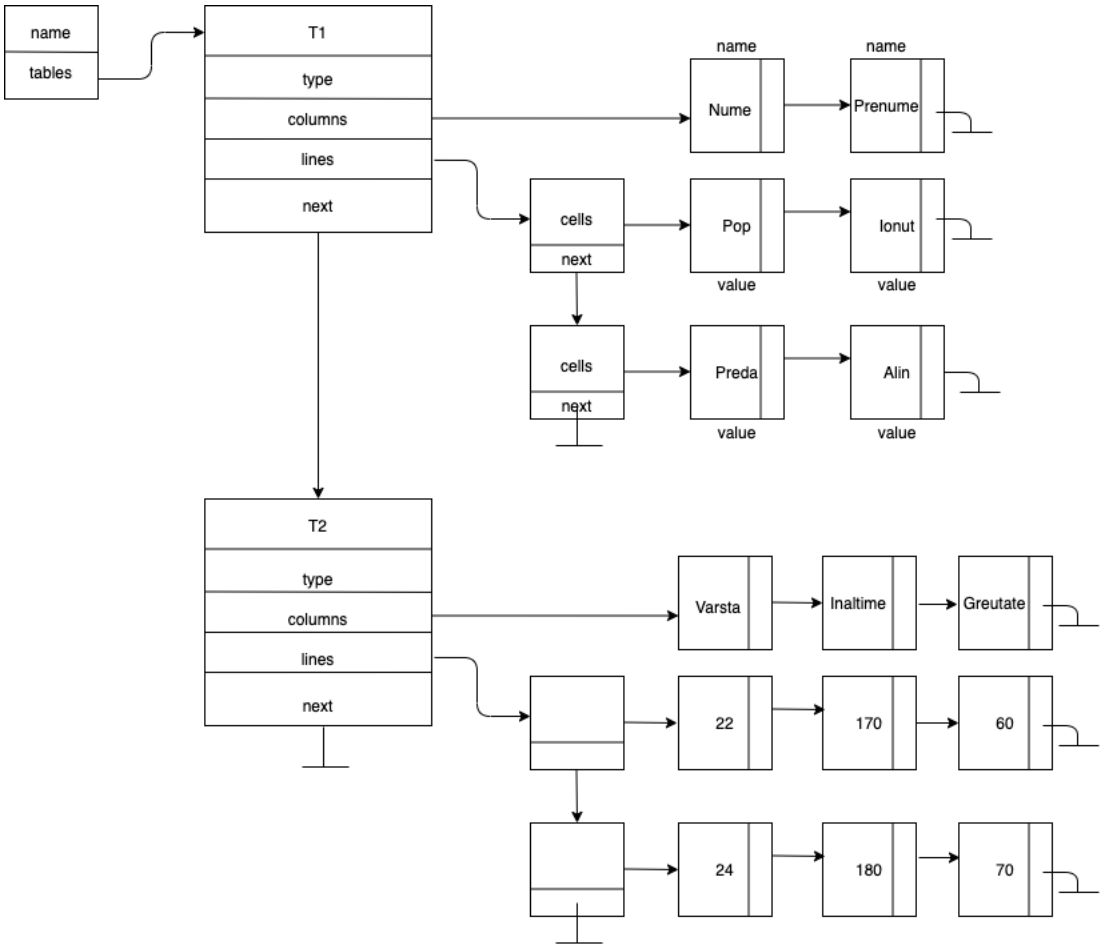

Figura 1. Structura bazei de date

4. Funcționalitate:

Se considera 3 tipuri de funcții: Data Definition, Data Query, Data Manipulation:

a. Data Definition:

INIT_DB <nume>

- inițializează baza de date
- se garantează ca va fi prima comanda

DELETE_DB

- eliberează toată memoria utilizată
- oprește execuția programului

- se garantează că va fi ultima comandă

CREATE <nume tabel> <tip de date> [<nume coloana>]

- creează un tabel cu coloanele specificate

- numele tabelului are dimensiune maximă MAX_TABLE_NAME_LEN

- tipul de date poate fi: INT/ FLOAT/ STRING (în cazul STRING, valorile pot avea orice lungime)

DELETE <nume tabel>

- șterge un tabel cu tot cu datele din el

b. Data Query:

PRINT_DB

- afișează toată baza de date

PRINT <nume tabel>

- afișează toate datele stocate în tabel

SEARCH <nume tabel> <nume coloană> <relație> <valoare>

- afișează liniile pentru care valoarea X de pe coloana specificată respectă condiția:

(X <relație> <valoare>)

- relație poate fi: “<” “<=” “==” “!=” “>=” “>”

c. Data Manipulation:

ADD <nume tabel> [<valoare pentru coloana>]

- adaugă o linie in tabel, ordinea valorilor respecta ordinea coloanelor

- se garantează că numărul de valori pentru coloane este același cu numărul de coloane

DELETE <nume tabel> <nume coloană> <relație> <valoare>

- elimină liniile pentru care valoarea X de pe coloana specificată respectă condiția:

(X <relație> <valoare>)

- relație poate fi: “<” “<=” “==” “!=” “>=” “>”

CLEAR <nume tabel>

- elimină toate liniile dintr-un tabel

5. Detalii suplimentare:

În cazul în care un tabel nu există în baza de date, va fi afișat următorul mesaj:

"Table "<nume tabel>" not found in database."

În cazul în care coloana căutată nu aparține tabelului, va fi afișat următorul mesaj:

"Table "<nume tabel>" does not contain column "<nume coloană>"."

În cazul în care, la CREATE, un tabel există deja, va fi afișat următorul mesaj:

"Table "<nume tabel>" already exists."

În cazul unei comenzi introduse greșit, va fi afișat următorul mesaj:

"Unknown command: "<comandă>"."

În cazul unui tip de date necunoscut, va fi afișat următorul mesaj:

"Unknown data type: "<tip de date>"."

Numele de tabele, coloane si bază de date au lungime maximă fixă (MAX_TABLE_NAME_LEN, MAX_COLUMN_NAME_LEN respectiv MAX_DB_NAME_LEN).

O comandă are lungimea maximă fixă (MAX_CMD_LEN).

La afișarea unui tabel, după afișarea coloanelor, va fi afișată, pentru fiecare coloana, secvența: MAX_COLUMN_NAME_LEN * '-' + ' '.

De asemenea, va fi folosit ' ' ca padding, pentru a alinia coloanele.

După afișarea titlului bazei de date sau a unui table, va fi lăsată o linie liberă.

6. Exemple

a. Inițializare bază de date, afișare și închidere program

```
INIT_DB School
PRINT_DB
DATABASE: School

DELETE_DB
```

b. Creare tabelă, adăugare linii și afișare tabelă

```
CREATE Students STRING First_Name Last_Name
ADD Students Popescu Ion
ADD Students Popescu Vasile
PRINT Students
TABLE: Students
First_Name                     Last_Name
------------------------------ ------------------------------
Popescu                        Ion
Popescu                        Vasile
```

c. Căutare în tabelă

```
SEARCH Students Last_Name == Ion
TABLE: Students
First_Name                     Last_Name
------------------------------ ------------------------------
Popescu                        Ion
```

d. Ștergere selectivă

```
DELETE Students Last_Name == Ion
PRINT Students
TABLE: Students
First_Name                     Last_Name
------------------------------ ------------------------------
Popescu                        Vasile
```

7. Notare

- **85 puncte** obținute pe testele de pe vmchecker
- **10 puncte: coding style**, codul trebuie să fie comentat, consistent și ușor de citit. De exemplu, tema nu trebuie sa conțină:
    - Warning-uri la compilare
    - linii mai lungi de 80 de caractere
    - tab-uri amestecate cu spații
    - denumire neadecvată a funcțiilor sau a variabilelor
    - folosirea incorectă de pointeri, neverificarea codurilor de eroare
    - utilizarea unor metode ce consumă resurse în mod inutil (alocare de memorie)
    - neeliberarea resurselor folosite (eliberare memoriei alocate, ștergerea fișierelor temporare, închiderea fișierelor)

    - alte situații nespecificate aici, dar considerate inadecvate
- **5 puncte: README** – va conține detalii despre implementarea temei, precum și punctajul obținut la teste (la rularea pe calculatorul propriu)
- **Bonus: 20 puncte** pentru soluțiile ce nu au memory leak-uri (bonusul se va considera numai în cazul în care a fost obținut punctajul aferent testului)
- **Temele care nu compilează, nu rulează sau obțin punctaj 0 la teste, indiferent de motive, vor primi punctaj 0**

8. Reguli de trimitere a temelor

Temele vor fi încărcate pe vmchecker (în secțiunea Structuri de Date seria CB: SD-CB), dar și pe cs.curs.pub.ro, în secțiunea destinată assignment-ului "Tema1".

Arhiva finala a temei rezolvate trebuie sa conțină:

- fișierele sursă
    - Fiecare fișier sursă creat sau modificat trebuie să înceapă cu un comentariu de forma: **/* NUME Prenume - grupa */**
- fișierul README în care va fi detaliat modul de implementare al rezolvării
- fișierul Makefile cu trei reguli (build și clean)
    - fișierul trebuie obligatoriu denumit **Makefile** și trebuie sa conțină cele 2 reguli menționate
    - Regula **build** va compila sursele și va crea executabilul numit **tema1**
    - Regula **clean** care va șterge executabilele create

Arhiva va conține numai fișierele menționate mai sus (nu se acceptă fișiere executabile sau obiect).

Dacă arhiva nu respectă aceste specificații, aceasta nu va fi acceptată la upload și implicit tema nu va fi luată în considerare.

9. Referințe

[1] https://ocw.cs.pub.ro/courses/programare/coding-style

[2] http://acs.curs.pub.ro/ - curs SD - seria CB - secțiunea Regulament SD - Reguli de realizare, verificare și trimitere a temelor