Tema 3 – ACSQL

Termen de predare

- Deadline **soft** : ~~**19 Ianuarie 2025, ora 23:55**~~ **21 Ianuarie 2025, ora 23:55**
- Deadline **hard** : **21 Ianuarie 2025, ora 23:55**

Responsabili

- Dabelea Ioana-Viviana
- Chira Vlad-Andrei

Changelog

- **19.01.2025**: Actualizare deadline soft.
- **16.01.2025**: Actualizare calcul in checker local
- **03.01.2025**: Actualizare teste task3 - calcul medie generala
- **27.12.2024**: Modificare checker coding style.
- **23.12.2024**: Stergere trailing whitespace la finalul testelor.
- **22.12.2024**: Actualizare teste 3, 4, 6, 8, 9 pentru big.db.
- **22.12.2024**: Adăugat fișiere de checker pt arm64.
- **21.12.2024, 22:00**: Corectat test 5.
- **21.12.2024, 14:15**: Corectat teste privind calculul mediei generale.
- **21.12.2024, 14:00**: Clarificare enunț privind rotunjirea mediei generale.
- **19.12.2024**: Publicare temă.

Întrebări

Dacă aveți nelămuriri, puteți să ne contactați pe forumul dedicat temei 3 [https://curs.upb.ro/2024/mod/forum/view.php?id=87692]. Nu se acceptă întrebări în ultimele 24 de ore înainte de deadline.

Citiți cu atenție tot enunțul temei.

Enunt

Secretariatul a decis să implementeze o nouă bază de date pentru a gestiona informațiile despre studenți, materiile predate și relațiile dintre acestea. Noua bază de date va fi gestionată printr-un sistem de interogare și actualizare denumit **ACSQL**.

În acest scop, s-a apelat la ajutorul vostru pentru a proiecta și implementa funcționalitățile esențiale ale sistemului.

Structuri de Date

Sistemul propus va fi modelat cu ajutorul următoarelor structuri:

1. Secretariat

Această structură reprezintă baza de date principală și centralizează:

- Lista tuturor studenților înscriși la facultate.
- Lista tuturor materiilor predate la facultate.
- Detalii despre înscrierea studenților la materii.

```
typedef struct secretariat {
    student *studenti;     // Vector de studenti
    int nr_studenti;       // Numarul total de studenti

    materie *materii;      // Vector de materii
    int nr_materii;        // Numarul total de materii

    inrolare *inrolari;    // Vector de inscrieri (relatii student-materie)
    int nr_inrolari;       // Numarul total de inscrieri
} secretariat;
```

2. Student

Structura student descrie fiecare student înscris la facultate și are următoarele câmpuri:

- *id*: Un identificator unic pentru fiecare student.
- *nume*: Numele complet al studentului.
- *an_studiu*: Anul de studiu al studentului (1, 2, 3 sau 4).
- *statut*: Tipul programului de studiu:
  - 'b' pentru buget.
  - 't' pentru taxă.
- *medie_generala*: Media generală a studentului, calculată pe baza notelor de la toate materiile.

Definiția structurii:

```
typedef struct student {
    int id;                // ID unic al studentului
    char nume[40];         // Nume complet
    int an_studiu;         // Anul de studiu
    char statut;           // 'b' (buget) sau 't' (taxă)
    float medie_generala;  // Media generală
} student;
```

3. Materie

Structura materie descrie detalii despre fiecare curs predat la facultate:

- *id*: Un identificator unic al cursului.
- *nume*: Numele cursului.
- *nume_titular*: Numele profesorului titular.

Definiția structurii:

```
typedef struct materie {
    int id;                // ID unic al materiei
    char *nume;            // Nume materie
    char *nume_titular;    // Profesor titular
} materie;
```

4. Înrolare

Structura *inrolare* descrie relația *many-to-many* dintre studenți și materii. Fiecare înrolare conține:

- *id_student*: ID-ul studentului înscris la materie.
- *id_materie*: ID-ul materiei la care este înscris studentul.
- *note*: Un vector cu trei note:
  - Nota pentru laborator și teme.
  - Nota pentru examenul parțial.
  - Nota pentru examenul final.

Suma acestor note reprezintă nota studentului la materia respectivă. De exemplu, un student cu notele 2.40, 2.00, respectiv 3.70 la o materie va avea nota 8.10/10 la materia respectivă.

Definiția structurii:

```
typedef struct inrolare {
    int id_student;        // ID-ul studentului
    int id_materie;        // ID-ul materiei
    float note[3];         // Notele studentului (laborator, parțial, final)
} inrolare;
```

Organizarea Datelor în Fișier

Pentru a asigura o structură clară și ușor de procesat, baza de date este stocată într-un fișier text organizat în secțiuni, fiecare dintre acestea delimitată de un antet.

- Un antet este un șir de caractere înconjurat de paranteze pătrate, de exemplu: [STUDENTI], [MATERII], [INROLARI].
- În continuarea antetului, datele sunt stocate pe linii separate, fiecare linie reprezentând un element (student, materie sau înrolare). Datele pe fiecare linie sunt separate prin virgule și urmează ordinea din structura aferentă.

Câmpul *medie_generala* nu este prezent în fișier. Acesta trebuie calculat pe baza notelor de la materiile la care este înscris studentul respectiv.

Media se va rotunji la două zecimale. Un student cu notele [0.51, 1.94, 2.71] la o materie, respectiv [2.80, 2.00, 0.85] la alta, va avea o medie de 5.405 ce se va rotunji la 5.41 în cadrul bazei de date.

Exemplu complet de fișier:

```
[STUDENTI]
0, Andrei Popescu, 2, b
1, Ioana Ionescu, 1, t

[MATERII]
0, PCLP, Radu Bran
1, USO, Maria Sandu

[INROLARI]
1, 1, 3.10 3.80 2.10
2, 2, 2.65 1.20 3.00
```

Task 1: Gestionarea bazei de date (10 puncte)

1.1. Incărcarea bazei de date dintr-un fișier (5 puncte)

Scrieți o funcție care citește datele dintr-un fișier organizat conform structurii descrise și le încarcă într-o structură de tip *secretariat* alocata dinamic.

```
secretariat *citeste_secretariat(const char *nume_fisier);
```

1.2. Adăugarea unui student (3 puncte)

Scrieți o funcție care adaugă un student în baza de date transmisă ca parametru.

```
void adauga_student(secretariat *s, int id, char *nume, int an_studiu, char statut, float medie_generala);
```

1.3 Eliberarea memoriei (2 puncte)

Pentru a preveni scurgerile de memorie, este necesar să implementăm o funcție care să elibereze memoria alocată dinamic pentru structura *secretariat* și elementele sale.

```
void elibereaza_secretariat(secretariat **s);
```

Task 2: Interogări, actualizări, ștergeri (70 de puncte)

2.1. Interogări (28 de puncte)

2.1.1 Interogări Simple (14 puncte)

Structura unei interogari este următoarea:

```
SELECT <campuri> FROM <tabel>;
```

1. <*campuri*> reprezintă lista de câmpuri pe care dorim să le afișăm, separate prin virgulă.
2. <*tabel*> reprezintă numele tabelei din care dorim să extragem informații (*studenti*, *materii*, *inrolari*).

Funcția de interogare va afișa, rând cu rând, valorile câmpurilor selectate din tabelul interogat.

Exemplu:

```
SELECT nume, an_studiu FROM studenti;
```

Pentru exemplul de bază de date de mai sus, programul va afișa:

```
Andrei Popescu 2
Ioana Ionescu 1
```

Operatorul * va putea fi folosit pentru a selecta toate câmpurile din tabel.

Exemplu:

```
SELECT * FROM materii;
```

Pentru exemplul de bază de date de mai sus, programul va afisa:

```
0 PCLP Radu Bran
1 USO Maria Sandu
```

Interogarea corespunzătoare vectorului *note* din structura *inrolare* va afișa toate cele 3 note, separate prin spațiu. Exemplu:

```
SELECT id_student, note FROM inrolari;
```

Pentru exemplul de mai sus se va afișa:

```
1 3.1 3.8 2.1
2 2.65 1.20 3.0
```

2.1.2 Interogări cu filtrare (7 puncte)

Structura unei interogări cu filtrare este următoarea:

```
SELECT <campuri> FROM <tabel> WHERE <camp> <operator> <valoare>;
```

- <*camp*> reprezintă numele câmpului după care se face filtrarea.
- <*operator*> reprezintă operatorul de comparație (**=**, **<**, **>**, **<=** , **>=**, **!=**).
- <*valoare*> reprezintă valoarea cu care se compară câmpul.

Exemplu:

```
SELECT nume, medie_generala FROM studenti WHERE an_studiu > 2;
```

Pentru exemplul de bază de date de mai sus, programul va afișa:

```
Andrei Popescu 8.72
```

2.1.3. Interogări cu filtrare complexă (7 puncte)

Structura unei interogări cu filtrare complexă este următoarea:

```
SELECT <campuri> FROM <tabel> WHERE <conditie1> AND <conditie2>;
```

1. <*conditie1*> și <*conditie2*> reprezintă două condiții care trebuie îndeplinite simultan pentru a selecta rândurile.

Exemplu:

```
SELECT nume, medie_generala FROM studenti WHERE an_studiu > 1 AND statut = b;
```

Pentru exemplul de bază de date de mai sus, programul va afișa:

```
Andrei Popescu 8.72
```

Pentru a putea rezolva task-ul 2.2 este necesar să implementaţi toate interogările de la task-ul 2.1.

2.2. Actualizări și Ștergeri (42 de puncte)

2.2.1. Update (21 de puncte)

Operația de tip UPDATE are următoarea structură:

```
UPDATE <tabel> SET <camp> = <valoare> WHERE <conditie>;
```

unde:

- <*tabel*> reprezintă numele tabelei care va fi actualizată (*studenti*, *materii*, *inrolari*).
- <*camp*> reprezintă numele câmpului care va fi actualizat.
- <*valoare*> reprezintă noua valoare a câmpului.
- <*conditie*> reprezintă condiția care trebuie îndeplinită pentru a actualiza rândurile.

Nu se vor face actualizari asupra campurilor de tip ID.

Exemplu:

```
UPDATE studenti SET medie_generala = 9.0 WHERE id = 0;
SELECT nume, medie_generala FROM studenti WHERE id = 0;
```

Pentru exemplul de bază de date de mai sus, programul va afișa:

```
Andrei Popescu 9.0
```

Pentru a face update la câmpul note din tabelul inrolari comanda o să arate în felul următor:

```
UPDATE inrolari SET note = 4.0 4.0 1.0 WHERE id_student = 2
```

2.2.2. Delete (14 puncte)

Operația de tip DELETE are următoarea structură:

```
DELETE FROM <tabel> WHERE <conditie>;
```

- <*tabel*> reprezintă numele tabelei care va fi actualizată (*studenti*, *materii*, *inrolari*).
- <*conditie*> reprezintă condiția care trebuie îndeplinită pentru a șterge intrarea din tabel.

Exemplu:

```
DELETE FROM studenti WHERE id = 0;
SELECT nume, medie_generala FROM studenti;
```

Pentru exemplul de bază de date de mai sus, programul va afișa:

```
Ioana Ionescu 7.98
```

Dacă toate comenzile au fost implementate, mai există un test mixt care valoreaza **7** puncte.

Task 3: Criptarea Bazei de Date (20 de puncte)

Pentru a proteja datele din baza de date a secretariatului, este necesar ca vectorul care conține studenții să fie criptat atunci când este stocat. Metoda de criptare folosită va fi o variantă simplificată de Cipher Block Chaining (CBC).

Concret, putem privi vectorul *studenti* ca fiind un vector de octeți oarecare pe care dorim sa îi criptăm. Impărțim acest vector de octeți în 4 blocuri de dimensiune egală; fiecare bloc va fi criptat pe rând, procesându-se de la primul bloc către ultimul. Criptarea fiecărui bloc se face folosind o rețea simplă de substituție și permutare (S-box și P-box).

Cipher Block Chaining (CBC)

Algoritmii de criptare lucrează în mod obișnuit cu blocuri de date de dimensiuni fixe. Din acest motiv, mesajul (în cazul de față, vectorul *studenti*) este împărțit în blocuri asupra cărora se aplică algoritmul de criptare.

Un mod de lucru simplu, numit **Electronic Codebook (ECB)**, criptează independent fiecare bloc, fără să țină cont de blocurile anterioare. Acest lucru prezintă însă un dezavantaj: blocuri identice de plaintext vor produce blocuri identice de text criptat, ceea ce poate oferi indicii nedorite privind structura datelor.

Pentru a evita acest neajuns, se folosește metoda **Cipher Block Chaining (CBC)**. În CBC, criptarea fiecărui bloc depinde de blocul criptat anterior. Primul bloc este “inițializat” folosind un vector special numit **Initialization Vector (IV)**, în timp ce fiecare bloc ulterior este procesat astfel încât conținutul său înainte de criptare să fie combinat (prin XOR) cu rezultatul criptat al blocului anterior. Astfel, două blocuri identice de plaintext nu vor mai produce aceeași ieșire criptată, deoarece criptarea lor depinde de blocul precedent.

Observatii:

- Dacă lungimea lui IV este mai mică decât dimensiunea unui bloc, atunci IV-ul se repetă până când atinge dimensiunea blocului.
- De asemenea, dacă mesajul nu se împarte perfect în 4 blocuri, se va adăuga padding (octeți 0x00) la final, astfel încât blocul final să aibă dimensiunea necesară.

Substitution and permutation networks

Rețelele de substituție și permutare (SPN) sunt structuri criptografice care aplică mai multe runde de substituții și permutări asupra datelor, pentru a obține o mai bună difuzie și confuzie. În cazul nostru, vom aplica doar două operații:

1. S-box (Substituție): Aceasta constă în aplicarea operației XOR, octet cu octet, între bloc și o cheie criptografică (*key*). Dacă *key* nu are lungimea necesară pentru a acoperi întreg blocul, se va repeta secvența de octeți din *key* până la atingerea dimensiunii blocului. Această etapă introduce confuzie, în sensul că datele rezultate sunt amestecate cu cheia criptografică.

2. P-box (Permutare): Aceasta constă într-o rearanjare a octeților din bloc pe baza unei funcții bijective. Funcția de permutare pe care o vom folosi este definită ca:

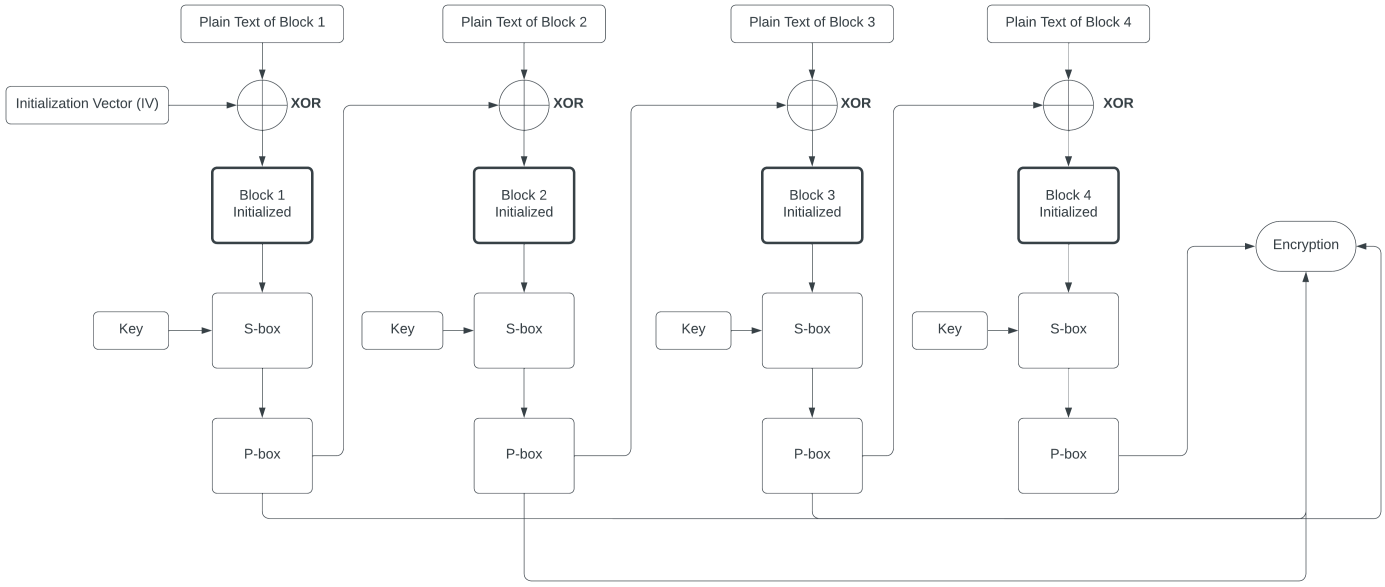
```
f(i) = (i * (n - 1) + 2)  mod  n
```

unde *i* este indicele octetului original, iar *n* este dimensiunea blocului. Octetul de la poziția *j* din blocul rezultat va fi acela aflat inițial la o poziție *i* pentru care *f(i) = j*. Cu alte cuvinte, permutarea schimbă poziția fiecărui octet, contribuind la difuzia datelor în moduri care fac criptanaliza mai dificilă.

Pseudocod

```
functie cripteaza_studenti(studenti, IV, key, cale_fisier_output)
    // Separarea datelor in 4 blocuri de dimensiune fixa
    // si adaugare padding cu 0x00 unde este necesar
    blocks = split_into_blocks_and_pad(studenti)

    // Criptarea primului bloc folosind IV-ul
    // XOR intre primul bloc si Initialization Vector (extins la dimensiunea blocului)
    blocks[0]_enc = XOR(blocks[0], IV)

    // Aplicare S-box
    blocks[0]_enc = S_BOX(blocks[0]_enc, key)

    // Aplicare P-box
    blocks[0]_enc = P_BOX(blocks[0]_enc)

    // Criptarea celor 3 blocuri ramase
    for i in [1, 3]
        // XOR intre blocul curent si blocul precedent deja criptat
        blocks[i]_enc = XOR(blocks[i], blocks[i - 1]_enc)

        // S-box
        blocks[i]_enc = S_BOX(blocks[i]_enc, key)

        // P-box
        blocks[i]_enc = P_BOX(blocks[i]_enc)

    // Scrierea blocurilor criptate in fisier_output
    // ...
```

Diagrama procesului de criptare



Exemplu

Să presupunem ca avem de criptat mesajul *Ana are mere!* folosind cheia *pclp1* si IV *ab*.

Începem prin a converti mesajul in octeți:

```
 0x41 0x6E 0x61 0x20 0x61 0x72 0x65 0x20 0x6D 0x65 0x72 0x65 0x21
```

Împărțim mesajul în 4 blocuri și adaugăm padding la ultimul bloc:

- Blocul 1: *0x41 0x6E 0x61 0x20*
- Blocul 2: *0x61 0x72 0x65 0x20*
- Blocul 3: *0x6D 0x65 0x72 0x65*
- Blocul 4: *0x21 0x00 0x00 0x00*

Transformăm IV-ul în octeți: *“ab” –> 0x61 0x62*

Transformăm cheia în octeți: *“pclp1” –> 0x70 0x63 0x6C 0x70 0x31*

Blocul 1

Aplicăm XOR între blocul 1 și IV:

```
0x41 0x6E 0x61 0x20   ^
0x61 0x62 0x61 0x62
-------------------
0x20 0x0C 0x00 0x42
```

Aplicăm S-box:

```
0x20 0x0C 0x00 0x42  ^
0x70 0x63 0x6C 0x70
-------------------
0x50 0x6F 0x6C 0x32
```

Aplicăm P-box:

Octetul de pe poziția 0 se duce pe poziția *(0 * 3 + 2) mod 4 = 2*

Octetul de pe poziția 1 se duce pe poziția *(1 * 3 + 2) mod 4 = 1* …

Obținem: *0x6C 0x6F 0x50 0x32*

Blocul 2

Aplicăm XOR între blocul 1 criptat și blocul 2:

```
0x6C 0x6F 0x50 0x32   ^
0x61 0x72 0x65 0x20
-------------------
0x0D 0x1D 0x35 0x12
```

Aplicăm S-box …

Aplicăm P-box …

Se repetă același procedeu pentru celelalte 2 blocuri rămase, iar rezultatul se obține prin concatenarea celor 4 blocuri criptate.

Schelet de cod

Pentru această temă trebuie să porniți de la scheletul de cod de aici: tema3.zip.

Pentru task-urile 1 și 3, este suficient să implementați funcțiile din fișierele corespunzătoare.

Pentru task-ul 2, baza de date va fi citită dintr-un fișier primit ca argument în linia de comandă, iar comenzile vor fi introduse de la tastatură în următorul format:

```
numărul_de_comenzi
comanda1
comanda2
...
```

Rezultatele comenzilor vor fi afișate la tastatura.

Scheletul include instrumente utile pentru a compila și valida tema. În fișierul `USAGE.md` regăsiți instrucțiunile de utilizare a instrumentelor.

Task-urile se rezolvă în fişierele corespunzătoare din schelet. Este recomandat să vă mai faceţi si alte fişiere cu funcţii auxiliare.

Atenție! Nu aveți voie să:

- redenumiți fișierele temei
- modificați parametrii funcțiilor din schelet

Notare

Valgrind

Există o depunctare de până la **-10p** pentru scurgeri de memorie, verificarea făcându-se cu valgrind [https://stackoverflow.com/questions/5134891/how-do-i-use-valgrind-to-find-memory-leaks].

Coding Style

Există o depunctare de până la **-20p** pentru coding style inadecvat. Checkerul verifică automat coding style-ul.

Validare Locală

Checkerul local se regăsește în scheletul temei. Consultați fișierul **USAGE.md** pentru instrucțiunile de utilizare.

Trimitere

Tema va fi trimisă prin Moodle, cursul **Programarea Calculatoarelor (CB & CD)**, activitatea **Tema 3**.

Toate temele sunt testate în mod automat pe Moodle.

Arhiva temei se va încărca prin formularul de submisie (butonul **Add submission**).

Rezultatele vor fi disponibile în secțiunea **Feedback** – nota apare la linia **Grade**, iar outputul checkerului și erorile apar la sectiunea **Feedback comments**. Dacă apare un buton albastru în formă de plus, trebuie să dați click pe el pentru a afișa întregul output al checkerului.

Citiți cu atenție informațiile afișate în **Feedback** pentru a vă asigura că tema a fost rulată cu succes. Asigurați-vă că conținutul arhivei respectă structura dorită (ex. fișierele sunt în directorul corect).

Arhiva se obtine in urma rulari scriptului **archive.sh**.

Arhiva trebuie să fie în format ZIP.

Nu includeți fișierele checkerului în arhivă.

Depunctări

Lista nu este exhaustivă.

- O temă care nu compilează și nu a rulat pe Moodle nu va fi luată în considerare.
- O temă care nu rezolvă cerința și trece testele prin alte mijloace nu va fi luată în considerare.
- **Nu acceptăm teme copiate.** În cazul unei teme copiate se scade punctajul aferent temei din punctajul total.
- **(-20p)** Nerezolvarea tuturor erorilor și warningurilor de coding style.
- **(-10p)** Nerezolvarea tuturor erorilor de la valgrind.
- **(-5p)** Lipsa README.md

programare/teme_2024/tema3_2024_cbd.txt · Last modified: 2025/01/19 13:16 by ioana.dabelea