λPP Paradigme de Programare

Racket: Mini Database

- Responsabil: Mihai Nan [mailto:mihai.nan.cti@gmail.com]
- Deadline soft: **05.04.2018**, apoi depunctare 0.5p/zi
- Deadline hard: 13.04.2018 ora 23.59
- Data publicării: 20.03.2018
- Data ultimei modificări: 28.03.2018 changelog
- Tema se va încărca pe **vmchecker**
- Data tester-ului: 28.03.2018
- Forum tema 1 [http://cs.curs.pub.ro/2017/mod/forum/view.php?id=5436]
- vmchecker [https://vmchecker.cs.pub.ro/ui/#PP]

Descriere

Tema va avea ca scop dezvoltarea unui sistem minimalist de management al unei baze de date, folosind paradigma funcțională. Acest sistem va permite crearea și ștergerea tabelelor, precum și efectuarea operațiilor elementare asupra acestora: **insert**, **select**, **update** și **delete**.

O **bază de date** reprezintă o modalitate de reprezentare și stocare a datelor în vederea extragerii facile a informațiilor și realizarea prelucrărilor ulterioare.

O **tabelă** reprezintă un ansamblu de date organizate pe anumite criterii de funcționalitate.

Un **atribut** este o proprietate ce descrie o anumită caracteristică a unei tabele.

- **Exemplul 1**

Tabela `Studenți`

Număr matricol	Nume	Prenume	Grupă	Medie
123	Ionescu	Gigel	321CA	9.82
124	Popescu	Maria	321CB	9.91
125	Popa	Ionel	321CC	9.99
126	Georgescu	Ioana	321CD	9.87

În cazul acestui exemplu, atributele tabelei **`Studenți`** sunt următoarele: **`Număr matricol`**, **`Nume`**, **`Prenume`**, **`Grupă`** și **`Medie`**.

Se dorește implementarea unui program care:

- să îi permită utilizatorului să creeze și să șteargă tabele;
- să insereze, să extragă, să actualizeze și să șteargă informații;
- pentru bonus, să extragă și să combine rezultatele din mai multe tabele.

Observație: **Reprezentarea internă** pentru tabele și baza de date va fi la alegere, însă va trebui să implementați conform specificațiilor setul de funcții de control.

Cerințe

Definirea elementelor de control - 20 de puncte

În continuare, se va descriere funcționalitatea fiecărei funcții.

- `init-database` – inițializează o bază de date vidă;
- `create-table` – primește ca argumente numele tabelei și numele coloanelor care o definesc și returnează o tabelă, fără intrări, având aceste informații.
- `get-name` – primește ca argument o tabelă și întoarce numele acesteia;
- `get-columns` – primește ca argument o tabelă și întoarce numele coloanelor acesteia;
- `get-tables` – primește ca parametru o bază de date și întoarce o listă cu tabelele din respectiva bază de date;
- `get-table` – primește ca argumente o bază de date și un nume de tabelă și returnează tabela din baza de date cu numele indicat;
- `add-table` – primește ca argumente o bază de date și un tabel pe care îl inserează în acea bază de date, returnând rezultatul (o bază de date);
- `remove-table` – primește ca argumente o bază de date și numele tabelei pe care urmează să o șteargă din baza de date, returnând rezultatul (o bază de date).

Operația insert - 10 puncte

Pentru a adăuga o intrare într-o tabelă se va folosi operația `insert`.

Forma generală a acestei operații este următoarea:

```
(insert db table-name record)
```

- `db` – baza de date în care căutăm tabela în care vom insera;
- `table-name` – numele tabelei în care urmează să inserăm informația;
- `record` – o listă de perechi ce va conține informația corespunzătoare liniei care urmează a fi inserată în tabelă. Primul element al perechii reprezintă numele coloanei, iar al doilea valoarea pe care o va avea coloana respectivă.

Observație: Nu este obligatoriu să se specifice valori pentru toate coloanele tabelei. În cazul în care vor exista coloane pentru care nu sunt specificate valori, acestea vor rămâne cu valoarea implicită, `NULL`.

Observație: Pentru orice operație modificatoare pe tabel, **NU** puteți folosi funcția `equal?` pe tabele întregi.

Important: Înregistrarea se va insera la finalul tabelei.

Operația select - 40 de puncte

Pentru a extrage informații dintr-o tabelă se va folosi operația `select` care, în cadrul acestei implementări, va avea două forme: `simple-select` și `select`.

Operația simple-select - 10 puncte

Această operație este folosită pentru a extrage informațiile din toate înregistrările unei tabele pentru anumite coloane.

Sintaxa acestei instrucțiuni este următoarea:

```
(simple-select db table-name columns)
```

- `db` – baza de date în care se află tabela din care se vor extrage informațiile;
- `table-name` – numele tabelei din care sunt selectate înregistrările;
- `columns` – o listă ce va conține numele coloanelor pentru care se dorește extragerea informației din tabelă.

Rezultatul acestei operații este o listă care conține valorile extrase pentru fiecare coloană indicată. Pentru fiecare coloană rezultatele sunt reținute într-o listă. În concluzie, rezultatul este o listă care conține n liste, unde n reprezintă numărul de coloane din lista `columns`.

Observație: Lista rezultat va conține listele cu informațiile pentru fiecare coloană exact în ordinea în care apar numele coloanelor în lista `columns`. Ordinea valorilor din fiecare listă o să fie cea din tabelă.

Operația select - 30 de puncte

Această operație are același efect ca `simple-select` însă permite adăugarea unor condiții, urmând să extragă din tabelă doar înregistrările care îndeplinesc toate condițiile specificate. Operația select va avea următoarea sintaxă:

```
(select db table-name columns conditions)
```

- `db` – reprezintă baza de date în care se găsește tabela din care sunt extrase informațiile;
- `table-name` – reprezintă numele tabelei din care vor fi selectate înregistrările;
- `columns` – este o listă care poate conține un nume de coloană sau o pereche formată dintr-o operație și un nume de coloană;
- `conditions` – este o listă care va conțin condițiile pe care trebuie să le îndeplinească înregistrările selectate.

Observație: Operațiile posibile care ar putea să apară în lista `columns` sunt următoarele:

- `min` – va păstra elementul minim ca valoare din coloana respectivă;
- `max` – va păstra elementul maxim ca valoare din coloana respectivă;
- `count` – va determina numărul de valori din coloana respectivă;
- `sum` – va determina suma elementelor din coloana respectivă;
- `avg` – va determina media aritmetică a elementelor din coloana respectivă;
- `sort-asc` – va sorta crescător elementele din coloana respectivă;
- `sort-desc` – va sorta descrescător elementele din coloana respectivă.

În cazul operațiilor `min`, `max`, `count`, `sum` și `avg` lista din rezultat pentru coloana pe care se aplică această comandă va avea un unic element, reprezentând rezultatul operației.

Important: Pentru a simplifica lucrurile, va trebui să aplicați aceste operații pe valorile returnate de `select` pentru coloana indicată.

Operațiile de sortare (`sort-asc`, `sort-desc`) vor fi aplicate pe lista cu valorile pentru coloana specificată, iar lista din rezultat pentru coloana respectivă o să fie sortată. În cazul coloanelor pentru care nu este aplicată sortarea, valorile o să apară în lista rezultat în ordinea în care ele au fost inserate în tabelă.

Important: Fiecare operație este dată ca literal.

Observație: O condiție este o listă de forma:

```
(list comparator column-name value)
```

unde comparator poate lua una din următoarele valori:

```
<, >, <=, >=, equal?
```

Important: Fiecare comparator este dat ca funcție.

Observație: Indiferent de condiție, valoarea `NULL` nu o va îndeplini.

Important: Nu este nevoie ca numele coloanei din condiție să fie una din coloanele pentru care comanda `select` extrage informațiile.

Operația update - 20 de puncte

Modificarea uneia sau a mai multor înregistrări dintr-o tabelă se realizează folosind instrucțiunea `update` a cărei formă generală este următoarea:

```
(update db table-name values conditions)
```

- `db` – baza de date în care se găsește tabela care urmează a fi modificată;
- `table-name` – numele tabelei în care se vor modifica înregistrările;
- `values` – o listă de perechi (Nume coloană / valoare), conținând valorile care se vor scrie în coloanele indicate pentru înregistrările care îndeplinesc condițiile;
- `conditions` - o listă de condiții pe care trebuie să le îndeplinească o înregistrare pentru a putea fi modificată. Condițiile au aceeași formă și semnificație ca la operația `select`.

Operația delete - 10 puncte

Ștergerea uneia sau a mai multor linii dintr-o tabelă se va face utilizând operația `delete` a cărei sintaxă este următoarea:

```
(delete db table-name conditions)
```

- `db` – baza de date în care se află tabelă din care se vor șterge înregistrări;
- `table-name` – numele tabelei care urmează a fi modificată;
- `conditions` – o listă de condiții care ar trebui să fie îndeplinite de o înregistrare pentru a putea fi ștearsă din tabelă. Condițiile au aceeași formă și semnificație ca la operația `select`.

Observație: Dacă lista `conditions` este vidă, se vor șterge toate liniile din tabelă, însă structura tabelei rămâne (se va șterge doar conținutul tabelei, nu și tabela propriu-zisă).

Bonus - Natural Join - 20 de puncte

În exemplele și comenzile prezentate de până acum, am realizat operații pe o singură tabelă. Atunci când dorim să extragem informații din două sau mai multe tabele avem nevoie să folosim un **join**. Pentru a se face un **join** între n tabele, avem nevoie de minimum $n - 1$ condiții de **join**.

În cadrul temei, voi va trebui să implementați **natural join** care este unul dintre cele mai simple tipuri de **join**. Acest tip de **join** se bazează pe toate coloanele din tabele care au același nume și selectează din aceste tabele rândurile care au valori egale în toate coloanele care s-au potrivit.

Sintaxa acestei comenzi o să semene foarte mult cu cea a operației `select`.

```
(natural-join db tables columns conditions)
```

- `db` – reprezintă baza de date în care se găsesc tabelele;
- `tables` – o listă care va conține numele tabelelor din care urmează să fie extrase informațiile (se garantează că va exista cel puțin o coloană comună între toate aceste tabele);
- `columns` și `conditions` – vor avea aceeași semnificație ca la `select`.

Observație: Pentru a simplifica implementarea operației, se va testa această funcție doar pentru două tabele din baza de date.

Exemple

Operația `insert`

```
(insert db "Studenți" (list '("Nume" . "Ene")
                            '("Prenume" . "Alina")
                            '("Număr matricol" . 132)
                            '("Grupa" . "322CA")
                            '("Medie" . 9.52)))
```

Rezultatul acestei comenzi, considerând **`db`** o baza de date care conține tabela **`Studenți`** din **Exemplul 1**, este următorul:

Număr matricol	Nume	Prenume	Grupă	Medie
123	Ionescu	Gigel	321CA	9.82
124	Popescu	Maria	321CB	9.91
125	Popa	Ionel	321CC	9.99
126	Georgescu	Ioana	321CD	9.87
132	Ene	Alina	322CA	9.52

Observație: Dacă operația **`insert`** nu va primi valori pentru toate atributele tabelei, atunci se va insera valoarea **`NULL`** în cele lipsă.

Dacă nu ar fi fost specificată grupa, atunci rezultatul ar fi fost următorul:

```
(insert db "Studenți" (list '("Nume" . "Ene")
                            '("Prenume" . "Alina")
                            '("Număr matricol" . "132")
                            '("Medie" . 9.52)))
```

Număr matricol	Nume	Prenume	Grupă	Medie
123	Ionescu	Gigel	321CA	9.82
124	Popescu	Maria	321CB	9.91
125	Popa	Ionel	321CC	9.99
126	Georgescu	Ioana	321CD	9.87
132	Ene	Alina	**NULL**	9.52

Operația `simple-select`

```
(simple-select db "Studenți" '("Nume" "Prenume"))
```

Rezultatul acestei comenzi, considerând **db** o baza de date care conține tabela **Studenți** din **Exemplul 1**, este următorul:

Ionescu	Gigel
Popescu	Maria
Popa	Ionel
Georgescu	Ioana

```
'(("Ionescu" "Popescu" "Popa" "Georgescu")
  ("Gigel" "Maria" "Ionel" "Ioana"))
```

Operația `delete`

```
(delete db "Studenți" (list (list < "Medie" 9.9)))
```

Rezultatul acestei comenzi, considerând **db** o baza de date care conține tabela **Studenți** din **Exemplul 1**, este următorul:

Număr matricol	Nume	Prenume	Grupă	Medie
124	Popescu	Maria	321CB	9.91
125	Popa	Ionel	321CC	9.99

```
(delete db "Studenți" (list (list <= "Medie" 9.85)
                            (list < "Număr matricol" 125)))
```

Rezultatul acestei comenzi, considerând **db** o baza de date care conține tabela **Studenți** din **Exemplul 1**, este următorul:

Număr matricol	Nume	Prenume	Grupă	Medie
124	Popescu	Maria	321CB	9.91
125	Popa	Ionel	321CC	9.99
126	Georgescu	Ioana	321CD	9.87

Operația `select`

```
(select db "Studenți" (list "Nume" (cons 'min "Medie"))
                      (list (list >= "Medie" 9.9)))
```

Rezultatul acestei comenzi, considerând **db** o baza de date care conține tabela **Studenți** din **Exemplul 1**, este următorul:

Nume	Medie
Popescu	9.91
Popa	

```
'(("Popescu" "Popa")
  9.91)
```

Natural join

Pentru acest exemplu, vom folosi următoarele tabele.

Tabela `Category`

CATEGORY_ID	CATEGORY_NAME
1	Mobiles
2	Laptops
3	Laptops
4	Cameras
5	Gaming

Tabela `Product`

CATEGORY_ID	PRODUCT_NAME
1	Nokia
1	Samsung
2	HP
2	Dell
3	Apple
4	Nikon
Null	Playstation

Comanda este următoarea:

```
(natural-join db '("Category" "Product") '("CATEGORY_ID" "PRODUCT_NAME" "CATEGORY_NAME")
              (list (list >= "CATEGORY_ID" 2)))
```

Rezultatul este acesta:

CATEGORY_ID	PRODUCT_NAME	CATEGORY_NAME
2	HP	Laptops
2	Dell	Laptops
3	Apple	Tablet
4	Nikon	Cameras

Precizări

- În arhiva temei includeți un fișier `README` cu detalii despre cum ați implementat reprezentarea bazei de date.
- Sunt interzise efectele laterale de orice fel (`set!`, `set-car!` etc.).

- Este indicată utilizarea funcționalelor. Folosirea adecvată a acestora sau nefolosirea acestora aduc modificări în punctajul temei (în limita a 1 punct).
- Se va lucra exclusiv în fișierul `mini-database.rkt`. Elementele de implementat sunt clar indicate.
- Operația `count` va determina numărul de elemente unice dintr-o coloană.
- Operațiile de tip `select` vor întoarce lista vidă dacă sunt apelate pentru o tabelă goală sau dacă nicio înregistrare din tabelă nu îndeplinește condițiile specificate.
- Pentru operațiile `min`, `max`, `count`, `sum`, `avg` rezultatele vor fi adăugate direct în lista rezultat, fără a fi incluse într-o altă listă.
- Pentru cerința bonus, se garantează că tabelele pentru care se ralizează operația de tip `natural-join` au o singură coloană comună.
- În cazul tabelei `Studenți`, coloanele **Nume**, **Prenume**, **Grupă** au valori de tip string (ex. „Ionescu“), iar coloanele **Număr matricol** și **Medie** au valori numerice (ex. `123` / `9.82`).
- În cazul tabelei `Cursuri`, coloanele **Anul**, **Semestrul** și **Disciplină** au valori de tip string (ex. „I“), iar coloanele **Număr credite** și **Număr teme** au valori numerice.

Arhiva de pornire

Arhiva de pornire

Arhiva de pornire + checker

Changelog

1. 20.03.2018 – am specificat tipul pentru operațiile din lista `columns` și pentru comparatorii din lista `conditions`;
2. 23.03.2018 – am adăugat câteva Precizări și am detaliat rezultatele unor operații;
3. 27.03.2018 – am adăugat arhiva cu checker-ul temei și am corectat greșelile semnalate pe forum pentru exemplu;
4. 28.03.2018 – am adăugat câteva Precizări și am actualizat exemplele;
5. 30.03.2018 – am corectat cele trei teste (`simple-select-&-insert3`, `simple-select-&-insert4`, `simple-select-&-insert5`), eliminând caracterul `'` și corectând output-ul.
