Filip Živković

Implementacija alata za rad sa bazama podataka u Matlabu

*Cilj rada je implementacija rutinâ za manipulisanje bazama podataka u Matlabu, preciznije RDBMS (*Relational database management system*) rutina, gde su svi podaci smešteni u tabele a relacije između tabela su takođe, u nekoj formi, smeštene u tabele. Pošto su rutine implementirane kao biblioteka funkcija, Matlab je istovremeno i upitni jezik. Pošto se upiti (i sve ostale operacije) daju u formi Matlab funkcija, moguće je praktično beskonačno ugnežđivanje upita. Time ovaj projekat povezuje baze podataka i Matlab, pošto se bazama manipuliše koristeći Matlab funkcije, Matlab sintaksu, Matlab strukture i uopšteno, Matlab „način razmišljanja“.*

Uvod

Matlab je program (i programski jezik) prvenstveno namenjen numeričkim izračunavanjima (postoji paket za Matlab koji u njega implementira funkcije programa Maple, namenjenog simboličkim izračunavanjima). Matlab koristi matrice kao osnovni tip podataka, a svi ostali se dobijaju kao specijalni slučajevi (brojevi su nuladimenzionalne matrice, nizovi, odnosno vektori, jednodimenzionalne). Matlab je napravljen tako da vrlo brzo izvršava operacije nad matricama, pa mu je za neke složene operacije (množenje i deljenje matrica, zamena redosleda kolona u matrici) potrebno znatno manje vremena nego u drugim programskim jezicima.

Sa razvojem baza podataka počelo se sedamdesetih godina dvadesetog veka i od tada se baze podataka prilagođavaju novim potrebama i evoluiraju. Ovaj rad odnosi se na podoblast relacionih baza podataka koje se oslanjaju na model objekata i veza, gde su baze predstavljene kao niz tabela koje mogu biti povezane relacijama. U bazama podataka se skladišti ogroman broj podataka kojima se manipuliše upotrebom različitih za to namenjenih alata (upitni jezici, razni programi). Standardni alat za manipulaciju bazama podataka kada su u pitanju skript jezici je SQL (*Structured Querry Language*). SQL se zasniva na relacionoj algebri, a uređen je da liči na engleski jezik i samim tim bude jednostavan (kasnije, evolucijom mogućnosti RDBMS – *Relational database management system*, došlo je do toga da postane komplikovan).

Relaciona algebra je (matematički) osnov na kome se zasnivaju operacije koje se vrše nad bazom podataka. Namenjena je prvenstveno operacijama nad relacijama, ali pošto se svaka tabela može predstaviti u vidu relacije možemo je koristiti i pri izvođenju operacija nad tabelama, a samim tim i nad bazama podataka.

Realizacija

Sistem koji je razvijen u okviru ovog projekta vidi bazu podataka kao skup tabela. Namena sistema je da se omogući kompletno kreiranje baze podataka (kreiranje tabela, dodavanje vrednosti u tabelu), održavanje baze podataka (izmena podataka, dodavanje novih podataka, brisanje zastarelih unosa), korišćenje podataka unetih u bazu, njihovo filtriranje (selekcija, projekcija, spajanje po kriterijumu).

Tabele su potpuno nezavisne jedna od druge i potpuno su ravnopravne sa ostalim promenljivima koje korisnik koristi. Mogu biti memorisane u više promenljivih ili u jedan niz tabela.

Fizički, sistem pamti tabelu kao strukturu koja se sastoji od pet polja. Prvo polje služi za memorisanje vrednosti koje se nalaze u tabeli, drugo pamti imena kolona, treće domene svake kolone, četvrto pamti u kojim kolonama se nalazi primarni ključ. Peto polje služi za razlikovanje tabele od pogleda. Implementacija pogleda je neophodna (detaljnije u odeljku o implementiranim funkcijama). Pogled se

Filip Živković (1991), Beograd, Mileševska 12/17, učenik 3. razreda Računarske gimnazije iz Beograda

razlikuje od tabele po tome što se kod njega ne održava integritet tabele. Poželjno je da se radi sa tabelama, a da se pogledi koriste samo kad je neophodno (npr. kad treba prikazati neke vrednosti bez kolona sa primarnim ključem). U daljem tekstu pojam tabela odnosiće se i na poglede i na tabele, pošto većina skripti ne pravi razliku, a gde ima razlike biće naglašeno.

Implementirane funkcije

Implementirane su sledeće funkcije:

1. Kreiranje novih tabela

Sintaksa:

<imeNoveTabele>=napraviTabelu(<imenaKolona>, <vrednostiDomena>, <imenaPKova>)

ili

<imeNoveTabele>=napraviTabelu_poIndexu(<imenaKolona>, <vrednostiDomena>, <vektorPK>)

Funkcija koja kreira novu tabelu kao argumente uzima imena kolona, domene za svaku od kolona i imena kolona koje čine primarni ključ (dozvoljen je kompozitni primarni ključ). Dozvoljeni domeni su skup celih brojeva (*int*), skup realnih brojeva (*double*) i jedan domen za tekstualne podatke od jednog ili više slova (*char*). Postoji i pomoćna funkcija koja se od glavne razlikuje samo po trećem parametru – njoj se umesto imena kolona prosleđuje nula-jedan vektor koji ima jedinice na mestima gde je predviđeno da bude primarni ključ. Glavna funkcija formira taj vektor i poziva pomoćnu. Na ovaj način se težilo jednostavnosti korišćenja: kako korisnik ne bi bio primoran da pamti indekse kolona, on kolone referencira po njihovom simboličkom nazivu.

Primer:

radnici=napraviTabelu({'ID','Ime','Prezime','JMBG','Adresa','Plata','GodineRS},{'int','char','char','int','char','double','double'},{'ID'});

(Ova rečenica bi napravila tabelu sa kolonama ID(primarni ključ, domen int), Ime(domen char), Prezime(char), JMBG(int), Adresa(char), Plata(double) i GodineRS(double))

2. Dodavanje novih vrednosti u tabelu

Sintaksa:

<imeTabele>=dodaj(<imeTabele>, {<vrednost1>, <vrednost2>, … <vrednostN>})

<imeTabele>=dodaj_u_pogled(<imeTabele>, {<vrednost1>, <vrednost2>, … <vrednostN>})

Funkcija kojom se dodaju nove vrednosti u tabelu ima za argumente tabelu u koju se dodaju vrednosti i vrednosti koje se dodaju. Funkcija vraća dva rezultata. Jedan je nova tabela (sa dodatim vrednostima), a drugi govori o uspešnosti dodavanja, uzima vrednost 1 ukoliko je uspešno a 0 ukoliko je neuspešno. Funkcija neće dodati ni jednu vrednost ukoliko bi time mogao biti narušen integritet primarnog ključa, ili ograničenje domena (forsira se atomičnost ove operacije). Ukoliko se decimalni broj upisuje u kolonu čiji je domen skup celih brojeva, ne dolazi do povrede ograničenja domena već se zadati broj zaokružuje na najbliži ceo broj, pa onda dodaje. Takođe, postoji funkcija koja uvek doda zadate vrednosti, a ukoliko je narušen integritet primarnog ključa tabelu pretvori u pogled. Ukoliko je potrebno dodati više vrsti u tabelu, može se pisati posebna rečenica dodavanja za svaku, ili se mogu ugnezditi upiti (<imeTabele>=dodaj_u_pogled(dodaj_u_pogled(<imeTabele>, {<vrednost11>, <vrednost12>, … <vrednost1N>}), {<vrednost21>, <vrednost22>, ... <vrednost2N>})).

Primer:

radnici=dodaj(radnici,{1,'Pera','Peric',1234567890123,'Ulica br.1',50000});

3. Operacije selekcije i projekcije

Sintaksa:

<ime_nove_tabele>=selekcija(<vektor>,<imePrvobitneTabele>)

<ime_nove_tabele>=projekcija(<imena_izabranih_kolona>, <imePrvobitneTabele>)

Selekcija uzima samo određene vrste iz tabele i smešta ih u novu tabelu. Izlazni podatak je tabela sa izabranim vrednostima, a ulazni podaci su tabela iz koje se uzimaju vrednosti i nula-jedan vektor kojim se govori koje vrste se uzimaju (ako je na *k*-tom mestu u vektoru 1, ta vrsta se dodaje u novu matricu, inače se ne dodaje). Pošto formiranje takvog vektora može biti teško izvodljivo za tabele sa mnogo podataka, napravljene su dve pomoćne skripte koje automatski formiraju traženi vektor po zadatom uslovu (više o ovim funkcijama pod tačkom 4).

Projekcija uzima određene kolone (sa svim vrednostima u njima) i prepisuje ih u novu tabelu. Kao ulazne podatke ima tabelu nad kojom se projekcija vrši i kolone koje se uzimaju, zadate imenima. Projekcija tabele ne mora uključivati sve kolone prima-

rnog ključa. U slučajevima da je integritet primarnog ključa u novonastaloj tabeli narušen, ta tabela se pretvara u pogled (originalna tabela ostaje nepromenjena). Ovo je jedan od razloga zašto je neophodna implementacija pogleda.

Primer:

```
samoImena=projekcija({'Ime'},radnici);
```

(Ova rečenica bi prikazala imena svih radnika koji se nalaze u tabeli radnici)

```
jedanOdPet=selekcija([0 1 0 0 0],radnici);
```

(Ova rečenica bi uzela samo drugog radnika od pet, koliko ih imamo u tabeli)

4. Pomoćne funkcije u vezi sa selekcijom

Sintaksa:

```
<vektor>=uslov1(<imeTabele>,<imeKolone>,<kriterijum_i_vrednost>)
<vektor>=uslov2(<imeTabele>,<imeKolone1>,<kriterijum>, <imeKolone2>)
```

Dve pomoćne funkcije olakšavaju kreiranje vektora koji se koristi kod selekcije. Prva poredi vrednost iz određene kolone sa konstantom, dok druga poredi vrednosti dve kolone. Prva kao ulazne parametre ima tabelu za koju formiramo vektor, kolonu iz te tabele (reprezentovanu imenom), operaciju koja se koristi za poređenje i (konstantnu) vrednost sa kojom se poredi. Druga kao ulazne podatke dobija tabelu, ime prve kolone, operaciju poređenja i ime druge kolone. Ovde je moguće realizovati samo najjednostavnije logičke izraze, a podrazumeva se da osoba koja koristi Matlab zna dovoljno o teoriji skupova da bi složene upite dobila operacijama unije, preseka i razlike tabela.

Primer:

```
vektor=uslov1(radnici, 'Plata','>=1000000');
rukovodioci=selekcija(vektor, radnici);
```

(Ove dve rečenice bi izabrale sve radnike koji imaju platu veću od 1000000)

```
rukovodioci=selekcija(uslov1(radnici, 'Plata', '>=1000000'), radnici);
```

(Ova rečenica je ekvivalentna sa prethodne dve)

```
vektor=uslov2(radnici,'Plata','*2>700*','GodineRS');
preplaceni=selekcija(vektor, radnici);
```

(Ove dve rečenice bi izdvojile radnike kojima je odnos plate i godina radnog staža veći od 2:700)

```
preplaceni=selekcija(uslov2(radnici,'Plata','*2>700*','GodineRS'), radnici);
```

(Ova rečenica je ekvivalentna sa prethodne dve)

5. Unija, presek i razlika

Sintaksa:

```
<imeNove>=unija(<imeTabele1>,<ImeTabele2>)
<imeNove>=presek(<imeTabele1>,<ImeTabele2>)
<imeNove>=razlika(<imeTabele1>,<ImeTabele2>)
```

Ove funkcije kao ulazne parametre imaju dve tabele, a kao izlaz tebelu koja je rezultat izvršene operacije. Problem je što kod uniranja tabela može doći do ponavljanja primarnog ključa. Zato operacija unije u tim slučajevima vraća pogled umesto tabele.

6. Spajanje tabela po kriterijumu

Sintaksa:

```
<imeNove>=equijoin(<imeTabele1>,<imeKolone1>,<ImeTabele2>,<imeKolone2>)
```

Implementirana je samo operacija spajanja dve tabele u jednu po kriterijumu jednakosti (*equijoin*). Funkcija kao parametre uzima tabele koje spaja i kolone iz svake tabele koju poredi. Kao rezultat vraća tabelu sa svim kolonama koje su se pojavile u barem jednoj od polaznih tabela. Ukoliko se ime kolone ponavlja, na njega će u jednom slučaju biti dodato '_2' da bi se imena tabela razlikovala.

Primer:

U bazi imamo tabelu sa radnicima i tabelu sa telefonima. Telefoni su povezani sa radnicima jedan na prema više (jednom radniku odgovara više telefona). Potrebno je izdvojiti unose u formatu za telefonski imenik za svakog radnika. U tabeli telefoni postoji polje ID_radnika koje se odnosi na radnika čiji je to telefon.

```
imenik=equijoin(projekcija({'ID','Ime','Prezime'}, radnici),'ID',telefoni,'ID_radnika')
imenik=projekcija({'Ime','Prezime','Broj'},imenik)
```

Prvo se izvršava projekcija u prvoj rečenici da bi odmah izbacili gomilu nepotrebnih podataka (adresu, platu i sl). Zatim se vrši spajanje tabele nastale projekcijom i tabele telefoni. Posle spajanja opet imamo gomilu nepotrebnih podataka (ID iz tabele radnika, ID iz tabele telefona, ID_radnika iz tabele telefona), pa ih eliminišemo još jednom projekcijom.

7. Prikazivanje

Sintaksa:

```
prikazi(radnici)
```

Implementirana je funkcija koja prikazuje tabelu u formatu razumljivom krajnjem korisniku. Naime, Matlab prikazuje varijable kako su fizički zapisane, te bi tabelu prikazao tako što bi naveo polja koja ona ima (kao struktura). Za krajnjeg korisnika to je potpuno nerazumljivo, jer njega zanimaju vrednosti u tabeli, pa ova skripta omogućava da korisnik vidi koje su vrednosti ovog trenutka u tabeli.

8. Pomoćne funkcije

Sintaksa:

<Rbkolone>=imeURB(<imeTabele>,<imeKolone>);
<ImeKolone>=RBUime(<imeTabele>,<rbKolone>);
<bezDuplikata>=ukloniDuplikate(<saDuplikatima>);
<logicka_varijabla>=element({1,'Pera','Peric', 1234567890123,'Ulica br.1',50000}, radnici);
<logicka_varijabla>=unicomp(<imeTabele1>,<imeTabele2>);

Implementirane su i neke funkcije koje nisu neophodne, ali rad učine lakšim. To su funkcija koja pretvara ime kolone u redni broj pod kojim je tabela memorisana u fizičkoj bazi i njoj inverzna funkcija (koja pretvara redni broj u ime kolone), funkcija koja uklanja duplikate vrste iz tabele, funkcija koja proverava da li u tabeli postoji vrsta sa određenim vrednostima i funkcija koja proverava da li su dve tabele unijski kompatibilne.

Operacije koje nisu implementirane

Funkcije deljenja i Dekartovog proizvoda nisu implementirane, pošto se mogu zameniti (jednim ili više) pozivanjem funkcije spajanja po kriterijumu jednakosti i funkcija unije, preseka, razlike, selekcije i projekcije. Takođe, ove operacije zahtevaju previše memorije i vremenski su prilično zahtevne. U Dekartovom proizvodu spaja se svaka kolona sa svakom, pa vreme izvršavanja kvadratno raste i nije ga moguće optimizovati a, takođe, krajnji rezultat zauzima previše memorije. Kod deljenja se provera vrši u kvadratnom vremenu, pa je, takođe, vrlo spora.

Zaključak

Po završetku pisanja funkcija napisana je i dokumentacija, tako da svako može da koristi funkcije.

Jedan od problema najkorišćenijeg jezika za manipulaciju bazama podataka, SQL-a je to što ne postoji standardni način za pravljenje višestruko ugnežđenih rečenica, što je u ovom projektu moguće (moguće je praktično beskonačno ugnežđivanje rečenica). Takođe, Matlab ima mogućnost višedimenzionih matrica (3D; 4D; 5D; :::;nD), tako da je moguće pratiti promenu baze npr. kroz vreme.

Predlozi za dalji rad. U ovom radu nije bilo potrebno implementirati referencijalni integritet baza podataka, a pošto je to neophodno u realnom sistemu za upravljanje relacionim bazama podataka, taj deo se može realizovati u nekom od narednih projekata. Takođe bi bilo korisno napisati program koji bi bazu iz formata koji se ovde koristi (standardni Matlab format) prevodio u neki drugi format, čitljv drugim alatima namenjenim radu sa bazama podataka.

Zahvalnost. Realizaciju ovog projekta značajno su pomogli: Dragan Toroman, vođa seminara računarstva u IS Petnica, Milan Gornik, saradnik na seminaru računarstva, Branislava Bajković-Lazarević i Filip Marić, profesori u Računarskoj gimnaziji u Beogradu, kao i kolege sa seminara računarstva. Ovom prilikom im se svima zahvaljujem.

Filip Živković

Implementation of Database Manipulation Tools in Matlab

The goal of this project is the implementation of database manipulation tools in Matlab, more precisely RDBMS (Relational database management system) routines, with all data held in tables, and in some form all relations between tables also held in tables. Since the routines are implemented as a function library, it is possible to make infinite query nesting. This is the connection between database and Matlab, as it is possible to manipulate databases using Matlab functions, Matlab syntax, Matlab structures, and more generally the Matlab "way of thinking".