Математичка гинмазија

МАТУРСКИ РАД

из предмета
Рачунарство и информатика

Израда програма за рад са базом података у програмском језику *C*

Ученик:
Војин Радовановић, IVц

Ментор:
Милош Арсић

Београд, јун 2021.

Садржај

1 Увод

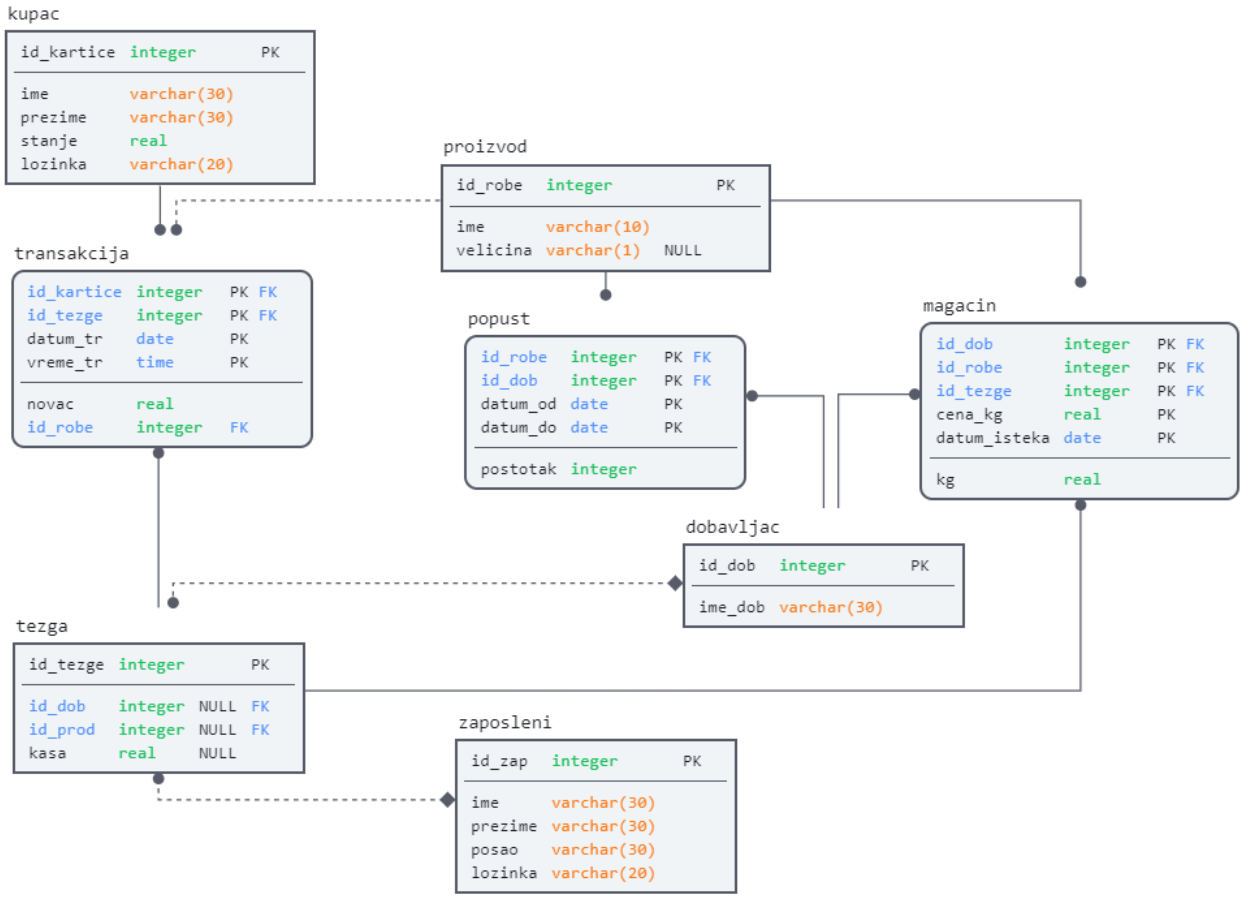
База података *Db2* је релациона база података, део шире линије производа *Db2* компаније *IBM*, подржана од стране оперативних система *Linux, Unix* и *Windows*. *Db2* пружа могућност уграђивања *SQL* наредби у програмске језике *C, C++* и *COBOL*, односно извршавања дела кода, написаног *SQL*-ом, унутар програма, написаног једним од тих програмских језика.

Овај рад приказује како се *SQL* може комбиновати са програмским језиком *C* унутар *Db2* и истражује могућности таквог начина рада на примеру базе података е-пијаце. У раду се, уз разноврсне примере, приказује и програм који омогућава пријављивање на апликацију е-пијаце и обављање задатака у зависности од типа корисника. На крају рада биће представљени недостаци рада са базом података само помоћу *SQL*-а и како се они могу премостити употребом програмског језика, као што је *C*.

База података и упити над њом су израђени у *Lubuntu*-у, дистрибуцији оперативног система *Linux*.

2 База података

База података е-пијаце се састоји из осам табела, које ће бити представљене у овом делу рада. То су: *производ*, *добављач*, *запослени*, *тезга*, *купац*, *трансакција*, *попуст* и *магацин*.



слика 1: шематски приказ базе података

2.1 Табела *добављач*

Табела *добављач* садржи основне податке једног добављача, чији се производи продају на пијаци. Атрибути ове табеле су:

- *id_dob* – примарни кључ табеле *добављач*, тип података је *integer* са ограничењем *not null*. Садржи интерну шифру добављача на пијаци.
- *ime_dob* – садржи име добављача, тип података је *varchar(30)* са ограничењем *not null*.

Структура табеле *добављач* је описана *SQL* наредбом:

```
create table dobavljac(
id_dob integer not null,
ime_dob varchar(30) not null,
primary key(id_dob)
);
```

2.2 Табела *производ*

Табела *производ* садржи основне податке једног производа, који се продаје на пијаци. Атрибути ове табеле су:

- *id_robe* – примарни кључ табеле *производ*, тип података је *integer* са ограничењем *not null*. Садржи интерну шифру производа на пијаци.
- *ime* – садржи име производа, тип података је *varchar(10)* са ограничењем *not null*.
- *velicina* – описује величину производа, која може бити *s* (мала величина), *m* (средња величина) или *l* (велика величина), тип података је *varchar(1)*.

Структура табеле *производ* је описана *SQL* наредбом:

```
create table proizvod(
id_robe integer not null,
ime varchar(10) not null,
velicina varchar(1),
primary key(id_robe)
);
```

2.3 Табела *запослени*

Табела *запослени* садржи податке запосленог на пијаци. Атрибути ове табеле су:

- *id_zap* – примарни кључ табеле *запослени*, тип података је *integer* са ограничењем *not null*. Садржи интерну шифру запосленог на пијаци.
- *ime* – садржи име запосленог, тип података је *varchar(30)* са ограничењем *not null*.
- *prezime* – садржи презиме запосленог, тип података је *varchar(30)* са ограничењем *not null*.
- *posao* – садржи посао, који запослени обавља, тип података је *varchar(30)* са ограничењем *not null*.
- *lozinka* – садржи лозинку, коју запослени користи да би се улоговао на свој налог апликације е-пијаце, тип података је *varchar(20)* са ограничењем *not null*.

Структура табеле *запослени* је описана *SQL* наредбом:

```
create table zaposleni(
id_zap integer not null,
ime varchar(30) not null,
prezime varchar(30) not null,
posao varchar(30) not null,
lozinka varchar(20) not null,
primary key(id_zap)
);
```

2.4 Табела *тезга*

Табела *тезга* садржи основне податке тезге на пијаци. Атрибути ове табеле су:

- *id_tezge* – примарни кључ табеле *тезга*, тип података је *integer* са ограничењем *not null*. Садржи интерну шифру тезге на пијаци.
- *id_dob* – страни кључ, који реферише на *id_dob* табеле *добављач*, тип података је *integer*. Садржи информацију о добављачу, који је закупац тезге.
- *id_prod* – страни кључ, који реферише на *id_zap* табеле *запослени*, тип података је *integer*. Садржи информацију о продавцу, који опслужује тезгу.
- *kasa* – описује количину новца, који се налази у каси тезге, тип података је *real*.

Структура табеле *тезга* је описана *SQL* наредбом:

```
create table tezga(
id_tezge integer not null,
id_dob integer,
id_prod integer,
kasa real,
primary key(id_tezge),
foreign key(id_dob) references dobavljac(id_dob),
foreign key(id_prod) references zaposleni(id_zap)
);
```

2.5 Табела *купац*

Табела *купац* садржи основне податке о купцу кориснику услуга е-пијаце. Атрибути ове табеле су:

- *id_kartice* – примарни кључ табеле *купац*, тип података је *integer* са ограничењем *not null*. Садржи број платне картице купца.
- *ime* – садржи име купца, тип података је *varchar(30)* са ограничењем *not null*.
- *prezime* – садржи презиме купца, тип података је *varchar(30)* са ограничењем *not null*.
- *stanje* – садржи тренутно стање купца на платној картици, тип података је *real* са ограничењем *not null*.
- *lozinka* – садржи лозинку, коју купац користи да би се улоговао на свој налог апликације е-пијаце, тип података је *varchar(20)* са ограничењем *not null*.

Структура табеле *купац* је описана *SQL* наредбом:

```
create table kupac(
id_kartice integer not null,
ime varchar(30) not null,
prezime varchar(30) not null,
stanje real not null,
lozinka varchar(20) not null,
primary key(id_kartice)
);
```

2.6 Табела *трансакција*

Табела *трансакција* садржи основне податке о трансакцијама, које се дешавају на пијаци. Атрибути ове табеле су:

- *id_kartice* – страни кључ, који реферише на *id_kartice* табеле *купац*, тип података је *integer* са ограничењем *not null*. Садржи број картице, којом је извршена трансакција.
- *novac* – садржи количину новца, која је плаћена, тип података је *real* са ограничењем *not null*.
- *id_tezge* – страни кључ, који реферише на *id_tezge* табеле *тезга*, тип података је *integer* са ограничењем *not null*. Садржи шифру тезге, на којој је извршена трансакција.
- *id_robe* – страни кључ, који реферише на *id_robe* табеле *производ*, тип података је *integer* са ограничењем *not null*. Садржи шифру робе, која је купљена.
- *datum_tr* – означава датум, на који се извршила трансакција, тип података је *date* са ограничењем *not null*.
- *vreme_tr* – означава време, када се десила трансакција, тип података је *time* са ограничењем *not null*.
- Примарни кључ ове табеле заједно чине *id_kartice, id_tezge, datum_tr* и *vreme_tr*.

Структура табеле *трансакција* је описана *SQL* наредбом:

```
create table transakcija(
id_kartice integer not null,
novac real not null,
id_tezge integer not null,
id_robe integer not null,
datum_tr date not null,
vreme_tr time not null,
primary key(id_kartice, id_tezge, datum_tr, vreme_tr),
foreign key(id_kartice) references kupac(id_kartice),
foreign key(id_tezge) references tezga(id_tezge),
foreign key(id_robe) references proizvod(id_robe)
);
```

2.7 Табела *попуст*

Табела *попуст* садржи основне податке о попустима на пијаци, који су били, јесу или ће бити на снази. Атрибути ове табеле су:

- *id_robe* – страни кључ, који реферише на *id_robe* табеле *производ*, тип података је *integer* са ограничењем *not null*. Садржи шифру робе, која је на попусту.
- *id_dob* – страни кључ, који реферише на *id_dob* табеле *добављач*, тип података је *integer* са ограничењем *not null*. Садржи информацију о добављачу, чији је производ на попусту.
- *postotak* – означава величину попуста у процентима, тип података је *integer* са ограничењем *not null*.
- *datum_od* – означава датум, од којег почиње промоција, тип података је *date* са ограничењем *not null*.
- *datum_do* – означава датум, до којег траје промоција, тип података је *date* са ограничењем *not null*.
- Примарни кључ ове табеле заједно чине *id_robe, id_dob, datum_od* и *datum_do*.

Структура табеле *попуст* је описана *SQL* наредбом:

```
create table popust(
id_robe integer not null,
id_dob integer not null,
postotak integer not null,
datum_od date not null,
datum_do date not null,
primary key(id_robe, id_dob, datum_od, datum_do),
foreign key(id_robe) references proizvod(id_robe),
foreign key(id_dob) references dobavljac(id_dob)
);
```

2.8 Табела *магацин*

Табела *магацин* садржи основне податке о производима, којима је попуњен магацин пијаце. Атрибути ове табеле су:

- *id_dob* – страни кључ, који реферише на *id_dob* табеле *добављач*, тип података је *integer* са ограничењем *not null*. Садржи информацију о добављачу, чији се производ чува у магацину.
- *id_robe* – страни кључ, који реферише на *id_robe* табеле *производ*, тип података је *integer* са ограничењем *not null*. Садржи шифру робе, која се чува у магацину.
- *id_tezge* – страни кључ, који реферише на *id_tezge* табеле *тезга*, тип података је *integer* са ограничењем *not null*. Садржи шифру тезге, којој припада производ, који се чува у магацину.
- *kg* – описује колико килограма одређеног производа одређеног произвођача са одређене тезге има у магацину, тип података је *real* са ограничењем *not null*.
- *cena_kg* – описује колико кошта одређена роба у магацину по килограму, тип података је *real* са ограничењем *not null*.
- *datum_isteka* – садржи датум, када истиче одређени производ у магацину, тип података је *date* са ограничењем *not null*.
- Примарни кључ ове табеле заједно чине *id_dob*, *id_robe*, *id_tezge*, *cena_kg* и *datum_isteka*.

Структура табеле *магацин* је описана *SQL* наредбом:

```
create table magacin(
id_dob integer not null,
id_robe integer not null,
id_tezge integer not null,
kg real not null,
cena_kg real not null,
datum_isteka date not null,
primary key(id_dob, id_robe, id_tezge, cena_kg, datum_isteka),
foreign key(id_dob) references dobavljac(id_dob),
foreign key(id_robe) references proizvod(id_robe),
foreign key(id_tezge) references tezga(id_tezge)
);
```

3 Рад у терминалу

У даљем тексту, како се за фајлове, написане помоћу *C* и *SQL* програмских језика, у *Db2* користи екстензија *.sqc*, тако написане програме означаваћемо као *SQC* програме.

У *Linux*-у је уобичајено да се за манипулисање програмима користи *Linux*-ов терминал. Ради удобности се наводи пар команди терминала:

- *cd* – прелазак у наведени директоријум (*cd Desktop/c-upiti* нас премешта у директоријум *c-upiti*, који се налази на десктопу),
- *ls* – излиставање садржаја тренутног директоријума,
- ./ – служи за извршавање програма из директоријума, у којем се налазимо,
- *db2start* – покреће базу података *Db2*.

Пре извршавања *SQC* програма потребно је да се најпре изврши команда *db2start*, а потом преведе потребни *SQC* програм, што се постиже помоћу *SQC* преводиоца. Да би се *SQC* програм успешно превео потребно је да *SQC* преводилац буде у истом директоријуму, као и *SQC* програм, који преводимо. Пример превођења програма *upit1*.sqc је:

```
./prevodjenje1 upit1 c-upiti vradovanovic kvmngh
```

Након превођења ствара се извршни програм истог имена као *SQC* програм, у примеру назив је *upit1*. Након превођења извршни програм покрећемо, као у следећем примеру:

```
./upit1
```

4 Програмирање уграђених *SQL* апликација

SQC програми имају дефинисану структуру и садрже:

- укључивање потребних библиотека,
- декларације хост променљивих, које ће учествовати у *SQL* упитима,
- повезивање на базу података,
- извршавање *SQL* упита,
- обраду *SQL* грешака и упозорења у вези са извршавањем *SQL* упита,
- уклањање везе са базом података.

Такође, важно је напоменути да сви делови *SQC* програма, који се односе на *SQL*, почињу са *EXEC SQL*, а завршавају се знаком **;** , као у примеру:

```
EXEC SQL delete from table transakcija;
```

4.1 Декларација хост променљивих

Хост променљиве, односно променљиве, које се користе за чување резултата *SQL* упита, декларишу се у делу кода, познатом као *DECLARE SECTION*. *DECLARE SECTION* почиње командом *„EXEC SQL BEGIN DECLARE SECTION;“*, а завршава се командом *„EXEC SQL END DECLARE SECTION;“*. Унутар њега се декларишу променљиве, које по свом типу одговарају променљивама резултата *SQL* упита. На пример, *SQL* променљивама типа *integer* одговарају *C* променљиве типа *int*, док *SQL* променљивама типа *boolean* одговарају *C* променљивама типа *bool*. Изузетак представљају *SQL* променљиве типа *varchar(x)* (*varchar* дужине *x*), којима одговарају *C* променљиве *char [x+1]* (јер *string* променљиве са *x* карактера представљају низ од *x+1* чланова, где је последњи члан терминирајућа нула). На сличан начин *SQL* променљивама типа *date* одговарају *C* променљиве типа *char [9]*.

Унутар делова програма, написаних *SQL*-ом, хост променљиве обележавају знаком **:** , као у примеру:

```
EXEC SQL select id_dob into :Cid_dob;
```

Посебно се декларишу променљиве за атрибуте табела, који могу имати вредност *null*. Тада се уз хост променљиве декларишу индикаторске променљиве типа *short*, које садрже информацију о томе, да ли хост променљива има вредност *null*. Унутар делова програма, написаним *SQL*-ом, индикаторске променљиве се обележавају знаком **:** и стоје одмах иза хост променљивих, као у примеру:

```
EXEC SQL select velicina into :Cvelicina:id_velicina
        from proizvod;
```

Уколико индикаторска променљива има негативну вредност, то значи да је у том случају вредност атрибута заиста *null*, па се не може користити даље у коду. Супротно, уколико је вредност ненегативна, хост променљива се може даље користити у коду.

4.2 *SQLCA* и функција грешке

Ради провере исправности *SQL* кода унутар *SQC* програма, свака грешка у извршавању наредбе у *SQC* програму је обележена негативним бројем, који се чува у променљивој *SQLCODE*. Да би програм успешно одредио број, који одговара тренутној грешци, користи се *SQLCA* – структура података која се ажурира након сваког извршеног *SQL* упита. *SQLCA* се у програм укључује наредбом:

```
EXEC SQL INCLUDE SQLCA;
```

Упити у овом раду ће имати функцију за обраду грешке испред *main* функције, која се позива након сваке *EXEC SQL* наредбе, а у случају грешке исписује *SQLCODE* и прекида извршавање програма. Функција за обраду грешке је у рада дефинисана на следећи начин:

```
void greska(char gr[])
{
        if (SQLCODE < 0) {
        printf("SQLCODE %d %s\n\n", SQLCODE, gr);

        EXEC SQL CONNECT RESET;
        exit(EXIT_FAILURE);
        }
}
```

4.3 Функција *main*

Унутар функције *main* врши се повезивање са базом помоћу наредбе *CONNECT TO* и, након завршетка програма, уклањање те везе помоћу наредбе *CONNECT RESET*.

Следећи пример показује како се улазни параметри могу користити у *SQC* програмима. Такође, у овом и свим осталим примерима у раду биће изостављен почетни део, у који спада укључивање библиотека и *SQLCA*, *DECLARE SECTION* и функција грешке, који су раније већ представљени у овом раду.

Пример 1:

Исписати име добављача, на чијој тезги ради продавац, чије се име и презиме уноси. Ако не постоји, исписати „не постоји такав продавац“.

```
int main(){

  printf("Unesite ime, pa prezime prodavca\n");
  scanf("%s %s", Cime, Cprezime);

  EXEC SQL CONNECT TO e-pijaca USER vradovanovic USING kvmngh;
  greska("Povezivanje");

  EXEC SQL
  select ime_dob into :Cime_dob
  from dobavljac d join tezga t on d.id_dob=t.id_dob
```

```
        join zaposleni z on t.id_prod=z.id_zap
  where z.ime=:Cime and z.prezime=:Cprezime;
  greska("Upit");

  if(Cime_dob[0]=='/')                    //unutar DECLARE SECTION je Cime_dob
    printf("Ne postoji takav prodavac.\n");   //dodeljena pocetna vrednost „/“
  else
    printf("%s\n",Cime_dob);

  EXEC SQL CONNECT RESET;
  greska("Uklanjanje veze");

  exit(EXIT_SUCCESS);
}
```

У сагласности са структуром *SQC* програма, и пример 1 има потребне библиотеке, хост промењиве, функцију грешке, успостављање и уклањање везе са базом података и *SQL* упите. Унутар *DECLARE SECTION* хост промењивој *Cime_dob* додељена је почетна вредност „/“, која се потом користи при проверавању тога, да ли је *SQL* упит дао излазни податак или не.

4.4 Курсори

Резултати *SQL* упита су обично привремене табеле са својим атрибутима и инстанцама. С друге стране, програмски језик *C* не дозвољава да променљива или структура, у којој чувамо вредност појединачне инстанце, истовремено садржи више различитих вредности. Мотивисано овим, али и жељом да се инстанце резултата упита одвојено обрађују, *SQC* програми нуде могућност курсора – посредника, у коме се може чувати инстанца упита, те се потом даље може манипулисати резултатима упита у програму.

Сви курсори се обрађују на следећи начин:

- декларише се курсор помоћу наредбе *DECLARE CURSOR*,
- курсор се отвара помоћу наредбе *OPEN*,
- вредности инстанци се, једна по једна, узимају помоћу наредбе *FETCH*,
- уколико је потребно, извршити *SQL* наредбе *UPDATE* и *DELETE*,
- затворити курсор наредбом *CLOSE*.

Слично као *SELECT INTO*, помоћу наредби *FETCH INTO* се вредности инстанце чувају у хост промењивама. Наредба *FETCH* је програмирана тако, да сваки пут кад се изврши, курсор пређе на следећи ред, односно инстанцу, у табели. Овај процес се завршава уколико након наредбе *FETCH* промењива *SQLCODE* добије вредност 100.

Постоји више врста курсора: они могу бити *FOR READ ONLY*, односно само за читање резултата упита, и *FOR UPDATE*, односно за ажурирање података у табели. Приликом дефинисања курсора се његова врста не мора истаћи, при чему се сматра да је тада курсор врсте *READ ONLY*.

Рад са *FOR READ ONLY* курсором илуструје пример 2, а рад са *FOR UPDATE* курсором пример 3.

Пример 2:

За све попусте, чија је промоција истекла до унетог датума, одредити средњу вредност по произвођачу, а резултате уредити у растућем поретку.

```
int main(){
  EXEC SQL CONNECT TO e-pijaca USER vradovanovic USING kvmngh;
  greska("Povezivanje");

  printf("Unesite zeljeni datum\n");
  scanf("%s", Cdatum);

  EXEC SQL DECLARE kursor CURSOR FOR
  with pom as(
          select id_dob, postotak
          from popust
          where datum_do<:Cdatum
  )
  select ime_dob, cast(avg(postotak*1.0) as decimal(4,2)) as sr_vr
  from dobavljac d join pom on d.id_dob=pom.id_dob
  group by ime_dob
  order by sr_vr
  FOR READ ONLY;
  greska("Deklarisanje kursora");

  EXEC SQL OPEN kursor;
  greska("Otvaranje kursora");

  printf("DOBAVLJAC          SR. POPUST \n-------------------------\n");
  while(1)
  {
    EXEC SQL FETCH kursor
    INTO :Cime_dob, :Csr_vr;
    greska("Uzimanje iz kursora");

    if (SQLCODE==100) break;

    printf("%s              %.2f\n", Cime_dob, Csr_vr);
  }
  EXEC SQL CLOSE kursor;
  greska("Zatvaranje kursora");

  EXEC SQL CONNECT RESET;
  greska("Uklanjanje veze");

  exit(EXIT_SUCCESS);
}
```

За разлику од примера 1, где се у хост променљивој чувао резултат *SQL* упита, за који је постојала само једна одговарајућа вредност, у примеру 2 то није случај (јер се исписује читава табела резултат), те се прибегава коришћењу курсора. Пример садржи све елементе обраде курсора: његово декларисање и отварање, коришћење функције *FETCH* и затварање курсора, а такође користи и улазне податке.

Пример 3:

Променити у табели *производ* сва имена производа, који се зову *„paradaiz“*, у *„paradajz“*.

```
int main(){

  EXEC SQL CONNECT TO vstud USER student USING abcdef;
  greska("Povezivanje");

  EXEC SQL DECLARE kursor CURSOR FOR
  select id_robe
  from proizvod
  where ime='paradaiz'
  FOR UPDATE OF ime;                  //azurirace se atribut ime
  greska("Deklarisanje kursora");

  EXEC SQL OPEN kursor;
  greska("Otvaranje kursora");

  while(1)
  {
    EXEC SQL FETCH kursor INTO :Cid_robe;
    greska("Uzimanje iz kursora");

    if (SQLCODE==100) break;

    EXEC SQL UPDATE proizvod
    SET ime='paradajz'
    WHERE CURRENT OF kursor;
  }

  EXEC SQL CLOSE kursor;
  greska("Zatvaranje kursora");

  EXEC SQL CONNECT RESET;
  greska("Uklanjanje veze");

  exit(EXIT_SUCCESS);
}
```

У случају примера 3 користи се курсор врсте *FOR UPDATE*. Такав курсор се декларише слично као *FOR READ ONLY* курсор; разлика је у томе, што се при навођењу врсте наводи и атрибут, односно атрибути, који ће се касније у коду ажурирати.

Унутар петље се врши наредба *FETCH* и провера да ли је курсор дошао до краја, а после њих се врши *SQL* наредба *update*. За разлику од ажурирања у *SQL*-у, услов ажурирања у примеру 3 је наредба *WHERE CURRENT OF kursor*, односно ажурира се инстанца, на којој се тренутно налази курсор.

Постоји и могућност да више курсора оперише у истом коду, те да инстанце које показује други курсор варирају од вредности инстанце, која се садржи у првом курсору, што показује следећи пример:

Пример 4:

За сваку тезгу продавца *„Matijevic“* исписати све производе класе *„m“*.

```
int main(){

  EXEC SQL CONNECT TO e-pijaca USER vradovanovic USING kvmngh;
  greska("Povezivanje");

  EXEC SQL DECLARE k1 CURSOR FOR
  select id_tezge
  from tezga join dobavljac on tezga.id_dob=dobavljac.id_dob
  where ime_dob='Matijevic'
  FOR READ ONLY;
  greska("Deklarisanje k1");

  EXEC SQL DECLARE k2 CURSOR FOR
  select rtrim(ime)
  from proizvod p join magacin m on p.id_robe=m.id_robe
  where velicina='m' and m.id_tezge=:Cid_tezge
  FOR READ ONLY;
  greska("Deklarisanje k2");

  EXEC SQL OPEN k1;
  greska("Otvaranje k1");

  int i;

  while(1)
  {
    EXEC SQL FETCH k1
    INTO :Cid_tezge;
    greska("Uzimanje iz k1");

    if (SQLCODE==100) break;
```

```
   printf("TEZGA %d \n------------------\n", Cid_tezge);

   EXEC SQL OPEN k2;
   greska("Otvaranje k2");

   i=0;            //indikator za broj proizvoda klase m na tezgi, ciji je id Cid_tezge

   while(1)
   {
     EXEC SQL FETCH k2 INTO :Cime;
     greska("Uzimanje iz k2");

     if(SQLCODE==100) break;

     printf("%s\n", Cime);
     i++;
   }

   if(i==0) printf("Nema\n");          //ukoliko nema takvih proizvoda, program ce ispisati „Nema“

   printf("\n");

   EXEC SQL CLOSE k2;
   greska("Zatvaranje k2");
  }

  EXEC SQL CLOSE k1;
  greska("Zatvaranje k1");

  EXEC SQL CONNECT RESET;
  greska("Uklanjanje veze");

  exit(EXIT_SUCCESS);
}
```

У овом случају се курсор *k2* отвара тек након курсора *k1*, јер његови резултати зависе од хост променљиве *Cid_tezge*, у којој се чувају резултати упита курсора *k1*.Сходно томе, курсор *k2* се и затвара пре курсора *k1*, а тачније у сваком циклусу петље, јер са сваким новим циклусом хост променљива *Cid_tezge* добија нову вредност. Унутар програма се користи и променљива *i*, која бележи број потребних производа; у случају да је њена вредност 0, исписује се „*Nema*“.

4.5 Курсор у курсору

Курсор у курсору је име за појаву, у којој два курсора раде паралелно, а резултат њиховог рада су инстанце, које представљају резултат скуповне операције над привременим табелама резултата тих курсора.

На пример, уколико се траже тезге, које немају продавца или су на другом спрату, могуће је решити тај задатак декларисањем два *READ ONLY* курсора, једног за тезге без продавца, а другог за тезге на другом спрату. Након тога се у петљи упоређују резултати два курсора, да не би дошло до преклапања, и исписују се. Добија се својеврсна унија резултата два курсора, што може бити врло корисно у раду са више курсора.

Следећи пример илуструје појаву курсора у курсору као операцију пресека.

Пример 5:

Данас је пијачни дан и све породице добијају додатних 5% попуста на сву робу. Такође су и Цвети, па све особе, које се зову Цвета, добијају купон од 2% на прву куповину. Сазнати да ли постоји особа која може да искористи оба попуста.

```
int main()
{
  int kraj1=0, kraj2=0; //oznacavaju da su prvi, odnosno srugi kursor dosli do kraja
  int i;                //indikator, koji oznacava da li postoji trazena osoba

  EXEC SQL CONNECT TO e-pijaca USER vradovanovic USING kvmngh;
  greska("Povezivanje");

  EXEC SQL DECLARE cvetaK CURSOR FOR
    select id_kartice, ime, prezime
    from kupac
    where ime='Cveta'
    order by id_kartice;
  greska("Deklarisanje cvetaK");

  EXEC SQL DECLARE porodicaK CURSOR FOR
    select k1.id_kartice, k1.ime, k1.prezime
    from kupac k1 join kupac k2 on k1.prezime=k2.prezime
    where k1.id_kartice<>k2.id_kartice
    order by k1.id_kartice;
  greska("Deklarisanje porodicaK");

  EXEC SQL OPEN cvetaK;
  greska("Otvaranje cvetaK");

  EXEC SQL OPEN porodicaK;
  greska("Otvaranje porodicaK");
```

```
EXEC SQL FETCH cvetaK INTO :Cid_kartice1, :Cime1, :Cpr1;
greska("Uzimanje iz cvetaK");

if(SQLCODE == 100)
  kraj1=1;

EXEC SQL FETCH porodicaK INTO :Cid_kartice2, :Cime2, :Cpr2;
greska("Uzimanje iz porodicaK");

if(SQLCODE == 100)
  kraj2=1;

i=0;
while(1){
  if(kraj1 || kraj2)       //ukoliko je jedan od dva kursora dosao do kraja
    break;

  if(Cid_kartice1 > Cid_kartice2){
    EXEC SQL FETCH porodicaK INTO :Cid_kartice2, :Cime2, Cpr2;
    greska("Uzimanje iz porodicaK");

    if(SQLCODE == 100)
      kraj2=1;
  }
  else if(Cid_kartice1 < Cid_kartice2){
    EXEC SQL FETCH cvetaK INTO :Cid_kartice1, :Cime1, :Cpr1;
    greska("Uzimanje iz cvetaK");

    if(SQLCODE == 100)
      kraj1=1;
  }
  else {
    printf("%d: %s %s\n", Cid_kartice1, Cime1, Cpr1);
    i++;

    EXEC SQL FETCH cvetaK INTO :Cid_kartice1, :Cime1, :Cpr1;
    greska("Uzimanje iz cvetaK");
    if(SQLCODE == 100)
      kraj1=1;

    EXEC SQL FETCH porodicaK INTO :Cid_kartice2, :Cime2, Cpr2;
    greska("Uzimanje iz porodicaK");
    if(SQLCODE == 100)
      kraj2=1;
  }
}
```

```
  EEC SQL CLOSE cvetaK;
  greska(Close cvetaK");
  EXEC SQL CLOSE porodicaK;
  greska("Close porodicaK");
  if(i==0)
    printf("Ne postoji takva osoba.\n");

  EXEC SQL CONNECT RESET;
  greska("Uklanjanje veze");

return 0;
}
```

У примеру 5 су се поред два курсора користиле и променљиве *kraj1* и *kraj2*, које чувају информацију о доласку курсора до краја својих резултата. Унутар *while* петље се прво проверава да ли је један од курсора дошао до краја и вратио *SQLCODE 100*. Уколико није, користи се то што су резултати упита курсора уређени растуће, те се упоређују вредности курсора. Уколико је *id_kartice* једног мањи од *id_kartice* другог, мањи се повећава, а уколико је исти, то значи да се особа са датим *id_kartice* налази у пресеку резултата оба курсора, те се исписују њен *id_kartice*, име и презиме.

У задатку се такође употребљавао и бројач *i*, који је бележио колико има особа, које одговарају условима упита. Уколико би тај број био нула, било би исписано да не постоји таква особа.

5 Апликација е-пијаце

База података е-пијаце у овом раду је разрађена са идејом да се имплементира и апликација е-пијаце, која, користећи *SQC* програм, помаже корисницима апликације у обављању њихових задатака. Апликација је разрађена тако да је могу користити продавци пијаце за унос нових попуста и нових набавки робе за магацин у базу података е-пијаце, док је купци могу користити за обављање својих трансакција. Даље ће бити изложена *main* функција у деловима, који се односе на пријављивање на апликацију, корисничке могућности продаваца и корисничке могућности купаца.

5.1 Пријављивање на апликацију

Корисник апликације најпре уноси своје корисничко име, што је за продавце њихов *id_zap*, док је за купце то њихов *id_kartice*, након чега се уноси лозинка. Уколико се погреши приликом уноса података, корисник има још највише два покушаја да правилно унесе своје корисничке податке, а, ако не успе у томе, апликација се затвара, односно кориснику се забрањује приступ.

```
int main(){
  EXEC SQL CONNECT TO e-pijaca USER vradovanovic USING kvmngh;
greska("Povezivanje");

  int ind=0, br=0;

  while(ind==0 && br<3)
  {
          printf("Unesite korisnicki id.\n");
          scanf("%d", &id_kor);
          printf("Unesite svoju lozinku.\n");
          scanf("%s", loz);

    EXEC SQL
    select ime into :Cime
    from zaposleni
    where id_zap=:id_kor and lozinka=:loz and posao='prodavac';
    greska("Logovanje");

    if(Cime[0]=='/')
      {
       EXEC SQL
       select ime into :Cime
             from kupac
          where id_kartice=:id_kor and lozinka=:loz;
          greska("Logovanje");

          if(Cime[0]=='/')
          {
```

```
                br++;
        printf("Pogresno ste uneli korisnicko ime ili lozinku, imate jos %d pokusaja.\n", 3-br);
        }
        else
        {
                printf("Zdravo, %s!\n",Cime);
        ind=2;
        }
    }
  else
    {
     printf("Zdravo, %s!\n",Cime);
     ind=1;
    }
}

 if(br==3)
 {
        printf("Prekoracili ste dozvoljen broj pokusaja. Zabranjen Vam je pristup.\n");

        EXEC SQL CONNECT RESET;
  greska("Uklanjanje veze");

  exit(EXIT_SUCCESS);
 }
```

Променљива *ind* се односи на тип корисника: њена вредност је 1 уколико је корисник продавац, а 2 уколико је корисник купац.

5.2 Корисник продавац

Након што се продавац улогује, он има две опције: или уноси број 1 и уноси нов попуст добављача, код којег је запослен, или уноси број 2 и уноси измену података у катастар магацина.

```
 if(ind==1)
 {
  printf("Pred Vama je korisnicki meni za prodavce. Unesite:\n");
  printf("       1 za unos novog popusta Vaseg dobavljaca.\n");
  printf("       2 za unos nove nabavke.\n");

  int opcija;
  scanf("%d", &opcija);

  if(opcija==1)
  {
   EXEC SQL select id_dob into :Cid_dob
   from tezga
```

```
    where id_prod=:id_kor;
    greska("Nalazenje id_dob");

    printf("Unesite odgovarajuci id robe, postotak i datume pocetka i kraja trajanja promocije.\n");
    scanf("%d %d %s %s", &Cid_robe, &Cpostotak, Cdatum_od, Cdatum_do);

    EXEC SQL
    insert into popust(id_robe, id_dob, postotak, datum_od, datum_do) values
    (:Cid_robe, :Cid_dob, :Cpostotak, :Cdatum_od, :Cdatum_do);
    greska("Unos popusta");
    printf("Popust je uspesno unet.\n");
   }

   else if(opcija==2)
   {
    EXEC SQL select id_tezge, id_dob into :Cid_tezge, :Cid_dob
    from tezga
    where id_prod=:id_kor;
    greska("Nalazenje id_dob");

    printf("Unesite odgovarajuci id robe, kolicinu robe u kg, cenu robe po kg i datum isteka.\n");
    scanf("%d %f %f %s", &Cid_robe, &Ckg, &Ccena_kg, Cdatum_isteka);

    EXEC SQL
    insert into magacin(id_dob, id_robe, id_tezge, kg, cena_kg, datum_isteka) values
    (:Cid_dob, :Cid_robe, :Cid_tezge, :Ckg, :Ccena_kg, :Cdatum_isteka);
    if(SQLCODE==-803)
    {
     EXEC SQL
     update magacin
     set kg=kg+:Ckg
     where id_dob=:Cid_dob and id_robe=:Cid_robe and id_tezge=:Cid_tezge;
     greska("Azuriranje");
     }
    else greska("Unos u magacin");
    printf("Uspesno ste uneli robu u katastar magacina.\n");
   }
  }
```

Најпре се у првој опцији тражи *id_dob* који одговара добављачу, за којег ради продавац корисник апликације. Затим се уносе остале информације о попусту и врши се *INSERT* упит.

Што се тиче друге опције, прво се траже *id_dob* и *id_tezge*, који одговарају продавцу кориснику апликације. Затим продавац уноси друге битне податке, потребне за

магацин и врши се њихово убацивање у табелу командом *INSERT*. Уколико се, пак, деси да производ са свим подацима које је унео продавац већ постоји, врши се ажурирање и нови број килограма се додаје на већ постојећи.

Након обављених упита продавац се обавештава да је уношење података извршено.

5.3 Корисник купац

Корисник купац, након пријављивања на апликацију, има само могућност обављања трансакција, за шта прво бира производ, па потом добављача, тезгу, цену по килограму, број килограма који жели да узме и датум истека робе из понуђених, након чега се обавља трансакција уколико су испуњени услови да купац има довољно новца за трансакцију, као и да није затражио више килограма производа него што је у понуди.

```
if(ind==2)
  {
    printf("Pred Vama je korisnicki meni za kupce. Ovo su svi proizvodi nase pijace:\n");

    EXEC SQL DECLARE pr CURSOR FOR
    select *
    from proizvod
    FOR READ ONLY;
    greska("Deklarisanje pr");

    EXEC SQL OPEN pr;
    greska("Otvaranje pr");

    printf("id | ime        | velicina\n");
    while(1)                              //izlistavanje svih proizvoda, koji imaju svoju sifru u e-
pijaci
    {
      EXEC SQL FETCH pr INTO :Cid_robe, :Cime_robe, :Cvelicina:id_velicina;
      greska("Uzimanje iz pr");

      if(SQLCODE==100)
      {
        printf("\n");
        break;
      }

      printf("%d | %s | ", Cid_robe, Cime_robe);
      if(id_velicina>=0) printf("%c\n", Cvelicina);
      else printf("/\n");
    }
    EXEC SQL CLOSE pr;
    greska("Zatvaranje pr");
```

```
   printf("Unesite id proizvoda, koji zelite da kupite.\n");
   scanf("%d", &Cid_robe);

   EXEC SQL DECLARE mag CURSOR FOR
   select ime_dob, m.id_tezge, kg, cena_kg, coalesce(postotak, 0), m.datum_isteka
   from dobavljac d right join magacin m on d.id_dob=m.id_dob
     left join popust p on m.id_dob=p.id_dob and m.id_robe=p.id_robe
   where m.id_robe=:Cid_robe and m.datum_isteka>=CURRENT_DATE
   FOR READ ONLY;
   greska("Deklarisanje mag");

   ind=0;          //promenljiva ind sada belezi koja je po redu linija ispisa
   while(ind==0)        //ispisivanje svih instanci tabele magacin, koje sadrze trazeni proizvod
   {
     EXEC SQL OPEN mag;
     greska("Otvaranje mag");

     while(1)
     {
        EXEC SQL FETCH mag INTO :Cime, :Cid_tezge, :Ckg, :Ccena_kg, :Cpostotak,
:Cdatum_isteka;
       greska("Uzimanje iz mag");

       if(SQLCODE==100)
       {
         printf("\n");
         break;
       }

       ind++;
       if(ind==1)  printf("Br | ime dobavljaca | id tezge | kolicina u kg | cena po kg sa %% |
popust (%%) | datum isteka\n");

        printf("%d | %s | %d | %.2f | %.2f | %d | %s\n", ind, Cime, Cid_tezge, Ckg,
Ccena_kg/100*(100-Cpostotak), Cpostotak, Cdatum_isteka);
     }
     EXEC SQL CLOSE mag;
     greska("Zatvaranje mag");

     if(ind==0)
       {
        printf("Datog proizvoda trenutno nema na lageru. Unesite id drugog proizvoda ili 0 za
izlaz.\n");
         scanf("%d", &Cid_robe);
         if(Cid_robe==0) break;
       }
```

```
   }
  if(ind!=0)
   {
     float kg;
     printf("Unesite red iz prethodnog menija, odakle zelite da kupite proizvod, kao i broj kilograma, koji zelite da kupite.\n");
     scanf("%d %f", &ind, &kg);

     EXEC SQL OPEN mag;
     greska("Otvaranje mag");
     while(ind>0)
     {
       EXEC SQL FETCH mag INTO :Cime, :Cid_tezge, :Ckg, :Ccena_kg, :Cpostotak, :Cdatum_isteka;          //promenljiva Cime se u ovom slucaju koristi za ime dobavljaca
       greska("Uzimanje iz mag");       //a ne korisnika aplikacije

       if(SQLCODE==100)
       {
         printf("\n");
         break;
         }

       ind--;
     }
     EXEC SQL CLOSE mag;
     greska("Zatvaranje mag");

     while(kg>Ckg)
     {
       printf("Nema dovoljne kolicine proizvoda na tezgi. Molimo Vas da unesete kolicinu kg ponovo.\n");
       scanf("%f", &kg);
     }
     Ckg=kg;

     Ccena_sa_popustom=Ccena_kg/100*(100-Cpostotak);

     EXEC SQL select stanje into :Cstanje
     from kupac
     where id_kartice=:id_kor;
     greska("Nalazenje stanja");

     if(Cstanje<Ccena_sa_popustom*Ckg) printf("Nemate dovoljno novca na racunu.\n");
     else
     {
       EXEC SQL update kupac
```

```
        set stanje=stanje-:Ckg*:Ccena_sa_popustom
        where id_kartice=:id_kor;
        greska("Skidanje novca");

        EXEC SQL update magacin
        set kg=kg-:Ckg
        where id_tezge=:Cid_tezge and id_robe=:Cid_robe and cena_kg=:Ccena_kg and
datum_isteka=:Cdatum_isteka;
        greska("Promena kg");

        EXEC SQL update tezga
        set kasa=kasa+:Ckg*:Ccena_sa_popustom
        where id_tezge=:Cid_tezge;
        greska("Uplacivanje novca");

        EXEC SQL insert into transakcija(id_kartice, novac, id_tezge, id_robe, datum_tr,
vreme_tr)
        values (:id_kor, :Ckg*:Ccena_sa_popustom, :Cid_tezge, :Cid_robe, CURRENT_DATE,
CURRENT_TIME);
        greska("Belezenje transakcije");

        printf("Transakcija je izvrsena! Vase trenutno stanje je %.2f.\n", Cstanje-
Ckg*Ccena_sa_popustom);
      }
    }
  }

  EXEC SQL CONNECT RESET;
  greska("Uklanjanje veze");

  exit(EXIT_SUCCESS);
}
```

Приликом излиставања свих производа, које нуди пијаца, коришћена је индикаторска променљива, поменута у потпоглављу 4.1. Уколико се догоди да одређени производ нема унету величину, уместо ње ће се исписати знак / .

Занимљиво је приметити да се, уколико се погреши приликом уноса променљиве *Cid_robe*, курсор затвара, а у следећем циклусу петље поново отвара. На тај начин се врши враћање курсора на почетак табеле резултата (која се изменила услед измене *Cid_robe*), након чега ће он поново излиставати резултате као да је први пут отворен.

Важно је напоменути да променљива *Cdatum_isteka* није типа *char [9]*, као што би се очекивало (јер се датуми уносе у *SQC* програм у формату *'ggggmmdd'*), већ *char [11]*. Разлог је тај, што се након уноса датума врши имплицитна конверзија, која за интерне потребе *SQC* програма формат *'ggggmmdd'* претвара у формат *'mm/dd/gggg'*, који се потом као такав и исписује. Како новонастали стринг има 10 карактера, потребно је да *Cdatum_isteka* буде типа променљиве *char [11]*.

Након одређивања тезге, на којој ће се одвити трансакција, као и провере да ли жељена количина килограма премашује укупну количину на тезги, уводи се променљива *Ccena_sa_popustom*, која чува цену производа (коју смо добили из курсора) са урачунатим попустом на њу. Трансакција ће бити успешно обављена уколико купац на својој картици има довољно платних средстава да би се трансакција обавила. Обављање трансакције се састоји из четири корака:

- скидање новца са картице купца,
- умањивање броја килограма производа на тезги, на којој се врши трансакција, за купљени број производа,
- уплаћивање новца у касу тезге, на којој се врши трансакција,
- бележење трансакције у табелу *трансакција* е-пијаце, за шта се користе тренутни датум и тренутно време.

Након обављања трансакције купац се обавештава о извршавању трансакције и исписује се његово тренутно стање на рачуну.

6 Закључак и могућности за унапређење базе података

Имајући у виду све веће ослањање света на информатику, податке и базе података, неминовно настаје потреба за унапређивањем ресурса који нам већ стоје на располагању и померањем постојећих граница. Инспирисан тиме, овај рад спаја два суштински различита језика, *SQL* и програмски језик *C*, и истражује могућности њиховог комбиновања.

Показано је како *SQL* не може да оперише са улазним подацима, али се кроз *SQC* програме тај проблем лако превазилази, као у примерима 1 и 2. Показано је и како се може руковати свим инстанцама резултата *SQL* упита помоћу курсора и како се они могу примењивати на најразличитије начине у примерима 2, 3, 4 и 5. На крају је показано како *SQC* програми могу да се користе не само за појединачне задатке и упите, већ и да могу послужити најразличитијим сврхама, као у апликацији е-пијаце.

Следећи принцип унапређивања постојећих знања, овај рад може послужити као темељ за даља истраживања и испитивања. Неке од могућности дораде и измене овог рада су:

- могућност брисања трансакција и повраћаја новца унутар апликације за купце и продавце,
- понуда персонализованих предности за честе купце, као што су попусти на робу, коју одређени купац често купује,
- проширење услуга апликације на друге послове, као што су директор и чистач,
- увођење интерфејса апликације, у којој би корисник, било купац, било продавац, након логовања могао да обавља своје трансакције и има преглед попуста и других погодности.

7 Литература

1. IBM. 2013. *Developing Embedded SQL Applications. IBM DB2 10.5 for Linux, UNIX and Windows.* IBM.
2. Čabarkapa, M. 1996. *C - Osnovi programiranja.* Beograd: Krug.