Univerzitet Džemal Bijedić Mostar

Fakultet informacijskih tehnologija Mostar

Mostar, 03.03.2011

UBP – Uvod u baze podataka

Seminarski rad

POS (point of sale) aplikacija

ver. 1.0.1

Student: **Husremović Ernad, DL 2792**

1. Uvod

Cilj seminarskog iz predmeta “Uvod u baze podataka” (nadalje UBP) je upoznavanje studenta sa osnovnim principima i praksama razvoja database aplikacija - aplikacija za rukovanje relacijskim bazama podataka.

U ovom radu ćemo obraditi tu temu sa primjerom POS (eng. point of sale) aplikacije. U predmetu UBP fokus je na ravno na modeliranju podataka, dok je realizacija aplikacije u funkciji demonstracije korištenja dobijenog modela.

Prvi artifakt u procesu modeliranja baze E-R dijagram. Na osnovu E-R dijagrama konstruišemo R (relacioni) model baze podataka. R dijagram se može implementirati na konkretnoj relacijskog bazi podataka.

2. Relacioni model, teoretske osnove

Relacioni model[1] je osnova implementacije svih relacijskih baza podataka (relational database). Relacioni model baze podataka formalno je definisao Codd 1970-1972[2]. Teoretske osnove Codd-ovog modela su bazirane na matematičkom modelu relacione algebre. Codd definiše tri glavne komponente relacionog modela:

1. strukturalna (the structural component)
2. integritetna (the integrity component)
3. manipulativna - operativna (the manipulative component)

Kada kažemo da radimo modeliranje baze podataka, tada prevashodno mislimo na prve dvije komponente Codd-ovog modela. R model definiše upravo te dvije komponente.

Posljednja komponenta se odnosi se metode na osnovu kojih se podaci u relacionoj bazi podataka (nadalje: baza)[3] kreiraju i mjenjaju, te za potrebe manipulacije “dohvataju”.

Sve današnje baze manipulativnu komponentu realizuju nekom implementacijom SQL-a[4].

2.1 “The Definitive Guide to SQLite” [sqlite] - odličan spoj teorije i prakse

Knjiga me je ugodno iznenadila. Mislio sam da se radi isključivo o priručniku za korištenje SQLite baze podataka. Kako sam se odlučio za implementaciju baze u SQLite-u, koristio sam ovu knjigu. U njoj sam međutim našao čitavo poglavlje o relacionom modelu podataka.

Iako je pisana za SQLite, tematika u knjizi je obrađena tako da čitaocu obezbjedi univerzalna znanja. Knjiga na interesantan način čitaocu objašnjava relacioni model, historijske činjenice i uzroke njegovog nastajanja, te na jedan prirodan način sa teoretskih postavki prelazi na praktična pitanja.

1 [sqlite], poglavlje 3 - “The relational model”

2 Codd's paper: “Relational completness of Database Sublanguages”, 1972

3 Baza podataka ne mora biti relaciona. Posljednjih godina “vruća” tema u informatičkim krugovima su upravo NoSQL baze podataka. S obzirom da u ovom materijalu pričamo isključivo o relacionim bazama podatka, termin “baza” koristićemo za relacionu bazu podataka.

4 SQL - Structured query language

3. Model podataka POS aplikacije, analiza

Da bi došli do odgovarajućeg modela, radimo analizu poslovnih procesa realnog sistema. Posmatranjem tih procesa dolazimo do informacija o organizaciji podataka naše aplikacije.

POS aplikacija prima niz podataka od ERP[5] aplikacije. Njena funkcija je isključivo evidencija prodaje na jednom prodajnom mjestu (prodavnici).

Od ERP aplikacije POS prima podatke o artiklima.

Svaki artikal ima svoju jedinstvenu šifru, opis, jedinicu mjere i cijenu.

Svi ovi elementi se definišu u ERP aplikaciji prilikom zaduženja robe u prodavnicu ili nivelacije cijena.

POS od ERP aplikacije kao ulazni podatak prima listu artikala sa ažuriranim cijenama.

Račun se u našoj POS aplikaciji izdaje isključivo za neimenovanog kupca.

POS račun sadrži slijedeće elemente:

Datum, Mjesto izdavanja računa, Adresu prodajnog mjesta, i broj. Nakon toga slijede stavke računa:

- Redni broj
- Jedinica mjere
- Količina
- Cijena
- Iznos

Na kraju računa navodimo:

- ukupna vrijednost računa bez PDV-a
- iznos PDV
- račun sa uračunatim PDV-om

Evidencija po vrsta plaćanja (kartica, gotovina, banka) nas u našoj POS aplikaciji ne interesuje - sav promet smatramo gotovinskim.

3.1 Primjer računa:

```
bring.out doo Sarajevo
PDV Broj: 123456789012
Prodavnica 1
Juraja Najtharta br. 3
tel 033/269-290
fax 033/269-292
REDOVNI Račun br: 798
Sarajevo, 24.02.2011
--------------------------------------
R.br  Sifra Naziv(jmj)
                  kol. cij. ukupno
--------------------------------------
1.    SP9 instalacija i podešenje (h)
                  2    100   200.00
2.    POS bring.out POS aplikacija (kom)
                  1    500   500.00
--------------------------------------
ukupno bez PDV              598.29
iznos PDV (17%)             101.71
--------------------------------------
Ukupno sa PDV               700.00
--------------------------------------
```

5 ERP - Enterprise resource planning http://en.wikipedia.org/wiki/Enterprise_resource_planning

Treba naglasiti da je stopa PDV-a za svaki artikal jedinstvena i iznosi 17%[6].

Rezimirajmo podatkovnu strukturu **računa**:

- Račun ima broj, datum, iznos bez pdv, iznos pdv-a, iznos sa PDV-om.
- Račun se sastoji od stavki koje opet imaju redni broj, šifru artikla, naziv artikla, količinu, cijenu, ukupna vrijednost stavke.

Podaci o nazivu firme, prodajnog mjesta, su **parametri** aplikacije. Oni su identični za svaki račun:

- Naziv firme
- PDV broj (12 cifri)
- Naziv prodavnice
- Adresa prodavnice
- telefon
- fax

Svi gornji podaci su opisnog karaktera.

3.2 Unos računa

Sam sadržaj računa nam ukazuje na operacije koje predviđa unosa računa:

- određuje tip računa: Redovni ili reklamirani račun(*).
- operater unosi šifru ili barkod artikla
- količina. količina mora biti > 0
- program utvrđuje cijenu i naziv na osnovu unesene šifre, te ispisuje trenutnu kupno vrijednost računa
- unosom šifre "X" završava se unos računa.

(*) Reklamirani račun u knjigovodstvenom smislu predstavlja storno računa (povrat količina i povrat novca)

Na osnovu iznešenog, dolazimo do strukture entiteta **artikli**:

- Artikli imaju jedinstvenu šifru (obavezno), barkod (koji nije obavezan, ali je takođe jedinstven na nivou liste artikala), jedinicu mjere, cijenu (sa uračunatim PDV-om).

Dodajmo i jedno ograničenje (constraint) na nivou domena atributa cijena:

- Prema Zakonu o trgovini u FBiH cijena sa PDV-om mora biti izražena u minimalnim novčanim apoenima (5 pfeninga) [bout_r_1].

Gornji constrant cijene možemo iskazati odgovarajućim SQL izrazom. Evo kako bi to na nivou sql-a mogli riješiti:

```
sqlite> select case (5.85 * 100 % 5) when 0 then 5.85 else -9999 end;
5.85

sqlite> select case (5.82 * 100 % 5) when 0 then 5.82 else -9999 end;
-9999
```

3.2.1 Triger artikli_5f, artikli_5fu

Kasnije sam našao implementaciju ovog ograničenja na nivou baze[7]. Rješenje je kreiranje trigera.

Triger (okidač) je komanda koja se izvršava (before/after) prije/poslije određenog događaja na bazi podataka:

- insert
- update

6 POS rješenje iz realnog svijeta bi trebalo omogućiti definiciju različitih stopa poreza za pojedine artikle, ali ćemo i ovo pojednostaviti - pretpostavljamo jedinstvenu stopu 17%.

7 http://linuxgazette.net/109/chirico1.html

- delete

Naš triger obezbjeđuje da artikle koji ne zadovoljavaju pravilo 5pf ne ulaze ulaze u tabelu artikli:

```
sqlite> create trigger artikli_5f after insert ON artikli
   ...> begin
   ...> update artikli set cijena=(select case (new.cijena * 100 % 5) when 0 then new.cijena else -9999 end) where artikal_id=new.artikal_id;
   ...> delete from artikli where cijena<0;
   ...> end;
sqlite> create trigger artikli_5fu of cijena ON artikli
   ...> begin
   ...> select case (NEW.cijena * 100 % 5) when 0 then NEW.cijena else RAISE(ROLLBACK, 'cijena na p5f')  end;
   ...> end;
```

Test trigera:

```
sqlite> select cijena from artikli where kod='POS';
5.55
sqlite> update artikli set cijena=5.54 where kod='POS';
Error: cijena na p5f <<<<<<<<<<<<<<<<<<<<<<<<< triger u funkciji
sqlite> select cijena from artikli where kod='POS';
5.55
```

3.2.2 Centralni entiteti: računi, rn_stavke

Jedan račun sadrži jednu ili više stavki. Stavka računa pripada jednom i samo jednom računu.

Stavka računa sadrži tačno jednu šifru artikla. Svaki artikal može da se pojavi u više stavki, ili nijednom (slučaj da artikal nije do sada prodavan).

Nakon što smo hodali “malo 'vamo - malo tamo” između konkretnih operacija i opisa podataka, vrijeme je da lociramo konkretne entitete:

Uočavamo da je centralni entitet Račun (**racuni**) sa pripadajućim stavkama (**rn_stavke**).

"racuni" ima sljedeće atribute:

- *racun_id [8]
- broj
- datum
- ukupno_s_pdv
- tip

Uočimo sljedeće stvar:

- evidentiran je iznos ukupno_s_pdv bez iznosa pdv-a i iznosa ukupno bez pdv. Radi se o zavisnim veličinama u odnosu na ukupno_s_pdv, tako da su ona nepotrebna. Očigledno je da bi njihova egzistencija takođe ugrozila 2NF.
- dodan je atribut "id" koji će biti ujedno i primarni ključ entiteta **racun**. Iako je ovo polje striktno gledajući suvišno, u praksi se pokazuje da je bolje da vidljivi broj računa bude neovisan atribut kome aplikacija daje poseban broj.
  Taj broj se može na dnevnom ili mjesečnom nivou resetovati. Ako uključimo mogućnost resetovanja broja računa, onda kandidat za ključ može biti samo kombinacija (broj, datum). Zato je “skriveni” brojač (id, autoincrement) je “najčišće” rješenje, tako da ćemo njega odabrati.

8 Znak “*” sa lijeve strane znači primarni ključ.

Analizirajmo atribut **tip** entiteta **racun**. Tip ćemo kodirati sa dvije vrijednosti:

- 1 - redovan račun
- 2 - reklamirani račun

Uvođenje nekog entiteta "tipovi" je moguće, ali nepotrebno. Treba samo definisati ograničenje (constraint) na nivou domena atributa tako da tip mora biti jedan od ove dvije vrijednosti (posljedično naravno NOT NULL).

entitet **rn_stavke**:

- *rn_stavka_id
- racun_id
- artikal_id
- kolicina
- cijena

Ovdje eliminišemo "ukupno" za račun kao zavisni atribut[9]. Istim principom eliminišemo i "jmj" i "barkod" kao atribute koje u stvari "čupamo" iz entiteta "roba".

Interesantno je uočiti da je cijena stavljena kao atribut **rn_stavke**. Zašto, ako se i ona se “uzima” iz entiteta roba !?

Cijena je, za razliku od "jmj" i "barkod", dinamična vrijednost. Svako novo zaduženje ili nivelacija mogu promijeniti ovu veličinu. Zato moramo evidentirati cijenu po kojoj je račun izdan unutar rn_stavke.

Time dolazimo do entiteta **artikli**:

- *artikal_id
- kod [UNIQUE]
- barkod [UNIQUE]
- jmj
- cijena
- naziv

Gotovo sve baze omogućava setovanje UNIQUE constraint-a za višestruka polja. Ispitao sam i utvrdio da moja ciljna baza sqlite3 to takođe podržava([sqlite], Data Integrity/Entity integriti str. 128). Zato ću postaviti još jedno ograničenje jedinstvenosti para (naziv, cijena).

Naime, ne treba ograničavati da se naziv artikla pojavljuje višestruko, ali postojanje višestrukih kombinacija (cijena, naziv) mogle bi zbuniti korisnika. Zato ovakvo ograničenje nije zgorega napraviti.

3.3 ostali entiteti

Entitet **parametri**:

- *parametar_id
- svrha (unique - jedinstveno na nivou entiteta)
- opis

Svrha ima set mogućih vrijednosti koje ćemo za potrebe naše aplikacije predefinisati - kodirati:

- F => opis sadrži Naziv firme
- P => PDV broj (12 cifri)
- N => Naziv prodavnice
- A => Adresa prodavnice
- T => telefon
- U => fax

9 Pravila vertikalne normalizacije, 2NF

Atribut **”opis”** ćemo odrediti prema "najširem" sadržaju. To je očigledno Adresa prodavnice. 200 znakova za širinu polja opis će sigurno biti dovoljno. Ovaj podataka nan za E-R dijagram ne treba, ali zašto čekati prelazak na R model da bi nešto konstatovali :)

Kod ove tabele, bili smo bili poprilično "raskalašni" sa stanovišta prostora. Zašto ?

Zato što se radi o tabeli koja će sadržati samo 6 redova. Nekakvo modeliranje koje bi štedilo na prostoru samo bi zakomplikovalo stvar, a dobitak bi bio upitan.

Kod ove tabele je bitno uočiti ograničenje na nivou tabele/entiteta - svrha je "UNIQUE".

3.3.1 Haj'mo stvari malo zakomplikovati - reklamirani račun

S obzirom da je ovo seminarski rad, dobro je navesti što više različitih relacija među entitetima-tabelama. Zato ću dodati još jednan uslov koji je potrebno obuhvatiti modelom:

Ukoliko je račun **tipa** reklamirani, potrebno je predvidjeti mogućnost navođenja originalnog računa koji se reklamira. Ovo polje, međutim nije obavezno.

Jedan račun može imati jedan ili nijedan originalni račun.

"Iz aviona" se vidi da imamo rekurzivnu vezu reklamiranog racuna sa njegovim originalom.

Račun može imati nula ili jedan originalni račun (0..1). U obrnutom smjeru, originalni račun može imati jedan i samo jedan reklamirani račun.

3.3.2 Dodajmo još koji entitet - “operateri”

U "erviz"-u [erviz] sam napravio ER dijagram prema gornjoj specifikaciji. Četiri entiteta i dvije veze između njih. Još malo k'o u Bosni entiteta ... Taman tol'ko entiteta traži Čović. Bosni su i dva entiteta previše, ali je za seminarski ovo fakat malo. Da mi kolege nastavnici ne bi vratili seminarski sa komentarom: "Jel' ti to nas zezaš ?!", nastavljam priču priču o POS aplikaciji ...

POS aplikaciju može opsluživati više operatera. Jedan račun je izrađen od strane jednog i samo jednog operatera. Jedan operater može izraditi više računa, ili nijedan.

operateri:

- *operater_id
- naziv

Očigledno je da u entitet "racuni" trebamo dodati atribut **operater_id**.

3.3.3 Podrška fiskalizaciji

Po Zakonu o fiskalizaciji, nijedan POS software ne može više funkcionisati samostalno - direktno štampati na printer. POS software se mora priključiti na fiskalni uređaj. POS aplikacija se sa fiskalnim sistemom vezuje putem fiskalnog drajvera. Taj drajver kao ulazne podatke prima podatke o računu, a kao povratnu informaciju daju broj fiskalnog odsječka, ako je operacija uspješna.

Zato uvodimo entitet "**rn_fiskalni**" koji će sadržavati informacije o procesu slanja računa na fiskalni sistem. POS aplikacija šalje račun fiskalnom uređaju. To se može desiti u više pokušaja. S obzirom da se prvo formira račun, moguće je da postoji račun koji još nije poslan na fiskalni uređaj. Obrnuto, za jednu stavku račun u entitetu "rn_fiskalni" postoji tačno jedan račun koji se fiskalizira.

Datum i vrijeme svakog pokušaja treba zabilježiti.

Ako je fiskalizacija neuspješna, od fiskalnog drajvera dobijamo ne dobijamo broj fiskalnog isječka (NULL). U slučaju uspješnog procesiranja našeg računa, dobijamo od fiskalnog drajvera broj fiskalnog isječka > 0. Pored broja fiskalnog isječka, dobijamo i podatak o ukupnom iznosu fiskalnog računa.

Taj iznos bi se trebao odgovarati ukupnom iznosu sa pdv na našem "običnom" računu. Razlike usljed zaokruženja su moguće (+/- 0.1 - 0.2 KM).

rn_fiskalni:

- *racun_id
- *datum
- *vrijeme
- br_fiskalnog_isjecka [UNIQUE]
- ukupno_s_pdv

Pošto je svaka stavka u run_fiskalni egzistencijalno ovisna o računima, "**rn_fiskalni**" je "slabi" (eng. weak) entitet.

Na dijagramu po IE notaciji on je prikazan kao pravougaonik zaobljenih ivica.

4. Pristup i zapažanja kod analize modela baze podataka

Odlučio sam se da seminarski radim onako kako to u svakodnevnoj praksi - realnom životu, database programeri rade.

Naime, prilikom analize sistema, informatičar dobija masu suvišnih ili nedovoljno preciznih informacija.

Njegov zadatak je, posebno u fazi analize i dizajna, da te informacije non-stop filtrira i "premeće".

Pri tome je od ključne važnosti da se informatičar što bolje upozna sa poslovnim domenom koji obrađuje.

Iako seminarski rada ne obrađuje realni sistem (ista osoba radi i analizu problema i definiše poslovni domen), želio sam ovdje ukazati na "koncept upgrade"-a koji je vrlo bitan sa metodološke strane ovog posla.

Naime, ja sam bilježio razradu modela, onako kako sam sam sa sobom pričao. Svaka nova informacija predstavljala je upgrade modela koji konstruišem U prvim verzijama nisam imao podršku za operatere i fiskalni sistem.

Ovaj proces nadogradnje modela usljed promjene korisničkih zahtjeva, zakonskih promjena ili jednostavno novih saznanja o sistemu je svakodnevan.

Zato ga i ne treba ispuštati. Bilješke programera, čak i **pogrešni zaključci** koji su očekivani u početnim fazama su veoma korisni za razvoj i održavanje sistema koji konstruišemo[10].

4.1 Upgrade software-a i tehničke dokumentacije su jedan posao

Informatičari najčešće dokumentaciji prilaze "otpozada" - naprave to što su zamislili pa onda pišu tehničku dokumentaciju.. I ja to nerijetko činim. Takav pristup mi se redovno "razbije o glavu".

To je pogrešna praksa. Software je, to će se svako složiti, proizvod koji je predmet stalnog "upgrade"-a. Taj isti princip treba primjeniti kod dokumentacije. Tačnije upgrade software i upgrade tehničke dokumentacije trebaju se smatrati **jednim poslom**.

Software bez dokumentacije je loš proizvod.

To je razlog zbog koga sam u toku pisanja seminarskog bilježio stvari na onaj način kako su se razvijale. To tehničkoj dokumentaciji projekta, makar on bio i hipotetski, daje dodatnu težinu.

4.2 Niko nije dobar napravio software ako to isto ne zna "uraditi na papiru"

Poznavanje poslovnih procesa je neophodno za uspostavljanje ispravnog modela baze podataka.

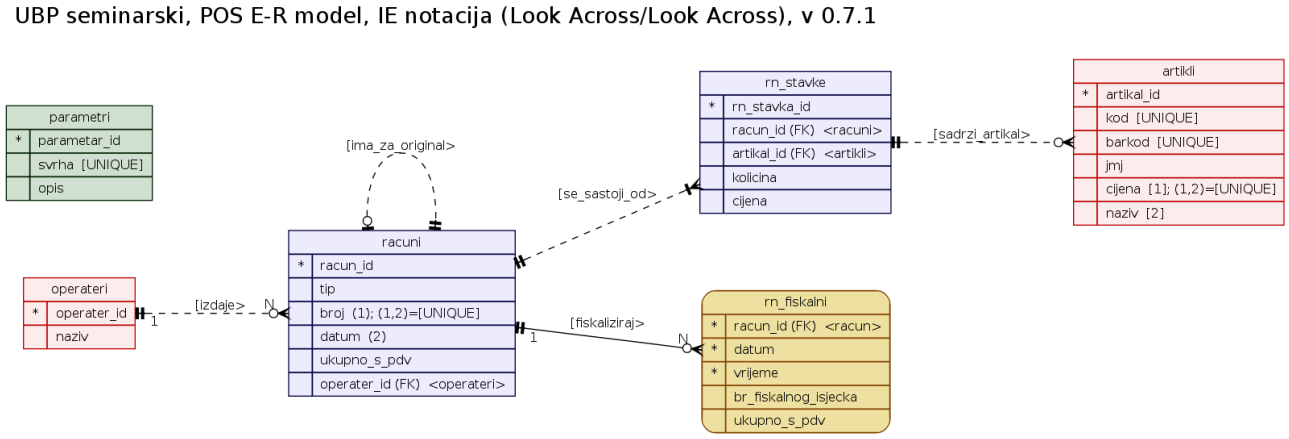
Iskustvo dizajnera baze podataka je bitno u procesu modeliranja. Redundatnost podataka je u mnogim situacijama bolja od "puritanskog" modeliranja. U našem konkretnom primjeru, entitet **racuni** mogao bi egzistirati bez id polja sa višestrukim primarnim ključem (broj, datum), ali mi iskustvo govori da je neovisno polje racun_id bolji izbor. Njegovo uvođenje neznatno povećava bazu[11], dok sa druge strane unutar aplikacije dobijamo jednostavnost manipulacije.

10 Velika je vjerovatnoća da će se greške koje su napravljene, a nisu dokumentovane ponovo desiti. Posebno ako na projektu radi više ljudi. Svaki softverski projekat je inkrementalan - postupan procesa. Stoga je ispuštanje informacija o pojedinačnim fazama loša praksa.

11 Čak i to je upitno, so obzirom da se po primarnom ključu "racuni" povezuje "sa rn_stavke", a ovo su tabele sa najvećom količinom podataka

5. Naš prvi artifakt: E-R model

Na osnovu 4. je uz pomoć programa erviz napravljen E-R model[ermodel][12]. Korištena je IE notacija:



Na E-R modelu su crvenom bojom označeni entiteti koje POS aplikacija dobija od ERP aplikacije.

U "centru" POS aplikacije su entiteti označeni **plavom bojom**: "racuni" i "rn_stavke".

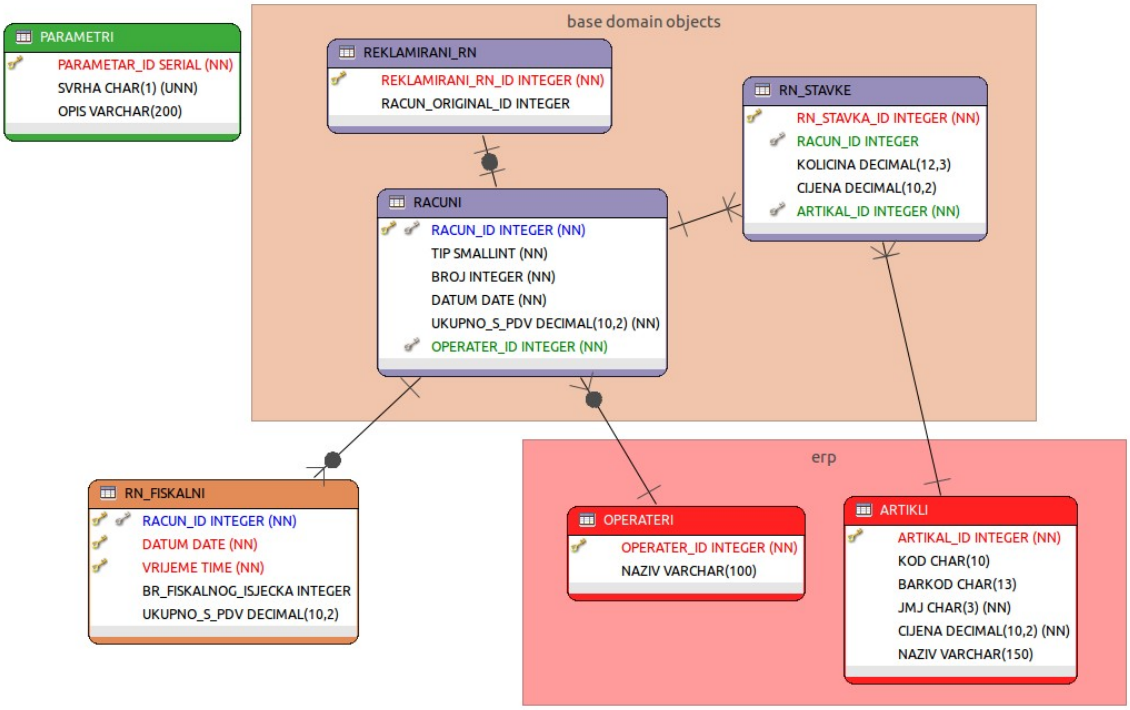
Zabilješke i zapažanja:

- Meni je ova notacija puno pregledanija od Chen-ove notacije koje smo koristili na nastavi. Chen-ovi “oblačići” za prikaz atributa zauzimaju veliki prostor :). Simboli koji se koriste za završetke veza u IE notaciji su krajnje intuitivni
- Iako sam ovu notaciju u početku koristio radi erviz alata, na kraju mi je IE notacija “pravo sjela”.

12 https://github.com/hernad/FIT_UBP/blob/master/seminarski/pos_ubp.pdf

5. Artifakt No2: R model

U ermaster[ermaster] na osnovu E-R modela napravio R model. S obzirom da sam sqlite odredio za ciljnu bazu, kod dizajna sam rekao da se R model dizajnira za Sqlite. Na osnovu ovog podešenja "ermaster" nudi atribute relacija (odnosno kolone tabela SQL jezikom) u skladu sa onim što konkretna baza ima implementirano.



Na osnovu napravljenog gornjeg dijagrama navodim komponente R-modela[13]:

5.1 Strukturalna komponenta - Relacije (relations)

1. parametri (parametar_id, svrha, opis)
2. operateri (operater_id, naziv)
3. racuni (racun_id, tip, broj, datum, ukupno_s_pdv, operater_id)
4. reklamirani_rn (reklamirani_rn_id, racun_original_id)
5. rn_stavke(rn_stavka_id, racun_id, artikal_id, kolicina, cijena)
6. artikli(artikal_id, kod, barkod, jmj, cijena, naziv)
7. rn_fiskalni(racun_id, datum, vrijeme, br_fiskalnog_isjecka, ukupno_s_pdv)

Komentar:

- kao rezultat rekurzivne veze entiteta racuni u E-R dijagramu nastala je nova relacija reklamirani_rn

13 Da se ne radi o seminarskom, R model ne bih navodio, jer ermaster model je upravo dobijen prevođenjem E-R modela u R.

5.2 Integritetna komponenta - Veze između Relacija (relationships)[14]

1. racuni(racun_id) << reklamirani_rn(reklamirani_rn_id)
2. rn_stavke(racun_id) << racuni(racun_id)
3. rn_stavke(artikal_id) << artikli(artikal_id)
4. racuni(operater_id) << operateri(operater_id)
5. rn_fiskalni(racun_id) << racuni(racun_id)

Napomene:

- gore su pobrojana isključivo interrelacijska ograničenja
- integritetnoj komponenti pripadaju i sva ograničenja na nivou atributa/kolone. Tako npr ograničenje cijene na 5pf može biti ograničenje na nivou kolone[15] ili NOT NULL ograničenje.
- takođe, integritetnoj komponenti pripadaju i sva ograničenja na nivou relacije/entiteta/tabele (npr. UNIQUE ograničenja po poljima ili kombinacijama polja)

*5.3 Integritetna komponenta **racun.ukupno_s_pdv** (trigger: rn_stavke_insert)*

Uz pomoć ovog trigera kolona ukupno_s_pdv tabele računi se ažurira:

```
sqlite> create trigger rn_stavke_insert after insert on rn_stavke
   ...> begin
   ...> update racuni set ukupno_s_pdv = ukupno_s_pdv + new.kolicina * new.cijena where racuni.id=new.racun_id;
   ...> end;
```

Zahvaljujući ovom trigeru, aplikacija se uopšte “ne brine” sadržaju polja **ukupno_s_pdv** tabele racuni.

14 Koristiću notaciju: *r1(fk_1) << r2(pk_2) znači: relacija r1 sa stranim ključem pk_1 je povezana sa relacijom r2 sa primarnim ključem pk_2*

15 Ovo baš i nije uobičajeno raditi na nivou baze jer se zakoni mijenjaju. Navodim to prije svega kao primjer.

6. Aplikacija

6.1 Inicijalni podaci

Aplikacijski dio sam započeo sa pravljenjem pos_db.py[16] koji ima sljedeće funkcije (__main__):

- puni tabele koje u realnom okruženju dostavlja ERP aplikacija
- testira osnovne funkcije baze
- pos_db.py je back-end web klijenta (svo handliranje bazom dešava se isključivo u pos_db.py - on je business + data layer)

6.2 Web klijent

Odabir baze je bio lagan posao. Međutim, kod odabira tehnologije za kljenta razmatrao sam puno varijanti. Na kraju sam odlučio na web - HTML klijenta iz sljedećih razloga:

- standard
- podržano na svim softverskim sistemima
- nikada nisam pisao "hard core" web dinamičku aplikaciju[17]

Iako mi je u startu bilo jasno da će aplikacija biti "ružno pače", na kraju priče mislim da je web najbolji izbor za ovakvav rad.

S obzirom da nisam koristio nikakav application framework, puno toga sam naučio o baznim konceptima web programiranja.

6.2.1 server-side validacija

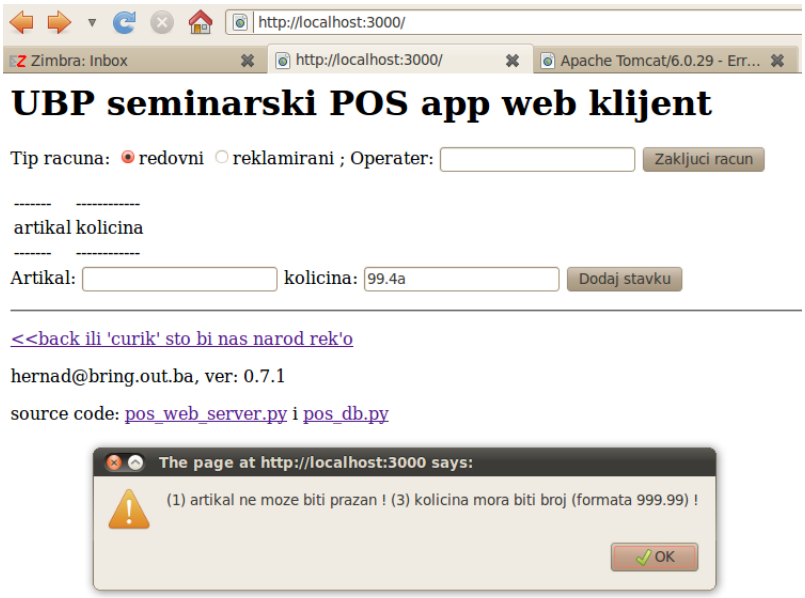
16 https://github.com/hernad/FIT_UBP/blob/master/seminarski/py/pos_db.py

17 neki web framework (rails, django) bi mi sigurno značajno olakšao život, ali svaki od tih frameworka u database dijelu ima neki ORM (object relational manager). ORM povećava nivo apstrakcije, što u svru učenja nije dobro. Išao sam logikom da je bolje doći do rješenja rudimentarih funkcija izgleda, a da se što više toga proba i nauči.

6.2.2 Client-side validacija

S obzirom da je HTML/web, napravio sam par kontrola unosa na klijentskoj strani:

1. provjera da polje artikal, količina nisu prazni
2. provjera da količina ne sadrži alfanumerike[18]



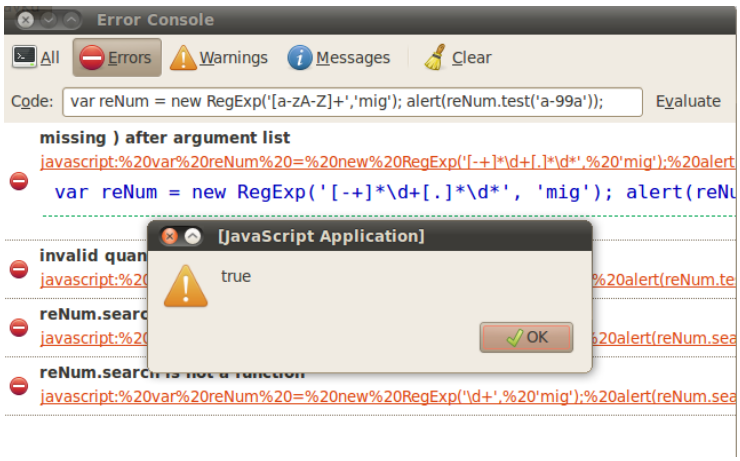
Primjer: Jednostavna client-side validaciju količine sa regular expression-om:

javscript snippet:

```
var reNum = new RegExp('[a-zA-Z]+', 'mig');
if (reNum.test(kolicina)) {
  err = err + " (3) kolicina mora biti broj (formata 999.99) !";
  ret = false;
}
```

6.2.3 Firefox javascript konzola je moj "friend"

Puno mi je pomogla firefox javascript konzola. Firefox ima odličan javascript debugger:



18 [javascript] Chapter 9 - Data validation techniques

6.3 Unos podataka

Procedura unosa podataka je sljedeća:

1. unosimo artikal/kolicina. Za završetak unosa aktiviramo dugme “dodaj stavku”
2. kada završimo unos, unosimo parametre računa (tip, operater)
3. za ažuriranje aktiviramo dugme “zakljuci racun”
4. Pri gornjim operacijama vrše se sljedeće validacije:
    1. validacija artikal, količina (client side)
    2. postojanje šifre artikla (server-side)
    3. postojanje operatera sa zadanim imenom (server-side)
5. Po uspjeđnom unosu operater dobija prikaz pos računa kao na slici:

Brojač računa se resetuje na dnevnom nivou. Odgovarajući python code[19]:

```
def get_next_broj_racuna(self, datum):
   # nadji najveci za ovaj datum pa dodaj +1
   print "datum=", datum
   ret = self.cursor.execute("select max(broj) from racuni where datum=:datum",
         {'datum': datum} ).fetchone()[0]
   if not ret:
      print "na datum ", datum, "nema racuna, krecemo od 1"
      return 1
   else:
      print "max broj racuna je", ret, " za datum ", date
      return ret+1
```

Pravimo upit na bazu koji nalazi maksimalni broj računa za željeni datum. Taj broj uvećavamo za 1.

19 https://github.com/hernad/FIT_UBP/blob/master/seminarski/py/pos_db.py

6.4 Izvještaji

6.4.1 Račun /report/racun

Zadavanjem URL-a: http://localhost:3000/**report**/**racun**/**1**

gdje je:

- /report/ - za sve izvještaje
- /racun/ - izvještaj račun u toku dana
- 1 - račun broj 1

Znači, izvještaj je ograničen na račune izdate u toku dana:

```
bring.out doo Sarajevo
PDV Broj: 0123456789012
Prodavnica 1
Juraja Najtharta 6
tel: 033/269-291
fax: 033/269-292
REKLAMIRANI Racun br. 1
Sarajevo, 2011-02-28

----------------------------------
R.br Sifra       Naziv(jmj)
              kol.    cij. ukupno
----------------------------------
  1. bring.out POS software(kom)
POS             2     440       880
  2. bring.out instalacija POS(kom)
SP9             1     100       100
----------------------------------
Ukupno bez PDV               837.61
iznos   PDV                 142.39
----------------------------------
Ukupno sa PDV                 980.0
```

6.4.2 Kartica prodaje artikla - /report/kartica_prodaje

http://localhost:3000/**report**/**kartica_prodaje**/**POS**

I ovdje URL koristimo za definisanje parametara izvještaja:

- /report/ - izvještajni dio
- /kartica_prodaje/ - izvještaj kartica prodaje
- POS - kod artikla čiju karticu želimo prikazati

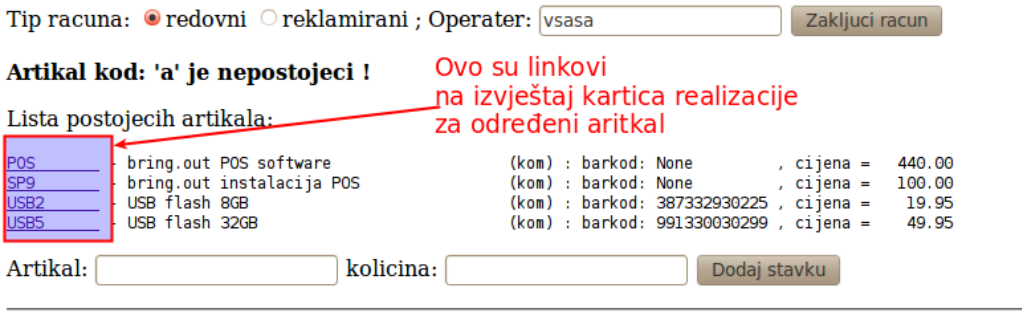
```
Izvjestaj: Kartica prodaje artikla
----- ---------- ---------- ------------ ---------- ---------- ----------
Artikal: POS bring.out POS software
----- ---------- ---------- ------------ ---------- ---------- ----------
  r.br   datum   racun broj        cijena     kolicina   stanje   ukupno

----- ---------- ---------- ------------ ---------- ---------- ----------
   1. 2011-02-27          1          440          2          2        880
   2. 2011-02-27          2          440          3          5       2200

 ....
  13. 2011-02-27         34          440         -3         65      28600
```

```
 14. 2011-02-27         37          440          1         66      29040
 15. 2011-02-27         38          440          2         68      29920
 16. 2011-02-27         49          440        0.5       68.5    30140.0
 17. 2011-02-27         50          440       -0.5       68.0    29920.0
 18. 2011-02-27         53          440          1       69.0    30360.0
 19. 2011-02-28          1          440         -2       67.0    29480.0
----- ---------- ---------- ----------- ---------- ---------- ----------
                                                     67.0    29480.0
----- ---------- ---------- ----------- ---------- ---------- ----------
```

6.4.3 Reporti putem URL-ova su 'cool' stvar

Kada sam napravio izvještaj kartice prodaje, uočio sam da mogu napraviti jednostavnu vezu između liste artikala i ovog izvještaja, zahvaljujući poziva putem URL-a. To sam implementirao na listi artikala.

Ova lista se dobije u slučaju unosa nepostojećeg artikla:



6.5 Šta nije implementirano u aplikaciji

- U aplikaciji ništa nije rađeno po pitanju podrške fiskalizaciji (3.3.3)
- reporting sistem bi brat-bratu mogao imati još jedno 4-5 izvještaja. Međutim, u postojećoj implementaciji napravljena je dobra osnova za jednostavno dodavanje novih izvještaja (6.4.3)
- Iako web aplikacija, ona je sa stanovišta unosa podataka - nisu podržane izolovane sesije za različitog korisnika. To se može uraditi tako što će podaci o računu biti posebni za svakog operatera.
- Web aplikacija je skroz otvorena - ne postoji nikakav sistem logiranja - zaštite
- handliranje grešaka u slučaju nepostojećih podataka i slično uglavnom nije riješeno[20].

7. Zašto ?

7.1 Zašto nisam radio po uputama nastavnika ?

- Zato što je moje radno okruženje Linux. MS Access nema podršku za linux, tako da bi mi rad na seminarskom iz ovih razloga bio otežan[21].
- Zato što nemam licencu za korištenje MS Access-a, a ne želim koristiti nelegalan sofware[22].
- Zato što smatram da je MS Access loš za učenje osnova relacijskih baza podataka[23]

---

20 Recimo ovo će izbaciti grešku: http://fit.bring.out.ba/ubp/report/racun/99999, ali će aplikacija nastaviti rad i nakon greške.

21 Morao bih obezbjediti “mahsuz” Windows mašinu, pa onda se sa svog radnog desktopa prebacivati na taj ... To bi mi uveliko otežavalo rad. Pored dodatnih materijalnih resursa, podjednak problem leži u činjenici da kao sam za klasu produktivniji na sistemu na kome svakodnevno radim - Ubuntu/Linux.

22. Korištenje nelegalnog software-a je haram. Meni kao muslimanu to nije dozvoljeno koristiti.

- Na kraju, ali najbitnije, zato što smatram da FIT kao javna obrazovna institucija studenta ne smije ograničavati na jednu platformu i rješenja pojedinih vendora gdje god za to nema potrebe[24].

7.2 Zašto python ?

- Zato što je otvoren software
- Zato što je zastupljen na svim značajnim platformama - multiplatformski
- Zato što je jedan od oficijelnih programskih jezika za razvoj “google” aplikacija
- zato što ga koristi knowhow ERP[knowhow]

7.3 Zašto SQLite ?

- Zato što je sjajna SQL baza za učenje relacionih baza
- Zato što je otvoren software
- Zato što je zastupljen na svim značajnim platformama
- Zato što je web SQL storage svih implementacija HTML5 Storage standarda

8. Zaključak

Rezultat mog rada je jedna “hello world” database aplikacija[25].

Aplikacija je napravljena isključivo na otvorenim, multiplatformskim softverskim alatima i tehnologijama. Nisu korišteni nikakvi aplikacijski framework-i.

Iako je krajnji database model krajnje jednostavan, mislim da sam ipak postiga sve glavne nastavne ciljeve koji je ovaj seminarski pretpostavio.

Došao sam do cjelovitog toolseta za dizajniranje i implementaciju sistema: erviz, ermaster, sqlite3, python su odlični alati za edukaciju iz tematike koju pokriva UBP.

Zadovoljan sam što sam postigao da je rad jedna logična cjelina. Izbjegao sam izradu na principu “praktičnog” i “teoretskog” dijela rada. Mislim da sam taj cilj ostvario, što smatram najvećim rezultatom ovog rada.

Svjestan sam da da je rad napravljen mimo smjernica koje su nastavnici dali (MS Access). Međutim, moje očekivanje je da će kolege nastavnici uvažiti činjenicu da je u rad uložena velika energija i trud.

Nadam se da će ovaj rad biti koristan i interesantan za svakoga ko ulazi u tematiku osnova baza podataka.

23 http://hernad.bring.out.ba/uvod-u-baze-podataka-ms-access-je-je-pogresan
24 http://hernad.bring.out.ba/mogucnost-izbora-je-takode-stvar-koja-se-uci
25 http://hernad.bring.out.ba/hello-world

9. Reference

1. [fit_fernala] Seminarski rad iz UBP-a “Farma krava” , Student: Fernala Sejmen-Banjac, 2517 DL
2. [bout_r_1] http://redmine.bring.out.ba/boards/8/topics/4342
3. [comp_anal] www.ischool.drexel.edu/faculty/song/publications/p_Jcse-erd.PDF
4. [erviz] http://www.ab.auone-net.jp/~simply/en/index.html
5. [ermaster] http://ermaster.sourceforge.net
6. [ermodel] https://github.com/hernad/FIT_UBP/blob/master/seminarski/pos_ubp.pdf?raw=true
7. [javascript] “Beginning JavaScript with DOM Scripting and Ajax”, izdavač Apress 2006, autor Christian Heilmann
8. [knowhow] Knowhow ERP opensource projekat http://redmine.bring.out.ba/projects/knowhow/wiki
9. [python] “The Quick Python Book”, second edition, izdavač Manning 2010, autor Vernon L. Ceder
10. [sqlite] "The Definitive Guide to SQLite", izdavač Apress book 2006, autor Michael Owens

10. Radno okruženje

- Radna stanica OS: Ubuntu desktop 10.04
- Aplikacije:
  - Libreoffice 3.3.0
  - Eclipse 3.5.2
  - vi editor
- erviz 1.0.6
- ruby 1.8.7 sa irb utility-jem
- sqlite3 ver. 3.6.22
- python 2.6.5
  - Package: python-pysqlite2 , ver: 2.5.5-3

11. artifakti na github.com

Na lokaciji nalaze se svi artifakti ovog rada: *https://github.com/hernad/FIT_UBP/tree/master/seminarski*

12. fit.bring.out.ba/ubp

Aplikacija je “podignuta” na ovoj adresi. Računi se mogu dodavati, izvještaji pregledati.

Primjer:

- kartica prodaje artikla SP9: http://fit.bring.out.ba/ubp/report/kartica_prodaje/SP9
- račun broj 1 na današnji dan[26]: http://fit.bring.out.ba/ubp/report/racun/1

26 U slučaju da računa ne postoji program će prijaviti grešku - handliranje grešaka također nije rješeno

Prilog 1: erviz E-R dijagram generator

10.1 višestruke veze između dva entiteta, erviz tutorial 09

source kod za ovaj dijagram je:

```
{title: "višestruke relacije"}
[Pošiljka]
*id
od_posta_id*
prema_posta_id*
težina
opis

[Pošta]
*id
naziv
# veze između entiteta
[Pošiljka] *--1 [Pošta] <poslano od->
[Pošiljka] *--1 [Pošta] <poslano prema->
```

10.2 oznaka kardinaliteta[27]

?	0..1	kardinalnost jedan, opciono
1	1	kardinalnost jedan, obavezno
*	0..N	kardinalnost više, opciono
+	1..N	kardinalnost više, obavezno
-	-	nije specificirano

Moje bilješke i zapažanja:

- IE - look accross notacija za kardinalitet (max brojnost) i participaciju (minimalna brojnost veze)
- U Komparativnoj analizi E-R dijagrama[comp_anal] stoji da kod Chen -a participacija koristi "Look Here" notaciju, a kardinalitet "Look Across" notaciju.
- Interesantno je da se u FIT materijalima se koristi kombinacija[comp_anal](Str. 10, slika 3, opcija "e" na slici) koja je "Look Here"/"Look Here" notacija za participaciju i kardinalitet respektivno.
- Kada sam uporedio dva dijagrama (Chen i IE), moj dojam je da je grafički prikaza sa "Look Accross/Look Across" (LA/LA) prirodniji za čitanje.
- Ustvari, vrlo bitno za preglednost E-R dijagrama je završetke veza “ukrasiti “odgovarajućim grafičkim simbolima kako je to kod IE notacije urađeno.
- LA/LA shema kod 1-N relacije pokazuje višestrukost na onoj tabeli koja sadrži N stavki sa poljem po kojem se relacija veže - tabela koja sadrži FK.

27 http://www.ab.auone-net.jp/~simply/en/works/erviz/references/cardinality-symbols.html

Prilog 2: Konačna šema SQLite pos_db

sqlite> .schema

```
CREATE TABLE ARTIKLI
(
        ARTIKAL_ID INTEGER PRIMARY KEY,
        KOD CHAR(10) UNIQUE,
        BARKOD CHAR(13) UNIQUE,
        JMJ CHAR(3) NOT NULL,
        CIJENA DECIMAL(10,2) NOT NULL,
        NAZIV VARCHAR(150),
        CONSTRAINT cijena_naziv UNIQUE (CIJENA, NAZIV)
);
CREATE TABLE OPERATERI
(
        OPERATER_ID INTEGER PRIMARY KEY,
        NAZIV VARCHAR(100)
);
CREATE TABLE PARAMETRI
(
        PARAMETAR_ID INTEGER PRIMARY KEY,
        SVRHA CHAR(1) NOT NULL UNIQUE,
        OPIS VARCHAR(200)
);
CREATE TABLE RACUNI
(
        RACUN_ID INTEGER PRIMARY KEY,
        TIP SMALLINT DEFAULT 1 NOT NULL,
        BROJ INTEGER NOT NULL,
        DATUM DATE NOT NULL,
        UKUPNO_S_PDV DECIMAL(10,2) NOT NULL,
        OPERATER_ID INTEGER NOT NULL,

        CONSTRAINT broj_datum UNIQUE (BROJ, DATUM),

        FOREIGN KEY (OPERATER_ID) REFERENCES OPERATERI (OPERATER_ID),
         FOREIGN KEY (RACUN_ID) REFERENCES REKLAMIRANI_RN (REKLAMIRANI_RN_ID)
);
CREATE TABLE REKLAMIRANI_RN
(
        REKLAMIRANI_RN_ID INTEGER NOT NULL,
        RACUN_ORIGINAL_ID INTEGER,
        PRIMARY KEY (REKLAMIRANI_RN_ID)
);
CREATE TABLE RN_FISKALNI
(
        RACUN_ID INTEGER NOT NULL,
        DATUM DATE NOT NULL,
        VRIJEME TIME NOT NULL,
        BR_FISKALNOG_ISJECKA INTEGER,
        UKUPNO_S_PDV DECIMAL(10,2),
        PRIMARY KEY (RACUN_ID, DATUM, VRIJEME),
        FOREIGN KEY (RACUN_ID) REFERENCES RACUNI (RACUN_ID)
);
CREATE TABLE RN_STAVKE
(
        RN_STAVKA_ID integer primary key,
        RACUN_ID INTEGER,
        ARTIKAL_ID INTEGER NOT NULL,
        KOLICINA DECIMAL(12,3),
        CIJENA DECIMAL(10,2),
```

```
        FOREIGN KEY (ARTIKAL_ID) REFERENCES ARTIKLI (ARTIKAL_ID),
        FOREIGN KEY (RACUN_ID) REFERENCES RACUNI (RACUN_ID)

);
CREATE TRIGGER artikli_5f after insert ON artikli
begin
update artikli set cijena=(select case (new.cijena * 100 % 5) when 0 then new.cijena else -9999 end) where
artikal_id=new.artikal_id;
delete from artikli where cijena<0;
end;
CREATE TRIGGER artikli_5fu before update ON artikli
begin
select case (NEW.cijena * 100 % 5) when 0 then NEW.cijena else RAISE(ROLLBACK, 'cijena na p5f')  end;
end;
CREATE TRIGGER rn_stavke_insert after insert on rn_stavke
begin
update racuni set ukupno_s_pdv = ukupno_s_pdv + new.kolicina * new.cijena where racuni.racun_id=new.racun_id;
end;
```

Prilog 3: Sqlite

Foreign key

http://www.sqlite.org/foreignkeys.html#fk_composite

ermaster generiše:

```
ALTER TABLE RACUNI
        ADD FOREIGN KEY (OPERATER_ID)
        REFERENCES OPERATERI (OPERATER_ID)
        ON UPDATE RESTRICT
        ON DELETE RESTRICT
;
```

treba napraviti ispravku, jer ovo ALTER TABLE za sqlite ne funkcioniše::

```
CREATE TABLE RACUNI
(
        RACUN_ID INTEGER NOT NULL,
        TIP SMALLINT DEFAULT 1 NOT NULL,
        BROJ INTEGER NOT NULL,
        DATUM DATE NOT NULL,
        UKUPNO_S_PDV DECIMAL(10,2) NOT NULL,
        OPERATER_ID INTEGER NOT NULL,
        PRIMARY KEY (ID),
        CONSTRAINT broj_datum UNIQUE (BROJ, DATUM)
        FOREIGN KEY(OPERATER_ID) REFERENCES OPERATERI(OPERATER_ID)
);
```

Autoincrement

Operater_id će postati autoinkrement polje[28] ako ovako stavimo:

```
sqlite> create table operateri(operater_id integer primary key, naziv varchar(100));
```

Uobičajene operacije nad SQLite bazom

```
bringout@nd-273:~/FIT/github/FIT_UBP/seminarski/ermaster$ sqlite3 ubp_pos.db < ubp_pos.sql

bringout@nd-273:~/FIT/github/FIT_UBP/seminarski/ermaster$ sqlite3 ubp_pos.db

sqlite> .schema
CREATE TABLE ARTIKLI
(
         ARTIKAL_ID INTEGER PRIMARY KEY,
...
         NAZIV VARCHAR(150),
         CONSTRAINT cijena_naziv UNIQUE (CIJENA, NAZIV)
);
...
```

Insert:

```
sqlite> insert into operateri(naziv) values('hernad') ;
sqlite> insert into operateri(naziv) values('vsasa');
sqlite> insert into operateri(naziv) values('bjasko');
sqlite> insert into operateri(naziv) values('vzeljka');
```

28 http://www.sqlite.org/autoinc.html

```
sqlite> select * from operateri;
1|hernad
2|vsasa
3|bjasko
4|vzeljka
```

UNIQUE različit od implementacije do implementacije SQL baze

Tokom čitanja[sqlite] mi je bilo interesantno saznanje da se različite implementacije SQL baza po pitanju UNIQUE ograničenja različito ponašaju.

Tako MS SQL ne dozvoljava da neko UNIQUE polje ima ovakav sadržaj:

```
1
NULL
NULL
2
3
4
NULL
```

S druge strane, sqlite i PostgreSQL to omogućavaju. Na osnovu tog saznanja sam sam na artikli.barkod postavio UNIQUE ograničenje, iako polje ***barkod*** *nije obavezno (nije mandatory, može imati NULL vrijednost).*