UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET SARAJEVO

ZADAĆA br. 4

Osnove baza podataka

Ime i prezime: Nur Osmanbegović
Br. Indexa: 17501
Odsjek: Računarstvo i informatika
Godina: 2018/19

Sadržaj

ZADATAK 1 – Normalizacija

1NF – Prva normalna forma

Tabele: TIP_FIZICKIH_LICA, FIZICKA_LICA, UPOSLENICI, GRAD, DRZAVE, KONTINENTI

1. *Relacija je u prvoj normalnoj formi ako je svaka kolona (atribut) prostog tipa.*
   U slucaju nasih tabela je to ispunjeno. Problem je mogao nastati oko boravista lica (koju sacinjavaju, kako je bilo specificirano u prvoj zadaci, adresa i grad). Medjutim, ovaj podatak je razbijen na dva atributa, adresu (kao ulica, broj) i grad, tako da ova relacija ispunjava uvjete da bi bila u prvoj normalnoj formi. Ova relacija ne bi bila u prvoj normalnoj formi da smo imali jedan atribut boraviste koji bi bio sacinjen od adrese i grada (odvojeni npr. zarezom, dvotackom...) , tj. Da u istom atributu cuvamo ove podatke jer bi time taj atribut bio slozenog tipa. Svaka kolona u nasim tabelama je prostog tipa te je ovaj uslov ispunjen za sve relacije.

2. *Još jedan zahtjev prve normalne forme je da je svaki red moguce jedinstveno identificirati*
   Ovaj zahtjev je ispunjen u svakoj tabeli obzirom da nam svaka tabela ima primary key koji je UNIQUE (sama definicija primary key-a podrazumijeva da je jedinstven u datoj tabeli) i na taj nacin je svaki njen red jedinstveno odredjen preko tog jedinstvenog kljuca (id-a).

3. *Posljednji zahtjev prve normalne forme je da ne postoje repetirajuce grupe podataka*
   Ovaj zahtjev smo ispunili izdvajanjem gradova, drzava i kontinenata u posebne tabele te povezivanjem tih tabela (preko primarnih i stranih kljuceva) i ukljucivanjem id-a grada samo kao informacije za boraviste lica (ovo smo koristili i u drugim tabelama, ali samo u tabeli FIZICKA_LICA od ovih 6 odabranih). Da nismo na ovaj nacin dizajnirali ove tabele, ovaj kriterij ne bi bio ispunjen, jer bismo imali npr. Vise redova u tabeli FIZICKA_LICA kojima su atributi grad drzava i kontinent identicni, ili gradovi razliciti ali drzave i kontinenti isti. Svaka drzava je vezana samo za jedan kontinent, svaki grad samo za jednu drzavu. Ako bismo svaki put pored grada navodili drzavu i kontinent kojem taj grad pripada u tabeli FIZICKA_LICA, imali bismo ogromnu kolicinu dupliciranih podataka. Takodjer ovaj uslov ne bi bio ispunjen da smo u tabeli UPOSLENICI imali sve informacije a ne samo dopunske (koje nemaju kupci) jer bismo u tom slucaju za uposlenike duplali informacije ime, prezime, datum_rodjenja, adresu i grad.

Napomena: Drugi i treci uslov ne spadaju u uslove prve normalne forme, vec u uslove UNF-a odnosno unnormalised form. A da bi relacija bila u prvoj normalnoj formi, prvo moraju biti ispunjeni kriteriji UNF-a. Obzirom da se nije trazila ta provjera, ove kriterije sam navela kao kriterije prve normalne forme da utvrdim da li su i oni ispunjeni.

Zakljucak: Date relacije jesu u prvoj normalnoj formi jer ispunjavaju sve gore navedene uslove (zahtjeve UNF-a i zahtjev na atribute relacija prve normalne forme). Ovaj zahtjev ujedno vrijedi i za sve relacije baze po slicnoj analogiji.

2NF – Druga normalna forma

Relacija je u drugoj normalnoj formi ako je u prvoj normalnoj formi i ako su svi njeni nekljucni atributi (oni koji nisu kandidati za kljuc, niti dio kandidata za kljuc) funkcionalno zavisni od primarnog kljuca (citavog primarnog kljuca a ne samo dijela, u slucaju da je sastavljen od vise atributa). Smjestanje podataka u drugu normalnu formu sastoji se od premjestanja u druge tabele onih podataka koji su zavisni samo od dijela kljuca.

Osnovni zahtjevi druge normalne forme su:

1. Uklanjanje podskupova koji se nalaze u vise redova i njihovo smjestanje u posebne tabele
2. Kreiranje veza izmedju novih tabela i tabela sa kojima su spojene koristenjem stranih kljuceva

Svaka relacija kod koje je primarni kljuc sastavljen od samo jednog atributa je u drugoj normalnoj formi.

Napomena: Ovdje se naravno misli na primarni kljuc u logickom smislu.

Tabele: TIP_FIZICKIH_LICA, FIZICKA_LICA, UPOSLENICI, GRAD, DRZAVE, KONTINENTI, SKLADISTA, PROIZVODI, KOLICINE_U_SKLADISTIMA

1. *Prvi zahtjev je da relacija bude u prvoj normalnoj formi*
   Utvrdili smo da je ovaj zahtjev ispunjen za sve relacije
2. *Drugi zahtjev je da svi nekljucni atributi funkcionalno zavise od primarnog kljuca*
   Ovaj zahtjev jeste ispunjen za sve relacije. Krenimo od tabele KONTINENTI – naziv u potpunosti zavisi od id-a kontinenta koji je primarni kljuc. Zatim, tabela DRZAVE, naziv i id_kontinenta su u potpunosti odredjeni na osnovu id-a drzave. Ovo takodjer vrijedi za relaciju GRADOVI. Imali bismo krsenje druge normalne forme imali da smo u tabeli KOLICINE_U_SKLADISTIMA drzali sve informacije o proizvodu i skladistu tj id_skladista, naziv_skladista, id_proizvoda, naziv_proizvoda, kolicina: Ovdje bi primarni kljuc bio id_skladista i id_proizvoda (kompozitni primarni kljuc), medjutim, kolicina je jedini atribut koji bi ovisio od kompletnog primarnog kljuca (kolicina zaista ovisi i od proizvoda i od skladista o kojem se radi), dok naziv skladista ovisi samo o id-u skladista, a naziv proizvoda samo o id-u proizvoda. Te da bi ova relacija bila u drugoj normalnoj formi trebali bismo izdvojiti tabelu skladista, tabelu proizvodi i tabelu veze kolicina (sto je u mom slucaju uradjeno u ERD-u, samo sa vise atributa). Relacije TIP_FIZICKIH_LICA, FIZICKA_LICA, UPOSLENICI takodjer ispunjavaju zahtjeve druge normalne forme jer svi atributi ovisi od primarnog kljuca u potpunosti.

Zakljucak: Date relacije jesu u drugoj normalnoj formi jer ispunjavaju sve gore navedene uslove (zahtjeve prve normalne forme i zahtjev na funkcionalnu zavisnost atributa druge normalne forme). Ovaj zahtjev ujedno vrijedi i za sve relacije baze po slicnoj analogiji.

3NF – Treca normalna forma

Relacija je u drugoj normalnoj formi ako je u drugoj normalnoj formi i ako ne postoji tranzitivna funkcionalna zavisnost. Ovaj zahtjev se zapravo odnosi na to da ako imamo atribute A,B i C, te da je atribut B potpuno funkcionalno zavisan od A, a C od B, onda je C tranzitivno zavisno od A preko B. Tacnije, da nijedan atribut koji ne pripada kljucu ne zavisi od nekljucnog atributa.

Osnovni zahtjevi trece normalne forme su:

1. Da su relacije u drugoj normalnoj formi
2. Da ne postoje tranzitivne zavisnosti

Tabele: TIP_FIZICKIH_LICA, FIZICKA_LICA, UPOSLENICI, GRAD, DRZAVE, KONTINENTI

1. *Prvi zahtjev je da relacija bude u drugoj normalnoj formi*
   Utvrdili smo da je ovaj zahtjev ispunjen za sve relacije
2. *Drugi zahtjev je da ne postoje tranzitivne zavisnosti*
   Ovaj zahtjev jeste ispunjen za sve relacije. Relacija KONTINENTI sigurno zadovoljava trecu normalnu formu obzirom da ima samo dva atributa pri cemu je nekljucni atribut potpuno funksionalno zavisan od primarnog kljuca. Zatim prelazimo na tabelu DRZAVE, da smo pored id_kontinenta cuvali naziv kontinenta, imali bismo tranzitivnu zavisnost naziva kontinenta od id-a grada. Izdvajanjem tabele kontinenti smo se rijesili te tranzitivne zavisnosti i ispunili zahtjev trece normalne forme. Sto se tice relacije GRADOVI, vrijedi ista analogija kao i za relaciju DRZAVE, samo sa malo vise atributa. Npr. Da smo pored id-a grada, naziva grada i id-a drzave imali i naziv drzave (ili jos losiji slucaj, i id i naziv kontinenta) imali bismo tranzitivnu zavisnost i to duplu. Prvo naziv drzave od id-a grada preko id-a drzave, a zatim id-kontinenta i naziva kontinenta preko id-drzave. Ovdje spajanje tabela UPOSLENICI i FIZICKA_LICA ne bi igrala neku ulogu, samo sam odlucila odvojiti ove tabele (pri cemu se u tabeli UPOSLENICI nalaze samo dodatne informacije, koje ne postoje za slucaj kupada) kako bih izbjegla veliki broj NULL polja u tabeli. Ali, po pitanju zahtjeva trece normalne forme, igralo bi ulogu da smo tabelu TIP_FIZICKIH_LICA spojili sa tabelom FIZICKA_LICA (ili UPOSLENICI, iako malo besmisleno). Naime, naziv tipa bi tranzitivno preko id-a tipa ovisio od id-lica. Izdvajanjem relacija na ovaj nacin tj njihovim projekcijama smo osigurali da ne postoji tranzitivna zavisnost medju atributima relacije.

Zakljucak: Date relacije jesu u trecoj normalnoj formi jer ispunjavaju sve gore navedene uslove (zahtjeve druge normalne forme i zahtjev na tranzitivnu zavisnost atributa trece normalne forme). Ovaj zahtjev ujedno vrijedi i za sve relacije baze po slicnoj analogiji.

BCNF – Boyce-Coddova normalna forma

BCNF kaze da kad god u relaciji postoji netrivijalna funkcionalna zavisnot X->Y izmedju disjunktivnih, nepraznih skupova njenih atributa, onda domen te zavisnosti sadrzi kljuc relacije. U postupku normalizacije takav entitet se reprezentuje posebnom relacijom. U sustini, ovaj zahtjev se svodi na to da ako je atribut A funkcionalno zavisan od atributa B koji nije dio kljuca, atribut A ne smije biti dio kljuca.

Osnovni zahtjevi Boyce-Codd normalne forme su:

3. Da su relacije u trecoj normalnoj formi
4. Da atributi dio kljuca nisu u funkcionalnoj zavisnosti od atributa koji nisu dio kljuca

Tabele: TIP_FIZICKIH_LICA, FIZICKA_LICA, UPOSLENICI, GRAD, DRZAVE, KONTINENTI

3. *Prvi zahtjev je da relacija bude u trecoj normalnoj formi*
   Utvrdili smo da je ovaj zahtjev ispunjen za sve relacije
4. *Drugi zahtjev je da ne postoje funkcionalne zavisnosti dijelova kljuca od atributa koji ne pripadaju kljucu*
   Ovaj zahtjev jeste ispunjen za sve relacije. Relacija KONTINENTI sigurno zadovoljava Boyce-Coddovu normalnu formu obzirom da ima samo dva atributa pri cemu je nekljucni atribut potpuno funksionalno zavisan od primarnog kljuca. Zatim prelazimo na tabelu DRZAVE, ovdje nam je primarni kljuc atribut id date relacije, i taj entitet nije u funkcionalnoj zavisnosti od bilo kojeg drugog atributa, vec su svi drugi atributi u funkcionalnoj zavisnosti od naseg primarnog kljuca - id. Ista analogija vrijedi i za ostale tabele. I u tabelama UPOSLENICI, FIZICKA_LICA i TIP_FIZICKIH_LICA, primarni kljuc je prost odnosno sastavljen od samo jednog atributa pri cemu ostali atributi funkcionalno ovise od primarnog kljuca.

Zakljucak: Date relacije jesu u Boyce-Coddovoj normalnoj formi jer ispunjavaju sve gore navedene uslove (zahtjeve trece normalne forme i zahtjev na funkcionalnu zavisnost primarnog kljuca). Ovaj zahtjev ujedno vrijedi i za sve relacije baze po slicnoj analogiji.

4NF – Cetvrta normalna forma

Relacija je u cetvrtoj normalnoj formi ako je u BC normalnoj formi i ako ne postoji multi-vrijednosna zavisnost. Tacnije, atributi jednog ili vise redova u tabeli ne bi trebali rezultirati vise od jednog reda u samoj tabeli. Multi-vrijednosna zavisnost nastaje kada jedan kljuc determinise vise vrijednosti druga dva ili vise atributa a da ti atributi medjusobno ne zavise.

Osnovni zahtjevi cetvrte normalne forme su:

1. Da su relacije u Boyce Codd normalnoj formi
2. Da ne postoji vise od jedne multi-vrijednosne zavisnosti

Tabele: ISPORUKE, PRAVNA_LICA, UGOVORI, GRAD, DRZAVE, KONTINENTI

1. *Prvi zahtjev je da relacija bude u bc normalnoj formi*
   Utvrdili smo da je ovaj zahtjev ispunjen za sve relacije
2. *Drugi zahtjev je da ne postoji multivrijednosna zavisnost*
   Ovaj zahtjev jeste ispunjen za sve relacije. Relacija KONTINENTI sigurno zadovoljava cetvrtu normalnu formu obzirom da ima samo dva atributa a za multivrijednosnu zavisnost su potrebne barem dvije kolone. Tabela DRZAVE, za jedan id je moguce imati samo jednu vrijednost u tabeli te je takodjer ispunjen uslov. Analogno vazi za relacije GRAD i TIPOVI_PRAVNIH_LICA. Do krsenja zahtjeva cetvrte normalne forme je moglo doci u slucaju da smo spojili tabele PRAVNA_LICA, ISPORUKE i UGOVORI jer jedno pravno lice moze imati i vise ugovora i vise isporuka, a isporuka i ugovor ni na koji nacin nisu vezani.

Zakljucak: Date relacije jesu u cetvrtoj normalnoj formi jer ispunjavaju sve gore navedene uslove (zahtjeve BC normalne forme i zahtjev na multivrijednosnu zavisnost).

ZADATAK 2 – Pogledi

Ovaj zadatak nije bio zahtjevan u smislu da su upiti za kreiranje pogleda vec napisani u prethodnoj zadaci. Ono sto me najvise namucilo u ovom zadatku jeste davanje smislenih naziva kolonama u pogledu, prvenstveno zbog ogranicenja na 30 karaktera a zatim jer su specificni upiti bili pa je malo zahtjevno bilo naci naziv koji bi adekvatno imenovao kolone te bar na neki nacin nagovijestio koji se podaci nalaze u datim kolonama. Takodjer, obzirom da je fokus posljednje zadace bio na podupitima, upiti su bili glomazni, i prilicno specificni, te je bilo tesko probrat nad kojim upitima bi bilo koliko toliko smisleno kreirati poglede. Za navedene upite koje sam odabrala, smatram da je najsmislenije kreirati poglede (od onih uradjenih u prethodnoj zadaci) i da bi bilo korisno za funkcionisanje firme imati te podatke izdvojene na ovaj nacin.

1. *Prvi upit nad kojim sam kreirala pogled je iz kategorije 2, treci upit:*
   Izlistati 10 skladista koja imaju najvise proizvoda sa popustom na stanju u protekla 3 mjeseca.
   Za kolonu naziv skladista sam se odlucila za naziv najpristupacnija skladista, iz razloga sto su skladista u kojima je najvise proizvoda na popustu pristupacnija u smislu da su cijene bolje (u odnosu na standardne za te proizvode). Za kolonu kolicina, nisam se mnogo dvoumila, smisleno mi je bilo da se zove Ukupno snizenih proizvoda jer to i predstavlja.
2. *Drugi upit nad kojim sam kreirala pogled je iz kategorije 3, peti upit:*
   Pronaci uposlenike koji su u TechFreak-u potrosili vise novca od prosjecnog iznosa koji su potrosili ostali kupci.
   Za kolonu id_lica sam se odlucila za naziv Uposlenici-nadprosjecni kupci iz razloga sto to jesu uposlenici a obzirom da su potrosili vise novca od prosjeka svih ostalih kupaca, oni spadaju u nadprosjecne kupce, vise potrose od ostalih – bolji su kupci. A za kolonu gdje je suma potrosenog novca, naziv kolone Ukupna potrosnja se sam nametnuo.
3. *Treci upit nad kojem sam kreirala pogled je iz kategorije 4, prvi upit:*
   Napisati upit koji ce pronaci one isporuke koje ukljucuju fakturu kod koje barem jedna stavka fakture ima i popust i garanciju.
   Naziv Isporuke – garancija i popust je jedini naziv koji je mogao stati (ispuniti zahtjev od max 30 karaktera) a koji je pri tom koliko toliko opisivao sta predstavlja ta kolona.

ZADATAK 3 – Trigeri

Sa ovim zadatkom sam imala najvise problema. Prvi razlog je sto se ne moze update-ovati tabela nad kojom je triger pozvan. Veliki broj smislenih trigera nisam mogla kreirati iz ovog razloga. Zatim pise da je moguce pozvat triger unutar trigera, medjutim iz nekog razloga mi to nije dozvoljavao.

Napravila sam triger koji je BEFORE, radi prije INSERTA ili UPDATEA tabele, u slucaju dodavanja ili mijenjaja cijene provjerava da li je cijena pozitivan broj. U slucaju da nije, prekida operaciju izvrsavanja – baca izuzetak s porukom. Ovaj triger se nalazi u fajlu 5 TRIGGERI posto je jedini koji se moze izvrsiti (dodavanje ostalih u istu skriptu bi ucinilo ovu neizvrsivom zbog errora).

U drugom fajlu 5 TRIGGERI 2 se nalaze ostali trigeri, koji nisu potpuno ispravni ali sam ih odlucila ipak priloziti zbog, po meni smislenih ideja.

Triger GarancijaDo bi za svaki red koji se unosi u garancije, racunao atribut datum_do odnosno datum do kojeg vrijedi garancija na osnovu datuma unosa garancije, i broja mjeseci u tabeli proizvod (preko id stavke se pristupa tabeli proizvodi) racunao datum do kada vazi garancija. Ovo bi bio after insert triger.

Druga dva trigera su ujedno i medjusobno povezana, UpdateCijenaStavke je triger koji bi se pozivao prije azuriranja kolicina u stavci fakture, i taj triger bi racunao novu cijenu te stavke dok bi se zatim izvrsavao after update triger za cijenu fakture koja se takodjer treba u tom slucaju azurirati.

ZADATAK 4 – Funkcije

Napisati funkciju koja ce za zadatu vrijednost u procentima (+/ – 0 – 99%) izracunati novu cijenu za sve proizvode koji pripadaju odredenoj kategoriji i azurirati vrijednost u tabeli Proizvod, koloni cijena. Funkcija prima dva parametra, kategorija id i promjena procenat.

Funkcija je prilicno lagana za napisati, obzirom da nije specificirano sta funkcija treba vratiti, odlucila sam vratiti broj proizvoda na koje je funkcija uticala (tj. Ciju je cijenu promijenila). Samo sam prebrojala koliko proizvoda u tabeli proizvodi ima id_kategorije koji je isti kao onaj zadan kao parametar te dodijelila taj broj lokalnoj varijabli funkcije - total, i nju vratila kao rezultat.

ZADATAK 5 – Procedure

1. Napisati proceduru koja ce obrisati sve podatke u svim tabelama za fakture starije od godine zadane u prametru koji procedura prima.
   Ovu proceduru sam nazvala brisi. U sustini, koristila sam jedan eksplicitni kursor na virtuelnu tabelu faktura (u kojoj se nalaze samo one fakture kod kojih je datum stariji od onog zadan parametrom). Onda sam kursorom prolazila kroz sve redove tako napravljene tabele, i koristila for loop zbog implicitnog kursora koji se u tom slucaju kreira i koji prolazi kroz sve stavke fakture kod kojih je id_fakture jednak id-u fakture na koju je kursor trenutno pozicioniran (trenutno pokazuje). A za svaku takvu stavku izbrisala iz garancija one garancije kod kojih je id_stavke jednak id-u stavke na koju implicitni kursor petlje pokazuje. Nakon brisanja garancija za pronadjenu stavku, sam brisala stavku, a nakon sto implicitnim kursorom prodjem kroz sve stavke vezane za fakturu na koju pokazuje eksplicitni kursor, brisala sam i tu fakturu.
   Zbog ogranicenja na strane kljuceve se moralo brisati ovim redom: Nadjem fakturu, nadjem njene stavke, nadjem garancije stavke, izbrisem garancije, izbrisem stavke, izbrisem fakturu.

2. Napisati proceduru koja ce izlistati sve fakture i njihove stavke za kupca koji je naveden u parametru procedure.
   Ovdje sam koristila samo implicitne kursor for loop-a. Prolazim kroz sve fakture kod kojih je id_kupca jednak id-u kupca koji je proslijedjen kao parametar. Zatim prolazim kroz stavke i nadjem koje su stavke vezane za fakturu na koju implicitni kursor u tom trenutku pokazuje. Tako da se prvo ispisuje faktura a zatim se ispisuju stavke te fakture.

BONUS ZADATAK

Dva sql fajla su vezana za ovaj zadatak, u jednom se samo nalazi kod za kreiranje tabele LOG u koju ce se unositi trazeni redovi. U drugom fajlu se nalaze triger i procedura. Procedura je univerzalna i vazi za sve tabele. Medjutim, nije moguce kreirati triger koji je vezan za vise tabela, tj. Za svaku tabelu je potrebno kreirati triger koji bi pozivao proceduru.

U sustini, broj trigera bi trebao biti jednak broju tabela, pri cemu bi se jedino mijenjala dva dijela: ON naziv_tabele i pri pozivu procedure, string koji se salje kao prvi argument proceduri koji bi trebao predstavljati naziv tabele. Ti dijelovi su u nastavku naznaceni crvenom bojom:

```
CREATE OR REPLACE TRIGGER LOG_TRIGGER
  AFTER INSERT OR UPDATE OR DELETE
  ON naziv_tabele
DECLARE
  log_akcija  varchar2(50);
BEGIN
  IF INSERTING THEN
    log_akcija := 'Insert';
  ELSIF UPDATING THEN
    log_akcija := 'Update';
  ELSIF DELETING THEN
    log_akcija := 'Delete';
  ELSE
    DBMS_OUTPUT.PUT_LINE('This code is not reachable.');
  END IF;

  insert_into_log('naziv_tabele', log_akcija);

END;
```

Ja sam kreirala triger nad tabelom PROIZVODI, koji ce pri insertu, updateu ili deleteu nad tom tabelom pozvati proceduru koja ce izvrsiti upis u tabelu LOG, analogno bi se trebalo uraditi i za ostale tabele.