SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ivan Milas

NORMALIZACIJA SHEME RELACIJSKE BAZE PODATAKA

ZAVRŠNI RAD

Varaždin, 2018.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ivan Milas

Matični broj: 42257

Studij: Informacijski sustavi

NORMALIZACIJE SHEME RELACIJSKE BAZE PODATAKA

ZAVRŠNI

Mentor:

Prof. dr. Sc. Mirko Maleković

Varaždin, veljača 2018.

Ivan Milas

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

U ovom završnom radu bavimo se tematikom relacijskih baza podataka i njihovom normalizacijom. Ponajprije ćemo obraditi temu relacijskih baza podataka i njihovih osobina. Zatim ćemo spomenuti motivaciju za normaliziranje baza podataka i navesti slučajeve u kojima je poželjno koristiti normalizaciju ali i one slučajeve u kojima to nije preporučljivo. Svaki slučaj ćemo poduprijeti sa primjerom i objašnjenjem toga primjera. Spomenuti ćemo koje vrste normalnih formi postoje te ćemo o svakoj normalnoj formi napisati ono najvažnije. Ponajviše ćemo se zadržati na trećoj normalnoj formi za koju ćemo prikazati primjer sinteze na temelju odabrane relacijske sheme(R,F). Na primjeru sinteze ćemo objasniti što je to ključ relacije, što je to zatvarač skupa te kanonski prekrivač. Za kraj ćemo odabranu relacijsku shemu implementirati u SQL-u te prikazati rezultate implementacije u SQL-u.

Ključne riječi: normalizacija, normalne forme, sinteza , relacijske baze, relacije, zatvarač skupa, kanonski pokriv

Sadržaj

1. Uvod

Ovaj završni rad bavi se tematikom normalizacije relacijske baze podataka. Posebna pozornost je posvećena algoritmu sinteze treće normalne forme za smanjenje redundancije, očuvanje informacija i zavisnosti unutar relacijske baze podataka.

-U drugom poglavlju obratiti ćemo pozornost na relacijski model baza podataka na kojemu se temelje normalne forme koje ćemo spomenuti u četvrtom poglavlju ovoga rada. Obraditi ćemo pojmove relacija i ključeva relacija te relacijsku shemu i način njenoga zapisivanja.

Treće poglavlje sam posvetio isključivo da objasnim razloge i motivacije za normalizacijom baze podataka. Spomenuti ćemo poteškoće u radu sa nenormaliziranim podatcima te onim slučajevima u kojima nema potrebe za normalizacijom.

Normalne forme će biti obrađene u četvrtom poglavlju. Obraditi ćemo prve tri normalne forme te Boyce-Coddovu i četvrtu normalnu formu. O svakoj formi ćemo napisati kratki opis da shvatimo čemu nam služe normalne forme i kolika je njihova važnost. Ponajviše pozornosti ćemo posvetiti trećoj normalnoj formi iz razloga što ćemo u ovom radu obraditi sintezu te normalne forme. 3NF je praktično važna normalna forma jer se algoritmom sinteze postiže smanjenje redundancije uz čuvanje informacije i zavisnosti.

U petom poglavlju ćemo prikazati kako funkcionira algoritam sinteze 3. normalne forme za izabranu relacijsku shemu(R,F).

Zadnje poglavlje posvećeno je implementaciji dobivenih relacija iz petog poglavlja u SQL.

2. Relacijski model podataka

Relacijski model podataka predstavlja nam podatke preko dvodimenzionalnih tablica koje nazivamo relacije. Ovaj pristup zasnovan je krajem 60. godina prošloga stoljeća a njegov začetnik je Edgar Codd.

„Model se dugo vremena pojavljivao samo u akademskim raspravama i knjigama. Prve realizacije na računalu su bile suviše spore i neefikasne. Zahvaljujući intenzivnom istraživanju te napretku samih računala, efikasnost relacijskih baza postupno se poboljšavala. Sredinom 80-ih godina 20. stoljeća relacijski model postao je prevladavajući. I danas se većina DBMS-ova koristi tim modelom.“(Manger, 2012)

2.1. Relacija

Svaki redak u tablici relacijskog modela predstavlja predstavlja određeni entitet ili predstavlja vezu između dva ili više primjeraka. Svaka relacija ima svoje ime koje je razlikuje od drugih relacija unutar baze.

Relacija ili tablica ne smije sadržavati dva ista retka. Stupac u tablici predstavlja određene atribute entiteta koji se nalaze u tablici. Različiti entitet mogu imati iste atribute te oni onda predstavljaju identične podatke, no moramo paziti da dva različita entiteta nemaju iste vrijednosti na svim atributima relacijske sheme.

Vrijednosti atributa moraju biti jednostavne i jednoznačne što bi značilo da predstavljaju točno određeni podatak i da se ne mogu rastaviti na manje dijelove. Ovisno o konfiguraciji relacije neki atributi ne moraju biti uneseni, to jest njihova vrijednost može biti nepoznata ili sadržavati NULL vrijednost. U tablici 2.1 možemo primijetiti da su nam atributi tablice *naslov, godina, trajanje* i ***žanr***.

Tablica 1: Tablica filmovi

naslov	*godina*	*trajanje*	*žanr*
Ratovi	1977	124	sciFi
Avatar	2003	134	sciFi
Umri	1998	115	Akcija

Vidimo da u stupcu sa atributom *godina* imamo zapisane godine kada su filmovi nastali. Također, u stupcu sa atributom žanr vidimo da imamo dvije iste vrijednosti što nam govori da unutar istoga stupca možemo imati jednake vrijednosti.

K---od tabličnog prikazivanja relacije nikad se ne ponavljaju redovi, jer je relacija skup slogova. Poredak redova i stupaca nije bitan.(Maleković, 2016)

2.2. Ključ relacije

Kao što smo to i ranije spomenuli unutar relacije ne smiju postojati dvije jednake n-torke. Da bismo to ostvarili trebamo jednoznačno definirati svaki redak u tablici. Stoga, trebamo definirati ključ za n-torke. Relacija može imati više kandidata za ključ nakon čega se jedan od njih proglašava primarnim ključem. Svi ključevi moraju biti jednoznačni i njihove vrijednosti ne smiju ostati neupisane. Shema relacije za tablicu 2.1 može se napisati kao: (naslov,godina,trajanje,žanr) gdje su podcrtane vrijednosti ključevi atributa. U ovom slučaj naslov i godinu koristimo kao ključ relacije no ovo nije dobra praksa zbog toga što je moguće da se dogodi da u istoj godinu izađu dva filma sa istim naziva. U tom slučaju, ne možemo jednoznačno definirati entitet u tablici filmovi. Dok postoji mogućnost da atribute naslov i godinu koristimo kao ključ relacije u mnogim bazama podataka koriste se ID vrijednosti koje osiguravaju da se vrijednosti ključeva neće ponavljati.

Definicija 2.1. Ključ relacije:

Ključ K relacije r je podskup skupa atributa R od r sa sljedećim svojstvima:

(Manger, 2012, str. 38):

1. Vrijednosti atributa iz K jednoznačno određuju n-torku u r. Dakle, ne mogu u r postojati dvije n-torke s istim vrijednostima atributa K.

2. Ako izbacimo iz K bilo koji atribut, tada se narušava svojstvo 1.

Definiranjem ključa zapravo definiramo ograničenje nad tablicom, tj. uvjet integriteta koje relacija mora zadovoljavati u svakom trenutku vremena. Relacija može imati više ključeva a jedan od njih se označava kao primarni ključ(kraće PK). Strogo je zabranjeno da primarni ključ poprimi NULL vrijednost jer se narušava entitetski integritet.

2.2.1. Primarni ključ

Sve n-torke unutar relacije moraju biti različite stoga ključ relacije K uvijek postoji. Što zadovoljava prvo svojstvo iz definicije 1.1. Izbacivanjem svih suvišnih atributa dolazimo do

podskupa koji zadovoljava i svojstvo pod rednim brojem 2. Može se dogodit da relacija ima više ključeva. Stoga jednog od njih proglašavamo *primarnim ključem*. Vrijednost primarnog ključa u nijednoj relaciji ne smije ostati neupisana. (Manger, 2012)

Atribut je ne ključni atribut ako ne participira niti u jednom ključu.

2.3. Logička posljedica

Definicija 2.2. Neka je $F \subseteq FVS(R)$ i $f \in FVS(R)$. f je logička posljedica od F ako za svaku relaciju r nad R vrijedi: ako je r model za F, onda je r model i za f ili drugim riječima, ako F vrijedi u r, onda f vrijedi u r.

Oznaka za logičku posljedicu je $\models$. Stoga pišemo $F \models f$, a čitamo f je logička posljedica od F.

3. Normalne forme

Logičko oblikovanje sheme relacijske baze podataka temelji se na postupku normalizacije relacijske sheme (R,F) koji rezultira shemom relacijske baze podataka u odgovarajućoj normalnoj formi. U postupku normalizacije primjenjuju se određena svojstva zavisnosti i svojstva dekompozicije relacijske sheme.(Maleković & Schatten, 2017)

U daljnjem dijelu ovoga rada ćemo opisati jednostavnije dijelove normalizacije a to bi bilo prevođenje u prve tri normalne forme. Zatim će na red doći Boyce-Coddova normalna forma te normalne forme višeg reda koje ćemo samo spomenuti.

Da bismo razumjeli čemu nam služe normalne forme prvo moramo reći nešto o njima općenito a to je da su normalne forme ograničenja koja neka relacijska shema mora zadovoljavati. Forme koje ćemo spomenuti u ovom radu su: prva normalna forma(1NF),druga normalna forma(2NF),treća normalna forma(3NF),Boyce-Coddova normalna forma(BCNF),četvrta normalna forma(4NF),peta normalna forma(5NF),šesta normalna forma(6NF). Što je normalna forma viša to postoji više ograničenja za relacijsku shemu(R,F). To bi značilo da 1NF ima najmanje zahtjeve za (R,F) dok 6NF ima najveće.

3.1. Zavisnosti

Većina normalnih formi temelji se na pojmu funkcionalne ovisnosti između atributa ili skupina atributa. One predstavljaju važan dio integritetne komponente relacijskog modela jer relacije moraju zadovoljavati propisane uvjete integriteta. Zbog toga ćemo prije nego počnemo govoriti o normalnim formama nešto reći o funkcionalnim ovisnostima.

3.1.1. Funkcijska zavisnost

Definicija 3.1

Neka je R relacijska shema, X,Y ⊆ R. Izraz X → Y je funkcijska zavisnost nad R. FZ(R) = {X→Y | X,Y ⊆ R} je skup svih funkcijskih zavisnosti nad R.

Kažemo da X → Y ∈ FZ(R) vrijedi u relaciji r(R) ako

$\forall t_1, t_2 \in r\ (t_1[X] = t_2[X] => t_1[X] = t_2[X])$

Kažemo da funkcijska zavisnost X→ Y ∈ FZ(R) vrijedi u relaciji r(R) ako je za bilo koja dva sloga t_1 i t_2 iz r ispunjeno da iz jednakosti slogova t_1 i t_2 na skupu atributa X slijedi njihova jednakost i na skupu atributa Y.

3.1.2. Netrivijalna zavisnost

Definicija 3.2

Neka je zadana relacijska shema (R,F) . Za funkcijsku zavisnost $X \rightarrow Y \in FVZ(R)$ kažemo da je trivijalna ako je $Y \subseteq X$. U protivnom ako ne vrijedi $Y \subseteq X$, onda kažemo da je $X \rightarrow Y$ netrivijalna zavisnost.

3.1.3. Parcijalna zavisnost

Definicija 3.3

Neka je zadana shema (R,F), $F \in FVS(R)$. Funkcijska zavisnost $X \rightarrow Y \in FVS(R)$ je parcijalna s obzirom na F *ako* $\exists Z \subset X : F \models Z \rightarrow Y$

3.1.4. Tranzitivna zavisnost

Definicija 3.4

Neka je zadana shema (R,F), $F \in FVS(R)$. Funkcijska zavisnost $X \rightarrow Y \in FVS(R)$ je tranzitivna s obzirom na F*ako* $\exists Z \subseteq R : F \models X \rightarrow Z, F \models Z \rightarrow Y$, gdje je $Z \rightarrow Y$ netrivijalna zavisnost , $F \not\models Z \rightarrow X$.

3.2. Dekompozicija

Definicija 2.2.

$d(R) = R_1, \ldots, R_k$ je dekompozicija relacijske sheme (R,F) ako su ispunjeni sljedeći uvjeti:

a) $R_i \neq \emptyset$ za svako i = 1,2, …, k
b) $R_1 \cup \ldots \cup R_k = R$

Za kvalitetan logički dizajn sheme relacijske baze podataka poželjno je da kompozicija ima svojstvo čuvanja informacija i zavisnosti.(Maleković & Schatten, 2017)

3.2.1. Čuvanje informacija i zavisnosti

Definicija 2.3.

Za dekompoziciju $d(R_1, \ldots, R_k)$ relacijske sheme (R,F) kažemo da čuva informaciju ako vrijedi $F \models \bowtie(R_1, \ldots, R_k)$.

Ova definicija nam govori da dekompozicija d čuva informaciju ako za svaku relaciju r nad R vrijedi: ako je r model za F, onda je r model i za zavisnost spoja $\bowtie (R_1, \ldots, R_k)$, r se dobije

prirodnim spojem svojih projekcija na komponente dekompozicije. (Maleković & Schatten, 2017)

Definicija 2.4.

Za dekompoziciju $d(R_{1, ...,}R_k)$ relacijske sheme (R,F) kažemo da čuva zavisnosti F ako za svaku relaciju r nad R vrijedi: ako je $\sqcap_{R1}(r)$ model za $\sqcap_{R1}(F)$ i …$\sqcap_{Rk}(r)$ model za $\sqcap_{Rk}(F)$, onda je r model za F.

U ovoj definiciji za zavisnosti imamo skupove $\sqcap_{R1}(F)$, i =1, …, k koji se nazivaju projekcije skupa zavisnosti F na komponente dekompozicije R_1, …, R_k. Dakle, dekompozicija d® čuva zavisnosti ako je za svaku relaciju r(R) ispunjeno : iz činjenice da $\sqcap_{Ri}(F)$ vrijedi u $\sqcap_{Ri}(r)$ za svako i= 1, …, k slijedi da F vrijedi u r.

3.3. Prva normalna forma

Kada smo govorili o relacijama u prethodnom dijelu ovoga rada spomenuli smo da podatci unutar tablica trebaju biti atomarni ili jednostavni. To znači da u jednu kućicu unutar tablice ne možemo upisati više od jednoga podatka te da taj podatak mora biti jednostavan, tj. nedjeljiv. To svojstvo se naziva svojstvo prve normalne forme. Ako su vrijednosti atributa relacije jednostruke i nedjeljive kažemo da je relacijska shema u prvoj normalnoj formi. Ukoliko razmislimo, vidimo da je ovo svojstvo ugrađeno u sam model i definiciju relacije te da ne predstavlja nikakve posebne zahtjeve nad relacijom.

U relacijskoj bazi podataka i ne može postojati relacijska shema koja ne bi već od početka bila u 1NF. Pojam 1NF zapravo je izmišljen zbog drugih modela podataka gdje podaci ne moraju biti normalizirani čak ni u tom najjednostavnijem obliku.(Manger, 2012, str. 50)

Uzmimo za primjer tablicu Osoba sa atributima OIB, Ime, Prezime, Potomci.

- Osoba(<u>OIB</u>, Ime, Prezime, Potomci)
    - OIB →Ime,Prezime,Potomci

Atributi Ime i Prezime mogu sadržavati samo jednu vrijednost u redu no atribut Potomci je atribut koji zahtjeva da spremimo više različitih podataka za taj atribut. Razlog je taj što osoba može imati veći broj potomaka stoga dolazimo do problematike kako to zapisati u relaciju. Ovaj problem rješavamo na način da razdvojimo tablicu Osoba na dvije tablice. Tablicu Osoba koja će imati atribute OIB, Ime, Prezime te tablicu Potomci koja će imati atribute Šifra_pretka te Šifra_djeteta, gdje su oba atributa vanjski ključ te zajedno čine dvokomponentni primarni ključ.

- Osoba(OIB,Ime,Prezime)
    - OIB →Ime,Prezime
- Potomci(Šifra_pretka,Šifra_djeteta)

Time smo osigurali da svaki entitet Osoba u bazi podataka ima svoga pretka te svoga potomka.

3.4. Druga normalna forma

Druga normalna forma zahtjeva da je zadovoljena 1NF te da sva polja moraju biti jednoznačna i u potpunosti ovisiti o primarnom ključu. Unutar tablice se zapisuju podatci o samo jednom objektu. Unutar tablice ne smiju postojati podatci koji ne ovise o primarnom ključu. Relacijska shema (R, F) je u 2NF ako ne postoji parcijalna zavisnost neključnog atributa od ključa. U slučaju da relacija ima jednostavni ključ ona je automatsku u 2NF, uz uvjet da je u prvoj, a možemo reći da definicija 2NF ima smisla samo ukoliko je ključ relacije složen.(Garcia Molina, D.Ullman, & Widom, 2009)

Kao primjer za 2NF uzeti ćemo tablicu koja će sadržavati podatke o vozilu. Uzmemo li za primjer relacijsku shemu Vozilo_putnik s primarnim ključem Putnik,Vozilo i ne ključnim atributima Ime, Prezime i Naziv vozila, te zavisnosti Putnik,Vozilo→Ime,Prezime,Naziv_vozila, Putnik→Ime,Prezime i Vozilo→Naziv vozila, vidimo postojanje parcijalnih zavisnosti ne ključnog atributa od ključa te donosimo zaključak da relacija nije u 2NF.

- Vozilo_putnik(<u>Putnik</u>,<u>Vozilo</u>, Ime, Prezime, Naziv_vozila)
    - Putnik,Vozilo →Ime, Prezime, Naziv_vozila
    - Putnik →Ime, Prezime
    - Vozilo →Naziv_vozila

Da bi relacijska shema bila u 2NF potrebno je napraviti dekompoziciju skupa atributa R prema kojem ćemo napraviti više manjih relacija po zavisnosti ne ključnog atributa.

U polaznu relaciju ćemo dodati onoliko novih relacija koliko imamo primarnih ključeva koji sudjeluju u parcijalnim ovisnostima. Zatim ćemo iz polazne relacije izbaciti sve one atribute koji su parcijalno ovisni o ključu i dodati ih u novu relaciju. U novu relaciju idu svi oni atributi koji su ovisni o istom dijelu primarnog ključa. Na kraju u novu relaciju premještamo odgovarajući dio primarnog ključa koji postaje primarni ključ nove tablice. U našem primjeru ćemo dobiti tablicu Putnik sa atributima Šifra, Ime i Prezime te tablicu Vozilo_putnik sa atributom Putnik, što je vanjski ključ tablice Putnik, te Vozilo i Naziv vozila. U zadnjoj tablici primarni ključ je Putnik,Vozilo pa postoji zavisnost Vozilo → Naziv vozila što se krši sa pravilima 2NF stoga je potrebno izvršiti daljnju dekompoziciju po istim pravilima ali prema drugoj zavisnosti. Sada trebamo izvršiti dekompoziciju tablice Vozilo_putnik koja sadrži

atribute Putnik, što je primarni ključ ove relacije, te atribute Vozilo,Naziv. Nakon primjene pravila 2NF dobivamo tablicu Vozilo_putnik te tablicu Vozilo. Tablica Vozilo_putnik sadrži atribute Vozilo i Putnik što predstavlja vanjske ključeve tablica Vozilo i Putnik. Tablica Vozilo sadrži atribute Šifra, što predstavlja primarni ključ tablice, i Naziv vozila.

Nakraju dekompozicije imamo tri relacijske sheme Putnik, Vozilo i Vozilo_putnik koje su u 2NF.

- Putnik(Šifra, Ime, Prezime)
    - Šifra →Ime, Prezime
- Vozilo(Šifra, Naziv_vozila)
    - Šifra →Naziv_vozila
- Vozilo_putnik(Vozilo,Putnik)

3.5. Treća normalna forma

Relacijska shema (R, F) je u 3NF ako ne postoji tranzitivna zavisnost neključnog atributa od ključa. (R, F) je u 3NF ako je u 2NF i kad god imamo netrivijalnu zavisnost $X \rightarrow A$ onda X sadrži ključ za (R, F) ili A je ključni atribut.

Ako je entitet ili veza u drugoj normalnoj formi i postoji samo jedan atribut

- Student_detalji(JMBAG, Ime, Grad, Poštanski broj)
    - JMBAG →Ime, Grad, Poštanski broj
    - Poštanski broj → Grad

Kao što možemo primijetiti relacijska shema nije u 3NF iz razloga što postoji atribut koji ovisi o drugom atributu koji nije primarni ključ. U ovom slučaju to bi bio atribut Grad koji ovisi o atributu poštanski broj iz razloga što svaki grad ima jedinstveni poštanski broj. Dekompozicijom ove tablice dobivamo dvije tablice. Prva tablica koju smo dobili je tablica Student koja sadrži primarni ključ JMBAG te ne ključne atribute Ime i Poštanski broj koji je vanjski ključ na tablicu Pošta_grad koja je druga tablica nastala nakon dekompozicije. Tablica Pošta_grad koja sadrži primarni ključ Poštanski broj te ne ključni atribut Grad. Obje tablice su sada u 3NF.

- Student(JMBAG, Ime, Poštanski broj)
    - JMBAG →Ime, Poštanski broj
- Pošta_grad(Poštanski broj, Grad)
    - Poštanski broj →Grad

Ono što možemo primijetiti je da svaka tablica koja je u 3NF je također i u 2NF. To nam govori da ukoliko se relacijska shema nalazi u nekoj normalnoj formi ona se također nalazi i u svim

normalnim formama nižega reda. Tako se relacijska shema koja je u 3NF nalazi i u 2NF i u 1NF.

3.6. Boyce-Coddova normalna forma

Pokazalo se da je u nekima slučajevima 3NF potrebno dodati još neka ograničenja da bi se izbjegle anomalije. Tu dolazi do potrebe za Boyce-Coddovom normalnom formom. Ova forma je zasnovana na pojmu determinante.

„Determinanta je atribut(ili kombinacija atributa) unutar neke relacijske sheme o kojem je neki drugi atribut unutar iste relacije potpuno funkcionalno ovisan. Relacijska shema je u BCNF ako je svaka njezina determinanta ujedno kandidat za ključ.“(Manger, 2012, str. 56)

„(R,F) je u BCNF ako za svaku netrivijalnu funkcijsku zavisnost $X \rightarrow Y$ iz F^+ vrijedni : lijeva strana X sadrži ključ od (R,F).“(Maleković & Schatten, 2017)

Za primjer relacijske sheme(R,F) kojom ćemo prikazati kako možemo staviti tablicu u BCNF koristiti ćemo tablicu Upisao koja će sadržavati dvokomponentni primarni ključ koji sadrži atribute JMBAG,Šifra_predmeta te ne ključne atribute Naziv_predmeta, OIB_nastavnika, Broj_kabineta, Ocjena. Kada malo bolje pogledamo relaciju uočavamo sljedeće zavisnost: Šifra_predmeta→Naziv_predmeta,Šifra_predmeta→OIB_nastavnika,OIB_nastavnika→Broj_ kabineta. Kao što smo ranije naveli da je determinanta atribut unutar relacije o kojem je drugi atribut potpuno ovisan nije teško zaključiti da su determinante ove relacije Šifra_predmeta i OIB_nastavnika. No nijedan od ovih atributa ne predstavlja kandidata za ključ iz čega možemo zaključiti da relacijska shema nije u BCNF.

- Upisao(<u>JMBAG,Šifra_predmeta</u>, Naziv_predmeta, OIB_nastavnika, Broj_kabineta, Ocjena)
    - Šifra_predmeta →Naziv_predmeta
    - Šifra_predmeta → OIB_nastavnika
    - OIB_nastavnika→Broj_kabineta

Postupak prevođenja relacije u BCNF je sljedeći:

- Dodajemo, uz početnu relaciju, onoliko relacija koliko imamo determinanti
- Iz polazne relacije izbacujemo atribute koji imaju ovisnost o determinantama
- Svaki atribut ide u relaciju u kojoj je determinanta o kojoj je on ovisan
- Determinanta postaje ključ relacije u kojoj se nalaze o njoj ovisni atributi

Nakon prethodnih koraka dobivamo tri relacije koje se nalaze u BCNF. Prva relacija je Upisao koja sadrži primarni ključ JMBAG i Šifra_predmeta i ne ključni atribut Ocjena. Zatim relaciju Predmet koja sadrži primarni ključ Šifra_predmet te ne ključni atribut Naziv_predmeta te zadnja relacija Nastavnik koja sadrži primarni ključ OIB_nastavnika te ne ključni atribut Broj_kabineta.

- Upisao(JMBAG,Šifra_predmeta,Ocjena)
  - JMBAG→Šifra_predmeta
  - JMBAG→Ocjena
- Predmet(Šifra_predmeta,Naziv_predmeta)
  - Šifra_predmeta→Naziv_predmeta
- Nastavnik(OIB_nastavnika, Broj_kabineta)
  - OIB_nastavnika→

Na prvi pogled možemo pretpostaviti da BCNF zapravo ekvivalentna 2NF i 3NF iz razloga što imamo jako malo relacijskih shema koje nisu u BCNF a jesu u 2NF i 3NF. No iako svaka relacijska shema koja je u BCNF mora biti u 2NF i 3NF nije nužno da vrijedni obratno.

3.7. Normalne forme višeg reda

Pod pojmom normalnih formi višega reda uvrstio sam 4NF, 5NF, 6NF. Ove forme se ne koriste toliko u praksi ali dobro ih je poznavati ukoliko bude postojala potreba za jednom od njih. Preporuka je da se relacijska shema(R,F) dovede do najviše moguće normalne forme. Međutim, iz praktičnih razloga možemo se zadovoljiti i sa 3NF.

3.7.1. Četvrta normalna forma

Relacijska shema (R,F) se nalazi u 4NF ukoliko za svaku netrivijalnu višeznačnu zavisnost vrijedi da lijeva strana zavisnosti sadrži ključ relacije. Višeznačna zavisnost $X \rightarrow\rightarrow Y \in FVS(R)$ je netrivijalna ako $Y \not\subset X$ ili $XY \neq R$.

4NF ćemo obraditi na primjeru relacijske sheme Program koja sadrži atribute Predmet,Nastavnik i Knjiga.

- Program(Predmet, Nastavnik, Knjiga)
  - Predmet→Nastavnik
  - Predmet→Knjiga

Pretpostaviti ćemo da je ova relacijska shema u BCNF no mogu nastati problemi zbog višeznačne zavisnosti koja se odnosi na višeznačne zavisnosti unutar relacije: Predmet→

Nastavnik, Predmet→ Knjiga. Da bi izbjegli ovaj problem stvaramo dvije nove relacije, relaciju Raspored koja sadrži atribute Predmet,Nastavnik i relaciju Knjiga koja sadrži atribute Predmet i Knjiga.

- Raspored(Predmet,Nastavnik)
    - Predmet→Nastavnik
- Knjiga(Predmet, Knjiga)
    - Predmet→Knjiga

3.7.2.Peta normalna forma

(R,F) je u 5NF ako za svaku netrivijalnu zavisnost spoja (R1,...,R_k) ∈ F^+ vrijedi: svaka komponenta R_i, i=1, ..., k, sadrži neki ključ od (R,F). Zavisnost spoja ⋈(R_1, …, R_k) ∈ FVS(R) je netrivijalna ako je R_i ≠ R za svako i = 1, …, k. (Maleković & Schatten, 2017)

Petu normalnu formu lakše ćemo objasniti pomoću primjera. Koristiti ćemo se prethodnim primjerom gdje smo pretpostavili da nije važno koji nastavnik koristi koju knjigu. Ako ovakva veza postoji onda smo se dekompozicijom relacije Program na relacije Raspored i Knjiga izgubili informacije o tome koji nastavnik koristi koju knjigu. Ono što je trebalo napraviti je da dekomponiramo relacijsku shemu Program na 3 relacije. Dvije smo već imali u prošlom primjeru a nova relacija bi bila Nastavnik_knjiga koja bi sadržavala atribute Nastavnik i Knjiga.

- Nastavnik(Id,Nastavnik)
    - Predmet→Nastavnik
- Knjiga(Id, Knjiga)
    - Predmet→Knjiga
- Nastavnik_knjiga(Nastavnik, Knjiga)
-

3.7.3.Šesta normalna forma

Relacijska shema (R,F) je u šestoj normalnoj formi (6NF) ako i samo ako ne postoji netrivijalna zavisnost spoja.

(R,F) je u 6NF ako u F^+ nema netrivijalnih zavisnosti spoja.(Maleković & Schatten, 2017)

Šesta normalna forma se u većini slučajeva smatra suvišnom u praksi, osim ako radimo sa vremenskim, odnosno temporalnim bazama podataka.

4. Motivacija za normalizacijom baze podataka

Normalizacija je važna sastavnica baza podataka. Danas kada imamo velike sustave koji koriste jako velik broj tablica u svojim bazama podataka jako je bitno da se izbjegnu anomalije koje mogu naštetiti integritetu baze podataka. Jako velike poteškoće se javljaju kada želimo raditi sa nenormaliziranim podacima. Normalizacija nam je isto tako korisna kada želimo otkriti postoje li greške u oblikovanju etniteta, veza i atributa. Postoje slučajevi u kojima normalizacija nije poželjna i u tim se slučajevima odustaje od normalizacije. Glavni razlog provođenja normalizacije je redundancija. Možemo navesti tri tipa anomalija koje nas potiču da vršimo normalizaciju baze podataka.

4.1. Redundancija i anomalije

Redundancija se manifestira ponavljanjem podataka, ali nije svako ponavljanje podataka redundantno ponavljanje. Ukoliko se podatci ponavljaju a oni ne slijede iz danih podataka, to ponavljanje nije redundantno.

Da bi lakše razumjeli redundantnost dati ćemo jedan primjer na kojem je lako uočiti redundanciju.

Tablica 2: Tablica student

Student	*Ime*	*Prezime*	*Predmet ID*	*Učionica*
1	Ivan	Milas	01	03
2	Ante	Peric	02	04
3	Ivan	Milas	02	04
4	Mate	Matic	02	04

Vidimo se da podatak o premetu i učionici u kojoj se taj predmet izvodi ponavljaju stoga je to redundantni podatak.

Kao posljedicu redundancije imamo poteškoće u ažuriranju baze podataka.

4.1.1. Anomalija brisanja

Ukoliko izbrišemo podatke o studentu sa rednim brojem 2 gubimo podatak o pripadnom gradu što je nepoželjan efekt.

4.1.2. Anomalija upisivanja

Nemogućnost dodavanja grada ukoliko ne postoji niti jedan student koji živi u tom gradu.

4.1.3. Anomalija ažuriranja

Nemogućnost nezavisnog ažuriranja vrijednosti atributa grad.

4.2. Rad sa nenormaliziranim podaci

Da bismo lakše razumjeli motivaciju za normalizacijom podataka pokazati ćemo što se događa ukoliko su podaci nenormalizirani te koliko je teško raditi sa tim podacima.

Ako relacije nisu normalizirane, dolazi do anomalija. Bez obzira o kojoj je to normalnoj formi riječ, teškoće su otprilike slične.(Manger, 2012)

Uzeti ćemo za primjer relacija Vozilo_putnik s primarnim ključem Putnik,Vozilo i ne ključnim atributima Ime, Prezime i Naziv vozila ova relacija nije normalizirana stoga je dobar primjer da pokažemo koliko je teško raditi sa nenormaliziranim podacima. Ukoliko želimo dodati podatke o vozilu trebamo unijeti i podatke o Osobi, isto tako ukoliko želimo unijeti podatke o osobi moramo unijeti i podatke o vozilu.

Ukoliko bi htjeli promijeniti podatke o Nazivu vozila trebali bismo pronaći sve n-torke koje sadrže odgovarajuću vrijednost za ovaj atribut te ih izmijeniti. Imati ćemo onoliko promjena koliko imamo osoba koje posjeduju taj tip vozila.

Imamo i slučaj u kojem ako se nitko ne vozi u nekom automobilu izgubiti ćemo podatke o vozilu.

Još jedan primjer ćemo napraviti na relaciji Knjiga sa ključnim atributima Šifra_knjige, Naslov_knjige, OIB_autora, Adresa_autora koja nije u 3NF. Do poteškoća u radu sa ovom relacijom će doći ukoliko želimo unijeti novoga autora i njegovu adresu jer to ne možemo napraviti sve dok autor nema barem jednu knjigu koju piše. Zatim, da bismo promijenili adresu autora trebamo promijeniti vrijednosti u svakoj n-torki kojoj odgovara da je taj autor napisao tu knjigu. Ukoliko ne obavimo promjenu na svakoj n-torki imati ćemo kontradiktorne podatke unutar tablica. Još jedna anomalija nastaje ako autor nema ni jednu knjigu koju piše on će nestati iz baze.

4.3. Odustajanje od normalizacije

Prilikom pitanja odustajanja od normalizacije moramo se pitati je li nam poboljšanje značajki poput brzine izvršavanja upita ili lakoće održavanja baze podataka vrijedno nenormaliziranih podataka. Hoćemo li u kasnijem korištenju baze podataka imati problema koji će nastati zbog nenormaliziranih podataka.

U nekim slučajevima potrebno je denormalizirati podatke zbog efikasnosti baze podataka. Kada imamo primjer gdje je Ime_grada funkcionalno ovisno o poštanskom broju ovaj odnos se ne preporuča razbijati na dvije manje relacije iz razloga što oni zajedno čine smislenu cjelinu to jest adresu.

„Normalizacijom se velike relacije često razbijaju na potreban broj manjih relacija. U aplikacijama je često potrebno podatke iz manjih relacija ponovno spajati u veće nenormalizirane n-torke. Uspostavljanje veza među podacima u manjim relacijama traje znatno dulje nego čitanje podataka koji su već povezani i upisani u jednu veliku n-torku.“(Manger, 2012)

,

5. Algoritam sinteze 3NF

„Algoritam sinteze 3NF omogućuje sintezu sheme relacijske baze podataka, koja je u 3NF, čuva informaciju i zavisnosti. Algoritam koristi kanonski prekrivač skupa funkcijskih zavisnosti.“(Maleković & Schatten, 2017)

5.1. Kanonski pokrivač

Kanonski pokrivač od F, kp(F), dobije se iz F primjenom sljedeća tri koraka u navedenom redoslijedu

1) Desna dekompozicija

2) Lijeva redukcija

3) Izbacivanje redundantnih zavisnosti

Kanonski pokrivač dobivamo primjenom sljedeća tri koraka:

1) Desna dekompozicija

Desna dekompozicija se temelji na ekvivalenciji

$X \rightarrow A_1A_2...A_m = X \rightarrow A_1, X \rightarrow A_{2,...,} X \rightarrow A_m$

Možemo reći da smo dekompozicijom zavisnosti $X \rightarrow A_1A_2...A_m$ dobili skup zavisnosti $X \rightarrow A_1, X \rightarrow A_{2,...,} X \rightarrow A_m$.

Ukoliko imamo relaciju R=ABCD, te funkcijske zavisnosti F=AB→D, BC→A, B→D, A→CD primjenom dekompozicije dobiti ćemo skup zavisnosti *novi*(F) koji sadrži:

AB→D

BC→A

B→D

A→C

A→D

Sada imamo skup funkcijskih zavisnosti *novi*(F) koji je ekvivalentan skupu F. Možemo primijetiti da sve zavisnosti iz skupa *novi*(F) imaju na desnoj strani jedan atribut.

2) Lijeva redukcija

Lijevom redukcijom izbacujemo suvišne atributi sa lijeve strane zavisnosti. Po definiciji lijeve redukcije zavisnost X→Y je lijevo reducirana s obzirom na F ako u X nema suvišnih atributa, tj. ne postoji X1 →X : F X1 →Y.

3) Izbacivanje redundantnih zavisnosti

„Za zavisnost f ∈ F kažemo da je suvišna(redundantna) u F ako f slijedi iz preostalih zavisnosti u F, tj. ako vrijedi F-{f}=f. Za skup zavisnosti F kažemo da je redundantan skup ako ima suvišnih zavisnosti; u protivnom, F je neredundantan skup. U koraku[3] izbacuju se sve suvišne zavisnosti; rezultat je neredundantan skup zavisnosti.“(Maleković & Schatten, 2017)

5.2. Algoritam

Ulaz: relacijska shema (R, F), F ⊆FZ(R).

Izlaz: Dekompozicija relacije R na više relacija => d(R):R_1,...,R_k koje imaju sljedeća svojstva:

a. d(R) je 3NF dekompozicija
b. d(R) čuva informacije
c. d(R)čuva zavisnosti

Postupak za provođenje algoritma:

1. Računanje kanonskog pokrivača kp(F);
2. Sinteza komponenti R_1,...,R_k
3. Eventualno dodavanje ključa za (R,F)
4. Smanjenje broja komponenti dekompozicije

5.2.1.Sinteza komponenti

Sinteza komponenti bila bi kreiranje relacijske sheme iz funkcijskih zavisnosti. Ukoliko pretpostavimo da imamo kanonski pokrivač kp(F)= $X_1→A_1$,$X_2→A_2$,...,$X_K→A_K$ onda bi relacije koje bi dobili sintezom bile sljedeće:

$R_1 = X_1A_1$

$R_2 = X_2A_2$

$R_k = X_kA_k$

5.2.2.Eventualno dodavanje ključa

Ukoliko niti jedna od komponenti R_1, R_2,..., R_k iz prethodnog dijela ne sadrži ključ za (R,F), onda se računa ključ K za (R,F) te dodajemo K kao novu komponentu. Time ćemo dobiti dekompoziciju koja glasi d(R): R_1, R_2,..., R_k,K.

5.2.3.Smanjenje broja komponenti dekompozicije

Smanjenje broja komponenti dekompozicije možemo provesti na dva načina:

a) U 2. koraku algoritma umjesto da svakoj zavisnosti iz kp(R) pridružimo jednu komponentu , možemo svakoj grupi koja ima istu lijevu stranu pridružiti po jednu komponentu, koja se sastoji od zajedničke lijeve strane i preostalih atributa na desnim stranama zavisnosti iz grupe. Tako za primjer možemo uzeti kp(F):A→B, A→C, D→E,D→F,D→H iz kojih bi dobili R_1=ABC, R_2= DEFH.
b) Eliminacija svih onih komponenti dekompozicije koje su podskupovi neke druge komponente iz dekompozicije. Ukoliko imamo d(R): ABC,AB,DEF,DEFG, nakon eliminacije ostati će nam d_1(R): ABC, DEFG

6. Primjeri izračuna algoritma za sintezu 3NF

6.1. Zadatak 1.

Za primjer ćemo uzeti relaciju shemu (R,F) , gdje je R= ABCDE i skup zavisnosti :

F : A→BC,

AC→BD.

CD→A,

DE→B,

D→AC.

Trebamo oblikovati bazu podataka koja će čuvati informacije i zavisnosti te će zadovoljavati uvijete 3NF. To ćemo postići sa algoritmom sinteze 3NF.

6.2. Kanonski pokrivač

6.2.1. Desno razbijanje

Trebamo provesti desno razbijanje zavisnosti iz F. To radimo na način da sve zavisnosti tipa X→A dekomponiramo tako da A bude jednočlani atribut. Nakon što dekomponiramo naše zavisnosti imamo sljedeći skup zavisnosti G:

A→B,

A→C,

AC→B,

AC→D,

CD→A,

DE→B,

D→A,

D→C.

Nakon što smo proveli desno razbijanje na redu je lijeva redukcija.

6.2.2. Lijeva redukcija

Lijevu redukciju provodimo tako da sve redundantne atribute sa lijeve strane zavisnosti eliminiramo. Ako je FZ XB→A podskup od G, B je jednočlan atribut i X→A je logička posljedica od G, tada je B redundantan atribut.

Ispitujemo svaku zavisnost. Prva zavisnost je AC→B.

$(A)^+ = ABC$

$(C)^+ = C$

Vidimo da je B podskup od $(A)^+$. Prema definiciji, A→B je logička posljedica od G. Možemo primijetiti da je C redundantni atribut stoga njega eliminiramo iz zavisnosti i dobivamo zavisnost A→B.

Ispitujemo za AC→D.

$(A)^+ = ABCD$

$(C)^+ = C$

Možemo primijetiti da je D podskup od $(A)^+$, te da D nije podskup od $(C)^+ = C$ stoga po definiciji možemo zaključiti da je redundantni atribut C. Njega ćemo izbaciti iz zavisnosti te ćemo dobiti zavisnost A→D.

Sada ćemo ispitati zavisnost CD→A.

$(C)^+ = C$

$(D)^+ = DA$

Zaključujemo da je A podskup od $(D)^+$ stoga zaključujemo da je C redundantni atribut. Nakon izbacivanja suvišnog atributa ostaje nam relacija D→A.

Zadnja zavisnost koju moramo ispitati je DE→B.

$(D)^+ = DAB$

$(E)^+ = E$

Zaključujemo da je B podskup od $(D)^+$ stoga zaključujemo da je E redundantni atribut. Nakon izbacivanja suvišnog atributa ostaje nam relacija D→B.

6.2.3.Izbacivanje suvišnih zavisnosti

Za zavisnost f ∈ F kažemo da je suvišna(redundantna) u F ako f slijedi iz preostalih zavisnosti u F, tj. ako vrijedi F-{f}=f.

Ispitujemo zavisnost A→B dali je redundantna.

$(A)^+$ = ADB

Vidimo da je je zavisnost redundantna stoga ju izbacujemo.

Ispitujemo za zavisnost A→C, dali je redundantna.

$(A)^+$ = ABDC

Vidimo da je je zavisnost redundantna stoga ju izbacujemo.

Ispitujemo za zavisnost A→D, dali je redundantna.

$(A)^+$ = A

Vidimo da zavisnost nije redundantna stoga ostaje.

Ispitujemo za zavisnost D→A, dali je redundantna.

$(D)^+$ = DBC

Vidimo da zavisnost nije redundantna stoga ostaje.

Ispitujemo za zavisnost D→B, dali je redundantna.

$(D)^+$ = DAC

Vidimo da zavisnost nije redundantna stoga ostaje.

Ispitujemo za zavisnost D→C, dali je redundantna.

$(D)^+$ = DBA

Vidimo da zavisnost nije redundantna stoga ostaje.

Kp(F) : A→D,

D→B,

D→A,

D→C.

6.3. Sinteza komponenti

G: A→D : R1 = AD

D→B : R2 = DB

D→A : R3 = DA

D→C : R4 = DC.

6.4. Smanjenje broja komponenti dekompozicije

Relacije R2, R3, R4 možemo spojiti u jednu relaciju stoga će nam za kraj ostati dvije relacije:

R1 = AD, R2 = DABC. Ovo predstavlja smanjenje broja komponenti relacije. Sada trebamo provjeriti da li neka od komponenti R1, R2 sadrži ključ za (R, F). Ako ne sadrži, onda treba izračunati jedan ključ i dodati ga kao novu komponentu R3. U protivnome, ako bilo R1 ili R2 sadrži ključ, onda ključ ne treba dodavati, te ostajemo sa R1 i R2.

Računamo zatvarač za R1:

$(R1)_{F}^{+}$= ABCD

Računamo zatvarač za R2:

$(R2)_{F}^{+}$= ABCD

Vidimo da ove relacije ne sadrže ključ stoga dodajemo R3 = EA. Provjeravamo dali R3 sadrži ključ relacije:

$(R3)_{F}^{+}$= ABCDE

Smanjenjem broja komponenti dekompozicije dobivamo :

R1 = AD

R2 = DAB

R3 = EAD

7. Algoritam sinteze 3NF na realnom primjeru

Zadatak:

Nad relacijskom shemom R(#id_racuna,#id_artikla,naziv artikla, cijena artikla, jedinična mjera, količina, vrijeme izdavanja, iznos računa, #id_konobara, ime konobara) vrijede sljedeće zavisnosti:

#id_racuna→ vrijeme izdavanja, iznos racuna, #id_konobara,

#id_artikla→naziv artikla, cijena artikla, jedinična mjera

#id_konobara→ime konobara.

Da bismo imali veću preglednost prilikom rješavanja zadatka pridružiti ćemo atribute relacijske sheme velikim tiskanim slovima.

A - #id_racuna

B - #id_artikla

C - naziv artikla

D - cijena artikla

E - jedinična mjera

F - količina

G - vrijeme izdavanja

H - iznos računa

I - #id_konobara

J - ime konobara

Da pojednostavimo, imamo sljedeći problem:

R = ABCDEFGHIJ

F: A→GHI,

B→CDE,

I→J.

Izvršimo algoritam sinteze 3NF:

1. Kanonski pokrivač
    a. Desno razbijanje funkcijskih zavisnosti

A → G
A → H
A → I
B → C
B → D
B → E
I → J

    b. Lijeva redukcija
    Kao što možemo primijetiti na lijevoj strani funkcijskih zavisnosti nemamo višekomponentne atribute stoga nije potrebno raditi lijevu redukciju.

    c. Izbacivanje redundantnih zavisnosti
    Provjeravamo za svaku zavisnost posebno.
    Ispitujemo za zavisnost A → G, da li je redundantna.

    $(A)^+$ = AHIJ

    Vidimo da G nije element od $(A)^+$ stoga zavisnost A→ G nije redundantna i nju ne eliminiramo.
    Ispitujemo za zavisnost A → H, da li je redundantna.

    $(A)^+$ = AIJ

    Vidimo da H nije element od $(A)^+$ stoga zavisnost A→ H nije redundantna i nju ne eliminiramo.
    Ispitujemo za zavisnost A → I, da li je redundantna.

    $(A)^+$ = A

    Vidimo da I nije element od $(A)^+$ stoga zavisnost A→ I nije redundantna i nju ne eliminiramo.
    Ispitujemo za zavisnost B → C, da li je redundantna.
    $(B)^+$ = BDE

Vidimo da C nije element od $(B)^+$ stoga zavisnost B → C nije redundantna i nju ne eliminiramo.

Ispitujemo za zavisnost B → D, da li je redundantna.

$(B)^+ = BE$

Vidimo da D nije element od $(B)^+$ stoga zavisnost B → D nije redundantna i nju ne eliminiramo.

Ispitujemo za zavisnost B → E, da li je redundantna.

$(B)^+ = B$

Vidimo da E nije element od $(B)^+$ stoga zavisnost B → E nije redundantna i nju ne eliminiramo.

Ispitujemo za zavisnost I → J, da li je redundantna.

$(I)^+ = I$

Vidimo da J nije element od $(I)^+$ stoga zavisnost I → J nije redundantna i nju ne eliminiramo.

Sada je dobiveni skup zavisnosti ujedno i kanonski pokrivač od R :

Kp(F): A → G
A → H
A → I
B → C
B → D
B → E
I → J

2. Sinteza komponenti

   Iz kanonskog pokrivača sintezom komponenti dobivamo sljedeće skupove atributa :

   A → G : R1 = AG
   A → H : R2 = AH
   A → I : R3 = AI
   B → C : R4 = BC
   B → D : R5 = BD
   B → E : R6 = BE
   I → J : R7 = IJ

3. (Eventualno) dodavanje ključa uz smanjenje broja komponenti dekompozicije

   Atributi ABF nisu u desnoj strani niti jedne od zavisnosti iz F odnosno kp(F). Zato ABF moraju participirati u bilo kojem ključu za (R, F). Kako je (ABF)F+= ABFCEDGHIJ =

R, zaključujemo da je jedini ključ K = ABF. Budući niti jedna od komponenti R1, R2,.., R7 ne sadrži ključ K, dodajemo K kao
Novu komponentu R0 = ABF. Nakon smanjenja broja komponenti konačno dobivamo: R0, R1, R2, R3.
Provjerimo da li je ABF ključ za (R,F).
$(ABF)^+$ = ABGHICDEJ = R. ABF je ključ za (R,F). Dodajemo R0=ABF.

Smanjenjem broja komponenti dekompozicije dobivamo :

R0 = ABF

R1 = AGHI

R2 = BCDE

R3 = IJ

Zaključno:

Primjenom algoritma sinteze 3NF dobivamo shemu relacijske baze podataka koja izgleda ovako:

R0 = ABF ; R1 = AGHI ; R2 = BCDE ; R3 = IJ , gdje su podcrtani primarni ključevi. Ponovimo da je dobivena shema u 3NF te da čuva informaciju i zavisnosti.

Ako se vratimo na stvarna imena relacijskih shema i atributa dobivamo:

R0 = RAČUNI_ARTIKLI(ID_RAČUNA, ID_ARTIKLA, KOLIČINA)

R1 = RAČUNI(ID_RAČUNA, VRIJEME IZDAVANJA, IZNOS RAČUNA, ID_KONOBARA)

R2 = ARTIKLI(ID_ARTIKLA, NAZIV, CIJENA, JEDINIČNA MJERA)

R3 = KONOBAR(ID_KONOBARA, IME KONOBARA)

8. SQL implementacija

U ovom poglavlju napraviti ćemo primjer SQL implementacije nad primjerom iz poglavlja 7.

```
CREATE TABLE ARTIKLI
(ID_ARTIKLA NUMERIC(10) UNSIGNED NOT NULL,
NAZIV ARTIKLA VARCHAR(20),
CIJENA ARTIKLA DECIMAL(15,2),
JEDINIČNA MJERA DECIMAL(15,2),
PRIMARY KEY(ID_ARTIKLA));

CREATE TABLE KONOBAR
(ID_KONOBARA NUMERIC(10) UNSIGNED NOT NULL,
IME KONOBARA VARCHAR(35),
PRIMARY KEY(ID_KONOBARA));

CREATE TABLE RAČUNI
(ID_RACUNA NUMERIC(8) UNSIGNED NOT NULL,
VRIJEME IZDAVANJA DATE,
IZNOS RAČUNA DECIMAL(15,2),
ID_KONOBARA NUMERIC(10)
PRIMARY KEY(ID_RACUNA),
FOREIGN KEY (ID_KONOBARA) REFERENCES KONOBAR(ID_KONOBARA));

CREATE TABLE RAČUN_ARTIKLI
(ID_ARTIKLA NUMERIC(10) UNSIGNED NOT NULL,
ID_RACUNA NUMERIC(8) UNSIGNED NOT NULL,
KOLIČINA INT(10),
```

PRIMARY KEY (ID_ARTIKLA,ID_RAČUNA,KOLIČINA)

FOREIGN KEY (ID_ARTIKLA) REFERENCES ARTIKLI(ID_ARTIKLA)

FOREIGN KEY (ID_RAČUNA) REFERENCES RAČUNI(ID_RAČUNA));

Možemo dodati indekse zbog lakšeg pretraživanja. Dodati ćemo indekse za pretraživanje računa prema datumu, pretraživanje zaposlenika prema imenu te artikala iz nekog računa.

8.1. Indexi

CREATE INDEX RD_IND ON RAČUN(DATUM);

CREATE INDEX ZI_IND ON KONOBAR(IME);

CREATE INDEX AR_IND ON RAČUN_ARTIKLI(ID_ARTIKLA);

8.2. Jednostavni upiti

Sada ćemo postaviti par upita da vidimo kako bi to izgledalo.

Prvi upit koji ćemo postaviti biti će traženje svih proizvoda koji imaju cijenu veću od 50kn.

SELECT NAZIV ARTIKLA FROM ARTIKLI WHERE CIJENA > 50;

Drugi upit bi bio za ispisati sve podatke o konobarima čije ime počinje na slovo m.

SELECT * FROM KONOBAR WHERE IME KONOBARA LIKE 'M%';

8.3. Složeni upit

Ispis konobara koji je izdao račun pod rednim brojem 25.

SELECT ID_KONOBARA, IME KONOBARA

FROM KONOBAR WHERE ID_KONOBARA IN

(SELECT ID_KONOBARA FROM RAČUNI WHERE ID_RAČUNA = 25);

9. Zaključak

Glavna tema ovoga rada bio je algoritam sinteze 3NF i njegova primjena kao i implementacija. Da bi lakše objasnio algoritam morao sam pažnju posvetiti relacijskom modelu kao i normalnim formama koje sam obradio u trećem poglavlju moga rada.

Ovaj rad daje dobru podlogu za shvaćanje načina rada algoritma. U primjerima su temeljito objašnjeni postupci primjene algoritma. Na praktičnom primjeru sam želio na što prirodniji primjer pokazati kako funkcionira algoritam sinteze 3NF.

Daljnje istraživanje ovog algoritma dovelo bi i do izrade algoritma sinteze 3NF u nekom od programskih jezika te fizička implementacija baze podataka.

Popis literature

Garcia Molina, H., D.Ullman, J., & Widom, J. (2009). Database systems.

Krstev, C. (2012, prosinac 25). Materijali.

Maleković, M. (2016). Nastavni materijali. Preuzeto 15. veljača 2018., od https://elfarchive1617.foi.hr/course/view.php?id=26

Maleković, M., & Schatten, M. (2017). *Teorija i primjena baza podataka*.

Manger, R. (2012). *Baze podataka*.

Popis tablica