7

ZAVISNOSTI I NORMALNE FORME

U svim prethodnim poglavljima smo polazili od datih šema relacija i šeme relacione baze podataka BIBLIOTEKA, ne ulazeći u to zašto je njihova struktura baš takva kakva je. Iz primera za upite, kao i sadržaja baze podataka BIBLIOTEKA, nismo mogli da uočimo nikakve nepogodnosti. U tom smislu, mogli smo da zaključimo da su te šeme kao i sama baza podataka"dobre" strukture.

U poglavlju 4 (elementi relacionog modela) uveli smo formalne definicije:

- šema relacije: skup atributa i ograničenja nad njima;
- šema relacione baze podataka: skup šema relacije i ograničenja nad njima.

Jedina ograničenja koja smo tada razmatrali bila su podrazumevana:

- unikatnost atributa u šemi relacije i unikatnost torki u relaciji (proizilazi iz skupovnog karaktera njihovih definicija);
- uslov identifikacionog integriteta za primarni ključ (ni jedan atribut u sastavu primarnog ključa ne sme imati NULL vrednost u relaciji);
- uslov referencijalnog integriteta za strani ključ (svaki strani ključ u relaciji može imati ili vrednost primarnog ključa u ciljnoj relaciji ili NULL vrednost ako je dozvoljena).

Osim poštovanja navedenih ograničenja, sadržaj svake relacije je u pogledu vrednosti ostalih atributa bio potpuno proizvoljan.

Iz prakse su poznati primeri šema relacija koje su "loše" strukture, u tom smislu da se kod relacija nad njima javlja niz nepogodnosti. Vrlo brzo je uočen i osnovni razlog za to: između atributa šeme relacije mogu postojati određene zavisnosti koje ograničavaju vrednosti tih atributa u torkama relacije. Na osnovu tih saznanja, koja su prvo bila opisnog karaktera, vremenom je formulisana posebna oblast istraživanja pod nazivom "Teorija zavisnosti". U okviru te teorije formulisane su i tzv. "normalne forme" kao kriterijumi za valjanost šeme relacije, kao i postupci "normalizacije", odnosno dekompozicije šeme relacije "loše strukture" na dve ili više šema relacija koje su sve u skladu sa željenom "normalnom formom".

7.1 Šeme relacija loše strukture

Loše strukture šeme relacije ilustrovaćemo kroz nekoliko pogodno odabranih primera izvedenih iz baze podataka BIBLIOTEKA. Ukazaćemo na nepogodnosti koje se pri tome javljaju

Primer 1

Posmatrajmo šemu relacije koja je rezultat nastojanja da se šema relacione baze podataka BIBLIOTEKA smanji tako što će se podaci o autorstvu naslova i autorima evidentirati u jednoj jedinoj relaciji šeme

AUTOR (SIFA, IME, SIFN, KOJI)

u kojoj je primarni ključ SIFA,SIFN , s obzirom da je određeni autor samo jednom autor određenog naslova.

U nekom trenutku u prošlosti, kada baza podataka BIBLIOTEKA nema konačan sadržaj dat u prilogu A, mogući sadržaj relacije **autor** bio bi

```
autor ( SIFA  IME         SIFN  KOJI )
        ---------------------------
        JN0   J.Nikolic   RBP0  2
        ZP0   Z.Petrovic  PP00  1
        ---------------------------
```

Pretpostavimo sada da u bazu podataka želimo da unesemo podatke o autorstvu naslova "PJC0 Programski jezik C", koji ima dva autora, kako je dato u prilogu A. Nakon toga, imali bi sledeći sadržaj relacije **autor**:

```
autor ( SIFA  IME         SIFN  KOJI )
        ---------------------------
        JN0   J.Nikolic   RBP0  2
        ZP0   Z.Petrovic  PP00  1   *
        AP1   A.Petrovic  PJC0  1
        ZP0   Z.Petrovic  PJC0  2   *
        ---------------------------
```

U budućnosti, mogla bi se javiti situacija da unosimo podatke o tome da je za neki naslov autor ili koautor "ZP0 Z.Petrović", nakon čega bi u relaciji **autor** imali 3 puta uneto ime Z.Petrović. U realnim okolnostima, uz imena bi evidentirali i neki dodatni podatak o autorima, i taj podatak bi se za Z.Petrovića takoće javljao 3 puta u relaciji (torke označene sa *).

Ako bi pokušali da izbegnemo višestruko unošenje imena autora Z.Petrovića tako što bi drugi i svaki naredni put unosili samo vrednost atributa SIFA , a za IME ostavljali NULL vrednost, stvorili bi novi problem: gubitak informacija. Naime, jedan broj upita, na primer "imena svih autora određenog naslova" ili "svi naslovi autora određenog imena" davao bi nepotpune informacije.

Iz navedenog jednostavnog primera, evidentan je osnovni nedostatak relacije **autor** , koji je posledica strukture šeme relacije AUTOR:

- redundansa: višestruko ponavljanje istog podatka u relaciji.

Ovaj nedostatak se ogleda kod sva tri vida ažuriranja relacije:

- višestruko unošenje: ime autora moramo uneti onoliko puta koliko je on napisao naslova
- višestruko menjanje: eventualnu promenu imena autora (ili nekog drugog podatka, recimo adrese) moramo izvršiti onoliko puta koliko je on napisao naslova;
- višestruko uklanjanje: ako želimo da potpuno uklonimo podatke o autoru, to moramo učiniti onoliko puta koliko je on napisao naslova.

Uz navedene nedostatke ažuriranja prisutna su i dva drastična nedostatka čiji je karakter egzistencijalne prirode:

- anomalija unošenja: ne možemo uneti podatke o nekom autoru ako pri tome ne unesemo i podatke o bar jednom njegovom naslovu;
- anomalija uklanjanja: uklanjanjem podatka o jedinom naslovu koji je napisao neki autor uklanjamo i podatke o tom autoru.

Svi nedostaci koje smo naveli posledica su okolnosti da pored podrazumevanih ograničenja nad šemom relacije AUTOR postoji i jedno dodatno:

- svakoj vrednosti atributa SIFA koji je deo primarnog ključa odgovara jedna vrednost atributa IME, odnosno kada god se u relaciji **autor** ponovi vrednost atributa SIFA mora se ponoviti i njoj odgovarajuća vrednost atributa IME.

Za razliku od posmatrane šeme relacije AUTOR, par šema relacija AUTOR i JE_AUTOR iz šeme relacione baze podataka BIBLIOTEKA nema navedene nedostatke: ime svakog autora unosi se samo jednom i nezavisno od podataka o naslovu.

Primer 2

Neka je u nameri da se podaci o naslovima i oblastima evidentiraju u jednoj relaciji nastala šema relacije NASLOV sledeće strukture:

NASLOV (SIFN, NAZIVN, SIFO, NAZIVO)

Za stanje koje odgovara sadržaju baze podataka BIBLIOTEKA: odgovarajuća relacija ***naslov*** bi bila:

```
naslov ( SIFN  NAZIVN                   SIFO NAZIVO           )
         ------------------------------------------------------
         RBP0  Relacione baze podataka  BP   Baze podataka
         RK00  Racunarske komunikacije  RM   Racunarske mreze
         PP00  PASCAL programiranje     PJ   Programski jezici *
         PJC0  Programski jezik C       PJ   Programski jezici *
          -----------------------------------------------------
```

Ako bi uneli podatke o novom naslovu, na primer CPP0 "C++", imali bi 3 puta unete nazive oblasti PJ "Programski jezici". Pokušaj izbegavanja višestrukog unošenja podataka na način kao u prethodnom primeru doveo bi do istog neželjenog efekta - gubitka informacija.

Evidentno je da su u i relaciji **naslov** prisutni nedostaci višestrukog ažuriranja i anomalija unošenja i uklanjanja, kao i anomalije egzistencije. Uzrok tome je postojanje dodatnog ograničenja nad šemom relacije NASLOV:

- svakoj vrednosti atributa SIFO koji nije deo primarnog ključa odgovara jedna vrednost atributa NAZIVO koji takođe nije deo primarnog ključa.

Navedeni nedostaci nisu prisutni u šemi relacione baze podataka BIBLIOTEKA, kod koje umesto jedne šeme relaciie NASLOV imamo dve šeme, NASLOV i OBLAST.

Primer 3

Posmatrajmo šemu relacije koja evidentira podatke o pozajmicama i o tome koje knjige postoje i sa kojim naslovima (posebna šema relacije KNJIGA ne postoji):

POZAJMICA (SIFN, SIFC, DATUM, DANA, SIFK)

Primarni ključ ove šeme je odabran uz pretpostavku da jedan član nikada istog dana ne uzima dva ili više puta knjigu istog naslova.

Za stanje koje odgovara sadržaju baze podataka BIBLIOTEKA uz dodatnu pozajmicu knjige 004 naslova PJC0 od strane člana JJ1 u trajanju od 2 dana, odgovarajuća relacija **pozajmica** bila bi

pozajmica (SIFN SIFC DATUM DANA SIFK)

SIFN	SIFC	DATUM	DANA	SIFK
PJC0	JJ0	01.09.95	5	004 *
PP00	PP0	02.09.95	2	007
PJC0	JJ1	03.09.95	6	005
PP00	JJ0	04.09.95	7	008
RBP0	PP0	05.09.95	4	002
PP00	JJ1	06.09.95	3	009
PJC0	JJ1	07.09.95	2	004 *

I u ovakvoj relaciji **pozajmica** su prisutni svi do sada navedeni nedostaci egzistencijalnog karaktera i neostaci sva tri vida ažuriranja, ali u nešto blažoj formi, i to iz sledećih razloga:

- atribut SIFN je kao primarni ključ u šemi relacije NASLOV kompaktan, pa je višestruko ponavljanje njegove vrednosti prihvatljivo;
- atribut SIFN je kao primarni ključ u šemi relacije NASLOV stabilan; njegova naknadna promena je nemoguća ili vrlo malo verovatna.

I u ovom slučaju uzrok nepogodnosti leži u postojanju jednog dodatnog ograničenja nad šemom relacije

- svakoj vrednosti atributa SIFK koji nije deo primarnog ključa odgovara jedna vrednost atributa SIFN koji je deo primarnog ključa.

Nedostaci koje smo uočili u ovom primeru nisu ispoljeni kod šeme relacione baze podataka BIBLIOTEKA , gde umesto šeme relacije POZAJMICA imamo dve šeme, POZAJMICA i KNJIGA.

Zaključak

Iz prethodna 3 primera, kao i osobina primarnog ključa, možemo da zaključimo sledeće:

- ako nad nekom šemom relacije postoje situacije da jednoj vrednosti nekog atributa koji nije primarni ključ odgovara jedna vrednost nekog drugog atributa, usled višestrukog pojavljivanja ove druge vrednosti javljaće se problemi pri unošenju, menjanju i uklanjanju torki;
- okolnost da nad nekom šemom relacije jednoj vrednosti primarnog ključa odgovara jedna vrednost nekog atributa ne može da dovede do višestrukog pojavljivanja ove druge vrednosti, pošto se svaka vrednost primarnog ključa javlja samo jednom u relaciji;
- šema relacije nad kojom postoje neželjene veze između vrednosti atributa može se zameniti sa više šema relacija kod kojih ti nedostaci nisu prisutni.

Dakle, rešenje prethodno opisanih problema je u dekompoziciji šeme relacije, odnosno rastavljanju šeme relacije na više (za sada dve) šema relacija. Međutim, dekompozicija jedne šeme relacije može se sprovesti na više načina. Koje dekompozicije su loše a koje dobre i zašto?

7.2 Loše i dobre dekompozicije

Pre nego što razmotrimo konkretne primere, neophodno je nekoliko napomena opšte prirode:

- dekompozicija je dvojaka: dekomponuje se i šema relacije i relacija nad njom;
- dekompozicija mora biti takva da su u nastalim šemama relacije prisutni svi atributi polazne šeme, pošto bi u suprotnom došlo do gubitka dela podataka;
- dekompozicijom šeme relacije u nastalim šemama relacija se ne mogu pojaviti atributi koji se ne nalaze u polaznoj šemi;
- u najnepovoljnijem slučaju, relacija koja se dekomponuje nije prazna, pa njen sadržaj treba preneti dekompozicijom u nastale relacije.

Kao osnova za primere poslužiće nam šema relacije POZAJMICA i relacija **pozajmica** iz prethodnog odeljka. Pri tome, nastojaćemo da eliminišemo atribut SIFN iz šeme POZAJMICA, kako bi otklonili ranije navedene nepogodnosti.

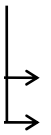
Primer 1

Ako za polaznu relaciju

```
pozajmica ( SIFN  SIFC  DATUM    DANA  SIFK )
          ------------------------------
          PJC0  JJ0   01.09.95  5     004
          PP00  PP0   02.09.95  2     007
          PJC0  JJ1   03.09.95  6     005
          PP00  JJ0   04.09.95  7     008
          RBP0  PP0   05.09.95  4     002
          PP00  JJ1   06.09.95  3     009
          PJC0  JJ1   07.09.95  2     004
          ------------------------------
```

sprovedemo dekompoziciju

POZAJMICA (SIFN, SIFC, DATUM, DANA, SIFK)



POZ1 (SIFC, DATUM, DANA, SIFK)

POZ2 (SIFN)

odgovarajuće relacije **poz1** i **poz2** dobićemo kao rezultate operacija projekcije po odgovarajućim podsskupovima atributa

$$\pi_{SIFC,DATUM,DANA,SIFK}\ (\textbf{pozajmica}) \rightarrow \textbf{poz1}\ (SIFC,DATUM,DANA,SIFK)$$

$$\pi_{SIFN}\ (\textbf{pozajmica}) \rightarrow \textbf{poz2}\ (SIFN)$$

sa sledećim sadržajem

```
poz1 ( SIFC  DATUM     DANA  SIFK )
       -------------------------
       JJ0   01.09.95  5     004
       PP0   02.09.95  2     007
       JJ1   03.09.95  6     005
       JJ0   04.09.95  7     008
       PP0   05.09.95  4     002
       JJ1   06.09.95  3     009
       JJ1   07.09.95  2     004
       -------------------------
```

```
poz2 ( SIFN )
       ----
       PJC0
       PP00
       RBP0
       ----
```

Uvidom u sadržaj nastalih relacija možemo zaključiti sledeće:

- veza između šifara naslova i ostalih podataka o pozajmicama ne postoji direktno ni u jednoj od nastalih relacija;
- ako pokušamo da rekonstruišemo sve podatke o pozajmicama prirodnim spajanjem relacija **poz1** i **poz2** dobićemo Dekartov proizvod sa 21 torkom (tabela koja sledi), pošto njihove šeme relacija nemaju ni jedan zajednički atribut; od tih torki samo 7 odgovara prvobitnom sadržaju relacije **pozajmica** , a preostalih 14 su suvišne (označeno sa ?):

pozajmica (SIFN SIFC DATUM DANA SIFK)

SIFN	SIFC	DATUM	DANA	SIFK
PJC0	JJ0	01.09.95	5	004
PJC0	PP0	02.09.95	2	007 ?
PJC0	JJ1	03.09.95	6	005
PJC0	JJ0	04.09.95	7	008 ?
PJC0	PP0	05.09.95	4	002 ?
PJC0	JJ1	06.09.95	3	009 ?
PJC0	JJ1	07.09.95	2	004
PP00	JJ0	01.09.95	5	004 ?
PP00	PP0	02.09.95	2	007
PP00	JJ1	03.09.95	6	005 ?
PP00	JJ0	04.09.95	7	008
PP00	PP0	05.09.95	4	002 ?
PP00	JJ1	06.09.95	3	009
PP00	JJ1	07.09.95	2	004 ?
RBP0	JJ0	01.09.95	5	004 ?
RBP0	PP0	02.09.95	2	007 ?
RBP0	JJ1	03.09.95	6	005 ?
RBP0	JJ0	04.09.95	7	008 ?
RBP0	PP0	05.09.95	4	002
RBP0	JJ1	06.09.95	3	009 ?
RBP0	JJ1	07.09.95	2	004 ?

Za navedeni primer evidentno je da je dekompozicija dovela do gubitka podataka. Konkretno, dobili smo veći broj torki od kojih samo neke odgovaraju stvarnom stanju, ali je pitanje koje? Uzrok gubitka informacija i nastajanja suvišnih n-toski pri prirodnom spajanju je evidentan:

- dekomponovane relacije nemaju ni jedan zajednički atribut (presek njihovih atributa je prazan skup). pa se svaka torka relacije **poz2** spaja sa svakom torkom relacije **poz1** , i obrnuto.

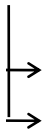
Primer 2

Ako za istu relaciju

```
pozajmica ( SIFN  SIFC  DATUM    DANA  SIFK )
          ------------------------------
          PJC0  JJ0   01.09.95  5    004
          PP00  PP0   02.09.95  2    007
          PJC0  JJ1   03.09.95  6    005
          PP00  JJ0   04.09.95  7    008
          RBP0  PP0   05.09.95  4    002
          PP00  JJ1   06.09.95  3    009
          PJC0  JJ1   07.09.95  2    004
          ------------------------------
```

sprovedemo dekompoziciju

POZAJMICA (SIFN, SIFC, DATUM, DANA, SIFK)



POZ1 (SIFC, DATUM, DANA, SIFK)

POZ2 (SIFC, SIFN))

relacije **poz1** i **poz2** dobićemo kao rezultate odgovarajućih operacija projekcije, sa sledećim sadržajem

```
poz1 ( SIFC  DATUM     DANA  SIFK )
       -------------------------
       JJ0   01.09.95  5     004
       PP0   02.09.95  2     007
       JJ1   03.09.95  6     005
       JJ0   04.09.95  7     008
       PP0   05.09.95  4     002
       JJ1   06.09.95  3     009
       JJ1   07.09.95  2     004
       -------------------------
```

```
poz2 ( SIFC  SIFN )
       ----------
       JJ0   PJC0
       PP0   PP00
       JJ1   PJC0
       JJ0   PP00
       PP0   RBP0
       JJ1   PP00
       ----------
```

Ovoga puta, nastale šeme relacija imaju kao presek zajednički atribut SIFC. Za ovaj slučaj možemo zaključiti:

- veza između šifara naslova i ostalih podataka o pozajmicama i dalje ne postoji direktno ni u jednoj od nastalih relacija;
- ako pokušamo da rekonstruišemo sve podatke o pozajmicama prirodnim spajanjem relacija **poz2** i **poz1** dobićemo relaciju sa 14 torki (tabela na sledećoj strani); od tih torki samo 7 odgovara prvobitnom sadržaju relacije **pozajmica** , a preostalih 7 su suvišne.

Gubitak podataka koji je nastupio i pored postojanja zajedničkog atributa između nastalih relacija može se objasniti na sledeći način:

- određene vrednosti zajedničkog atributa SIFC pojavljuju se više od jedanput u obe relacije, pa se neke torke relacije **poz2** spajaju sa više torki relacije **poz1**, i obrnuto.

pozajmica (SIFN SIFC DATUM DANA SIFK)

SIFN	SIFC	DATUM	DANA	SIFK
PJC0	JJ0	01.09.95	5	004
PJC0	JJ0	04.09.95	7	008 ?
PP00	PP0	02.09.95	2	007
PP00	PP0	05.09.95	4	002 ?
PJC0	JJ1	03.09.95	6	005
PJC0	JJ1	06.09.95	3	009 ?
PJC0	JJ1	07.09.95	2	004
PP00	JJ0	01.09.95	5	004 ?
PP00	JJ0	04.09.95	7	008
RBP0	PP0	02.09.95	2	007 ?
RBP0	PP0	05.09.95	4	002
PP00	JJ1	03.09.95	6	005 ?
PP00	JJ1	06.09.95	3	009
PP00	JJ1	07.09.95	2	004 ?

Iz svih prethodnih primera možemo naslutiti osnovni kriterijum za očuvanje podataka pri dekompoziciji šeme relacije i relacije:

- dekompozicija je bez gubitaka podataka ako je reverzibilna, odnosno ako se prirodnim spajanjem nastalih relacija dobija polazna relacija.

Primer 3

U prethodnim primerima smo uvideli da je višestruko spajanje torki nastalih relacija osnovni uzročnik gubitka podataka pri dekompoziciji, pa se kao prirodno rešenje tog problema nameće dekompozicija kod koje će se u bar jednoj od nastalih relacija svaka vrednost zajedničkog atributa javljati samo jednom.

Prethodni uslov se može formulisati na sledeći način: zajednički atribut (u opštem slučaju skup zajedničkih atrubita) treba da je kandidat-ključ u bar jednoj od nastalih relacija.

Sledeći takav pristup, za istu polaznu relaciju

pozajmica (SIFN SIFC DATUM DANA SIFK)

SIFN	SIFC	DATUM	DANA	SIFK
PJC0	JJ0	01.09.95	5	004
PP00	PP0	02.09.95	2	007
PJC0	JJ1	03.09.95	6	005
PP00	JJ0	04.09.95	7	008
RBP0	PP0	05.09.95	4	002
PP00	JJ1	06.09.95	3	009
PJC0	JJ1	07.09.95	2	004

dobijamo dekompoziciju koja odgovara onoj u bazi podataka BIBLIOTEKA:

POZAJMICA (SIFN, SIFC, DATUM, DANA, SIFK)

- **POZ1 (SIFC, DATUM, DANA, SIFK)**
- **POZ2 (SIFK, SIFN))**

pri čemu se dobija sledeći sadržaj relacija **poz1** i **poz2**:

poz1 (SIFC DATUM DANA SIFK)

SIFC	DATUM	DANA	SIFK
JJ0	01.09.95	5	004
PP0	02.09.95	2	007
JJ1	03.09.95	6	005
JJ0	04.09.95	7	008
PP0	05.09.95	4	002
JJ1	06.09.95	3	009
JJ1	07.09.95	2	004

poz2 (SIFK SIFN)

SIFK	SIFN
004	PJC0
007	PP00
005	PJC0
008	PP00
002	RBP0
009	PP00

Za ovaj slučaj važi sledeće:

- veza između šifara naslova i ostalih podataka o pozajmicama i dalje ne postoji direktno ni u jednoj od nastalih relacija;
- ako pokušamo da rekonstruišemo sve podatke o pozajmicama prirodnim spajanjem relacija **poz2** i **poz1** dobićemo 7 torki koje odgovaraju prvobitnom sadržaju relacije **pozajmica** (tabela koja sledi), pošto se svaka torka relacije **poz1** spaja sa tačno jednom torkom relacije **poz2**:

pozajmica (SIFN SIFC DATUM DANA SIFK)

SIFN	SIFC	DATUM	DANA	SIFK
PJC0	JJ0	01.09.95	5	004
PP00	PP0	02.09.95	2	007
PJC0	JJ1	03.09.95	6	005
PP00	JJ0	04.09.95	7	008
RBP0	PP0	05.09.95	4	002
PP00	JJ1	06.09.95	3	009
PJC0	JJ1	07.09.95	2	004

Zaključak

Neka su R i $\mathbf{r}$ šema relacije i relacija koje želimo da dekomponujemo na šeme relacija R_1 i R_2, odnosno na relacije $\mathbf{r_1}$ i $\mathbf{r_2}$, pri čemu postoji bar jedan zajednički atribut, odnosno važi $R_1 \cap R_2 \neq \varnothing$. Tada važi:

- uslov očuvanja atributa pri dekompoziciji možemo formulisati kao

$$R_1 \cup R_2 = R$$

- očuvanje podataka pri dekompoziciji (reverzibilnost) možemo u notaciji relacione algebre izraziti kao

$$\pi_{R1}(\mathbf{r}) >\!*\!< \pi_{R2}(\mathbf{r}) = \mathbf{r}$$

- dekompozicija je sa očuvanjem podataka (bez gubiraka podataka, reverzibilna) ako je skup zajedničkih atributa nastalih relacija kandidat-ključ u bar jednoj od nastalih relacija, odnosno ako je zadovoljeno (značenje simbola "$\rightarrow$" je "jednoznačno određuje"):

$$R_1 \cap R_2 \rightarrow R_1 \ \vee \ R_1 \cap R_2 \rightarrow R_2$$

7.3 Osnovi terorije funkcijskih zavisnosti

Do sada smo u više navrata pominjali zavisnosti između atributa šeme relacije koje se svode na to da svakoj vrednosti jednog atributa uvek odgovara samo jedna vrednost drugog atributa. U opštem slučaju, to može važiti i između vrednosti podskupova atributa šeme relacije. Za takvu situaciju se kaže da predstavlja funkcijsku zavisnost, i ona se može formalno definisati na više načina

7.3.1 Definicija funkcijske zavisnosti

Neka su:

R	šema relacije nad nekim skup atributa;
r	relacija nad šemom R;
X,Y,Z	podskupovi atributa relacije R;
R[Z]	šema relacije nad podskupom atributa Z;
r[Z]	projekcija relacije r po podskupu atributa Z;
t_i	jedna torka relacije r
t_i[Z]	projekcija jedne torke relacije r po podskupu atributa Z.

Definicija 1

Nad šemom relacije R postoji funkijska zavisnost $X \rightarrow Y$ ako u relacji **r** nad tom šemom uvek važi da je **r** [XY] funkcija. Tada kažemo da Y funkcijski zavisi od X , odnosno da X funkcijski uslovljava Y.

Definicija 2

Nad šemom relacije R postoji funkcijska zavisnost $X \rightarrow Y$ ako u relaciji **r** nad tom šemom uvek važi da se svaki element skupa **r** [X] preslikava na samo jedan element skupa **r** [Y].

Definicija 3

Nad šemom relacije R postoji funkcijska zavisnost $X \rightarrow Y$ ako u relaciji r nad tom šemom za bilo koje dve torke t_1 i t_2 za koje je $t_1[X] = t_2[X]$ uvek važi da je i $t_1[Y] = t_2[Y]$. Formalno, ovo možemo iskazati kao:

$$\forall t_1 t_2 \, (\, (\, t_1 \in r \wedge t_2 \in r \wedge t_1[X] = t_2[X] \,) \Rightarrow t_1[Y] = t_2[Y] \,)$$

Napomene

U vezi pojma funkcijske zavisnost i prethodnih definicija treba naglasiti:

- Podskupovi atributa X i Y iz zavisnost mogu biti bilo koji podskupovi za koje važi $X \subseteq R$ i $Y \subseteq R$, što uključuje i specijalne slučajeve njihove jednakosti sa R;
- X i Y ne moraju biti disjunktni, odnosno može važiti $X \cap Y \neq \varnothing$, što uključuje i specijalan slučaj $X = Y$;
- Funkcijska zavisnost $X \rightarrow Y$ je nešto što postoji "po prirodi stvari" i što proizilazi iz značenja svojstava koja odgovaraju X i Y i odnosa između njih; samo na osnovu okolnosti da u nekoj relaciji **r** u nekom trenutku svakoj vrednosti X odgovara samo jedna vrednost Y ne možemo zaključiti da će to važiti uvek (primer: iz sadržaja relacije **pozajmica** baze podataka BIBLIOTEKA sledi da svakoj vrednosti atributa SIFK odgovara samo jedna vrednost podskupa atributa SIFP,SIFC,SIFN,DANA, ali se to može narušiti kada se ista knjiga ponovo pozajmi);
- Na osnovu trenutnog sadržaja neke relacije **r** ne možemo prema prethodnom utvrditi postojanje neke funkcijske zavisnosti, ali zato možemo pouzdano utvrditi da neka pretpostavljena funkcijska zavisnost $X \rightarrow Y$ ne važi; treba samo da formiramo projekcije **r**[X] i **r**[XY] i zatim proverimo da li one sadrže isti broj torki. Ako to nije slučaj, ne važi $X \rightarrow Y$, pošto bar jednoj vrednosti X odgovaraju dve ili više vrednosti Y.

7.3.2 Izvođenje funkcijskih zavisnosti

U praksi je vrlo brzo uočeno da na osnovu postojanja nekog skupa funkcijskih zavisnosti mogu da se izvedu dodatne funkcijske zavisnosti. Neka kao ilustracija posluže sledeća dva jednostavna primera:

Primer

Posmatrajmo šemu relacije POZAJMICA iz baze podataka BIBLIOTEKA. Okolnost da je atribut SIFP primarni ključ možemo izraziti kao

$$SIFP \rightarrow SIFC,SIFK,SIFN,DANA$$

Na osnovu okolnosti da se svaka vrednost atributa SIFP pojavljuje samo jednom u relaciji **pozajmica** možemo zaključiti i da svakoj vrednosti atributa SIFP odgovara samo po jedna vrednost atributa SIFC, SIFK , SIFN i DANA pojedinačno, odnosno da važi

$$SIFP \rightarrow SIFC \qquad SIFP \rightarrow SIFK \qquad SIFP \rightarrow SIFN \qquad SIFP \rightarrow DANA$$

Primer

Posmatrajmo šemu relacije NASLOV iz jednog od naših ranijih primera:

NASLOV (<u>SIFN</u>, NAZIVN, SIFO, NAZIVO)

Neka jednoj šifri naslova odgovara samo jedan naziv naslova, a jednoj šifri oblasti samo jedan naziv oblasti, odnosno:

$$SIFN \rightarrow NAZIVN \qquad SIFO \rightarrow NAZIVO$$

Na osnovu prethodnog, nameće se da važi i zavisnost

$$SIFN,SIFO \rightarrow NAZIVN,NAZIVO$$

Primer

Za prethodnu šemu relacije NASLOV važi i to da jednoj šifri naslova odgovara samo jedna šifra oblasti kojoj opet odgovara samo jedan naziv oblasti, što možemo iskazati preko funkcijskih zavisnosti:

$$SIFN \rightarrow SIFO \qquad SIFO \rightarrow NAZIVO$$

Na osnovu toga, jednoj šifri naslova odgovara samo jedan naziv oblasti, osnosno:

$$SIFN \rightarrow NAZIVO$$

Mogli bi tako da nabrajamo niz primera i uočavamo niz novih situacija izvođenja funkcijskih zavisnosti, ali sistematičan pristup celoj toj problematici nameće nalaženje odgovora za sledeće pitanje: Postoji li neki konačni i minimalni skup pravila izvođenja novih funkcijskih zavisnosti iz nekog zadatog skupa F funkcijskih zavisnosti, a da pri tome zadovoljava sledeća dva uslova:

- pouzdanost: ne može se izvesti ni jedna zavisnost koja ne proizilazi iz F ;
- kompletnost: mogu se izvesti sve zavisnosti koje proizilaze iz F.

Dokazano je da takav skup postoji. Čine ga tri pravila koja se po autoru nazivaju Armstrongova pravila. U njihovom navođenju polazimo od toga da su X, Y, Z i W podskupovi skupa atributa R koji predstavlja šemu relacije.

1. Armstrongovo pravilo: Reflektivnost

$$Y \subseteq X \subseteq R \Rightarrow X \rightarrow Y$$

Svaki podskup atributa šeme relacije jednoznačno određuje svaki svoj sastavni deo: Pr tome postoje dva specijalna slučaja:

- za $Y=X$ sledi $X \rightarrow X$, odnosno svaki podskup atributa jednoznačno određuje sam sebe;
- za $Y=\varnothing$ sledi $X \rightarrow \varnothing$, odnosno svaki podskup atributa jednoznačno određuje prazan skup atributa.

2. Armstrongovo pravilo: Uvečanje

$$X \rightarrow Y \wedge Z \subseteq W \Rightarrow XW \rightarrow YZ$$

Kod ovog pravila postoje tri specijalna slučaja:

- za $Z=\varnothing$ sledi $XW \rightarrow Y$;
- za $Z=W$ sledi $XW \rightarrow YW$;
- za $Z=W=X$ sledi $X \rightarrow YX$.

3. Armstrongovo pravilo: Tranzitivnost

$$X \rightarrow Y \wedge Y \rightarrow Z \Rightarrow X \rightarrow Z$$

Specijalni slučajevi za ovo pravilo ne postoje.

Izvođenje funkcijskih zavisnosti samo na osnovu Armstrongovih pravila može biti složeno i nezgrapno, pa se za to koriste i dodatna tri pravila koja slede iz Armstrongovih pravila.

4. pravilo: Unija

$$X \rightarrow Y \wedge X \rightarrow Z \Rightarrow X \rightarrow YZ$$

5. pravilo: Dekompozicija

$$X \rightarrow Y \wedge Z \subseteq Y \Rightarrow X \rightarrow Z$$

6. pravilo: Pseudotranzitivnost

$$X \rightarrow Y \wedge WY \rightarrow Z \Rightarrow XW \rightarrow Z$$

Primenu pravila izvođenja funkcijskih zavisnosti ilustrovaćemo na pogodnom primeru.

Primer

Posmatrajmo šemu relacije iz ranijih primera

NASLOV (SIFN, NAZIVN, SIFO, NAZIVO)

kao i skup funkcijskih zavisnosti:

F = { SIFN→NAZIVN,SIFO,NAZIVO NAZIVN→SIFO SIFO→NAZIVO }

Iz datog skupa F možemo redom izvršiti sledeća izvođenja:

po pravilu reflektivnosti:

SIFN→NAZIVN SIFN→SIFO,NAZIVO

po pravilu uvečanja:

SIFO→SIFO,NAZIVO SIFO,NAZIVN→NAZIVO,NAZIVN

po pravilu tranzitivnosti:

NAZIVN→NAZIVO

7.3.3 Zatvarač skupa funkcijskih zavisnosti

U prethodnom primeru izveli smo svega nekoliko funkcijskih zavisnosti. U opštem slučaju, broj funkcijskih zavisnosti izveden iz nekog datog skupa funkcijskih zavisnosti F može biti vrlo veliki, pogotovo kada su u pitanju zavisnosti izvedene po pravilu reflektivnosti. Pitanje koje se pri tome nameće je sledeće:

- da li je broj izvedenih zavisnosti konačan, odnosno da li ikada nastupa situacija kada više ne možemo da izvedemo ni jednu novu funkcijsku zavisnost iz F?

Odgovor na prethodno pitanje je potvrdan. Konačnost ukupnog broja izvedenih funkcijskih zavisnosti proizilazi iz konačnosti broja atributa koji se javljaju u F i okolnosti da su najveće moguće leve i desne strane izvedenih zavisnosti jednake skupu svih tih atributa.

U praksi, postupak izvođenja funkcijskih zavisnosti iz datog skupa zavisnosti F tekao bi na sledeći način: Prvo bi formirali novi skup F^+ koji bi kao početni sadržaj imao sve zavisnosti iz F. Zatim bi izvodili jednu po jednu novu funkcijsku zavisnost i dodavali je u F^+ samo ako se već ne nalazi u njemu. U jednom trenutku, nastupila bi situacija kada ne možemo da izvedemo ni jednu novu funkcijsku zavisnost koja se već ne nalazi u F^+. Takav skup F^+ nazivamo zatvaračem skupa funkcijskih zavisnosti F.

Definicija

Zatvarač F^+ skupa funkcijskih zavisnosti F zadatog nad skupom atributa R je minimalni skup funkcijskih zavisnosti koji zadovoljava sledeća dva uslova:

- zadati skup F je podskup F^+, odnosno $F \subseteq F^+$;;
- primenom Armstrongovih pravila na funkcijske zavisnosti u F^+ ne može se izvesti ni jedna zavisnost koja se već ne nalazi u F^+, odnosno važi:

$$\neg\exists(X\rightarrow Y)\ (\ XY\subseteq R \wedge F^+\Rightarrow(X\rightarrow Y) \wedge (X\rightarrow Y) \notin F^+\ )$$

Primer

Posmatrajmo šemu relacije iz prethodnog primera

NASLOV (SIFN, NAZIVN, SIFO, NAZIVO)

i skup funkcijskih zavisnosti

F = { SIFN→NAZIVN,SIFO SIFO→NAZIVO NAZIVO→SIFO }

Primenom pravila izvođenja dobija se kao zatvarač F^+ (navedeno od gore na dole prvo sa leve strane a zatim sa desne):

SIFN→∅
SIFN→SIFN
SIFN→NAZIVN
SIFN→SIFO
SIFN→NAZIVO
SIFN→SIFN,NAZIVN
SIFN→SIFN,SIFO
SIFN→SIFN,NAZIVO
SIFN→NAZIVN,SIFO
SIFN→NAZIVN,NAZIVO
SIFN→SIFN,NAZIVN,SIFO
SIFN→SIFN,NAZIVN,NAZIVO
SIFN→SIFN,SIFO,NAZIVO
SIFN→NAZIVN,SIFO,NAZIVO
SIFN→SIFN,NAZIVN,SIFO,NAZIVO
NAZIVN→∅
NAZIVN→NAZIVN
SIFO→∅
SIFO→SIFO
SIFO→NAZIVO
SIFO→SIFO,NAZIVO
NAZIVO→∅
NAZIVO→SIFO
NAZIVO→NAZIVO
NAZIVO→SIFO,NAZIVO
SIFN,NAZIVN→∅
SIFN,NAZIVN→SIFN
SIFN,NAZIVN→NAZIVN
SIFN,NAZIVN→SIFO
SIFN,NAZIVN→NAZIVO
SIFN,NAZIVN→SIFN,NAZIVN
SIFN,NAZIVN→SIFN,SIFO
SIFN,NAZIVN→SIFN,NAZIVO
SIFN,NAZIVN→NAZIVN,SIFO
SIFN,NAZIVN→NAZIVN,NAZIVO
SIFN,NAZIVN→SIFO,NAZIVO
SIFN,NAZIVN→SIFN,NAZIVN,SIFO
SIFN,NAZIVN→SIFN,NAZIVN,NAZIVO
SIFN,NAZIVN→SIFN,SIFO,NAZIVO
SIFN,NAZIVN→NAZIVN,SIFO, NAZIVO
SIFN,NAZIVN→SIFN,NAZIVN,SIFO,NAZIVO
. . .

. . .
SIFN,SIFO→∅
SIFN,SIFO→SIFN
SIFN,SIFO→NAZIVN
SIFN,SIFO→SIFO
SIFN,SIFO→NAZIVO
SIFN,SIFO→SIFN,NAZIVN
SIFN,SIFO→SIFN,SIFO
SIFN,SIFO→SIFN,NAZIVO
SIFN,SIFO→NAZIVN,SIFO
SIFN,SIFO→NAZIVN,NAZIVO
SIFN,SIFO→SIFO,NAZIVO
SIFN,SIFO→SIFN,NAZIVN,SIFO
SIFN,SIFO→SIFN,NAZIVN,NAZIVO
SIFN,SIFO→SIFN,SIFO,NAZIVO
SIFN,SIFO→SIFN,NAZIVN,SIFO,NAZIVO
SIFN,NAZIVO→∅
SIFN,NAZIVO→SIFN
SIFN,NAZIVO→NAZIVN
SIFN,NAZIVO→SIFO
SIFN,NAZIVO→NAZIVO
SIFN,NAZIVO→SIFN,NAZIVN
SIFN,NAZIVO→SIFN,SIFO
SIFN,NAZIVO→SIFN,NAZIVO
SIFN,NAZIVO→NAZIVN,SIFO
SIFN,NAZIVO→NAZIVN,NAZIVO
SIFN,NAZIVO→SIFO,NAZIVO
SIFN,NAZIVO→SIFN,NAZIVN,SIFO
SIFN,NAZIVO→SIFN,NAZIVN,NAZIVO
SIFN,NAZIVO→SIFN,SIFO,NAZIVO
SIFN,NAZIVO→SIFN,NAZIVN,SIFO,NAZIVO
NAZIVN,SIFO→∅
NAZIVN,SIFO→NAZIVN
NAZIVN,SIFO→SIFO
NAZIVN,SIFO→NAZIVO
NAZIVN,SIFO→NAZIVN,SIFO
NAZIVN,SIFO→NAZIVN,NAZIVO
NAZIVN,SIFO→SIFO,NAZIVO
NAZIVN,SIFO→NAZIVN,SIFO,NAZIVO
NAZIVN,NAZIVO→∅
NAZIVN,NAZIVO→NAZIVN
NAZIVN,NAZIVO→SIFO
NAZIVN,NAZIVO→NAZIVO
NAZIVN,NAZIVO→NAZIVN,SIFO
NAZIVN,NAZIVO→NAZIVN,NAZIVO
NAZIVN,NAZIVO→SIFO,NAZIVO
NAZIVN,NAZIVO→NAZIVN,SIFO,NAZIVO
SIFO,NAZIVO→∅
SIFO,NAZIVO→SIFO
SIFO,NAZIVO→NAZIVO
SIFO,NAZIVO→SIFO,NAZIVO
SIFN,NAZIVN,SIFO→∅
SIFN,NAZIVN,SIFO→SIFN
SIFN,NAZIVN,SIFO→NAZIVN
SIFN,NAZIVN,SIFO→SIFO
SIFN,NAZIVN,SIFO→NAZIVO
SIFN,NAZIVN,SIFO→SIFN,NAZIVN
SIFN,NAZIVN,SIFO→SIFN,SIFO
SIFN,NAZIVN,SIFO→SIFN,NAZIVO
SIFN,NAZIVN,SIFO→NAZIVN,SIFO
SIFN,NAZIVN,SIFO→NAZIVN,NAZIVO
SIFN,NAZIVN,SIFO→SIFO,NAZIVO
SIFN,NAZIVN,SIFO→SIFN,NAZIVN,SIFO
SIFN,NAZIVN,SIFO→SIFN,NAZIVN,NAZIVO
SIFN,NAZIVN,SIFO→SIFN,SIFO,NAZIVO
SIFN,NAZIVN,SIFO→SIFN,NAZIVN,SIFO,NAZIVO
SIFN,NAZIVN,NAZIVO→∅
SIFN,NAZIVN,NAZIVO→SIFN
SIFN,NAZIVN,NAZIVO→NAZIVN
SIFN,NAZIVN,NAZIVO→SIFO
SIFN,NAZIVN,NAZIVO→NAZIVO
SIFN,NAZIVN,NAZIVO→SIFN,NAZIVN
SIFN,NAZIVN,NAZIVO→SIFN,SIFO
SIFN,NAZIVN,NAZIVO→NAZIVN,SIFO
SIFN,NAZIVN,NAZIVO→NAZIVN,NAZIVO
SIFN,NAZIVN,NAZIVO→SIFO,NAZIVO
SIFN,NAZIVN,NAZIVO→SIFN,NAZIVN,SIFO
SIFN,NAZIVN,NAZIVO→SIFN,NAZIVN,NAZIVO
SIFN,NAZIVN,NAZIVO→SIFN,SIFO,NAZIVO
SIFN,NAZIVN,NAZIVO→NAZIVN,SIFO,NAZIVO
SIFN,NAZIVN,NAZIVO→SIFN,NAZIVN,SIFO,NAZIVO
SIFN,SIFO,NAZIVO→∅
SIFN,SIFO,NAZIVO→SIFN
SIFN,SIFO,NAZIVO→NAZIVN
SIFN,SIFO,NAZIVO→SIFO
SIFN,SIFO,NAZIVO→NAZIVO
SIFN,SIFO,NAZIVO→SIFN,NAZIVN
SIFN,SIFO,NAZIVO→SIFN,SIFO
SIFN,SIFO,NAZIVO→SIFN,NAZIVO
SIFN,SIFO,NAZIVO→NAZIVN,SIFO
SIFN,SIFO,NAZIVO→NAZIVN,NAZIVO
SIFN,SIFO,NAZIVO→SIFO,NAZIVO
SIFN,SIFO,NAZIVO→SIFN,NAZIVN,SIFO
SIFN,SIFO,NAZIVO→SIFN,NAZIVN,NAZIVO
. . .

. . .
SIFN,SIFO,NAZIVO→SIFN,SIFO,NAZIVO
SIFN,SIFO,NAZIVO→NAZIVN,SIFO,NAZIVO
SIFN,SIFO,NAZIVO→SIFN,NAZIVN,SIFO,NAZIVO
NAZIVN,SIFO,NAZIVO→∅
NAZIVN,SIFO,NAZIVO→NAZIVN
NAZIVN,SIFO,NAZIVO→SIFO
NAZIVN,SIFO,NAZIVO→NAZIVO
NAZIVN,SIFO,NAZIVO→NAZIVN,SIFO
NAZIVN,SIFO,NAZIVO→NAZIVN,NAZIVO
NAZIVN,SIFO,NAZIVO→SIFO,NAZIVO
NAZIVN,SIFO,NAZIVO→NAZIVN,SIFO,NAZIVO
SIFN,NAZIVN,SIFO,NAZIVO→∅
SIFN,NAZIVN,SIFO,NAZIVO→SIFN
SIFN,NAZIVN,SIFO,NAZIVO→NAZIVN
SIFN,NAZIVN,SIFO,NAZIVO→SIFO
SIFN,NAZIVN,SIFO,NAZIVO→NAZIVO
SIFN,NAZIVN,SIFO,NAZIVO→SIFN,NAZIVN
SIFN,NAZIVN,SIFO,NAZIVO→SIFN,SIFO
SIFN,NAZIVN,SIFO,NAZIVO→SIFN,NAZIVO
SIFN,NAZIVN,SIFO,NAZIVO→NAZIVN,SIFO
SIFN,NAZIVN,SIFO,NAZIVO→NAZIVN,NAZIVO
SIFN,NAZIVN,SIFO,NAZIVO→SIFO,NAZIVO
SIFN,NAZIVN,SIFO,NAZIVO→SIFN,NAZIVN,SIFO
SIFN,NAZIVN,SIFO,NAZIVO→SIFN,NAZIVN,NAZIVO
SIFN,NAZIVN,SIFO,NAZIVO→SIFN,SIFO,NAZIVO
SIFN,NAZIVN,SIFO,NAZIVO→NAZIVN,SIFO,NAZIVO
SIFN,NAZIVN,SIFO,NAZIVO→SIFN,NAZIVN,SIFO,NAZIVO

Dobili smo da je jedini kandidat-ključ date šeme relacije SIFN.

Rezultat je iznenađujuće obiman – za slučaj 3 zavisnosti nad šemom relacije sa 4 atributa dobili smo da zatvarač F^+ sadrži 159 zavisnosti. Ako bi iz svake grupe zavisnosti $X \rightarrow Y$ sa istom desnom stranom X uzeli samo onu sa maksimalnim Y (sve druge iz grupe su izvodive iz nje po Armstrongovim pravilima) dobili bi netrivijalni zatvarač F^+_n daleko umerenijeg obima, odnosno sa svega 15 zavisnosti:

SIFN→SIFN,NAZIVN,SIFO,NAZIVO
NAZIVN→NAZIVN
SIFO→SIFO,NAZIVO
NAZIVO→SIFO,NAZIVO
SIFN,NAZIVN→SIFN,NAZIVN,SIFO,NAZIVO
SIFN,SIFO→SIFN,NAZIVN,SIFO,NAZIVO
SIFN,NAZIVO→SIFN,NAZIVN,SIFO,NAZIVO
NAZIVN,SIFO→NAZIVN,SIFO,NAZIVO
NAZIVN,NAZIVO→NAZIVN,SIFO,NAZIVO
SIFO,NAZIVO→SIFO,NAZIVO
SIFN,NAZIVN,SIFO→SIFN,NAZIVN,SIFO,NAZIVO
SIFN,NAZIVN,NAZIVO→SIFN,NAZIVN,SIFO,NAZIVO
SIFN,SIFO,NAZIVO→SIFN,NAZIVN,SIFO,NAZIVO
NAZIVN,SIFO,NAZIVO→NAZIVN,SIFO,NAZIVO
SIFN,NAZIVN,SIFO,NAZIVO→SIFN,NAZIVN,SIFO,NAZIVO

7.3.4 Zatvarač skupa atributa i njegova primena

Prethodni primer je dobra ilustracija za kompleksnost zatvarača skupa funkcijskih zavisnosti. Možemo samo da zamislimo šta bi bilo u slučaju skupa od recimo 5 atributa i 4 zavisnosti u F.

Primer koji smo uradili otkriva i to da je pristup traženju izvedenih funkcijskih zavisnosti kombinatornog karaktera. Naime, mi formiramo redom kombinacije od jednog, dva, tri i četri atributa sa leve strane i onda za takvu levu stranu izvodimo sve moguće zavisnosti sa različitim desnim stranama. Sama primena pravila izvođenja pri tom postupku je vrlo obiman i zamoran posao, a najgore je što odsustvo neke zavisnosti može da znači ne samo da ona stvarno ne postoji u F^+, nego i to da ona postoji a da nismo uspeli da je izvedemo. Usled toga pravila izvođenja imaju uglavnom teoretski značaj, u smislu da služe za dokazivanje valjanosti raznih algoritama u teoriji zavisnosti i nodmalnih formi.

Nalaženje F^+ bi bilo znatno olakšano kada bi nekim algoritmom za zadati skup atributa X sa leve strane funkcijske zavisnosti uspeli da odredimo skup Y svih atributa sa desne strane, odnosno sve atribute koje jednoznačno određuje X.

Takav algoritam postoji i zasnovan je na pravilima izvođenja funkcijskih zavisnosti.

Definicija

Neka je R skup atributa, X neki njegov podskup, a F skup funkcijskih zavisnosti nad R. Zatvarač X^+ skupa atributa X čini skup atributa Y koji odgovara desnoj strani zavisnosti $X \rightarrow Y$ u F^+ sa maksimalnim Y (za sve ostale zavisnosti u F^+ oblika $X \rightarrow Z$ važi $Z \subset Y$).

Primer

Uvidom u funkcijske zavisnosti u F^+ iz prethodnog primera dobijamo:

$SIFN^+ = SIFN,NAZIVN,SIFO,NAZIVO$ $(NAZIVO,SIFO)^+ = NAZIV,SIFO$

Uočavamo da zatvarač nekog skupa atributa uvek sadrži i taj skup atributa, što je posledica reflektivnosti.

U daljem tekstu u ovom poglavlju izložićemo niz algoritama koristeći kao notaciju pseudo-jezik opisan u prilogu C.

Sledeći algoritam na osnovu zadatog skupa atributa X i skupa funkcijskih zavisnosti F nalazi skup atributa Y koji predstavlja zatvarač X^+.

```
ZatvaracX ( > X, > F )
:  Inicijalizacija
:  :  XzatStaro = ∅
:  :  Xzat      = X
:  Ponavljanje do prolaza bez izmene zatvaraca
:  :  DO WHILE ( Xzat ≠ XzatStaro )
:  :  |  Pamcenje prethodnog zatvaraca
:  :  |  :  XzatStaro = Xzat
:  :  |  Provera za sve zavisnosti
:  :  |  :  DO FOR EACH ( Z→W IN F )
:  :  |  :  |  Provera da li je Z sadrzano u zatvaracu
:  :  |  :  |  :  IF ( Z ⊆ Xzat )
:  :  |  :  |  :  |  Uvecanje zatvaraca sa W
:  :  |  :  |  :  |  :  Xzat := Xzat ∪ W
:  RETURN Xzat
```

Dokaz ispravnosti ovog algoritma zasniva se na Armstrongovim aksiomima.

Primer

Posmatrajmo šemu relacije

NASLOV (SIFN, NAZIVN, SIFO, NAZIVO)

i skup funkcijskih zavisnosti

F = { SIFN→SIFN,NAZIVN,SIFO,NAZIVO SIFO→NAZIVO NAZIVO→SIFO }

Na osnovu prethodnog algoritma dobijamo, pri čemu smo međurezultate razdvojili sumbolom | a sa √ označili konačni rezultat:

$SIFN^+ = SIFN \mid SIFN,NAZIVN,SIFO,NAZIVO \surd$

$SIFO^+ = SIFO \mid SIFO,NAZIVO \surd$

$NAZIVN^+ = NAZIVN \surd$

$(SIFO,NAZIVN)^+ = SIFO,NAZIVN \mid SIFO,NAZIVN,NAZIVO \surd$

Dobili smo da je SIFN kandidat-ključ, pošto je njegov zatvarač cela šema relacije.

Algoritam *ZatvaracX* je sigurno najznačajniji algoritam u oblasti funkcijske zavisnosti podataka. Na njemu se zasniva većina drugih algoritama iz te oblasti, od kojih su neki navedeni u ovom poglavlju a neki u prilogu E. Pri tome ćemo smatrati da su nam na raspolaganju dve pomoćne fukcije:

- *SkupPodskupova (> X, > n)* : funkcija koja kao rezultat daje skup čiji su elementi svi podskupovi formirani od n elemenata skupa X (u praksi, to će biti podskupovi od 1, 2 i više atributa šeme relacije R);
- *BrojElemenata (> X)* : funkcija koja kao rezultat daje broj elemenata nekog skupa (u praksi, to će najčešće biti broj atributa u šemi relacije R).

Prva primena algoritma *ZatvaracX* je u određivanju da li se neka funkcijska zavisnost $X \rightarrow Y$ nalazi u zatvaraču F^+ nekog skupa funkcijskih zavisnosti F. Ako dobijemo da važi $Y \subseteq X^+$, iz toga sledi $(X \rightarrow Y) \in F^+$, na osnovu Armstrongovih pravila reflektivnosti i tranzitivnosti. Sledeći algoritam na osnovu zadate funkcijske zavisnosti $X \rightarrow Y$ i skupa funkcijskih zavisnosti F utvrđuje da li se ta zavisnosti nalazi u F^+ ili ne:

```
ElementPlus ( > X→Y, > F )
:  Inicijalizacija
:  :  YuF = FALSE
:  Provera da li je Y sadrzano u zatvaracu
:  :  IF ( Y ⊆ ZatvaracX ( X, F )
:  :  |  YuF = TRUE
:  RETURN YuF
```

Drugu primenu algoritma *ZatvaracX* predstavlja postupak za izračunavanja zatvarača F^+ skupa funkcijskih zavisnosti F zadatih nad šemom relacije $R=\{A_1,..,A_N\}$. Suština tog postupka koji je kombinatorne prirode je u sledećem:

- zatvarač F^+ je u početku prazan skup;
- redom se formiraju skupovi S svih podskupova WuS, sastavljenih prvo od jednog, zatim od dva i tako sve do svih atributa šeme relacije R;

Za svaki takav podskup WuS u svakom S uradi se sledeće:

- odredi se zatvarač Y , odnosno podskup Y svih atributa šeme relacije R koji funkcijski zavise od WuS;
- u F^+ se doda zavisnost $WuS \rightarrow \varnothing$;
- redom se formiraju skupovi T svih podskupova ZuT, sastavljenih prvo od jednog, zatim od dva i tako sve do svih atributa podskupa Y;
- za svaki takav podskup ZuT u F^+ se doda zavisnost $WuS \rightarrow ZuT$; poslednje takvo dodavanje obuhvatiče zavisnost $WuS \rightarrow Y$.

Algoritam *ZatvaracF* glasi:

```
ZatvaracF  ( > R, > F  )
:  Inicijalizacija
:  :  Fplus = ∅
:  :  NatrR = BrojElemenata ( R )
:  Ponavljanje onoliko puta koliko ima atributa u R
:  :  DO FOR ( i = 1 to NatrR )
:  :  |  Formiranje skupa podskupova od i atrubuta iz R
:  :  |  :  S = SkupPodskupova ( R, i )
:  :  |  Nalazenje svih zavisnosti od svih podskupova
:  :  |  :  DO FOR EACH ( WuS IN S )
:  :  |  :  |  Nalazenje zatvaraca jednog podskupa
:  :  |  :  |  :  Y = ZatvaracX ( WuS, F )
:  :  |  :  |  Dodavanje zavisnosti WuS→∅ u Fplus
:  :  |  :  |  :  Fplus  = Fplus ∪ {WuS→∅}
:  :  |  :  |  Dodavanje svih ostalih zavisnosti u Fplus
:  :  |  :  |  :  NatrY = BrojElemenata ( Y )
:  :  |  :  |  :  DO FOR ( j = 1 to NatrY )
:  :  |  :  |  :  |  Formiranje skupa podskupova od j atributa Y
:  :  |  :  |  :  |  :  T = SkupPodskupova ( Y, j )
:  :  |  :  |  :  |  Dodavanje jedne zavisnosti u Fplus
:  :  |  :  |  :  |  :  DO FOR EACH ( ZuT IN T )
:  :  |  :  |  :  |  :  |  Fplus = Fplus ∪ {(WuS→ZuT)}
:  RETURN Fplus
```

Treću i izuzetno značajnu primenu algoritma *ZatvaracX* predstavlja nalaženje skupova atributa koji su kandidat-ključevi, odnosno određivanje skupa kandidat-ključeva K. Suština algoritma je u sledećem:

- skup K je u početku prazan;
- redom se formiraju skupovi S svih podskupova XuS, sastavljenih prvo od jednog, zatim od dva i tako sve do svih atributa šeme relacije R;

Za svaki takav podskup XuS koji nije već sadržan u ili jednak nekom elementu skupa K uradi se sledeće:

- odredi se zatvarač ;
- ako je zatvarač XuS cela šema relacije R , odnosno ako važi Y=R, XuS se dodaje skupu kandidat-ključeva K.

Odgovarajući algoritam glasi:

```
KandidatKljucevi ( > R, > F )
:  Inicijalizacija
:  :  KandK = ∅
:  :  NatrR = BrojElemenata ( R )
:  Ponavljanje onoliko puta koliko ima atributa u R
:  :  DO FOR ( i = 1 to NatrR )
:  :  |  Formiranje skupa podskupova od i atrubuta iz R
:  :  |  :  S = SkupPodskupova ( R, i )
:  :  |  Ponavljanje za svaki podskup atributa u S
:  :  |  :  DO FOR EACH ( XuS IN S )
:  :  |  :  |  Provera da je XuS vec u KandK ili je superkljuc
:  :  |  :  |  :  Sadrzan = FALSE
:  :  |  :  |  :  DO FOR EACH ( Kljuc IN KandK )
:  :  |  :  |  :  |  IF ( Kljuc ⊆ XuS )
:  :  |  :  |  :  |  |  Sadrzan = TRUE
:  :  |  :  |  <:--|--|--EXIT
:  :  |  :  |  Provera da li je XuS novi kandidat-kljuc
:  :  |  :  |  :  IF ( NOT Sadrzan )
:  :  |  :  |  :  |  Nalazenje zatvaraca XuS
:  :  |  :  |  :  |  :  ZatvXuS = ZatvaracX ( XuS, F )
:  :  |  :  |  :  |  :  Dodavanje u KandK ako je kandidat-kljuc
:  :  |  :  |  :  |  :  :  IF ( ZatvXuS == R )
:  :  |  :  |  :  |  :  :  |  KandK = KandK ∪ XuS
:  RETURN KandK
```

Poslednja primena algoritna *Zatvarac* koju ovde navodimo je algoritam *SuperKljučevi (R, F)* koji kao rezultat daje skup svih super-ključeva S šeme relacije R na osnovu skupa funkcijskih zavisnosti F. Taj algoritam se razlikuje od prethodnog samo po tome što bezuslovno dodaje u S svako XuS čiji je zatvarač cela šema relacije R.

Odgovarajući algoritam glasi:

```
SuperKljucevi ( > R, > F )
:  Inicijalizacija
:  :  SuperK = ∅
:  :  NatrR  = BrojElemenata ( R )
:  Ponavljanje onoliko puta koliko ima atributa u R
:  :  DO FOR ( i = 1 to NatrR )
:  :  |  Formiranje skupa podskupova od i atrubuta iz R
:  :  |  :  S = SkupPodskupova ( R, i )
:  :  |  Ponavljanje za svaki podskup atributa u S
:  :  |  :  DO FOR EACH ( XuS IN S )
:  :  |  :  |  Nalazenje zatvaraca XuS
:  :  |  :  |  :  ZatvXuS = ZatvaracX ( XuS, F )
:  :  |  :  |  Dodavanje u KandK ako je super-kljuc
:  :  |  :  |  :  IF ( ZatvXuS == R )
:  :  |  :  |  :  |  SuperK = SuperK ∪ XuS
:  RETURN SuperK
```

Primer

Posmatrajmo šemu relacije:

OBLAST (SIFO, NAZIV)

i skup funkcijskih zavisnosti, zasnovan na okolnosti da su i nazivi oblasti unikatni:

F = { SIFO→NAZIV NAZIV→SIFO }

Algoritam *KandidatKljucevi* daje kao rezultat samo:

SIFO NAZIV

Primena algoritma *SuperKljucevi* daje sledeće podskupove atributa:

SIFO NAZIV SIFO,NAZIV

7.3.5 Dekompozicija skupa funkcijskih zavisnosti

Očuvanje atributa i podataka su bitni uslovi za valjanost dekompozicije neke polazne šeme relacije R u dve ili više novih šema R_i. Ako je nad polaznom šemom važio neki skup funkcijskih F, postavlja se pitanje kakvi će biti skupovi zavisnosti F_i nad šemama R_i posle dekompozije.

Šta praktično znače funkcijske zavisnosti za neku relacionu bazu podataka? One su posledice ograničenja koja važe između nekih svojstava u sistemu koga ta baza podataka predstavlja, i ta ograničenja moraju uvek biti ispoštovana pri ažuriranju baze podataka. Ako važi $X \rightarrow Y$, onda u našoj bazi podataka svakoj vrednosti X, ma koliko puta se pojavila skupa sa Y, mora da odgovara samo jedna vrednosti Y. Pri tome su moguća dva slučaja u vezi načina zapisivanja X i Y u bazi podataka:

- X i Y se nalaze u istoj šemi relacije, odnosno njihove vrednosti su uvek u istoj relaciji ;
- X i Y se nalaze razdvojeni u dve šeme relacija, odnosno njihove vrednosti su u dve relacije, ali se mogu pojaviti u jednoj relaciji izvršavanjem upita koji spaja te dve relacije;

Razmotrimo prvo slučaj kada su X i Y u istoj šemi relacije. Neka nam kao ilustracija posluži raniji primer relacije **naslov**, odnosno torke označene sa (a) i (b), pri čemu je prvo uneta torka (a) a posle nje i torka (b):

```
naslov ( SIFN NAZIVN                  SIFO  NAZIVO           )
       ------------------------------------------------------
       RBP0 Relacione baze podataka BP    Baze podataka
       RK00 Racunarske komunikacije RM    Racunarske mreze
  (a)  PP00 PASCAL programiranje    PJ    Programski jezici
  (b)  PJC0 Programski jezik C      PJ    Programski jezici
       ------------------------------------------------------
```

U ovoj relaciji važi, pored ostalih, i funkcijska zavisnost SIFO→NAZIVO.

Pri unosu torke (a) jedino je važilo ograničenje da SIFN mora biti unikatno i ne-NULL. I pored zavisnosti SIFO→NAZIVO, uz vrednost 'PJ' za SIFO mogli smo da unesemo potpuno proizvoljnu vrednost za NAZIVO, i to zato što se vrednost 'PJ' pojavljuje prvi put. Međutim, pri unosu torke (b) kada se ponovo unosi vrednost 'PJ' za SIFO važi dodatno ograničenje: pošto se u relaciji **naslov** već nalazi vrednost 'PJ', u torci (b) za NAZIVO *moramo* uneti vrednost 'Programski jezici', pošto bi u suprotnom narušili zavisnost SIFO→NAZIVO. Proveru da li je to zadovoljeno možemo implementirati jednostavno - u definiciji tabele **naslov** treba da navedemo ograničenje tabele **`UNIQUE(SIFO,NAZIVO)`**.

Za ilustraciju slučaja kada su X i Y razdvojeni u dve relacije poslužiće nam dekompozicija prethodne relacije **naslov** na relacije **n1** i **n2**, pri čemu su naznačene torke koje odgovaraju torkama (a) i (b) u relaciji **naslov**:

```
n1 ( SIFN NAZIVN                   NAZIVO          ) n2( SIFN SIFO )
     ------------------------------------------------     ---------
     RBP0 Relacione baze podataka Baze podataka          RBP0 BP
     RK00 Racunarske komunikacije Racunarske mreze       RK00 RM
 (a) PP00 PASCAL programiranje    Programski jezici (a) PP00 PJ
 (b) PJC0 Programski jezik C      Programski jezici (b) PJC0 PJ
     ------------------------------------------------     ---------
```

Ova dekompozicija je bez gubitaka podataka, pošto je **n1**∩**n2**→**n1**. Posmatrajmo istu funkcijsku zavisnost kao i u prvom slučaju, SIFO→NAZIVO.

Pri unosu para torki (a) u relacije **n1** i **n2** jedino je važilo ograničenje da vrednost SIFN mora biti unikatna i različita od NULL , dok smo za atribute NAZIVO i SIFO mogli da unesemo bilo koji par vrednosti, s obzirom da se vrednost 'PJ' za SIFO pojavljuje prvi put. Međutim, pri unosu para torki (b) pojavljuje se vrednost 'PJ" za SIFO koja već postoji, pa za atribut NAZIVO *moramo* uneti vrednost 'Programski jezici', pošto bi svaka druga vrednost narušila funkcijsku zavisnost SIFO→NAZIVO. Na primer, ako bi na NAZIVO uneli 'Baze podataka', upit koji spajanjem relacija **n1** i **n2** daje šifre i nazive oblasti dao bi rezultat koji narušava zavisnost SIFO→NAZIVO:

```
SELECT N2.SIFO,N1.NAZIVO            BP   Baze podataka
 FROM  N2,N1                        RM   Racunarske mreze
 WHERE N2.SIFN=N1.SIFN ;            PJ   Programski jezici  ?
                                    PJ   Baze podataka      ?
```

Pošto se sada SIFO i NAZIVO nalaze razdvojeni u dve relacije, jedini način za proveru važenja zavisnosti SIFO→NAZIVO je preko pravila opšteg integriteta

```
CREATE ASSERTION Zavisnost
       CHECK ( NOT EXISTS ( SELECT    N2.SIFO
                            FROM      N2,N1
                            WHERE     N2.SIFN=N1.SIFN
                            GROUP BY  N2.SIFO,
                            HAVING    COUNT(DISTINCT N1.NAZIVO)>1 ))
```

ili rečima: "u spoju **n2** i **n1** ni jedno SIFN ne sme da se javlja sa više od jednim SIFO".

Dobar sistem za upravljanje relacionom bazom podataka ne sme da dozvoli situaciju narušavanja funkcijske zavisnosti kakva je prethodno opisana. Zbog toga se prilikom kreiranja baze podataka, odnosno opisa njene strukture, zadaju i sva dodatna ograničenja zasnovana na funkcijskim zavisnostima, a prilikom svake izmene sadržaja baze podataka sistem automatski proverava saglasnost i sa tim dodatnim ograničenjima. Međutim, tu su moguća dva slučaja, što smo već naveli:

- za zadatu funkcijsku zavisnost u bazi podataka postoji bar jedna relacija koja sadrži sve njene atribute;
- za zadatu funkcijsku zavisnost u bazi podataka ne postoji ni jedna relacija koja sadrži sve njene atribute.

Prvi slučaj obezbeđuje efikasnu proveru, pošto se pristupa samo jednoj relaciji. U drugom slučaju može se desiti (ne mora uvek) da je provera moguća samo ako se izvrši spajanje dve ili više relacija u okviru pravila opšteg integriteta, što je daleko sporiji postupak provere.

Šta se ustvari dogodilo u našem prethodnom primeru? Polaznu relaciju **naslov** dekomponovali smo na dve relacije, **n1** i **n2**, što možemo prikazati šematski, uz navođenje skupova funkcijskih zavisnosti za relacije, kao:

NASLOV (SIFN, NAZIVN, SIFO, NAZIVO)	SIFN→NAZIVN,SIFO,NAZIVO SIFO→NAZIVO
→ N1 (SIFN, NAZIVN, NAZIVO)	SIFN→NAZIVN,NAZIVO
→ NV2 (SIFN, SIFO)	SIFN→SIFO

Skupovi zavisnosti koji važe nad relacijama nastalim dekompozicijom neke polazne relacije nazivaju se projekcijama polaznog skupa zavisnosti. U našem primeru, imamo valjanu dekompoziciju koja zadovoljava uslove očuvanja podataka, ali je došlo do tzv. "gubitka zavisnosti", pošto zavisnost SIFO→NAZIVO nije primenjiva ni na jednu od nastalih relacija i može se proveravati samo spajanjem tih relacija. Sa druge strane, dekompozicija

NASLOV (SIFN, NAZIVN, SIFO, NAZIVO)	SIFN→NAZIVN,SIFO,NAZIVO SIFO→NAZIVO
→ N1 (SIFN, NAZIVN, SIFO)	SIFN→NAZIVN,SIFO
→ N2 (SIFO, NAZIVO)	SIFO→NAZIVO

je bolja, pošto ne dovodi do gubitka zavisnosti SIFO→NAZIVO.

Definicija

Neka je F skup funkcijskih zavisnosti $\{X_i \rightarrow Y_i\}$ nad šemom relacije R i neka je S neki podskup atributa u R. Projekcija F po S, odnosno $F[S]$, je skup funkcijskih zavisnosti F' koji čine sve funkcijske zavisnosti oblika $X_i \rightarrow Z_i$ za koje je zadovoljen uslov $X_i \subseteq S \wedge Z_i = Y_i \cap S \wedge Z_i \neq \varnothing$.

Definicija

Neka je F skup funkcijskih zavisnosti nad šemom relacije R i neka je data dekompozicija R na skup šema relacija $\{R_1,...,R_N.\}$ Projekcija F po dekompoziciji $\{R_1,...,R_N\}$, odnosno $F[\{R_1,...,R_N.\}]$, je skup funkcijskih zavisnosti F_D za koji važi

$$F_D = F[R_1] \cup ... F[R_N]$$

Navedimo prvo algoritam koji za dati skup funkcijskih zavisnosti F i podskup atributa S nalazi skup F' kao projekciju F po S :

```
ProjekcijaFpoS  ( > F, > S )
:  Inicijalizacija
:  :  FpoS = ∅
:  Projekcija svih zavisnosti
:  :  DO FOR EACH ( X→Y in F )
:  :  |  Projekcija jedne zavisnosti
:  :  |  :  IF ( X ⊆ S )
:  :  |  :  |  Z := Y ∩ S
:  :  |  :  |  IF ( Z ≠ ∅ )
:  :  |  :  |  |  FpoS = FpoS ∪ { X→Z }
:  RETURN FpoS
```

Na osnovu tog algoritma može se sastaviti algoritam koji za dati skup funkcisjkih zavisnosti F i dekompoziciju D datu kao skup skupova atributa odnosno šema relacija $\{R_1,...,R_N.\}$ nalazi F_D kao projekciju po toj dekompoziciji:

```
ProjekcijaFpoD ( > F, > D )
:  Inicijalizacija
:  :  FpoD = ∅
:  Projekcija po svim semama relacija
:  :  DO FOR EACH ( R in D )
:  :  |  Projekcija po jednoj semi relacije
:  :  |  :  FpoD = FpoD ∪ ProjekcijaFpoS ( F,R )
:  RETURN FpoD
```

Funkcije *PrijekcijaFpoS* i *ProjekcijaFpoD* mogu se primeniti i za nalaženje odgovarajućih projekcija zatvarača F^+ odnosno netrivijalnog zatvarača F^+_n .

Pre nego što se ozbiljnije posvetimo problemu očuvanja funkcijskih zavisnosti pri dekompoziciji šeme relacije razmotrimo dva karakteristična primera. U oba slučaja poslužiće nam šema relacije

NASLOV (SIFN, NAZIVN, SIFO, NAZIVO)

i skup funkcijskih zavisnosti

F = { SIFN→NAZIVN,SIFO,NAZIVO SIFO→NAZIVO NAZIVO→SIFO }

Primer

Ako izvršimo dekompoziciju šeme relacije NASLOV na šeme relacija

NASLOV1 (SIFN, NAZIVN, SIFO) NASLOV2 (SIFO, NAZIVO)

dobijamo po prethodnim algoritmima dekompoziciju F na dva skupa, F1 i F2 :

F1 = { SIFN→NAZIVN,SIFO } F2 = { SIFO→NAZIVO NAZIVO→SIFO }

Imamo da je F1∪F2=F, odnosno ni jedna zavisnost iz F nije izgubljena.

Primer

Ako šemu relacije NASLOV dekomponujemo na drugi način

NASLOV1 (SIFN, NAZIVN,NAZIVO) NASLOV2 (SIFN, SIFO)

dobijamo drugačija dva skupa, F1 i F2 :

F1 = { SIFN→NAZIVN,NAZIVO } F2 = { SIFN→SIFO }

Sada imamo da je F1∪F2≠F i da su izgubljene dve zavisnosti: SIFO→NAZIVO i NAZIVO→SIFO , pošto ni na koji način ne mogu da se izvedu iz F1∪F2. Da li su stvarno izgubljene ?

Pri davanju odgovora na to pitanje treba imati na umu da smo dekomponovali samo zavisnosti koje su bile eksplicitno date u F, a ne i one koje se kao izvodive iz njih nalaze u F^+ odnosno u F^+_n. Da smo uključili i te zavisnosti, moglo bi se desiti da neka od izgubljenih zavisnosti bude izvodiva iz njih i onih eksplicitno datih. Iz ovoga je jasno da možemo da govorimo o dve vrste gubitka zavisnosti:

- prividni gubitak: zavisnost iz F se eksplicitno ne nalazi u projekciji F po dekompoziciji niti je izvodiva iz nje, ali se nalazi ili je izvodiva iz projekcije F^+ (odnosno F^+_n)po dekompoziciji (u stvari, zavisnost je indirektno sačuvana);
- suštinski gubitak: zavisnost iz F se eksplictno ne nalazi u projekciji F po dekompoziciji niti je izvodiva iz nje, niti se nalazi i nije izvodiva iz projekcije F^+ (odnosno F^+_n) po dekompoziciji.

Drugačije rečeno: za svaku zavisnost "izgubljenu" pri dekompoziciji moramo proveriti da li prividno ili suštinski izgubljena. Ta provera se može sprovesti na dva načina:

- analitički: za polazni skup zavisnosti određuje se F^+ (ili F^+_n), projektuje se po dekompoziciji i proverava se da li se izgubljena zavisnost nalazi u dekompoziciji zavisnosti ili je izvodiva iz nje;
- sintetički: nad svakom šemom R_i nastalom dekompozicijom se u odgovarajuće F_i kombinatornim postupkom dodaju zavisnosti koje nisu u njemu a mogu se izvesti iz polaznog skupa zavisnosti F.

U svim ovim razmatranjima pod "izvođenje" podrazumevamo primenu algoritma za nalaženje zatvarača skupa atributa.

Vratimo se sada na naš drugi primer kod koga su "izgubljene" zavisnosti SIFO→NAZIVO i NAZIVO→SIFO. Za taj slučaj polazne šeme relacije i skupa funkcijskih zavisnosti smo već ranije odredili netrivijalni zatvarač F^+_n. Njegove projekcije po dekompoziciji { NASLOV1 (SIFN, NAZIVN,NAZIVO), NASLOV2 (SIFN, SIFO) } su, redom:

F1 = { SIFN→SIFN,NAZIVN,NAZIVO NAZIVN→NAZIVN NAZIVO→NAZIVO }

F2 = { SIFN→SIFO }

Ako nad ovakvim F1∪F2 potražimo zatvarače atributa SIFO i NAZIVO dobijamo da su zavisnosti SIFO→NAZIVO i NAZIVO→SIFO neizvodive, odnosno da su suštinski izgubljene.

Isti ishod bi dala i sintetička provera, što se prepušta čitaocima.

7.4 Funkcijske normalne forme i postupci normalizacije

Već smo naglasiti da se normalizacija svodi na pogodnu dekompoziciju šeme relacije i relacije u cilju otklanjanja anomalija ažuriranja. Uzroci tih anomalija su usled prisustva neželjenih funkcijskih zavisnosti. Pod normalnim formama podrazumevamo određene kriterijume valjanosti neke šeme relacije, u smislu da funkcijske zavisnosti koje važe nad tom šemom moraju zadovoljavati određene uslove, koji mogu biti manje ili više strogi. Pre razmatranja te oblasti, neophodno je da se osvrnemo na neke specijalne slučajeve funkcijskih zavisnosti.

7.4.1 Specijalne funkcijske zavisnosti

Navedimo u formi definicija neke specijalne fukcijske zavisnosti koje su važne za definisanje određenih normalnih formi. Neka su pri tome R šema relacije, a X, Y i Z podskupovi R.

Definicija

Funkcijska zavisnost $X \rightarrow Y$ je superključna ako važi $X \rightarrow R$.

Definicija

Funkcijska zavisnost $X \rightarrow Y$ je trivijalna ako važi $X \subseteq Y$

Definicija

Funkcijska zavisnost $X \rightarrow Y$ je totalna ako ne postoji ni jedan pravi podskup Z od X za koji važi $Z \rightarrow Y$, odnosno ako važi:

$$X \rightarrow Y \ \wedge \ \neg \exists Z \ (Z \subset X \wedge Z \rightarrow Y)$$

Definicija

Funkcijska zavisnost $X \rightarrow Y$ je parcijalna ako postoji neki pravi podskup Z od X za koji važi $Z \rightarrow Y$, odnosno ako važi:

$$X \rightarrow Y \ \wedge \ \exists Z \ (Z \subset X \wedge Z \rightarrow Y)$$

Definicija

Funkcijska zavisnost $X \rightarrow Y$ je tranzitivna ako postoji neki podskup atributa Z različit i od X i od Y za koji važi $X \rightarrow Z$ i $Z \rightarrow Y$, odnosno ako važi:

$$X \rightarrow Y \ \wedge \ \exists Z \ (Z \neq X \wedge Z \neq Y \wedge X \rightarrow Z \wedge Z \rightarrow Y)$$

Definicije normalnih formi koje slede zasnivaju se upravo na specijalnim funkcijskim zavisnostima.

Primer

Posmatrajmo jednu krajnje nezgrapnu šemu relacije o pozajmicama

POZAJMICA (SIFN, SIFC, DATUM, DANA, SIFK, NAZIVN, SIFO, NAZIVO)

kao i skup funkcijskih zavisnosti

F = { SIFN,SIFC,DATUM→DANA SIFN,SIFC,DATUM→SIFK
SIFN,SIFC,DATUM→NAZIVN SIFN,SIFC,DATUM→SIFO
SIFN,SIFC,DATUM→NAZIVO SIFK→SIFN SIFN→SIFO
SIFN,SIFC→SIFN SIFO→NAZIVO NAZIVO→SIFO }

Ovde imamo slučajeve svih specijalnih funkcijskih zavisnosti:

- superključne: SIFN,SIFC,DATUM→DANA SIFN,SIFC,DATUM→SIFK
  SIFN,SIFC,DATUM→MAZIVN SIFN,SIFC,DATUM→SIFO
  SIFN,SIFC,DATUM→NAZIVO ;
- trivijalna: SIFN,SIFC→SIFN ;
- totalne: SIFN,SIFC,DATUM→DANA SIFN,SIFC,DATUM→SIFK
  SIFK→SIFN SIFN→SIFO
  SIFO→NAZIVO NAZIVO→SIFO ;
- parcijalme: SIFN,SIFC,DATUM→NAZIVN SIFN,SIFC,DATUM→SIFO ;
- tranzitivna SIFN,SIFC,DATUM→NAZIVO.

Superključne zavisnosti mogu ujedno biti i parcijalne ili tranzitivne.

7.4.2 Opšti postupak funkcijske normalizacije

Neka polazna šema relacije R nije u određenoj normalnoj formi ako u skupu funkciskih zavisnosti F koje važe nad tom šemom postoji bar jedna zavisnost oblika $X \rightarrow Y$ koja narušava definiciju te normalne forme. Pri tome je prirodno da se postupak normalizacije do određene normalne forme sprovodi u koracima. Početna vrednost normalizovane dekompozicije D (skupa relacija R_i na koje je dekomponovana R) je polazna šema relacije R, dok je početna vrednost projekcije F_D polaznog skupa zavisnosti F po dekompoziciji D taj polazni skup F.

Pri svakom koraku normalizacije posmatra se jedna šema R_i (na samom početku to je polazno R) iz dekompozicije D. Za nju se prvo određuje skup svih kandidat-ključeva i nalazi se prva zavisnost $X \rightarrow Y$ koja narušava željenu normalnu formu. Zatim se vrši dekompozicija posmatrane šeme relacije R_i tako što se neželjena zavisnost izdvaja u posebnu šemu $R_{ik}(\underline{X}, Y)$ koja se dodaje dekompoziciji D, a R_i se redukuje tako što se iz nje izostavi skup atributa Y, odnosno svodi se na R_i-Y. Pri tome se vrši i dekompozicija odgovarajućeg skupa funkcijskih zavisnosti F_i na svoje projekcije F_i po redukovanom R_i i F_{ik} po novostanalom R_{ik} i F_{ik} se dodaje u F_D. Postupak se ponavlja sve dok se ne postigne da su sve šeme relacije u D u željenoj normalnoj formi.

Na osnovu toga, možemo formulisati sledeći opšti algoritam normalizacije u neku željenu normalnu formu N:

```
NormalizacijaN ( > R, > F, < D, < FpoD )
:  Inicijalizacija
:  :  D      = {R}
:  :  FpoD   = {F}
:  Normalizacija dok se ne postigne da su sve seme normalizovane
:  :  DO FOREVER
:  :  |  Inicijalizacija brojaca izdvojenih zavisnosti
:  :  |  :  Izdvojeno = 0
:  :  |  Normalizacija svih sema
:  :  |  :  DO FOR EACH ( Ri in D )
:  :  |  :  |  Normalizacija jedne seme
:  :  |  :  |  :  Odredjivanje kandidat-kljuceva
:  :  |  :  |  :  :  K = KandidatKljucevi ( Ri,Fi )
:  :  |  :  |  :  Incijalizacija indikatora ispitivanja
:  :  |  :  |  :  :  Izdvojiti = FALSE
:  :  |  :  |  :  Ispitivanje svih zavisnosti
:  :  |  :  |  :  :  DO FOR EACH ( X→Y IN Fi )
:  :  |  :  |  :  :  |  Ispitivanje jedne zavisnosti
:  :  |  :  |  :  :  |  :  IF ( Y ⊄ X )
:  :  |  :  |  :  :  |  :  |  IF ( NOT ZavisnostN ( X→Y,Fi,K ) )
:  :  |  :  |  :  :  |  :  |  |  Nadjena zavisnost za izdvajanje
:  :  |  :  |  :  :  |  :  |  |  :  Izdvojiti = TRUE
:  :  |  :  |  :  :  |  :  |  |  :  Izdvojeno = Izdvojeno + 1
:  :  |  :  |  :  <:--|--:--|--|--:--EXIT
:  :  |  :  |  :  Redukcija i izdvajanje (ako treba)
:  :  |  :  |  :  :  IF ( Izdvojiti )
:  :  |  :  |  :  :  |  ReduR   = Ri – Y
:  :  |  :  |  :  :  |  NovoR   = X ∪ Y
:  :  |  :  |  :  :  |  D = ( D - {Ri} ) ∪ ReduR ∪ NovoR
:  :  |  :  |  :  :  |  ReduF   = ProjekcijaFPoS ( Fi,ReduR )
:  :  |  :  |  :  :  |  NovoF   = ProjekcijaFPoS ( Fi,NovoR )
:  :  |  :  |  :  :  |  FpoD  = ( D - {Fi}) ∪ ReduF ∪ Novo
:  :  |  Kraj ako su sve seme normalizovane
:  :  |  :  IF ( Izdvojeno == 0 )
:  <:--|--:--|-- EXIT
```

Pri tome, usvojili smo oznake

R - polazna relacija;

F - polazni skup funkcijskih zavisnosti;

D - rezultat - dekompozicija $\{R_1,...,R_N.\}$;

FpoD - rezultat - projekcija $F[R_1] \cup ... F[R_N]$;

K - skup kandidat-ključeva relacije R_i ;

ReduR - redukovana relacija R_i nastala izostavljanjem Y ;

NovoR - nova relacija R_j nastala izdvajanjem $X \rightarrow Y$;

ReduF - projekcija F_i po $RedR_i$;

NovoF - projekcija F_i po $NovR_j$;

Sa *ZavisnostN* smo označili funkciju koja ispituje da li je zadata netrivijalna funkcijska zavisnost u normalnoj formi N. Ta funkcija će biti naknadno precizirana za svaku od normalnih formi koje razmatramo.

Suština ovog na izgled složenog algoritma je u sledećem:

- na početku, dekompoziciju D čini polazna relacija R, a projekciju FpoD polazni skup funkcijskih zavisnosti;
- koraci normalizacije se sprovode sve dok se ne postigne da su sve relacije R_i u dekompoziciji D u željenoj normalnoj formi;
- relacija R_i u dekompoziciji D je u željenoj normalnoj formi kada nad njom ne važi ni jedna funkcijska zavisnosti koja narušava tu normalnu formu;
- svaka relacija R_i koja nije u željenoj normalnoj formi zbog neke zavisnosti $X \rightarrow Y$ redukuje se tako što gubi iz svog sastava Y, a pri tome nastaje nova relacija R_j koju čine X i Y.

Algoritam *NormalizacijaN* daje dekompoziciju D bez gubitaka podataka, pošto je pri svakom koraku dekompozicije R_i na redukovano R_i i novo R_j zadovoljen uslov reverzibilnosti $R_i \cap R_j \rightarrow R_j$, s obzirom da je taj presek desna strana zavisnosti $X \rightarrow Y$ koju izdvajamo u R_j, odnosno ključ je u R_j. Sa druge strane, ne postoji garancija da će pri tome biti očuvane i sve polazne funkcijske zavisnosti.

Treba naglasiti da je procedura *NormalizacijaN* navedena u pojednostavljenoj formi. Kompletna forma te procedure podrazumeva i određivanje zatvarača F^+ polaznog skupa zavisnosti F i korišćenje zavisnosti iz F^+ u postupku normalizacije.

7.4.3 Druga normalna forma

Definicija

Šema relacije R je u drugoj normalnoj formi ako nad njom ne postoji ni jedna funkcijska zavisnost po kojoj neki neključni atribut parcijalno zavisi od bilo kog kandidat-ključa.

Formulišimo sada algoritam *Zavisnost2* koji za dati skup kandidat-ključeva K ispituje da li je funkcijska zavisnost X→Y u skladu sa definicijom druge normalne forme. Podrazumeva se da i X i Y pripadaju šemi relacije R.

```
Zavisnost2 ( > X→Y, > K )
:  Inicijalizacija
:  :  XnepripadaK = TRUE
:  :  YpripadaK   = TRUE
:  Ispitivanje
:  :  1. ispitivanje za sve kandidat-kljuceve
:  :  :  DO FOR EACH ( Z in K )
:  :  :  |  Ispitivanje da li je X deo kandidat-kljuca
:  :  :  |  :  IF ( X ⊂ Z )
:  :  :  |  :  |  XnepripadaK = FALSE
:  : <:--|--:--|--EXIT
:  :  2. ispitivanje za sve kandidat-kljuceve (ako treba)
:  :  :  IF ( NOT XnepripadaK )
:  :  :  |  DO FOR EACH ( Z in K )
:  :  :  |  |  Ispitivanje da li Y nije deo kandidat-kljuca
:  :  :  |  |  :  IF ( Y ⊄ Z )
:  :  :  |  |  :  |  YpripadaK = FALSE
:  : <:--|--|--:--|--EXIT
:  RETURN XnepripadaK ∨ YpripadaK
```

Ovde treba napomenuti: to što je leva strana X zavisnosti X→Y pravi podskup nekog kandidat-ključa Z još uvek ne znači da je narušena druga normalna forma, pošto pri tome desna strana Y može da ne sadrži ni jedan neključni atribut.

Primer

Posmatrajmo šemu relacije kojom smo u odeljku 7.1 ilustrovali efekte loše strukture relacije:

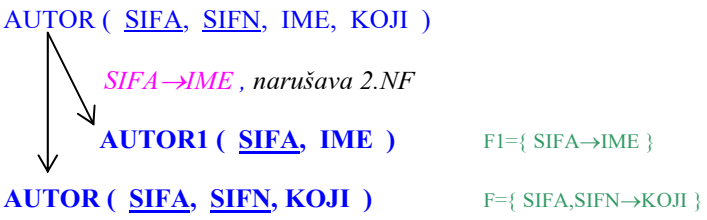
AUTOR (SIFA, SIFN, IME, KOJI)

kao i skup funkcijskih zavisnosti nad njom

F = { SIFA,SIFN→IME,KOJI SIFA→IME }

Kandidat-ključ ove šeme je SIFA,SIFN . Zavisnost SIFA,SIFN→IME je parcijalna, pošto IME zavisi od dela kandidat-ključa po zavisnosti SIFA→IME.

Postupak normalizacije na drugu normalnu formu sastoji se u tome da zavisnost SIFA→IME izdvojimo u posebnu šemu relacije a iz polazne šeme uklonimo desnu stranu te zavisnosti, odnosno IME. To daje sledeću dekompoziciju na šeme relacija i skupove funkcijskih zavisnosti:



Pri tome je uklonjena neželjena parcijalna funkcijska zavisnost.

7.4.4 Treća normalna forma

Definicija

Šema relacije R je u trećoj normalnoj formi ako nad njom ne postoji ni jedna funkcijska zavisnost po kojoj neki neključni atribut tranzitivno zavisi od kandidat-ključa.

Umesto da navodimo kakve funkcijske zavisnosti nisu dozvoljene, možemo formulisati definiciju treće normalne forme u kojoj se navode dozvoljene funkcijske zavisnosti. To je pogodnije kod normalnih formi iznad druge, kod kojih je mnogo manje toga dozvoljeno nego što je zabranjeno.

Definicija

Šema relacije R je u trećoj normalnoj formi ako svaka funkcijska zavisnost $X \rightarrow Y$ koja važi nad njom zadovoljava bar jedan od uslova:

- zavisnost je trivijalna, odnosno $Y \subseteq X$;
- zavisnost je superključna, odnosno $X \rightarrow R$;
- Y je deo kandidat-ključa, odnosno $\exists Z \, (Y \subset Z \wedge Z \rightarrow R)$.

Iz ove definicije i prethodne definicije druge normalne forme jasno je da je svaka šema relacije koja je u trećoj normalnoj formi istovremeno i u drugoj normalnoj formi: ni jedan od navedena tri uslova ne dozvoljava postojanje zavisnosti neključnog atributa od dela kandidat-ključa.

Algoritam *Zavisnost3*, koji za dati skup dati skup kandidat-ključeva K ispituje da li je funkcijska zavisnost X→Y u skladu sa definicijom treće normalne forme, glasi:

```
Zavisnost3  ( > X→Y, > K )
:  Inicijalizacija
:  :  Jeste = FALSE
:  Ispitivanje
:  :  Provera da li je zavisnost trivijalna
:  :  :  IF ( Y ⊆ X )
:  :  :  |  Jeste = TRUE
: <:--:--|--EXIT
:  :  Provera za sve kandidat-kljuceve
:  :  :  DO FOR EACH ( Z IN K )
:  :  :  |  Provera da li Y deo kandidat-kjuca
:  :  :  |  :  IF ( Y ⊆ Z )
:  :  :  |  :  |  Jeste = TRUE
: <:--:--|--:  |--EXIT
:  :  :  |  Provera da li je X super-kljuc
:  :  :  |  :  IF ( Z ⊆ X )
:  :  :  |  :  |  Jeste = TRUE
: <:--:--|--:--|--EXIT
:  RETURN Jeste
```

Primer

Posmatrajmo relaciju o naslovima koja sadrži sve podatke o naslovima, oblastima i autorima:

NASLOV (SIFN, SIFA, KOJI, NAZIVN, IME, SIFO, NAZIVO)

kao i odgovarajući skup funkcijskih zavisnosti:

F = { SIFN,SIFA→KOJI,NAZIVN,IME,SIFO,NAZIVO
SIFN→NAZIVN,SIFO SIFA→IME SIFO→NAZIVO NAZIVO→SIFO }

Ako primenimo ranije opisani univerzalni postupak normalizacije uz primenu kriterijuma za treću normalnu formu, dobijamo dekompoziciju:

NASLOV (SIFN, SIFA, KOJI, NAZIVN, IME, SIFO, NAZIVO)

SIFO→NAZIVO , narušava 3. NF

NASLOV1 (SIFO, NAZIVO) F1={SIFO→NAZIVO NAZIV→OSIFO}

NASLOV (SIFN, SIFA, KOJI, NAZIVN, IME, SIFO)

SIFN→NAZIVN,SIFO , narušava 2.NF

NASLOV2 (SIFN, NAZIVN, SIFO) F2={ SIFN→NAZIVN,SIFO }

NASLOV (SIFN, SIFA, KOJI, IME)

SIFA→IME , narušava 2.NF

NASLOV3 (SIFA, IME) F3={ SIFA→IME }

NASLOV4 (SIFN, SIFA, KOJI) F={ SIFN,SIFA→KOJI }

Za dekompoziciju F' polaznog skupa funkcijskih zavisnosti F dobili smo:

F = { SIFO→NAZIVO SIFN→NAZIVN,SIFO SIFA→IME SIFN,SIFA→KOJI }

U ovom skupu u odnosu na polazni "izgubljena" je zavisnost

SIFN,SIFA→KOJI,NAZIVN,IME,SIFO,NAZIVO

ali se to da je ona izvodiva iz F' može pokazati nalaženjem zatvarača $(SIFN,SIFA)^+$ nad skupom zavisnosti F' ili primenom pravila izvođenja. Drugim rečima, sve zavisnosti iz polaznog skupa F, odnosno njihovo važenje, su očuvani.

7.4.5 Bojs-Kodova normalna forma

Za prethodne dve normalne forme, lako uočavamo da je u oba slučaja uzročnik nevolja postojanje neželjenih tranzitiivnih funkcijskih zavisnosti, pošto parcijalnu zavisnost možemo smatrati za specijalni slučaj tranzitivne: neključni atribut funkcijski zavisi od dela kandidat-ključa, a ovaj zavisi od celog kandidat-ključa.

U opštem slučaju, nad šemom relacije mogu da postoje četri vrste tranzitivnih funkcijskih zavisnosti koje dovode do anomalija ažuriranja:

- zavisnost neključnog atributa posredstvom dela kandidat ključa;
- netrivijalna zavisnost neključnog atributa posredstvom podskupa neključnih atributa;
- netrivijalna zavisnost ključnog atributa posredstvom dela kandidat ključa;
- zavisnost ključnog atributa posredstvom podskupa neključnih atributa.

Druga i treća normalna forma iz dekompozicije neke šeme relacije eliminišu samo tranzitivne zavisnosti prve i druge vrste, pa se opravdano postavlja pitanje definisanja neke strožije normalne forme koja sasvim eliminiše tranzitivne zavisnosti.

Definicija

Šema relacije R je u Bojs-Kodovoj (Boyce-Codd) normalnoj formi ako svaka funkcijska zavisnost $X \rightarrow Y$ koja važi nad njom zadovoljava jedan od uslova:

- zavisnost je trivijalna, odnosno $Y \subseteq X$;
- zavisnost je superključna, odnosno $X \rightarrow R$.

Razlika u odnosu na definiciju treće normalne forme je samo u tome što se ne dozvoljava treći slučaj (tranzitivna zavisnost dela kandidat-ključa), pa je šema relacije koja je u Bojs-Kodovoj normalnoj formi istovremeno i u trećoj normalnoj formi.

Sledeći algoritam za dati skup kandidat-ključeva K utvrđuje da li je funkcijska zavisnost X→Y saglasna definiciji Bojs-Kodove normalne forme:

```
ZavisnostBC ( > X→Y, > K )
:  Inicijalizacija
:  :  Jeste = FALSE
:  Ispitivanje
:  :  Provera da li je zavisnost trivijalna
:  :  :  IF ( Y ⊆ X )
:  :  :  |  Jeste = TRUE
: <:--:--|--EXIT
:  :  Provera za sve kandidat-kljuceve
:  :  :  DO FOR EACH ( Z IN K )
:  :  :  |  Provera da li je zavisnost super-kljucna
:  :  :  |  :  IF ( Z ⊆ X )
:  :  :  |  :  |  Jeste = TRUE
: <:--:--|--:--|--EXIT
:  RETURN Jeste
```

Uočimo da se ovaj algoritam razlikuje od prethodnog algoritma *Zavisnost3* samo po tome što je uslov ispitivanja strožiji, bez dodatne mogućnosti Y⊂ Z.

Bojs-Kodova normalna forma je najviša normalna forma koja se može postići na bazi funkcijskih zavisnosti.

Normalizacija u Bojs-Kodovu normalnu formu (i uopšte univerzalnim postupkom normalizacije) ima i svoju cenu: ne postoji garancija očuvanja funkcijskih zavisnosti. Šta više, redosled eliminisanja neželjenih funkcijskih zavisnosti može uticati na stepen gubitka zavisnosti. Sa druge strane, postoji specijalni algoritam normalizacije u treću normalnu formu koji garantuje očuvanje svih funkcijskih zavisnosti. Iz tih razloga, danas se treća normalna forma smatra za sasvim zadovoljavajuću kod relacionih baza podataka. Navedeni specijalni algoritam nećemo razmatrati, pošto uključuje pojmove minimalnog zatvarača i prstenastog pokrivača skupa funkcijskih zavisnosti koji nisu obuhvaćeni ovim udžbenikom.

Primer

Posmatrajmo raniju šemu relacije o pozajmicama koja sadrži podatke o naslovima, članovima:i knjigama:

POZAJMICA (SIFN, SIFC, DANA, NAZIVN, SIFK)

kao i odgovarajući skup funkcijskih zavisnosti:

F = { SIFN,SIFC,DANA→NAZIVN,SIFK SIFK→SIFN SIFN→NAZIVN }

Primenom algoritma normalizacije dobijamo sledeću dekompoziciju:

Prazan skup zavisnosti {} označava da u odgovarajućoj relaciji važe samo trivijalne funkcijske zavisnosti. I u ovom primeru nemamo gubitak zavisnosti, što se lako pokazuje na način koji smo već opisali.

7.4.6 Prva normalna forma

U jednom značajnom broju udžbenika iz oblasti baza podataka prva normalna forma razmatra se tek nakon svih ostalih normalnih formi na bazi funkcijskih zavisnosti. To nije bez razloga: u određenim situacijama, kada se prvo izvrši normalizacija u prvu normalnu formu, dolaze do izražaja nove vrste zavisnosti koje nisu funkcijskog karaktera.

U svim primerima do sada, i u poglavljima koja su se bavila jezicima relacionih baza podataka i u ovom poglavlju, podrazumevalo se da su vrednosti svih atributa skalari. To je prilikom formulisanja relacionog modela podataka pretpostavio i njegov tvorac Kod - otud ono "prva" u nazivu. Ovo predstavlja jednu vrstu implementacionog ograničenja nad tipovima atributa koja ne postoji u savremenim programskim jezicima. Primera radi još je programski PASCAL dozvolio ograničeni skupovni tip nad određenim baznim tipom, dok programski jezik Java dozvoljava skupove proizvoljnog prostog ili složenog tipa uključujući i skupove skupova.

Primer

Relacija **je_autor** iz naše baze podataka BIBLIOTEKA može se napisati na način koji ukazuje na to da svakom naslovu, odnosno svakoj vrednosti atributa SIFN može da odgovara skup parova vrednosti atributa SIFA i KOJI:

```
je_autor ( SIFN ( SIFA  KOJI )           )
         ------------------------------
         RBP0 {<AP0,1>,<JN0,2>}

         RK00 {<DM0,1>}

         PP00 {<ZP0,1>,<DM0,2>,<IT0,3>}

         PJC0 {<AP1,1>,<ZP0,2>}
         ------------------------------
```

Saglasno konceptu univerzalnog relacionog modela koji je izložen na kraju poglavlja 4, šema ovakve relacije zapisuje se u formi "relacije u relaciji"

JE_AUTOR (SIFN, (SIFA, KOJI))

čime se naglašava da jednoj vrednosti SIFN može da odgovara više parova vrednosti atributa SIFA i KOJI. Pri tome, uočavamo da se te dve vrednosti ponavljaju vezano, dakle ne nezavisno jedna od druge.

Ovakve situacije su se često javljale u praksi i način na koji su razrešavane u klasičnim informacionim sistemima svodio se na nešto što bi se moglo opisati šemom relacije sa promenljivim brojem ponavljanja atributa:

JE_AUTOR (SIFN, $SIFA_1$, $KOJI_1$, $SIFA_2$, $KOJI_2$, ...)

Ovakvo rešenje podrazumeva promenljivu dužinu sloga, što se vrlo teško realizuje u uslovima rada sa direktnim pristupom slogovima. Sa druge strane, korišćenje fiksne dužine sloga zahteva da se unapred opredelimo za maksimalni mogući broj ponavljanja atributa. Za slučaj da je usvojeno maksimalno tri ponavljanja, imali bi šemu relacije

JE_AUTOR (SIFN, $SIFA_1$, $KOJI_1$, $SIFA_2$, $KOJI_2$, $SIFA_3$, $KOJI_3$)

a sama relacija **je_autor** bi u tom slučaju glasila:

je_autor (SIFN	$SIFA_1$	$KOJI_1$	$SIFA_2$	$KOJI_2$	$SIFA_3$	$KOJI_3$)
RBP0	AP0	1	JN0	2	null	null
RK00	DM0	1	null	null	null	null
PP00	ZP0	1	DM0	2	IT0	3
PJC0	AP1	1	ZP0	2	null	null

Ovim smo postigli da u relaciji imamo onoliko torki koliko ima naslova, ali su mane ovakvog rešenja evidentne:

- ako se ikada pojavi naslov sa više od tri autora, nismo u mogućnosti da to unesemo u ovakvu relaciju;
- upiti i druge operacije nad ovakvom relacijom su izuzetno komplikovani (čitaocu se prepušta da sastavi na bilo kom jeziku upit koji daje šifre autora po naslovima, ili šifre naslova koje je napisao autor šifre DM0.

Situacija je još gora ako vrednosti koje se ponavljaju mogu nastajati i nestajati proizvoljan broj puta i proizvoljnim redosledom. Neka na osnovu pretpostavke da neki član biblioteke može kod sebe da drži najviše tri knjige imamo šemu relacije

DRZI (SIFC, $SIFK_1$, $DATUM_1$, $SIFK_2$, $DATUM_2$, $SIFK_3$, $DATUM_3$)

Probleme sa upitima kod ovakvih šema relacija smo upravo opisali. Navedimo sada samo neke od problema ažuriranja:

- ako član koji je uzeo tri knjige vrati dve od njih, za vrednosti šifara tih knjiga i datuma moramo upisati NULL vrednosti;
- ako taj član, koji kod sebe drži jednu knjigu, uzme još jednu, šifru te knjige i datum uzimanja moramo upisati u relaciju, ali gde – na drugom ili trećem mestu ?

Iz ovih nekoliko primera zaključujemo: situacija da vrednosti nekog atributa u jednoj torci u relaciji odgovara skup vrednosti je neprihvatljiva, kako sa gledišta organizacije podataka tako i sa gledišta operacija nad takvim podacima.

Definicija

Šema relacije R je u prvoj normalnoj formi ako je svaki njen atribut skalarnog domena.

U pitanju je ograničenje koje nije pominjano u definicijama druge, treće i Bojs-Kodove normalne forme, pa bi u tim uslovima svaku od tih defincija trebalo dopuniti na samom početku klauzulom "ako je u prvoj normalnoj formi".

Normalizacija, odnosno svođenje šeme relacije na prvu normalnu formu, krajnje je jednostavna. Za neku šemu relacije R , ta normalizacija se može formalno prikazati kao transformacija

$$R (X, (Y)) \quad == \quad R1 (X, Y)$$

gde smo sa Y označili (jedini) skup atributa koji se zajedno ponavljaju, a sa X ostale atribute. U najnepovoljnijem slučaju, ključ šeme relacije R1 će činiti ključ šeme relacije R zajedno sa Y , ali usled funkcijskih zavisnosti koje važe nad R1 umesto celog Y u sastav ključa R1 može ući samo jedan njegov deo.

Što se tiče relacija **r** i **r1** nad tim šemama, stanje je sledeće: od svake torke $x_i,(y_i)$ u **r** nastaje onoliko torki x_i,y_k u **r1** koliko je bilo vrednosti u skupu (y_i). Ako je (y_i) prazno, neće nastati ni jedna torka. Iz toga zaključujemo da od N(r) torki polazne relacije **r** u relaciji **r1** nastaje $N(r1)=N_1+...+N_{N(r)}$ torki, gde je N_i broj vrednosti u skupovima (y_i).

Primer

Posmatrajmo šemu relacije iz prethodnog primera

JE_AUTOR (SIFN, (SIFA, KOJI))

nad kojom važi skup zavisnosti

F = { SIFN→(SIFA, KOJI) , SIFN,SIFA→KOJI }

gde sa X→ (Y) označavamo da svakoj vrednosti X odgovara jedan skup vrednosti Y.

Normalizacijom dobijamo šemu relacije

JE_AUTOR1 (SIFN, SIFA, KOJI)

pri čemu je skup zavisnosti nad njom sveden na

F1 = { SIFN,SIFA→KOJI }

što je uslovilo da KOJI ne uđe u sastav jedinog kandidat-ključa.

Normalizacijom relacije **je_autor** dobija se relacija **je_autor1**

je_autor1 (SIFN SIFA KOJI)

SIFN	SIFA	KOJI
RBP0	AP0	1
RBP0	JN0	2
RK00	DM0	1
PP00	ZP0	1
PP00	DM0	2
PP00	IT0	3
PJC0	AP1	1
PJC0	ZP0	2

pri čemu smo umesto slogovnog atributa <SIFA,KOJI> uveli elementarne atribute SIFA i KOJI, pa se u prikazu sadržaja relacije izgubilo "<>".

Do sada smo imali slučaj sa samo jednim skupom atributa Y koji se zajedno ponavljaju. U opštem slučaju, u šemi relacije R može biti prisutno više takvih skupova:

$$R (X, (Y1), ..., (YM))$$

Odmah treba naglasiti jednu bitnu stvar: skupovi (Y1), ..., (YM) su uzajamno potpuno nezavisni. To će biti vrlo jasno iz primera koji će uslediti.

Normalizacija ovakve šeme relacije formalno se može predstaviti kao transformacija:

$$R (X, (Y1), ..., (YM)) == R (X, Y1,...,YM)$$

Ključ konačne šeme relacije R činiće u najnepovoljnijem slučaju ključ šeme relacije R (neki podskup X) i svi skupovi atributa Yi , ali funkcijske zavisnsosti nad R mogu usloviti i manji ključ.

Da bi bolje razumeli šta se dešava sa relacijom **r** zamislimo da normalizaciju sprovodimo u korak po korak, pri čemu svaki put razrešavamo jedan skup atributa (Yi). Radi preglednosti relaciju nastalu u i-tom koraku označićemo sa ri, mada treba imati na umu da se sve dešava nad jednom relacijom. Neka je N broj torki u polaznoj relaciji **r** i neka su $Ni_1,...,Ni_{N(r)}$ redom ukupni brojevi vrednosti u skupovima $(yi_1),...,(yi_{N(r)})$. U prvom koraku, imali bi kao rezultat normalizacije po (Y1) relaciju **r1** sa N(r1) torki

$$\mathbf{r} (X, (Y1), ..., (YM)) == \mathbf{r1} (X, Y1, (Y2),...,(YM)) \quad N(r1)=N1_1+...+N1_{N(r)}$$

dok bi drugi korak normalizacije po (Y2) dao

$$\mathbf{r1} (X, Y1, (Y2),...,(YM)) == \mathbf{r2} (X, Y1, Y2, (Y3)...,(YM)) \quad N(r2)=N2_1+...+N2_{N(r1)}$$

I bez navođenja daljih koraka je jasno šta se dešava: na kraju, naša potpuno normalizovana relacija **rM** može imati znatno veći broj torki od polazne.

Primer

Posmatrajmo šemu relacije NASLOV koja je rezultat nastojanja da se u jednoj relaciji uz naslove evidentiraju i podaci o autorima i članovima koji su pozajmljivali te naslove jednom ili više puta:

NASLOV (SIFN, (SIFA), (SIFC))

Skup funkcijskih zavisnosti nad tom šemom je:

F = { SIFN→(SIFA) , SIFN→ (SIFC) }

Posle dva koraka normalizacije dobijamo šemu relacije

NASLOV (SIFN, SIFA, SIFC)

i skup netrivijalnih funkcijskih zavisnosti

F2 = { }

Pošto je netrivijalno F2 prazno, zaključili smo da su svi atributi šeme relacije NASLOV2 ključni.

Pogledajmo sada šta se dobija iz polazne relacije **naslov** čiji bi sadržaj, prema podacima iz naše baze podata BIBLIOTEKA, bio sledeći

```
naslov ( SIFN (SIFA)        (SIFC)        )
         -------------------------------
         RBP0 {AP0,JN0}     {PP0}

         RK00 {DM0}         {}

         PP00 {ZP0,DM0,IT0} {PP0,JJ0,JJ1}

         PJC0 {AP1,ZP0}     {JJ0}
         -------------------------------
```

Obratimo pažnju na to da niko nije pozajmljivao naslov RK00 i da je usled toga odgovarajuća vrednost atributa SIFC prazan skup.

Neka je prvi korak normalizacije sproveden u odnosu na atribut SIFA. Relacija **naslov** će posle toga imati sadržaj:

```
naslov ( SIFN  SIFA  (SIFC)        )
         ------------------------
         RBP0  AP0   {PP0}
         RBP0  JN0   {PP0}

         RK00  DM0   {}

         PP00  ZP0   {PP0,JJ0,JJ1}
         PP00  DM0   {PP0,JJ0,JJ1}
         PP00  IT0   {PP0,JJ0,JJ1}

         PJC0  AP1   {JJ0}
         PJC0  ZP0   {JJ0}
         ------------------------
```

Kod drugog koraka normalizacije suočavamo se sa specifičnom situacijom. Za naslov RK00 odgovarajući skup vrednosti SIFC je prazan, što ima za posledicu da u relaciji **naslov** ne nastaje ni jedna torka sa 'RK00' kao vrednošću atributa SIFN. Eventualni pokušaj da takvu torku ipak formiramo sa NULL kao vrednošću atributa SIFC bio bi u suprotnosti za uslovom da ni jedan deo ključa ne sme biti NULL vrednost (uslov egzistencijalnog integriteta).

Konačno normalizovana relacija **naslov2** glasi:

```
naslov ( SIFN  SIFA  SIFC )
       ---------------
       RBP0  AP0   PP0
       RBP0  JN0   PP0

       PP00  ZP0   PP0
       PP00  ZP0   JJ0
       PP00  ZP0   JJ1
       PP00  DM0   PP0
       PP00  DM0   JJ0
       PP00  DM0   JJ1
       PP00  IT0   PP0
       PP00  IT0   JJ0
       PP00  IT0   JJ1

       PJC0  AP1   JJ0
       PJC0  ZP0   JJ0
       ---------------
```

Konačna šema relacije NASLOV je sa gledišta funkcijskih zavisnosti maksimalno normalizovana i nalazi se u Bojs-Kodovoj normalnoj formi. Cela šema je ujedno i jedini kandidat-ključ, i nad njom važe samo trivijalne funkcijske zavisnosti. Međutim, ako je suditi po sadržaju relacije **naslov** , ova šema je daleko od idealne:

- prisutna je redundansa: unutar markirane grupe torki, podatak o tome da je neko autor nekog naslova mora da je prisutan onoliko puta koliko je članova pozajmljivalo taj naslov; obrnuto, podatak o tome da je neko pozajmljivao neki naslov prisutan je onoliko puta koliko taj naslov ima autora;
- prisutna je anomalija unošenja: podatke o tome ko su autori nekog naslova ne možemo uneti ako niko nije pozajmio taj naslov;
- prisutna je anonalija brisanja: ako uklonimo podatke o pozajmicama nekog naslova, gubimo i podatke o autorstvu tog naslova.

Razlog za navedene anomalije je u tome što smo normalizaciju u prvu normalnu formu sproveli "unarno", transformišući redom jednu te istu relaciju do željene forme kako bi zadovoljili ograničenje skalarnosti atributa koje je u samoj osnovi relacionog modela u tradiconalnom smislu. A kao posledica takvog pristupa vremenom su uvedene dodatne normalne forme - četvrta i peta - koje su komplikovane i potpuno nepotrebne u uslovima univerzalnog relacionog modela.

Uz prethodni postoji i drugi pristup normalizaciji u prvu normalnu formu koji se svodi na postepeno izdvajanje jedne po jedne zavisnost X→(Yi), slično onome što smo imali kod funkcijskih zavisnosti, pri čemu je zadovoljen uslov da dekompozicija bude bez gubitaka (sa Z smo označili ostale atribute koji su skalari):

r (X, Z, (Y1), ... , (YM)) F = { X →Z, X →(Y1) , ..., X →(YM) }

X→(Y1) , narušava 1.NF

r1 (X, (Y1)) == **r1** (X, Y1)

r (X, Z, (Y2), ... , (YM)) F = { X →(Y2) , ..., X →(YM) }

X→(Y2) , narušava 1.NF

r2 (X, (Y2)) == **r2** (X, Y2)

r (X, Z, (Y3), ... , (YM)) F = { X →(Y3) , ..., X →(YM) }

: :

r (X, Z, (YM)) F = { X →(YM) }

X→(YM) , narušava 1.NF

r2 (X, (YM)) == **r1** (X, Y1))

r (X,Z) F = { X →Z }

U našem konkretnom slučaju šeme relacije NASLOV imali bi

NASLOV (SIFN, (SIFA), (SIFC)) F = { SIFN→(SIFA) , SIFN→ (SIFC) }

SIFN→(SIFA) , narušava 1.NF

NASLOV1 (SIFN, (SIFA)) == **NASLOV1** (SIFN,SIFA)

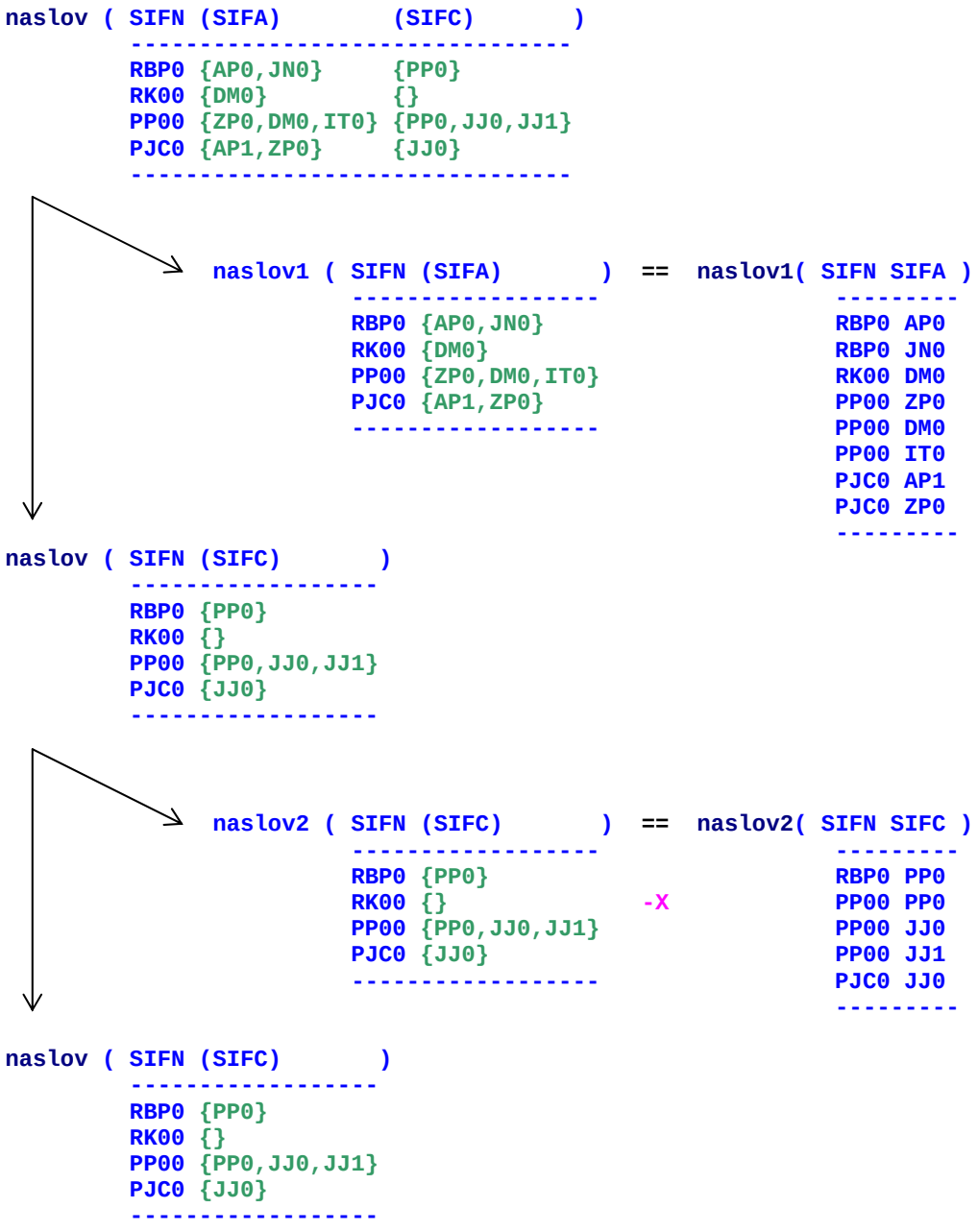
NASLOV (SIFN, (SIFC)) F = { SIFN→ (SIFC) }

SIFN→(SIFC) , narušava 1.NF

NASLOV2 (SIFN, (SIFC)) == **NASLOV2** (SIFN,SIFC)

NASLOV (SIFN) F = { }

Pri tome bi redom imali odgovarajuće krajnje relacije:



Ako bi želeli da prethodni postupak sprovedemo u skladu sa notacijom univerzalnog relacionog modela izloženom na kraju poglavlja 4, imali bi sledeću polaznu situaciju

NASLOV (SIFN, AUTOR(), POZAJMICA()) F = { SIFN → AUTOR () ,POZAJMICA() }

AUTOR (SIFA) F = {}

POZAJMICA (SIFC) F = {}

i shodno tome dekompoziciju:

NASLOV (SIFN, AUTOR(), POZAJMICA()) F = { SIFN→AUTOR(),POZAJMICA() }

SIFN→AUTOR() , narušava 1.NF

NASLOV_A (SIFN, AUTOR())

|

NASLOV_A (SIFN, (SIFA)) == **NASLOV_A** (SIFN,SIFA)

NASLOV (SIFN, POZAJMICA()) F = { SIFN→POZAJMICA() }

SIFN→POZAJMICA() , narušava 1.NF

NASLOV_P (SIFN, POZAJMICA())

|

NASLOV_P (SIFN, (SIFC)) == **NASLOV_P** (SIFN,SIFC)

NASLOV (SIFN) F = {}

U ovom primeru atributi-relacije su bili jednostavni, sa samo jednim skalarnim atributom koji je ujedno bio i jedini kandidat-ključ, ali za jednu instancu relacije odnosno unutar jedne torke relacije **naslov**. Jedna vrednost SIFA može je da se pojavi samo jednom u svakoj torci te relacije, ali se može zato javiti u drugim torkama, odnosno unutar cele relacije **naslov** pojaviti više puta.

U slučaju složenih atributa-relacija, sastavljenih od više skalarnih atributa, unutar tog skupa atributa mogu postojati funkcijske zavisnosti, pri čemu su moguća dva slučaja:

- zavisnost je "lokalna": $X \rightarrow Y$ važi unutar jedne instance relacije, odnosno unutar jedne torke obuhvatajuće relacije, a ne važi izvan toga;
- zavisnost je "globalna": $X \rightarrow Y$ važi ne samo unutar jedne instance relacije, nego i u svim instancama obuhvatajuće relacije, odnosno u svim njenim torkama; jednoj vrednosti X u svim torkama obuhvatajuće relacije odgovara jedna vrednost Y.

Iz prethodnog proizilazi da je za svaki skup zavisnosti potrebno naznačiti ne samo za koju šemu relacije važi, nego i to koje zavisnosti su lokalne a koje globalne. Za lokalne zavisnosti usvajamo uobičajenu oznaku "$\rightarrow$", a za globalne "$\gamma\rightarrow$".

7.4.7 Univerzalni postupak normalizacije

Na osnovu do sada izloženog i usvajajući univerzalni relacioni model, možemo da formulišemo univerzalni postupak normalizacije neke šeme relacije sa atributima-relacijama. Neka je ta šema iz oblika

$$R (Z, S_1<>, ... , S_j<>, ... , S_m<>, R_1(), ... , R_k(), ... , R_n())$$

gde je Z skup skalarnih atributa, $S_j<>$ su atributi-slogovi a $R_k()$ atributi-relacije prvo prevedena u oblik

$$R (X, R_1(), ... , R_k(), ... , R_p())$$

gde X uz prvobitne skalarne atribute Z sadrži i skalarne atribute iz svih $S_j<>$, a niz atributa-relacija $R_1(), ... , R_p()$ uz prvobitnih n sadrži i one koji su se nalazili u sastavu atributa-slogova. Neka nad tom šemom važi skup zavisnosti oblika

$$F = \{ Y_u \rightarrow Z_u \} \cup \{ Y_v \rightarrow R_v() \}$$

gde su Y_u, Y_v i Z_u podskupovi X. Neka slično tome važi i za atribute-relacije

$$R_k (X_k, R_{k1}(), ... , R_{kj}(), ... , R_{kp}())$$

$$F_k = \{ Y_{ku} \rightarrow Z_{ku} \} \cup \{ Y_{kv} \rightarrow R_{kv}() \}$$

pri čemu se na kraju dolazi do relacija koje imaju samo skalarne atribute

$$R_{k..q} (X_{k..q})$$

$$F_{k..q} = \{ Y_{k..q} \rightarrow Z_{k..q} \}$$

Sam univerzalni postupak se sastoji u sledećem:

1. Primenom algoritma *KandidatKljucevi* za skup zavisnosti F određuje se skup kandidat-ključeva K za šemu R i bira jedan kao primarni ključ P.
2. Na osnovu datog skupa zavisnosti F i dobijenog skupa kandidat-ključeva vrši se normalizacija šeme R u željenu normalnu formu izdvajanjem svake zavisnosti $Y_u \rightarrow Z_u$ koja je narušava, pri čemu za svaku takvu zavisnosti nastaje nova šema relacije $R_{Nu}(\underline{Y_u}, Z_u)$ a iz šeme relacije R nestaje atribut Z_u.
3. Na osnovu datog skupa zavisnosti F i dobijenog skupa kandidat-ključeva vrši se normalizacija šeme R preostale posle koraka 2 u željenu normalnu formu izdvajanjem svake zavisnosti $Y_v \rightarrow R_v()$ koja je narušava, pri čemu za svaku takvu zavisnosti nastaje nova šema relacije $R_{Nv}(\underline{Y_v}, R_v())$ a iz šeme relacije R nestaje atribut-relacija $R_v()$.
4. Na osnovu datog skupa zavisnosti F i dobijenog skupa kandidat-ključeva vrši se normalizacija šeme R preostale posle koraka 3 u prvu normalnu formu izdvajanjem svake zavisnosti $P \rightarrow R_v()$ koja je narušava, pri čemu za svaku takvu zavisnosti nastaje nova šema relacije $R_{Nv}(\underline{P}, R_v())$ a iz šeme relacije R nestaje atribut-relacija $R_v()$.
5. Svaka šema relacije $R_{Nv}(\underline{P}, R_v())$ nastala koracima 3 i 4 prevodi se u prvu normalnu formu do strukture $(P, X_v, R_{v1}(), ... , R_{vj}(), ... , R_{vp}())$ pri čemu se odgovarajući skup zavisnosti $F_v = \{ Y_{vu} \rightarrow Z_{vu} \} \cup \{ Y_{vw} \rightarrow R_{vw}() \}$ transformiše dopunom levih strana lokalnih zavisnosti sa Y_v odnosno P.
6. Sekvenca postupaka 1 do 4 se ponavlja za svaku novonastalu šemu relacije sve do sekvence bez efekta.

U svemu ovome bitan je redosled kod koga se u šemi relacije prvo izvrši sva normalizacija na osnovu funkcijskih zavisnosti a tek onda normalizacija u prvu normalnu formu. U suprotnom, neophodna je primena postupaka normalizacije za četvrtu i petu normalnu formu.

Primer

Posmatrajmo šemu relacije sa atributom-slogom i atributima-relacijama

R (SIFN, NAZIVN, OBLAST<>, AUTOR(), POZAJMICA())
OBLAST < SIFO, NAZIV >
AUTOR (SIFA, IMEA, KOJI)
POZAJMICA (SIFC, TELEFON())
TELEFON (BROJ)

gde su odgovarajući skupovi netrivijalnih zavisnosti, na nivou naznačenih relacija

F_R = { SIFN $\rightarrow$ NAZIVN,OBLAST<>,AUTOR (),POZAJMICA () }
F_{OBLAST} = { SIFO $\rightarrow$ NAZIVO, NAZIVO$\rightarrow$SIFO }
F_{AUTOR} = { SIFA $\gamma\rightarrow$ IMEA, SIFA $\rightarrow$ KOJI }
$F_{POZAJMICA}$ = { SIFC $\gamma\rightarrow$ TELEFON() }
$F_{TELEFON}$ = {}

Kao prvo, ravojem jedinog atributa-sloga dobijamo šemu relacije

R (SIFN, NAZIVN, SIFO, NAZIVO, AUTOR(), POZAJMICA ())

i objedinjeni skup zavisnosti za tu šemu

FR={SIFN→NAZIVN,SIFO,NAZIVO,AUTOR(),POZAJMICA(), SIFO→NAZIVO, NAZIVO→SIFO}

Nakon toga, korakom 1 utvrđujemo da je jedini kandidat-ključ i ujedno primarni ključ šeme R atribut SIFN.

U okviru koraka 2 nad šemom R izdvajamo jedinu zavisnost koja narušava 3. a time i BC normalnu formu:

R (SIFN, NAZIVN, SIFO, NAZIVO, AUTOR(), POZAJMICA())

F_R={SIFN→NAZIVN,SIFO,NAZIVO,AUTOR(),POZAJMICA(), SIFO→NAZIVO, NAZIVO→SIFO }

SIFO→NAZIVO (ujedno i NAZIVO→SIFO) , narušava 3.NF

R1 (SIFO, NAZIVO)

F_{R1}={ SIFO→NAZIVO SIFO→NAZIVO }

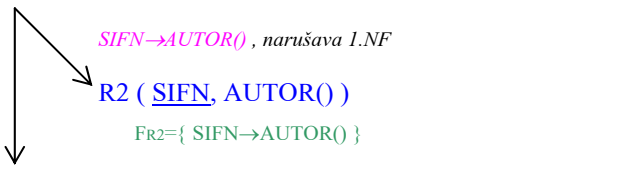
R (SIFN, NAZIVN, SIFO, AUTOR(), POZAJMICA())

F_R = { SIFN→NAZIVN,SIFO,AUTOR(),POZAJMICA() }

Ovim smo od polaznog R dobili dekompoziciju {R, R_1}. Korak 3 u ovom konkretnom slučaju nema efekata, dok korak 4 dovodi do izdvajanja svih zavisnosti oblika $P \rightarrow R_v()$ u posebne šeme sa atributima-relacijama:

R (SIFN, NAZIVN, SIFO, AUTOR(), POZAJMICA())

F_R={SIFN→NAZIVN,SIFO,AUTOR(),POZAJMICA() }



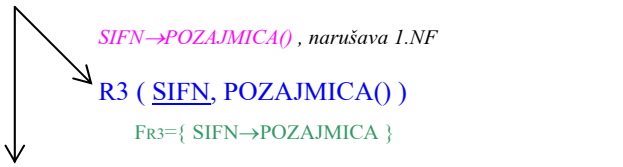
SIFN→AUTOR() , narušava 1.NF

R2 (SIFN, AUTOR())

F_{R2}={ SIFN→AUTOR() }

R (SIFN, NAZIVN, SIFO, POZAJMICA())

F_R = { SIFN→NAZIVN,SIFO,POZAJMICA() }



SIFN→POZAJMICA() , narušava 1.NF

R3 (SIFN, POZAJMICA())

F_{R3}={ SIFN→POZAJMICA }

R (SIFN, NAZIVN, SIFO)

F_R = { SIFN→NAZIVN,SIFO }

Sada imamo dekompoziciju {R, R1, R2, R3}. Preostaje još da u okviru koraka 5 šeme R2 i R3 prevedemo u prvu normalnu formu:

R2 (SIFN, AUTOR()) == R2 (SIFN, SIFA, IMEA, KOJI)

FR2 = { SIFA → IMEA, SIFN,SIFA → KOJI }

R3 (SIFN, POZAJMICA) == R3 (SIFN, SIFC, TELEFON())

FR3 = { SIFC → TELEFON() }

Posle prvog prolaza kroz sve korake dobijena dekompozicija je {R, R1, R2, R3}. Sada preostaje da prethodno izvedeni prolaz ponovimo za svaku šemu relacije iz dekompozicije. Kao prvo određujemo skupove kandidat-ključeva svih šema:

K = { SIFN } K1 = { SIFO, NAZIVO } K2 = { SIFA,SIFN } K3 = { SIFN,SIFC }

Na osnovu toga i odgovarajućih skupova zavisnosti uočavamo:
- šeme R i R1 su u prvoj i BC normalnoj formi i kao takve su konačne;
- šema R2 jeste u prvoj ali nije u BC normalnoj formi;
- šema R3 nije ni u prvoj ni u BC normalnoj formi.

Ponavljanjem koraka 2 do 5 (treba samo 2) za šemu R2 dobijamo:

R2 (SIFN,SIFA, IMEA, KOJI)

FR2 = { SIFA → IMEA, SIFN,SIFA → KOJI }

SIFA→IMEA , narušava 2.NF

R21 (SIFA, IMEA)

FR21={ SIFA→IMEA }

R2 (SIFN,SIFA, KOJI)

FR2 = { SIFN,SIFA→KOJI }

Ponavljanje koraka 2 do 5 (treba 2, 4 i 5) daje:

R3 (SIFN,SIFC, TELEFON())

FR3 = { SIFC → TELEFON() }

SIFC→TELEFON() , narušava 1. i 2.NF

R31 (SIFC, TELEFON()) == **R31 (SIFC,BROJ)**

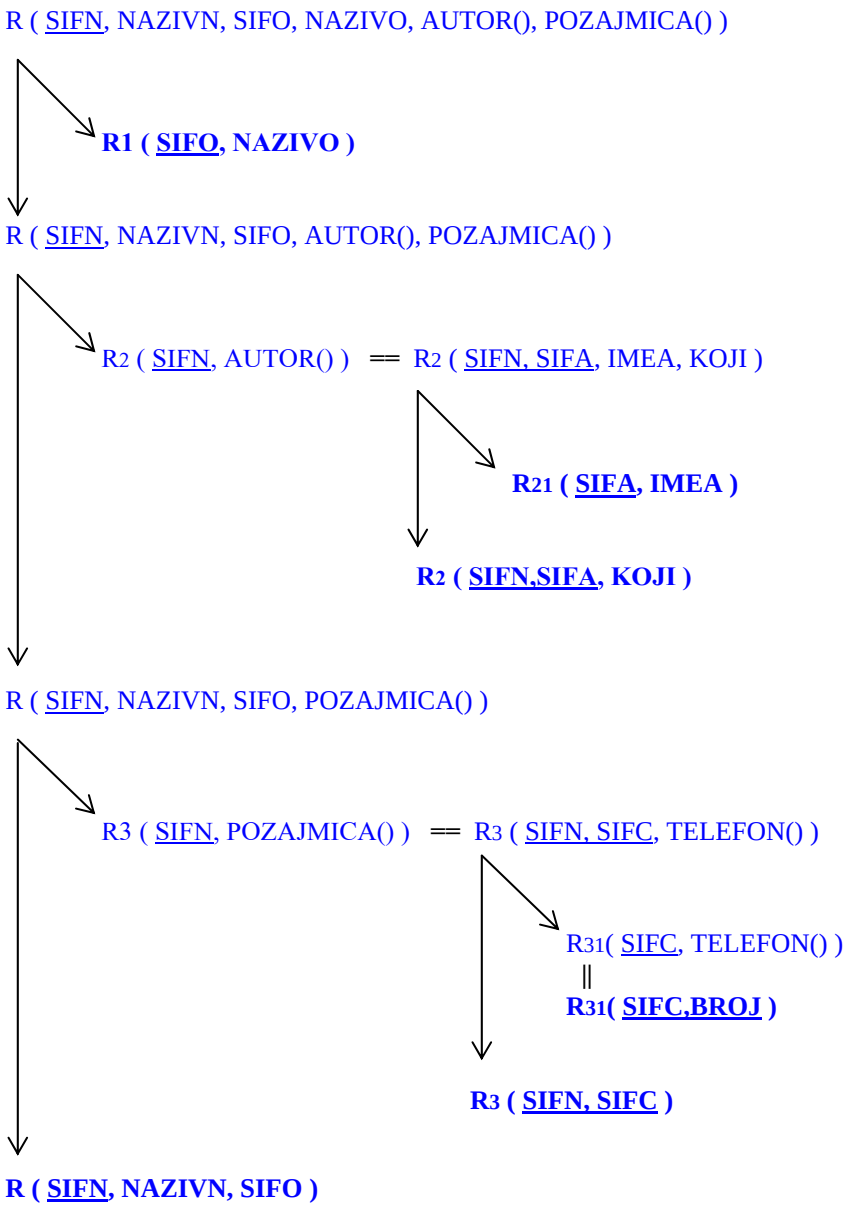
FR31={ SIFC→TELEFON() } FR31={}

R3 (SIFN,SIFC)

FR3 = {}

Kao rezultat svega ovoga dobili smo dekompoziciju {R, R_1, R_2, R_{21}, R_3, R_{31}}. Uvidom u skupove zavisnosti i skupove kandidat-ključeva zaključujemo da naredni prolazi nizu potrebni, pošto su sve šeme u dekompoziciji i u BC i u prvoj normalnoj formi.

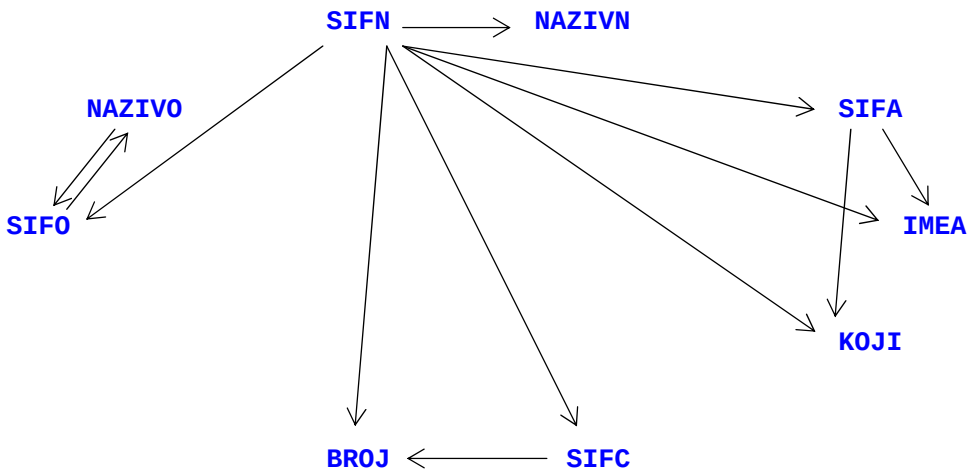
Na kraju ovog primera, prikažimo čitav postupak normalizacije i integralno:



Preostaje da razmotrimo još jedno pitanje u okviru univerzalnog pristupa problemu normalizacije. Kako bi bilo potpuno jasno o čemu se radi, konstruišimo za prethodni primer sa razvijenim atributima-relacijama graf zavisnosti na sledeći način:

- svaki atribut šeme relacije R postaje čvor;
- za svaku zavisnost is skupa F $X \rightarrow Y$ gde Y ili njegovi delovi mogu biti i relacije od svakog atributa sadržanog u X povlačimo poteg ka svakom atributu sadržanom u Y.

Navedenim postupkom dobijamo za naš primer



Na osnovu ovog dijagrama uočavamo da su svi čvorovi obuhvaćeni jednim grafom zavisnosti, odnosno da ne postoje izolovani podgrafovi i čvorovi. U tom smislu, "univerzalna" polazna relacija R je imala opravdanja i mogla je odmah da se normalizuje, pošto su svi atributi u njenom sastavu bili nekako povezani, jednoznačno ili višeznačno, direktno ili indirektno.

Posmatrajmo sada slučaj izmenjene polazne šeme relacije R

R (SIFN, NAZIVN, SIFO, NAZIVO, (SIFA, IMEA, KOJI), (SIFC,(BROJ)))

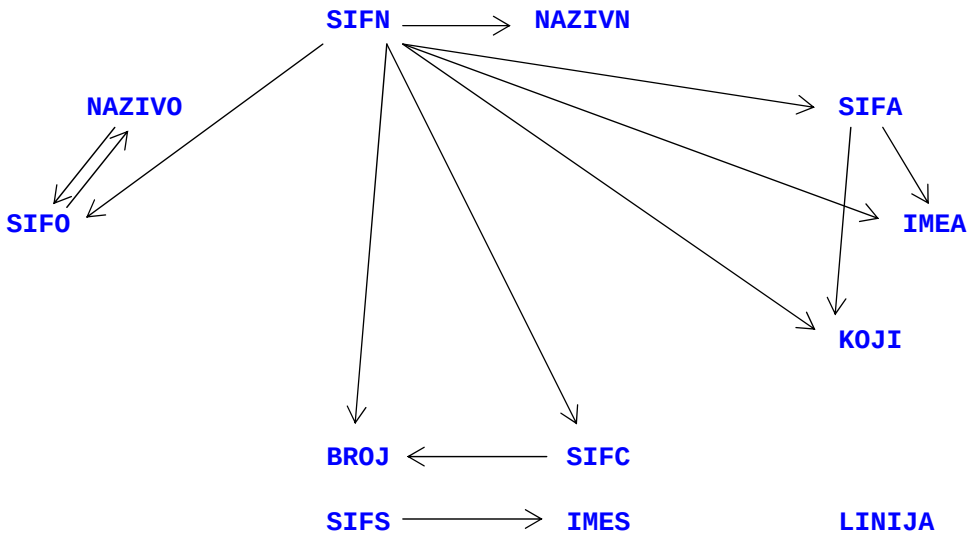
kojoj dodajemo još i atribute šifre škole, naziva škole i autobuske linije

SIFS, NAZIVS i LINIJA

čime "po prirodi stvari" dodajemo u F i zavisnost

$SIFS \rightarrow NAZIVS$

Ako sada za takav primer sastavimo graf zavisnosti dobijamo:



Graf zavisnosti sada ima tri izolovana podgrafa.

Nije teško zaključiti da u ovom slučaju sa polaznom šemom relacije R nešto nije u redu. Naime, jednom te istom šemom relacije obuhvatili smo podatke biblioteke, podatke o školama koje sa time nemaju nikakve veze i podatke o autobuskim linijama koje ni sa jednim od prethodnog nemaju nikakve veze. Posledica takve loše strukture su brojne anomalije. Primera radi, pošto bi u praksi sigurno bilo najviše naslova pa autobuskih linija pa škola, višak redova u odnosu na autobuske linije i škole morali bi da popunimo NULL-vrednostima. U slučaju da umesto autobuskih linija imamo autobuse moglo bi se dogoditi da vremenom njihov broj postane jednak broju naslova i od tog trenutka ne bi mogli da evidentiramo ni jedan novi autobus pošto bi time narušili integritet kandidat-ključa SIFN (ili bi za naslove morali da unesemo NULL-vrednosti ili bi morali da ponovimo vrednosti za neki naslov).

Šta možemo da zaključimo iz prethodnog razmatranja? Kao prvo, u jednu šemu relacije ne treba veštački stavljati podatke koji nemaju nikakve uzajamne veze. Kao drugo, kod normalizacije neke šeme relacije neophodno je prvo primeniti postupak tzv. "prednormalizacije" kojim će se polazna šema relacije na osnovu grafa zavisnosti dekomponovati na odgovarajući broj šema relacije. Kod primera kojeg smo upravo razmatrali, dobili bi tri šeme relacije:

R1 (SIFN, NAZIVN, SIFO, NAZIVO, (SIFA, IMEA, KOJI), (SIFC,(BROJ)))

R2 (SIFS, NAZIVS) R3 (LINIJA)