Logicko projektovanje BP

Friday, February 16, 2018 10:16 AM

Logicko projektovanje baze podataka

- ER model je predstavljao takozvani semanticki pristup
  - Realni svijet smo opisivali pomocu nekih znacenja, veza, objekata(entiteta)
  - Sve ovo nije bilo matematicki precizno vec je bilo ukljuceno dosta 'slobode' prilikom projektovanja baze bez neke mogucnosti matematickog provjeravanja valjanosti baze

- Odnosi medju atributima se zovu zavisnosti
  - Mozemo reci i znanja o atributima
  - Ovo su odnosi tipa: 'Ako je ovaj atribut ovo onda je onaj ono'
  - Ovo u principu znaci da se odredjuje neki 'poredak' u smislu medjusobnog uticanja atributa jedni na druge

Tipovi zavisnosti

1) Funkcionalne
2) Viseznacne
3) Zavisnosti Spajanja
4) …

Motivacioni Primjer

- Data je relaciona sema:
  - Snadbijevanje(ime_s, adresa, proizvod, cijena)
    - PK : {ime_s, proizvod}
- Takodje, data je instanca ove seme u oznaci *r* :

ime	adresa	proizvod	cijena
s1	a1	p1	3.4
s1	a1	p2	4.3
s2	a2	p3	7.1

- Uocimo nedostake (**anomalije**)
  - Nagomilavanje podataka
    - Ime i adresa snabdjevaca pojavljuju se onoliko puta koliko on proizvoda isporucuje
  - Update anomalija
    - Prilikom mijenjanja adrese ona se mora mijenjati u svakome redu
    - Potencijalna nesaglasnost podataka
  - Insert anomalija
    - Ne mozemo unijeti sanbdjevaca bez proizvoda!
    - Snabdjevac ne moze postojati ako za njega njema odgovarajuci proizvod
  - Delete anomalija
    - U slucaju da izbrisemo sve isporuke nekog snabdjevaca, automatski time i gubimo sve podatke o tome snabdjevacu

Dekompozicija seme

- Resenje je predstavljanje to jest dekompozicija pocetne seme na vise sema, u nasem slucaju jedno resenje bi bilo:
  - Snabdjevac(ime, adresa)
  - Isporuka(ime_s, proizvod, cijena)
- Pocetni skup atributa (atributa pocetne relacije) je unija atributa novonastalih relacija to jest njihovih atributa
- Postupak:
  - Relacinu semu *R* razbijamo na relacione seme *R1* i *R2*
  - Vazi: att(R) = att(R1) U att(R2)
    - Pri cemu je att(R) predstavlja atribute relacije R
  - Torke novonastalih relacije cine:

    $r1 = \pi_{R1}(r)$

    $r2 = \pi_{R2}(r)$

    - Torke prva novonastale relacije dobijaju se projetkovanjem na atribute prve relacione seme na pocetnu relaciju r
    - Slicno i za drugu novonastalu relaciju

snabdjevac			isporuka		
ime	adresa		ime_s	proizvod	cijena
s1	a1		s1	p1	3.4
s2	p2		s1	p2	4.3
			s2	p3	7.1

Postavljaju se pitanja

1) Zasto smo obavili dekompoziciju na ove dvije seme sa ovim atributima
2) Zasto su ovo primarni kljucevi
3) Da li se ovo moglo odraditi na drugi nacin
4) Koji od potencijalno vise nacina je bolji

Sledeci primjer

- Data je relaciona sema:
  - Imenik(ime, telefon, grad)

imenik		
ime	telefon	grad
a	t1	g1
a	t2	g2
b	t3	g1

- Obavljamo dekompoziciju na
  - R1(ime, telefon)
  - R2(ime, grad)

r1			r2	
ime	telefon		ime	grad
a	t1		a	g1
a	t2		a	g2
b	t3		b	g1

- S' obzirom da su se torke relacijama r1, r2 dobile projetkovanjem na odgovarajuce atribute njihovih relacionih sema, pocetnu relaciju bi trebalo dobiti prirodnim spajanjem relacija r1 i r2
- Prirodnim spajanjem relacija r1 i r2 dobijamo:

imenik		
ime	telefon	grad
a	t1	g1
a	t1	g2
a	t2	g1
a	t2	g2
b	t3	g1

- Vidimo da ova relacije nije ista kao i pocetna relacija imenik

Ovakva pojava, pojavljivanje ili gubitak torki prilikom dekompozicije naziva se 'Gubljenje informacija pri dekompoziciji'

Teorija Funkcionalnih Zavisnosti

Friday, February 16, 2018 11:07 AM

Definicija

- Neka su $X=\{x_1, \dots, x_n\}$ i $Y=\{y_1, \dots, y_n\}$ neprazni podskupovi skupa atributa relacione seme R ($X, Y \subseteq attr(R)$)
- Kazemo da u relaciji r(R) izmedju skupova X i Y postoji *funkcionalna zavisnost* u oznaci X -> Y (x odredjuje y) ako za svake 2 torke t1 i t2 ∈ r vazi ako je t1[x] = t2[x] => t1[y] = t2[y]
- U *r* postoji takozvano X -> Y ako vazi : kada su torke iste na X tada su one iste i na Y
- Tada:
  - Kazemo da su vrijednosti Y odredjene sa vrijednostima X
  - Kazemo da X odredjuje Y
  - Kazemo da X funkcionalno odredjuje Y
  - Kazemo da Y funkcionalno zavisi od X

Alternativna definicija

- Uslov X -> Y ekvivalentan je uslovu
- $\forall p \in Dom(X)$ skupa $\pi_y(\delta_{x=p}(r))$ je najvise jednoclan skup, gdje je $p=(p_1,\dots,p_k)$ a $Dom(X) = Dom(X_1 \times X_2 \times \dots X_k)$
- $\pi_y(\delta_{x=p}(r))$
  - Ovo mora dati tacno jednu torku ili prazan skup
  - Ovo u principu znaci da ako se u relaciji nalazi torka sa skupom atributa *p* tada ovaj izraz daje jednu torku (upravo tu istu torku), u suprotnom ovaj izraz daje prazan skup
    - Ovo direktno znaci da ako je torka ista na nekim atributima p onda je ona ista i na atributima y

Opstije

- Funkcionalna zavisnot je skup ogranicenja izmedju dva atirbuta u relaciji

- Definicija kaze da ako su tvije torke iste na atributima $A_1, A_2, \ldots, A_n$ onda te torke moraju biti iste na atributima $B_1, B_2, \ldots, B_n$
- Kazemo da funkcionalna zavisnot X -> Y vazi na semi R ako ona vazi na svakoj instanci te seme
    - Postojanje Funkcionalne Zavisnosti ogranicava skup torki koje se pojavljuju u toj relaciji

Primjer

Imenik(ime, adresa, grad, postanski_br, telefon)

Neke funkcionalne zavisnoti:

- ime -> adresa
- ime -> grad
- ime -> {adresa, grad}
- {adresa, grad} -> postanski_br
- {postanski_br} -> grad

Neke netacne funkcionalne zavisnoti:

- adresa -> grad
- grad -> postanski_broj

U opstem slucaju komutativnost ne vazi

X -> Y =!> Y -> X

Kljuc relacije

- Data je relaciona sema $R(A_1,\ldots,A_n)$
- Skup atributa K je kljuc relacione seme $R(A_1,\ldots,A_n)$ ako $K \rightarrow A_1,\ldots,A_n$
- Kljuc je skup atributa koji funkcionalno odredjuje torku

Trivijalna zavisnost

- Zavisnost X -> Y je **trivijalna** ako je $Y \subseteq X$

Aksiomi izvodjenja

Friday, February 16, 2018 11:48 AM

Ako imamo skup atributa na nekoj semi R, ukupno postoji $2^n - 1$ nepraznih podskupova tih atributa

- Bilo bi neprakticno da za svaki od ovih podskupova definisemo neku zavisnost

Bolji nacin

- Krecemo od nekog skupa funkcionalnih zavisnosti koji oznacavamo sa F
- Skup F^+ predstavlja zatvorenje skupa funkcionalnih zavisnosti
  - Ovaj skup sadrzi sve zavisnosti koje mogu da se izvedu iz skupa F

Armstrongovi aksiomi

- Pravila za izvodjenje novih funkcionalnih zavisnosti iz postojecih funkcionalnih zavisnosti
- Zovu se aksiomi jer se koriste mehanicki, a u stvari se radi o tvrdjenjima koja se dokazuju iz definicije funkcionalne zavisnosti i relacione algebre

Pravilo 1

Refleksivnost : $X \rightarrow X$

Pravilo 2

Prosirenje : $X \rightarrow Y \Rightarrow XZ \rightarrow ZY$ (Y ?)

Pravilo 3

Aditivnost : $X \rightarrow Y$ && $X \rightarrow Z \Rightarrow X \rightarrow YZ$

Pravilo 4

Projektivnost: $X \rightarrow YZ \Rightarrow X \rightarrow Y$ && $X \rightarrow Z$

Pravilo 5

Tranzitivnost: $X \to Y$ && $Y \to Z$ => $X \to Z$

Pravilo 6

Pseudo Tranzitivnost: $X \to Y$ && $YZ \to W$ => $XZ \to W$

Dokaz Aksioma

1. Dokaz refleksivnosti: X -> X
   - Uzmimo proizvoljno *p* iz Dom(X)
   - $\pi_x(\delta_{x=p}(r))$ je najvise jednoclan skup, to jest ili je rezultat *p* ili prazan skup
     - Prvo radimo selekciju po skupu atributa p (taj skup je podskup Dom(X)) a zatim radimo projekciju na sam skup atributa X
   - Odavde slijedi da X -> X
2. Dokaz prosirenja : X->Y => XZ -> Y
   - Uzmimo prizvoljno *p* iz Dom(X) i uzmimo proizvoljno *u* iz Dom(Z)
   - $\pi_y(\delta_{xz=pu}(r))$
     - Treba dokazati da je ovo najvise jednoclan skup
   - $\delta_{xz=p}(r) = \delta_{x=p\ \&\&\ z=u}(r) \subseteq \delta_{x=p}(r)$
     - Selekciju u zagradi mozemo zapisati kao selekciju po atributima p i selekciju po atributima u
     - Iz ovoga zakljucujemo da je selekcija po predikatu x=p && z=u podskup selekcije po predikatu x=p
       - Jednostavno dodavanjem uslova && z=u moze samo da izbaci torke koje bi se nasle u rezultatu samo uslovom x=p, ne moze dodati nove torke
   - $\pi_y(\delta_{xz=pu}(r)) \subseteq \pi_y(\delta_{x=p}(r))$
     - Pod pretpostavkom da je prva selekcija podskup druge, tada ce i njena projekcija biti podskup
   - Pod pretpostavkom da X -> Y vazi

- $\pi_y(\delta_{x=p}(r))$ je najvise jednoclan skup
- Onda je svaki podskup ovog skupa takodje najvise jednoclan skup cime smo dokazali ovo pravilo

3. Dokaz aditivnosti: X -> Y && X -> Z => X -> YZ
   - Pretpostavimo suprotno da ne vazi X -> YZ
   - Iz ovoga slijedi da $\pi_{yz}(\delta_{x=p}(r))$ ima vise od jedne troke za neko p ∊ Dom(X)
   - Dalje slijedi da jedan od ova dva uslova je tacan
     - $\pi_y(\delta_{x=p}(r))$ ima vise od jedne torke
     - $\pi_z(\delta_{x=p}(r))$ ima vise od jedne torke
   - Ako je bilo koji od ova dva uslova tacan to nije saglasno sa nasom pretpostavkom da X -> Y i X -> Z cime smo dokazali da suprotno ne vazi
4. Dokaz projektivnosti: X -> YZ => X -> Y && X -> Z
   - $\pi_{yz}(\delta_{x=p}(r))$ je onda najvise jednoclan skup
   - $\pi_y(\pi_{yz}(\delta_{x=p}(r)))$ je po pravilu projekcije $\pi_y(\delta_{x=p}(r))$ (1)
   - $\pi_z(\pi_{yz}(\delta_{x=p}(r)))$ je po pravilu projekcije $\pi_z(\delta_{x=p}(r))$ (2)
   - Znamo da ova dva novodobijena skupa mogu biti najvise jednoclani te iz (1) slijedi X -> Y i iz (2) slijedi X -> Z
5. Dokaz Tranzitivnosti: X -> Y && Y -> Z => X -> Z
   - Uzmimo proizvoljno t_1, t_2 ∊ R takvo da je $t_1[x]=t_2[x]$
   - Iz X -> Y => $t_1[Y]=t_2[Y]$
   - Iz Y -> Z => $t_1[Z]=t_2[Z]$
   - Dakle, za t1, t2 ∊ R ako je $t_1[x]=t_2[x]$ => $t_1[z]=t_2[z]$
6. Dokaz Pseudo Tranzitivnosti: X -> Y && YZ -> W => XZ -> W
   - Dokazacemo koristeci prethodna svojsta to jest predhodna pravila
     (1) Z -> Z
       - iz P_1

     (2) XZ -> Z

- Iz P_2 i (1)

(3) X -> Y

- Po pretpostavci

(4) XZ -> Y

- Iz P_2 i (3) I (2)

(5) XZ -> YZ

- Iz P_3 i (4)

(6) YZ -> W

- Po pretpostavci

(7) XZ -> W

- Iz (5) i (6) koristenjem P_5

- Vidimo da u slucaju da je Z = ∅ dobijamo P_5 to jest pravilo tranzitivnosti
  - Zakljucujemo da je tranzitivnost (pravilo P_5) samo specijalni slucaj pseudo tranzitivnosti

Dokazano je da je skup $P_1 \ldots P_6$ potpun. Ovo znaci da se svaka funkcionalna

zavisnost moze izvesti iz pomocu ovog skupa, to jest koristenjem ovih aksioma

Aksiomi nisu medjusobno nezavisni

Skup $\{P_1, P_2, P_3\}$ je potpun i nezavistan

- Pomocu njega mozemo izvesti sve funkcionalne zavisnosti

Alternativna definicija Armstrongovih aksioma

- U nekim knjigama date su sledece definicije prva tri aksioma
  1. P_1 : Refleksivnost : $Y \subseteq X$ => X -> Y
  2. P_2 : Prosirenje : X -> Y => XZ -> YZ
  3. P_3 : Tranzitivnost : X -> Y && Y -> Z => X -> Z

Primjeri

- Neka je dato XY->Z i X -> Y, dokazati X -> Z
    (1) X -> X
        - Iz P_1
    (2) X -> Y
        - Po pretpostavci
    (3) X -> XY
        - Iz P_3 i (1) i (2)
    (4) XY -> Z
        - Po pretpostavci
    (5) X -> Z
        - Iz P_5 i (3) i (4)

Zatvorenje Skupa FZ

Thursday, February 22, 2018 5:54 PM

Zatvorenje skupa funkcionalnih zavisnosti F u oznaci F^+ je skup svih funkcionalnih zavisnosti koji mogu da se izvedu iz F koristenjem Armstrongovih aksioma

Primjer:

- Ako je F = {A→B} sta je F^+?
  - F^+ = { X→Y, A→B, B→B, A→AB, AB→A, AB→B }
    - Od ovih 6 zavisnosti 2 su trivijalne (AB→A i AB→B)

Vidimo da cak i za skup sa jednim elementom, zatvorenje toga skupa ima cak 6 elemenata

Racunanje zatvorenja nekog skupa funkcionalnih zavisnosti je neisplativ posao

- Ovdje se misli na trazenje svih mogucih kombinacija elemenata toga skupa

Zatvorenje skupa atributa u odnosu na F

- Definicija:
  - Neka su X, Y neprazni podskupovi skupa atributa relacione seme R
    - X, Y ⊆ attr(R), X, Y != ∅
  - Neka je F skup funkcionalnih zavisnosti na R
  - Kazemo da je Y zatvorenje skupa X u odnosu na F ako je Y maksimalan skup atributa takav da se X→Y izvodi iz F
  - Oznaka za zatvorenje skupa atributa X (sve su ekvivalentne):
    - X_F^+
    - x^+
    - (X, F)
- Primjer

- Naci A_F^x ako je F={A→B, B→C, B→D}
  - A→A, A→B => A→AB
  - A→B && B→C => A→C
  - A→B && B→D => A→D
  - A→ABCD
- $A_F^x = \{ABCD\}$
  - U ovom primjeru mozemo zakljuciti da je A kljuc!

Algoritam za nalazenje zatvorenja skupa atributa X

- Opis algoritma:
  - Ulaz:
    - X - skup atributa
    - F - skup funkcionalnih zavisnosti
  - Izlaz:

    A_F^x − zatvorenje skupa atributa X u odnosu na F
- Pseudo kod na nestrogom paskalu:

```
begin
    A_F^x = X
    stariSkup = ∅
    while A_F^x != stariSkup do
        stariSkup= A_F^x
        foreach funkcionalnu_zavisnost W→Z iz F do
            if W ⊆ A_F^x then
                A_F^x = A_F^x + Z
end
```

- Objasnjenje
  - Na pocetku rezultat je jednak samo pocetnom skupu atributa a predhodni skup je prazan skup
  - Dokle god je predhodni razlicit od rezultujuceg skupa petlja ce se izvrsavati
  - Pri svakoj iteraciji petlje prolazimo kroz dati skup funkcionalnih zavisnosti
    - Ako posmatramo funkcionalnu zavisnost W→Z koja pripada Z mozemo zakljuciti:

- ▫ Ako se W nalazi u dosadanjem rezultujucem skupu onda treba dodati desnu stranu u rezultujuci skup (dodati Z)
- ▫ Ovaj uslov provjerava if dio unutar foreach petlje

- Postavlja se pitanje: da li vazi X→Y to jest da li X→Y ϵ F^+
- Da li mozemo odrediti da li funcionalna zavisnost X→Y pripada zatvorenju F^+
    - Da bi vazilo X→Y mora da vazi da Y ⊆ A_F^x
        - Znaci da bi X funkcionalno odredjivalo Y, Y mora da bude podskup zatvorenja skupa atributa X
        - Odnosno, Y moze da se izvede iz X

Ako se X→Y moze izvesti iz F, to jest ako X→Y ϵ F^+ kazemo da je X→Y logicka posledica F ili da se X→Y izvodi iz F

- Oznacavamo: F |= X→Y

Algoritam Clan

- Opis algoritma:
    - Ulaz:
        - X→Z - funkcionalna zavisnost
        - F - skup funkcionalnih zavisnosti
    - Izlaz:
        - True ako F |= X→Y to jest ako X→Y ϵ F^+
        - False inace
- Pseudo kod na nestrogom paskalu:

```
begin
    if(Y ⊆ zatvorenje(X, F)) then
        return true
    else
        return false
end
```

Pokrivanje Skupa FZ

Friday, February 23, 2018 11:11 AM

F_1 = {A→B, B→C, A→C, AB→C, A→BC}
F_2 = {A→B, B→C}

$F_1^+ = F_2^+$ te slijedi da su ovi skupovi ekvivalentni

- Sve sto mozemo dobiti iz prvog, mozemo dobiti iz drugog

Dva skupa funkcionalnih zavisnosti F i F' su ekvivalentni u oznaci F'≡F ako $F^+ = F'^+$

- Ako su dva skupa funkcionalnih zavisnosti ekvivalentni i ako F nije vece od F' u smislu kardinalnosti onda kazemo da F pokriva F'
- Jednostavnije, ako je F manji skup a njegovo zatvorenje je isto kao i zatvorenje F', kazemo da F pokriva F'
    - Primjer gore, F_2 pokriva F_1
- U nekim definicijama se ne pominje kardinalnost, vec se kaze da ako su ekvivalentni onda se oni pokrivaju

Ako za svako |= X→Y iz G^+ (to jest za svako X→Y za koje G |= X→Y) vazi u F |= X→Y kaze se da se skup G izvodi iz F odnosno da je logicka posledica F i oznacava se F |= G

- $G^+ \subseteq F^+$

Algoritam *Izvodi*

- Opis algoritma:
    - Ulaz:
        - G, F skupovi funkcionalnih zavisnosti
    - Izlaz:
        - True ako F |= G ($G^+ \subseteq F^+$)
        - False inace
- Pseudo kod algoritma na nestrogom paskalu

    begin

        Rezultat = true

```
        foreach X→Y iz G do
            if (clan(X→Y, F))=false then
                Rezultat = false
        return rezultat
    end
```

- Lako se pokazuje da:
    - F≡G ako i samo ako F |= G i G |= F

Algoritam Ekvivalentni

- Opis algoritma:
    - Ulaz:
        - F, G skupovi funkcionalnih zavisnosti
    - Izlaz:
        - True ako je F≡G
        - False inace
- Pseudo kod algoritma na nestrogom paskalu

```
    begin
        if(izvodi(F,G) and izvodi(G, F) then
            return true
        else
            return false
    end
```

- Primjer algoritma:
    - Polazni skupovi
        - F = {A→BC, A→D, CD→E}
        - G = {A→BCE, A→ABD, CD→E}
        - H = {A→BCDE}
    - Pitanje
        - F≡G
    - Koraci
        - Izvodi(F, G) - Provjeravamo da li se F moze izvesti iz G
            - Clan(A→BCE, F)
                - BCE ⊆ A_F^+
                - Gledamo da li u skupu F, iz A mozemo izvesti BCE

- Clan(A→ABD, F)
  - ABD ⊆ A_F^+
  - U skupu zavisnosti F, iz A moze se izvesti ABD
- Clan(CD→E, F)
  - E ⊆ A_F^+
  - U skupu zavisnosti F, iz CD moze se izvesti E
- Sva tri poziva funkcije clan vracaju true, te sada znamo da F |= G

- Izvodi(G, F) - Provjeravamo da li se G moze izvesti iz F
  - Clan(A→BC, G)
    - BC ⊆ A_G^+
    - Gledamo da li u skupu zavisnosti G, iz A moze da se izvede BC
  - Clan(A→D, G)
    - D ⊆ A_G^+
    - Gledamo da li u skupu zavisnosti G, iz A moze da se izvede D
  - Clan(CD→E, G)
    - E ⊆ CD_G^+
    - Gledamo da li u skupu zavisnosti G, iz CD moze da se izvede E
  - Sva tri poziva funkcije clan vracaju true, te sada znamo da G |= F
- Kako su oba poziva funkcije izvodi vratila true, zakljucujemo da F≡G

- Pitanje
  - F≡H
- Koraci
  - Izvodi(F, H) - Provjeravamo da li se F moze izvesti iz H
    - Clan(A→BCDE, F)
      - BCDE ⊆ A_F^+
      - Gledamo da li u skupu zavisnosti F, iz A moze da se dobije BCDE
  - Izvodi(H, F) - Provjeravamo da li se H moze izvesti iz F
    - Clan(A→BC, H)

- BC ⊆ A_H^+
- Gledamo da li u skupu zavisnosti H, iz A mozemo izvesti BC

- Clan(A→D, H)
  - D ⊆ A_H^+
  - Gledamo da li u skupu H, iz A moze da se izvede D
- Clan(CD→E, H)
  - E ⊆ CD_H^+
  - Gledamo da li u skupu H, iz CD moze da se izvede E
- Prva dva poziva funkcije clan vracaju true, dok poslednji (CD→E, H) vraca false, jer se iz skupa H, to jest iz CD nikako ne moze izvesti E

- Kako je drugi poziv funkcije Izvodi vratio false, zakljucujemo da F!≡H

Neredudantnost pokrivanja skupa funkcionalnih zavisnosti

Skup funkcionalnih zavisnosti je neredudantan ako ne sadrzi pravi podskup takav da je F'≡F

Skup funkcionalnih zavisnosti neredudantno pokriva skup zavisnosti G ako je F pokrivanje skupa G i ako je F neredudantan

Primjer

- Dati su skupovi
  - F_1={AB→C, A→B, B→C, A→C}
  - F_2={AB→C, A→B}
  - F_3={A→B, B→C}
- Moze se dokazati da je F_1≡F_2 i F_2≡F_3
- U ovom slucaju F_3 je neredudantno pokrivanje skupa F_2

Funkcionalna zavisnost X→Y je suvisna u skupu funkcionalnih zavisnosti F ako F \ {X→Y} |= X→Y

- Zavisnost je suvisna u skupu, ako taj skup bez te zavisnosti, ponovo moze izvesti istu tu zavisnost

Alternativna definicija neredudantnosti:

- Skup je neredudantan ako u njemu nema suvisnih funkcionalnih zavisnosti

Algoritam za neredudantnost skupa

- Opis algoritma:
  - Ulaz
    - F - skup funkcionalnih zavisnosti
  - Izlaz
    - True ako je F neredudantan
    - False inace
- Pseudo kod algoritma na nestrogom paskalu:

```
begin
    rezultat = true
    foreach X→Y do
        if(clan(X→Y, F - X→Y)) then
            rezultat = false
    return rezultat
end
```

Algoritam za pronalazenje neredudantnog pokrivanja

- Opis algoritma
  - Ulaz
    - G - skup funkcionalnih zavisnosti
  - Izlaz
    - F - neredudantno pokrivanje od G
- Pseudo kod algoritma na nestrogom paskalu

```
begin
    F = G
    foreach X→Y iz G do
        if(clan(X→Y, F - X→Y)) then
            F = F - X→Y
    return F
end
```

- Primjer
  - Ulaz
    - G = {A→B, B→A, B→C, A→C}

- Izlaz
  - F = {A→B, B→A, A→C}
- F je neredudantno pokrivanje skupa G
- Primijetimo da ako G zapisemo drugacije na primjer:
  - G = {A→B, A→C, B→A, B→C}
  - F = {A→B, B→A, B→C}
- Dobijamo razlicite rezultate u odnosu na redosled ispitivanja!

- Moze da postoji vise od 1 neredudantnog pokrivanja
- Ovaj algoritam trazi neredudantna pokrivanja koja su podskup ulaznog skupa
  - Naravno, moze postojati neredudantno pokrivanje koje nije podskup ulaznog skupa

Minimalno pokrivanje skupa funkcionalnih zavisnosti je neredudantno pokrivanje koje ima minimalan broj elemeneta

- Moze ih biti vise
- Konstruisan je algoritam koji nalazi minimalno pokrivanje
  - Pripada kategoriji NP kompletnih problema!

Kanonicko Pokrivanj Skupa FZ

Thursday, March 1, 2018 6:03 PM

Neka je F skup funkcionalnih zavisnosti

Atribut A je suvisan (na lijevoj strani) funkcionalne zavisnosti $X \rightarrow Y$ gdje je $X=AZ$: $X!=Z$. Ako vazi $\{F \backslash \{X \rightarrow Y\}\} \cup \{Z \rightarrow Y\} \equiv F$ pri cemu je $Z=X-A$

- Primjer $ABC \rightarrow PQ$
  - Ako $AC \rightarrow PQ$ onda je B suvisan

Kanonicko pokrivanje skupa funkcionalnih zavisnosti G je neredudantno pokrivanje F sa sledecim svojstvima

- Sve zavisnosti u skupu F su oblika $X \rightarrow Y$, gdje je Y jednoclani skup
- Lijeva strana svake funkcionalne zavisnosti ne sadrzi suvisne atribute

Primjer:

$G = \{A \rightarrow BCE, AB \rightarrow DE, BI \rightarrow J\}$

Njegovo kanonicko pokrivanje je $F=\{A \rightarrow B, A \rightarrow C, A \rightarrow D, A \rightarrow E, BI \rightarrow J\}$

Postupak Kanonickog pokrivanja

1) Svaku funkcionalnu zavisnost $X \rightarrow Y$ iz F zamijeniti sa skupom zavisnosti $\{X \rightarrow A \mid A_i \in Y\}$
   - Rastavljanje zavisnosti tako da desna strana bude jednoclan skup
2) Iz svake dobijene funkcionalne zavisnosti uklanjaju se suvisni atributi
   - Za svaku funkcionalnu zavisnost $X \rightarrow Y$ iz F i za svaki $A \in X$ provjeriti da li je *clan(X-A→Y, F)*
     - Uklanjanje suvisnih atributa iz lijeve strane
     - Pitamo se da li se Y moze izvesti iz skupa atributa koji se dobije kada od X (lijeve strane) oduzmemo A (atribut za koji provjeravamo da li je suvisan)
3) Odrediti neredudantno pokrivanje prethodno dobijenog skupa koriscenjem algoritma za neredudantno pokrivanje

Primjer:

$F=\{A \rightarrow C, AC \rightarrow J, AB \rightarrow DE, AB \rightarrow CDI, C \rightarrow M, A \rightarrow M\}$

Optimalno pokrivanje

Skup funkcionalnih zavisnosti je optimalan ako ne postoji ekvivalentan skup funkcionalnih zavisnosti koji ima manje simbola atributa u donosu na F

- Pri razmatranju ovih kriterijuma, pored pojma suvisnosti na lijevoj strani razmatra se pojam suvisnosti na desnoj strani

Ako X→Y i A ∈ Y, kazem da je A suvisan na desno strani zavisnosti, odnosno suvisan u Y ako vazi:

- {F\X→Y} U {X→Z} ≡ F gdje je Z=Y-A
- Y = AZ, Z!=Y
  - Y se sastoji od atributa A (koji je suvisan) i od atributa Z
  - A je suvisan ako skup FZ bez date zavisnosti X→Y zajedno sa zavisnoscu X→Z (zavisnosti kada se iz Y ukloni atribut A) daje ponovo skup zavisnosti F

Primjer:

F = {A→BC, B→C, AB→D}

Da li postoje suvisni atributi na lijevoj strani?

- B U AB→D

Da li postoje suvisni atributi na desnoj strani?

C U A→BC

Dokazano je da je optimalno pokrivanje istovremeno i minimalno pokrivanje bez suvisnih atributa

Algoritam za nalazenje optimalnog pokrivanja mogao i se napraviti iz algoritma za nalazenje minimalnog pokrivanja iz koga se izbace suvisni atributi

Kako je algoritam za nalazenje minimalnog pokrivanja NP kompletan, tako bi i ovaj algoritam bio NP kompletan

Losless Join Decompozition

Friday, March 2, 2018 12:34 PM

Neka je data relaciona sema $R(A_1, \ldots, A_n)$

Njena dekompozicija je svaka kolekcija $\Phi(R_1, \ldots, R_s)$ takvih da je $attr(R) = attr(R_1) \cup \ldots \cup attr(R_s)$

- Manje formalno : $R = R_1 \cup \ldots \cup R_s$

Za svaku relaciju vazi da je:

- $Ri = \pi_{Ri}(r)$

Ono sto mi ocekujemo je da cemo svaku dekompoziciju moci 'vratiti' nazad pomocu prirodnog spajanja

- $R = R_1 \bowtie R_2 \bowtie \ldots \bowtie R_s$
- $\bowtie_1^s \; Ri$

Ovo nazivamo dekompozicija koja cuva informacije (lossles join decompozition) ili dekompoziija bez gubitka pri spajanju

Pretpostavimo da na R vaze neke zavisnosti

- Na R_i vaze zavisnosti F_i takve da su to one iz F^+ u kojima su iskljucivo atributi iz R_i
- Vazi
  - $F \equiv F_1 \cup F_2 \cup \ldots \cup F_s$
  - $F^+ \equiv (F_1 \cup F_2 \cup \ldots \cup F_s)^+$

Ovo se zove dekompozicija koja cuva zavisnosti

Restauriranje prirodnim spajanje

Neka je attr(R) skup atributa relacione seme R

Ako u R za sva neprazna podskupa atributa $X,Y \subseteq attr(R)$ vazi funkcionalna zavisnost $X \rightarrow Y$ tada za svaku instancu r(R) ove seme vazi jednakost

$r = \pi_{xy}(r) \bowtie_{x=x} \pi_{xz}(r)$ gdje je Z=attr(R) - XY

Znaci da mozemo uraditi dekompoziciju po zavisnost X→Y na sledeci nacin:

- U prvu relaciju idu atiributi X→Y, dok u drugu ide atribut X i sve ostali atributi koji nisu X i Y
- Teorema nam govori da ako uradimo dekompoziciju na ovaj nacin, sigurno cemo moci da dobijemo pocetnu relaciju putem prirodnog spajanja bez ikakvog gubitka informacija

Dokaz teoreme o restauriranje

- Razlikujemo tri razlicita slucaja
  a) $Y \subseteq X$
  b) $Y \cap X = \emptyset$
  c) Opsti slucaj

a) $Y \subseteq X$ (predstavlja trivijalnu zavisnost)

- Z=attr(R) \ XY = attr(R) \ X
  - Jer je Y podskup od X
- XZ = attr(R)
  - Zajedno X i Z daju pocetni skuop atributa
- $\pi_{xz}(r) = r$
  - Slijedi da vazi $r = \pi_{xy}(r) \bowtie_{x=x} \pi_{xz}(r)$

b) $Y \cap X = \emptyset$

- $r = \pi_{xy}(r) \bowtie \pi_{xz}(r)$
- Cilj je dokazati da ove dvije projekcije daju pocetnu relaciju
- Radimo iz dva koraka, prvo dokazujemo da ako uzmemu proizvolju torku koja tima oblik t=xyz da ona pripada projekciji i u drugom smjeru, ako imo torku koja pripada projekciji da ona priada pocetnoj relaciji
  1) Uzmimo proizvoljnu torku $t \in r$
     - Torka ima oblik t = xyz
     - Iz ovoga slije da ce projekcija na atribute $x \in \pi_{xy}(r)$ i $xz \in \pi_{xz}(r)$
     - Iz ovoga slije da $xyz \in \pi_{xy}(r) \bowtie \pi_{xz}(r)$

2) Uzmimo da $t \in \pi_{xy}(r) \bowtie \pi_{xz}(r)$

- Neka t ima oblik t=xyz, kako se torka nalazi u gore navedenom izrazu (znaci po pretpostavci) iz toga slijedi da postoje 2 torke
  - $xy \in \pi_{xy}(r)$
  - $xz \in \pi_{xz}(r)$
- Ovo znaci da u r postoje torke oblika $t_1=xyz'$ i i $t_2=xy'z \in r$
- Imamo da vazi X→Y kako je $t_1[X]=t_2[2]$ => $t_1[Y]=t_2[Y]$ => y = y'
- Imamo $t_2=xyz \in r$ to jest $t_2=t \in r$

c) **Opsti slucaj**

- U ovom slucaju ne vazi Y podskup X i X i Y nisu disjunktni
  - Tada je presjek X i Y neprazan skup
- Oznacimo
  - $X \cap Y = Y_1$
  - $Y = Y_1Y_2$, Y != ∅
- Vazi
  - $X \to Y$
  - $X \to Y_1Y_2$
  - $X \to Y_2$ (po pravilu P4)
- Primijetimo da su X i Y_2 disjunktni te da za njih vazi slucaj b)
  - $r = \pi_{xy2}(r) \bowtie \pi_{xz}(r)$
    - $Z = attr(r) \setminus XY_2$
  - $XY_2 = XY$
- $r = \pi_{xy}(r) \bowtie \pi_{xz}(r)$ a znamo da je $Z=attr(r) \setminus XY$

Primjer

Data je relacija R(ABC) i skup funkcionalnih zavisosti F={A→B, B→C}

- Moguce dekompozicije bez gubitka pri spajanju:
  a) R1(AB) R2(AC)
  b) R1(BC) R2(BA)
- Sada moramo razmotriti jesmo li izgubili neko pravilo to jest

neku funkcionalnu zavisnost

a) $F_1=A \to B$, $F_2=A \to C$
   - Postavlja se pitanje da li je $F \equiv F_1 \cup F_2$
   - Vidimo da smo izgubili zavisnost $B \to C$, jer ne postoji nacin da iz ovog skupa zavisnosti dobijemo $B \to C$

b) $F_1=B \to C$, $F_2=A \to B$

c) Postavlja se pitanje da li je $F \equiv F_1 \cup F_2$
   - Vidimo da se iz ova dva skupa dobija ekvivalentan skup pocetnom
   - Zakljucujemo da je ova dekompozicija valjana jer nema gubitka prilikom spajanja i takodje cuva zavisnosti

Dependency Preserving Decomposition

Saturday, March 24, 2018 11:22 AM

Dekompozicija koja cuva zavisnosti

Algoritam za racunjanje skupa funkcionalnih zavisnosti koji vazi na novonastaloj relaciji

- **Ulaz**
  - Relacija R i njena dekompozicija R_1 koja je nastala projekcijom na skup atributa R_1 ($\pi_{Ri}(r)$)
  - Skup funkcionalnih zavisnost F koji vazi u R
- **Izlaz**
  - Skup funkcionalnih zavisnosti F_1 koji vazi u R_1
- **Pseudo kod na nestrogom Paskalu**

  Neka T bude skup funkcionalnih zavisnosti koji vazi u R_1
  - na pocetku T = Ø

  Za svaki skup atributa X koji je podskup R_1 naci zatvorenje X^+
  - zatvorenje se trazi u odnosu na pocetni skup zavisnosti F
  - ovo zatvorenje moze da sadrzi atribute koji ne pripadaju relacionoj semi R_1

  Dodajemo u T sve netrivijalne azavisnosti $X \rightarrow A$ takve da se A nalazi i u X^+ i u attr(R_1)

  Na kraju dobijamo skup funkcionalnih zavisnosti T koji vazi u R_1
  - T mozda nije minimalan skup funkcionalnih zavisnosti
  - Ovo resavamo nalazenjem kanonickog pokrivanja ovo skupa

Normalne Forme

Saturday, March 3, 2018 10:27 AM

Normalne forme su pravila po kojima atributi treba da budu grupisani u relacione seme da bi se izbjeglo nepozeljno ponasanje odnosno anomalije prilikom dekompozicije

- Pravila koja treba slijediti
  - Dekompozicija treba da bude bez gubitka pri spajanju
  - Dekompozicija treba da cuva zavisnosti
- Postoje i druge anomalije koje normalne forme pokusavaju da rijese

Normalne forme su sa jedne strane pravila, a sa druge to su algoritmi kako nesto treba raditi i sta ne treba raditi

Spisak normalnih formi

- Sledece 4 norme su zasnovane na pojmu funkcionalne zavisnosti:
  - 1NF
  - 2NF
  - 3NF
  - BCNF (Boyce Codd normal form)
- Postoje i druge normalne forme:
  - 4NF
    - Vezana za pojam viseznacne zavisnosti
  - 5NF (PJNF - Project join normal form)
    - Vezana za zavisnost spajanja
  - 6NF
    - Vezana za opste zavisnosti

1NF

Saturday, March 3, 2018 10:34 AM

Relacija je u 1NF ako su svi atributi prosti to jest ako su domeni svih atributa prosti i ako nema ponavljanja vrijednosti unutar jednog polja i jednog zapisa (torke)

		nenormalizovana relacija
ime	prezime	telefon
Marko	Markovic	69785432, 068554123
Nikola	Nikolic	069321453, 067122095, 068123444

		normalizovana relacija
ime	prezime	telefon
Marko	Markovic	69785432
Marko	Markovic	68554123
Nikola	Nikolic	68123444
Nikola	Nikolic	67122095
Nikola	Nikolic	69321453

Danas je ova forma sadrzana u definiciji relacije, to jest danas svaka relacija ima proste atribute

Neka odstupanja su pokusaji da se naprave slozeni atributi pomocu umetanja drugih tabela kao atributi (Oracle, Postgrec ..)

- Ovaj pristup se naziva RODBS (Relational object nested table)

2NF

Saturday, March 3, 2018 10:50 AM

Atribut koji nije dio kjuca mora biti cinjenica po cijelom kljucu a ne samo po nekom njegovom dijelu

- Ovo treba razmatrati u situacijama kada kljuc ima vise od jednog atribut

Primjer

Imamo relacionu semu R(S_NO, P_NO, City) i

- Na njoj definisan primarni kljuc {S_NO, P_NO}→City
- Kao i funkcionalnu zavisnost S_NO→City

S_NO	P_NO	City
s1	p1	PG
s2	p1	NK
s1	p2	PG
s1	p3	PG

- Vidimo da atribut 'City' nije informacija o cijelom kljucu vec informacija o njegovom dijelu
    - Atribut 'City' odnosi se na snadbjevaca a ne na par snadbjevac, proizvod

Sporedni atribut zavisi od dijela kljuca a ne od cijelog kljuca

- Za ovakvu relaciju vaze insert, delete, update anomalije

Resenje navedenih problema

- Moramo odraditi dekompoziciju po sporednom atributu koji zavisi od dijela kljuca (S_NO→City)
- Ako koristimo losless join teoremu dobijamo sledece dvije relacije
    - R_1(S_NO, City) R_2(S_NO, P_NO)
    - Moramo provjeriti koje funkcionalne zavisnosti vaze u nove relacije
        - F_1 = S_NO → City
        - F_2 = {S_NO, P_NO}→S_NO, {S_NO, P_NO}→P_NO
            - Trivijalne zavisnosti
        - Vidimo da iz F_1 i F_2 mozemo dobiti sve zavisnosti koje smo mogli dobiti iz pocetnog skupa zavisnosti

Za funkcionalnu zavisnost $X \rightarrow Y$ kazemo da je potpuna ako za svaki pravi podskup $X' \subset X$ vazi da je $X' ! \rightarrow Y$ (X' ne odredjuje Y)

- U principu znaci da cijeli skup atributa mora da odredjuje Y, a ne samo neki njegov dio
- Kazemo da Y potpuno funkcionalno zavisi od X
    - U suprotnom kazemo da je zavisnost parcijalna

Atribut A relacione seme R naziva se kljucnim atributom te seme ako ucestuvje u nekom o njenih kandidatskih kljuceva

- U suprotom kazemo da je atribut A sporedni

Relaciona sema je u 2NF ako svaki njen sporedni atribut potpuno zavisi od svakog njenog kandidatskog kljuca

- Ako ne postoji parcijalna zavisnost sporednog atributa od nekog kljuca

Teorema i algoritam 2NF normalizacije

Ako $R(A_1, \ldots, A_n)$ nije u 2NF tada postoji njena dekompozicija takva da su sve dobijene seme u 2NF i koja je bez gubitka pri spajanju i koja cuva zavisnosti

Ulaz

- $R(A_1, \ldots, A_n)$
- F : skup funkcionalnih zavisnosti

Izlaz

- Dekompozicija $\Phi(R_1, \ldots, R_s)$, R_i je u 2NF
    - Dekompozicija bez gubitka informacija
    - Dekompozicija cuva zavisnosti

i = 1;

R_1 = 1;

WHILE (R_i *nije u* 2NF) DO

- To jest dok postoji parcijalna zavisnosti sporednog atributa A_k od kljuca X

BEGIN

uocimo parcijalnu funkcionalnu zavisnost sporednog atributa A_k od kljuca X

vazi : $X = X_1X_2$ ($X_1 != X$ i $X_1 \rightarrow A_k$)

neka je $Z=attr(R_i)\backslash X_1A_k$

zamjenjujemo R_i sa $R[X_1A_k]$ i $R'[X_1Z]$

- *to jest $R_i=R[X_1A_k]$*

$i = i + 1$

$R_i = R[X_1Z]$

END

Objasnjenje algoritma

- Kada algoritam uoci parcijalnu zavisnost nekog atributa A_k od nekog kljuca X nastoji da razdvoji datu relaciju tako da ne postoji parcijalna zavisnost
- Kljuc X se moze podijeliti na dva diskunktna skupa atributa X_1 i X_2 tako da X_1 predstavlja sve atribute koji odredjuju A_k, dok X_2 predstavlja sve ostale atribute kljuca X
- Zatim se data relacija dijeli na dvije nove relacije sa sledecim atributima:
  - Relacija R_1 ima atribute X_1A_k
    - Znamo da skup atributa X_1 odredjuje potpuno atribut A_k
  - Relacija R_2 ima atribute X_1Z
    - Napomena : $Z = attr(R) - X_1A_k$
    - Skup atributa Z predstavlja sve atribute pocetne relacije koji nisu X_1(dio kljuca koji odredjuje A_k) i koji nisu A_k(atribut koji ima parcijalnu zavisnost od kljuca X)
    - Tako da nova relacija R_2 u sebi sadrzi pocetni dio kljuca koji je odredjivao sporedni atribut A_k i sve atribute pocetne relacije koji nisu A_k i dio pocetnog kljuca koji je odredjivao A_k

Primjer

Data je relacija R(S_NO, P_NO, City) kao i skup funkcionalnih zavisnosti F={S_NO→City, {S_NO,P_NO}R→City}

Nad datom relacijom mozemo odraditi dekompoziciju tako da obje novonastale relacije budu u 2NF

- R_1(S_NO, City) R_2(S_NO, P_NO)
- Ako poredimo sa oznakama u definiciji dobijamo:
  - A_k = City

        - Atribut City zavisi od dijela kljuca te je on atribut sa parcijalnom zavisnoscu
    - X1 = S_NO
        - Dio kljuca od koga zavisi atribut City
    - Z = P_NO
        - Atributi pocetne relacije koji nisu X_1 i A_k u ovom slucaju samo jedan
- Dvije relacije koje dobijamo:
    - $R_1(A_k X_1)$ i $R_2(X_1 Z)$

Napomene vezane za 2NF

- Uvijek moze da se uradi
- Uvijek daje dekompoziciju bez gubitka informacija
- Uvijek cuva zavisnosti
- Uvijek mora da se uradi

Ako je kljuc jednoclan onda je relacija u 2NF

- Ako kljuc ima samo jedan atribut ne moze postojati neki drugi atribut koji parcijalno zavisi od dijela kljuca

Ako je relacija dvoclana tada je ona u 2NF

- U slucaju dvoclane relacije mozemo imati:
    - Jednoclan kljuc
        - Svodi se na gornje tvrdjenje
    - Dvoclan kljuc
        - Ako je kljuc dvoclan, a relacija je takodje dvoclana, nije ostalo vise atributa koji mogu parcijalno zavisiti od kljuca

3NF

Friday, March 9, 2018 10:20 AM

Zabranjuje funkcionalne odnose izmedju sporednih atributa

- Sporedni atribut ne moze biti informacija odrugom sporednom atributu

Primjer:

Data je relacija: R(radnik, firma, grad) i sledece informacije:

- PK : {radnik}
  - radnik→firma
  - radnik→grad
- I funkcionalna zavisnosti
  - firma→grad

Vidimo da ovdje vaze anomalije:

- **Insert**
  - Nije moguce unijeti grad za firmu koja nema radnika
- **Update**
  - Promjena lokacije (grada) bi zahtijevalo promjenu na svim torkama
- **Delete**
  - Brisanje podataka o radnicima, gube se i svi podaci o firmi

Kako je kljuc jednoclan, ova sema jeste u 2NF iako ima vise anomalija

Neka su X, Y, Z skupovi atributa relacione seme R

Ako vazi X→Y && Y!→X && Y→Z tada kazemo da Z tranzitivno zavisi od X

Definicija 3NF

Relaciona sema R je u 3NF ako ni jedan njen sporedni atribut ne zavisi tranzitivno ni od jednog njenog kljuca

Ako je relacija u 3NF onda je ona i u 2NF (iz 3NF => 2NF)

Dokaz teoreme:

- Pretpostavimo da relacija nije u 2NF, a jeste u 3NF
    - Postoji sporedni atribut A koji parcijalno zavisi od nekog kljuca X
- Znaci da postoji X' ⊂ X takav da X'→A
- Kako je X kljuc, tada vazi da X→X' && X' ⊂ X (pravi podskup) && X'!→X && X'→A
- Ako uporedimo ovo sa definicijom, X' bi predstavljalo Y
    - Kljuc odredjuje X', a X' odredjuje A, to znaci da kljuc odredjuje A
- Atribut A tranzitivno zavisi od kljuca X sto znaci da relacije nije ni u 3NF
    - Dokazano suprotno

Alternativna definicija 3NF

Relaciona sema R je u 3NF u odnosu na skup funkcionalnih zavisnosti F ako za sve funkcionalne zavisnosti u F oblika X→Y, X,Y⊆attr(R) vazi bar jedan od sledecih uslova:

1) X→Y je trivijalna zavisnost
2) X je superkljuc
3) Svaki atribut A u Y-X sadrzan je u nekom kandidatskom kljucu za R
    - Da li cijeli skup Y-X mora da se sadrzi u nekom kandidatskom kljucu ili pojedinacni atributi moraju da se sadrze

3NF Dekompozicija

- Za svaku relacionu semu R i skup funkcionalnih zavisnosti F postoji dekpompozicija u 3NF koja je bez gubitka pri spajanju i koja cuva zavisnosti
- **Ulaz:**
    - elaciona sema R i skup funkcionalnih zavisnosti F
- **Izlaz:**
    - $\Phi(R_1, \ldots, R_s)$ tako da su R_i u 3NF i tako da cuva zavinosti i informaciju
- **Pseudo kod na nestrogom paskalu:**

    naci kanonicko pokrivanje F skupa Fc

```
i = 0
foreach funkcionalnu zavisnost X→Y iz F  do
    begin
        if nijedna od sema R  ( j ≤ 0 ≤ i ) ne zadrzi XY do
            begin
                i = i + 1
                    Ri = XY
            end
        if nijedna od sema R  ( j ≤ 0 ≤ i ) ne sadrzi kandidatski kljuc za R do
            begin
                i = i + 1
                Ri => proizvoljan kandidatski kljuc od R
            end
    end
return (R1, ..., Rs)
```

- Dekompozicija se uvijek radi po nekoj funkcionalnoj zavisnosti te je ona bez gubitka pri spajanju
- Koristimo sve zavisnosti, te kompozicija cuva zavisnosti
- Uvijek treba sprovesti 3NF i uvijek se moze sprovesti

Primjer 1 algoritma 3NF dekompozicije

- Data je relaciona sema R(radnik, firma, grad) i skup funkcionalnih zavisnosti
  - F = {radnik→firma, radnik→grad, firma→grad}
- Kanonicko pokrivanje skupa F je skup F_c
  - F_c = {radnik→firma, firma→grad}
- Prateci algoritam dobijamo:
  - R_1(radnik, firma)
    - Iz prve funkcionalne zavisnosti
  - R_2(firma, grad)
    - Iz druge funkcionalne zavisnosti

Primjer 2 algoritma 3NF dekompozicije

- Data je relaciona sema:
  - R(A, B, C, D, E)
- I skup funkcionalnih zavisnosti
  - $F = \{AB \rightarrow C, C \rightarrow B, A \rightarrow D\}$
- Pratimo algoritam:

1) Trazimo kanonicko pokrivanje od F
   - Vidimo da su desne strane vec jednoclani atributi
   - Za svaku 'lijevu' stranu zavisnosti (AB, C, A) trazimo zatvorenje tog skupa atributa u odnosu na F
     - Dolazimo do zakljucka da ni jedna od date tri zavisnosti se ne moze izvesti iz druge te da skup F ne sadrzi pravi podskup funkcionalnih zavisnosti cije bi zatvorenje bilo F^+
   - Na kraju treba provjeriti ima li suvisnih atributa sa lijeve strane (ovo se konkretno odnosi na zavisnosti $AB \rightarrow C$)
     - Ukoliko ukloni A, vidimo da ne postoji nacin da izvedemo $B \rightarrow C$
     - Slicno, ako uklonimo B, ne postoji nacin da izvedemo $A \rightarrow C$
   - Vidimo da je dati skup F ekvivalentan njegovom kanonickom pokrivanju F_c
2) Po algoritmu, prolazimo kroz svaku funkcionalnu zavisnost iz F_c te vrsimo dekompoziciju po toj zavisnosti
   - Kako imamo tri funkcionalne zavisnosti dobijamo:
     - $R_1(ABC)$
     - $R_2(BC)$
     - $R_3(AD)$
   - Vidimo da je $attr(R_2) \subseteq attr(R_1)$ tako da nju mozemo eliminisati, te dobijamo:
     - $R_1(ABC)$
     - $R_3(AD)$
3) Primijetimo kandidatske kljuceve za pocetnu relaciju

- {ABE} i {ACE}

4) Algoritam kaze da gledamo da li neka od novonastalih relacija ima u sebi sadrzan neki od kandidatski kljuceva
   - U ovom slucaju ni attr(R_1) ni attr(R_2) ne sadrze {ABE} ni {ACE}
   - Zbog toga dodajemo novu relaciju koja sadrzi kanditaski kljuc
     - Mozemo izabrati bilo koji
   - R_4(ABE)
5) Finalna dekompozicija
   - R_1(ABC)
   - R_3(AD)
   - R_4(ABE)

BCNF

Saturday, March 10, 2018 11:22 AM

Neka je data relacija S_p(S_No, S_Name, P_No, Quantity) i jedna instanca te relacije

S_No	S_Name	P_No	Quantity
s1	Ana	p1	200
s1	Ana	p2	300
s2	Jasna	p1	500
s2	Jasna	p3	400

Neka su {S_No, P_No} i {S_Name, P_No} kandidatski kljucevi

- Vidimo da onda vazi S_No→S_Name i S_Name→S_No

Ova relacija jeste u 2NF jer quantity zavisi od cijeloga kljuca

- Ne zavisi parcijalno od dijela kljuca

Ova relacija jeste u 3NF

- postoji samo jedan sporedni atribut (Quantity) te on ne moze kreirati zavisnosti sa drugim sporednim atributima jer ih nema

Prisutne su zavisnosti medju kljucnim atributima!

- Zbog ovoga ponovo su prisutne insert, update, delete anomalije

Resenje u ovom slucaju

- Mozemo odraditi dvije dekompozicije, po {S_No, S_Name}, i po {S_Name, S_No}

Ako odradimo dekompoziciju po {S_No, S_Name} dobijamo

- R_1(S_No, S_Name) i R_2(S_No, P_No, Quantity)

	R1				R2
S_No	S_Name		S_No	P_No	Quantity
s1	Ana		s1	p1	200
s2	Jasna		s1	p2	300
			s2	p1	500
			s2	p3	400

Ako odradimo dekompoziciju po {S_Name, S_No} dobijamo

- R_1(S_Name, S_No) i R_2(S_Name, P_No, Quantity)

	R1				R2
S_Name	S_No		S_Name	P_No	Quantity
Ana	s1		Ana	p1	200
Jasna	s2		Ana	p2	300
			Jasna	p1	500
			Jasna	p3	400

Vidimo da je u ovom slucaju dekompozicija dobra, da cuva zavisnosti i da je bez gubitka pri spajanju

- U ovom slucaju!

Definicija BCNF (Boyce-Codd Normal Form)

Relaciona sema R je u BCNF ako egzistencija funkcionalne zavisnosti X→Y gdje su X,Y ⊆ attr(R) takvi da je X∩Y = Ø implicira postojanje funkcionalnih zavisnosti $X \rightarrow A_i$ gdje je $1 \leq i \leq n$ za sve atribute A ⊆ attr(R) to jest X je kljuc

- Relaciona sema je u BCNF ako nema zavisnosti ni medju kljucnim atributima
- **BCNF => 3NF => 2NF => 1NF**

Alternativna Definicja

Relaciona sema R je u BCNF u odnosu na skup funkcionalnih zavisnosti F ako za svaku funkcionalnu zavisnosti X→Y iz F^+ vazi jedan od sledecih uslova:

- X→Y je trivijalna funkcionalna zavisnost
- X je kljuc

Alternativna Definicija

Relacija je u BCNF ako i samo ako: Za svaku netrivijalnu funkcionalnu zavisnost X→Y vazi da je X superkljuc relacije R

- Lijeva strana svake netrivijalne zavisnosti mora da sadrzi super kljuc
  - Jedan od super kljuceva mora biti podskup lijeve strane

BCNF zabranjuje bilo koju funkcionalnu zavisnost osim trivijalnih i kljucnih

Algoritam BCNF dekompozicija

- **Ulaz**
  - Relaciona sema $R(A_1 \ldots A_n)$ i skup funkcionalnih zavisnosti F
- **Izlaz**
  - Dekompozicija $\Phi(R_1, \ldots, R_s)$ tako da je R_i u BCNF i ona je bez gubitka pri spajanju
  - Ne pominje se nista o cuvanju funkcionalnih zavisnosti!
- **Pseudo kod na nestrogom paskalu:**

if nije u BCNF then

begin

nadji funkcionalnu zavisnost koja nije trivijalna i kojoj lijeva strana nije kljuc ($X \rightarrow Y$, $X \cap Y = \emptyset$)

$Z = attr(R) - XY$

Zamijenimo R njenom projekcijom $\pi_{xy}(R)$ i $\pi_{xz}(R)$ *to jest mijenjamo R sa nove dvije relacije* R_1 *i* R_2

Rekurzivno ponoviti postupa za nove relacije R_1 *i* R_2

end

else

dati R na izlaz

- Vidi se da je dekompozicija koja je bez gubitka pri spajanju
  - Nista se ne garantuje za cuvanje zavisnosti

Primjer

Data je relaciona sema R(student, predmet, profesor)

Neka su kandidatski kljucevi {student, profesor} i {student, predmet}

Neka su date zavisnosti:

1) {student, profesor}→predmet
2) {student, predmet}→profesor
3) {profesor}→predmet

Student	Predmet	Profesor
s1	baze	p1
s1	fizika	p2
s2	baze	p1
s2	fizika	p3

Data relacije nije u BCNF, ali jeste u 3NF

Da bi postigli BCNF moramo odraditi dekompoziciju po netrivijalnoj zavisnosti koja predstavlja problem a to je profesor→predmet te dobijamo

- R_1(profesor, predmet)
  - Odavde dobijamo zavisnost {profesor→predmet}
- R2(profesor, student)
  - Odavde dobijamo samo trivijalne zavisnosti
  - {profesor, student} → student
  - {profesor, student} → profesor
- Vidimo da od pocetne tri zavisnosti nakon dekompozicije moze da dobijemo samo 1 i 3
  - Zavisnost broj 2 smo izgubili!!

Ovim smo izgubili vazno znanje o onome sto modeliramo

- Na primjer, u relaciju R_2 mozemo da ubacimo novu torku (s_1, p_3) bez ikakvih problema
- Medjutim, ako pogledamo pocetnu relaciju i njene zavisnosti, vidimo da u nju ne bi mogli da ubacimo (s_1, p_3) jer to rusi zavisnosti {student, predmet}→profesor
  - Student s_1 vec slusa fiziku kod profesora p_2
  - Medjutim mi ovo ne mozemo provjeriti nakon dekompozicije te bi ovakav unos bio moguc

U ovakvim situacijama suocavacemo se sa ili update, delete anomalijama ili gubljenjem zavisnosti!

4NF

Friday, March 16, 2018 1:47 PM

Definicija

Relaciona sema $R(A_1,\dots,A_n)$ je u 4NF ako za svaku netrivijalnu viseznacnu zavisnost $X \to\to Y$ vaze i funkcionalne zavisnosti $X \to A_1 \dots X \to A_n$ to jest X je kljuc

Alternativna definicja

Relaciona sema je u 4NF ako za svaku viseznacnu zavisnost $X \to\to Y$ vazi jedan od sledeca dva uslova:

1) Zavisnost je trivijalna ($Y \subseteq X$)
2) X je superkljuc

Ako je relacija u BCNF ne znaci da je u 4NF

- BCNF !=> 4NF

Ako je relacija u 4NF onda je i u BCNF

- 4NF => BCNF => 3NF => 2NF => 1NF

Teorema

Neka je R relaciona sema X, Y, Z neprazni podskupovi atributa R i $Z = attr(R/XY)$

R se moze restaurirati prirodnim spajanjem ($\forall r(R)$) iz projekcija R[XY] i R[XZ] ako i samo ako vazi $X \to\to Y$

Posledica

Moguca je dekompozicija bez gubitka pri spajanju ako i samo ako

- $R_1 \cap R_2 \to\to R_1$ ili $R_1 \cap R_2 \to\to R_2$
- $R_1 \cap R_2 \to\to R_1 - R_2$ ili $R_1 \cap R_2 \to\to R_2 - R_1$

Algoritam dekompozicije u 4NF

Algoritam dekompozicije u 4NF je potpuno isti kao algoritam dekompozicije u BCNF samo se umjesto funkcionalne koristi viseznacna zavisnost

Po ovoj teoremi dekompozicija ce garantovano biti bez gubitka pri spajanju

- Moze i ne mora da cuva zavisnosti

5NF

Friday, March 16, 2018 2:07 PM

Zavisnosti spajanja i 5NF

- Project join NF ili PJNF

Zavisnost spajanja je prosirenje koncepta spajanje bez gubitka informacija za vise sema

Za neku relaciju R i njenu dekompoziciju $\varphi(R_1,\dots,R_2)$ vazi:

- $attr(R) = attr(R_1) \cup \dots \cup attr(R_s)$
- $R = R_1 \cup R_2 \cup \dots \cup R_s$

Kazemo da na relaciji r(R) vazi zavisnost spajanja u oznaci $*(R_1,\dots,R_s)$ ako vazi da je:

- $r = \pi_{R1}(r) \bowtie \pi_{R2}(r) \bowtie \dots \bowtie \pi_{Rs}(r)$

Primjer

Razmotrimo $*(R_1,R_2)$ na semi R i pretpostavimo da vazi zavisnost spajanja za ove dvije relacije, tada vazi

- $r = \pi_{R1}(r) \bowtie \pi_{R2}(r)$

Uzmimo proizvoljne 2 torke iz *r* takve da vazi:

$t_1[R_1-R_2] = a_1\dots a_i$

$t_1[R_1 \cap R_2] = a_{i+1}\dots a_j$

$t_1[R_2-R_1] = a_{j+1}\dots a_n$

$t_2[R_1-R_2] = b_1\dots b_i$

$t_2[R_1 \cap R_2] = a_{i+1}\dots a_j$

$t_2[R_2-R_1] = b_{j+1}\dots b_n$

Tabelarni prikaz ove dvije torke

	R_1-R_2	$R_1 \cap R_2$	R_2-R_1
t_1	$a_1\dots a_i$	$a_{i+1}\dots a_j$	$a_{j+1}\dots a_n$
t_2	$b_1\dots b_i$	$a_{i+1}\dots a_j$	$b_{j+1}\dots b_n$

Tada je

$\pi_{R1}(r)$:

R_1-R_2	$R_1 \cap R_2$
$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$
$b_1 \ldots b_i$	$a_{i+1} \ldots a_j$

$\pi_{R2}(r)$:

R_1-R_2	$R_1 \cap R_2$
$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$
$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$

Tada je $\pi_{R1}(r) \bowtie \pi_{R2}(r)$

	R_1-R_2	$R_1 \cap R_2$	R_2-R_1
t_1	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$
t_2	$b_1 \ldots b_i$	$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$
t_3	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$
t_4	$b_1 \ldots b_i$	$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$

$*(R_1, R_2) \Rightarrow R = \pi_{R1}(r) \bowtie \pi_{R2}(r)$

- U stvari smo dokazali
  - $R_1 \cap R_2 \rightarrow\rightarrow R_1 - R_2$
  - $R_1 \cap R_2 \rightarrow\rightarrow R_2 - R_1$

Zavisnost spajanja za dvije relacije je ekvivalentna sa viseznacnom zavisnoscu

- Zavisnost spajanja za dvije relacije je viseznacna zavisnost!!!

Primjer

Data je relaciona sema R(A, B, C) i jedna instanca te seme

a	**b**	**c**
a_1	b_1	c_2
a_2	b_1	c_1
a_1	b_2	c_1
a_1	b_1	c_1

Vazi: *(AB, BC, AC)

Ne Vazi:

- A→→B A→→C
- B→→A B→→C
- C→→A C→→B

Zavisnost spajanja nije ekvivalentna sa viseznacnom zavisnoscu osim kada pricamo o 2 relacije!

- Zavisnost spajanja je opstiji odnos od viseznacne zavisnosti

Relaciona sema je u 5NF na skupu svih funkcionalnih viseznacnih zavisnosti i zavisnosti spajanja D ako za sve zavisnosti spajanja iz D oblika $*(R_1 \ldots R_s)$ gdje je $R_i \subseteq R$, $R = R_1 \cup \ldots \cup R_s$ vazi:

- $*(R_1 \ldots R_s)$ je trivijalna zavisnost spajanja to jest $R_i = R$ za neko *i*
- Svako R_i je super kljuc u R

Algoritam za dekompoziciju u 5NF ne postoji

5NF => 4NF => BCNF => 3NF => 2NF => 1NF

6NF

Saturday, March 17, 2018 1:46 PM

Deklaracija domena

Neka je A atribut i dom(A) skup vrijednosti

Deklaracija domena u oznaci $A \subseteq dom(A)$ zahtijeva da vrijednost atributa A na svim torkama pripada ovom skupu

Deklaracija kljuca

Neka je R relaciona sema i $K \subseteq attr(R)$

Deklaracija kljuca K je zahtijev da K bude super kljuc to jest da $K \to R$

Opsta ogranicenja

Opsta ogranicenja su bilo koji predikat na skupu svih relacija relacione seme R

- Primjeri opstih ogranicenja bi bili
  - Funkcionalne zavisnosti
  - Viseznacne zavisnosti
  - Zavisnosti spajanja
- Primjer ogranicenja
  - "ako je prva cifra 8 iznos racuna je veci od 100"

Ogranicenja domena i kljuca se relativno lako implementiraju

- Opsta ogranicenja se tesko provjeravaju!

Cemu se tezi u dizajnu?

Neka je:

- **D** : skup svih ogranicenja domena na nekoj semi
- **K** : skup svih ogranicenja kljuceva na nekoj semi
- **G** : skup svih datih opstih ogranicenja

Tezimo da $D \cup K \models G$

- Da smo u mogucnosti da iz ogranicenja domena i kljuceva (koji se lako provjeravaju) izvedemo sva potrebna opsta ogranicenja

Ako ovo vazi onda je sema u DKNF

DKNF (Domain Key Normal Form)

Predstavlja najopstiju normalnu formu

- Ostale su samo specijalan slucaj

Ne postoji algoritam za dekompoziciju u DKNF

- Postoji motivacija prema cemu treba teziti u dizajnu relacionih sema

Primjer

Razmatramo opste ogranicenje "ako je prva cifra 8 iznos racuna je veci od 100"

- racun(broj, ..., iznos)
  - broj ne pocinje sa 8
- specRacun(broj, ..., iznos)
  - broj pocinje sa 8, iznos > 100

Vidimo da je opste ogranicenje prevedeno u ogranicenje domena

- Time smo ovu relacionu semu doveli u DKNF

Viseznacne Zavisnosti

Sunday, March 11, 2018 11:13 AM

Slucaj kada jedan atribut iz X odredjuje vise (skup) atributa iz Y

Oznaka X→→Y

Primjer: student→→jezik_koji_zna

- A → {engleski, francuski}
- B → {engleski, ruski, italijanski}

Primjer: student→→boja

- A → {plava, zelena}
- B → {crvena, zelena}

Postavlja se pitanje kako u jednoj tabeli predstaviti vise viseznacnih zavisnosti?

student	jezik	boja
A	engleski	plava
A	engleski	zelena
A	francuski	plava
A	francuski	zelena
B	engleski	crvena
B	engleski	zelena
B	ruski	crvena
B	ruski	zelena
B	italijanski	crvena
B	italijanski	zelena

Kada imamo vise viseznacnih zavisnosti, one se predstavljaju po principu svako sa svakim

- Primijetimo da je gornja instanca u BCNF

Vidimo da gornja relacija ima prije svega viska informacija (redundancy)

- Svaki jezik se pojavljuje onoliko puta koliko ima boja
- Boja se pojavljuje jednom za svaki jezik

Ovo je neophodno da bi se izrazile nezavisnosti izmedju viseznacnih zavisnosti, medjutim ovakvo ponavljanje informacija nije pogodno

Definicija visestruke zavisnosti

Neka je R relaciona sema i X, Y $\subseteq$ attr(R). Viseznacna zavisnosti $X \rightarrow\rightarrow Y$ vazi nad R ako za svaku dozvoljenu relaciju r(R) i za svako t_1, t_2 iz r za koje vazi $t_1[x]=t_2[x]$ vazi da postoje torke t_3, t_4 iz r takve da je

$t_1[X]=t_2[X]=t_3[X]=t_4[X]$

$t_3[Y]=t_1[Y]$

$t_3[R-Y]=t_2[R-Y]$

$t_4[Y]=t_2[Y]$

$t_4[R-Y]=t_1[R-Y]$

Alternativna definicija

Za svaku torku iz r(R) t_1, t_2 za koje vazi t1[x]=t2[x] (slazu se na X), postoj torka t_3 koja se slaze:

- Sa t_1, t_2 na x ($t_1[x]=t_2[x]=t_3[x]$)
- Sa t_1 na Y ($t_1[y]=t_3[y]$)
- Sa t_2 na svim ostalim atributima iz R koji nisu medju X i Y (R - XY)
    - $t_2[R-XY]=t_3[R-XY]$

Takodje prema istom principu mozemo reci da postoji jos jedna torka t_4 koja se slaze:

- Sa t_1, t_2, t_3 na x ($t_1[x]=t_2[x]=t_3[x]=t_4[x]$)
- Sa t_2 na Y ($t_2[y]=t_4[y]$)
- Sa t_1 na svim ostalim atributima iz R koji nisu medju X i Y (R - XY)
    - $t_1[R-XY]=t_4[R-XY]$

Sta se ovim dobija ?

- Za svaku fiksiranu vrijednost atributa X, atributi Y i svi ostali atributi (R - XY) se pojavljuju u svim mogucim kombinacijama u razlicitim torkama!

Postavlja se pitanje da li X→Y znaci da X→→Y ?

Tvrdjenje : Svaka funkcionalna zavisnosti X→Y je takodje i viseznacna zavisnosti X→→Y

- Pretpostavimo da je R neka relacija na kojoj vazi funkcionalna zavisnosti X→Y
- Pretpostavimo da su t_1 i t_2 dvije torke koje se slazu na X
- Da bi dokazali X→→Y moramo dokazati da R takodje sadrzi torku t_3 koja se slaze sa torkama t_1, t_2 na X, ali takodje i sa t_1 na Y, i t_2 na svim ostalim atributima
  - t_3 mora biti razlicito od t_2

Ako X→Y onda X→→Y

- Funkcionalna zavisnost je istovremeno i viseznacna zavisnost
- Funckionalna zavisnost je specijalan slucaj viseznacne zavisnosti
- Viseznacna zavisnosti je opstiji pojam

Aksiomi za viseznacne zavisnosti

Friday, March 16, 2018 12:58 PM

Opsta definicija viseznacne zavisnosti

t_1	a	b_1	c_1
t_2	a	b_2	c_2
t_3	a	b_1	c_2
t_4	a	b_2	c_1

Konkretan primjer viseznacne zavisnosti

student	jezik	boja
a	eng	plava
a	franc	zelena
a	eng	zelena
a	franc	plava

U primjeru vaze zavisnosti

- student →→ jezik
- student →→ boja

Primijetimo da vazi:

- $X \to\to Y \Rightarrow X \to\to R\text{-}X\text{-}Y$
- $X \to Y \Rightarrow X \to\to Y$

Bice dalje objasnjeno

Kratak dokaz da kada $X \to Y \Rightarrow X \to\to Y$

$t_1 = t_3$	a	b	c_1
$t_2 = t_4$	a	b_2	c_2

Viseznacna zavisnosti trazi da postoje jos dvije torke koje se slazu na atributima a, a koje su onda medjusobno ukrstene na atributima b i c

- Vidimo da mozemo uzeti da je torka t_1 u stvari torka t_3 i obratno cime smo direktno dokazali da je funkcionalna zavisnost istovremeno i viseznacna

Armstrongovi aksiomi za viseznacne zavisnosti

1) **Y ⊆ X => X→Y**
    - Refleksivnost
2) **X→Y => WX → WX**
    - Prosirenje
3) **X→Y && Y→Z => X→Z**
    - Tranzitivnost
4) **X→→Y => X→→R-X-Y**
    - Komplementacija
5) **X→→Y => WX →→ VY kada je V ⊆ W**
    - Viseznacno prosirenje
6) **X→→Y && Y→→Z => X→→Z-Y**
    - Viseznacna tranzitivnost
7) **X→Y => X→→Y**
8) **X→→Y, Z⊆Y, ∃W⊆R : W∩Y=Ø && W→Z => X→Z**

Dodatni aksiomi

1) **X→→Y && X→→Z => X→→YZ**
    - Viseznacna unija
2) **X→→Y && X→→Z => X→→Y∩Z**
    - Viseznacni presjek
3) **X→Y && X→→Z => X→→Y-Z && X→→Z-Y**
    - Viseznacna razlika

Primjer

Data je relaciona sema R(A, B, C, G, H, I) i skup zavisnosti
F={A→→B, B→→HI, CG→H}

Dokazati da vazi:

a) A→→CGHI
b) A→→HI

c) B→H

d) A→→CG

a. A→→B => A→→ABCGHI - AB => A→→CGHI
   - Iz komplementacije
b. A→→B && B→→HI => A→→HI-B => A→→HI
   - Iz tranzitivnosti
c. B→→HI && H⊆HI && CG→H && CG⊆HI=Ø => B→→H
   - Iz pravila 8.
   - Nasli smo podskup desne strane (H)
   - Nasli smo neki skup atributa koji odredjuje H (CG)
   - Taj skup nema nista zajednicko sa pocetnom desnom stranom (HI i CG nemaju nista zajednicko)
   - Iz svega ovoga slijedi da poceni skup atributa B odredjuje H
d. A→→CGHI (iz a.) && A→→HI (iz b.) => A→→CGHI - HI => A→→CG
   - Koristimo razliku

Primjer

Data je relaciona sema CPK(kurs, profesor, knjiga) i jedna instanca te seme:

krus	profesor	knjiga
c_1	p_1	k_1
c_1	p_2	k_2
c_1	p_2	k_1
c_1	p_1	k_2
c_2	p_3	k_3
c_2	p_3	k_4
c_2	p_3	k_5

Vidimo da: kurs→→profesor i kurs→→knjiga

Primarni kljuc: PK{kurst, profesor, knjiga}

Primijetimo da relacija jeste u BCNF ali i dalje imamo nagomilavanje podataka, update anomalije itd..

Mozemo uraditi dekompoziciju:

- **R_1(kurs, profesor)**

kurs	**profesor**
c_1	p_1
c_1	p_2
c_2	p_3

- **R_2(kurs, knjiga)**

kurs	knjiga
c_1	k_1
c_1	k_2
c_2	k_3
c_2	k_4
c_2	k_5

Vidimo da se cuva informacija i da vaze trivijalne funkcionalne zavisnosti

- U ovom konkretnom slucaju

Smjernice Prilikom Dizajna Baze

Saturday, March 17, 2018 2:02 PM

1) Modelirati preko ER dijagram
   - ER dijagram je semanticki metod i ne pruza dobru matematicku osnovu
   - Sa druge strane pruza odlicne mogucnosti modelovanja
2) Napraviti relacione seme po ER dijagramu
3) Uvijek treba raditi 3NF dekompoziciju
   - Znamo da cuva zavisnosti
   - Znamo da je bez gubitka pri spajanju
   - Znamo da resava neke anomalije (u nekim slucajevima)
4) Uociti da li je dobijena relacija u BCNF a onda i u 4NF
   - U slucaju da nije uraditi BCNF dekompoziciju i 4NF dekompoziciju ali provjeriti da li se cuvaju zavisnosti (moze biti komplikovano)
5) Razmisliti o Project Join i DKNF normama

Nemoguce je doci do sintaksnog metoda koji daje gotovu relacionu semu