Poglavje VIII

Normalizacija relacij (tabel)

Normalizacija in podatkovne baze

- Spada na širše področje načrtovanja PB
- Samostojna uporaba primerna za manj kompleksne PB (do cca. 10 tabel)
- Imamo problem iz realnega sveta, opisan z atributi in omejitvami njihovih vrednosti.
- Normalizacija odgovarja na vprašanja:
  - ali je **obstoječa** struktura relacij oz. tabel v PB primerna za obravnavo danega problema (pasivna vloga normalizacije)
  - kako **združiti atribute v tabele**, da dobimo najprimernejšo strukturo PB (aktivna vloga normalizacije)

Modeliranje omejitev s funkcionalnimi odvisnostmi

- Relacija je model nekega stanja v svetu, torej njena vsebina ne more biti poljubna.
- Realne omejitve ne dovoljujejo, da bi bili odnosi v svetu kakršnikoli; možna so le določena stanja (tudi: poslovna pravila).
- Omejitve v relacijskem modelu formalno opišemo s pomočjo odvisnosti
- Odvisnosti so sredstvo, s katerim v relacijskem podatkovnem modelu povemo, katere vrednosti relacij (kombinacije vrednosti atributov v vrsticah) so veljavne in katere sploh ne morejo obstajati

Funkcionalne odvisnosti..

- Predpostavimo, da obstaja relacijska shema R z množico atributov, katere podmnožici sta X in Y.
- V relacijski shemi R velja X → Y
  (X funkcionalno določa Y oziroma Y je funkcionalno odvisen od X),
  če v nobeni relaciji, ki pripada shemi R, ne obstajata dve n-terici (vrstici), ki bi se ujemali v vrednostih atributov X in se ne bi ujemali v vrednostih atributov Y.
- Preprosto povedano, **obstaja** neka funkcija f, s pomočjo katere lahko iz vrednosti X izračunamo vrednosti Y = f(X).

Primeri funkcionalnih odvisnosti

- Imamo relacijo s shemo

  Stevila(X,Y,Z)

  z naslednjim pomenom:

  X in Y sta celi števili, Z pa njuna vsota

- Funkcionalne odvisnosti relacijske sheme so:

  $F \equiv \{ XY \rightarrow Z, XZ \rightarrow Y, YZ \rightarrow X\}$

- S pomočjo funkcionalnih odvisnosti definiramo tudi pojem ključa, ki implementira določene integritetne omejitve.

Primeri funkcionalnih odvisnosti

- Imamo relacijo s shemo

  Izpit(VpŠt, Priimek, Ime, ŠifraPredmeta, Datum izpita,
  OcenaPisno, OcenaUstno)

  z naslednjim pomenom:

  Študent z vpisno številko VpŠt ter priimkom Priimek in imenom Ime je na DatumIzpita opravljal izpit iz predmeta s šifro ŠifraPredmeta. Dobil je oceno OcenaPisno in OcenaUstno.

- Funkcionalne odvisnosti relacijske sheme Izpit so:

  F ≡ { VpŠt → (Priimek, Ime), (VpŠt, ŠifraPredmeta, DatumIzpita) → (OcenaPisno, OcenaUstno) }

Iskanje naravnih ključev relacije na podlagi definiranih funkcionalnih odvisnosti

- Izhajamo iz relacijske sheme R (množica atributov) in nad njo definirane množice funkcionalnih odvisnosti F
- Pomožni pojmi (zaprtje množice atributov)
- Različni postopki (algoritmi) za določanje enega ali vseh naravnih ključev
- Zakaj te postopke potrebujemo (integritetne omejitve, normalizacija)
- Kompaktna notacija zapisa: $X \rightarrow Y$
  X in Y sta množici atributov

Zaprtje (closure) množice atributov

- Zaprtje X^+ množice atributov X glede na množico funkcionalnih odvisnosti F
- X^+ = {A: A lahko izračunamo iz X s pomočjo odvisnosti iz F}

- Postopek za izračun X^+

```
Vhod: X, F
Izhod: X+

X+ = X
ponavljaj
    stari X+ = X+
    za vsako odvisnost Y → Z ∈ F naredi
        če Y ⊆ X potem
            X+ = X+ ∪ Z
dokler ni stari X+ = X+
```

Primer izračuna zaprtja

R=ABCDEFG

F={A$\rightarrow$B, BE$\rightarrow$G, EF$\rightarrow$A, D$\rightarrow$AC}

Iščemo $\{EF\}^+$:

1. $\{EF\}^+ = EFA$
2. $\{EFA\}^+ = EFAB$
3. $\{EFAB\}^+ = EFABG$
4. $\{EFABG\}^+ = EFABG$

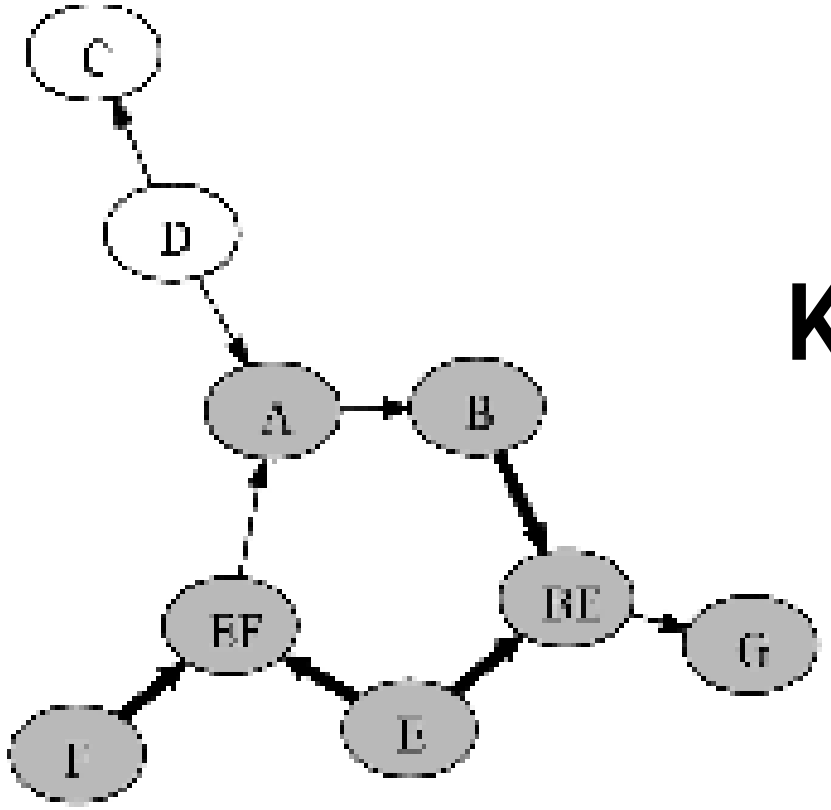
5. Končni rezultat: $\{EF\}^+ = EF^+ = EFABG$

Ali je EF ključ ali vsak kandidat za ključ? Ne, ker $\{EF\}^+ \subset R$.

Za vsakega kandidata za ključ K namreč velja: $K^+=R$.

Primer izračuna zaprtja

R=ABCDEFG
F={A→B, BE→G, EF→A, D→AC}
$\{EF\}^+$ = ?



KONEC!

Iskanje naravnih ključev relacije na podlagi podanih funkcionalnih odvisnosti

- Splošni veljavne resnice
  - Atribut, ki ne nastopa na desni strani nobene funkcionalne odvisnosti, mora biti vsebovan v vsakem ključu
  - Atribut, ki nastopa na desni strani neke funkcionalne odvisnosti in ne nastopa na levi strani nobene funkcionalne odvisnosti, ne more biti vsebovan v nobenem ključu
  - Dobri kandidati za ključe so leve strani funkcionalnih odvisnosti in njihove unije

Elmasari-Navathe algoritem za določanje enega ključa

- Vhod: relacijska shema R,
  množica funkcionalnih odvisnosti F

1. Postavi K= začetni kandidat, npr. R (vsi atributi)
2. Za vsak atribut $X \in K$
   Izračunaj $\{K-X\}^+$ glede na F
   Če $\{K-X\}^+ = R$ (vsebuje vse atribute R)
   postavi K= K- {X}
3. Kar ostane v K je ključ.

- Problem: vrne samo en ključ, odvisen od vrstnega reda pregledovanja atributov

Primer (Elmasari-Navathe):

R=ABCDEFG

F={A $\rightarrow$D, AG $\rightarrow$B, B $\rightarrow$G, B $\rightarrow$E, E $\rightarrow$B, E $\rightarrow$F}

- K=ABCDEFG, X=A
  K-X= BCDEFG, $(K-X)^+$= BCDEFG
  manjka A
- K=ABCDEFG, X=B
  K-X= ACDEFG, $(K-X)^+$= ABCDEFG (AG$\rightarrow$B)
- K=ACDEFG, X=C
  K-X= ADEFG, $(K-X)^+$= ABDEFG (AG$\rightarrow$B)
  manjka C
- K=ACDEFG, X=D
  K-X= ACEFG, $(K-X)^+$= ABCDEFG (AG$\rightarrow$B, A$\rightarrow$D)
- K=ACEFG, X=E
  K-X= ACFG, $(K-X)^+$= ABCDEFG (AG$\rightarrow$B, A$\rightarrow$D,B$\rightarrow$E)
- K=ACFG, X=F
  K-X= ACG, $(K-X)^+$= ABCDEFG (AG$\rightarrow$B, A$\rightarrow$D,B$\rightarrow$E,E $\rightarrow$F)
- K=ACG, X=G (A$\rightarrow$D)
  K-X= AC, $(K-X)^+$= ACD manjkajo BEFG
-

Ključ je
ABCDEFG - BDEF
torej
ACG

Saiedian-Spencer algoritem za določanje vseh ključev

- Vhod: relacijska shema R,
  množica funkcionalnih odvisnosti F

1. Poišči množice $\mathcal{L}$ (atributi samo na levi strani odvisnosti in atributi ki ne nastopajo v nobeni odvisnosti), $\mathcal{R}$ (atributi samo na desni strani odvisnosti) in $\mathcal{B}$ (atributi na levi in desni strani odvisnosti)

2. Preveri množico $\mathcal{L}$. Če $\mathcal{L}^+$=R, je edini ključ in lahko končaš, sicer nadaljuj na koraku 3.

3. Preveri množico $\mathcal{B}$ tako da v $\mathcal{L}$ vstavljaš po vrsti vse možne podmnožice atributov X iz $\mathcal{B}$, začenši s posameznimi atributi. Kadar dobimo $\{\mathcal{L} \cup X\}^+$ =R, smo našli ključ. N-teric, ki vsebujejo X, dalje ne obravnavamo več.

Primer (Saiedian-Spencer):

R=ABCDEFG

F={A $\rightarrow$D, AG $\rightarrow$B, B $\rightarrow$G, B $\rightarrow$E, E $\rightarrow$B, E $\rightarrow$F}

---

1. $\mathcal{L}$= CAGBE – BGE = CA
   $\mathcal{R}$= DBGEF – BGE = DF
   $\mathcal{B}$ = BGE

2. $\mathcal{L}^+$=CAD $\subseteq$ R

3. X=B $\mathcal{L}$=CAB
   $\mathcal{L}^+$=CABDGEF = R

4. X=G $\mathcal{L}$=CAG
   $\mathcal{L}^+$=CAGDBEF = R

5. X=E $\mathcal{L}$=CAE
   $\mathcal{L}^+$=CAEDBFG = R

Ključi: ABC, ACG, ACE

Ključ kot integritetna omejitev

- Integritetna omejitev v splošnem zagotavlja smiselno vsebino podatkov – njihovo celovitost
- Primarni ključ ne služi le potrebam enolične identifikacije vrstic
- Primarni ključ zagotavlja tudi integritetno omejitev enoličnosti, kar zagotavlja možnost enolične identifikacije tudi v prihodnje
- To še posebej velja za naravne ključe.

Normalizacija

- Problemi zaradi redundance podatkov v osnovnih relacijah.
- Namen normalizacije.
- Uporabnost normalizacije pri načrtovanju relacijske podatkovne baze.
- Postopek normalizacije.

Normalizacija

- Normalne oblike glede na restriktivnost in stopnjo (ne)redundantnosti:
  - I. normalna oblika,
  - II. normalna oblika,
  - III. normalna oblika,
  - Boyce-Coddova normalna oblika,
  - IV. normalna oblika,
  - V. normalna oblika.,
  - VI. normalna oblika

Kaj je normalizacija

- Normalizacija je postopek, s katerem pridemo do množice primernih (primerno strukturiranih) relacij, ki ustrezajo potrebam uporabe
- Nekaj lastnosti primernih relacij:
  - Relacije imajo minimalen nabor atributov → zgolj tiste, ki so potrebni za pokritje potreb poslovnega sistema;
  - Atributi, ki so logično povezani, so zajeti v isti relaciji;
  - Med atributi relacij je minimalna redundanca → vsak atribut (razen tujih ključev) je predstavljen samo enkrat.

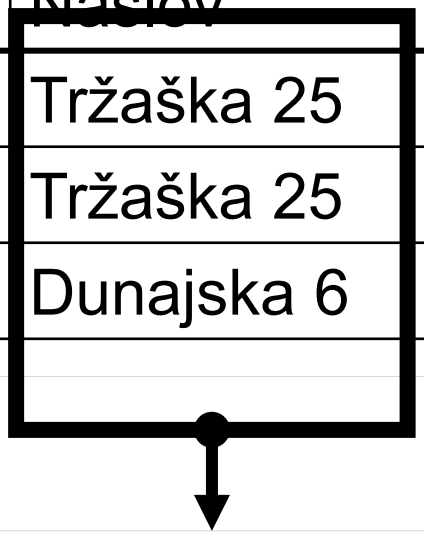
Načrtovanje PB in normalizacija

- Osnovni cilj načrtovanja relacijske podatkovne baze je smiselno grupirati atribute v relacije na način, da bo med podatki čim manj redundance.
- Potencialne koristi pravilnega načrtovanja so:
  - Spremembe podatkov v podatkovni bazi dosežemo z minimalnim številom operacij → večja učinkovitost; manj možnosti za podatkovne nekonsistentnosti.
  - Manjše potrebe po diskovnih kapacitetah za shranjevanje osnovnih relacij → manjši stroški.
- Normalizacija je pomemben korak v postopku načrtovanja.

Primer nenormalizirane relacije

- Relacija Osebje ima odvečne podatke.

Ime	Priimek	Oddelek	Naslov
Janez	Novak	1A	Tržaška 25
Peter	Klepec	1A	Tržaška 25
Marija	Kovač	2A	Dunajska 6



Odvečni (ponavljajoči se) naslovi.

Ažurirne anomalije

- Relacije, ki vsebujejo odvečne podatke lahko povzročajo ažurirne anomalije pri operacijah nad podatki.
- Poznamo več vrst anomalij:
  - Anomalije pri dodajanju n-teric (vrstic) v relacijo
  - Anomalije pri brisanju n-teric (vrstic) iz relacije
  - Anomalije pri spreminjanju n-teric (vrstic) v relaciji

Anomalije pri dodajanju vrstic

- Dodajanje novih članov oddelka: ponovno moramo (pravilno) vpisati naslov oddelka
- Dodajanje novega oddelka: za podatke o članu vpišemo NULL; problem: kaj je ključ?

Ime	Priimek	Oddelek	Naslov
Janez	Novak	1A	Tržaška 25
Peter	Klepec	1A	Tržaška 25
Marija	Kovač	2A	Dunajska 6
Janko	Jankovič	1A	Tržaška 52
NULL	NULL	3A	Celovška 12

Anomalije pri brisanju vrstic

- Brisanje edinega člana oddelka: izgubimo tudi vse informacije o tem oddelku (šifra oddelka, naslov)

Ime	Priimek	Oddelek	Naslov
Janez	Novak	1A	Tržaška 25
Peter	Klepec	1A	Tržaška 25
~~Marija~~	~~Kovač~~	~~2A~~	~~Dunajska 6~~

Anomalije pri spreminjanju vrstic

- Oddelek 1A se preseli na Jadransko 21. Naslov je treba pravilno popraviti pri vseh članih oddelka!

Ime	Priimek	Oddelek	Naslov
Janez	Novak	1A	Tržaška 25
Peter	Klepec	1A	Tržaška 25
Marija	Kovač	2A	Dunajska 6

Postopek normalizacije

- Postopku preoblikovanja relacij v obliko, pri kateri do ažurirnih anomalij ne more priti, pravimo normalizacija.
- Obstaja več stopenj normalnih oblik (form):
  - I. normalna oblika,
  - II. normalna oblika,
  - III. normalna oblika,
  - Boyce-Coddova normalna oblika,
  - IV. normalna oblika,
  - V. normalna oblika,
  - VI. normalna oblika.

Prva normalna oblika..

- Relacija je v prvi normalni obliki, če:
  - Nima večvrednostnih atributov, kar pomeni, da ima vsak atribut lahko le eno vrednost (torej vrednost ne more biti množica). Primer: telefonska številka (fiksna, mobilna, službena)
  - Nima sestavljenih atributov (torej vrednost ne more biti relacija). Primer: naslov (ulica, hišna številka, pošta, kraj, država)
  - Ima definiran primarni ključ in določene funkcionalne odvisnosti
- Koraki normalizacije:
  - Elimininiranje ponavljajočih skupin (večvrednostne in sestavljene atribute skupno obravnavamo kot ponavljajoče skupine)
  - Določitev funkcionalnih odvisnosti
  - Določitev primarnega ključa

Prva normalna oblika..

- Primer relacijske sheme:
  - Študent(VŠ, priimek, ime, pst, kraj, (šifra predmeta, naziv, ocena))
- Določimo primarni ključ:
  - VŠ
- Odpravimo ponavljajoče skupine:
  - Študent(VŠ, priimek, ime, pst, kraj)
  - PredmetOcena(šifra predmeta, naziv, ocena)
- Določimo primarna ključa obeh relacijskih shem:
  - Študent(VŠ, priimek, ime, pst, kraj)
  - PredmetOcena(#VŠ, šifra predmeta, naziv, ocena)

Ključ glavne relacije se kot tuji ključ prenese k ponavljajočim skupinam in postane del njihovega primarnega ključa.

Prva normalna oblika

- Gledamo obe relacijski shemi ...
  - Študent(VŠ, priimek, ime, pst, kraj)
  - PredmetOcena(#VŠ, šifra predmeta, naziv, ocena)
- ... in določimo funkcionalne odvisnosti:
  - VŠ → Ime
  - VŠ → Priimek
  - VŠ → pst
  - VŠ → kraj
  - Pst → kraj

  - Šifra predmeta → naziv
  - VŠ, šifra predmeta → naziv
  - VŠ, šifra predmeta → ocena

Druga normalna oblika..

Shema: <u>ABC</u>DE

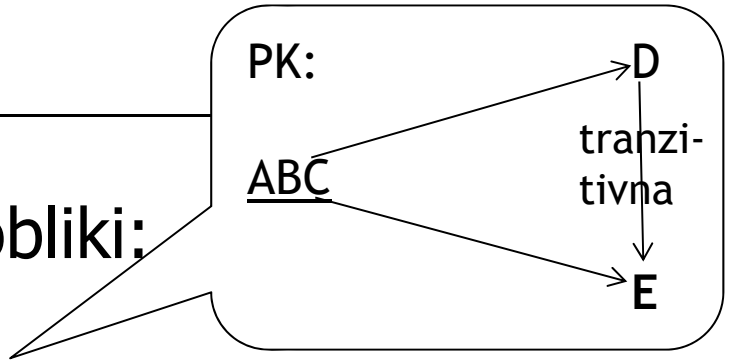
ABC → D
ABC → E
B → E parcialna

- Relacija je v drugi normalni obliki:
  - Če je v prvi normalni obliki
  - Ne vsebuje parcialnih (delnih) odvisnosti: noben atribut ni funkcionalno odvisen le od dela primarnega ključa, temveč le od celotnega ključa
  - Normalizacija: problematičnim (parcialnim) odvisnostim dodelimo nove relacijske sheme (vsaki svojo). Desne strani izločimo iz originalne sheme.
- Ugotovitev:
  - Relacija, katere primarni ključ je sestavljen le iz enega atributa, je v drugi normalni obliki
  - Relacija, katere primarni ključ je sestavljen iz vseh atributov, je v drugi normalni obliki

Druga normalna oblika

- Nadaljevanje predhodnega primera:
  - Študent(VŠ, priimek, ime, pst, kraj)
  - PredmetOcena(#VŠ, šifra predmeta, naziv, ocena)
- Funkcionalne odvisnosti:
  - VŠ → Ime
  - VŠ → Priimek
  - VŠ → pst
  - VŠ → kraj
  - Pst → kraj
  - **Šifra predmeta → naziv**
  - VŠ, šifra predmeta → naziv
  - VŠ, šifra predmeta → ocena
- Relacijska shema Študent je v drugi normalni obliki
- Relacijsko shemo PredmetOcena normaliziramo :
  - Predmet_študent (#VŠ, #šifra predmeta, ocena)
  - Predmet (šifra predmeta, naziv)

Tretja normalna oblika..



- Relacija je v tretji normalni obliki:
  - Če je v drugi normalni obliki
  - Če ne vsebuje tranzitivnih funkcionalnih odvisnosti: ni funkcionalnih odvisnosti med atributi, ki niso del primarnega ključa oz. ne obstaja atribut, ki ni del primarnega ključa, ki bi bil funkcionalno odvisen od drugega atributa, ki ravno tako ni del primarnega ključa
- Ugotovitev:
  - Relacija, katere primarni ključ je sestavljen iz vseh atributov, je v tretji normalni obliki
  - Relacija, kjer le en atribut izmed vseh ni del primarnega ključa, je v tretji normalni obliki

Tretja normalna oblika

- Nadaljevanje predhodnega primera:
  - Študent(VŠ, priimek, ime, pst, kraj)
  - Predmet (šifra predmeta, naziv)
  - Predmet_študent (#VŠ, #šifra predmeta, ocena)
- Funkcionalne odvisnosti:
  - VŠ → Ime
  - VŠ → Priimek
  - **VŠ → pst**
  - **VŠ → kraj**
  - **Pst → kraj**
  - Šifra predmeta → naziv
  - VŠ, šifra predmeta → naziv
  - VŠ, šifra predmeta → ocena

- Relaciji
  Predmet
  in
  Predmet_Študent
  sta v tretji normalni obliki

- Relacija Študent:
  - Študent (VŠ, priimek, ime, #pst)
  - Pošta (pst, kraj)

Postopek normalizacije v 3. NO

- Normalizacija relacijske sheme R v ρ
  - $\rho = \{\}$
  - Vsaki problematični odvisnosti $X \rightarrow A \in F$ priredimo novo relacijsko shemo XA v ρ, razen v primeru, če že obstaja kakšna shem, ki XA vključuje kot podmnožico. Desno stran odvisnosti (A) izločimo iz originalne sheme.
  - Kar ostane od originalne relacijske sheme dodamo v ρ, razen če v ρ že obstaja kakšna shema, ki jo vsebuje

    Odvisnost je problematična, kadar je tranzitivna (3. NO) ali parcialna (2. NO)

Primer normalizacije v 3. NO

R=ABCDEFG
F={A $\rightarrow$D, AG $\rightarrow$B, B $\rightarrow$G, B $\rightarrow$E, E $\rightarrow$B, E $\rightarrow$F}
Primarni ključ: ACG

1.	ρ={}		
2.	A $\rightarrow$D:	$\rho = \rho \cup$ {AD}	parcialna
3.	AG $\rightarrow$B:	ρ= {AD} $\cup$ {AGB}	parcialna
4.	B $\rightarrow$ G:	ni problematična	
5.	B $\rightarrow$ E:	ρ= {AD, AGB} $\cup$ {BE}	tranzitivna
6.	E $\rightarrow$ B:	{EB} $\subseteq$ {BE}	
7.	E $\rightarrow$ F:	ρ= {AD, AGB, BE} $\cup$ {EF}	tranzitivna
8.	Končamo:	ρ= {AD, AGB, BE, EF} $\cup$ {ACG}	

ρ= {AD, AGB, BE, EF, ACG}

Striktna 2. in 3. NO

- Primarni ključ je lahko poljubno izbran izmed vseh kandidatov za ključ
- Zato je, striktno gledano, pravilnejši naslednji pristop k normalizaciji:
  - Pogoje za drugo in tretjo normalno obliko preverjamo glede na VSE kandidate za ključ in ne samo glede na primarni ključ
  - Torej: izraz *"primarni ključ"* zamenjamo z *"neki kandidat za ključ"*
  - Vse kandidate za ključ dobimo npr. s Saiedian-Spencer algoritmom

Denormalizacija..

- Včasih zavestno uporabljamo relacije, ki ne ustrezajo najvišjim normalnim oblikam.
- Prve in druge normalne oblike nikoli ne kršimo.
- Višjim normalnim oblikam se včasih odrečemo zaradi dejanskega poznavanja problematike, doseganja boljše učinkovitosti ali ohranjanja omejitev (funkcionalnih odvisnosti).

Denormalizacija..

- Normalizirane sheme v nekaterih primerih predstavljajo oviro za učinkovito implementacijo
- Primer:
  - Rezultat (Startna stevilka, Ime, Priimek, Cas_Prvi_Tek, Cas_Drugi_Tek, Cas_Skupaj)
  - Ta relacija ni v tretji normalni obliki:
    Cas_Prvi_Tek,Cas_Drugi_Tek $\rightarrow$ Cas_Skupaj
  - Ni je potrebno dekomponirati v tretjo normalno obliko, pod pogojem, da kontroliramo ali izračunavamo vsebino tako, da nikoli ne more biti v *Cas_Skupaj* kaj drugega, kot vsota.

Preverjanje denormalizacije z omejitvijo

```
ALTER TABLE Rezultat
ADD CONSTRAINT PreveriSkupniCas
CHECK
   (NOT EXISTS
         (SELECT Startna_stevilka
          FROM Rezultat
          WHERE Cas_Skupaj != Cas_Prvi_Tek + Cas_Drugi_Tek
   ));
```

Vpisujemo vse tri atribute: Cas_Prvi_Tek, Cas_Drugi_Tek, Cas_Skupaj, omejitev pa ob vsaki spremembi tabele kontrolira vse vrstice, ce smo res povsod vnesli vsoto.

Implementacija denormalizacije s prožilcem

```
CREATE TRIGGER IzracunajSkupniCas
AFTER INSERT OR UPDATE ON Rezultat
FOR EACH ROW        -- za vsako dodano ali spremenjeno vrstico
BEGIN
   UPDATE Rezultat
        SET Cas_Skupaj = Cas_Prvi_Tek + Cas_Drugi_Tek;
   WHERE NEW.Startna_stevilka = Rezultat.Startna_stevilka;
         END;                    -- :NEW za Oracle, NEW za MySQL
```

Vpisujemo samo neodvisna atribute: Cas_Prvi_Tek in Cas_Drugi_Tek, prožilec pa ob vsaki spremembi izračuna vsoto v spremenjeni vrstici.

Izračun atributa preko uporabe pogleda

- Osnovna relacijska shema:
  Rezultat (Startna stevilka, Ime, Priimek, Cas_Prvi_Tek, Cas_Drugi_Tek)

```
CREATE VIEW SkupniRezultat AS (
   SELECT Rezultat.*, Cas_Prvi_Tek + Cas_Drugi_Tek AS Cas_Skupaj
   FROM Rezultat
);
```

Vpisujemo samo neodvisna atribute: Cas_Prvi_Tek in Cas_Drugi_Tek, pogled pa ob vsaki uporabi izračuna vsote v vseh vrsticah.