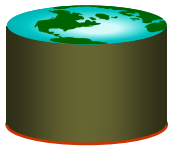
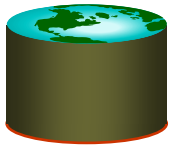
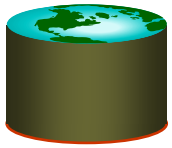


Dizajn Relacione Baze



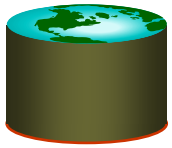
Dizajn Relacione Baze

- Karakteristike Dobrog Dizajna Relacione Baze
- Atomski Domeni i Prva Normalna Forma
- Dekompozicija Korišćenjem Funkcionalnih Zavisnosti
- Teorija Funkcionalnih Zavisnosti
- Dekompozicija Korišćenjem Viševrednosnih Zavisnosti
- Druge Normalne Forme
- Neki Aspekti Dizajna
- Modeliranje Privremenih Podataka



Date su Relacione Šeme

- Pretpostavljamo da je data relaciona šema *R*
  - *R* može biti generisano prevođenjem modela E-V u skup relacija-tabela.
  - *R* može biti jedna relacija koja sadrži *sve* atribute od interesa (takva relacija se naziva **univerzalna relacija**).
  - *R* može biti rezultat nekog ad hoc dizajna relacija.



Šema Bankarskog Sistema

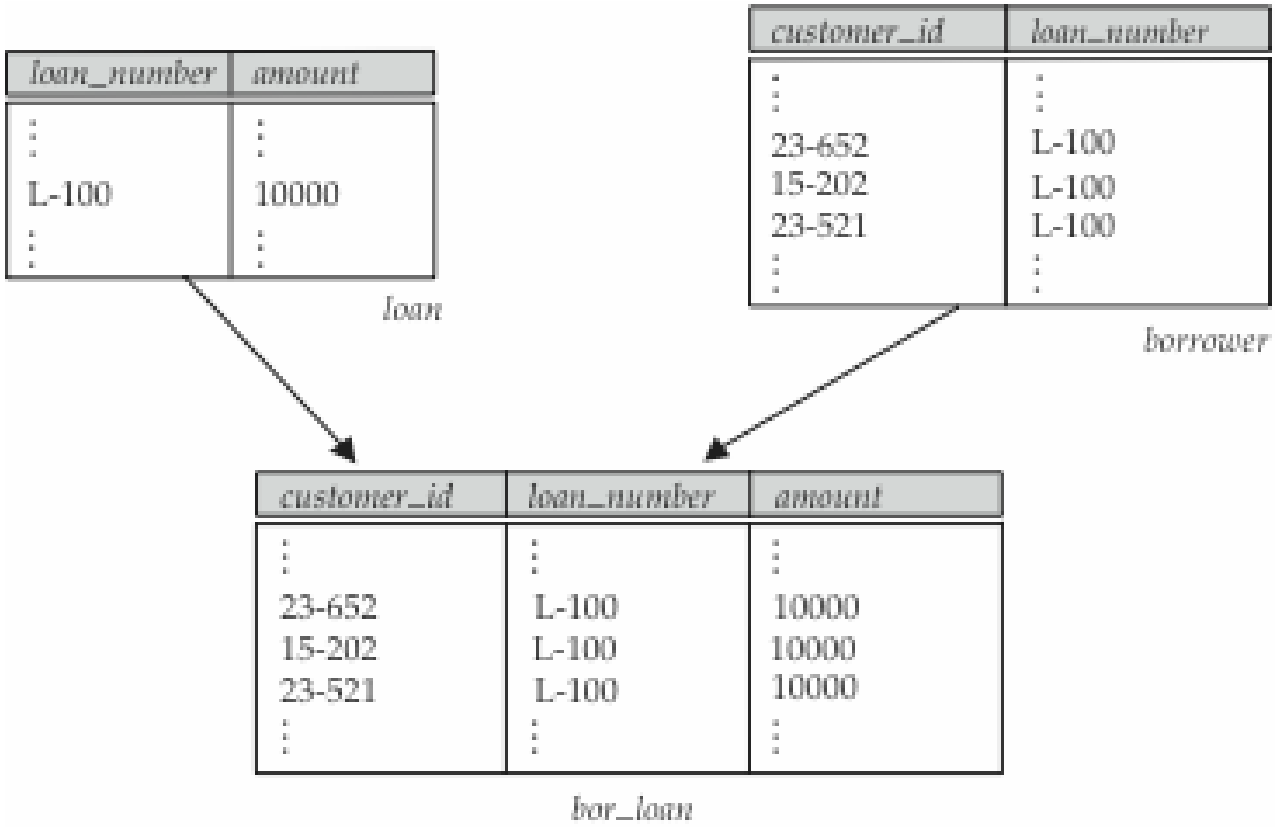
- *branch* = (*branch_name*, *branch_city*, *assets*)
- *customer* = (*customer_id*, *customer_name*, *customer_street*, *customer_city*)
- *loan* = (*loan_number*, *amount*)
- *account* = (*account_number*, *balance*)
- *employee* = (*employee_id*. *employee_name*, *telephone_number*, *start_date*)
- *dependent_name* = (*employee_id*, *dname*)
- *account_branch* = (*account_number*, *branch_name*)
- *loan_branch* = (*loan_number*, *branch_name*)
- *borrower* = (*customer_id*, *loan_number*)
- *depositor* = (*customer_id*, *account_number*)
- *cust_banker* = (*customer_id*, *employee_id*, *type*)
- *works_for* = (*worker_employee_id*, *manager_employee_id*)
- *payment* = (*loan_number*, *payment_number*, *payment_date*, *payment_amount*)
- *savings_account* = (*account_number*, *interest_rate*)
- *checking_account* = (*account_number*, *overdraft_amount*)

Spajanje Šema?

- Pretpostavimo da napravimo jednu relaciju od *borrower* i *loan*

  bor_loan = (*customer_id*, *loan_number*, *amount*)

- Rezultat je moguće ponavljanje informacija (L-100 u primeru)

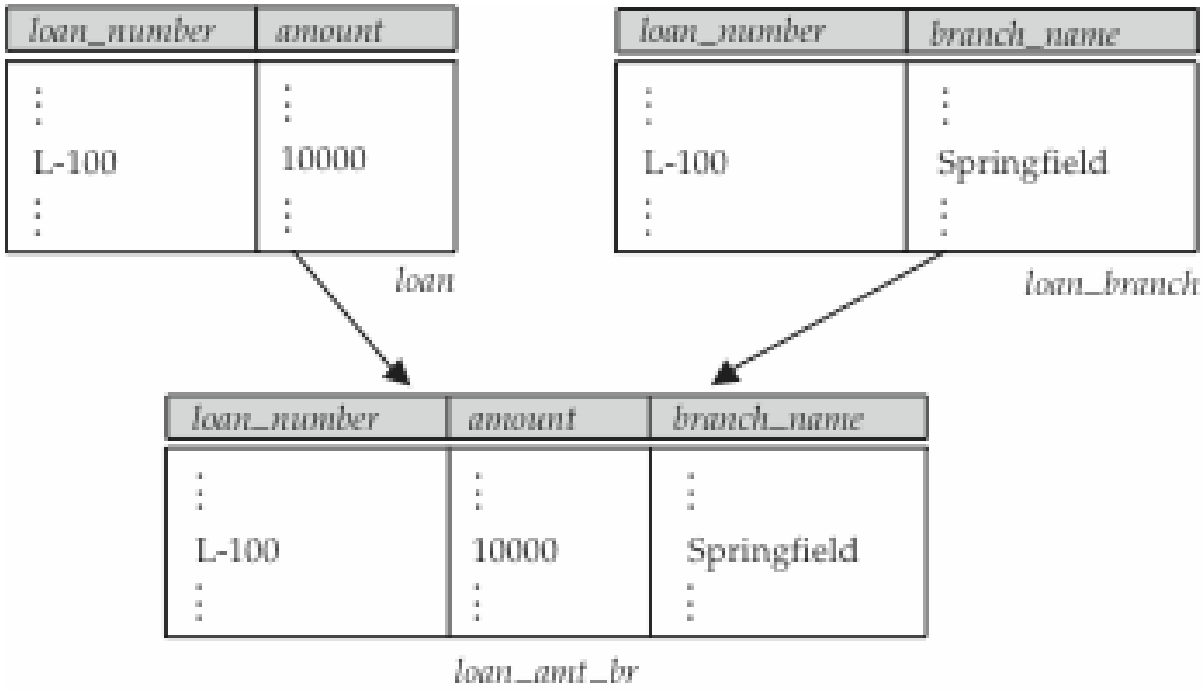


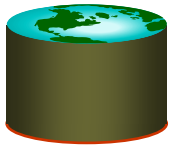
Kombinovana Šema bez Ponavljanja

- Pretpostavimo da napravimo jednu relaciju od *loan_branch* i *loan*

  loan_amt_br = (*loan_number*, *amount*, *branch_name*)

- Nema ponavljanja (primer)

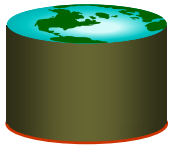




Spajanje Šema?

- Pretpostavimo da napravimo jednu relaciju od *borrower, loan, depositor i account*

  bor_loan_dep_acc = (*customer_id*, *loan_number*, *amount, account_number*, *balance*)
- Rezultat je moguće ponavljanje informacija.
- Rezultat je moguće neadekvatno predstavljanje informacija. Moramo uvesti null vrednosti.



Kako Dekomponovati?

- Pretpostavimo da startujemo sa *bor_loan.* Na osnovu čega odlučiti da treba da je **dekomponujemo** u *borrower* i *loan*?
- Posmatrajmo činjenicu "ako imamo šemu (*loan_number, amount*), tada je *loan_number* kandidat ključ"
- Nazovimo tu činjenicu **funkcionalna zavisnost**:

  $loan\_number \rightarrow amount$

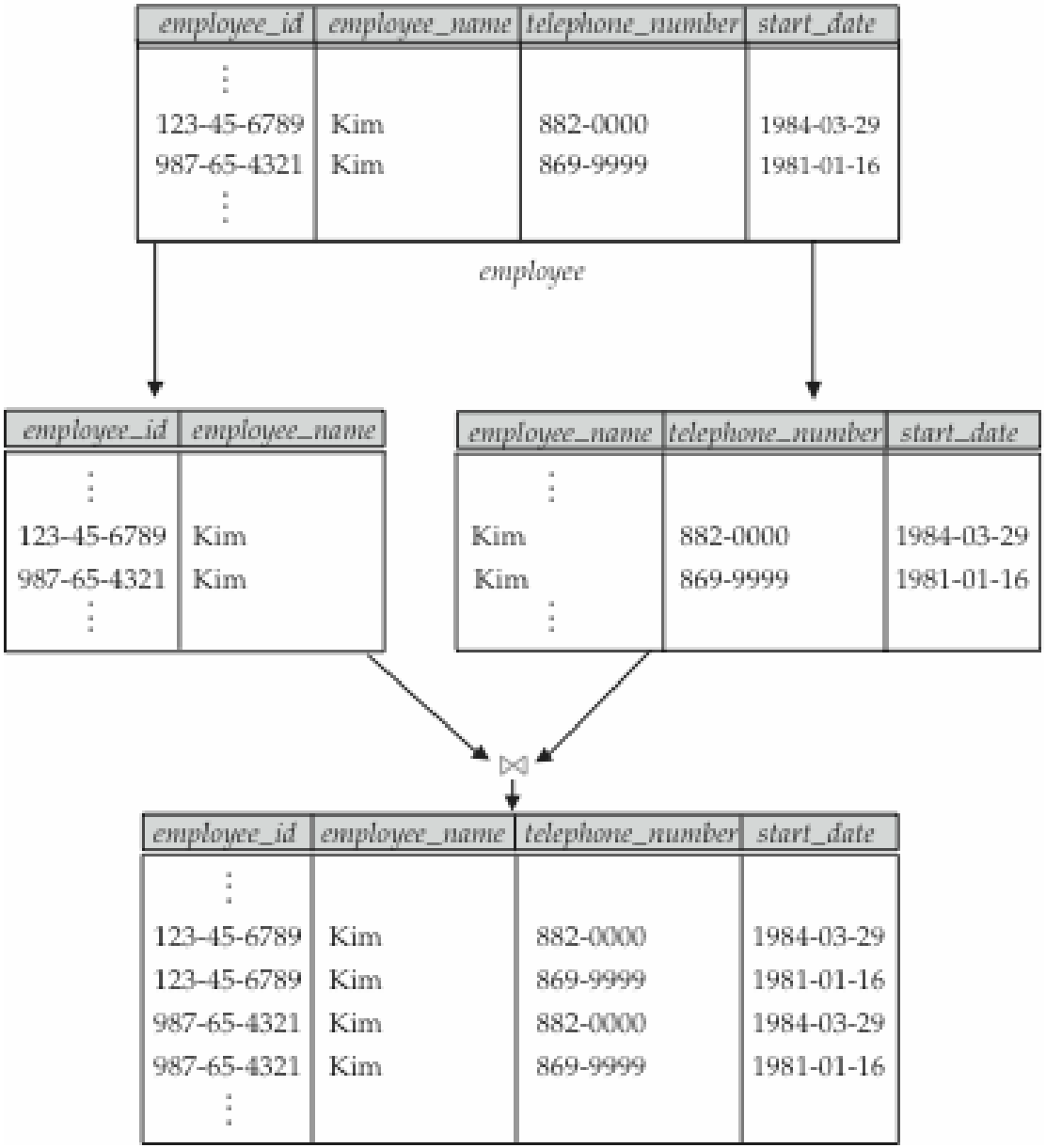
- U relaciji *bor_loan*, *loan_number* nije kandidat ključ, tako da se iznos kredita može ponavljati. To ukazuje na potrebu dekomponovanja relacije *bor_loan*.
- Međutim pri dekompoziciji se mogu pojaviti problemi. Pretpostavimo dekompoziciju relacije *employee* na dve relacije:

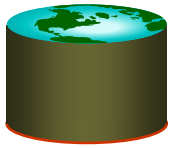
  employee1 = (*employee_id*, *employee_name*)

  employee2 = (*employee_name*, *telephone_number*, *start_date*)

- Dolazi do gubitka informacija – spajanjem se ne može rekonstruisati originalna relacija *employee* (primer na sledećem slajdu) – kažemo da je to **dekompozicija sa gubitkom pri spajanju**.

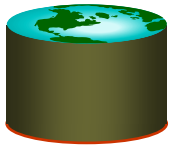
Dekompozicija sa Gubitkom pri Spajanju





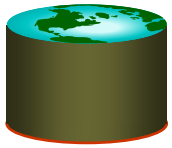
Prva Normalna Forma (1NF)

- Domeni atributa su atomski ako se njegovi elementi mogu smatrati nedeljivim celinama
  - Primeri ne-atomskih domena:
    - Skup imena, kompozitni atributi
    - Identifikatori koji se mogu podeliti na manje celine, npr RTI4BP
- Relaciona šema R je u Prvoj Normalnoj Formi ako su domeni svih atributa relacije R atomski.
- Ne-atomske vrednosti komplikuju memorisanje i uzrokuju redundantnost podataka
  - Primer: Skup računa memorisan sa klijentom, i skup vlasnika računa memorisan sa računom
  - Pretpostavljamo da su sve relacije u prvoj normalnoj formi



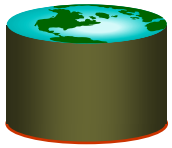
Prva Normalna Forma (1NF) - nastavak

- Da li je neki domen atomski ili ne ustvari zavisi od toga kako se elementi domena koriste.
  - Primer: Stringovi se normalno smatraju nedeljivim
  - Pretpostavimo da studenti dobijaju identifikatore predmeta u formi *CS0012* ili *EE1127*
  - Ako se prva dva karaktera koriste za identifikaciju odseka, domeni identifikatora predmeta nisu atomski.
  - To je loša praksa: informacije u aplikativnom programu umesto u bazi.



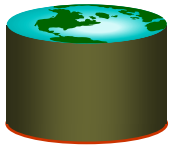
Cilj — Formalizam zasnovan na Teoriji

- Kako odrediti da li neka relacija *R* zadovoljava karakteristike dobrog dizajna (da li je u ”dobroj” formi)?
- U slučaju da ne zadovoljava, kako je dekomponovati na skup relacija $\{R_1, R_2, ..., R_n\}$ takvih da važi:
  - svaka relacija zadovoljava karakteristike dobrog dizajna
  - dekompozicija je bez gubitka pri spajanju
- Teorija se bazira na ograničenjima izraženim posredstvom:
  - Funkcionalnih zavisnosti (functional dependencies)
  - Viševrednosnih zavisnosti (multivalued dependencies)



Funkcionalne Zavisnosti

- Ograničenja na skupu legalnih relacija.
- Zahtevaju da su vrednosti nekog skupa atributa jednoznačno određene vrednostima drugog skupa atributa.
- Funkcionalna zavisnost je generalizacija pojma *ključa.*



Funkcionalne Zavisnosti (definicija)

- Neka je *R* relaciona šema, a α i β atributi ili skupovi atributa

$$\alpha \subseteq R \quad and \quad \beta \subseteq R$$

- Funkcionalna zavisnost

$$\alpha \rightarrow \beta$$

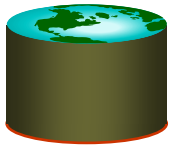
  važi na *R* ako i samo ako su, u svakoj legalnoj relaciji *r*(R), dve torke t_1 i t_2 koje su jednake na atributima α, takođe jednake i na atributima β,

$$t_1[\alpha] = t_2[\alpha] \quad \Rightarrow \quad t_1[\beta] = t_2[\beta]$$

- Primer: Posmatrajmo relaciju *r*(A,*B*) sa sledećim instancama:

1	4
1	5
3	7

- Na ovoj relaciji, $A \rightarrow B$ **NE** važi, ali $B \rightarrow A$ važi.



Funkcionalne Zavisnosti (nastavak)

- *K* je superključ relacione šeme *R* ako i samo ako važi $K \rightarrow R$
- *K* je kandidat ključ šeme *R* ako i samo ako važi
  - $K \rightarrow R$, i
  - ne postoji $\alpha \subset K$, $\alpha \rightarrow R$
- Funkcionalne zavisnosti omogućuju izražavanje ograničenja koja ne mogu biti izražena korišćenjem superključeva. Posmatrajmo šemu:

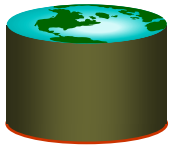
  bor_loan = (*customer_id, loan_number, amount*).

  Očekujemo da važi sledeća funkcionalna zavisnost:

  $$loan\_number \rightarrow amount$$

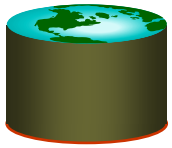
  a da ne važi:

  $$amount \rightarrow customer\_name$$



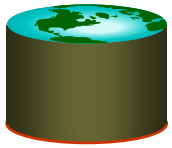
Funkcionalne Zavisnosti (nastavak)

- Funkcionalna zavisnost je trivijalna ako je zadovoljena za sve instance relacije
  - Primer:
    - *customer_name, loan_number* $\rightarrow$ *customer_name*
    - *customer_name* $\rightarrow$ *customer_name*
  - Generalno, funkcionalna zavisnost $\alpha \rightarrow \beta$ je trivijalna ako je $\beta \subseteq \alpha$



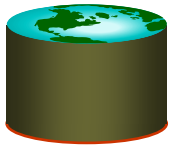
Korišćenje Funkcionalnih Zavisnosti

- Funkcionalne zavisnosti se koriste za:
  - testiranje legalnosti relacija na datom skupu funkcionalnih zavisnosti.
    - Ako je relacija *r* legalna na skupu *F* funkcionalnih zavisnosti, kažemo da *r* zadovoljava *F.*
  - specifikaciju ograničenja na skupu legalnih relacija
    - Kažemo da *F* važi na *R* ako sve legalne relacije na *R* zadovoljavaju skup funkcionalnih zavisnosti *F.*
- Primedba: Specifična instanca relacione šeme može zadovoljavati funkcionalnu zavisnost iako funkcionalna zavisnost ne važi na svim legalnim instancama.
  - Primer, specifična instanca relacije *loan* može zadovoljavati
    $amount \rightarrow customer\_name.$



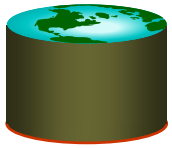
Zatvarač Skupa Funkcionalnih Zavisnosti

- Za dati skup funkcionalnih zavisnosti *F*, možemo naći i druge funkcionalne zavisnosti koje su logički implicirane skupom *F*.
  - Primer: Ako $A \rightarrow B$ i $B \rightarrow C$, tada važi i $A \rightarrow C$
- Skup svih funkcionalnih zavisnosti koje su logički implicirane skupom *F* se naziva *Zatvaračem* skupa *F*.
- *Zatvarač* skupa *F* se označava sa F^+.
- F^+ je nadskup skupa *F*.
- Zatvarač F^+ možemo naći primenom Armstrong-ovih Aksioma:
  1. Ako je $\beta \subseteq \alpha$, tada važi $\alpha \rightarrow \beta$ (refleksivnost(reflexivity))
  2. Ako važi $\alpha \rightarrow \beta$, tada važi $\gamma\ \alpha \rightarrow \gamma\ \beta$ (augmentativnost (augmentation))
  3. Ako važi $\alpha \rightarrow \beta$, i $\beta \rightarrow \gamma$, tada važi $\alpha \rightarrow \gamma$ (tranzitivnost (transitivity))
- Armstrong-ovi Aksiomi omogućuju nalaženje
  - samo važećih funkcionalnih zavisnosti i
  - svih važećih funkcionalnih zavisnosti.



Primer

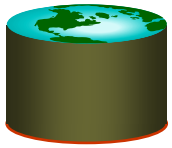
- $R = (A, B, C, G, H, I)$
  $F = \{ A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- Nalaženje nekih članova F^+
  - $A \rightarrow H$
    - primenom 3.: iz $A \rightarrow B$ i $B \rightarrow H$
  - $AG \rightarrow I$
    - primenom 2.: iz $A \rightarrow C$, dodavanjem G, dobijamo $AG \rightarrow CG$
      i primenom 3. sa $CG \rightarrow I$
  - $CG \rightarrow HI$
    - primenom 2.: iz $CG \rightarrow I$, dodavanjem CG, *dobijamo* $CG \rightarrow CGI$,
      i iz $CG \rightarrow H$, dodavanjem I, *dobijamo* $CGI \rightarrow HI$,
      i primenom 3.



Zatvarač (Nastavak)

- Sračunavanje F^+ se može dalje pojednostaviti korišćenjem dodatnih pravila:
  - Ako važi $\alpha \rightarrow \beta$ i $\alpha \rightarrow \gamma$, tada važi $\alpha \rightarrow \beta\gamma$ (unija)
  - Ako važi $\alpha \rightarrow \beta\gamma$, tada važi $\alpha \rightarrow \beta$ i $\alpha \rightarrow \gamma$ (dekompozicija)
  - Ako važi $\alpha \rightarrow \beta$ i $\gamma\ \beta \rightarrow \delta$, tada važi $\alpha\ \gamma \rightarrow \delta$ (pseudotranzitivnost)

  Ova pravila se mogu dobiti iz Armstrong-ovih aksioma.

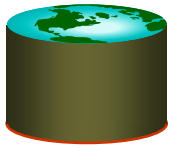


Zatvarač Skupa Atributa

- Zatvarač skupa atributa α, u oznaci α^+, na skupu funkcionalnih zavisnosti *F,* predstavlja skup atributa koji su funkcionalno zavisni od α, na skupu *F*.

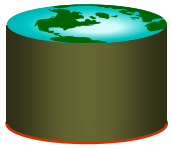
- Algoritam za sračunavanje α^+, na skupu *F*

result := α;
while (promene u *result*) **do**
  for each $\beta \rightarrow \gamma$ **in** *F* **do**
    begin
      if $\beta \subseteq$ *result* **then** *result* := *result* $\cup \gamma$
    end



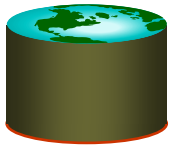
Primer

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. $result = AG$
  2. $result = ABCG$ $(A \rightarrow C$ i $A \rightarrow B)$
  3. $result = ABCGH$ $(CG \rightarrow H$ i $CG \subseteq AGBC)$
  4. $result = ABCGHI$ $(CG \rightarrow I$ i $CG \subseteq AGBCH)$
- Da li je AG kandidat ključ?
  1. Da li je AG super ključ?
     1. Da li važi $AG \rightarrow R$? == Da li je $(AG)^+ \supseteq R$
  2. Da li je neki podskup AG superključ?
     1. Da li važi $A \rightarrow R$? == Da li je $(A)^+ \supseteq R$
     2. Da li važi $G \rightarrow R$? == Da li je $(G)^+ \supseteq R$



Korišćenje Zatvarača Skupa Atributa

- Provera superključa:
  - Da bi proverili da li je α superključ, nalazimo α^+, i proveravamo da li α^+ sadrži sve atribute *R*.
- Provera funkcionalnih zavisnosti
  - Da bi proverili da li važi funkcionalna zavisnost $\alpha \rightarrow \beta$ (ili, drugim rečima, da li je u F^+), dovoljno je proveriti da li je $\beta \subseteq \alpha^+$.
  - Znači da umesto da nalazimo F^+, nalazimo α^+, i potom proveravamo da li sadrži β.
  - Nalaženje α^+ je znatno jednostavnije od nalaženja F^+.
- Nalaženje zatvarača F
  - Za svaki $\gamma \subseteq R$, nalazimo zatvarač γ^+, i za svako $S \subseteq \gamma^+$, generišemo funkcionalnu zavisnost $\gamma \rightarrow S$.



Boyce-Codd-ova Normalna Forma (BCNF)

Relaciona šema *R* je u BCNF u odnosu na skup *F* funkcionalnih zavisnosti ako za sve funkcionalne zavisnosti u F^+ oblika

$$\alpha \rightarrow \beta$$

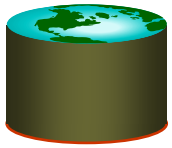
gde $\alpha \subseteq R$ i $\beta \subseteq R$, važi najmanje jedno od sledeća dva tvrđenja:

- $\alpha \rightarrow \beta$ je trivijalna (to jest., $\beta \subseteq \alpha$)
- α je superključ za *R*

Primer šeme *koja nije u* BCNF:

bor_loan = (*customer_id, loan_number, amount*)

Zato što važi *loan_number* $\rightarrow$ *amount* na *bor_loan* ali *loan_number* nije superključ



Dekompozicija Šema u BCNF

- Pretpostavimo šemu *R* i ne-trivijalnu zavisnost $\alpha \rightarrow \beta$ koja ne zadovoljava uslov za BCNF.

  Dekomponovaćemo relaciju *R* na sledeće relacije:

  - $(\alpha \cup \beta)$
  - $(R - (\beta - \alpha))$

- Za primer sa prethodnog slajda,

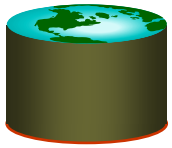
  bor_loan = (*customer_id, loan_number, amount*)

  loan_number $\rightarrow$ *amount* važi, ali *loan_number* nije superključ

  - α = *loan_number*
  - β = *amount*

  pa *bor_loan* dekomponujemo na

  - $(\alpha \cup \beta)$ = (*loan_number, amount*)
  - $(R - (\beta - \alpha))$ = (*customer_id, loan_number*)

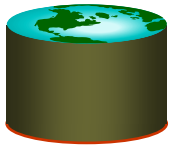


Dekompozicija bez gubitka pri spajanju

- Za dekompoziciju šeme $R = (R_1, R_2)$, za sve moguće relacije r na šemi R *mora da važi*

$$r = \Pi_{R1}(r) \bowtie \Pi_{R2}(r)$$

- Dekompozicija šeme R na šeme R_1 i R_2 je dekompozicija bez gubitka pri spajanju ako i samo ako je najmanje jedna od sledećih funkcionalnih zavisnosti u F^+:
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$



Primer

- $R = (A, B, C)$
  $F = \{A \rightarrow B, B \rightarrow C)$
  - Možemo izvršiti dekompoziciju na dva različita načina

1) $R_1 = (A, B), \quad R_2 = (B, C)$
   - Dekompozicija bez gubitka pri spajanju:

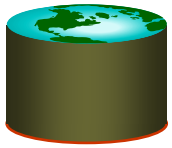
     $$R_1 \cap R_2 = \{B\} \text{ i } B \rightarrow BC$$

   - Očuvane zavisnosti

2) $R_1 = (A, B), \quad R_2 = (A, C)$
   - Dekompozicija bez gubitka pri spajanju:

     $$R_1 \cap R_2 = \{A\} \text{ i } A \rightarrow AB$$

   - Nisu očuvane zavisnosti
     (ne možemo proveriti $B \rightarrow C$ bez sračunavanja $R_1 \bowtie R_2$)

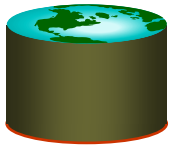


BCNF i Očuvanje Zavisnosti

- Provera ograničenja, uključujući i funkcionalne zavisnosti, je skupa ukoliko se ne može realizovati samo na jednoj relaciji.

- Kažemo da je dekompozicija sa *Očuvanjem Zavisnosti* ako je dovoljno testirati samo one zavisnosti koje važe na po samo jednoj relaciji da bi bili sigurni da važe sve funkcionalne zavisnosti.

- U opštem slučaju decompozicija u BCNF ne obezbeđuje očuvanje zavisnosti, pa se zadovoljavamo manje strogom normalnom formom, poznatom kao *treća normalna forma.*

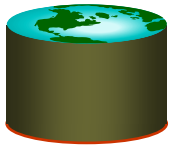


Očuvanje Zavisnosti

- Neka je F_i skup funkcionalnih zavisnosti iz F^+ koji uključuje samo atribute iz R_i.
  - Dekompozicija je sa očuvanjem zavisnosti, ako je

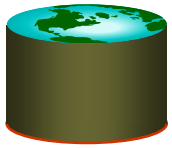
    $$(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$$

  - Ako nije, tada provera zadovoljenja funkcionalnih zavisnosti pri ažuriranju zahteva skupa prirodna spajanja.



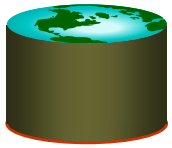
Provera Očuvanja Zavisnosti

- Za proveru očuvanja $\alpha \rightarrow \beta$ u dekompoziciji šeme R na R_1, R_2, …, R_n primenjujemo sledeći test (sa α^+ umesto F^+):
  - *result* = α
    while (promene u *result*) do
      for each R_i u dekompoziciji
        $t = (result \cap R_i)^+ \cap R_i$
        $result = result \cup t$
  - Ako *result* sadrži sve atribute u β, tada je funkcionalna zavisnost $\alpha \rightarrow \beta$ očuvana.
- Test primenjujemo na sve funkcionalne zavisnosti u F
- Algoritam je polinomijalne složenosti, umesto eksponencijalne složenosti za nalaženje F^+ i $(F_1 \cup F_2 \cup \ldots \cup F_n)^+$



Primer

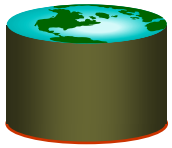
- $R = (A, B, C)$
  $F = \{A \rightarrow B$
  $B \rightarrow C\}$
  Ključ = $\{A\}$
- R nije u BCNF
- Dekompozicija $R_1 = (A, B)$, $R_2 = (B, C)$
  - R_1 i R_2 u BCNF
  - Dekompozicija bez gubitka pri spajanju
  - Očuvane zavisnosti



Algoritam Dekompozicije Šema u BCNF

result := {*R* };
done := false;
nalaženje F^+;
while (not *done)* **do**
  if (postoji šema R_i u *result* koja nije u BCNF)
    then begin
      neka je $\alpha \rightarrow \beta$ netrivijalna funkcionalna zavisnost koja važi na R_i
        takva da $\alpha \rightarrow R_i$ nije u F^+,
        i $\alpha \cap \beta = \varnothing$;
      result := (*result* – R_i) $\cup$ (R_i – β) $\cup$ (α, β);
      end
    else *done* := **true;**

Svaka R_i je u BCNF, i dekompozicija je bez gubitka pri spajanju.



Primer

- Originalna relacija *R* i *skup* funkcionalnih zavisnosti *F*

  R = (*branch_name, branch_city, assets,*

  customer_name, loan_number, amount)

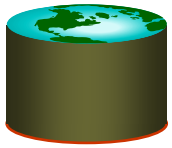
  F = {*branch_name* $\rightarrow$ *assets branch_city*

  loan_number $\rightarrow$ *amount branch_name* }

  Ključ = {*loan_number, customer_name*}

- Dekompozicija
  - R_1 = (*branch_name, branch_city, assets*)
  - R_2 = (*branch_name, customer_name, loan_number, amount*)
  - R_3 = (*branch_name, loan_number, amount*)
  - R_4 = (*customer_name, loan_number*)
- Finalna dekompozicija

  R_1, R_3, R_4



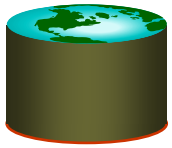
BCNF i Očuvanje Zavisnosti

Nije uvek moguće izvršiti dekompoziciju u BCNF i očuvati funkcionalne zavisnosti

- $R = (J, K, L)$
  $F = \{JK \rightarrow L$
  $L \rightarrow K\}$
  Dva kandidat ključa = JK i JL
- R nije u BCNF
- Bilo koja dekompozicija R ne obezbeđuje očuvanje

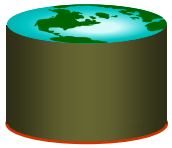
$$JK \rightarrow L$$

To implicira da moramo izvršiti spajanje da bi proverili funkcionalnu zavisnost $JK \rightarrow L$



Zašto Treća Normalna Forma (3NF)

- Kako postupiti u situacijama kada
  - BCNF ne obezbeđuje očuvanje zavisnosti, i
  - Efikasnost provere očuvanja zavisnosti je važna
- Rešenje: Definisati slabiju normalnu formu
  (Treća Normalna Forma (3NF))
  - Dozvoljavamo izvesno ponavljanje
  - Ali funkcionalne zavisnosti se mogu proveravati na pojedinačnim relacijama bez prirodnih spajanja.
  - Uvek postoji dekompozicija u 3NF bez gubitka pri spajanju i sa očuvanim zavisnostima.



Treća Normalna Forma (3NF)

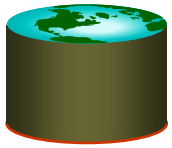
- Relaciona šema *R* je u Trećoj Normalnoj Formi (3NF) u odnosu na skup *F* funkcionalnih zavisnosti ako za sve funkcionalne zavisnosti u F^+ oblika

  $$\alpha \rightarrow \beta$$

  gde $\alpha \subseteq R$ i $\beta \subseteq R$, važi najmanje jedno od sledećih tvrđenja:

  - $\alpha \rightarrow \beta$ je trivijalna (to jest., $\beta \subseteq \alpha$)
  - α je superključ za *R*
  - Svaki atribut *A* u $\beta - \alpha$ je sadržan u kandidat ključu za *R.*

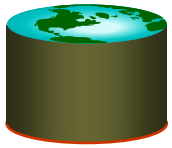
    (svaki atribut može biti u različitom kandidat ključu)
- Ako je relacija u BCNF tada je i u 3NF (pošto u BCNF jedan od prva dva uslova mora važiti).
- Treći uslov omogućava izvesno ponavljanje informacija u cilju očuvanja zavisnosti



Primer

- Relacija R:
  - $R = (J, K, L)$
    $F = \{JK \rightarrow L, L \rightarrow K\}$
  - Dva kandidat ključa: JK i JL
  - R je u 3NF

    $JK \rightarrow L$ JK je superključ
    $L \rightarrow K$ K je sadržano u kandidat ključu

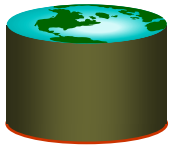


Primer

- Postoji ponavljanje u šemi R
- Primer
  - $R = (J, K, L)$
    $F = \{JK \rightarrow L, L \rightarrow K\}$

J	L	K
j_1	l_1	k_1
j_2	l_1	k_1
j_3	l_1	k_1
null	l_2	k_2

  - ponavljanje informacija (veza l_1, k_1)
  - Potreba za null vrednostima (predstavljanje veze l_2, k_2 ako nemamo odgovarajuću vrednost za J).

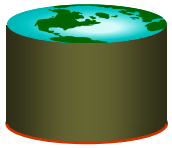


Algoritam Dekompozicije Šema u 3NF

Neka je F_c kanonički prekrivač *F;*
i := 0;
for each funkcionalnu zavisnost $\alpha \rightarrow \beta$ u F_c **do**
if nijedna od šema R_j, $1 \leq j \leq i$ ne sadrži $\alpha\ \beta$
then begin
i := *i* + 1;
R_i := $\alpha\ \beta$
end
if nijedna od šema R_j, $1 \leq j \leq i$ ne sadrži kandidat ključ *R*
then begin
i := *i* + 1;
R_i := neki kandidat ključ *R;*
end
return $(R_1, R_2, ..., R_i)$

Svaka R_i je u 3NF, i dekompozicija je bez gubitka pri spajanju i sa očuvanim zavisnostima.

*Kanonički prekrivač F je “minimalni” skup funkcionalnih zavisnosti ekvivalentan sa F, koji nema redundantnih zavisnosti ili redundantnih delova zavisnosti.



Primer

- Šema relacije:

  cust_banker_branch = (*customer_id, employee_id, branch_name, type*)

- Funkcionalne zavisnosti koje važe na ovoj šemi:

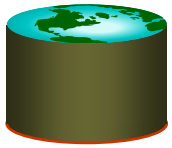
  customer_id, employee_id $\rightarrow$ *branch_name, type*
  employee_id $\rightarrow$ *branch_name*

- **for** petlja generiše:

  (*customer_id, employee_id, branch_name, type*)

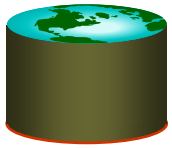
  if generiše (ali je podskup prve šeme pa ne ulazi u rezultat):

  (*employee_id, branch_name*)



Ciljevi Normalizacije

- Neka je *R* relaciona šema na kojoj važi skup *F* funkcionalnih zavisnosti.
- Proveriti da li je relaciona šema *R* u “dobroj” formi.
- U slučaju da relaciona šema *R* nije u “dobroj” formi, dekomponovati je na skup relacionih šema $\{R_1, R_2, ..., R_n\}$ takvih da važi
  - svaka relaciona šema je u dobroj formi
  - dekompozicija je bez gubitaka pri spajanju
  - Poželjno je da je dekompozicija sa očuvanjem zavisnosti.

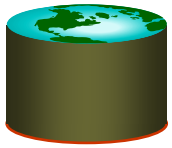


Koliko dobra je BCNF?

- Postoje šeme u BCNF koje ne izgledaju “dovoljno” normalizovane
- Posmatrajmo relaciju

  classes (*course, teacher, book*)

  takvu da (*c, t, b*) $\in$ *classes* znači da je nastavnik *t* kvalifikovan da predaje predmet *c,* i da je knjiga *b* predviđena knjiga za predmet *c*
- Za svaki predmet imamo skup nastavnika koji mogu predavati dati predmet i skup knjiga predviđenih za kurs, bez obzira ko ga predaje.



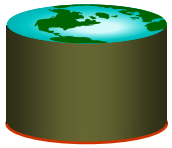
Koliko dobra je BCNF? (nastavak)

course	*teacher*	*book*
database	Avi	DB Concepts
database	Avi	DB Systems
database	Hank	DB Concepts
database	Hank	DB Systems
database	Mike	DB Concepts
database	Mike	DB Systems
operating systems	Avi	OS Concepts
operating systems	Avi	OS Basics
operating systems	Pete	OS Concepts
operating systems	Pete	OS Basics

classes

- Na ovoj relaciji ne važe ne-trivijalne funkcionalne zavisnosti pa je relacija u BCNF
- Anomalija pri unosu – ako želimo uneti novog nastavnika Mary koji može predavati baze podataka, moramo uneti dve torke

(database, Mary, DB Concepts)
(database, Mary, DB Systems)



Koliko dobra je BCNF? (nastavak)

- Bolji dizajn je ako dekomponujemo relaciju *classes* na:

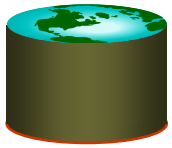
course	*teacher*
database	Avi
database	Hank
database	Mike
operating systems	Avi
operating systems	Jim

teaches

course	*book*
database	DB Concepts
database	DB Systems
operating systems	OS Concepts
operating systems	OS Basics

text

Ovo nas navodi na razmišljanje o potrebi za višim normalnim formama i novim mehanizmima izražavanja ograničenja.



Viševrednosne Zavisnosti

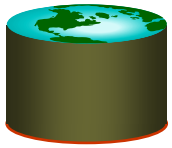
- Neka je R relaciona šema i $\alpha \subseteq R$ i $\beta \subseteq R$. Viševrednosna zavisnost (*multivalued dependency*)

$$\alpha \twoheadrightarrow \beta$$

važi na R ako u bilo kojoj legalnoj relaciji $r(R)$ u kojoj postoje parovi torki t_1 i t_2 takvi da je $t_1[\alpha] = t_2[\alpha]$, postoje i torke t_3 i t_4 u r takve da važi:

$$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$$
$$t_3[\beta] = t_1[\beta]$$
$$t_3[R - \beta] = t_2[R - \beta]$$
$$t_4[\beta] = t_2[\beta]$$
$$t_4[R - \beta] = t_1[R - \beta]$$

	α	β	$R - \alpha - \beta$
t_1	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$
t_2	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$b_{j+1} \ldots b_n$
t_3	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$
t_4	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$a_{j+1} \ldots a_n$



Primer

- Let R be a relation schema with a set of attributes that are partitioned into 3 nonempty subsets.

  $$Y, Z, W$$

- We say that $Y \rightarrow\rightarrow Z$ (Y multidetermines Z) if and only if for all possible relations r (R)

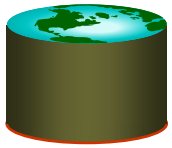
  $$< y_1, z_1, w_1 > \in r \text{ and } < y_2, z_2, w_2 > \in r$$

  then

  $$< y_1, z_1, w_2 > \in r \text{ and } < y_2, z_2, w_1 > \in r$$

- Note that since the behavior of Z and W are identical it follows that

  $Y \rightarrow\rightarrow Z$ if $Y \rightarrow\rightarrow W$

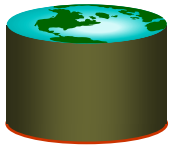


Primer

- In our example:

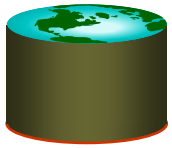
$$course \rightarrow\!\!\!\!\rightarrow \text{teacher}$$
$$course \rightarrow\!\!\!\!\rightarrow book$$

- The above formal definition is supposed to formalize the notion that given a particular value of *Y* (*course*) it has associated with it a set of values of *Z (teacher)* and a set of values of *W (book)*, and these two sets are in some sense independent of each other.
- Note:
  - If $Y \rightarrow Z$ then $Y \rightarrow\!\!\!\!\rightarrow Z$
  - Indeed we have (in above notation) $Z_1 = Z_2$
    The claim follows.



Korišćenje Viševrednosnih Zavisnosti

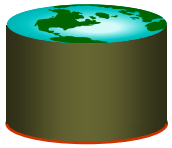
- We use multivalued dependencies in two ways:
  1. To test relations to determine whether they are legal under a given set of functional and multivalued dependencies
  2. To specify constraints on the set of legal relations. We shall thus concern ourselves *only* with relations that satisfy a given set of functional and multivalued dependencies.
- If a relation *r* fails to satisfy a given multivalued dependency, we can construct a relations *r′* that does satisfy the multivalued dependency by adding tuples to *r.*



Teorija Viševrednosnih Zavisnosti

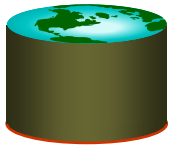
- From the definition of multivalued dependency, we can derive the following rule:
  - If $\alpha \rightarrow \beta$, then $\alpha \twoheadrightarrow \beta$

  That is, every functional dependency is also a multivalued dependency
- The **closure** D^+ of *D* is the set of all functional and multivalued dependencies logically implied by *D*.
  - We can compute D^+ from *D*, using the formal definitions of functional dependencies and multivalued dependencies.
  - We can manage with such reasoning for very simple multivalued dependencies, which seem to be most common in practice
  - For complex dependencies, it is better to reason about sets of dependencies using a system of inference rules.



Četvrta Normalna Forma (4NF)

- A relation schema *R* is in 4NF with respect to a set *D* of functional and multivalued dependencies if for all multivalued dependencies in D^+ of the form $\alpha \rightarrow\rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following hold:
  - $\alpha \rightarrow\rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)
  - α is a superkey for schema *R*
- If a relation is in 4NF it is in BCNF

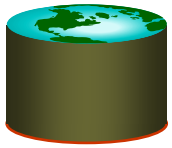


Restrikcija Viševrednosnih Zavisnosti

- The restriction of D to R_i is the set D_i consisting of
  - All functional dependencies in D^+ that include only attributes of R_i
  - All multivalued dependencies of the form

    $$\alpha \rightarrow\rightarrow (\beta \cap R_i)$$

    where $\alpha \subseteq R_i$ and $\alpha \rightarrow\rightarrow \beta$ is in D^+

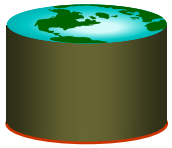


Algoritam Dekompozicije Šema u 4NF

result: = {*R*};
done := false;
compute D^+;
Let D_i denote the restriction of D^+ to R_i

while (**not** *done*)
  if (there is a schema $\mathbf{R}_i$ in *result* that is not in 4NF) **then**
    begin
      let $\alpha \rightarrow\rightarrow \beta$ be a nontrivial multivalued dependency that holds
        on R_i such that $\alpha \rightarrow R_i$ is not in D_i, and $\alpha \cap \beta = \phi$;
      result := (*result* - R_i) $\cup$ (R_i - β) $\cup$ (α, β);
    end
  else *done*:= true;

Note: each R_i is in 4NF, and decomposition is lossless-join



Primer

- $R = (A, B, C, G, H, I)$

  $F = \{ A \rightarrow\rightarrow B$

  $B \rightarrow\rightarrow HI$

  $CG \rightarrow\rightarrow H \}$

- R is not in 4NF since $A \rightarrow\rightarrow B$ and A is not a superkey for R
- Decomposition

  a) $R_1 = (A, B)$ (R_1 is in 4NF)

  b) $R_2 = (A, C, G, H, I)$ (R_2 is not in 4NF)

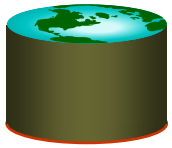
  c) $R_3 = (C, G, H)$ (R_3 is in 4NF)

  d) $R_4 = (A, C, G, I)$ (R_4 is not in 4NF)

- Since $A \rightarrow\rightarrow B$ and $B \rightarrow\rightarrow HI$, $A \rightarrow\rightarrow HI$, $A \rightarrow\rightarrow I$

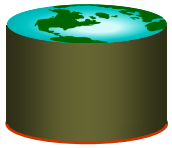
  e) $R_5 = (A, I)$ (R_5 is in 4NF)

  f) $R_6 = (A, C, G)$ (R_6 is in 4NF)



Ostale Normalne Forme

- **Join dependencies** generalize multivalued dependencies
  - lead to **project-join normal form (**PJNF) (also called **fifth normal form**)
- A class of even more general constraints, leads to a normal form called **domain-key normal form**.
- Problem with these generalized constraints: are hard to reason with, and no set of sound and complete set of inference rules exists.
- Hence rarely used



Denormalizacija zbog performansi

- May want to use non-normalized schema for performance
- For example, displaying *customer_name* along with *account_number* and *balance* requires join of *account* with *depositor*
- Alternative 1: Use denormalized relation containing attributes of *account* as well as *depositor* with all above attributes
  - faster lookup
  - extra space and extra execution time for updates
  - extra coding work for programmer and possibility of error in extra code
- Alternative 2: use a materialized view defined as
  account $\bowtie$ depositor
  - Benefits and drawbacks same as above, except no extra coding work for programmer and avoids possible errors