Elemente de proiectare BD (1)

Dependenţe Funcţionale

Decompoziţii

Forme Normale

Problema proiectării bazelor de date relaţionale

- Alegerea schemelor de relaţie şi modul de grupare al atributelor în relaţii pentru a reprezenta tipuri de entităţi sau legături între tipuri de entităţi.
- Există mai multe alegeri posibile, unele pot fi mai convenabile decât altele.
- Dependenţa datelor → Ideea centrală

  (influenţează proprietăţile relaţiilor în raport cu operaţiile curente)

Dependenţe Funcţionale

- $X \rightarrow Y$ este o declaraţie pentru relaţia R astfel încât oricând două tuple din R au aceleaşi valori pentru toate atributele ce formează X, atunci valorile din cele două tuple pentru toate atributele din setul Y trebuie să coincidă.
  - Se spune "Aserţiunea $X \rightarrow Y$ este valabilă în R."
  - X, Y, Z reprezintă seturi de atribute
  - A, B, C reprezintă atribute singulare

Divizarea membrului dreapta al unei DF

- $X \rightarrow A_1A_2...A_n$ este valabilă pentru R dacă şi numai dacă $X \rightarrow A_1$, $X \rightarrow A_2$,..., $X \rightarrow A_n$ sunt valabile pentru R.
- Exemplu: $A \rightarrow BC$ este echivalent cu $A \rightarrow B$ şi $A \rightarrow C$.
- Nu există regulă de divizare a membrului stânga al unei DF.
- În general DF au în partea dreapta un singur element.

Exemplu: DF

Drinkers(name, addr, beersLiked, manf, favBeer)

- Se poate gândi că:
  1. name → addr favBeer
     - Notă această FD este identică cu name → addr şi name → favBeer.
  2. beersLiked → manf

Exemplu: Date Posibile

name	addr	beersLiked	manf	favBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	Voyager	WickedAle	Pete's	WickedAle
Spock	Enterprise	Bud	A.B.	Bud

Deoarece name → addr

Deoarece name → favBeer

Deoarece beersLiked → manf

Chei ale Relaţiilor

- *K* este o *supercheie* pentru relaţia *R* dacă *K* determină funcţional toate atributele din *R*.
- *K* este o *cheie* pentru *R* dacă *K* este o supercheie, şi nici un subset al lui *K* nu este supercheie.

Exemplu: Supercheie

Drinkers(name, addr, beersLiked, manf, favBeer)

- {name, beersLiked} este o supercheie deoarece aceste atribute determină împreună toate celelalte atribute.
  - name → addr favBeer
  - beersLiked → manf

Exemplu: Cheie

- {name, beersLiked} este o cheie deoarece nici {name} nici {beersLiked} nu este o supercheie.
  - name NU → manf; beersLiked NU → addr.
- Nu există alte chei, dar există mai multe superchei.
  - Orice superset al {name, beersLiked}.

Deducerea DF

- Fiind date DF $X_1 \rightarrow A_1$, $X_2 \rightarrow A_2$,..., $X_n \rightarrow A_n$, presupunem că dorim să verificăm dacă o DF $Y \rightarrow B$ este valabilă în una din relaţiile ce satisfac DF date.
  - Exemplu: Dacă $A \rightarrow B$ şi $B \rightarrow C$ este valabilă, atunci $A \rightarrow C$ este valabilă.
- Este important pentru o bună proiectare a schemelor de relaţie.

Testul pentru Inferenţă

- Pentru a testa dacă $Y \rightarrow B$, se porneşte de la presupunerea că două tuple au aceleaşi valori pentru toate atributele ce formează Y.

← →

Y

00000**00**. . . 0

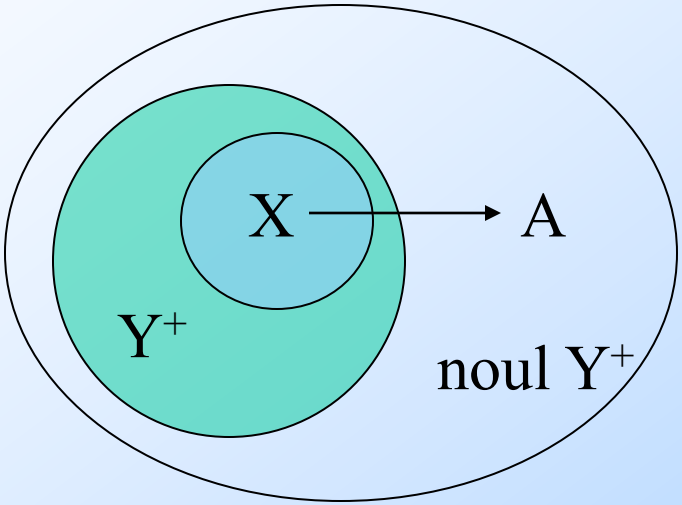
00000?? . . . ?

Testul pentru Inferenţă

- Se folosesc DF date pentru a infera că aceste tuple trebuie să aibă aceleaşi valori corespunzătoare altor atribute.
  - Dacă B este unul din aceste atribute, atunci $Y \rightarrow B$ este adevărat.
  - În caz contrar, cele două tuple, formează o relaţie formată din două tuple ce dovedeşte că $Y \rightarrow B$ nu este deductibilă din DF date.

Testul de Închidere Tranzitivă

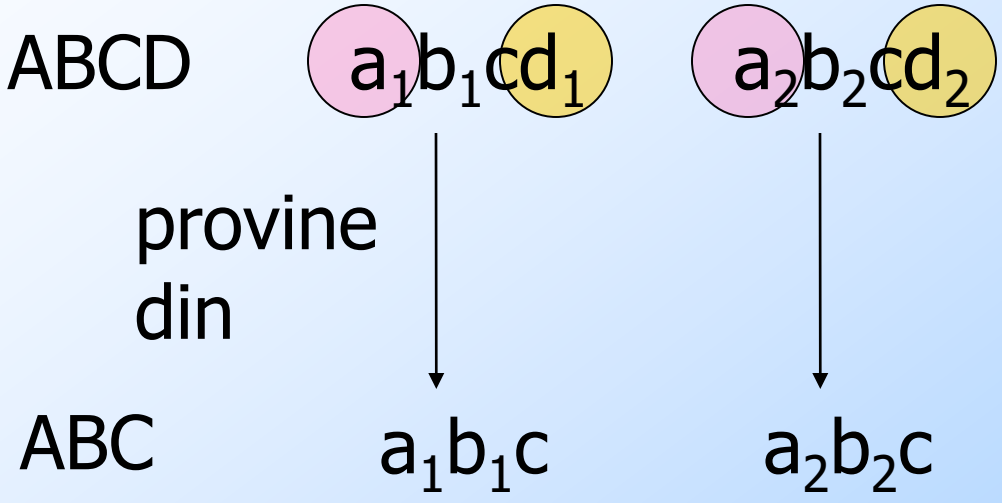
- O cale mai simplă de a testa inferența este calculul *închiderii* lui Y, notate Y^+.
- Ipoteza: $Y^+ = Y$.
- Inducţia: Se caută o DF ce are în stânga X ce este un subset al Y^+ curent. Dacă DF este $X \rightarrow A$, se adaugă A la Y^+.


X
A
Y^+
noul Y^+

Găsirea tuturor DF-le

- Motivaţie: "normalizarea," procesul prin care se "sparge" schema unei relaţii în două sau mai multe scheme.
- Exemplu: *ABCD* cu DF-le $AB \rightarrow C$, $C \rightarrow D$, şi $D \rightarrow A$.
  - Se decompune în *ABC*, *AD*. Ce DF-le sunt valabile în *ABC*?
  - Nu numai $AB \rightarrow C$, ci de asemenea $C \rightarrow A$!

Explicaţie



$d_1=d_2$ deoarece
$C \rightarrow D$

$a_1=a_2$ deoarece
$D \rightarrow A$

Aşadar, tuplele din proiecţie cu acelaşi C au acelaşi A;
$C \rightarrow A$.

Regula de Bază

1. Se începe cu DF date şi se caută toate DF *netriviale* ce pot fi deduse din DF date.
   - Netrivial = partea dreapta nu este conţinută în partea stânga.
2. Se păstrează doar acele DF ce implică atributele schemei după proiecţie.

Algoritm

1. Pentru fiecare set de atribute X, se calculează X^+.
2. Se adaugă $X \rightarrow A$ pentru toate A în $X^+ - X$.
3. Se elimină $XY \rightarrow A$ dacă $X \rightarrow A$.
   - Deoarece $XY \rightarrow A$ se deduce din $X \rightarrow A$ prin orice proiecţie.
4. În final, se folosesc doar DF ce implică atributele proiecţiei.

Trucuri

- Nu e nevoie să se calculeze închiderea tranzitivă pentru setul vid sau pentru setul format din toate atributele.
- Dacă X^+ = toate atributele, este închiderea tranzitivă a oricărui superset al lui X.

Exemplu: Deducerea DF

- ABC cu DF $A \rightarrow B$ şi $B \rightarrow C$. Se deduce DF AC.
  - $A^+ = ABC$; conduce la $A \rightarrow B$, $A \rightarrow C$.
    - Nu este nevoie să se calculeze AB^+ sau AC^+.
  - $B^+ = BC$; conduce la $B \rightarrow C$.
  - $C^+ = C$; nu conduce la nimic.
  - $BC^+ = BC$; nu conduce la nimic.

Axiomele lui Armstrong

- Reflexivitate (trivială)

  Dacă $Y \subseteq X$ atunci $X \rightarrow Y$

  Exemplu: (Nume, Adresa) → Nume

- Augmentare (trivială)

  Dacă $X \rightarrow Y$ atunci $X \cup Z \rightarrow Y \cup Z$

  Exemplu: (Nume,Produs) → (Adresa,Produs)

- Tranzitivitate

  Dacă $X \rightarrow Y$ și $Y \rightarrow Z$ atunci $X \rightarrow Z$

Proiectarea Schemei Relaţionale

- Obiectivul proiectării schemei relaţionale este evitarea anomaliilor şi a redundanţei.
  - *Anomalia la Adăugare*: nu se poate adăuga un fapt valid până ce nu se adaugă o tuplă
  - *Anomalia la Modificare*: una din apariţiile unui fapt se modifică, dar nu toate aparițiile
  - *Anomalia la Ştergere*: un fapt valid se pierde când se şterge o tuplă

Exemplu de Proiectare proastă

Drinkers(name, addr, beersLiked, manf, favBeer)

name	addr	beersLiked	manf	favBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	???	WickedAle	Pete's	???
Spock	Enterprise	Bud	???	Bud

Datele sunt redundante, deoarece fiecare poziţie ??? poate fi exprimată folosind DF name → addr favBeer şi beersLiked → manf.

Anomaliile la proiectare greşită

name	addr	beersLiked	manf	favBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	Voyager	WickedAle	Pete's	WickedAle
Spock	Enterprise	Bud	A.B.	Bud

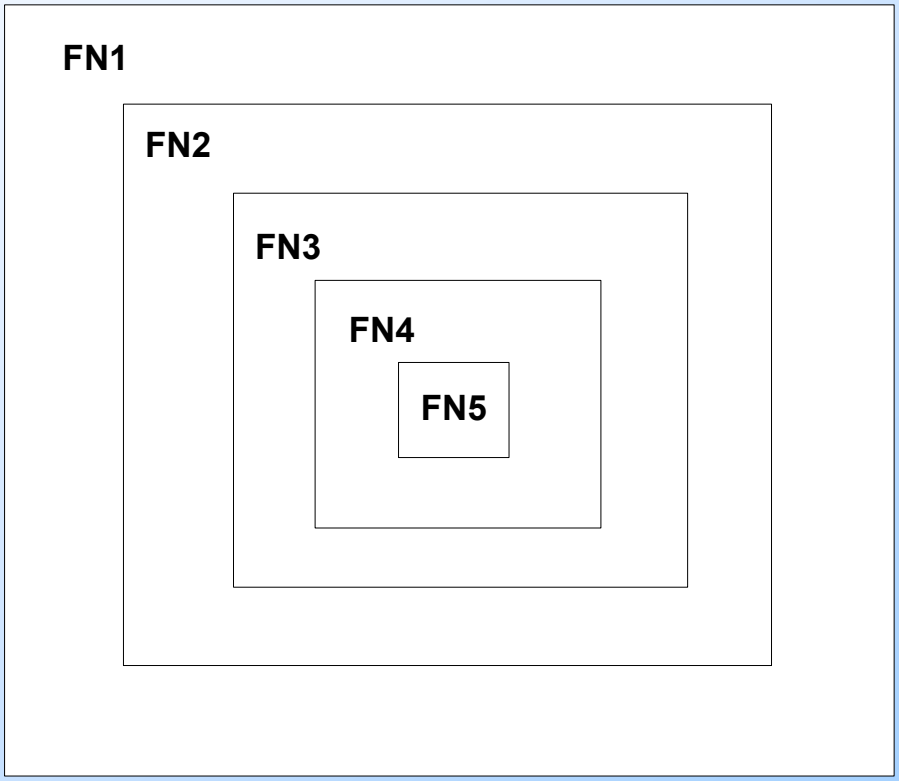
Anomalia la Adăugare : nu se poate menționa existența unei mărci noi de bere până ce nu se cunoaște o persoană căreia să îi placă berea.

Anomalia la Modificare : dacă Janeway își schimbă adresa la *Intrepid*, se va ţine minte să se modifice fiecare tuplă unde apare ea?

Anomalia la Ştergere: dacă nimănui nu-i place Bud, se pierde Faptul că Anheuser-Busch fabrică Bud.

Forme Normale

Relatii nenormalizate



FN1

Definiție: O relație R este în **FN1** dacă și numai dacă toate atributele sale iau valori atomice.

- În modelul relațional, prin definiție, toate atributele unei relații R sunt definite pe domenii ce conțin elemente atomice.
- Toate tuplele unei relații au același număr de câmpuri și aceeași dimensiune.

FN1

Drinkers(name, addr, beersLiked, manf, favBeer)

- Relația Drinkers este în FN1, ar fi nenormalizată dacă s-ar permite ca atributul addr să înregistreze (Localitate, Strada, Numar)

FN2

Definiție: O relație R este în **FN2** dacă este în FN1 și orice atribut neprim este total dependent față de orice cheie a relației.

- FN2 rezolvă problemele cauzate de dependențele între atribute prime și cele neprime.

Drinkers(name, addr, beersLiked, manf, favBeer)

cheia este {name, beersLiked}

DF-le: name → addr favBeer, beersLiked → manf

Drinkers NU este în FN2.

FN3

Definiție: O relație R este în **FN3** dacă este în FN2 și nici un atribut neprim nu este dependent față de un alt atribut neprim.

- Elimină anomaliile la adăugare, modificare, ștergere.

Beers(name, manf, addr_manf)

Beers este în FN2, dar NU este în FN3 (name este cheie, există DF manf → addr_manf).

Forma Normală Boyce-Codd

Definiţie: Se spune că relaţia *R* este în **FNBC** dacă totdeauna când $X \rightarrow Y$ este o DF netrivială valabilă pentru *R*, *X* este o supercheie.

- Ne reamintim: *netrivial* înseamnă *Y* nu este conţinut în *X*.
- Ne reamintim, o *supercheie* este orice superset al unei chei (nu neapărat unul anume).

Exemplu 1

Drinkers(name, addr, beersLiked, manf, favBeer)

DF-le: name → addr favBeer, beersLiked → manf

- Singura cheie este {name, beersLiked}.
- În fiecare DF, partea stânga *nu este* o supercheie.
- Oricare din DF-le demonstrează că *Drinkers* nu este în FNBC

Exemplu 2

Beers(name, manf, manfAddr)

DF-le: name → manf, manf → manfAddr

- Singura cheie este {name} .
- Name → manf nu violează FNBC, dar manf → manfAddr da.

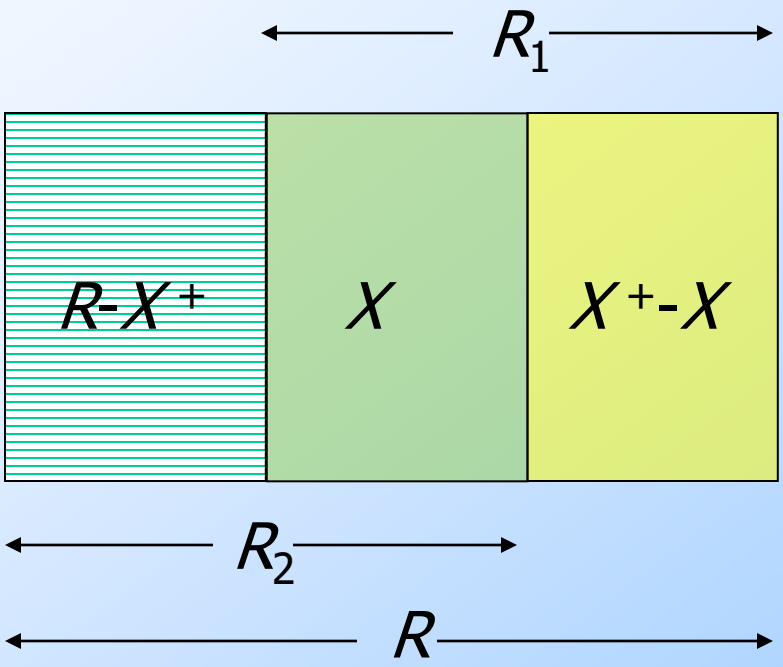
Descompunerea în BCNF

- Se dă: relaţia R şi DF-le F.
- Se caută DF-le $X \rightarrow Y$ ce violează FNBC.
  - Dacă există DF în F ce violează FNBC, atunci R necesită descompunere.
- Se calculează X^+.
  - Nu trebuie să conţină toate atributele, altfel X este o supercheie.

Descompunerea lui R Folosind $X \rightarrow Y$

- Se înlocuieşte R cu relaţiile ce au schemele:
  1. $R_1 = X^+$.
  2. $R_2 = R - (X^+ - X)$.
- *Se deduc* DF-le date F pentru cele două noi relaţii.

Descompunerea lui R



Exemplu: Descompunere FNBC

Drinkers(name, addr, beersLiked, manf, favBeer)

F = name → addr, name → favBeer,
beersLiked → manf

- Se alege una din DF ce violează FNBC
  name → addr.
- Se obţine închiderea tranzitivă a părţii stânga:
  $\{name\}^+ = \{name, addr, favBeer\}$.
- Se descompune relaţia în două relaţii:
  1. Drinkers1(name, addr, favBeer)
  2. Drinkers2(name, beersLiked, manf)

Exemplu: Descompunere FNBC

- Nu am terminat; trebuie verificat dacă Drinkers1 şi Drinkers2 respectă FNBC.
- Deducerea DF-le este simplă.
- Pentru Drinkers1(name, addr, favBeer), DF-le relevante sunt name → addr şi name → favBeer.
  - Aşadar, {name} este singura cheie şi Drinkers1 este în FNBC.

Exemplu: Descompunere FNBC

- Pentru Drinkers2(name, beersLiked, manf), singura DF este beersLiked → manf, şi singura cheie este {name, beersLiked}.
  - Ce violează FNBC.
- $beersLiked^+$ = {beersLiked, manf}, aşa că se descompune *Drinkers2* în:
  1. Drinkers3(beersLiked, manf)
  2. Drinkers4(name, beersLiked)

Exemplu: Descompunere FNBC

- Descompunerea rezultată pentru *Drinkers* :
  1. Drinkers1(name, addr, favBeer)
  2. Drinkers3(beersLiked, manf)
  3. Drinkers4(name, beersLiked)
- De notat: *Drinkers1* dă informaţii despre persoane, *Drinkers3* dă informaţii despre mărci de bere, iar *Drinkers4* dă informaţii despre relaţia de legătură între persoane şi mărcile de bere cu semnificaţia "îi place".

FN3 - Motivaţie

- Există o structură de DF-le ce cauzează probleme la descompunere.
- $AB \rightarrow C$ şi în acelaşi timp $C \rightarrow B$.
  - Exemplu: *A* = adresa_nr, *B* = adresa_strada, *C* = cod_postal.
- Există două chei, $\{A,B\}$ şi $\{A,C\}$.
- $C \rightarrow B$ este o violare FNBC, deci trebuie descompusă în *AC*, *BC*.

DF-le Nu pot fi Forţate

- Problema este că dacă se foloseşte *AC* şi *BC* pentru shema BD, DF $AB \rightarrow C$ nu poate fi forţată din DF-le din aceste relaţii rezultate în urma descompunerii.
- Exemplu cu *A* = street, *B* = city, şi *C* = zip.

DF ce Nu poate fi Forţată

street	zip
545 Tech Sq.	02138
545 Tech Sq.	02139

city	zip
Cambridge	02138
Cambridge	02139

Se face join pentru tuplele cu zip egal.

street	city	zip
545 Tech Sq.	Cambridge	02138
545 Tech Sq.	Cambridge	02139

Deşi nici una din DF-le nu a fost violată în relaţiile după descompunere, DF street city → zip este violată de BD ca un întreg.

FN3 Evită Această Problemă

- A treia Formă Normală (FN3) acţionează asupra condiţiei FNBC astfel ca descompunerea să nu fie necesară când se iveşte problema de mai sus.
- Un atribut este *prim* dacă este membru al unei chei.
- $X \rightarrow A$ violează FN3 dacă şi numai dacă X nu este o supercheie, şi A nu este prim.

Exemplu: FN3

- La exemplul precedent DF-le sunt $AB \rightarrow C$ şi $C \rightarrow B$, iar cheile sunt AB şi AC.
- Prin urmare A, B, şi C sunt fiecare prime.
- Deşi $C \rightarrow B$ violează BCNF, nu violează FN3.

Proprietăţile unei Descompuneri

1. *Cuplarea fără pierdere de informaţii* : relaţiile originale trebuie să poată fi obţinute din schema descompusă, adică să se reconstruiască originalul.
2. *Conservarea DF-le* : relaţiile obţinute prin descompunere trebuie să satisfacă toate DF-le iniţiale.

Proprietăţile unei Descompuneri

- Se poate obţine (1) la o descompunere FNBC.
- Se pot obţine ambele (1) şi (2) la o descompunere FN3.
- Nu totdeauna este posibil să se obţină ambele (1) şi (2) la o descomunere FNBC.
  - Exemplul street-city-zip demonstrează acest fapt.

Testul "Pierdere de Informaţii"

- Dacă se aplică proiecţia lui R în R_1, R_2,..., R_k, se poate obţine R prin join?
- Orice tuplă din R poate fi obţinută din fragmentele proiecţiei.
- Singura întrebare este: la join, este posibil să obţinem ceva ce nu exista în original?

Testul "Pierdere de Informaţii"

- Presupunem că tupla t apare în urma join.
- Deci t este rezultatul join al unor tuple din proiecţiile lui R, câte una din fiecare R_i ce formează descompunerea.
- Se pot folosi DF-le date pentru a arăta că una din aceste tuple trebuie să fie t ?

Testul "Pierdere de Informaţii"

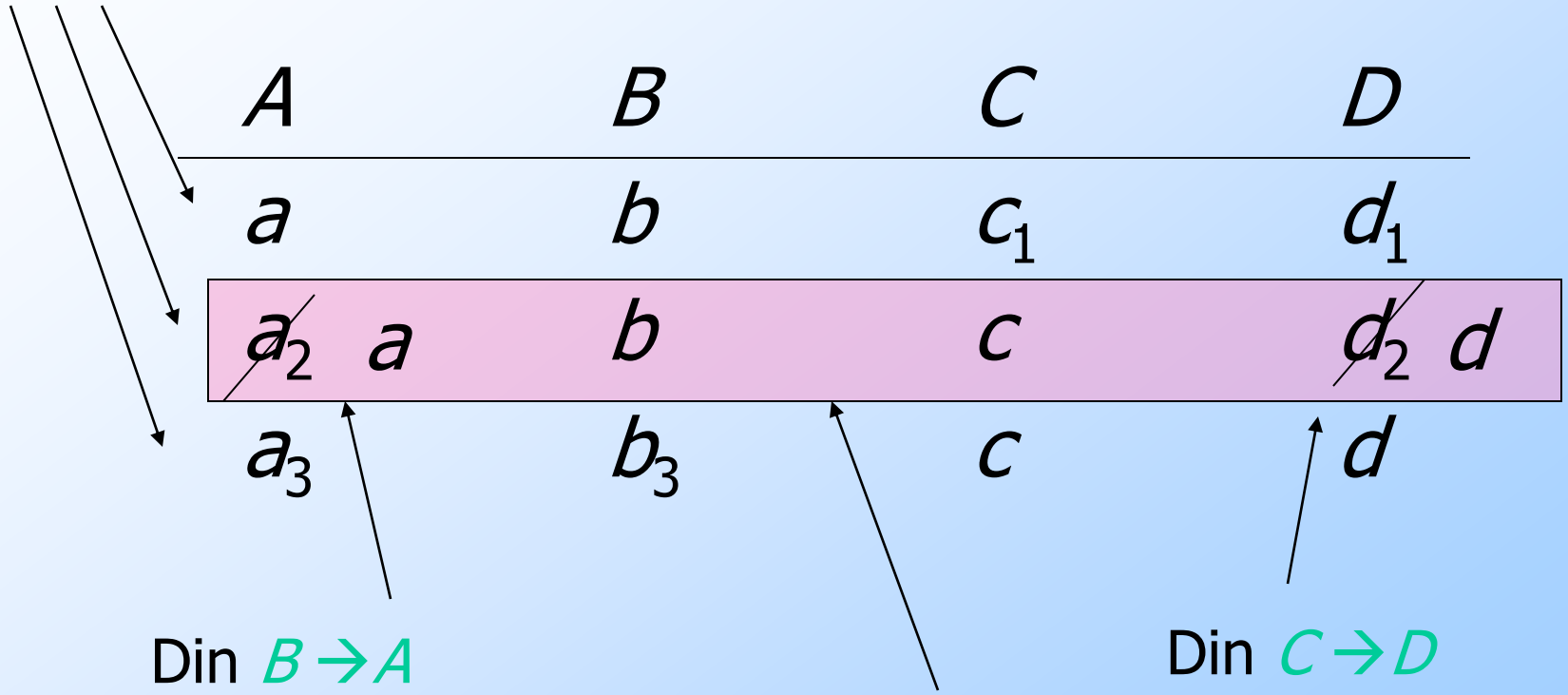
- Se începe prin a presupune $t = abc\ldots$.
- Pentru fiecare i, există o tuplă s_i din R ce are a, b, c,... printre atributele lui R_i.
- s_i poate avea orice valori pentru alte atribute.
- Se foloseşte aceeaşi literă ca în t, dar cu un indice al componentei.

Exemplu: Testul

- Fie $R = ABCD$, şi descompunerea AB, BC, şi CD.
- Fie DF-le $C \rightarrow D$ şi $B \rightarrow A$.
- Presupunem că tupla t = abcd este join-ul tuplelor proiecţiilor AB, BC, CD.

Tabloul

Tuplele din R pro-iectate pe AB, BC, CD.



S-a demonstrat că tupla a doua trebuie să fie *t*.

Testul "Pierdere de Informaţii" Concluzia

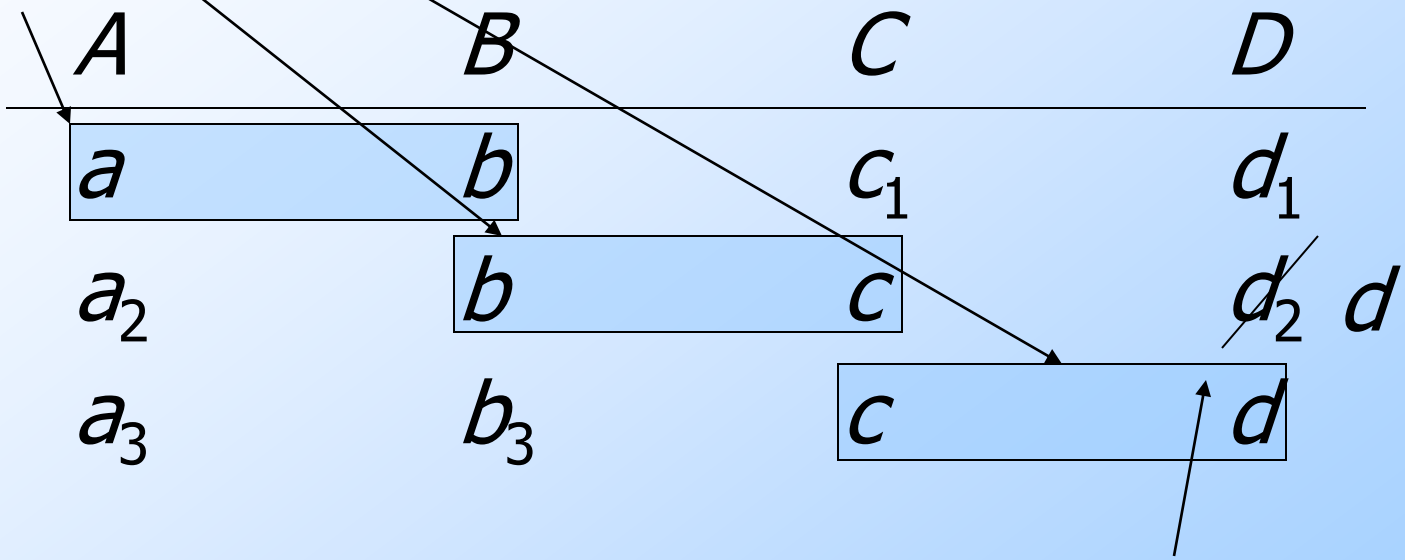
1. Dacă două tuple au aceeaşi parte stânga a unei DF, atunci pentru partea dreapta se face să fie de asemenea egale.
2. Totdeauna se înlocuieşte un simbol cu indice prin simbolul corespondent fără indice, dacă este posibil.
3. Atunci când se ajunge la o tuplă fără indice, se ştie că orice tuplă din project-join este în original (join fără pierdere de informaţii).
4. În caz contrar, tabloul final este exemplu negativ.

Exemplu: Join cu Pierdere

- Aceeaşi relaţie $R = ABCD$ şi aceeaşi descompunere.
- Considerăm singura DF $C \rightarrow D$.

Aceste proiecţii
se cuplează pentru
a forma *abcd*.

Tabloul



Din $C \rightarrow D$

Aceste trei tuple sunt un exemplu
R ce arată ce înseamnă join cu
pierdere de informaţii. *abcd*
nu există în *R*, dar prin proiecţie
şi apoi prin cuplare se obţine *abcd*.

Descompunere FN3
Concluzii

- Se poate construi totdeauna o decompoziţie formată din relaţii FN3 fără pierdere de informaţii şi conservarea DF-le iniţiale.
- Este nevoie de *condiţii minime* pentru DF-le:
  1. Părţile dreapta să fie un singur atribut.
  2. Nici o DF nu poate fi eliminată.
  3. Nici un atribut nu poate fi eliminat din partea stânga.

Descompunere FN3
Concluzii

1. Se “sparg” părţile dreapta.
2. Se încearcă să se elimine în mod repetat câte o DF şi se verifică dacă DF-le rămase sunt echivalente cu originalul.
3. Se încearcă să se elimine în mod repetat câte un atribut din partea stânga şi se verifică dacă DF-le rezultate sunt echivalente cu originalul.

Descompunere FN3 Concluzii

- O relaţie pentru fiecare DF în condiţiile minimale.
  - Schema este uniunea părţilor stânga şi dreapta.
- Dacă nici o cheie nu este conţinută într-o DF, atunci se adaugă o relaţie a cărei schemă este o cheie.

Exemplu: Descompunere FN3

- Relaţia R = ABCD.
- DF-le $A \rightarrow B$ şi $A \rightarrow C$.
- Descompunerea: AB şi AC din DF-le, plus AD pentru o cheie.

Dependențe multivalorice

Fie schema de relație R(X, Y, Z), unde X, Y și Z sunt atribute simple sau compuse. Se notează x, y și z valori ale atributelor X, Y și Z (eventual și cu indici).

Definiție: Se spune că există o dependență multivalorică a atributului Z față de atributul Y sau că Y multidetermină pe Z și se notează $Y \rightarrow\rightarrow Z$, dacă pentru orice valori x_1, x_2, y, z_1, z_2, $x_1 \neq x_2$, $z_1 \neq z_2$ astfel încât dacă tuplele (x_1, y, z_1), (x_2, y, z_2) fac parte din R, atunci și tuplele (x_2, y, z_1), (x_1, y, z_2) fac parte din R.

Dependențe multivalorice

- Din simetria definiției rezultă că dependența Y→→Z implică existența dependenței multivalorice Y→→X.
- O consecință este că între X și Z nu poate exista o dependență funcțională.
- O dependență funcțională este și o dependență multivalorică (dependențele multivalorice generalizează dependențele funcționale).

Dependențe multivalorice

Exemplu:

AGENTII(Nume_A, Nume_F, Nume_B)

Nume_A →→ Nume_F

Nume_A →→ Nume_B

dacă și numai dacă numele oricărei femei din evidența unei agenții este asociat cu numele oricărui bărbat din evidența aceleiași agenții.

Dependențe multivalorice

Dependențele multivalorice depind de context:

APROVIZIONARE(Beneficiar, Furnizor, Produs, Pret)

Dacă nu ar exista Pret, atunci

Beneficiar →→ Furnizor și Beneficiar →→ Produs

Dar în prezența acestuia dependențele depind de legăturile Pret cu celelalte atribute.

- Prețul unui produs este același la toți furnizorii

  Beneficiar →→ Furnizor și Beneficiar →→ (Produs, Pret)

- Prețul unui produs diferă de la furnizor la furnizor, deci există dependența functională (Furnizor, Produs) → Pret

  Beneficiar →→ (Furnizor, Produs, Pret)

  (irelevantă pentru problema modelată)

FN4

Este o generalizare a FNBC.

Definiție: O relație R este în **FN4** dacă oricare ar fi dependența multivalorică X →→ Y, unde Y nu este un subset a lui X și X ∪ Y nu conține toate atributele lui R, atunci atributul X (simplu sau compus) este o cheie sau conține o cheie a lui R.

Ca și în cazul trecerii de la FN3 la FNBC, descompunerea în relații FN4 se poate face fără pierdere de informații dar nu totdeauna se conservă dependențele multivalorice și/sau funcționale.

FN4

Exemplu:

CURSURI(Profesor, Disciplina, Limba)

Se presupune că un profesor poate susține cursuri la mai multe discipline și cunoaște mai multe limbi, prin urmare un curs poate fi susținut în toate limbile cunoscute de un anumit profesor.

Profesor →→ Disciplina

Profesor →→ Limba

FN4

PROFESORI

Profesor	Disciplina	Limba
Pop	Programare	Engleza
Costea	Baze de Date	Engleza
Costea	Baze de Date	Franceza
Costea	Baze de Date	Germana
Costea	Sisteme de Operare	Engleza
Costea	Sisteme de Operare	Franceza
Costea	Sisteme de Operare	Germana
Popovici	Programare	Franceza
Popovici	Programare	Germana

PD(Profesor, Disciplina)

PL(Profesor, Limba)

FN4

Cheia este {Profesor, Disciplina, Limba}.

Nu există dependențe funcționale, deci PROFESORI este în FNBC.

Deoarece nu este în FN4 există următoarele anomalii:

- Adăugare: dacă profesorul Costea acceptă să țină cursul "Programare", trebuie introdusă câte o tuplă pentru fiecare limbă cunoscută de acesta
- Ștergere: dacă se șterge singura disciplină predată de un profesor se pierd informațiile referitoare la limbile cunoscute de acesta
- Modificare: trebuie modificate mai multe tuple

FNPJ (FN5)

Dacă relația R este în FNBC și toate cheile sunt simple atunci poate fi garantată FN5. Prima condiție poate fi relaxată la FN3 (în loc de FNBC).

Demonstrație:

- dacă relația R este în FN3 și toate cheile sunt simple atunci relația R este în FNBC
- dacă relația R nu este în FN5, atunci este o DJ (dependență join)

  $\{X_1, ..., X_m\}$ a schemei ce nu este o consecință logică a cheilor candidate

FNPJ

SP

S#	P#
S1	P1
S1	P2
S2	P1

PJ

P#	J#
P1	J2
P2	J1
P1	J1

JS

J#	S#
J2	S1
J1	S1
J1	S2

SP * PJ

S#	P#	J#
S1	P1	J2
S1	P1	J1
S1	P2	J1
S2	P1	J2
S2	P1	J1

S = Furnizor
P = Produs
J = Proiect

Dacă proiectul J
folosește produsul P,
atunci toți furnizorii
produsului P îl
furnizează
proiectului J

SPJ

S#	P#	J#
S1	P1	J2
S1	P2	J1
S2	P1	J1
S1	P1	J1

*

Dependența join: (S1, P1) în SP, (P1, J1) în PJ, (J1, S1) în JS implică (S1, P1, J1) în SPJ.