Baze de Date

Cap. 8. Modelare date relaționale

Textbook: Ramakrishnan, Gehrke, “Database Management Systems”, McGraw Hill, C15

2023 UPT Conf.Dr. Dan Pescaru

Modelul Relațional

1. O colecție de relații (tabele) și legăturile dintre acestea
2. Relație (tabelă)
   - Schema: nume relației, numele și tipul atributelor (coloanelor)
   - Instanța: tabela fizică (pe disc) având un număr fix de coloane și un număr variabil de rânduri
   - Numărul de coloane (atribute): gradul relației
   - Numărul de rânduri (înregistrări): cardinalitatea
   - Rândurile sunt DISTINCTE și NU sunt ordonate!

Constrângeri de integritate

1. O colecție de expresii logice care trebuie să fie satisfăcute de orice instanță a bazei de date
2. Sunt specificate la definirea schemei bazei de date
3. Sunt verificate la fiecare modificare efectuată în baza de date
4. O instanță de bază de date care satisface toate constrângerile de integritate se numește ”legală” sau ”consistentă”

Chei

1. Un set de atribute formează o cheie pentru o relație dacă:
   - Nu există pereche de înregistrări care să aibă valori egale pentru toate atributele din set (asigură unicitatea)
   - Orice subset al acestuia încalcă cerința de mai sus (trebuie să fie minimă)
2. O supercheie: un set de atribute care include cel puțin o cheie

Cheie primară (PK)

1. Cheie primară: o cheie aleasă de proiectantul bazei de date din setul de chei, care îndeplinește următoarele condiții:
   - Este scurtă (cuprinzând un număr minim de atribute - ideal doar unul).
     Obs: (caz neobișnuit) doar o cheie candidat formată din toate câmpurile tabelei
   - Este reprezentativă în contextul problemei (de exemplu, SID vs. CNP) și nu permite valori NULL
2. Alternativă: introducerea unei chei artificiale (de obicei numerică, folosind AutoIncrement)

Integritate referențială

1. Cheie externă (FK) – leagă două tabele (corespunde cheii primare a relației principale)
2. Folosită pentru a verifica integritatea referenţială
   - Când se adaugă o înregistrare în tabelul secundar: cheia corespunzătoare trebuie să existe
   - La ștergerea unei înregistrări din tabelul principal: pentru evitarea înregistrărilor ”orfane” din tabelul secundar
   - La modificarea valorii unei chei primare sau a unei chei externe în tabelele aferente

Asigurarea integrității referențiale

1. La adăugare (INSERT)
   - ”Avoiding operation that breaks constraints”
2. La ștergere (DELETE) se poate alege
   - ”Cascade delete related records”
   - ”Avoiding operation that breaks constraints”
   - ”Voiding the reference” (nu prea des)
3. La modificare (UPDATE) se poate alege
   1. ”Cascade update related records”
   2. ”Avoiding operation that breaks constraints”
4. Probleme: blocaje (referințe încrucișate) – soluție: utilizarea tranzacțiilor (verificare întârziată)

Integritate referențială. Example

Student

marca	nume	an	media
AC2153	Pop Angela	2	8.50
AC1078	Avram Ioan	1	9.35
AC2056	Ionescu Mihai	2	7.80

Contract

cid	marca	lab	ex
PLA2	AC2153	9	8
UC1	AC1078	10	9
SO2	AC2056	8	7

1. INSERT un nou contract: verifică marca in tabela Student
2. DELETE șterge un student: verifică marca in Contract
3. UPDATE modifică marca in Student: verifică marca in Contract / modifică marca in Contract – verify marca in Student

Modelul relațional: redundanța

1. Redundanța este la baza multor probleme asociate cu BD relaționale
   - Risipirea spațiului de depozitare
   - Anomalii la inserare / ștergere / actualizare
2. Exemplu:

Student

sid	nume	an	media	facultatea	adresa
AC2153	Pop Angela	2	8.50	Automatică și Calculatoare	Bd. V. Pârvan no,2, Electro
AC1078	Avram Ioan	1	9.35	Automatică și Calculatoare	Bd. V. Pârvan no,2, Electro
AC2056	Ionescu Mihai	2	7.80	Automatică și Calculatoare	Bd. V. Pârvan no,2, Electro

Abordarea redundanței

1. Constrângerile de integritate, în special dependențele funcționale, pot fi utilizate pentru a identifica scheme cu redundanță mare și pentru a sugera pași de rafinare
2. Tehnica principală de rafinare: decompoziția
   - Înlocuirea unei relații ABCD cu, de exemplu, AB și BCD, sau ACD și ABD
3. Descompunerea ar trebui folosită cu prudență:
   - Există motive pentru a descompune o relație?
   - Ce probleme (dacă există) sunt provocate de decompoziție?

Dependențe funcționale DF

1. O dependență funcțională $X \rightarrow Y$ există în relația R dacă, pentru fiecare instanță admisă r a lui R:

   $t_1 \in R, t_2 \in R, \pi_X(t_1)=\pi_X(t_2) \Rightarrow \pi_Y(t_1)=\pi_Y(t_2)$
2. Având două înregistrări în R, daca valorile X sunt de egale, atunci si valorile Y trebuie sa fie egale - unde X și Y sunt seturi de atribute
3. O DF este o declarație despre toate instanțele posibile și trebuie să fie identificat pe baza semanticii sistemului de implementat
4. Având în vedere o instanță r1 a lui R, putem verifica dacă aceasta încalcă unele DF *f*, dar nu și dacă un set de DF *f* este adevărat pentru R

Exemplu de DF

1. Se consideră o relație Angajati_Sezonieri (Cnp, Nume, nivel_Pregatire, pRoiect, Salar_orar, Ore_lucrate)
2. Notație: vom nota această schemă prin enumerarea atributelor sale – CNPRSO – corespunzător setului de atribute a relației
3. Uneori, ne vom referi la toate atributele unei relații folosind numele relației
4. Exemple de DF pentru Angajati_Sezonieri :
   - cnp este cheia primară: $C \rightarrow NPRSO$
   - nivelPregătire determină salarOrar: $P \rightarrow S$

Exemplu de probleme generate de DF

1. Probleme din cauza P → S
   - Anomalii de actualizare: dacă schimbam S doar în prima înregistrare a relației?
   - Anomalie de inserare: ce se întâmplă dacă dorim să inserăm un angajat cu un salariu orar diferit decât cel pentru pregătirea sa?
   - Anomalie de ștergere: dacă ștergem toți angajații cu nivelPregatire 3, pierdem informațiile despre salariu pentru nivelPregătire 3!

C	N	P	R	S	O
1850213988166	Marius	4	P3	12	40
1870925988352	Ion	2	P1	8	36
2891105988058	Rodica	3	P1	10	30
1911204988125	Petru	1	P2	5	32
2900217988855	Mona	2	P3	8	40

Determinarea DF prin inferență

- Dacă știm DF de bază, putem infera DF adiționale:
  - cnp $\rightarrow$ codf $\wedge$ codf $\rightarrow$ decan implică cnp $\rightarrow$ decan
- O DF *f* este implicată de un set de DF *F* dacă *f* este adevărată oricând *F* este adevărat
  - F^+ (*închiderea* lui F): setul tuturor DF *fi* implicate de F
- Axiomele lui Armstrong (X, Y, Z: seturi de atribute):
  - Reflexivitatea: dacă X $\subseteq$ Y, atunci Y $\rightarrow$ X
  - Augmentarea: dacă X $\rightarrow$ Y, atunci XZ $\rightarrow$ YZ pentru orice Z
  - Tranzitivitatea: dacă X $\rightarrow$ Y și Y $\rightarrow$ Z, atunci X $\rightarrow$ Z
- Acestea reguli de inferență sunt *necesare și suficiente* pentru a determina toate dependențele funcționale!

Exemplu de extragere DF

- O relație R are atribute (S, C, T, A, N) care denotă student, curs, timp, amfiteatru și nota. Din cerințe se pot deduce următoarele DF:
  - $SC \rightarrow N$
  - $ST \rightarrow A$
  - $C \rightarrow T$
  - $TA \rightarrow C$

Normalizare

- În ceea ce privește proiectarea schemei bazei de date, o întrebare bună de pus este când este necesară rafinare ei. Un răspuns simplu: în caz de redundanță
- Rezolvare: normalizare prin decompoziție
- Motiv: dacă o relație este într-o anumită formă normală, anumite tipuri de probleme sunt evitate/minimalizate
- Utilizarea DF în detectarea redundanței: (R:ABC)
  - Nu există FD-uri: nu există redundanță
  - Dacă avem A $\rightarrow$ B: având mai multe înregistrări cu valori A rezultă mai multe cu aceeași valoare B!

Normalizare – scurt istoric

- Edgar F. Codd a propus procesul de normalizare și ceea ce a ajuns să fie cunoscut sub numele de prima formă normală în lucrarea sa <A Relational Model of Data for Large Shared Data Banks>: „Există o procedură de eliminare foarte simplă pe care o vom numi normalizare. Prin decompoziție, domeniile non-simple sunt înlocuite cu domenii ale căror elemente sunt valori atomice (ne-decompozabile).”
- El a stabilit inițial trei forme normale: 1NF, 2NF și 3NF. Există acum și altele care sunt acceptate, dar 3NF este considerat în general a fi suficient (BD sunt considerate normalizate dacă sunt în 3NF)

Decompoziția unei scheme relaționale

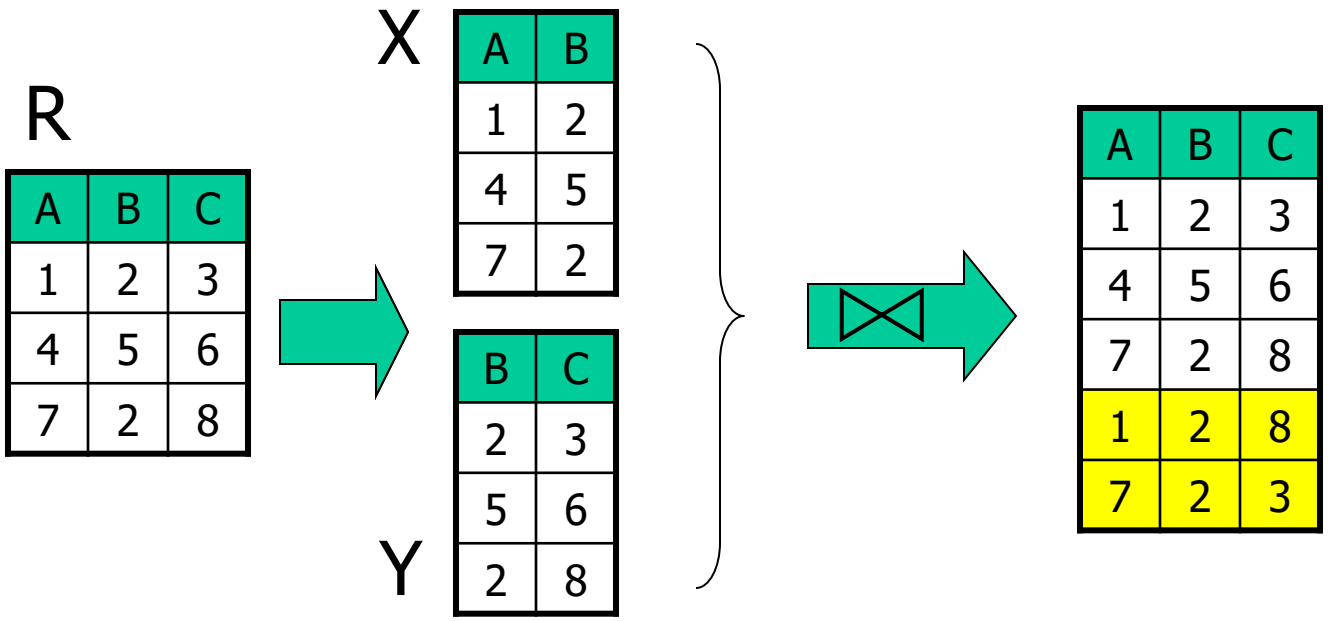
1. Să presupunem că relația R conține atributele A_1 ... A_n. O descompunere a lui R constă în înlocuirea lui R cu două sau mai multe relații astfel încât:
   - Fiecare nouă schemă conține un subset de atribute ale lui R (și nici un atribut care nu apare în R)
   - Fiecare atribut al lui R apare ca un atribut al cel puțin uneia dintre noile relații
2. Intuitiv, descompunerea lui R înseamnă că vom stoca instanțe ale schemei relației produse prin decompoziție în loc de instanțe ale lui R
3. De exemplu, putem descompune ABCDEFG în ABCDE și BFG

Decompoziții tip ”lossless join”

1. Decompoziția lui R în X și Y este fără pierderi la JOIN în raport cu o mulțime de DF *F* dacă, pentru fiecare instanță r care satisface *F*
   - $\Pi_X(r) \bowtie \Pi_Y(r) = r$
2. Definiția se extinde la decompoziția în 3 sau mai multe relații într-un mod simplu
3. Este esențial ca toate decompozițiile utilizate pentru reducerea redundanței să fie fără pierderi la JOIN!

Contra exemplu la "lossless join"

1. Decompoziția următoare a lui R în X și Y nu are proprietatea dorită!



Conservarea dependențelor

1. Decompozițiile trebuie să păstreze toate dependențele funcționale ale relației originale pentru a păstra toate constrângerile!
   - Condiția este: $(F1 \cup F2)^+ = F^+$
2. Exemplu

   R=(A,B,C), F={A→B, A→C, B→C}

   R1=(A,C), R2(B,C)

   $F1^+$={A→A, C→C, A→C, AC→CC}

   $F2^+$={B→B, C→C, B→C, BC→BC}

   - Obs: A→B nu este păstrată

Forma Normală 1 (1NF)

1. Formă normală 1 (1NF):
   1. Domeniul fiecărui atribut trebuie să conțină doar valori atomice; câmpurile compuse sau „relațiile în relație” sunt interzise
   2. Fiecare atribut conține doar o singură valoare în acel domeniu

Student

sid	nume	hobby	codf	adresa
AC6978	Popescu Mihai	șah, dans	AC	Timisoara, Parvan **no 1**, 0256112212
AC8967	Ionescu Georgeta	citit, muzică	AC	Timisoara, Parvan **no 1**, 0256112212

Forma Normală 1 - exemplu

1. Soluția: două diviziuni de atribute și o decompoziție

Student

sid	nume	hobby	codf	adresa
AC6978	Popescu Mihai	șah, dans	AC	Timisoara, Parvan **no 1**, 0256112212
AC8967	Ionescu Georgeta	citit, muzică	AC	Timisoara, Parvan **no 1**, 0256112212

Student

sid	prenume	nume	codf	oraș	stradă	telefon
AC6978	Mihai	Popescu	AC	Timisoara	Parvan no. 1	0256112212
AC8967	Georgeta	Ionescu	AC	Timisoara	Parvan no. 1	0256112212

Hobby

sid	hobby
AC6978	șah
AC6978	dans
AC8967	citit

Forma Normală 2 (2NF)

1. Formă normală 2 (2NF):
    1. Relația este deja în 1NF
    2. Orice atribut non-prim din R (nu face parte din cheia primară) trebuie să fie complet dependent funcțional de cheia primară din R SAU: Nu există atribute care să depindă doar de o parte a cheii primare

Factura

id	data	poz	cant	produs	pretUnitar	pretTotal
008978	01-03-2014	1	2	pâine	5	10
008978	01-03-2014	2	5	mere	8	40
099488	05-03-2014	1	1	brânză	26	26

Forma Normală 2 - Exemplu

1. Soluția: decompoziția

Factura

id	data
008978	01-03-2014
008978	01-03-2014
099488	05-03-2014

PozitiiFactura

id	poz	cant	produs	pretUnitar	pretTotal
008978	1	2	pâine	5	10
008978	2	5	mere	8	40
099488	1	1	brânză	26	26

Forma Normală 3 (3NF)

1. Formă normală 3 (3NF):
   1. Relația este deja în 2NF
   2. Fiecare atribut non-prim depinde în mod netranzitiv de fiecare cheie candidată din tabel. Cu alte cuvinte, nu este permisă nici o dependență tranzitivă
2. BCNF: Fiecare dependență funcțională netrivială din tabel este o dependență de o supercheie

PozitiiFactura

id	poz	produs	cant	pretUnitar	pretTotal
008978	1	pâine	2	5	10
008978	2	mere	5	8	40
099488	1	brânză	1	26	26

Forma Normală 3 - Exemplu

1. Soluția: decompoziția

PozitiiFactura

id	poz	produs	cant	pretUnitar	pretTotal
008978	1	pâine	2	5	10
008978	2	mere	5	8	40
099488	1	brânză	1	26	26

CatalogProduse

produs	pretUnitar
pâine	5
mere	8
brânză	26

Forma Normală 4 (4NF)

1. Formă normală 4 (4NF):
   1. Relația este deja în 3NF
   2. Nu există dependențe funcționale multivaloare
2. Dependențe multivaloare – la o schema ABC, dependența multivaloare există dacă fiecărui A îi corespund mai mulți B și mai mulți C, dar B și C sunt independente unul de celălalt

Componente

Departament	Proiect	Componenta
D1	Pr1 Pr2	C1 C2 C3
D2	Pr2 Pr3 Pr5	C2 C4

Componente (3NF)

Departament	Proiect	Componenta
D1	Pr1	C1
D1	Pr1	C2
D1	Pr1	C3
D1	Pr2	C1
D2	Pr2	C2
...	...	...
D2	Pr5	C4

Forma Normală 4 - Exemplu

1. Problemă: să presupunem că departamentele mențin un stoc de componente și dezvoltă proiecte care pot folosi unele dintre componentele disponibile sau toate. Rezolvare: decompoziția

Componente (3NF)

Departament	Proiect	Componenta
D1	Pr1	C1
D1	Pr1	C2
D1	Pr1	C3
D1	Pr2	C1
D2	Pr2	C2
...	...	...
D2	Pr5	C4

DP (4NF)

Departament	Proiect
D1	Pr1
D1	Pr2
D2	Pr2
...	...

DC (4NF)

Departament	Componenta
D1	C1
D1	C2
D1	C3
D2	C2
...	...