Baze de Date 1

Cuprins

Concepte de Baze de Date

Baza de Date – modalitate de stocare a datelor pe un suport extern de memorie, într-o anumita structura, în vederea interogărilor concurente și prelucrărilor ulterioare pentru extragerea informațiilor.

Sistem de Gestiune a Bazei de Date – ansamblu de programe software care permit utilizatorilor definirea obiectelor, administrarea bazei de date și accesul concurent la baza de date.

Dicționarul bazei de date – informații despre obiectele din baza de date și parametrii de sistem: structura tabelelor, indecși, constrângeri de integritate definite, spațiul fizic și organizarea logica a fișierelor, utilizatorii creați și drepturile de acces.

Structura bazei de date – descrierea entităților împreuna cu relațiile logice dintre ele

Entitate – un obiect al bazei de date cu o reprezentare unica (*tabelele*)

Atribut – o proprietate ce descrie o anumita caracteristica a unei entități (*coloanele*)

Relație logica – asocierea dintre mai multe entități

Modelul ierarhic	Modelul rețea	Modelul relațional	Modelul obiect
Arbore	Graf	Tabele	Obiecte

Data Definition Language – definirea structurii obiectelor din baza de date

Data Manipulation Language – manipularea datelor (operații pe tabele)

Constrângeri de integritate – reguli pe care datele trebuie sa le respecte la nivel de tabel sau relație

- Valori nenule
- Cheie unica
- Cheie primara – unica și nenula
- Cheie externa
- Verificare

Nivelul intern – organizarea interna a datelor (*schema fizica*); **Nivelul conceptual** – modelul de date (*schema logica*. Exemplu: modelul relațional); **Nivelul extern** – utilizatorii care accesează baza de date (*scheme externe*).

Independenta logica – posibilitatea de schimbare a schemei conceptuale a bazei de date fără modificarea schemelor externe.

Independenta fizica – posibilitatea se schimbare a scheme fizice a bazei de date fără modificarea schemei conceptuale și implicit a schemelor externe.

Introducere în Algebra Relațională

Fie relațiile R, S și T:

- Reuniunea: $T = R \cup S$
- Intersecția: $T = R \cap S$
- Diferența: $T = R - S; T = S - R; in\ general\ R - S \neq S - R$
- Proiecția: $T = \pi_{lista\_atribute}(R)$
- Selecția: $T = \sigma_F(R);\ F = conditie$
- Produsul Cartezian: $T = R \times S$
- Join: $T = R \bowtie_F S = \sigma_F(R \times S); F = conditie\ de\ join$
    - Join Natural: $T = R \bowtie S$; join după condiția coloanele cu același nume și eliminarea prin proiecție a atributelor duplicate (coloanele după care s-a făcut joinul)
    - Join Extern (Outer): apar și liniile care nu îndeplinesc condiția de join (din lipsa de date)
        - Join Extern Stânga: $T = R \lhd 0 \rhd_{L(F)} S$
        - Join Extern Dreapta: $T = R \lhd 0 \rhd_{R(F)} S$
        - Join Extern Complet: $T = R \lhd 0 \rhd_F S$
    - Semi Join: $T = R \rhd\!< S = \pi_R(R \bowtie S) = R \bowtie \pi_{R \cap S}(S)$; în general nu e simetric
- Redenumirea: $T = \rho_{R(A1,\dots,An)}$ permite redenumirea relațiilor/multiseturilor cat și a atributelor
- Constructorul: $Nume\_vechi \rightarrow Nume\_nou$ permite redenumirea într-un rezultat a unui atribut
- Eliminare duplicate: $\delta(R)$ transforma un multiset într-o relație
- Grupare: $\gamma_{atribute\&functii}(R); atribute - criterii\ de\ grupare, functii - functii\ de\ grup$
- Sortare: $\tau_{lista\_atribute}(R)$

- Proiecția extinsa: o proiecție în care pot sa apară și expresii nu doar atribute

Multiset – mulțime în care se admit duplicate (diferența, produsul cartezian, selecția, joinul, joinul natural, joinul extern nu produc duplicate decât daca operanzii conțin linii duplicat).

Modelul Relațional

Obiectele bazei de date se numesc entități și sunt relaționate între ele prin anumite constrângeri care formează așa numita diagrama de relații.

Domeniul – mulțime de valori identificata printr-un nume

Relația – o submulțime a unui produs cartezian de domenii asociat unui nume

Tupluri – elementele unei relații (*liniile*)

Schema unei relații – structura relației compusa din nume, atribute, domeniul atributelor și constrângerile de integritate asociate

Cheia unei relații – mulțime minimala de atribute ce identifica în mod unic un tuplu al relației.

TODO:

Expresii sigure și Calcul relațional

Modelul Entitate-Asociere

Modelul Entitate-Asociere – reprezentarea informațiilor: **entități**, **atribute** ale entităților și **asocieri** între entități.

- Entități – modelează clase de obiecte
  - Entități independente
  - Entități dependente
- Atribute – proprietăți atomice distincte ale entităților
  - Atribute de Identificare
  - Atribute de Descriere
- Asocieri – interdependentele dintre clasele de obiecte reprezentate prin entități
  - Gradul Asocierii: numărul de entități care participa la asociere
    - unare, binare, ternare, n-are.
  - Conectivitatea asocierii: specifica fiecărei ramuri
    - unu sau multi
  - Obligativitatea asocierii: specifica fiecărei ramuri
    - Obligatorie sau opțională
  - Atribute ale asocierii: atribute descriptive
  - Rolul unei ramuri
- Ierarhii

- Ierarhii de Generalizare – o entitate E este generalizarea entităților E_1, ..., E_n daca orice instanța a lui E este de asemenea instanța în una și numai una dintre entitățile E_1, ..., E_n (Angajați – Femei, Bărbați; criteriul de generalizare este genul).
- Ierarhii de Incluziune – o entitate E_k este o submulțime a entității E daca fiecare instanța a lui E_k este de asemenea o instanța a lui E (Angajați – Ingineri, Economiști, Colaboratori).

Clasificarea în Entități și Atribute

1. Entitățile au informații descriptive, pe când atributele nu posedă astfel de informații.
2. Atributele multivalorice vor fi reclasificate ca entități.
3. Atributele unei entități care au o asociere multi-unu cu o alta entitate vor fi reclasificare ca entități.
4. Atributele vor fi atașate la entitățile pe care le descriu în mod nemijlocit.
5. Folosirea identificatorilor compuși va fi evitata, pe cat posibil.
    a. Daca identificatorul unei entități este compus din mai multe atribute care sunt toate identificatori în alte entități, acea entitate se elimină. Informația conținută de aceasta va fi modelata sub forma unei asocieri între acele entități.
    b. Daca identificatorul unei entități este compus din mai multe atribute care nu sunt toate identificatori în alte entități:
        i. Entitatea respectiva se elimină și este înlocuita prin alte entități și asocieri astfel încât per ansamblu informația modelata în varianta inițiala sa fie păstrata.
        ii. Entitatea respectiva rămâne în forma inițiala, cu dezavantaje însă în privința vitezei operațiilor.

Identificarea ierarhiilor de generalizare și incluziune

1. În cazul în care despre anumite subclase ale unei clase de obiecte exista informații specifice, clasa și subclasele (care la pasul anterior au fost catalogate ca entități) sunt interconectate într-o ierarhie de incluziune sau generalizare.
2. La acest pas se face și o reatașare a atributelor pentru evitarea redundantei:
    a. La entitatea tata vor fi atașate atributele care formează identificatorul și descriptorii care modelează informații specifice întregii clase.
    b. La entitățile fiu vor fi atașate atributele de identificare (aceleași ca ale tatălui) plus atributele care modelează informații specifice doar acelei subclase de obiecte.
3. Identificatorul tata din cadrul ierarhiei se regăsește în identificatorul tuturor fiilor ierarhiei.
4. Descriptorii care apar și la tata și la fii, se elimină de la fii.
5. Descriptorii care apar la toți fiii unei ierarhii de generalizare, dar nu apar la tata, se muta la tata.

Transformarea diagramelor Entitate-Asociere în Modelul Relațional

1. **Relații provenite din entități**: ele conțin aceleași informații ca și entitățile din care au rezultat
2. **Relații provenite din entități și care conțin chei străine**: ele conțin pe lângă informațiile provenite din entitățile din care au rezultat și atribute care în alte entități sunt identificatori. Este cazul acelor entități care au asocieri multi-unu și parțial din cele care au asocieri unu-unu cu alte entități.
3. **Relații provenite din asocieri**: este cazul celor care apar din transformarea asocierilor binare multi-multi și a asocierilor de grad mai mare ca doi. Ele conțin ca atribute reuniunea identificatorilor entităților asociate și atributele proprii ale asocierilor.

Proiectarea Bazelor de Date

Relație – **schema relației** (descrierea structurii acesteia) și **instanța a relației** (datele propriu-zise).

Dependenta datelor – restricție asupra relațiilor și care pot constitui valoarea curenta a unei scheme de relație R. Datele trebuie văzute ca o legătura intre doua atribute.

Fie R o schema de relație și X, Y inclus în R doua mulțimi de atribute ale acesteia. Spunem ca X determină funcțional pe Y ($X \to Y$) daca și numai daca oricare ar fi doua tupluri t_1 și t_2 din orice instanță a lui R atunci $t_1[X]=t_2[X] \Rightarrow t_1[Y]=t_2[Y]$. Daca doua tupluri au aceleași valori pe atributele X atunci ele au aceleași valori și pe atributele Y.

Axiomele lui Armstrong

1. Reflexivitate: $X \subseteq R; daca\ Y \subseteq X\ atunci\ X \to Y$
2. Augmentare: $X, Y, Z \subseteq R; daca\ X \to Y\ atunci\ XZ \to YZ$
3. Tranzitivitate: $X, Y, Z \subseteq R; daca\ X \to Y\ si\ Y \to Z\ atunci\ X \to Z$
4. Descompunere: $X, Y, Z \subseteq R; daca\ X \to Y\ si\ Z \subseteq Y\ atunci\ X \to Z$
5. Reuniune: $X, Y, Z \subseteq R; daca\ X \to Y\ si\ X \to Z\ atunci\ X \to YZ$
6. Pseudotranzitivitatea: $X, Y, Z, W \subseteq R; daca\ X \to Y\ si\ YZ \to W\ atunci\ XZ \to W$

Închiderea mulțimii de dependente

Fie F un set de dependente funcționale. Închiderea mulțimii de dependente F: $F^+ = \{X \to Y | F \Rightarrow X \to Y\}$.

Acoperirea unei mulțimi de dependente funcționale

Fie F, G mulțimi de dependente funcționale pentru R. F acoperă pe G daca și numai daca $G \subseteq F^+$.

Echivalenta a doua mulțimi de dependente

Doua mulțimi de dependente F și G sunt echivalente daca și numai daca F acoperă pe G și G acoperă pe F.

Forma canonica a unei mulțimi de dependente funcționale

Orice dependenta are în partea dreapta un singur atribut.

Mulțimea de dependente este minimală, niciuna dintre dependențe neputând sa fie dedusa din celelalte (nu exista dependente redundante).

Implicații logice ale dependentelor

F implica logic $X \to Y$, daca orice relație r pentru R, care satisface dependentele din F, satisface $X \to Y$.

Închiderea mulțimii de dependente F, notata cu F^+, se mai definește ca mulțimea dependentelor funcționale implicate logic de către F.
Daca $F=F^+$, F este o familie completa de dependente.

Cheia

Fie R o schema de relație, F mulțimea de dependente funcționale asociata și $X \subseteq R$. Atunci X este cheie pentru R daca și numai daca:

- $F => X \to R$ (deci $X \to R$ se poate deduce din F)
- X este minimala
- $nicio\ submultime\ stricta\ a\ sa\ nu\ determina\ functional\ pe\ R$

Daca X nu este minimala, atunci este supercheie. Orice cheie este în același timp și supercheie.

Proiecția mulțimii de dependente

Este mulțimea dependentelor din F^+ care au și partea stânga și pe cea dreapta incluse în S.

$$\pi_S(F) = \{X \to Y \in F^+ | X, Y \subseteq S\}$$

Închiderea unei mulțimi de atribute

Fie R o schema de relație, F mulțimea de dependente asociata și $X \subseteq R$. Se poate defini X^+ ca fiind închiderea mulțimii de atribute X în raport cu F astfel: $X^+ = \{A | X \to A \in F^+\}$. Deci X^+ conține toate atributele care apar în partea dreapta a dependentelor din F sau care se pot deduce din F.

1. Se pornește cu $X^{(0)} = X$
2. $X^{(i)} = X^{(i-1)} \cup \{A | \exists\ Y \to A \in F\ cu\ Y \subseteq X^{(i-1)}\}$, i >= 1
3. Pana când $X^{(i)} = X^{(i-1)}$ sau $X^{(i)} = R$

Fie R o schema de relație, F mulțimea de dependente asociata și $X, Y \subseteq R$, atunci $X \to Y$ se poate deduce din F daca și numai daca $Y \subseteq X^+$.

Cheia

Fie R o schema de relație, F mulțimea de dependente funcționale asociata și $X \subseteq R$. Atunci X este cheie pentru R daca și numai daca:

- $X^+ = R$
- X este minimală
- $nicio\ submultime\ stricta\ a\ sa\ nu\ determina\ functional\ pe\ R$

Daca X nu este minimală, atunci este supercheie. Orice cheie este în același timp și supercheie.

Euristica de găsire a cheilor unei relații

Fie R o schema de relație și F mulțimea de dependente funcționale asociata (in forma canonica).

1. Se pornește de la mulțimea de atribute $X \subseteq R$ care nu apar în partea dreapta a niciunei dependente.
2. Se calculează X^+. Daca $X^+ = R$ atunci X este cheia unica minimala a relației R. Altfel:
3. Se adaugă la X cate un atribut din $R - X^+$ obținând-se o mulțime de chei candidat.
4. Se calculează X^+ pentru fiecare dintre candidate. Daca se obțin toate atributele lui R atunci acel X este o cheie a lui R.
5. Se repeta pașii 3 și 4 pornind de la acele mulțimi candidat X care nu sunt găsite ca și chei la pasul anterior. Nu se ia în considerare o mulțime care conține o cheie găsita anterior.
6. Procesul se oprește când nu se mai pot face augmentări.

Acoperiri de mulțimi dependente

Fiecare mulțime de dependente funcționale F este acoperita de o mulțime de dependente G, în care nicio dependenta nu are în partea dreapta mai mult de un atribut.

O mulțime de dependente F este minimală daca:

1. Partea dreapta a fiecărei dependente din F conține un singur atribut (nu este redundant)
2. Pentru nicio dependenta $X \to A$ din F, mulțimea F – $\{X \to A\}$ nu este echivalenta cu F (F nu este redundanta)
3. Pentru nicio dependenta $X \to A$ din F și pentru nicio submulțime $Z \subseteq X$, multimea F – $\{X \to A\}$ U $\{Z \to A\}$ nu este echivalenta cu F (niciun atribut din partea stânga nu este redundant)

Fiecare mulțime de dependente F este echivalenta cu o mulțime F' care este minimală.

Normalizarea Schemelor de Relație

Forma Normala 1

O relație R se găsește în Forma Normala 1 daca și numai daca:

- Pe toate atributele sale exista doar valori atomice ale datelor (nu exista atribute cu valori multiple);
- Nu exista atribute sau grupuri de atribute care se repeta.

Forma Normala 2

Fie A un atribut care nu face parte din cheie și X o mulțime de atribute din R care formează o cheie a relației.

- $Y \to A$ dependenta parțiala daca Y este strict inclusa într-o cheie a relației R;
- $Y \to A$ dependenta tranzitiva daca Y nu este inclusa în nicio cheie a relației R.

O relație R se găsește în Forma Normala 2 daca și numai daca:

- Se găsește în Forma Normala 1;
- Orice atribut care nu face parte din cheie va fi identificat de întreaga cheie, nu doar de unele atribute care fac parte din cheie (nu exista dependente parțiale).

Forma Normala 3

Un atribut $A \in R$ se numește **atribut prim** daca el aparține unei chei a lui R.

O relație R se găsește în Forma Normala 2 daca și numai daca oricare ar fi o dependenta netriviala $X \to A \; din \; F$:

- X este supercheie pentru R;
- Sau A este atribut prim.
- Nu se accepta dependente tranzitive.

Forma Normala Boyce-Codd

O relație R se găsește în Forma Normala Boyce-Codd daca și numai daca oricare ar fi o dependenta netriviala $X \to A \; din \; F$:

- X este supercheie pentru R.

- Fiecare dependenta din F are în partea stânga o supercheie.

Anomalii în baze de date

Anomalii de *proiectare* (redundante, tipuri greșite de atribute, constrângeri incorecte).

Anomalii de *funcționare* (inserare, modificare, ștergere).

Independenta datelor

Independenta *logica* a datelor se refera la imunitatea schemelor externe fata de oficierile efectuate în schema conceptuala (adăugarea de entități sau atribute noi; modificarea structurii entităților; ștergerea anumitor entități sau atribute; adăugarea, modificarea sau ștergerea de obiecte în baza de date – view-uri, proceduri, funcții; activarea sau dezactivarea unor constrângeri).

Independenta *fizica* de date se refera la imunitatea schemei conceptuale fata de modificările efectuate în schema interna (reorganizarea fișierelor de date; introducerea de noi dispozitive de stocare; mutarea bazei de date de pe un server pe altul; modificări în configurarea rețelei, în cazul bazelor de date distribuite; replicarea datelor din/pe mai multe servere; utilizarea serverelor redundante pentru o securitate sporita a datelor).

Descompunerea Schemelor de Relație

Descompunerea unei scheme de relație – procesul prin care se divide o relație în mai multe relații.

Fie R o schemă re relație, R = A_1 ... A_m. Se spune că ρ = (R_1, ..., R_n) este o descompunere a lui R dacă și numai dacă $R = R_1 \cup \ldots \cup R_n$. Schemele R_i conțin atributele din R, fiecare atribut A_j al schemei inițiale trebuie să se regăsească în cel puțin una dintre ele (nu este necesar ca schemele să fie disjuncte). Fie r o instanță a relației de schema R, atunci instanțele pentru relațiile din descompunerea ρ sunt $r_i = \pi_{R_i}(r)$.

Descompunerea cu join fără pierdere de date

Fie R o schemă de relație, F mulțimea de dependențe funcționale asociată și o descompunere ρ = (R_1, ..., R_n) a lui R. Descompunerea ρ este o descompunere cu join fără pierderi în raport cu F dacă și numai dacă pentru orice instanță r a lui R, care satisface dependențele F: $r = r_1 \bowtie \cdots \bowtie r_n$ unde $r_i = \pi_{R_i}(r)$.

Algoritm de testare a proprietății de join fără pierderi de date pentru o descompunere

1. Se construiește o tabelă având n linii și m coloane. Liniile sunt etichetate cu elementele descompunerii ρ iar coloanele cu atributele relației R. Elementul (i, j) al tabelei va fi egal cu a_j dacă $A_j \in R_i$ sau b_{ij} în caz contrar.
2. Se parcurg dependențele $X \to Y$ din F. Dacă două sau mai multe linii din tabelă au aceleași simboluri pe coloanele X aceste linii se egalează și pe coloanele din Y astfel:
    a. Dacă pe o coloană din Y apare un a_j atunci toate elementele de pe acea coloană din liniile respective devin a_j.
    b. Dacă pe o coloană din Y nu apare niciun a_j atunci se alege unul dintre elementele de tip b_{ij} și toate elementele de pe acea coloană din liniile respective devin egale cu acel b_{ij}.
3. Procesul se oprește

a. Fie când s-a obținut o linie în tabela care conține doar a-uri, caz în care descompunerea ρ are proprietatea de join fără pierderi de date.
b. Fie când la o parcurgere a dependențelor nu mai apar schimbări în tabelă și nu s-a obținut o linie doar cu a-uri, caz în care descompunerea ρ nu are proprietatea de join fără pierderi de date.

În cazul în care descompunerea are numai două elemente se poate testa proprietatea de join fără pierderi de date și în felul următor:
Fie R o schemă de relație, F mulțimea de dependențe funcționale asociată și ρ = (R_1, R_2) o descompunere a sa. Atunci ρ are proprietatea de join fără pierderi de date dacă una dintre dependențele următoare se poate deduce din F:

- $(R_1 \cap R_2) \rightarrow (R_1 - R_2)$
- $(R_1 \cap R_2) \rightarrow (R_2 - R_1)$

Descompunerea care păstrează dependențele funcționale

Fie R o schemă de relație, F mulțimea de dependențe funcționale asociată și o descompunere ρ = (R_1, ..., R_n) a lui R și $F_i = \pi_{R_i}(F)$ proiecția mulțimii de dependențe funcționale (mulțimile de dependențe funcționale ale elementelor descompunerii). Descompunerea ρ păstrează dependențele funcționale din F dacă și numai dacă orice dependență din F poate fi dedusă din reuniunea dependențelor din F_i.

Algoritm de testare dacă o dependență este sau nu păstrată după descompunere

Pentru fiecare dependență $X \rightarrow Y$ din F se procedează astfel:

1. Se pornește cu o mulțime de atribute Z = X.
2. Se parcurg repetat elementele descompunerii ρ. Pentru fiecare R_i se calculează o nouă valoare a lui Z astfel: $Z = Z \cup ((Z \cap R_i)^+ \cap R_i)$.
3. Procesul se oprește în momentul în care Z rămâne neschimbat la o parcurgere a elementelor R_i. Dacă $Y \subseteq Z$ atunci dependența $X \rightarrow Y$ este păstrată, altfel nu este păstrată.
4. Dacă toate dependențele din F sunt păstrate înseamnă că ρ păstrează dependențele din F, altfel nu.

<u>Algoritmi de descompunere a schemelor de relație</u>

Algoritm de descompunere în FN3 cu păstrarea dependențelor funcționale

Fie R o schemă de relație și F mulțimea de dependențe funcționale asociată, cu $F = \{X_1 \rightarrow Y_1, \dots, X_n \rightarrow Y_n\}$. Atunci descompunerea ρ = (X_1Y_1, ..., X_nY_n) este o descompunere în FN3 cu păstrarea dependențelor.

1. Toate dependențele sunt păstrate: dependența $X_i \rightarrow Y_i$ este în proiecția lui F pe X_iY_i.
2. Pentru a minimiza numărul de elemente din descompunere se aplică regula reuniunii: dacă sunt mai multe dependențe care au aceeași parte stânga, se reunesc într-una singură.
3. Dacă în descompunere există două elemente X_iY_i și X_jY_j astfel încât $X_iY_i \subseteq X_jY_j$ atunci X_iY_i se elimină.

Algoritm de descompunere în FN3 cu păstrarea dependențelor funcționale și join fără pierderi de date

Dacă la descompunerea cu păstrarea dependențelor funcționale se adaugă și o cheie a relației ca element al descompunerii se obține o descompunere care are atât proprietatea de join fără pierderi de date cât și cea a păstrării dependențelor funcționale.

Fie R o schemă de relație și F mulțimea de dependențe funcționale asociată, cu $F = \{X_1 \to Y_1, \dots, X_n \to Y_n\}$. Fie X o cheie pentru R care nu este inclusă în X_iY_i. Atunci descompunerea ρ = (X, X_1Y_1, ..., X_nY_n) este o descompunere în FN3 cu păstrarea dependențelor funcționale și join fără pierderi de date.
Dacă vreunul dintre elementele de forma X_iY_i conține deja o cheie a lui R atunci nu este necesară adăugarea unui element suplimentar în descompunere, adică dacă o cheie este deja inclusă într-o descompunere atunci nu trebuie adăugată ca element suplimentar.

Algoritm de descompunere în FNBC cu ~~păstrarea dependențelor funcționale și~~ join fără pierderi de date

Fie R o schemă de relație și F mulțimea de dependențe funcționale asociată, cu $F = \{X_1 \to A_1, \dots, X_n \to A_n\}$ în forma canonică.

1. Inițial ρ = (R).
2. La fiecare pas se alege o schemă T care conține o dependență de forma $X \to A$ care violează condițiile de FNBC. Schema respectivă este înlocuită în ρ prin T_1 = XA și T_2 = T – {A}.
3. Procesul se oprește când în ρ nu mai există elemente care nu sunt în FNBC.

Dependența moștenită de T_2 este din F^+. Ea se deduce astfel: din $D \to A$ prin augmentare cu B se obține $DB \to AB$ și împreună cu dependența $AB \to C$, prin tranzitivitate se obține $DB \to C$. Analog din $AB \to D$ se deduce $DB \to D$ dar aceasta este o dependență trivială (partea dreaptă e inclusă în cea stângă). În multe cazuri este nevoie de mai multe iterații, relațiile de tip T_2 (egale în algoritm cu T – A) nefiind uneori în FNBC. Ele se descompun din nou în același fel.

Dependențe Multivalorice. Formele normale FN4 și FN5

Dependențe multivalorice

Fie o relație R și două mulțimi de atribute X și Y incluse în R. Se spune că X multidetermină pe Y sau că există dependența multivalorică $X \to\to Y$, dacă și numai dacă, ori de câte ori există două tupluri t_1 și t_2 cu $t_1[X]=t_2[X]$ atunci există un tuplu t_3 în relația R pentru care:

- $t_3[X]=t_1[X]=t_2[X]$
- $t_3[Y]=t_1[Y]$ și $t_3[R – X – Y]=t_2[R – X – Y]$

Cu alte cuvine, construim un tuplu t_3 compus din $t_1[X]+ t_1[Y]+ t_2[R – X – Y]$ care trebuie să se regăsească de asemenea printre tuplurile din R. O consecință interesantă a aceste definiții este că, dacă se inversează tuplurile t1 și t2, rezultă că există și un tuplu t4 cu proprietățile de mai sus.
Tot din această definiție rezultă ca dacă există în R dependența multivalorică $X \to\to Y$ atunci există și dependența multivalorică $X \to\to R - X - Y$.
În definiția dependențelor multivalorice nu se cere ca t_3 să fie diferit de t_1 și t_2, astfel orice dependență funcțională este în același timp și o dependență multivalorică.

Axiome și reguli pentru dependențele multivalorice

1. Complementare: $X, Y \subseteq R; daca\ X \to\to Y\ atunci\ X \to\to R - X - Y$
2. Augmentare: $X, Y, Z, W \subseteq R; daca\ X \to\to Y\ si\ Z \subseteq W\ atunci\ XW \to\to YZ$
3. Tranzitivitate: $X, Y, Z \subseteq R; daca\ X \to\to Y\ si\ Y \to\to Z\ atunci\ X \to\to Z - Y$
4. $X, Y \subseteq R; daca\ X \to Y\ atunci\ X \to\to Y$

5. $X, Y, Z, W \subseteq R \; si \; W \cap Y = \emptyset; daca \; X \twoheadrightarrow Y, Z \subseteq Y si \; W \rightarrow Z \; atunci \; X \rightarrow Z$
6. Reuniune: $X, Y, Z \subseteq R; daca \; X \twoheadrightarrow Y \; si \; X \twoheadrightarrow Z \; atunci \; X \twoheadrightarrow YZ$
7. Pseudotranzitivitate: $X, Y, Z, W \subseteq R; daca \; X \twoheadrightarrow Y \; si \; WY \twoheadrightarrow Z \; atunci \; WX \twoheadrightarrow Z - WY$
8. Pseudotranzitivitate mixtă: $X, Y, Z \subseteq R; daca \; X \twoheadrightarrow Y \; si \; XY \twoheadrightarrow Z \; atunci \; X \twoheadrightarrow Z - Y$
9. Diferența: $X, Y, Z \subseteq R; daca \; X \twoheadrightarrow Y \; si \; X \twoheadrightarrow Z \; atunci \; X \twoheadrightarrow Y - Z \; si \; X \twoheadrightarrow Z - Y$
10. Intersecția: $X, Y, Z \subseteq R; daca \; X \twoheadrightarrow Y \; si \; X \twoheadrightarrow Z \; atunci \; X \twoheadrightarrow Y \cap Z$
11. Eliminare atribute comune: $X, Y \subseteq R; daca \; X \twoheadrightarrow Y \; atunci \; X \twoheadrightarrow Y - X$
12. Acoperire completă de atribute: $X, Y \subseteq R; daca \; X \cup Y = R \; atunci \; X \twoheadrightarrow Y \; si \; Y \twoheadrightarrow X$
13. Reflexivitate: $X, Y \subseteq R; daca \; Y \subseteq X \; atunci \; X \twoheadrightarrow Y$

Orice dependență funcțională este în același timp o dependență multivalorică, reciproca nu este neapărat adevărată.

Forma Normala 4

O relație R se găsește în Forma Normala 4 daca și numai daca oricare ar fi o dependenta multivalorică netriviala ($X, Y \subseteq R; daca \; Y \subseteq X \; sau \; daca \; X \cup Y = R \; atunci \; X \twoheadrightarrow Y \; triviala$) $X \twoheadrightarrow Y \; din \; F$:

- X este supercheie pentru R.
- Fiecare dependenta din F are în partea stânga o supercheie.

Atunci când o relație poate fi reconstruită fără pierderi din unele proiecții ale sale se spune că există o *dependență joncțională*. Dacă joncțiunile includ și relația, atunci este o *dependență joncțională trivială*.

Algoritm de descompunere în FN4 cu ~~păstrarea dependențelor funcționale și~~ join fără pierderi de date

Fie R o schemă de relație și F mulțimea de dependențe funcționale asociată, cu $F = \{X_1 \rightarrow A_1, \dots, X_n \rightarrow A_n\}$ în forma canonică.

1. Inițial ρ = (R).
2. La fiecare pas se alege o schemă T care conține o dependență de forma $X \twoheadrightarrow Y$ care violează condițiile de FN4. Schema respectivă este înlocuită în ρ prin T_1 = XY și T_2 = X(T – Y).
3. Procesul se oprește când în ρ nu mai există elemente care nu sunt în FN4.

Fiecare subschemă T obținută din descompunere moștenește de la relația originală T proiecția mulțimii de dependențe – funcționale și multivalorice – a lui T pe T_i.

Forma Normala 5

O relație R se găsește în Forma Normala 5 daca și numai daca:

- Dependența joncțională care reconstruiește schema originală este o dependență joncțională trivială; sau
- Fiecare relație în dependența joncțională constituie cu toate atributele sale o supercheie a relației originale.

Limbajul SQL

SQL – Structured Query Language: cererile se execută secvențial, linie cu linie, deci se prelucrează o singură înregistrare la un moment dat.

Optimizarea Cererilor de interogare

Echivalența expresiilor în algebra relațională

1. *Comutativitatea produsului cartezian și a joinului*
   a. $E_1 \times E_2 = E_2 \times E_1$
   b. $E_1 \bowtie E_2 = E_2 \bowtie E_1$
   c. $E_1 \bowtie_F E_2 = E_2 \bowtie_F E_1$
2. *Asociativitatea produsului cartezian și a joinului*
   a. $E_1 \times (E_2 \times E_3) = (E_1 \times E_2) \times E_3$
   b. $E_1 \bowtie (E_2 \bowtie E_3) = (E_1 \bowtie E_2) \bowtie E_3$
   c. $E_1 \bowtie_{F_1} (E_2 \bowtie_{F_2} E_3) = (E_1 \bowtie_{F_1} E_2) \bowtie_{F_2} E_3$
3. *Cascada de proiecții*
   a. $\pi_{A_1 \dots A_n}\left(\pi_{B_1 \dots B_m}(E)\right) = \pi_{A_1 \dots A_n}(A), \{A_i\} \subseteq \{B_j\}$
4. *Cascada de selecții*
   a. $\sigma_{F_1}\left(\sigma_{F_2}(E)\right) = \sigma_{F_1 \wedge F_2}(E)$
5. *Comutativitatea selecțiilor cu proiecțiile*
   a. $\pi_{A_1 \dots A_n}(\sigma_F(E)) = \sigma_F\left(\pi_{A_1 \dots A_n}(E)\right), F\ contine\ doar\ atribute\ A_i$
   b. $\pi_{A_1 \dots A_n}(\sigma_F(E)) = \pi_{A_1 \dots A_n}\left(\sigma_F\left(\pi_{A_1 \dots A_n, B_1 \dots B_m}(E)\right)\right), F\ conține\ si\ atribute\ B_j$
6. *Comutativitatea selecției cu produsul cartezian*
   a. $\sigma_F(E_1 \times E_2) = \sigma_F(E_1) \times E_2, F\ contine\ doar\ atribute\ din\ E_1$
   b. $\sigma_F(E_1 \times E_2) = \sigma_{F_1}(E_1) \times \sigma_{F_2}(E_2), F = F_1 \wedge F_2, F_i\ contine\ doar\ atribute\ din\ E_i$
   c. $\sigma_F(E_1 \times E_2) = \sigma_{F_2}(\sigma_{F_1}(E_1) \times E_2), F = F_1 \wedge F_2, F_1\ contine\ doar\ atribute\ din\ E_1\ si\ F_2\ este\ o\ expresie\ generala$
7. *Comutativitatea selecției cu reuniunea*
   a. $\sigma_F(E_1 \mathrm{U} E_2) = \sigma_F(E_1) \mathrm{U} \sigma_F(E_2)$
8. *Comutativitatea selecției cu diferența*
   a. $\sigma_F(E_1 - E_2) = \sigma_F(E_1) - \sigma_F(E_2)$
9. *Comutativitatea proiecției cu produsul cartezian*
   a. $\pi_{A_1 \dots A_n, B_1 \dots B_m}(E_1 \times E_2) = \pi_{A_1 \dots A_n}(E_1) \times \pi_{B_1 \dots B_m}(E_2)$
10. *Comutativitatea proiecției cu reuniunea*
    a. $\pi_{A_1 \dots A_n}(E_1 \mathrm{U} E_2) = \pi_{A_1 \dots A_n}(E_1) \mathrm{U} \pi_{A_1 \dots A_n}(E_2)$

Prin deplasarea operațiilor de selecție cât mai în stânga expresiilor algebrice se reduce numărul de tupluri care trebuie manipulate în procesul de execuție a cererii.

Strategii generale de optimizare a cererilor

1. Execuția cererilor de selecție înaintea operațiilor de join sau produs cartezian
2. Execuția operațiilor de proiecție înaintea operațiilor de join
3. Execuția cererilor de selecție înaintea proiecțiilor

Algoritm de optimizare a expresiilor relaționale

1. Fiecare selecție este transformată într-o cascadă de selecții, folosind regula de echivalență R_4
2. Fiecare selecție este deplasată în jos folosind regulile R_4 – R_8 cât mai aproape de frunza
3. Folosind regulile R_3, R_5, R_9 și R_{10}, fiecare proiecție este deplasată cât mai jos posibil în arborele sintactic. Dacă o proiecție ajunge să fie făcută după toate atributele atunci va fi eliminată.
4. Cascadele de selecții și proiecții sunt combinate într-o singură selecție folosind regulile R3 – R_5, o singură proiecție sau o selecție urmată de o proiecție.
5. Nodurile interioare ale arborelui rezultat sunt împărțite în grupuri. Fiecare nod intern care corespunde unei operații binare devine un grup împreună cu predecesorii săi imediați cu care sunt asociate operațiile unare. Din bloc face parte de asemenea orice lanț de noduri succesoare asociate cu operații unare și terminate cu o frunză, cu excepția cazului când operația binară este un produs cartezian neurmat de o selecție cu care se formează un equi-join.
6. Se evaluează fiecare grup astfel încât niciunul nu este evaluat înaintea descendenților săi. Rezultă astfel un program de evaluare a expresiei relaționale inițiale.

Optimizarea cererilor într-o bază de date relațională

1. Translatarea cererilor cu ajutorul algebrei relaționale (expresii algebrice relaționale)
2. Transformarea în expresii echivalente care pot să fie executate cât mai eficient
    a. Logic – se aplică prioritatea operațiilor (execuția operațiilor de proiecție înaintea operațiilor de join, execuția selecțiilor înaintea proiecțiilor, combinarea selecțiilor multiple etc.)
    b. Semantic – transformările sunt bazate pe proprietățile atributelor
3. Reprezentarea expresiilor echivalente cu ajutorul grafurilor de strategii
4. Pentru fiecare cerere se estimează costul de execuție bazându-se pe estimări ale parametrilor relevanți (de exemplu numărul tuplurilor)
5. În final se identifică costul de execuție cel mai mic

Graful de strategii – reprezintă o metodă pentru studierea tehnicilor de optimizare a interogărilor și un instrument foarte util pentru descrierea și studierea optimizării.

1. Pentru construirea grafurilor de strategii se folosesc dreptunghiuri pentru reprezentarea tabelelor și cercuri pentru operatori
2. Nodurile sunt asimilate tabelelor și operatorilor.
    a. Graful are un nod rezultat, fiind un nod tabela care reprezintă rezultatul interogării.
    b. Nodurile la care nu sosesc arcuri se numesc noduri de bază, iar celelalte sunt noduri intermediare.
3. Arcele au următoarea semnificație:
    a. Dacă tabela n_1 este intrare pentru operatorul n_2 atunci n_1 -> n_2;
    b. Dacă tabela n_3 rezultă prin aplicarea operatorului n_2 atunci n_2 -> n_3.
4. Dacă fiecare nod tabel are cel mult o intrare, graful va reprezenta o singură strategie.
5. Dacă sunt mai multe intrări într-un nod tabelă, atunci graful de strategii poate reprezenta complet mai multe strategii candidat pentru interogarea propusă.

Elemente de cost impuse de strategie

Produs cartezian între R și S cu r, respectiv s tupluri. Fie nr și ns numărul de tupluri din fiecare relație care încap într-un bloc pe disc și b numărul de blocuri disponibile în memoria internă. Se citesc numărul maxim posibil de blocuri din R (b-1 blocuri) și pentru fiecare înregistrare din R se citește în întregime în ultimul bloc disponibil relația S. În acest caz numărul de accesări este mai puternic

influențat de r/nr decât de s/ns, astfel se va considera ca relație R pe cea care are acest raport mai mic, dintre cele două relații.

1. Pentru implementarea joinurilor se folosesc de asemenea diverse metode:
    a. Sortarea uneia dintre relații după atributele implicate în join (în cazul equi-joinului)
    b. Folosirea indecșilor (dacă există) pe atributele implicate în join, pentru acces direct la tuplurile care dau elemente ale rezultatului
    c. Micșorarea numărului de tupluri luate în calcul, prin aplicarea mai întâi a selecțiilor, dacă acestea sunt prezente în cerere
2. În general, există o serie de principii care duc la micșorarea numărului de accesări ale discului:
    a. Realizarea cu prioritate a selecțiilor
    b. Combinarea anumitor selecții cu produse carteziene adiacente pentru a forma un join
    c. Combinarea secvențelor de operații unare (selecții sau proiecții) într-una singură
    d. Căutarea subexpresiilor comune, pentru a fi evaluate o singură dată
    e. Folosirea indecșilor sau sortarea relațiilor
    f. Evaluarea diverselor strategii posibile înainte de a începe procesul de calcul efectiv

Etapele execuției unei cereri SQL

1. **Parsarea** – detectarea eventualelor erori de sintaxă și cătarea unei copii deja parsate în memoria shared pool
2. **Optimizarea** – generarea mai multor planuri de execuție și identificarea celui optim
3. **Generarea** – generarea de cod pentru planul de execuție optim
4. **Execuția** – cererea este executată cu costuri minime

Resurse Utile

Online SQL Compiler: https://www.tutorialspoint.com/execute_sql_online.php (suport doar pentru left outer join, a se vedea exemplul)
SQL Fiddle: http://www.sqlfiddle.com/#!9/5d5f7/1 (suport doar pentru left și right outer join)

```
-- BEGIN TRANSACTION; /* uncomment for first */

CREATE TABLE STUD(Id integer, Nume text, IdS
integer);
CREATE TABLE SPEC(IdS integer, NumeS text,
NrStud integer);
INSERT INTO STUD VALUES(1, 'Ion', 10);
INSERT INTO STUD VALUES(2, 'Elena', 11);
INSERT INTO STUD VALUES(3, 'Vasile', 10);
INSERT INTO STUD VALUES(4, 'Maria', 14);
INSERT INTO SPEC VALUES(10, 'C', 450);
INSERT INTO SPEC VALUES(11, 'TI', 200);
INSERT INTO SPEC VALUES(12, 'IS', 400);
-- COMMIT; /* uncomment for first */
```

```
/* for first: only left
outer join */
SELECT *
FROM STUD LEFT OUTER JOIN
SPEC ON STUD.IdS = SPEC.IdS;
/* for second: full outer
join with union */
SELECT *
FROM STUD LEFT OUTER JOIN
SPEC ON STUD.IdS = SPEC.IdS
UNION
SELECT *
FROM STUD RIGHT OUTER JOIN
SPEC ON STUD.IdS = SPEC.IdS;
```

Relational Database Tools: http://raymondcho.net/RelationalDatabaseTools (suport pentru *relații, dependențe funcționale, chei, forme normale, descompuneri* și multe altele)