Optimizarea interogarilor bazelor de date relationale (partea 1)

Ioana Ciuciu
oana@cs.ubbcluj.ro
http://www.cs.ubbcluj.ro/~oana/

Planificare

Saptamana	Curs	Seminar	Laborator
S1	1. Concepte fundamentale ale bazelor de date	1. Modelul Entitate-Relatie. Modelul relational	1. Modelarea unei BD in modelul ER si implementarea ei in SQL Server
S2	2. Modelare conceptuala		
S3	3. Modelul relational de organizare a bazelor de date	2. Limbajul SQL – definirea si actualizarea datelor	2. Interogari SQL
S4	4. Gestiunea bazelor de date relationale cu limbajul SQL		
S5-6	5-6. Dependente functionale, forme normale	3. Limbajul SQL – regasirea datelor	3. Interogari SQL avansate
S7	7. Interogarea bazelor de date relationale cu operatori din algebra relationala	4. Proceduri stocate	
S8	8. Structura fizica a bazelor de date relationale		
S9-10-11	9-10-11. Indecsi. Arbori B. Fisiere cu acces direct	5. View-uri. Functii definite de utilizator. Executie dinamica	4. Proceduri stocate. View. Trigger
		6. Formele normale ale unei relatii. Indecsi	
S12	12. Evaluarea interogarilor in bazele de date relationale		
S13	13. Extensii ale modelului relational si baze de date NoSQL	7. Probleme	Examen practic
S14	14. Aplicatii		

Planul cursului

- Curs 11
  - Executia instructiunilor SQL
  - Evaluarea operatorilor relationali

Executia instructiunilor SQL

- In aplicatii (la client) se cere executia unei instructiuni SQL

- **Caracteristici**:
  - se doreste un **timp de raspuns minim**, la orice interogare
  - la crearea instructiunii SQL nu e necesar sa se cunoasca structura fizica a bazei de date

Executia instructiunilor SQL

- **Etape in executia instructiunilor**:
  - **la client**: instruciunea SQL (limbaj neprocedural) este generata si trimisa la server
  - **la server** (unele dintre etape se pot parcurge in paralel):
    - se face **analiza** instructiunii SQL (analiza sintactica, semantica)
    - se translateaza intr-o **forma interna** (expresie in algebra relationala)
    - forma interna se transforma intr-o **forma optimala**
    - se genereaza un **plan de executie** (procedural)
    - se **evalueaza** planul generat si se trimite rezultatul la client

Executia instructiunilor SQL

- **Planurile de executie** generate se analizeaza:
  - periodic
  - dupa schimbarea structurii fizice
  - dupa multe operatii de modificare a tabelelor
- Din aceste analize se pot deduce **solutii de optimizare**, deci de modificare a structurii fizice
- La generarea planului de executie se folosesc **operatorii din algebra relationala**

Executia instructiunilor SQL

- Pentru **interogare** sunt necesari urmatorii operatori:

  - **Selectia**: $\sigma_c(R)$
  - **Proiectia**: $\Pi_\alpha(R)$
  - **Produsul cartezian**: R1×R2
  - **Reuniunea**: $R_1 \cup R_2$
  - **Diferenta**: $R_1 - R_2$
  - **Intersectia**: $R_1 \cap R_2$
  - **Joinul conditional**: $R_1 \otimes_\Theta R_2$
  - **Joinul natural**: $R_1 * R_2$
  - **Joinul extern stanga**: R1 ⟕c R2
  - **Joinul extern dreapta**: R1 ⟖c R2
  - **Joinul extern complet**: R1 ⟗c R2
  - **Semi joinul stanga**: R1 ⊳ R2
  - **Semi joinul dreapta**: R1 ⊲ R2
  - **Câtul**: R1÷R2
  - **Eliminarea duplicarilor**: $\partial\ (R)$
  - **Sortarea inregistrarilor**: $S_{\{lista\}}(R)$
  - **Gruparea**: $G_{\{lista1\}\ group\ by\ \{lista2\}}(R)$

Executia instructiunilor SQL

- In SQL, o interogare poate fi scrisa in mai multe moduri

- **Exemplu** pentru o bdr (cheile primare sunt subliniate, cheile externe sunt marcate cu albastru):
  - `studenti(cnp, nume, prenume, grupa, media, adresa, email)`
  - `sectii(cod, denumire, act_functionare)`
  - `grupe(cod, sectia, anstudiu)`

- **Interogare**. Se cer studentii (nume, prenume, anstudiu, denumire sectie, media) de la o sectie data (ex. cu codul 2, poate fi parametru), cu media cel putin 9 (poate fi parametru). Inregistrarire se cer ordonate alfabetic pe sectii si ani de studiu.

Executia instructiunilor SQL

```
select nume,prenume, anstudiu, denumire, media
       from studenti st, grupe gr, sectii se
       where st.grupa=gr.cod and gr.sectia=se.cod
       and sectia=2 and media>=9
       order by anstudiu, nume, prenume
```

```
select nume,prenume, anstudiu, denumire, media
       from (studenti st inner join grupe gr on st.grupa=gr.cod)
       inner join sectii se on gr.sectia=se.cod
       where sectia=2 and media>=9
       order by anstudiu, nume, prenume
```

```
select nume, prenume, anstudiu, denumire, media
        from
       ((select nume, prenume, grupa, media from studenti where media>=9)
       st inner join
       (select * from grupe where sectia=2) gr on st.grupa=gr.cod
       ) inner join (select * from sectii where sectia=2) se
       on gr.sectia=se.cod
       order by anstudiu, nume, prenume
```

Executia instructiunilor SQL

- Toate aceste variante sunt echivalente (furnizeaza acelasi rezultat)
- Expresiile echivalente în algebra relationala sunt urmatoarele:

$$\prod_{\beta}(s_{\alpha}(\sigma_{C}(\text{studenti x grupe x sectii})))$$

$$\prod_{\beta}(s_{\alpha}(\sigma_{C1}((\text{studenti} \otimes_{C2} \text{grupe}) \otimes_{C3} \text{sectii})))$$

$$\prod_{\beta}(s_{\alpha}(\sigma_{C1}((\sigma_{C2}(\text{studenti}) \otimes_{C3}(\sigma_{C4}(\text{grupe}))) \otimes_{C5}(\sigma_{C6}(\text{sectii})))))$$

Executia instructiunilor SQL

- **Pentru o expresie in algebra relationala se poate construi un arbore de evaluare**

- **Probleme**:
  - care varianta este mai avantajoasa?
  - ce parametri se optimizeaza la generarea planului de executie?
  - la generarea planului de executie, ce informatii sunt necesare?
  - ce poate face un optimizator (componenta a SGBD)?

Evaluarea operatorilor relationali

- Operanzii pentru operatorii relationali sunt:
  - **tabele din baza de date -** pot sa existe **indecsi** atasati
  - **viewuri sau tabele temporare** (obtinute prin evaluarea unor operatori relationali) – pentru care nu exista indecsi

- **Pentru un operator din algebra relationala pot exista mai multi algoritmi de evaluare**

- La generarea planului de executie se alege operatorul cu complexitatea cea mai mica in contextul existent al bazei de date
  - se iau informatii din dictionarul bazei de date si informatii statistice preluate intr-o perioada de timp de la executia diverselor instructiuni

Evaluarea operatorilor relationali - Algoritmi

A1. Table Scan

- Pentru multi operatori este necesara parcurgerea tuturor inregistrarilor dintr-un tabel (in ordinea furnizata de unele tabele de alocare, de un clustered index sau un un nonclustered index)
- Fie **bR** numarul de blocuri in care se memoreaza inregistrarile tabelului
  - Din algoritmul de cautare secventiala se deduce ca aproximativ **bR/2** blocuri sunt necesare (in medie) la o cautare secventiala dupa valorile unei chei
  - Toate blocurile care contin inregistrari trebuie transferate în memoria interna (la o cautare secventiala dupa valorile unui atribut, selectie dupa o conditie complexa)
  - Timpul de transfer este mare pentru tabele cu multe inregistrari

Evaluarea operatorilor relationali - Algoritmi

A2. Index Seek

- Acest algoritm este cel mai performant si se foloseste la **cautarea dupa valoarea unei chei C**
- Cautarea este de forma: **C=K0**

- Aceasta cautare este **explicit** precizata (cautare in tabel, evaluare join) sau **implicita** (verificarea restrictiei de cheie).

- Se consulta un index (construit implicit pentru **restrictia de cheie unica** sau construit prin instructiunea **create index**) memorat ca un **B-arbore** sau **B+-arbore**.

- **Observatie**. **C** poate fi o cheie simpla, compusa, sau o expresie

Evaluarea operatorilor relationali - Algoritmi

A3. Index Scan

- Acest algoritm se foloseste pentru evaluarea operatorului $\sigma_c(R)$, unde conditia **C** este de forma:
- **A<v, A<=v, A>v, A>=v, A is null, A is not null**, indexul s-a construit pentru o **cheie A**
- **A=v, A<v, A<=v, A>v, A>=v, A is null, A is not null**, indexul s-a construit pentru un **atribut** sau o **expresie A**

- Inregistrarile se obtin prin parcurgerea partiala sau totala a indexului si citirea inregistrarilor din relatie conform adreselor furnizate de index (unele blocuri se pot citi de mai multe ori)

Evaluarea operatorilor relationali - Algoritmi

A3. Index Scan

- **Observatii**.
  - Timpul necesar pentru executia acestui algoritm poate fi mare si e posibil ca algoritmul **table scan** sa fie mai avantajos. Daca se poate estima numarul de inregistrari furnizate, atunci se poate deduce care algoritm ar fi mai eficient. Aceasta estimare se poate face daca se determina numarul de inregistrari obtinute de aceeasi cautare la evaluari anterioare (intr-un "istoric al evaluarilor")
  - Daca o astfel de estimare nu este posibila, atunci prin sirul de caractere din instructiunea select-SQL se **poate preciza** varianta propusa de client
  - Pentru o conditie de forma "**A<>v**" se poate folosi un algoritm **table scan** sau sa sefoloseasca de doua ori algoritmul index scan (**A<>v** ≡ (**A<v**) or (**A>v**))

Evaluarea operatorilor relationali - Algoritmi

A3. Index Scan

- **Observatii**.
  - Pentru o conditie de forma "(**A=v) and C1**", unde pentru **A** s-a construit un index, iar **C1** este de una din formele amintite la **index scan**, avem urmatoarea evaluare:

$$\sigma_C(R) = \sigma_{C1}(\sigma_{A=\upsilon}(R))$$

    - Pentru selectia interioara se foloseste un **index seek** sau **index scan**, iar pentru selectia exterioara un **index scan**
    - Scrierea conditiei **C** in aceasta forma este utila daca sistemul **nu poate** sau **nu face** o transformare a conditiei (din analiza planului de executie se observa daca sistemul transforma conditia in aceasta varianta)
    - Se poate descrie un algoritm pentru evaluarea operatorului de selectie prin sortarea sursei de date

Evaluarea operatorilor relationali - Algoritmi

A4. Cross Join

- Acest algoritm se foloseste la evaluarea unui produs cartezian:
  - **R cross join S**
  - **R inner join S on true**
  - **select ... from R, S ...,** iar intre R si S nu exista conditie de legatura

Evaluarea operatorilor relationali - Algoritmi

A4. Cross Join

- Fie:
  - **bR, bS** - numarul de blocuri in care se memoreaza **R**, respectiv **S**
  - **m, n** - numarul de blocuri ce pot apare simultan in memoria interna din R, respectiv S (in memoria interna exista m+n zone tampon pentru cele doua tabele)

Evaluarea operatorilor relationali - Algoritmi

A4. Cross Join

- Pentru a genera produsul cartezian {(r, s) | r∈R, s∈S} se poate folosi urmatorul algoritm:

- pentru fiecare grup de maxim **m** blocuri din **R**:
  - se citeste grupul de blocuri din **R** in memoria interna. Fie **M1** multimea de inregistrari din aceste blocuri
  - pentru fiecare grup de maxim **n** blocuri din **S**:
    - se citeste grupul de blocuri din **S** in memoria interna. Fie **M2** multimea de inregistrari din aceste blocuri
    - pentru fiecare **r** ∈ **M1**:
      - pentru fiecare **s** ∈ **M2** se include **(r, s)** in rezultat

Evaluarea operatorilor relationali - Algoritmi

A4. Cross Join

- Complexitatea algoritmului: **numarul total de blocuri citite** (din cele doua tabele):

$$b_R + \left\lceil \frac{b_R}{m} \right\rceil * b_S$$

(numarul de blocuri citite din R, pentru fiecare grup de maxim **m** blocuri din R se citeste S)

- Pentru a minimiza aceasta valoare ar trebui sa alegem pe **m** maxim posibil (restul operanzilor sunt constante)
- In memoria interna se poate alege o singura zona tampon pentru S (deci n=1) si spatiul ramas sa se foloseasca la maxim pentru R (m maxim)
- Daca se face un schimb intre cele doua relatii (in algoritm si in calculul complexitatii) se obtine o complexitate:

$$b_S + \left\lceil \frac{b_S}{n} \right\rceil * b_R$$

Evaluarea operatorilor relationali - Algoritmi

A4. Cross Join

- Dintre cele doua variante de lucru este avantajoasa varianta care da complexitatea cea mai mica
- Daca dimensiunea zonelor tampon este aceeasi pentru R si S, atunci este optima situatia in care ca **prim tabel** se ia acela cu dimensiune mai mica (min{ bR, bS })

- **Observatie**. Daca una din sursele de date incape in memoria interna, atunci complexitatea algoritmului este **bR + bS**

Evaluarea operatorilor relationali - Algoritmi

A5. Nested Loops Join

- Algoritmul Cross Join se poate folosi si la evaluarea unui join (intern, extern, semijoin) intre doua tabele daca se foloseste o **conditie compusa pentru join**

- La fiecare element **(r, s)** din produsul cartezian se evalueaza si conditia din operatorul de join

Evaluarea operatorilor relationali - Algoritmi

A6. Indexed Nested Loops Join

- Acest algoritm se foloseste pentru evaluarea $R \otimes_{\Theta} S$, unde C ≡ (R.A=S.B), iar pentru A (din R) sau B (din S) **exista un index**

- In descrierea algoritmului presupunem ca pentru B din tabelul S exista un index

Evaluarea operatorilor relationali - Algoritmi

A6. Indexed Nested Loops Join

pentru fiecare bloc din R:

- citeste blocul din R în memoria interna. Fie M multimea de înregistrari citite
- pentru fiecare **r** ∈ **M**:
  - determina v = **A**(r)
  - folosind indexul pentru B din S se determina înregistrarile **s** ∈ **S** cu valoarea v pentru B (în functie de tipul de index, pot fi mai multe înregistrari). Pentru fiecare înregistrare **s** determinata, perechea **(r,s)** se adauga la rezultat

Evaluarea operatorilor relationali - Algoritmi

A7. Merge Join

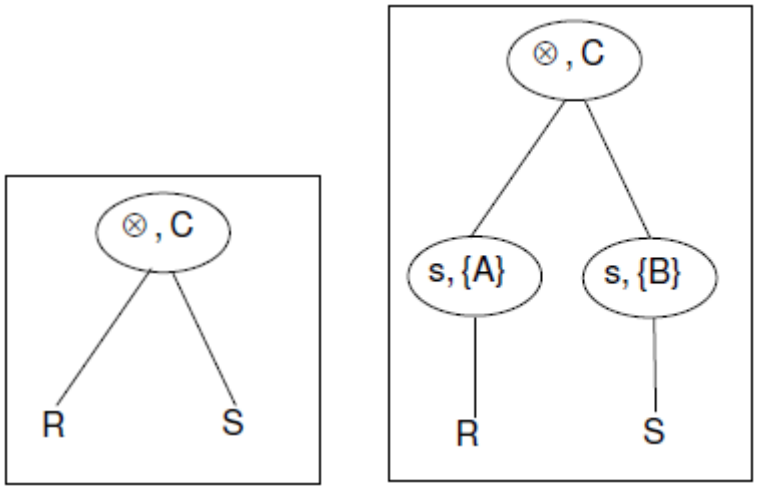
- Acest algoritm se foloseste pentru evaluarea $R \otimes_{\theta} S$, unde C ≡ (R.A=S.B), iar pentru A (din R) sau B (din S) **nu exista index**

- se sorteaza cele doua tabele dupa coloanele folosite la join: **R dupa A** si **S dupa B**
- se parcurg în paralel cele doua tabele obtinute. Fie r din A si s din B doua înregistrari curente
  - daca r.A=s.B, atunci (r',s') se adauga la rezultat; r' este in multimea inregistrarilor consecutive din R cu valoarea pentru A egala cu r.A, s' analog pentru S; next(r); next(s) (se ia cate o înregistrare cu urmtoarea valoare pentru A si S)
  - daca r.A<s.B, atunci next(r) (se determina o înregistrare din R sortat cu urmatoarea valoare pentru A)
  - daca r.A>s.B, atunci next(s) (se determina o înregistrare din S sortat cu urmatoarea valoare pentru B)

Evaluarea operatorilor relationali - Algoritmi

A7. Merge Join

- Acest algoritm inlocuieste un arbore de evaluare cu alt arbore



Evaluarea operatorilor relationali - Algoritmi

A8. Hybrid Merge Join

- Acest algoritm se foloseste pentru evaluarea $R \otimes_{\Theta} S$, unde C ≡ (R.A=S.B), iar pentru unul dintre atributele: A (din R) sau B (din S) este sortata sursa sau se sorteaza, iar pentru celalalt atribut exista un index de tip B+-arbore

- In acest caz se face o interclasare intre sursa sortata si terminalele aflate în indexul B+-arbore construit pentru al doilea tabel

Evaluarea operatorilor relationali - Algoritmi

A9. Hash Join

- Acest algoritm se foloseste pentru evaluarea $R \otimes_{\Theta} S$, unde C ≡ (R.A = S.B)

1. Se alege unul dintre tabele (se recomanda cel cu dimensiune mai mica) si se memoreaza (în memoria interna, sau în fisiere temporare) pe blocuri, conform unei functii de dispersare **h**. Presupunem ca acest tabel este S
2. Pentru fiecare r ∈ R se determina v = h(r.A). Inregistrarea r se compara numai cu inregistrarile din S care sunt in blocul h(v)

Evaluarea operatorilor relationali - Algoritmi

A9. Hash Join

- **Observatii**.
- Eficienta este mare daca unul dintre tabele se poate memora in memoria interna
- Rezultate bune se obtin si pentru tabele cu dimensiune mare.
- In cazul in care conditia de join este de forma:

  C = (R.A1 = S.B1) [and (R.A2 = S.B2)] ...

  si pentru una din coloanele A1, A2, ..., B1, B2, ... exista un index, atunci se poate folosi unul din algoritmii **Merge Join** sau **Hash Join** modificat
- In cazul în care conditia de join este complexa, se poate aduce la **forma normala conjunctiva** si sa se analizeze fiecare termen

Evaluarea operatorilor relationali - Algoritmi

Joinuri externe

- Se adapteaza algoritmii de la joinurile conditionale

Joinuri multiple

- La evaluarea unor joinuri multiple se pot folosi proprietatile de asociativitate si comutativitate
- Exemplu pentru joinul natural a patru tabele folosind numai asociativitatea
  - ((R1 * R2) * R3) * R4
  - (R1 * R2) * (R3 * R4)
  - (R1 * (R2 * (R3 * R4))
  - (R1 * (R2 * R3)) * R4
  - R1 * ((R2 * R3) * R4)

Evaluarea operatorilor relationali - Algoritmi

Joinuri multiple

- La fiecare expresie se poate asocia un arbore de evaluare, unde in nodurile interioare sunt rezultate intermediare
- Dintre planurile de executie posibile se foloseste acela cu un "timp de evaluare estimat" minim

Operatii pe multimi de inregistrari:

$$R_1 \cup R_2 \qquad R_1 - R_2 \qquad R_1 \cap R_2$$

- Se adapteaza algoritmii precedenti (cei de la join)

Bibliografie

- Leon Tambulea, Curs de Baze de Date, UBB Cluj
- [Si10] SILBERSCHATZ A., KORTZ H., SUDARSHAN S., *Database System Concepts*, McGraw-Hill, 2010, cap. 13
- [Si10a] SILBERSCHATZ A., KORTZ H., SUDARSHAN S., http://codex.cs.yale.edu/avi/dbbook/db6/slide-dir/PDF-dir/ch12.pdf, 2010
- [Ga08] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., *Database Systems: The Complete Book*, Pearson Prentice Hall, 2008, cap.16
- [Da04] DATE, C.J., An Introduction to Database Systems (8th Edition), Addison-Wesley, 2004, cap 13
- [Kn76] KNUTH, D.E., *Tratat de programare a calculatoarelor. Sortare si cautare*. Ed. Tehnica, Bucuresti 1976