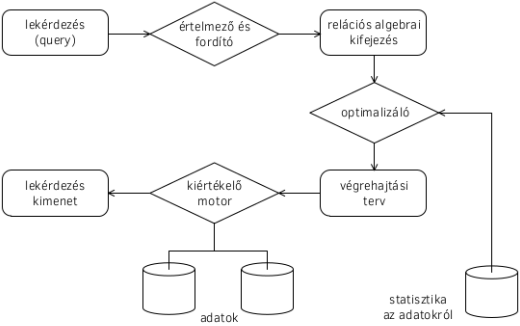
Relációs lekérdezések optimalizálása

Marton József

BME-TMIT
Gajdos Sándor diasorának felhasználásával

Adatbázisok VITMAB00
2023. október 16.

A lekérdezés-feldolgozás folyamata I.



- Cél: az adatok adatbázisból való kinyerése
- Mivel: egyértelmű, deklaratív megfogalmazás
- Hogyan: lássuk...

A lekérdezés-feldolgozás folyamata II.

1. Elemzés (szintaktikus), fordítás
   - helyesség-vizsgálat
   - valamilyen belső reprezentációba hozzuk
2. Költségoptimalizálás
   - Egyértelmű a lekérdezés, ill. a belső reprezentáció
     - a kiértékelés módját és
     - a lépések sorrendjét tekintve?
   - Formális módszerekkel ekvivalens alakok készítése
   - Hogyan kell kiértékelni?
   - Jobb-e egyik mint a másik (optimális)? Mi szerint optimális?
   - Összefoglalva: optimalizációs stratégiák alapján végrehajtási terveket kell készíteni, amelyeket előbb értékelni kell, majd közülük a legjobbat kiválasztani
3. Kiértékelés

Példa I.

SQL és relációalgebra

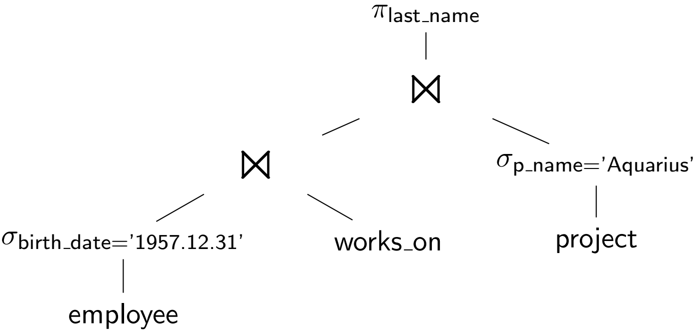
- Employee (employee_id, last_name, first_name, birth_date, ...)
- Project (project_id, p_name, ...)
- Works_on (project_id, employee_id)

```
select last_name
  from employee, works_on, project
 where employee.birth_date = '1957.12.31'
   and works_on.project_id = project.project_id
   and works_on.employee_id = employee.employee_id
   and project.p_name = 'Aquarius'
```

$$\pi_{last\_name}\left(\left(\sigma_{birth\_date='1957.12.31'}(E)\right) \bowtie W \bowtie \left(\sigma_{p\_name='Aquarius'}(P)\right)\right)$$

Példa II.

Egy lehetséges relációalgebrai fa



A lekérdezés-feldolgozás folyamata III.

1. Példa: SQL, relációs algebrai fa
2. Elemi műveletek (kiértékelési primitívek). Relációs algebrai belső reprezentáció esetén ezek ,,sorrendje" (egymásra épülése) a relációs algebrai fa.
3. Hogyan kell az egyes műveleteket: egy szelekciót végrehajtani? (lineáris, bináris, index?) És a join?
4. Hogyan kell a műveletek összességét kiértékelni? Materializáció/pipelining (workflow)
5. A végrehajtási terv:
   - 5.1 műveletek és ,,sorrendjük" (relációs algebrai fa)
   - 5.2 algoritmus-hozzárendelés
   - 5.3 workflow-választás

Egy operandusú műveletek azonosságai

1. Szelekció kaszkádosítása:

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. A szelekció kommutativitása:

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

3. Projekció kaszkádosítása, ha $L_1 \subseteq L_2 \subseteq \ldots \subseteq L_n$:

$$\pi_{L_1}(\pi_{L_2}(\ldots\pi_{L_n}(E)\ldots)) = \pi_{L_1}(E)$$

Illesztés-jellegű műveletek azonosságai

4. A Θ-illesztés és a Descartes-szorzat kapcsolata:

$$\sigma_\theta (E_1 \times E_2) = E_1 \underset{\theta}{\bowtie} E_2$$

$$\sigma_{\theta_1} \left(E_1 \underset{\theta_2}{\bowtie} E_2 \right) = E_1 \underset{\theta_1 \wedge \theta_2}{\bowtie} E_2$$

5. A Θ-illesztés kommutativitása:

$$E_1 \underset{\theta}{\bowtie} E_2 = E_2 \underset{\theta}{\bowtie} E_1$$

6. A természetes illesztés asszociativitása (Descartes-szorzat hasonlóan):

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

Illesztés-jellegű műveletek azonosságai

folytatás

7. A szelekció művelet disztributivitása a Θ-illesztés felett, ha a θ_0 csak E_1-beli attribútumokat tartalmaz:

$$\sigma_{\theta_0}\left(E_1 \underset{\theta}{\bowtie} E_2\right) = \sigma_{\theta_0}(E_1) \underset{\theta}{\bowtie} E_2$$

8. A projekció disztributív a Θ-illesztés felett, ha L_1 és L_2 E_1, illetve E_2-beli attribútumokat tartalmaz, és az illesztés feltételében csak $L_1 \cup L_2$-beli attribútumok vannak:

$$\pi_{L_1 \cup L_2}\left(E_1 \underset{\theta}{\bowtie} E_2\right) = \left(\pi_{L_1}\left(E_1\right)\right) \underset{\theta}{\bowtie} \left(\pi_{L_2}\left(E_2\right)\right)$$

Heurisztikus (szabály-alapú) optimalizálás

Tapasztalatok alapján:

- Átalakítási lépések
  1. kiindulás: kanonikus alak
  2. szelekciók süllyesztése (kaszkádosítás után)
  3. levelek átrendezése (asszociativitás)
  4. Θ-illesztés bevezetése
  5. projekció süllyesztése (újak bevezetése)
- Algoritmus- és workflow-hozzárendelés

Relációalgebrai kifejezés kanonikus alakja:

- egyetlen projekció
- egyetlen szelekció
- Descartes-szorzatok

Példa I.

SQL és relációalgebra

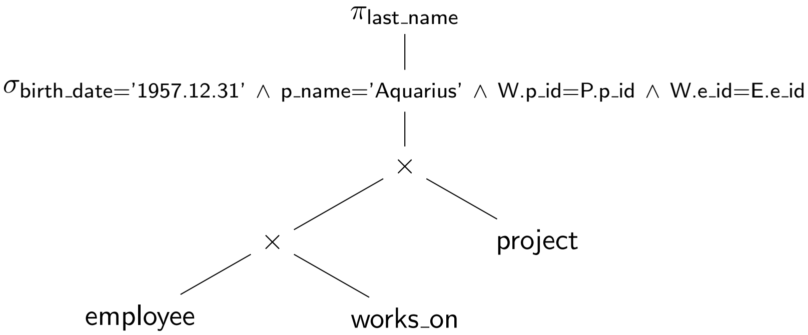
- Employee (employee_id, last_name, first_name, birth_date, ...)
- Project (project_id, p_name, ...)
- Works_on (project_id, employee_id)

```
select last_name
  from employee, works_on, project
 where employee.birth_date = '1957.12.31'
   and works_on.project_id = project.project_id
   and works_on.employee_id = employee.employee_id
   and project.p_name = 'Aquarius'
```

$$\pi_{last\_name}\left(\left(\sigma_{birth\_date='1957.12.31'}(E)\right) \bowtie W \bowtie \left(\sigma_{p\_name='Aquarius'}(P)\right)\right)$$

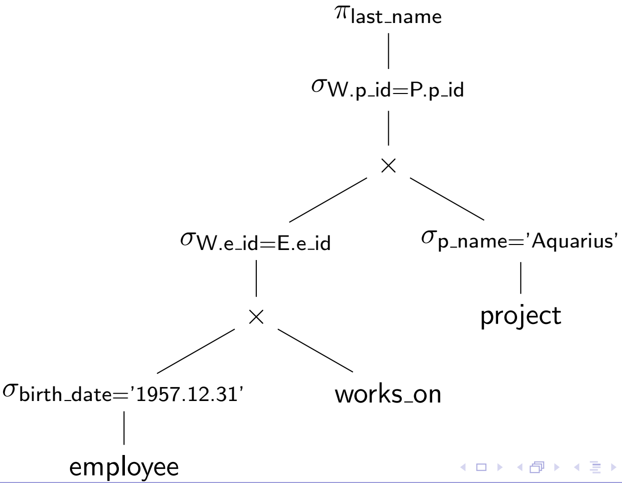
Kiindulás

Kanonikus alak



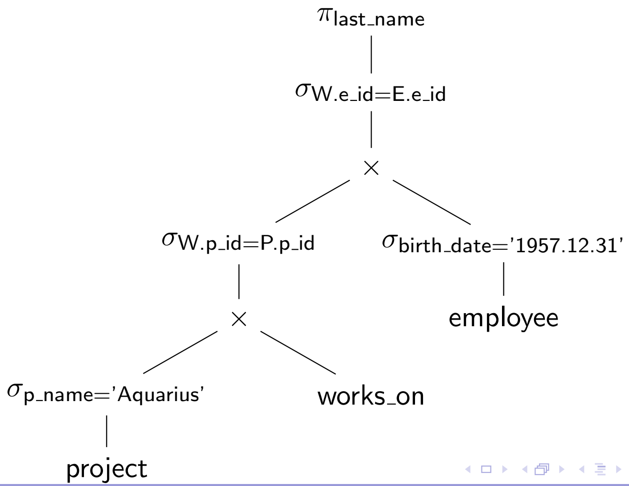
Átalakítás

Szelekciók süllyesztése – kaszkádosítás után



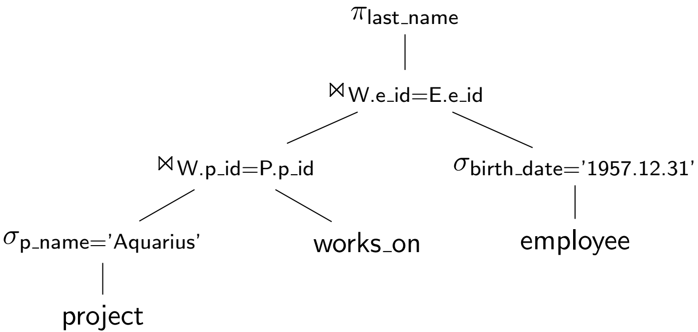
Átalakítás

Levelek átrendezése



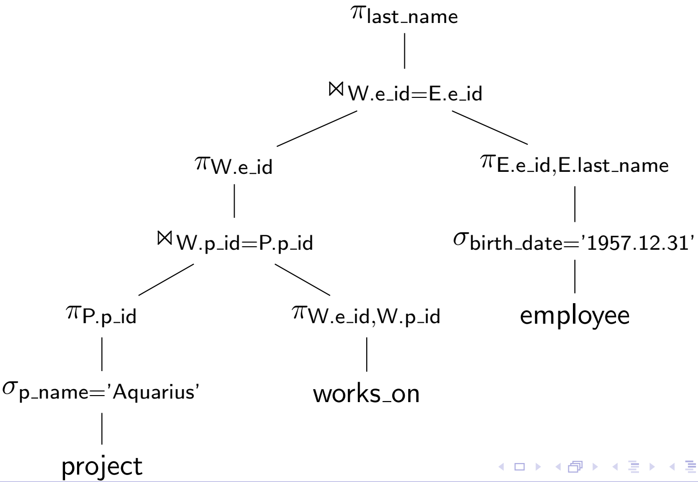
Átalakítás

Θ-illesztés bevezetése



Átalakítás

Projekció süllyesztése – újak bevezetése



Költség-alapú optimalizálás

Áttekintés

1. Szintaktikai elemzés, fordítás
2. Költségoptimalizálás
3. Kiértékelés

Költség-mérték

- lehetne: válaszidő, CPU idő, más erőforrás-szükséglet
- legyen a költség a diszk blokkok olvasásának és írásának a száma azzal a további megszorítással, hogy az írásba csak a köztes blokkírások számát számítjuk bele, hiszen a végeredmény kiírása mindenképpen szükséges.
- E_{alg}: az algoritmus becsült költsége

Katalógusinformáció

A relációkról

n_r: az r reláció rekordszáma

b_r: az r relációt tartalmazó blokkok száma

s_r: az r reláció egy rekordjának nagysága bájtokban

f_r: mennyi rekord fér az r reláció egy blokkjába

$\mathrm{V}(A, r)$: hány különböző értéke fordul elő az A attribútumnak az r relációban. $\mathrm{V}(A, r) = |\pi_A(r)|$. Speciálisan, ha az A kulcs, akkor $\mathrm{V}(A, r) = n_r$.

$\mathrm{SC}(A, r)$: az A attribútumra egyenlőségi feltételt kielégítő rekordok átlagos száma, ha legalább egy rekord kielégíti ezt az egyenlőségi feltételt.
Ha A szuperkulcs: $\mathrm{SC}(A, r) = 1$.
Általánosságban: $\mathrm{SC}(A, r) = \frac{n_r}{\mathrm{V}(A,r)}$.

Katalógusinformáció

Az indexekről

f_i: az átlagos pointer-szám a fa struktúrájú indexek csomópontjaiban, mint pl. a B* fáknál, azaz a csomópontokból induló ágak átlagos száma.

HT_i: az i index szintjeinek a száma, azaz az index magassága (Height of Tree). Az r relációt tartalmazó heap-szervezésű állományra épített B* fa esetén $\mathrm{HT}_i = \lceil \log_{f_i} b_r \rceil$, ill. hash-állománynál $\mathrm{HT}_i = 1$.

LB_i: az i index legalsó szintű blokkjainak a száma, azaz a levélszintű indexblokkok száma (Lowest level index Block).

Műveletek és algoritmusok

- szelekció egyenlőségi feltételre
  - alap: lineáris, bináris
  - indexelt: elsődleges index kulcson, elsődleges index nem kulcson, másodlagos index
- join
  - típusai: $\bowtie$, Θ-illesztés, külső illesztések
  - algoritmusok: jön...
- egyéb műveletek
  - rendezés
  - ismétlődések szűrése, projekció
  - unió, metszet, különbség
  - aggregáció

Szelekciós algoritmusok

Egyenlőségi feltételre

A1: Lineáris keresés: $E_{A1} = b_r$

A2: Bináris keresés: $E_{A2} = \lceil \log_2 b_r \rceil + \left\lceil \frac{\mathrm{SC}(A,r)}{f_r} \right\rceil - 1$

A3: Elsődleges index használatával, egyenlőségi feltételt a kulcson vizsgálunk: $E_{A3} = \mathrm{HT}_i + 1$

A4: Elsődleges index használatával egyenlőségi feltétel nem a kulcson:
$E_{A4} = \mathrm{HT}_i + \left\lceil \frac{\mathrm{SC}(A,r)}{f_r} \right\rceil$

A5: Másodlagos index használatával:
$E_{A5} = \mathrm{HT}_i + \mathrm{SC}(A, r)$
Ha az A egyediséget biztosít, akkor $E_{A5} = \mathrm{HT}_i + 1$.

Szelekciós algoritmusok

Összehasonlítás-alapú szelekció

$\sigma_{A \leq v}(r)$ alakú lekérdezés becsült rekordszáma (c):

- Ha v értékét nem ismerjük: $\frac{n_r}{2}$
- Ha v ismert, és egyenletes az eloszlás: $n_r \cdot \left(\frac{v - \min(A,r)}{\max(A,r) - \min(A,r)} \right)$

A6: Elsődleges index használatával:

- $E_{A6} = \mathrm{HT}_i + \frac{b_r}{2}$
- Ha v ismert: $E_{A6} = \mathrm{HT}_i + \left\lceil \frac{c}{f_r} \right\rceil$

A7: Másodlagos index használatával:
$E_{A7} = \mathrm{HT}_i + \frac{\mathrm{LB}_i}{2} + \frac{n_r}{2}$

Join-algoritmusok

A nested loop join-algoritmus I.

for *minden* $t_r \in r$ *rekordra* **do**
  for *minden* $t_s \in s$ *rekordra* **do**
    if *a* (t_r, t_s) *pár kielégíti az illesztés* θ *feltételét* **then**
      a $t_r * t_s$ rekordot az eredményhez adjuk
    end
  end
end

Költsége:

- „worst-case": $b_r + n_r * b_s$
- ha s elfér a memóriában: $b_r + b_s$

Join-algoritmusok

A block nested loop join-algoritmus

for *minden* $b_r \in r$ *blokkra* **do**
  for *minden* $b_s \in s$ *blokkra* **do**
    for *minden* $t_r \in b_r$ *rekordra* **do**
      for *minden* $t_s \in b_s$ *rekordra* **do**
        if *a* (t_r, t_s) *pár kielégíti az illesztés* θ *feltételét* **then**
          a $t_r * t_s$ rekordot az eredményhez adjuk
        end
      end
    end
  end
end

- „worst-case" költsége: $b_r + b_r * b_s$
- ha s elfér a memóriában: $b_r + b_s$

Join-algoritmusok

A nested loop join-algoritmus család

for *minden* $t_r \in r$ *rekordra* **do**
  for *minden* $t_s \in s$ *rekordra* **do**
    if *a* (t_r, t_s) *pár kielégíti az illesztés* θ *feltételét* **then**
      a $t_r * t_s$ rekordot az eredményhez adjuk
    end
  end
end

Join-algoritmusok

A nested loop join-algoritmus család

```
for minden t_r ∈ r rekordra do
    for minden t_s ∈ s rekordra do
        if a (t_r, t_s) pár kielégíti az illesztés θ feltételét then
            a t_r ∗ t_s rekordot az eredményhez adjuk
        end
    end
end
```

```
for minden t_r ∈ r rekordra do
    Lineáris keresés t_r szerint: minden t_s ∈ s rekordra
end
```

Join-algoritmusok

A nested loop join-algoritmus család

for *minden* $t_r \in r$ *rekordra* **do**
  for *minden* $t_s \in s$ *rekordra* **do**
    if *a* (t_r, t_s) *pár kielégíti az illesztés* θ *feltételét* **then**
      a $t_r * t_s$ rekordot az eredményhez adjuk
    end
  end
end

for *minden* $t_r \in r$ *rekordra* **do**
  Lineáris keresés t_r szerint: minden $t_s \in s$ rekordra
end

A nested loop join-algoritmus család:

- indexelt nested loop: indexelt keresés s-ben
- hash join: hash-keresés s-ben

Join-algoritmusok

A merge join

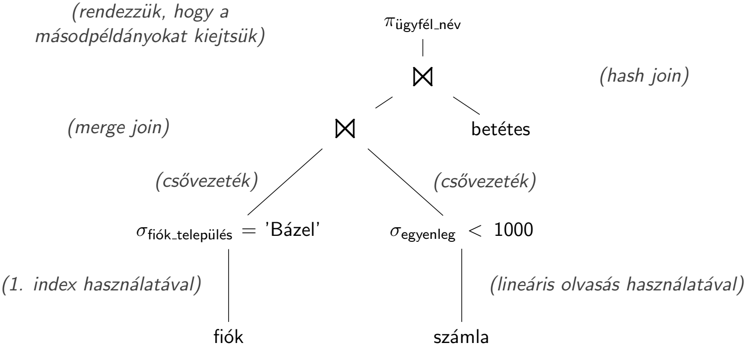
1. r és s rendezése a join attribútum szerint
2. a két reláció blokkjainak párhuzamos olvasása, találatok kiírása

Költsége: $b_r + b_s +$ a rendezés költsége

Kifejezéskiértékelés módjai

1. Materializáció
   - egyszerre egy művelet eredményének teljes kiszámítása
   - részeredmény tárolása („költség-hátrány”)
   - előnye: egyszerű implementálni
2. Pipelining
   - egymásra épülő műveletek szimultán kiértékelése
   - nem számítja ki előre a részeredményeket: igény- vagy termelőirányított
   - előnye: kiküszöböli a materializáció „költség-hátrányát”
   - hátránya: nem minden algoritmus ill. művelet támogatja

Végrehajtási terv



Költség-alapú optimalizálás

Végszó

Egyszerre jó és rossz: minden ekvivalens alak vizsgálata

- optimális terv
- túl sok munka

n reláció illesztése általános esetben:

- $\frac{(2 \cdot (n-1))!}{(n-1)!}$
- $n = 3$: 6; $n = 7$: 665 280; $n = 10$: több mint 17,6 milliárd

Megoldás:

- heurisztikus költség-alapú optimalizálás
- emberi optimalizálás:
  - a konkrét szemantika ismerete alapján
  - nagyobb szabadságfok a módszerek körében
  - szélsőséges helyzetekre jobban felkészíthető
  - statikus