Gagnasafnsfræði

Venslaalgebra og bestun fyrirspurna

Hallgrímur H. Gunnarsson

Inngangur

SQL:

SQL er *declarative* mál, segir bara **hvað** á að reikna, en ekki **hvernig**.

Það er undir gagnasafnskerfinu komið að framkvæma SQL fyrirspurnir á hagkvæman hátt.

Í dag munum við fjalla um hvernig það er gert.

Spurningar:

Hvernig breytum við SQL setningu í framkvæmanlegt "forrit" (query plan) ?

Á hvaða hátt má umrita slíkt forrit þ.a. það skili sömu niðurstöðu?

Á hvaða hátt má útfæra aðgerðirnar í forritinu?

Efni í dag

1. **Venslaalgebra**
2. **Þáttun fyrirspurna**
3. **Útfærsla venslavirkja**
4. **Framkvæmd fyrirspurna**
5. **Bestun fyrirspurna**

Venslaalgebra

SQL:

SQL er *declarative* mál, það segir hvað á að reikna, en ekki hvernig.

Venslaalgebra:

Venslaalgebra er fræðilegt fyrirspurnarmál byggt á venslalíkaninu.

Venslaalgebra er *procedural* fyrirspurnarmál.

Hægt að nota hana til að lýsa hvernig eigi að framkvæma fyrirspurn.

Málið byggist upp á svokölluðum venslavirkjum sem vinna með vensl

Hægt að tengja venslavirkja saman, úttak úr einum er inntak í næsta, o.s.frv.

Málfræði venslaalgebru

Segð í venslaalgebru er skilgreind endurkvæmt á eftirfarandi hátt

<expr> ::= <relation>
| <unary-op> <expr>
| <expr> <binop> <expr>

<unary-op> ::= σ | π | ρ

<binary-op> ::= $\cup$ | $\cap$ | $-$ | $\times$ | $\bowtie$

Virkjar

Val (e. selection): $\sigma_{\phi}(R)$

Velur hlutmengi staka (lína) úr venslum (töflum)

Dálkaval (e. projection): $\pi_{c}(R)$

Velur út einstaka dálka

Tenging (e. join): $A \bowtie_{c} B$

Sameinar tvö vensl með ákveðnum skilyrðum

Nafnabreyting (e. rename): $\rho_{a \to b}(R)$

Skiptir um nöfn á dálkum

Mengjavirkjar

Krossfeldi (e. cross product), $A \times B$:

Sameinar A og B með krossmargfeldi

Sammengi (e. set union), $A \cup B$:

Allar línur í A ásamt öllum línum í B

Mismunur mengja (e. set difference), $A - B$:

Allar línur í A en ekki í B

Sniðmengi (e. set intersection), $A \cap B$:

Allar línur í bæði A og B

Val (e. selection, filter)

Ritháttur:

$\sigma_{\phi}(R)$, þar sem ϕ er röksegð og R er segð í venslaalgebru

Merking:

Allar raðir í R sem uppfylla skilyrði ϕ

Dæmi:

$$\sigma_{A=\text{a1}} \left(\begin{array}{|c|c|c|} \hline \text{A} & \text{B} & \text{C} \\ \hline \text{a1} & \text{b1} & \text{c1} \\ \text{a2} & \text{b2} & \text{c2} \\ \text{a3} & \text{b3} & \text{c3} \\ \text{a1} & \text{b4} & \text{c4} \\ \hline \end{array} \right) = \left(\begin{array}{|c|c|c|} \hline \text{A} & \text{B} & \text{C} \\ \hline \text{a1} & \text{a2} & \text{c1} \\ \text{a1} & \text{b4} & \text{c4} \\ \hline \end{array} \right)$$

Dálkaval (e. projection)

Ritháttur:

$\pi_{a_1,a_2,..,a_n}(R)$, þar sem a_i er nafn á dálki

Merking:

Tilgreindir dálkar eru valdir út og öðrum dálkum er hent

(Skv. venslalíkaninu ætti einnig að henda út tvítekningum)

Dæmi:

$$\pi_{A,B}\left(\begin{array}{|c|c|c|}\hline \text{A} & \text{B} & \text{C} \\ \hline \text{a1} & \text{b1} & \text{c1} \\ \text{a2} & \text{b2} & \text{c2} \\ \hline \end{array}\right) = \left(\begin{array}{|c|c|}\hline \text{A} & \text{B} \\ \hline \text{a1} & \text{a2} \\ \text{b1} & \text{b2} \\ \hline \end{array}\right)$$

Krossfeldi (e. cross product)

Ritháttur:

$R \times S$, þar sem R og S eru segðir í venslaalgebru

Merking:

Krossmargfeldi R og S, þ.e. allar raðir í R tengdar við allar raðir í S

Dæmi:

$$\left(\begin{array}{|c|c|}\hline \text{A} & \text{B} \\ \hline \text{a1} & \text{b1} \\ \text{a2} & \text{b2} \\ \hline \end{array}\right) \times \left(\begin{array}{|c|}\hline \text{C} \\ \hline \text{c1} \\ \text{c2} \\ \hline \end{array}\right) = \left(\begin{array}{|c|c|c|}\hline \text{A} & \text{B} & \text{C} \\ \hline \text{a1} & \text{b1} & \text{c1} \\ \text{a2} & \text{b2} & \text{c1} \\ \text{a1} & \text{b1} & \text{c2} \\ \text{a2} & \text{b2} & \text{c2} \\ \hline \end{array}\right)$$

Tenging (e. join)

Ritháttur:

$R \bowtie_c S$, þar sem c er röksegð, R og S eru segðir í venslaalgebru

Merking:

Allar raðir í R og S sem uppfylltu tengiskilyrðið c, sameinaðar í nýjum venslum

Hægt að skilgreina $R \bowtie_c S$ sem $\sigma_c(R \times S)$

Dæmi:

$$\left(\begin{array}{|c|c|}\hline \text{A} & \text{B} \\ \hline \text{a1} & \text{b1} \\ \text{a2} & \text{b2} \\ \text{a3} & \text{b3} \\ \hline \end{array}\right) \bowtie_{A=C} \left(\begin{array}{|c|c|}\hline \text{C} & \text{D} \\ \hline \text{a1} & \text{d1} \\ \text{a1} & \text{d2} \\ \text{a5} & \text{d3} \\ \hline \end{array}\right) = \left(\begin{array}{|c|c|c|c|}\hline \text{A} & \text{B} & \text{C} & \text{D} \\ \hline \text{a1} & \text{b1} & \text{a1} & \text{d1} \\ \text{a1} & \text{b1} & \text{a1} & \text{d2} \\ \hline \end{array}\right)$$

Nafnabreyting (e. rename)

Riháttur:

$\rho_s(R)$, þar sem s er mengi af nafnabreytingum, t.d. $\{\ a \to x,\ b \to y\ \}$

Merking:

Ef $s = \{\ a \to x\ \}$ þá fær dálkur sem heitir a í inntakinu nafnið x í úttakinu

Dæmi:

$$\rho_{\{\ A \to X\ \}} \left(\begin{array}{|c|c|c|} \hline \text{A} & \text{B} & \text{C} \\ \hline \text{a1} & \text{b1} & \text{c1} \\ \text{a2} & \text{b2} & \text{c2} \\ \hline \end{array} \right) = \left(\begin{array}{|c|c|c|} \hline \text{X} & \text{B} & \text{C} \\ \hline \text{a1} & \text{b1} & \text{c1} \\ \text{a2} & \text{b2} & \text{c2} \\ \hline \end{array} \right)$$

Eiginleikar valvirkjans

Val er idempotent: (hefur engin áhrif að framkvæma sama val oft)

$$\sigma_{\phi}(R) = \sigma_{\phi}\sigma_{\phi}(R)$$

Val er víxlið: (röðin á vali skiptir ekki máli)

$$\sigma_{A}\sigma_{B}(R) = \sigma_{B}\sigma_{A}(R)$$

Brjóta upp val með AND skilyrði:

$$\sigma_{A \wedge B} = \sigma_{A}(\sigma_{B}(R)) = \sigma_{B}(\sigma_{A}(R))$$

Brjóta upp val með OR skilyrði:

$$\sigma_{A \vee B} = \sigma_{A}(R) \cup \sigma_{B}(R)$$

Eiginleikar tengivirkjans

Tenging er tengin (e. associative):

$$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$$

Selection push-down:

Ef ϕ vísar bara í dálka í S þá gildir:

$$\sigma_{\phi}(R \bowtie S) = R \bowtie \sigma_{\phi}(S)$$

Dæmi um notkun á reglum

Fyrirspurn:

$$\sigma_{R.a=100 \wedge S.b=200 \wedge R.x=S.y}(R \bowtie S)$$

Brjótum valið upp:

$$\sigma_{R.a=100}(\sigma_{S.b=200}(\sigma_{R.x=S.y}(R \bowtie S)))$$

Endurröðum:

$$\sigma_{R.x=S.y}(\sigma_{S.b=200}(\sigma_{R.a=100}(R \bowtie S)))$$

Notum selection push-down regluna:

$$\sigma_{R.x=S.y}((\sigma_{R.a=100}(R) \bowtie \sigma_{S.b=200}(S))$$

Þáttun fyrirspurna

Þáttun:

SQL fyrirspurn er þýdd yfir í jafngilda fyrirspurn í venslaalgebru (fáum segðatré með venslavirkjum) Segð í venslaalgebru lýsir því hvernig fyrirspurnin er framkvæmd (ólíkt SQL). Hægt að vinna með tréið og breyta segðinni skv. reglum til að breyta því hvernig niðurstaðan er reiknuð.

Logical vs. physical:

Einstakir hnútar í segðatrénu eru venslavirkjar. Ein tegund af venslavirkja, t.d. tenging (e. join), getur haft margar útfærslur sem hentar við mismunandi aðstæður.

Venslavirki af tiltekinni tegund er kallaður *logical operator*, en þegar það er búið að velja útfærslu þá tölum við um *physical operator*.

Útfærsla venslavirkja

Mismunandi aðstæður:

Til margar mismunandi útfærslur á hverjum venslavirkja. Allar útfærslur uppfylla sömu lýsingu, þ.e. hafa sömu eiginleika, en ganga til verka á ólíkan hátt.

Veljum útfærslu eftir aðstæðum, t.d. gæti ein útfærsla gert ráð fyrir að inntakið sé raðað en önnur ekki.

Útfærsla á tengingu (e. join): nested loops, hash join og merge join

Útfærsla á samantekt (e. aggregation): hashing og sorting

Útfærsla á lestri (e. scan): index scan, sequential scan, full index scan

Ritháttur

Fjöldi síðna (e. number of pages)

Fjöldi síðna í R er ritað R_p

Fjöldi raða (e. number of rows)

Fjöldi raða í R er ritað R_n

Tenging með nested loop join

Útfærsla:

```
function nljoin(R,S,c):
    foreach tuple r in R:
        foreach tuple s in S:
            if c(r,s):
                emit <r,s>
```

I/O kostnaður:

Lesum allar síður í ytri töflunni og fyrir hverja röð þar lesum við alla innri töfluna

Kostnaðurinn er því: $O(R_p + R_n S_p)$

Tenging með block nested loop join

Hugmynd:

Byggir á sömu hugmynd og nested loop join nema við nýtum okkur vinnsluminni til að vinna með margar síður úr ytri töflunni samtímis. Þá þarf ekki að lesa innri töfluna fyrir hverja röð í ytri töflunni, heldur bara einu sinni per sett af síðum úr ytri töflunni sem rúmast samtímis í vinnsluminni.

I/O kostnaður:

Látum M standa fyrir fjölda síðna af vinnsluminni sem má nota (buffer pages)

Þá er kostnaðurinn: $O(R_p + \frac{R_p}{M} S_p)$

Tenging með index nested loop join

Hugmynd:

Byggir á nested loop join en ef önnur taflan er með flýtivísi sem passar við tengiskilyrðið þá notum við hana sem innri töfluna og flettum upp í henni með flýtivísinum.

I/O kostnaður:

$O(R_p + R_n C(S))$, þar sem $C(S)$ er kostnaðurinn við að fletta upp lykli í S með flýtivísinum.

Það kostar um 1.2 I/O að fletta upp í hakkavísi, en 2-4 I/O að fletta upp í B+ trévísi.

Kostnaður við að sækja gagnasíðurnar veltur á klösun vísisins. Ef hann er alveg óklasaður (versta tilfelli) þá er það 1 I/O per röð.

Tætitenging (e. hash join)

Hugmynd:

Lesum smærri töfluna og byggjum tætitöflu í minni út frá tengilyklinum. Lesum síðan stærri töfluna og leitum í tætitöflunni eftir röð sem passar.

(Virkar bara þegar tengiskilyrðið er $=$, virkar ekki fyrir $>$, $>=$, $<$, $<=$, etc.)

I/O kostnaður:

Ef það er pláss fyrir alla tætitöfluna í minni þá er I/O kostnaðurinn bara lestur á báðum töflunum, þ.e. $O(R_p + S_p)$

Tenging með samröðun (e. sort-merge join)

Hugmynd:

Notum sameiningu (e. merge) á tvær raðaðar töflur til að finna þær raðir sem passa saman. (Virkar mjög svipað og venjulegt merge sort.)

Ef töflurnar eru ekki í röð samkvæmt tengilyklinum þá þarf að raða þeim fyrst.

Þessi aðferð hentar því best þegar inntakstöflurnar eru þegar í röð, t.d. þegar fyrri venslavirki raðaði þeim eða hægt að sækja töfluna í röð (B+ trévísir á tengilyklinum).

Samröðun virkar fyrir fleiri samanburðarvirkja en tætitenging, þ.e. virkar fyrir $<$, $\leq$, $=$, $>$, $\geq$

Tenging með samröðun, framhald

I/O kostnaður:

Ef R og S eru í röð þá þarf bara að sameina (e. merge) og kostnaðurinn er $O(R_p + S_p)$

Ef R og S eru ekki í röð þá þarf fyrst að raða þeim. Þá veltur kostnaðurinn á því hvort það sé nægilegt minni til að raða töflunum í einni umferð í minni eða hvort það þurfi margar umferðir og umsýslu á disk.

Ytri röðun (external n-way merge sort):

Fjöldi umferða: $1 + \lceil \log_{B-1} \lceil N/B \rceil \rceil$, þar sem N er fjöldi síða í skrá og B er fjöldi síða í minni sem má nota (buffer pages)

Kostnaður: $2N *$ fjöldi umferða

Aðgangsleiðir (e. access methods)

Heap scan: (sequential scan)

Runubundinn (sequential) lestur yfir alla töfluna

Index scan:

Lestur með flýtivísi, minna af gögnum, en vanalega random I/O

Full index scan:

Runubundinn (sequential) lestur yfir allan flýtivísinn

Útfærsla:

Vanalega til sérstakir venslavirkjar fyrir lestur (scan), þ.e. heapscan venslavirki og index scan venslavirki.

Framkvæmd fyrirspurna

Áætlun: (e. query plan)

Áætlun um framkvæmd (query plan) er táknuð sem tré þar sem hver hnútur er venslavirki.

Framkvæmd:

Tréið er framkvæmt "bottom-up", byrjum í laufunum og vinnum okkur upp. Rótarhnúturinn er síðasta aðgerðin. Laufin eru scan/lookup hnútar (heap scan, index scan).

Algeng útfærsla:

Allir venslavirkjar útfæra s.k. iterator interface. Hægt að biðja iterator um næsta gildi og athuga hvort hann sé kominn út á enda. Gögn eru "dregin" upp tréið, þegar rótin er beðin um gildi þá er reiknað niður tréið.

Bestun fyrirspurna

Bestunarferli:

Bestun er framkvæmd af svokölluðum query planner (eða query optimizer)

Til mörg tré (áætlanir/plön) sem gefa sömu niðurstöðu. Markmiðið er að finna hagkvæmasta tréið.

Notum tvær aðferðir:

1. Notum algebrureglur til að endurraða virkjum án þess að breyta niðurstöðunni

2. Reynum að velja hentugt reiknirit fyrir hvern venslavirkja

Gögn:

Bestun byggir oft á tölfræðilegum upplýsingum um töflurnar, t.d. dreifingu á gildum, o.s.frv. Getur hjálpað við að velja rétt reiknirit.

Bestun frh.

Kostnaður

Kostnaður hnúts (framkvæmd venslavirkjans í þeim hnúti) er fall af inntaki hans, mikilvægt að raða hnútum (aðgerðum) rétt

Minnka snemma:

Almennt gott að sigta raðir snemma og minnka stærðina á innri venslum milli hnúta (intermediate result sets)

Bókhald í trénu:

Gagnlegt að láta upplýsingar eins og röð gagna (sort order) fylgja með upp tréið. Oft hægt að velja hagkvæma venslavirkja ef við vitum að inntakið er raðað.

Kerfislistar (e. system catalog)

Kerfislisti:

Gagnasafnskerfi geymir ýmsar upplýsingar um gagnasöfn í svokölluðum kerfislistum.

Geymir allar upplýsingar um töflur, hvaða dálkar eru til staðar, hvaða flýtivísar eru til staðar, allar sýndartöflur, tölfræði um gögnin í töflunum, o.s.frv.

Bestun:

Bestunarferlið styðst við kerfislistann, nauðsynlegt að vita hvaða vísar eru til staðar, stærð á töflum, tölfræði um dreifingu o.s.frv.

Hvað þarf að ákveða í bestun?

Aðgangsleið:

Veljum aðgangsleið (e. access path) fyrir hverja töflu. Veljum þá aðgangsleið sem hentar best, þ.e. sem lágmarkar I/O aðgerðir.

Reiknirit fyrir venslavirkja:

Veljum reiknirit fyrir hvern venslavirkja í trénu. Veltur á stöðu gagnasafnsins (fengið úr kerfislistanum) og inntaki úr öðrum hnútum.

Röð venslavirkja:

Ákveða uppbyggingu trésins og röð aðgerða, hvað kemur á undan hverju.