Optimizacija poizvedb

Iztok Savnik, FAMNIT

Prosojnice & učbenik

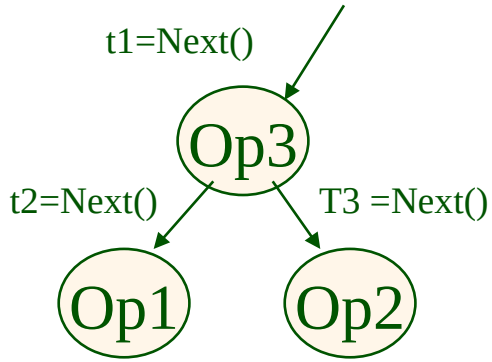
- Učbenik:
  - Raghu Ramakrishnan, Johannes Gehrke, *Database Management Systems, McGraw-Hill, 3rd ed., 2007.*
- *Prosojnice:*
  - *From „Cow Book“: R.Ramakrishnan, http://pages.cs.wisc.edu/~dbbook/*

Osnove optimizacije poizvedb

- Potek
  - Plan izvajanja poizvedbe.
  - Prevajanje SQL v RA
  - Ekvivalenca operacij RA
  - Ocena plana poizvedbe
  - Definicija problema
  - Prostor rešitev
  - Tipični relacijski optimizator
  - Naštevanje alternativnih planov
  - System R

Program v SUPB

- *Kaj je program v SUPB?*
  - Drevo operacij relacijske algebre.
  - Vozlišča so označena z algoritmi za izvajanje posameznih operacij.

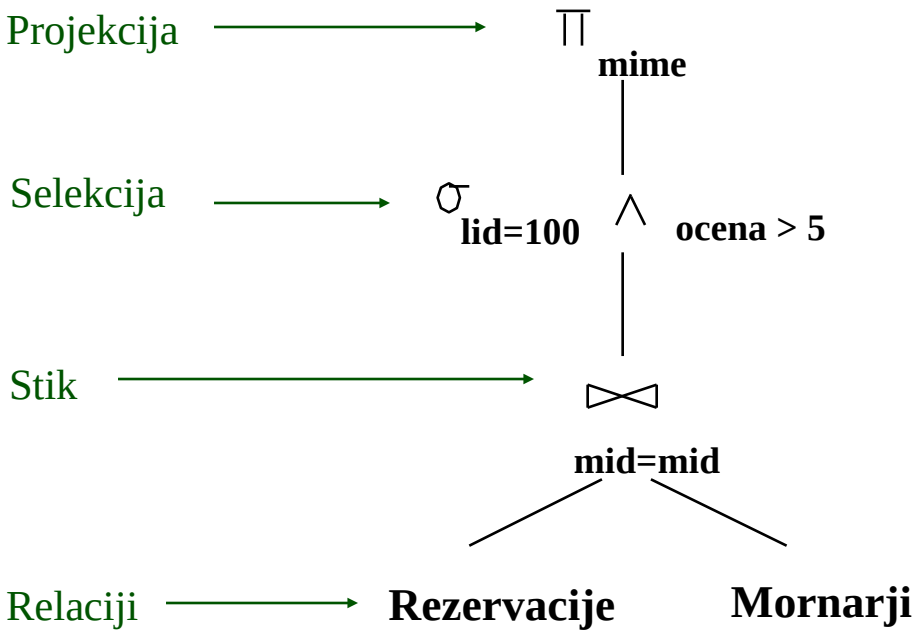


- Drevo iteratorjev
  - Vsaka operacija je implementirana z iteratorjem.
  - Vmesnik iteratorja: open(), next(), close()
  - Klic operacije next() sproži evaluacijo operacije next() na otrocih operacije.
  - Na rezultatih otrok operacija izvaja svojo kodo.
  - Rezultate pošilja staršem (zapis po zapis ali blok zapisov)

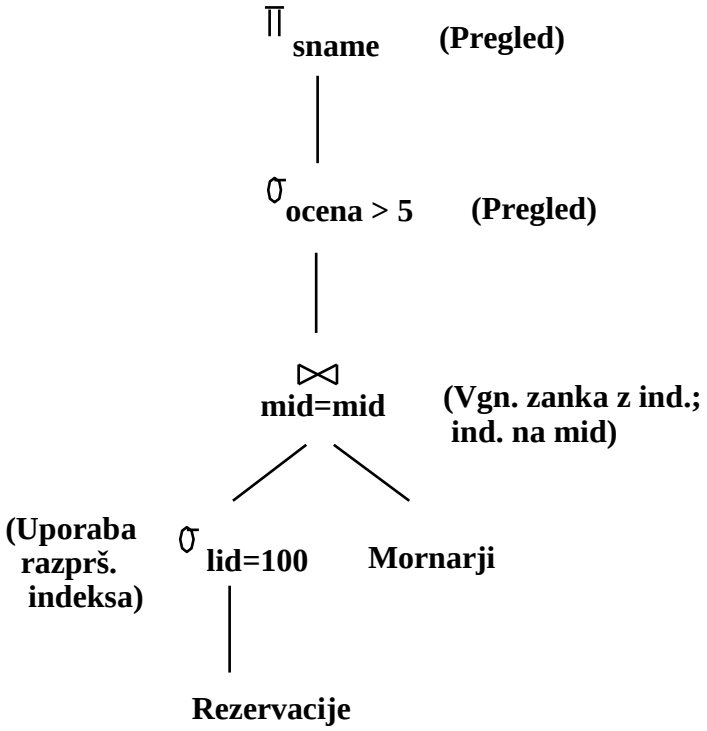
Program v SUPB

- Dobimo drevesno strukturiran cevovod po katerem se pretakajo n-terice.
  - Na nekaterih mestih pretok začasno ustavi zaradi materializacije tabel.
  - Diskovne strani beremo v listih plana poizvedbe.
  - N-terice konstruirane v operaciji RA se pošlje nadrejenim operacija.

Drevo operacij RA (brez izbranih algoritmov)



Drevo operacij RA z izbranimi algoritmi



Optimizacija drevesa operacij

- Kako optimiziramo drevo RA operacij?
  - Logična in fizična optimizacija. Nivoji se zelo prepletajo!
  - Implementacija rel. optimizatorja
  - Evaluator dreves RA

❖ Logična optimizacija

- Iščemo izraz relacijske algebre, ki se najhitreje izvede.
- Da bi ocenili izraz relacijske algebre je potrebno določiti fizično implementacijo drevesa RA.

❖ Fizična optimizacija.

- Uporabimo fizično algebro relacij, ki vsebuje samo dve operaciji: (1) dostopne poti in (2) stike.
- Iščemo optimalne algoritme za izvajanje fizičnih operacij RA.
- Fizična optimizacija se prepleta z logično optimizacijo.

Prevajanje SQL v relacijsko algebro

- Dekompozicija poizvedbe v bloke
  - Celoten SQL stavek se razdeli v bloke
  - Zaključeni SQL bloki se obravnavajo kot procedure
  - Posledica: vgnezdeni bloki se kličejo ob vsaki iteraciji v nadrejenem bloku
  - Med bloki ni optimizacije
- Blok vprašanja prevedemo v drevo operacij RA
  - Vsak blok posebaj se prevede v izraz RA
  - Vsak blok se optimizira ločeno od drugih blokov
  - Vgnezdeni blok se izvaja kot podprogram;
  - Poizvedbe s stiki imajo več možnosti za korektno optimizacijo.

Ekvivalence operacij RA

- Zapis izraza RA določa drevo operacij RA!
- Optimizacija izrazov RA
  - Naštevanje ekvivalentnih zapisov izrazov RA
  - Priredi vse možne implementacije operacijam
  - Izračunaj oceno vseh planov poizvedbe in izberi najboljšega
- Naštevanje ekvivalentnih izrazov RA
  - Logična algebra: pravila bodo predstavljena kmalu
  - Fizična algebra: preurejanje stikov, potiskanje Π in σ navzdol
- Algebrske lastnosti operacij RA omogočajo optimizacijo

Ekvivalence operacij RA

- *Selekcija:* $\delta_{c1 \wedge .. \wedge cn}(R) = \delta_{c1}(\ldots(\delta_{cn}(R)\ldots)$ (*Razcep*)

  $\delta_{c1}(\delta_{c2}(R)) = \delta_{c2}(\delta_{c1}(R))$ (*Komutativnost*)

- *Projekcija:* $\Pi_{a1}(R) = \Pi_{a1}(\ldots(\Pi_{an}(R)\ldots)$ *(Razcep)*

- *Stik:* $R \bowtie (S \bowtie T)$ $(R \bowtie S) \bowtie T$ *(Asociativnost)*

  $(R \bowtie S)$ $(S \bowtie R)$ (*Komutativnost*)

- ☞ Dokaži: $R \bowtie (S \bowtie T)$ $(T \bowtie R) \bowtie S$

Ekvivalence operacij RA

- Projekcija je komutativna s selekcijo, ki uporablja atribute projekcije
- Selekcija, ki vsebuje primerjavo atributov obeh argumentov kartezijskega produkta se prevede v stik
- Komutativnost Π in σ s ⋈; potiskanje selekcije/projekcije proti listom.

$\delta(R \bowtie S) \equiv \delta(R) \bowtie S$, če selekcija izbira samo atribute R.

$\Pi(R \bowtie S) \equiv \Pi(R) \bowtie S$, če projekcija ohrani atribute stika.

Ocena planov poizvedbe

- Rezultat je vedno ocena.
- Statistika je shranjena v sistemskih katalogih.
- Statistika je uporabljena za oceno
  - Čas evaluacije poizvedbe
  - Velikosti vmesnih rezultatov
- Ocentiti moramo vsako operacijo v drevesu poizvedbe.
  - Sekvenčni pregled, indeksni pregled, stiki, itd.
- Vzami v poštev CPU in I/O
  - Mi uporabimo # blokov branja/pisanja
  - Vedno več procesiranja v spominu.

Statistike in katalogi

- Podatki o relacijah in indeksih v podatkovni bazi.
- Katalogi običajno vsebujejo:
  - # n-teric za vsako relacijo.
  - # strani za vsako relacijo.
  - # št. različnih vrednosti za ključ indeksa.
  - # št. strani za vsak indeks.
  - višina indeksa, najnižja/najvišja vrednost ključa za vsak drevesni indeks.
- Katalogi se ažurirajo periodično.
  - Sprotno ažuriranje je preveč drago.
  - Ker je veliko pribljižkov je majhna nekonsistenca OK.
- Včasih so shranjene bolj podrobne vrednosti.
  - Npr. število posamezni vrednosti za atribut relacije.

Ocene selektivnosti pogojev operacij RA

- *Predpostavke:*
  - Vrednosti atributov so enakomerno porazdeljene.
  - Pogoji so med seboj neodvisni.
- Selekcija:
  - Pogoj A=value ima SP = 1 / Nkeys(A)
  - Pogoj $A_1=A_2$ ima SP = 1 / MAX(Nkeys(A_1),Nkeys(A_2))
  - Pogoj A>value ima SP = (High(A)-value)/(High(I)-Low(I))
  - $SP(p(A_i) \wedge p(A_j)) = SP(p(A_i))*SF(p(A_j))$
  - $SP(p(A_i) \vee p(A_j)) = SP(p(A_i)) + SP(p(A_j)) – (SP(p(A_i)) * SP(p(A_j)))$
  - $SP(A \in \{value\}) = SP(A= value) * card(\{values\})$
- Projekcija:
  - Selektivnost projekcije je SP = 1.

Ocene selektivnosti pogojev operacij RA

- Stik:
  - $R \bowtie_{A1=A2} S$, $A_1 \in R$ *in* $A_2 \in S$.
  - Stik je definiran na osnovi pogoja $A_1=A_2$.
  - Pogoj $A_1=A_2$ ima SP = $1 / MAX(Nkeys(A_1),Nkeys(A_2))$.

Ocene velikosti rezultatov operacij RA

- Selekcija
  - $card(\sigma_F(R)) = SP(F) \times card(R)$
- Projekcija
  - $card(\Pi_A(R))=card(R)$
- Kartezijski produkt
  - $card(R\ x\ S) = card(R) * card(S)$
- Stik
  - A je ključ R in B je tuj ključ v S: $card(R \bowtie_{A=B} S) = card(S)$
  - Splošno: $card(R \bowtie S) = SP * card(R) \times card(S)$

Ocene velikosti rezultatov operacij RA

- Unija
  - Zgornja meja: $card(R \cup S) = card(R) + card(S)$
  - Spodnja meja: $card(R \cup S) = max\{card(R), card(S)\}$
- Razlika
  - Zgornja meja: $card(R–S) = card(R)$
  - Spodnja meja: 0

Cena dostopne poti

- Dostopna pot do tabele ima vhodne podatke:
  - Logični izraz vsebuje atomične izraze, ki so uporabljeni kot parametri dostopne poti
  - Indeksi definirani na vhodni tabeli
  - Naštej vse možne dostopne poti!
- Cena dostopne poti je odvisna od
  - Selekt. atomarih pogojev uporabljenih za dostopne poti
  - Obstoječih indeksov
- Cena dostopne poti: cena branja dela tabele iz diska z uporabo dostopne poti
  - Najboljša dostopna pot prebere najmanj strani iz diska.

Ocena za plane z eno relacijo

- Uporaba indeksa I na primarnem ključu:
  - *Visina(I)[+1] za B+ drevo, cca. 1.2[+1] za razpršilni indeks.*
- Povezan indeks I, ki se ujema z enim ali več pogoji:
  - *(NStrani(I)+NStrani(R)) * produkt Si za pogoje selekcije.*
- Nepovezan indeks, ki se ujema z enim ali več pogoji:
  - *(NStrani(I)+NZapisov(R)) * produkt Si za pogoje selekcije.*
- Sekvenčni pregled tabele:
  - *NStrani(R)*

Ocene za planov nad večimi relacijami

```
SELECT seznam-izbire
FROM seznam-relacij
WHERE  pogoj1 AND...AND pogojk
```

- Blok poizvedbe:
- Maksimalno # n-teric v rezultatu je produkt kardinalnosti vseh vhodnih relacij iz FROM stavka.
- *Selektivnost* povezana z vsakim pogojem odseva vpliv pogoja na velikost rezultata.
  - *Kardinalnost rezultata* = Max # n-teric * produkt vseh SP.
- Multi-relacijski plani se gradijo pri stikih z dodajanjem ene same nove relacije naenkrat.
  - Cena metode stika + oceno kardinalnosti stika.

Shema za zgled

Mornarji (*mid*: integer, *mime*: string, *ocena*: integer, *star*: real)
Rezervacije (*mid*: integer, *lid*: integer, *dan*: dates, *rime*: string)

- Podobno kot stara shema; *rime* dodana.
- Rezervacije:
  - Velikost zapisa = 40 zlogov, 100 zapisov na stran, 1000 strani.
- Mornarji:
  - Velikost zapisa = 50 zlogov, 80 zapisov na stran, 500 strani.

Primer 1: Cena dostopne poti

- Razpršilni indeks H na <*mid*> za Mornarji
- Velikost indeksa: 200 strani
  - Velikost pod. vpisa = //ptr+int// 8+8 = 16B
  - # vpisov na stran = 4000/16 = 250
  - # strani pod.vpisov: 40000/250 = 160 strani
  - 80% zapolnjen: 160*1.25 = 200 strani
- Katalog: #ključ(H), #strani(Mornarji)
- Selektivnost: 1/#ključ(H) = 1/40000
- Ocena:
  - Povezan: (200+500) * 1/40000 = 1 stran
  - Nepovezan: (200+40000) * 1/40000 = 2 strani

Primer 2: Cena dostopne poti

- Selektivnost področja
- Pogoj: dan > 12/12/09 na Rezervacije
- B+ drevo T na atributu <*dan*>
  - 1000*100*(8+8)/4000 = 400 strani
  - 67% zapolnjen = 1.5*400 = 600 strani //izpustimo ind.strani//
- Selektivnost DP: (High(T)–vrednost)/(High(T)-Low(T))
  - Predpostavka: začetek rezervacij je 1/1/2000
  - 2018–2009 / 2018–2000 = 9 / 18 = ½
- Cena:
  - Povezan: (600+1000) *1/2 = 800 strani
  - Nepovezan: (600+100000)*1/2 = 50300 strani

Optimizacija poizvedb - definicije

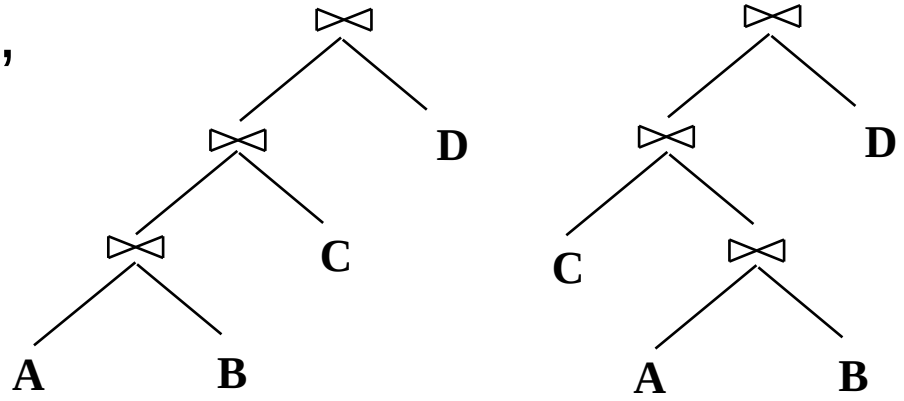
- Dana je poizvedba *q* nad podatkovno bazo *B*.
- Poišči najcenejši plan poizvedbe.
  - Optimizacijski algoritem
    - Sistem R uporablja dinamično programiranje
  - Imamo *ocenitveno funkcijo*, ki izračuna *ceno plana*.
  - Za vsako operacijo se *ocenijo vsi algoritmi* za izvajanje operacije, ki jih je možno uporabiti v danem okolju; izbere se najboljši algoritem.
- Algoritmi za optimizacijo.
  - Izčrpno preiskovanje.
  - System R uporabi dinamično programiranje.
  - Naključno preiskovanje, genetski algoritmi, itd.

Optimizacija poizvedb - definicije

- Ocenitvena funkcija upošteva:
  - Število prebranih blokov iz diska.
  - CPU uporabljen za izračun operacij.
  - Shranjevanje začasnih tabel.
  - Količina zaseženega RAM (zadnjih 20 let).
- Mi upoštevamo samo branje/pisanje disk. strani!

Poenostavitev problema

- Fizična optimizacija
- Vozlišče z metodo dostopa
  - Pregled zapisov
  - DP z indeksom, DP s sortiranjem, ...
  - Vsebuje (tudi) selekcijo in projekcijo
- Vozlišče s stikom
  - Različni algoritmi za izvajanje stika
  - Stik z vgnezdeno zanko, indeksom, zlivanjem, ...
  - Vsebuje (tudi) selekcijo in projekcijo



Prostor rešitev

- Prostor ekvivalentnih poizvedb.
- Lastnosti relacijske algebre omogočajo transformacije poizvedbe.
  - Transformirana poizvedba vrne isti rezultat.
  - Transformirana poizvedba omogoča spremembo plana.
  - Ponovno je potrebno določiti algoritme za impl. operacij.
  - Iščemo izraz, ki se najhitreje izvede in porabi najmanj prostora.
- Praksa:
  - Izogibamo se kartezijskim produktom.
  - Izogibajmo se najslabšim rešitvam.

Tipični relacijski optimizator

- Algoritem osnovan na dinamičnem programiranju
  - Optmalne plane gradimo iz optimalnih rešitev podproblemov: od spodaj navzgor
  - Optimalna rešitev problema z N stiki
    - Dodaj optimalno en stik k optimalnem planu za N-1 stikov
  - Omejimo prostor rešitev, uporabljamo samo levo-usmerjene plane, nimamo materializacije
  - Izčrpno preiskovanje bi naštelo vse permutacije
  - Dinamično programiranje je še vedno eksponentno
- Optimizator Sistema R

Naštevanje alternativnih planov

- Dva osnovna primera
  - Plan nad eno relacijo
  - Plan nad večimi relacijami

Poizvedbe nad eno relacijo

- Kombinacije selekcije, projekcije in agregacijskih operacij
  - Ni stikov!
- Pregledajo se vse možne dostopne poti
  - Dostopne poti brez indeksov
    - Pregled datoteke, pregled sortirane datoteke
  - Dostopne poti z indeksi
    - Indeksni dostop, samo index, več indeksov, sortiran indeks
  - Selekcija in projekcija sta integrirani
  - Rezultat (sortiran) se poveže z agregacijo
    - Bodisi obstoječa urejenost (zadnj.rezultat) ali sortiranje
- Izberemo metodo z najnižjo ceno

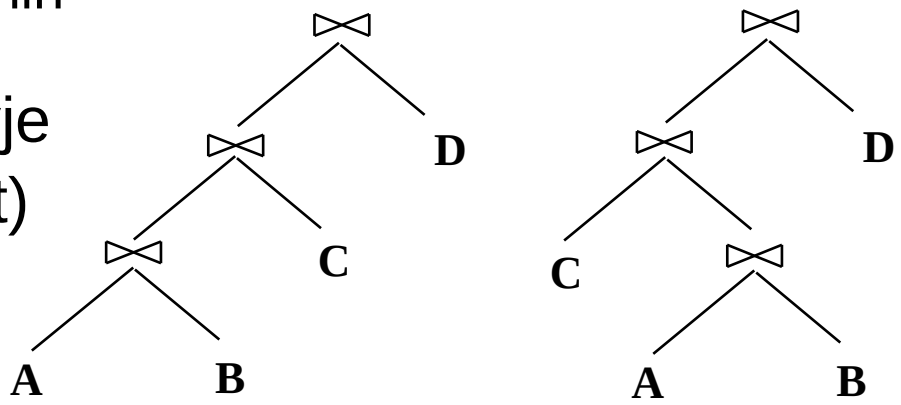
Primer 3

```
SELECT M.mid
FROM Mornarji M
WHERE M.ocena=8
```

- Imamo drev. indeks I na atr. *ocena*:
  - (1/#kljucev(I)) * #zapisov(R) = (1/10) * 40000 = 4000 zapisov.
  - Povezan indeks:
    Velikost indeksa: 12 //4+8// * 40000 / 4000 //velikost stani// = 120 *1.5 = 180
    Predpostavka: indeks je 67% zapolnjen
    (1/#ključev(I)) * (#strani(I)+#strani(R)) = (1/10) * (180+500) = 68 strani.
  - Nepovezan indeks:
    (1/#ključev(I)) * (#strani(I)+ #zapisov(R)) = (1/10) * (180+40000) = 4018 strani.
- Imamo indeks na *mid*:
  - Prebrati je potrebno vse n-terice/strani.
  - Razpršilni indeks = 40000*(8 + 8)/4000 = 160 strani * 1.25 = 200 strani
  - Povezan indeks = 200+500 strani;
    Nepovezan indeks = 200+40000 strani. Slabo !
- Sekvenčni pregled:
  - Vse strani tabele = 500 strani.

Poizvedbe nad večimi relacijami

- Prostor je prevelik in se ga pogosto omeji
  - Uporaba hevristike ali omejitev strukture dreves.
  - Tudi izčrpno preiskovanje se uporablja... (<10 stikov)
- Kateri del prostora pregledamo?
  - Odvisno od algoritma za preiskovanje.
  - System R: uporablja le levo-usmerjene plane
    - Levo-usmerjeni plani omogočajo izvedbo *cevovoda.*
    - Ni potrebno kreirati vmesnih začasnih tabel.
  - Zig-zag drevesa, grmičevje (tudi dovolijo vzporednost)



Naštevanje levo-usmerjenih planov

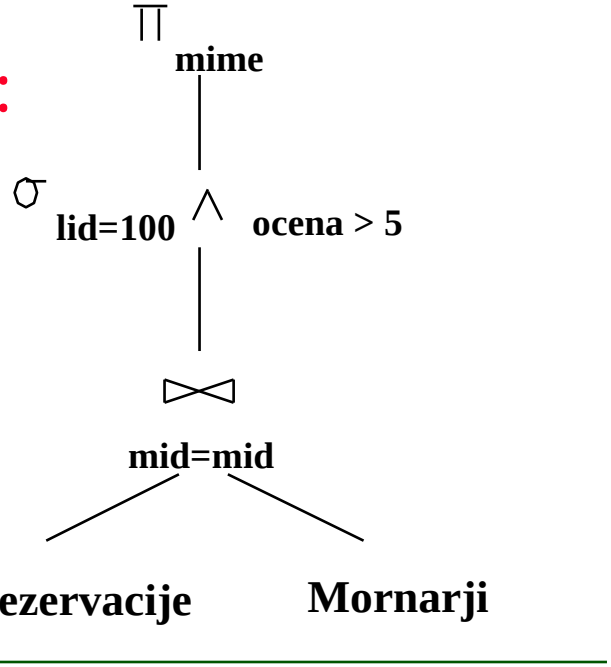
- Levo-usmerjeni plani se razlikujejo samo vrstnem redu relacij, metodi dostop za vsako relacijo in izvedbo stika za vsak stik
- Naštevanje z N prehodi (za N relacij):
  - Prehod 1: Poišči najboljši plan za vsako relacijo
  - Prehod 2: Poišči najboljšo izvedbo stika vseh 1-relacijskih planov z drugo relacijo (Vsi 2-relacijski plani)
  - Prehod N: Poišči najboljšo izvedbo stika rezultata plana nad N-1 relacijami z N-to relacijo (Vsi N-relacijski plani)
- Za vsako podmnožico sestavljeno iz *K* relacij je potrebno ohraniti samo
  - Najcenejši plan in
  - Najcenejši plan za vsako zanimivo urejenost n-teric.

Naštevanje levo-usmerjenih planov

- ORDER BY, GROUP BY, agreg. funkcije itd. izvajamo kot zadnjio fazo, bodisi z uporabo plana z `zanimivo urejenostjo' ali z dodatnim urejanjem.
- Plan za N-1 relacij se ne kombinira z dodatno relacijo, če ni pogoja stika med izrazi WHERE stavka, razen v primeru, da so bili izčrpani vsi predikati (iz WHERE stavka).
  - Izogni se Kartezijskemu produktu, če je to mogoče.
- Navkljub omejenem prostoru rešitev je predstavljen pristop še vedno eksponenten v številu tabel.

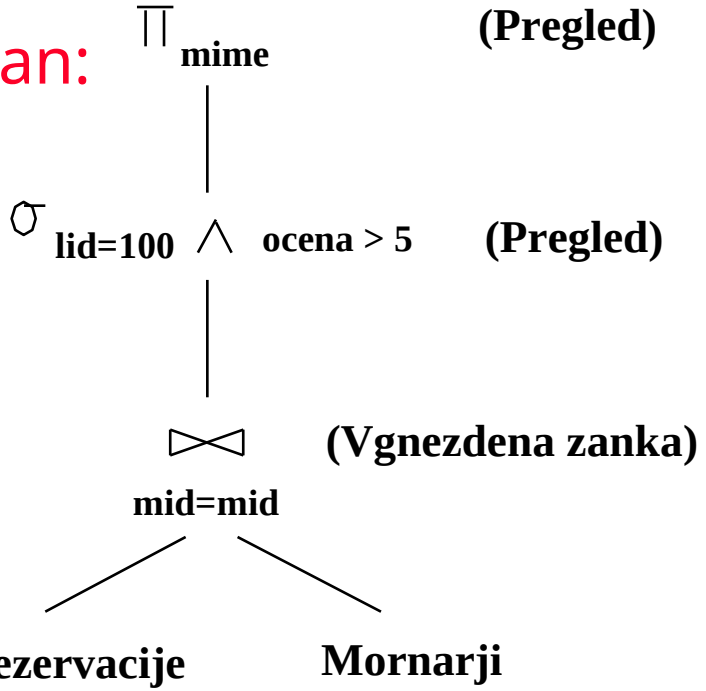
Primer 4

RA Drevo:



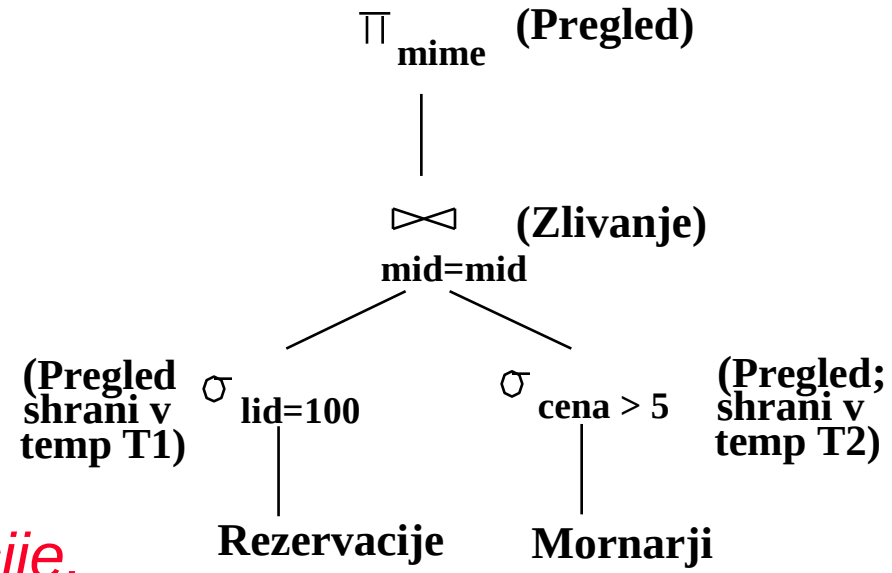

```
SELECT  M.mime
FROM  Rezervacije R, Mornarji M
WHERE  R.mid=S.mid AND
    R.lid=100 AND M.ocena>5
```

Plan:



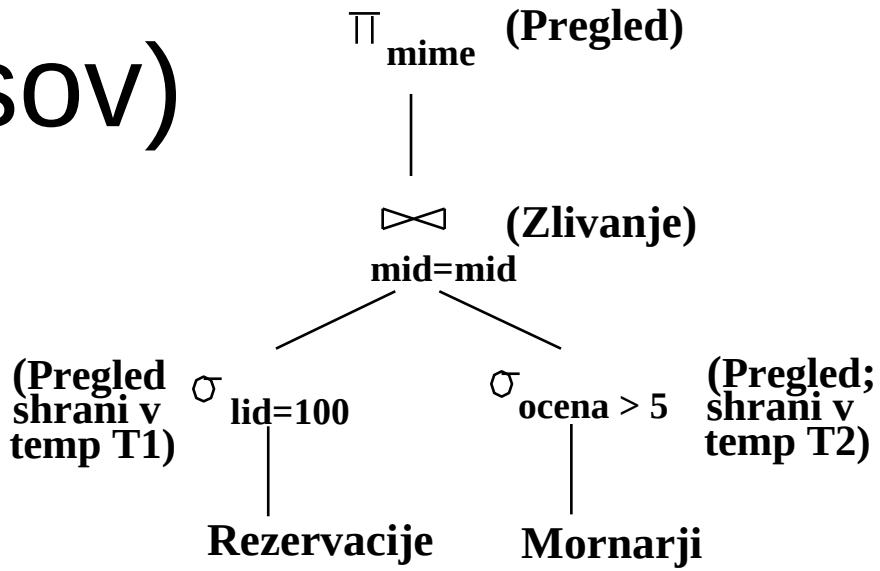
- Najslabši plan!
- Cena: 1000+1000*500[*100] V/I blokov
- Ne uporabi več možnosti: selekcije bi lahko “spustili” proti listom.
- Ni uporabe indeksov.
- *Cilj optimizacije:* Poiskati bolj optimalen plan, ki izračuna isti rezultat.

Plan 1 (brez indeksov)



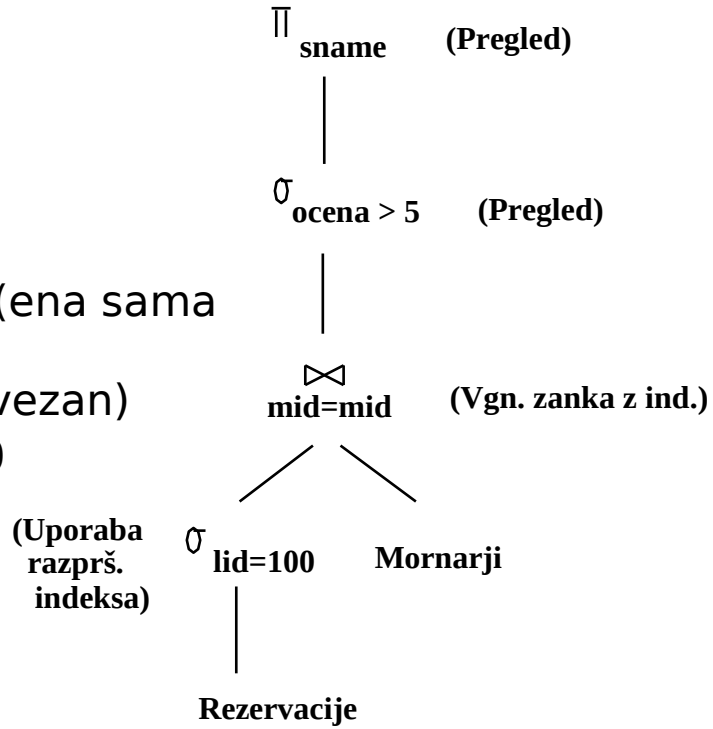
- ***Osnovna razlika:*** *spusti selekcije.*
- Cena je veliko bolj ugodna.
  - Tabela po selekciji na tabeli Rezervacije je majhna.
  - Tabela po selekciji na tabeli Mornarji je precej manjša kot original.
  - Stik se izvaja nad majhnimi tabelami.
- Ne prenašamo vseh atributov n-teric.
  - Potisni navzdol projekcije.
  - Prenašajo se samo tisti atributi, ki so potrebni.
  - Primer: T1 vsebuje samo še atribut *mid,* T2 pa atributa *mid* in *mime.*
  - Velikost n-teric tudi prispeva k ceni.

Plan 1
(brez indeksov)



- Cena plana:
  - Pregled Rezervacij (1000) + vpiši temp T1 (10 strani, 100 ladij, enakomerna porazdelitev).
  - Pregled Mornarjev (500) + vpiši temp T2 (250 strani, 10 ocen).
  - Sortiranje T1 (2*2*10), sortiranje T2 (2*3*250), zlivanje (10+250)
  - Skupaj: 1010+750+40+1500+260 = 3560 I/O blokov.

Plan 2 (z indeksi)



- Pohitritev selekcije nad Rezervacij.
  - Atribut *lid* v tabeli Rezervacije zelo selektiven (ena sama ladja).
  - Uporaba razprš. indeksa na *lid* (povezan/nepovezan)
  - Velikost: 16*1000*100/4000->400*1.25 = 500
- Pohitritev stika
  - Indeks za *mid* na tabeli Mornarji.
  - Majhno št. n-teric “notranje” relacije pri stiku.
  - Projekcija se pogosto “pridruži” selekciji.
- Pregled Rezervacij
  - Povezan=(500+1000) * 1/100 //selektivnost// =15
  - Nepovezan= (500+1000*100) * 1/100 = 1005
- Stik z indeksom
  - Cena = Pregled Rezervacij + 1000 //izbr.zapisi// * (1.2 [+1])
  - Cena = 1215 – 3205 V/I blokov

Vgnezdene poizvedbe

```
SELECT  M.mime
FROM  Mornarji M
WHERE EXISTS
(SELECT  *
 FROM  Rezervacije R
 WHERE  R.lid=103
 AND R.mid=M.mid)
```

- Vgnezden blok se optimizira neodvisno in zunanja n-terica se nad tem blokom preveri kot nad pogojem selekcije.
- Zunanji blok je optimiziran s ceno “klicanja” vgnezdenega bloka kot funkcijo.
- Implicitna urejenost vgnezdenih blokov omejuje pregled nekaterih dobrih strategij.
- *Poizvedbe brez gnezdenja se tipično bolje optimizirajo.*

Vgnezden blok za optim.:

```
SELECT  *
FROM  Rezervacije R
WHERE  R.lid=103
   AND M.mid= zun.vred.
```

Ekvivalentna p. brez vgnezd.p.:

```
SELECT  M.Mime
FROM Mornarji M, Rez R
WHERE  M.mid=R.mid
  AND R.lid=103
```

Povzetek

- Optimizacija poizvedb je pomembno opravilo relacijskega SUPB.
- Potrebno je poznati optimizacijo, da bi lahko razumeli vpliv načrtovanja podatkovne baze na izvajanje delovne obremenitve aplikacije.
- Dva dela optimizacije poizvedb:
  - Naštevanje alternativnih planov.
    - Iskalni prostor je potrebno zmanjšati: levo-usmerjeni plani.
  - Oceniti je potrebno vsak plan.
    - Velikost rezultata + cena plana vsakega vozlišča.
    - *Ključni parametri*: statistike, indeksi, implementacije operacij RA.

Povzetek

- Plani nad eno relacijo:
  - Pregledamo vse metode dostopa.
  - *Teme*: Selekcije, ki se ujemajo z indeksi; ali ima ključ indeksa vse potrebna polja; ali indeks vrne n-terice v pričakovanem vrstnem redu.
- Plani nad več relacijami:
  - Najprej naštejemo vse plane nad eno relacijo.
    - Selekcije in projekcije se izvedejo čim hitreje je mogoče.
  - Za vsak plan z *i* relacijami vzamemo najboljši plan nad *i-1* relacijami in ga povežemo z stikom z naslednjo “notranjo” relacijo oz. najboljšim planom nad eno relacijo.
  - Za vsako podmnožico relacij ohranimo samo najboljši plan.