Baze podataka

Predavanja

19. Optimiranje upita

Lipanj, 2021.

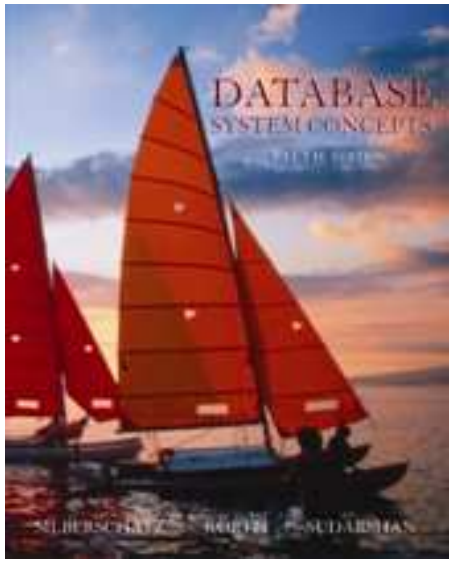
Obavljanje upita (*engl. Query Processing*)

- niz aktivnosti potrebnih da bi se dohvatilo podatke iz baze podataka
  - translacija upita pisanih u jeziku visoke razine u izraze koji se mogu primijeniti na fizičku razinu
  - optimiranje upita koje uključuje različite transformacije
  - izvršavanje upita

Literatura:

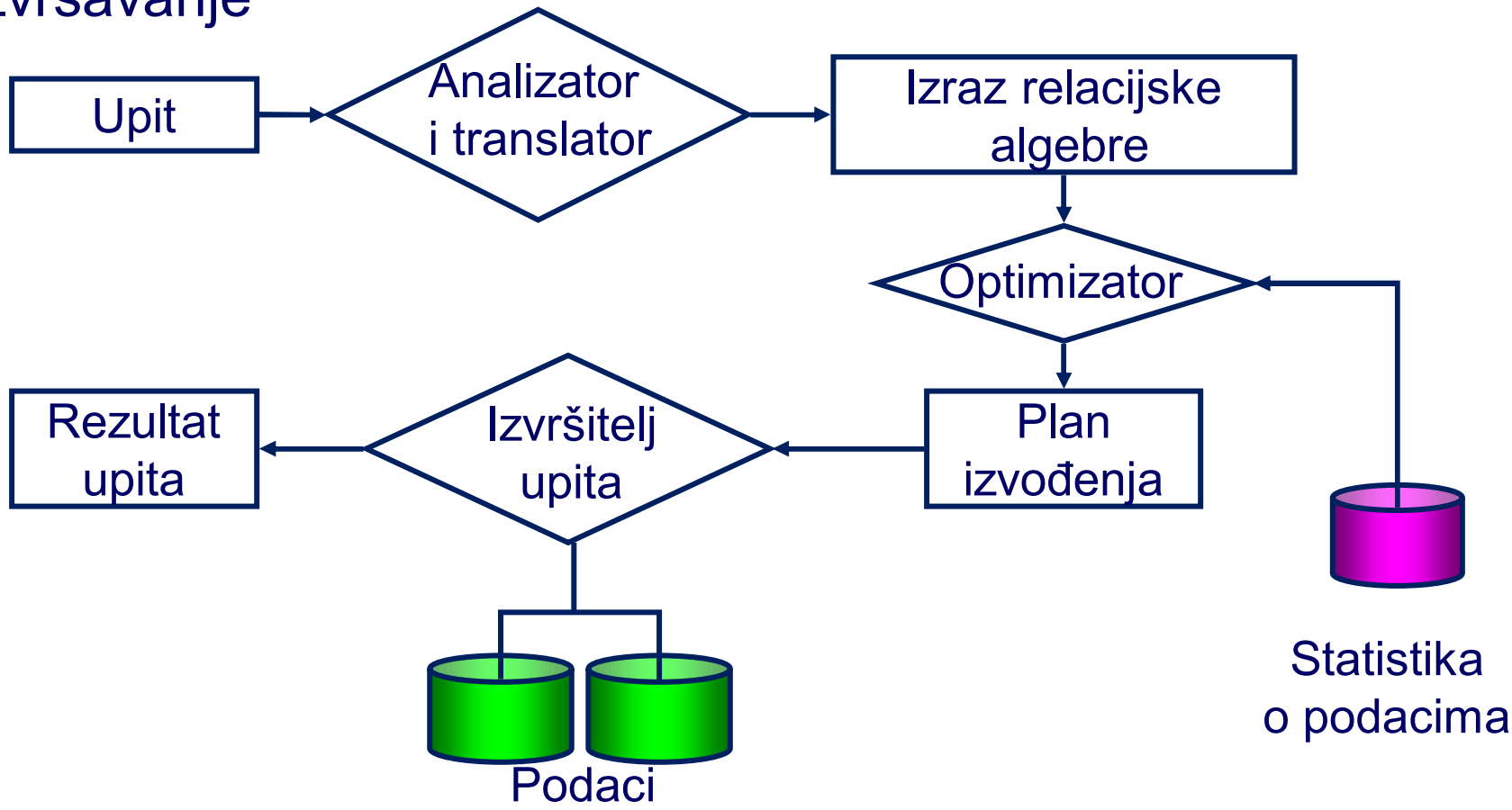
A. Silberschatz, H. Korth, S. Sudarshan,

Database System Concepts, McGraw-Hill, 2005



Osnovni koraci pri obavljanju upita

1. Analiza (parsiranje) i translacija
2. Optimiranje (engl. Query Optimization)
3. Izvršavanje



Procjena troškova - uloga rječnika podataka

Optimizator upita mora na raspolaganju imati određene podatke o tablicama koje sudjeluju u upitu.

- To su npr.
  - N(r) - broj n-torki u tablici
  - informacija o tome da li atribut sadrži jedinstvene vrijednosti
  - koji indeksi su izgrađeni nad tablicom, radi li se o uzlazno ili silazno poredanom indeksu, radi li se o *Cluster indexu* (podaci u podatkovnim blokovima su poredani prema ključu)
  - dubina B-stabla za indeks
  - V(A, r) - broj različitih vrijednosti atributa A u tablici r
    - V(A, r) = $\mathcal{G}_{\text{COUNT}(*)}(\pi_{\text{A}}(\text{ r }))$
    - A može biti skup atributa
  - broj jedinstvenih vrijednosti u indeksu
  - distribucija vrijednosti u pojedinim atributima tablice

Procjena troškova - uloga rječnika podataka

- Optimiranje bi postalo izuzetno neefikasno kada bi se ovi statistički podaci prikupljali iz baze podataka prilikom optimiranja svakog upita.
- Postoji posebna naredba kojom se pokreće prikupljanje ovih podataka, a rezultati se zapišu u rječnik podataka
    - PostgreSQL: administrator obavlja naredbu: **`VACUUM ANALYZE`**
- Odgovornost je administratora baze podataka da uvijek kada se bitno promijene podaci koji bi mogli utjecati na rad optimizatora podataka (npr. u tablicu se upiše relativno veliki broj zapisa) izvede naredbu za ažuriranje rječnika podataka.
- U suprotnom, moglo bi se dogoditi da optimizator, temeljeći svoje odluke na pogrešnim statističkim podacima, generira neoptimalne planove obavljanja.
- Danas svi sustavi imaju ugrađenu i automatiku (koja se može i isključiti) za periodičko ili *on-event* aktiviranje ažuriranja statistike.

Analiza (parsiranje) i translacija

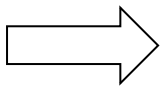
- Pretvorba upita pisanog u upitnom jeziku (SQL) u izraz relacijske algebre
- Zamjena virtualnih tablica s temeljnim tablicama koje proizlaze iz definicije virtualne tablice

Primjer:

FILMSKA ZVIJEZDA = {ime, adresa, spol, datrod} $K_{FILMSKAZVIJEZDA}$ = {ime}
GLUMIU = {naslov, godina, imeZvijezda} K_{GLUMIU} = {naslov, godina, imeZvijezda}

```
SELECT *
  FROM filmskaZvijezda, glumiU
 WHERE imeZvijezda = ime
   AND godina = 2012
   AND spol = 'M'
```

analiza i translacija u početni izraz relacijske algebre



Translacija u početni izraz relacijske algebre

FILMSKA ZVIJEZDA = {ime, adresa, spol, datrod} $K_{FILMSKAZVIJEZDA}$ = {ime}

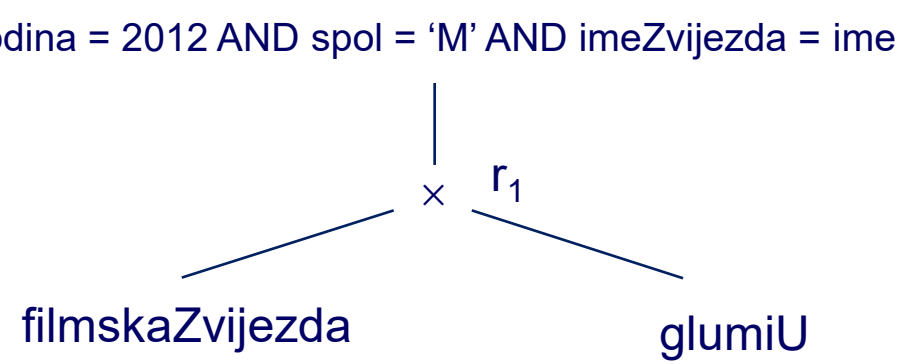
GLUMIU = {naslov, godina, imeZvijezda} K_{GLUMIU} = {naslov, godina, imeZvijezda}

```
SELECT *
  FROM filmskaZvijezda, glumiU
 WHERE imeZvijezda = ime
   AND godina = 2012
   AND spol = 'M'
```

izraz relacijske algebre:

$$\sigma_{godina=2012\ AND\ spol='M'\ AND\ imeZvijezda = ime}(filmskaZvijezda \times glumiU)$$

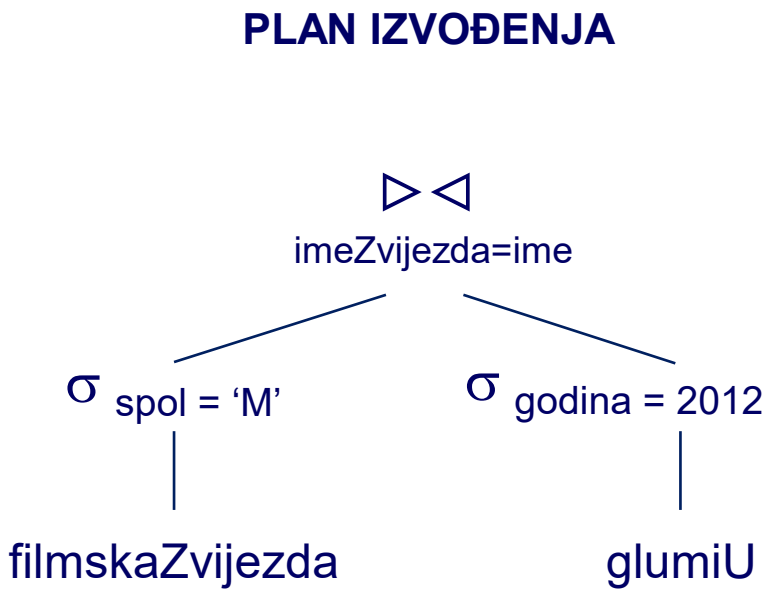
- stablo upita (*query tree*) je način prikaza izraza relacijske algebre
- listovi stabla su tablice, a ostali čvorovi su operacije relacijske algebre



card (r_1) = card(filmskaZvijezda)*card(glumiU)

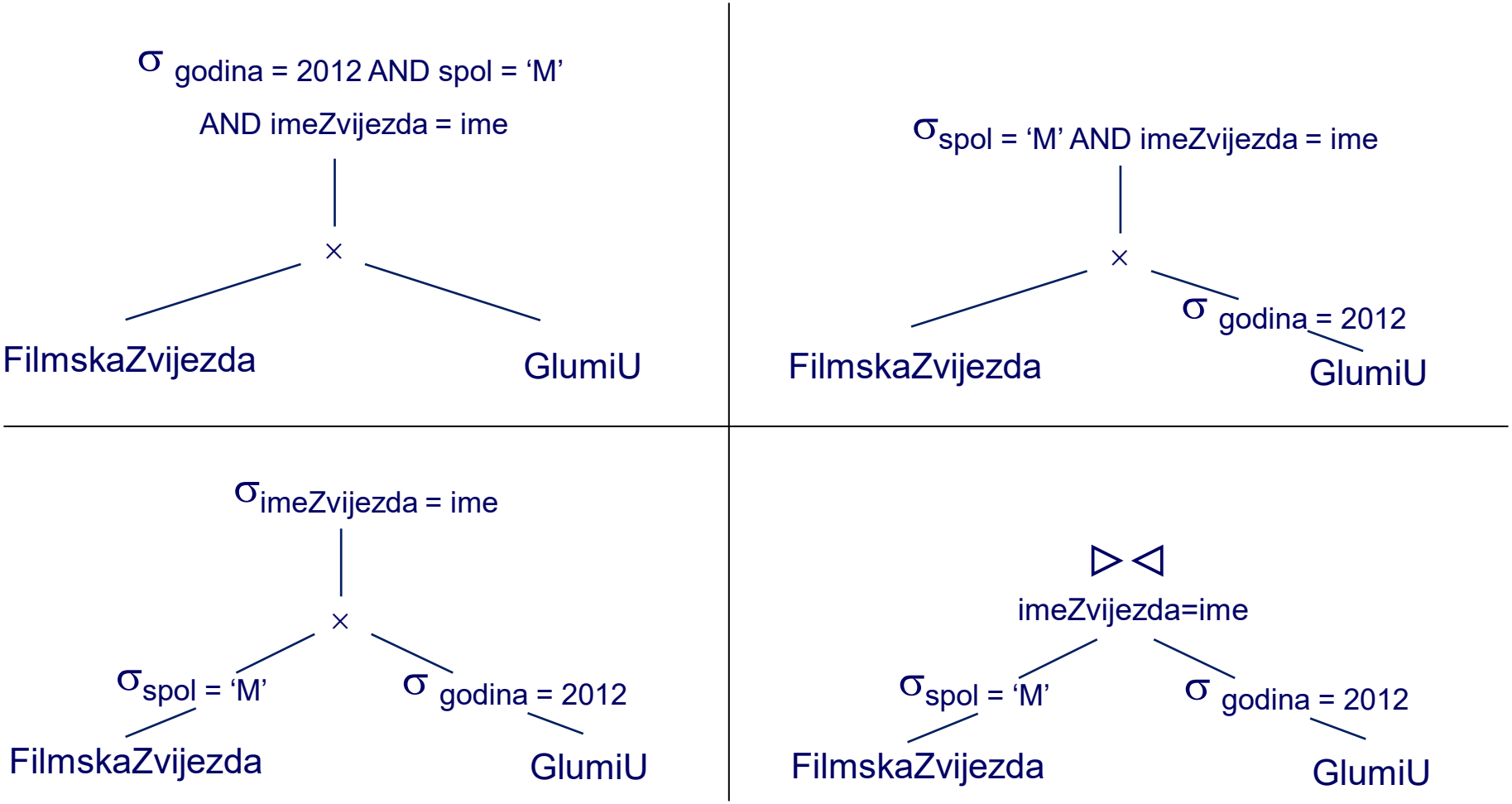
Plan izvođenja (engl. Execution Plan)

- Odabir operanada i operacija (σ, $\triangleright\triangleleft$, …)
- Redoslijed izvođenja operacija
- Detaljni plan izvođenja operacija
  - metode pristupa podacima
  - redoslijed spajanja
  - metode spajanja
  - stvaranje privremenih indeksa?
  - sortiranje privremenih rezultata
  - ….



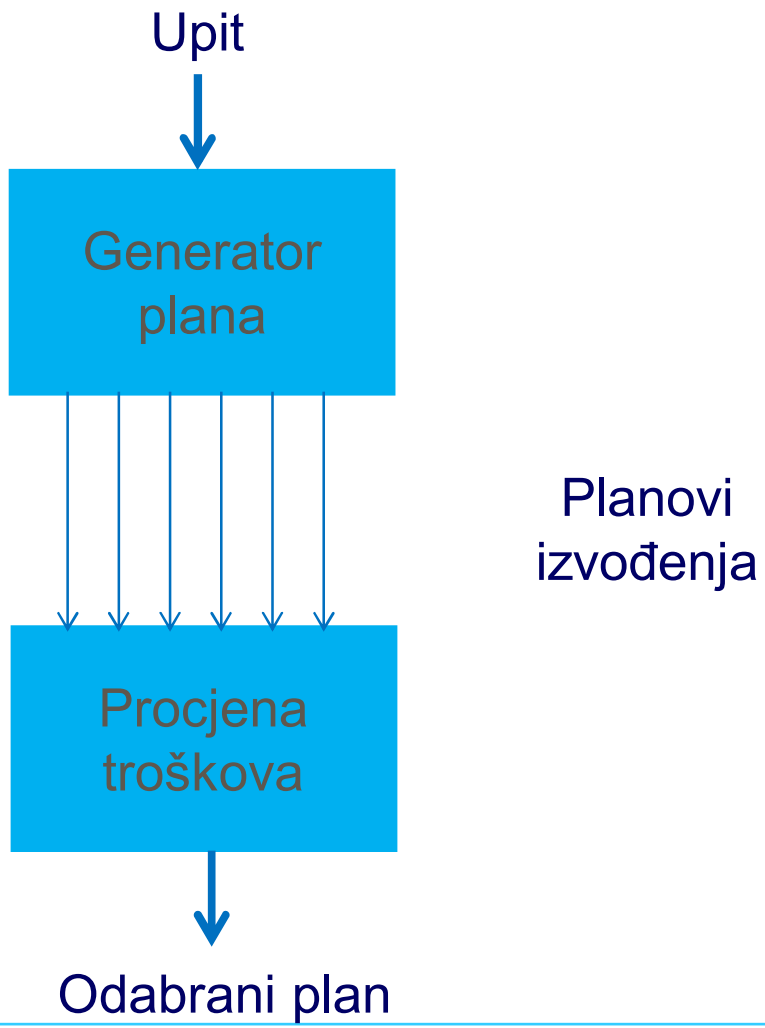
Plan izvođenja nije jednoznačno određen upitom

U općem slučaju se izraz relacijske algebre može zamijeniti ekvivalentnim, alternativnim izrazom relacijske algebre (jednim od mnogih):



Optimiranje upita

Proces odabira *najprikladnijeg* od mogućih planova izvršavanja.



Ekvivalentni izrazi relacijske algebre

- Različiti planovi izvođenja dobiju se za
  - različiti odabir operacija
  - različiti redoslijed obavljanja operacija
  - različit odabir metoda izvršavanja pojedinih operacija
  - …
- Dva izraza relacijske algebre su ekvivalentna ako nad svakom instancom baze podataka daju jednaki rezultat
- Alternativni izrazi se određuju na temelju pravila za transformaciju izraza relacijske algebre

Algebarske transformacije

- Prirodno spajanje
  - komutativno: $r \bowtie s = s \bowtie r$
  - asocijativno: $(r \bowtie s) \bowtie t = r \bowtie (s \bowtie t)$
- Operacije $\times$, $\cup$, $\cap$ su komutativne i asocijativne
- θ-spajanje nije uvijek asocijativno! Primjer: tablice **r(A,B)**, **s(C, D)**, **t(E,F)**

  $(r \underset{B>C}{\bowtie} s) \underset{A<F}{\bowtie} t \neq r \underset{B>C}{\bowtie} (s \underset{A<F}{\bowtie} t)$ ➔ **A** nije atribut iz **s**, niti iz **t** !!!

- θ-spajanje $(r \underset{\theta_1}{\bowtie} s) \underset{\theta_2}{\bowtie} t$ može biti asocijativno

  ako se uvjet θ_2 podijeli na θ_3 i θ_4 tako da θ_3 uključuje samo atribute iz **s** i **t** :

  $(r \underset{\theta_1}{\bowtie} s) \underset{\theta_2}{\bowtie} t = r \underset{\theta_1 \wedge \theta_4}{\bowtie} (s \underset{\theta_3}{\bowtie} t)$

Pravila koja se odnose na selekciju

- Podjela:
  - $\sigma_{C1\ AND\ C2}(r) = \sigma_{C1}(\sigma_{C2}(r)) = \sigma_{C2}(\sigma_{C1}(r))$
  - $\sigma_{C1\ OR\ C2}(r) = \sigma_{C1}(r) \cup \sigma_{C2}(r)$
- Potiskivanje selekcije
  - **ako je uvjet C primjenjiv samo na r:**

    $\sigma_C(r \bowtie s) = (\sigma_C(r)) \bowtie s$
  - **ako je uvjet C primjenjiv samo na s:**

    $\sigma_C(r \bowtie s) = r \bowtie (\sigma_C(s))$
  - **ako je uvjet C primjenjiv na r i na s:**

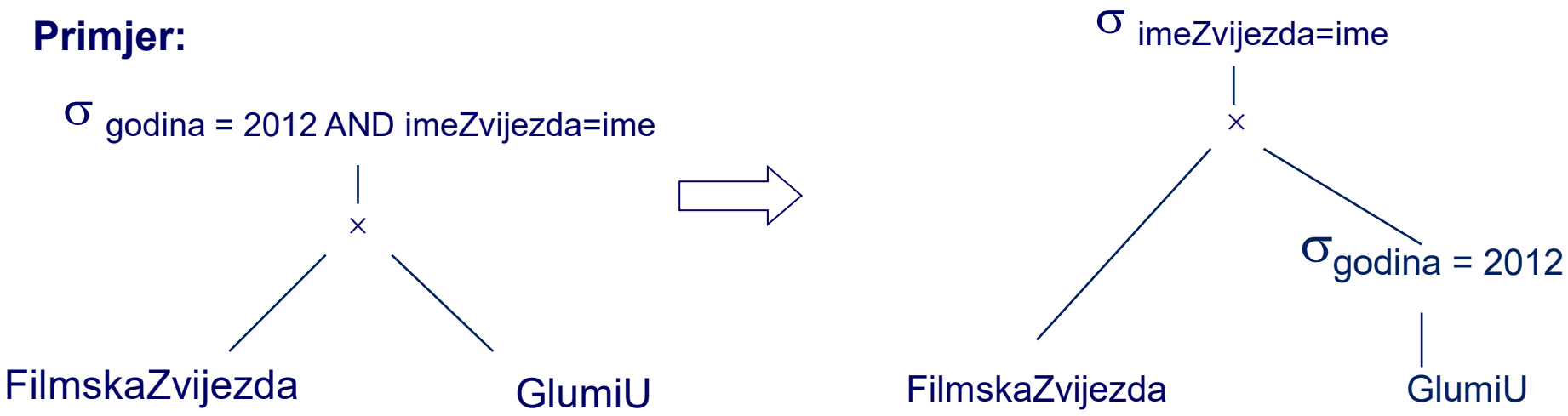
    $\sigma_C(r \bowtie s) = (\sigma_C(r)) \bowtie (\sigma_C(s))$
- Na isti se način selekcija može potiskivati u odnosu na Kartezijev produkt i spajanje uz uvjet
- U ovim predavanjima su navedena samo neka od pravila algebarskih transformacija (nedostaje npr. projekcija, grupiranje,…)

Heuristička pravila

Nije moguće generirati i analizirati sve moguće planove izvođenja (preskupo). Broj planova koji će se procjenjivati reducira se pomoću heurističkih pravila.

1. Potiskivanje selekcije: operaciju selekcije obaviti u čim ranijoj fazi.

Primjer:

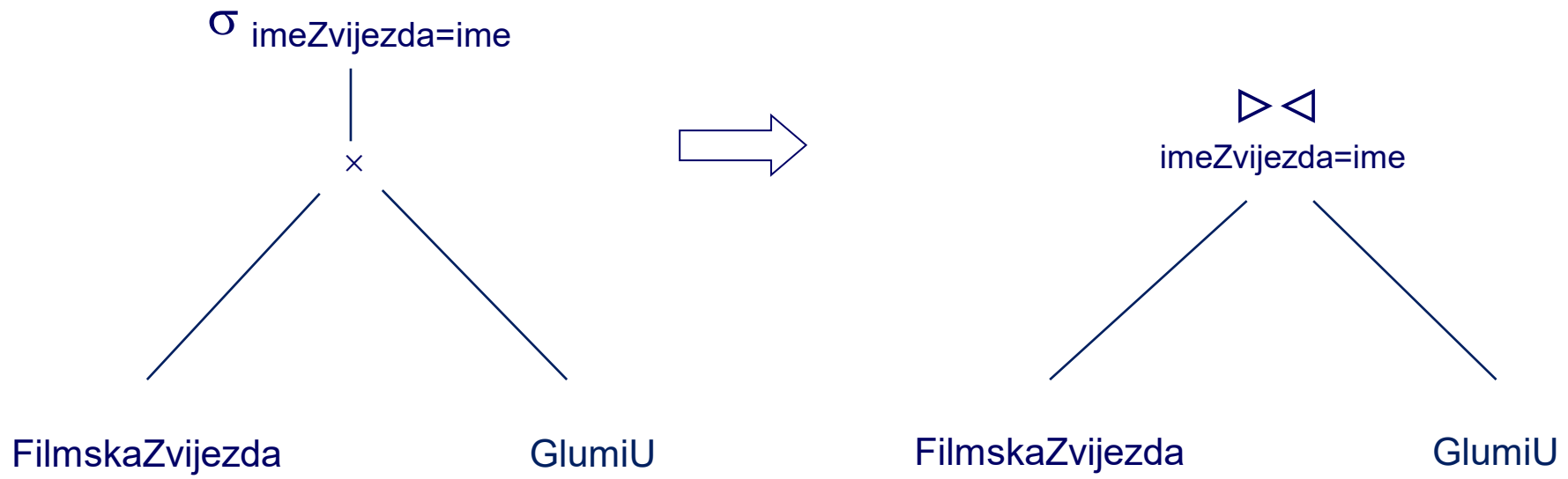


- **Potiskivanje selekcije je jedno od heurističkih pravila koja se koriste pri optimizaciji.**
- **Heuristička pravila**
  - **proizlaze iz iskustva, temelje se na ekvivalentnim izrazima relacijske algebre**
  - **njihova primjena u većini slučajeva rezultira efikasnijim planom izvršavanja**

Heuristička pravila

2. Kombiniranje operacije selekcije i Kartezijevog produkta

Primjer:



- u ovim predavanjima navedena su tek dva od mnogih heurističkih pravila

Izračunavanje troška izvršavanja operacija

- Primjenom heurističkih pravila ne dobiva se uvijek jednoznačan i/ili uvijek najbolji mogući plan
- Za svaki od dobivenih planova izvođenja procjenjuje se ukupni trošak izvršavanja. Odabire se plan s najmanjim procijenjenim troškom (*cost based optimization*)
- Najveći dio troška izvršavanja upita odnosi se na trošak obavljanja U/I operacija
- Važna je veličina međurezultata!
- Veličina međurezultata ovisi o
  - broju n-torki u međurezultatu (može se samo procijeniti)
  - veličini n-torke u međurezultatu (dobije se izravno iz metapodataka)

Procjena veličine rezultata selekcije

- Jednostavna pretpostavka:
  - Jednolika razdioba vrijednosti atributa A

- Selekcija uz uvjet **A** = c:

$$N(\sigma_{A=c}(r)) = N(r) / V(A, r)$$

  - ako vrijednost **c** uopće ne postoji???

- Selekcija koja uključuje nejednakost $\sigma_{A<c}(r)$
  - pretpostavka - 1/3 n-torki zadovoljava uvjet:

$$N(\sigma_{A<c}(r)) = N(r) / 3$$

- Složeni uvjeti: $\sigma_{A=c1 \text{ AND } B<c2}(r)$:

$$N(\sigma_{A=c1 \text{ AND } B<c2}(r)) = N(\sigma_{A=c1}(\sigma_{B<c2}(r)))$$
$$= (N(r)/3)/V(A, r) = N(r)/(3*V(A, r))$$

Procjena veličine rezultata spajanja

- Neka je t = r ▷◁ s. Neka su X i Y skupovi atributa iz **r**, odnosno **s**

1. **X ∩ Y = ∅** Spajanje je produkt - N(t) = N(r) * N(s)
   - nema eliminacije duplikata!
2. **X ∩ Y je ključ od r** - svaka n-torka iz **s** može se spojiti s najviše jednom n-torkom iz **r** : N(t) = N(s)
3. **X ∩ Y nije prazan skup, nije ključ od r ili s**, neka je A = X ∩ Y
   - Svaka n-torka iz **r** spaja se s N(s) / V (A, s)

     N(t) = N(r) * N(s) / V(A, s)
   - ili ako krenemo s n-torkama iz **s** N(t) = N(r) * N(s) / V(A, r)
   - Ako je V(A, s) ≠ V(A, r), tada se računa s većim od mogućih nazivnika:

     N(t) = N(r) * N(s) / max(V(A, r), V(A,s))

Primjer:

- Važna primjena procjene veličine je procjena planova u kojima se operandi spajaju različitim poretkom

r(A, B)	s(B, C)	t (C, D)
N (r) = 1000	N (s) = 2000	N (t) = 5000
V (B, r) = 20	V (B, s) = 50	V (C, t) = 500
	V (C, s) = 100	

Obavlja se operacija: $r \triangleright\triangleleft s \triangleright\triangleleft t$

- optimizator ne raspolaže informacijom o ključevima tablica

1. Prvo se spajaju **r** i **s**:

   $N (r \triangleright\triangleleft s) = 1000 \times 2000 / \max(20, 50) = 40.000$

   $V (C, r \triangleright\triangleleft s) = 100$

   $N ((r \triangleright\triangleleft s) \triangleright\triangleleft t) = 40.000 \times 5000 / \max(100, 500) =$ **400.000**

r(A, B)	s(B, C)	t (C, D)
N (r) = 1000	N (s) = 2000	N (t) = 5000
V (B, r) = 20	V (B, s) = 50	V (C, t) = 500
	V (C, s) = 100	

2. Prvo se spajaju **s** i **t**:

N (s ⋈ t) = 2000 × 5000 / max(100, 500) = 20.000

V (B, s ⋈ t) = 50

N (r ⋈ (s ⋈ t)) =1000×20.000 / max(20, 50) = **400.000**

3. Prvo se spajaju **r** i **t**:

N (r ⋈ t) = 1000 × 5000 = 5.000.000

V (B, r ⋈ t) = 20

V (C, r ⋈ t) = 500

N ((r ⋈ t) ⋈ s)) = 5.000.000 × 2000/(max(20, 50)×max(100, 500)) = **400.000**

Redoslijed spajanja

- procijenjena veličina je 400,000 n-torki bez obzira na redoslijed spajanja ➔ Slučajnost?

- za stvaranje i rukovanje međurezultatima potrebni su značajni resursi
- **kriterij odabira redoslijeda spajanja operanada je veličina međurezultata**

Metode pristupa podacima u tablici (*access plan*)

- Čitanje n-torki:
  - slijednim čitanjem blokova podataka (*sequential scan, table-scan*)
  - korištenjem indeksa
    - *index-only scan*
    - *index scan*

Metode pristupa podacima u tablici

Slijedno čitanje podataka iz tablice (*sequential scan*)

- koristi se kad nema 'upotrebljivog' indeksa prema kojem bi se obavljala selekcija ili kada ionako treba pročitati sve ili vrlo velik broj n-torki iz tablice

```
CREATE TABLE stud (
    mbrStud INTEGER
  , prezStud CHAR(25)
  , imeStud  CHAR(25)
);
CREATE INDEX prez_ime ON stud (prezStud, imeStud)
```

```
SELECT * FROM stud WHERE prezStud = 'Horvat'
```

– **za selekciju se koristi indeks *prez_ime***

```
SELECT * FROM stud WHERE imeStud = 'Ivo'
```

– **indeks *prez_ime* nije upotrebljiv**

Čitanje pomoću indeksa (*index scan*)

```
CREATE INDEX prez_ime ON stud (prezStud, imeStud)
```

- **index-only scan**
  ako su svi podaci koji se čitaju iz tablice dijelovi JEDNOG indeksa, sve potrebne vrijednosti se mogu pronaći u indeksnim blokovima

```
SELECT imeStud, prezStud
  FROM stud
 WHERE prezStud = 'Horvat'
```

- **index scan**
  ako se svi potrebni podaci ne mogu naći u indeksu, **mora se pristupati podatkovnim blokovima**

```
SELECT mbrStud, imeStud, prezStud
  FROM stud
 WHERE prezStud = 'Horvat'
```

Metode spajanja tablica (*join*)

- Prikazat će se samo neke metode spajanja tablica:
  - Spajanje ugniježđenim petljama (*nested loop join*)
  - Raspršeno spajanje (*hash join*)

Spajanje ugniježđenim petljama

Postupak:

- tablice koje se spajaju zovu se vanjska tablica (outer table) i unutarnja tablica (inner table). (Ovi pojmovi nemaju veze s vanjskim spajanjem (outer join) i unutarnjim spajanjem (inner join) u relacijskoj algebri, odnosno SQL-u!)
- iz vanjske tablice se čita svaka n-torka
  - ako postoji uvjet selekcije, čitaju se samo one n-torke koje zadovoljavaju uvjet. Pri tome, ukoliko postoji upotrebljiv indeks za obavljanje uvjeta selekcije, kao metoda za pristup n-torkama iz vanjske tablice koristi se index scan ili index-only scan
- za svaku pročitanu n-torku iz vanjske tablice traže se n-torke iz unutarnje tablice koje zadovoljavaju uvjet spajanja

Spajanje ugniježđenim petljama

SUBP mora za svaku n-torku iz vanjske tablice po jednom pretražiti cijelu unutarnju tablicu

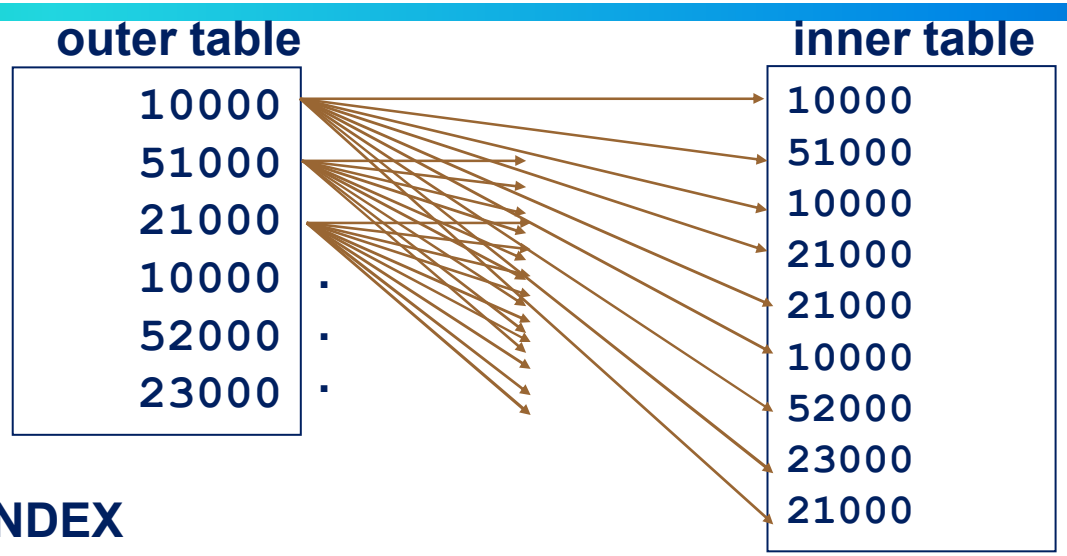
Način pristupa podacima iz unutarnje tablice:

a) ako se u unutarnjoj tablici nalazi mali broj n-torki može se koristiti slijedno pretraživanje

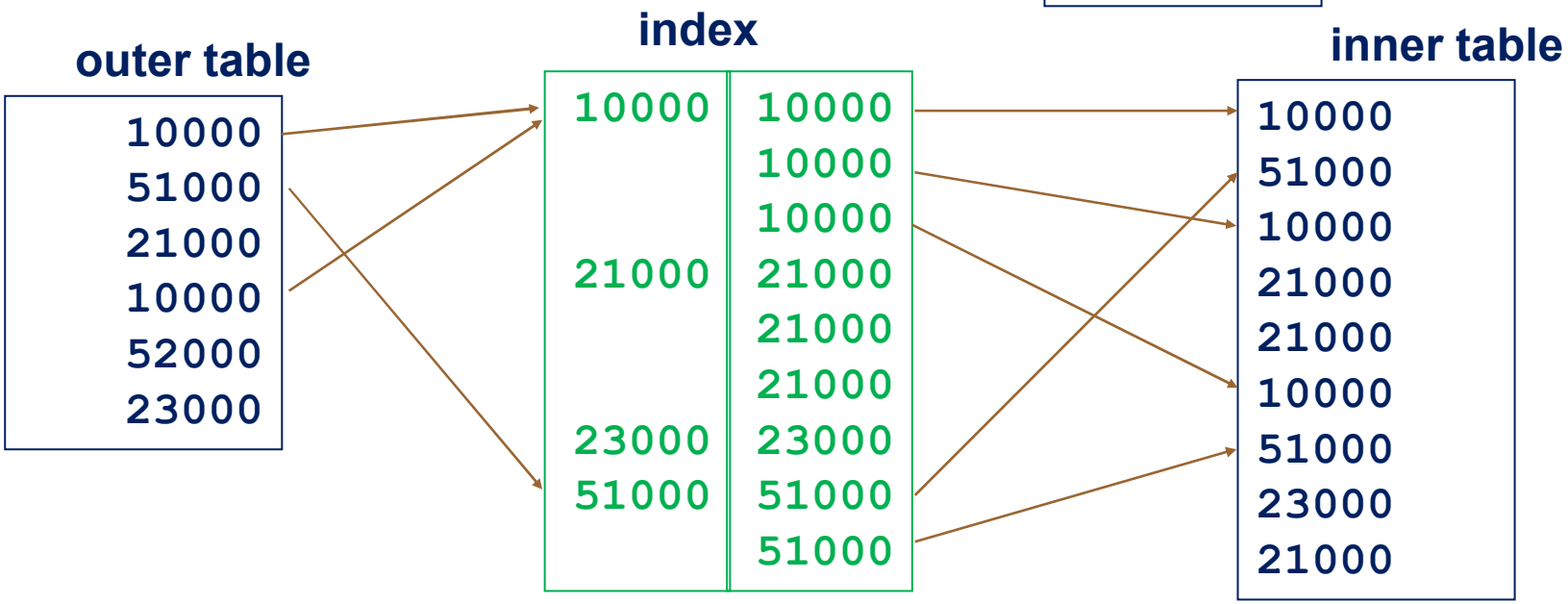
b) ukoliko postoji upotrebljivi indeks nad unutarnjom tablicom, za pretragu se koristi taj indeks

Spajanje ugniježđenim petljama

SLIJEDNO



INDEX ili AUTO-INDEX



Raspršeno spajanje

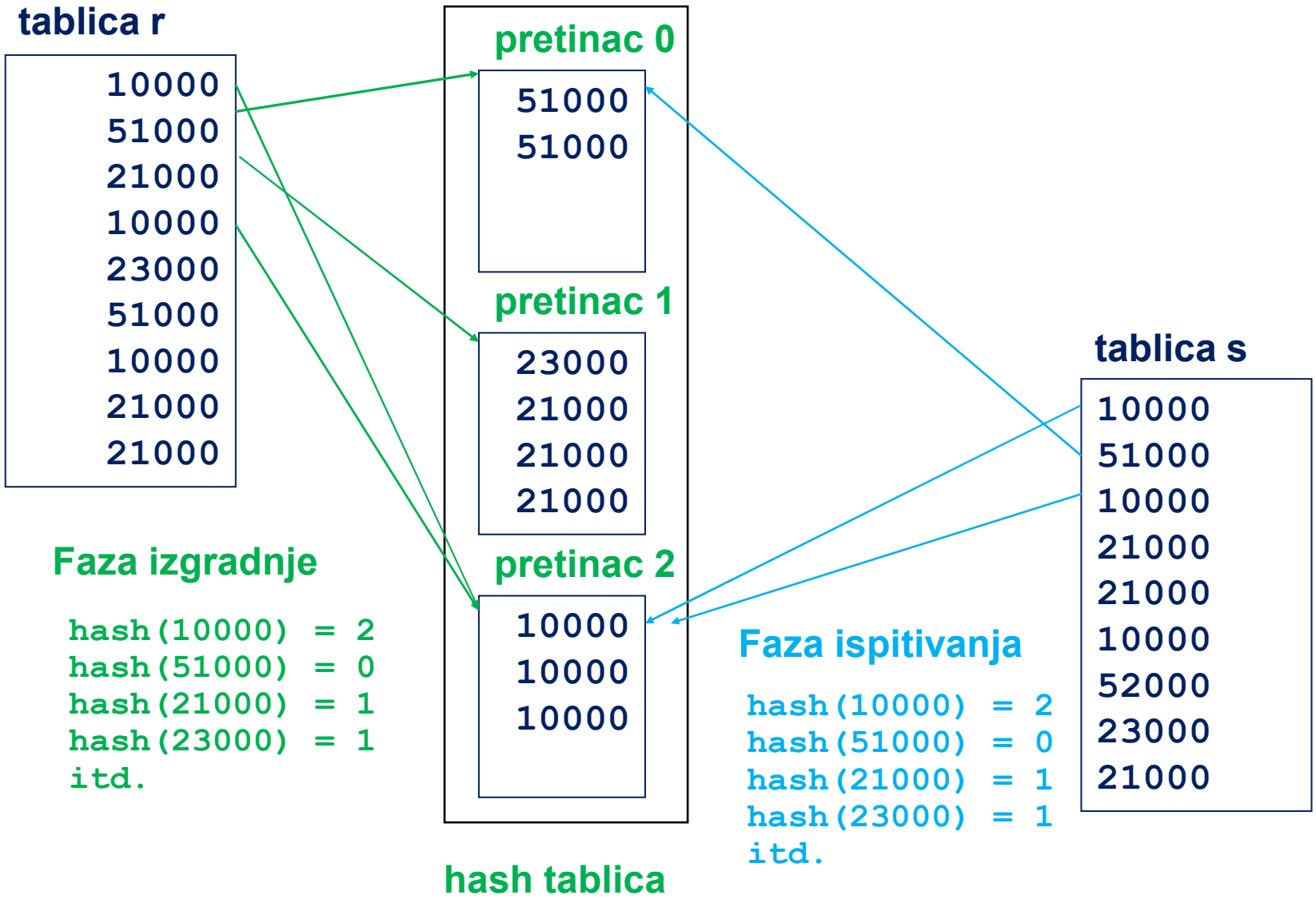
Faza izgradnje:

- Raspršena tablica se izgrađuje za samo jednu tablicu iz para kojeg treba spojiti (npr. tablicu r iz para tablica r i s).
- Funkcija raspršenja se primjenjuje na vrijednosti atributa ili vrijednosti skupa atributa prema kojima se obavlja spajanje.
- U obzir se uzimaju samo one n-torke iz tablice r koje zadovoljavaju eventualno postavljeni uvjet selekcije
- pri tome se za njihovo izdvajanje, ako je moguće, primjenjuje indeks, slično kao kod čitanja n-torki iz vanjske tablice kod spajanja s ugniježđenom petljom.

Faza ispitivanja:

- Čita sadržaj tablice **s** (uzimaju se samo n-torke koje zadovoljavaju eventualni uvjet selekcije, ukoliko je moguće koristi se indeks)
- Za svaku pročitanu n-torku pomoću funkcije raspršenja, a na temelju vrijednosti atributa iz tablice **s** prema kojima se obavlja spajanje, izračuna se u kojem se džepu raspršene tablice nalaze zapisi iz tablice **r** s kojima se ta n-torka treba spojiti.

Raspršeno spajanje



Primjer

Zadane su tablice *mjesto*, *stud* i *ispit* sa sljedećim shemama:

MJESTO = {pbr, nazMjesto} K_{MJESTO} = {pbr}
N(mjesto) = 500
V(nazMjesto, mjesto) = 500

STUD = {mbrStud, prezStud, pbr} K_{STUD} = {mbrStud}
N(stud) = 10 000
V(prezStud, stud) = 1000

ISPIT = {mbrStud, sifPred, datRok, ocjena} K_{ISPIT} = {mbrStud, sifPred, datRok}
N(ispit) = 100 000
V(ocjena, ispit) = 5

Dodatni indeks je kreiran samo nad atributom *prezStud* tablice *stud*.

Optimizator ima na raspolaganju sljedeće metode:
- pristup podacima u tablici bez indeksa (*sequential scan*)
- pristup podacima u tablici preko indeksa (index path)

- raspršeno spajanje (hash join)
- spajanje pomoću ugniježđene petlje (nested-loop join)

Primjer

Za upit

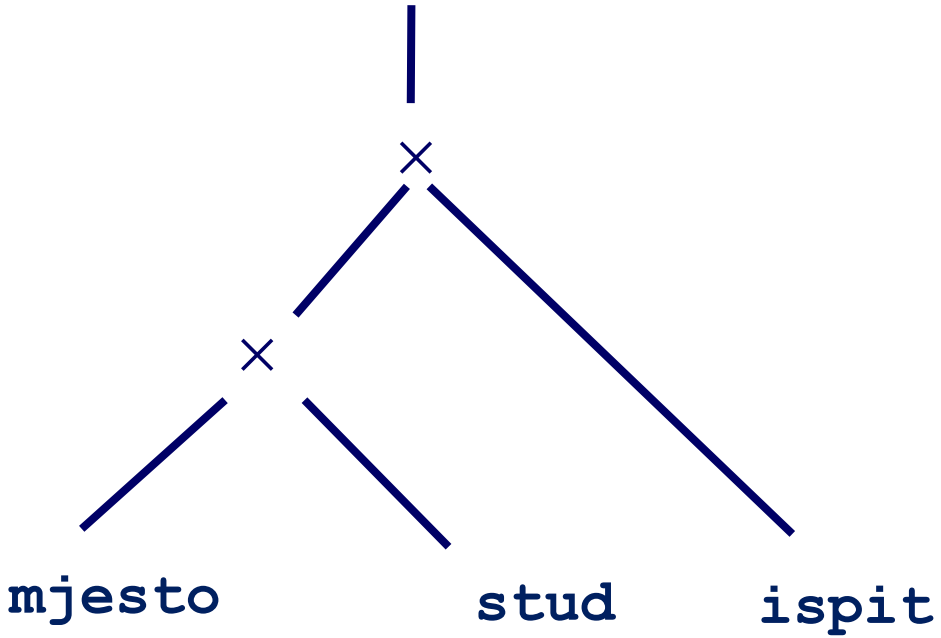
```
SELECT * FROM mjesto, stud, ispit
   WHERE mjesto.pbr = stud.pbr
     AND stud.mbrStud = ispit.mbrStud
     AND stud.prezStud = 'Horvat'
     AND ispit.ocjena = 5
```

a) Nacrtati stablo upita za početni plan izvođenja upita pri čemu je redoslijed spajanja tablica određen redoslijedom kojim su tablice navedene u FROM dijelu SELECT naredbe.
b) Nacrtati stablo upita nakon provedene heurističke optimizacije. Procijeniti broj n-torki u međurezultatima. U stablu upita naznačiti korištene metode pristupa podacima.
c) Procijeniti broj n-torki za različite moguće redoslijede spajanja međurezultata.

Primjer – a)

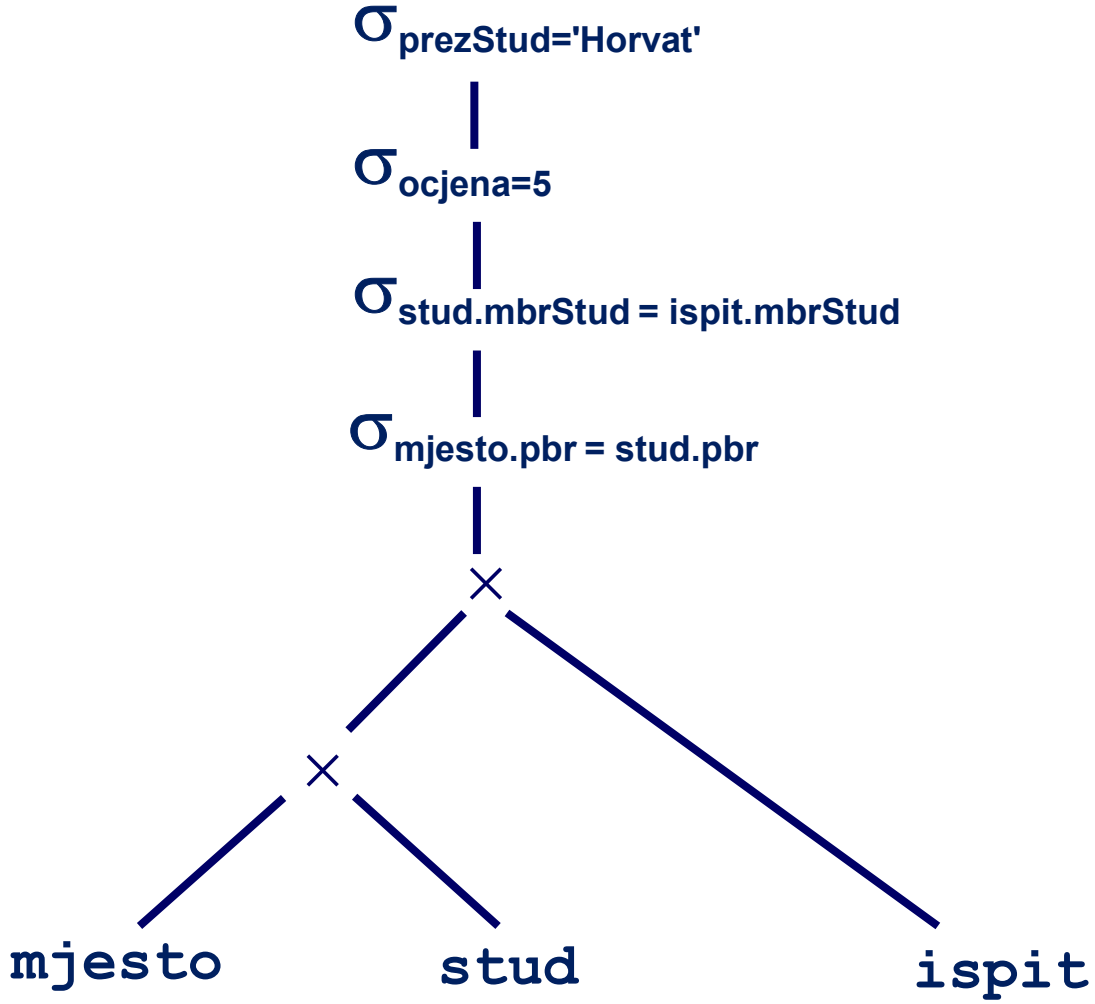
```
SELECT * FROM mjesto, stud, ispit
   WHERE mjesto.pbr = stud.pbr
     AND stud.mbrStud = ispit.mbrStud
     AND stud.prezStud = 'Horvat'
     AND ispit.ocjena = 5
```

$\sigma_{\text{prezStud='Horvat' AND ocjena=5 AND mjesto.pbr = stud.pbr AND stud.mbrStud = ispit.mbrStud}}$



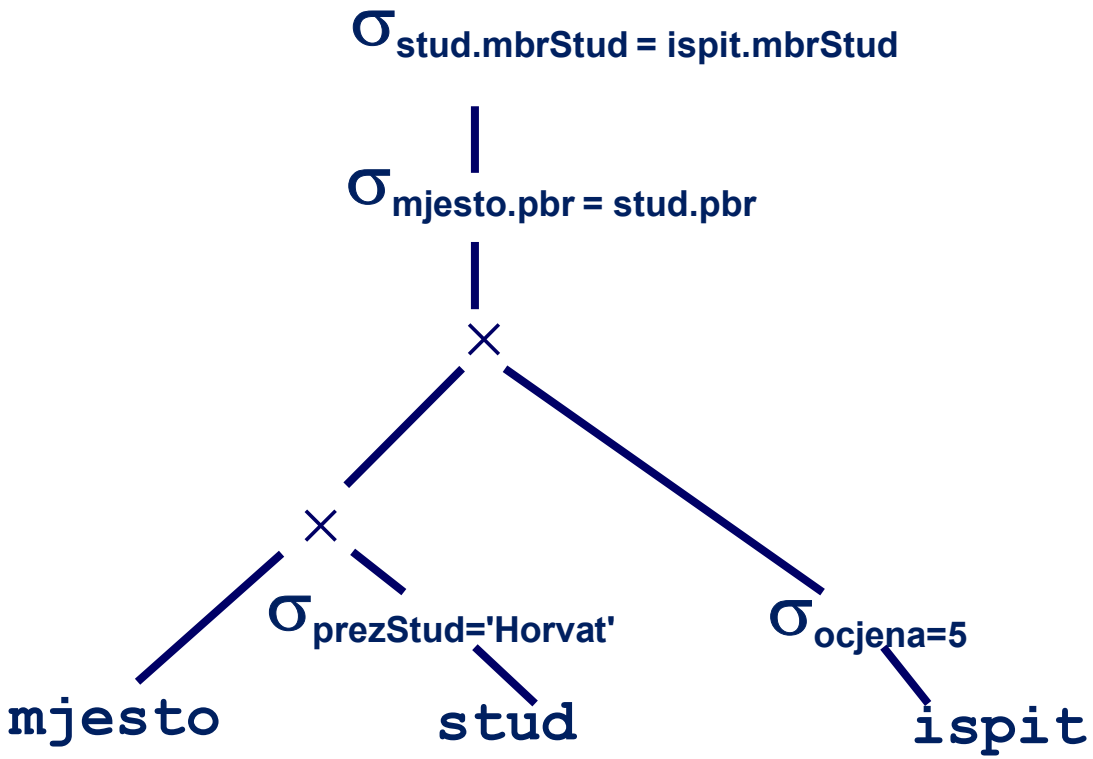
Primjer – b)

Rastavljanje uvjeta selekcije:



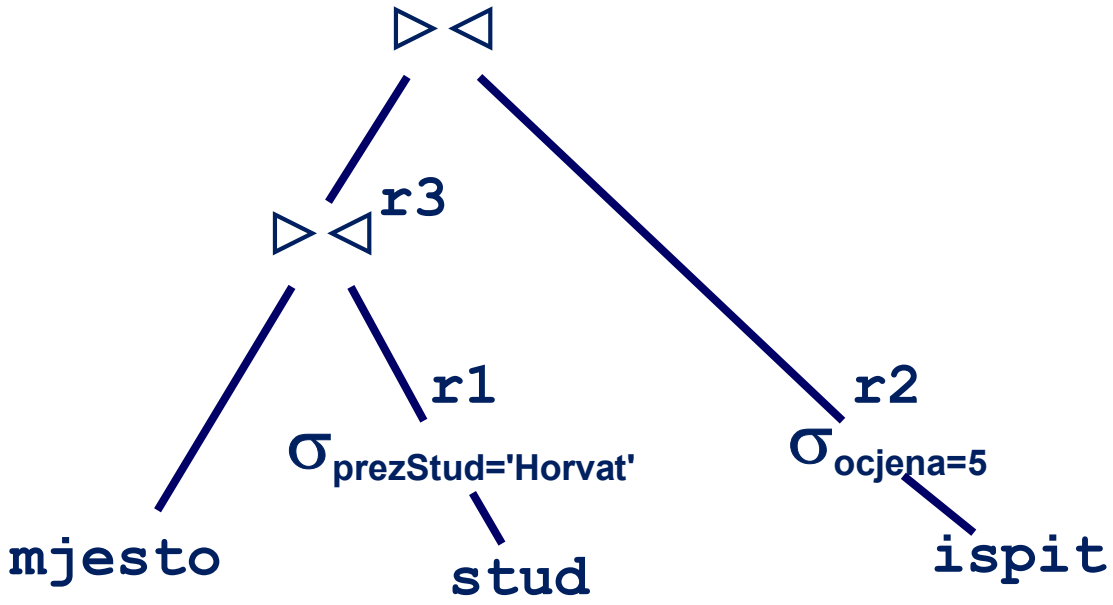
Primjer – b)

Potiskivanje selekcije:



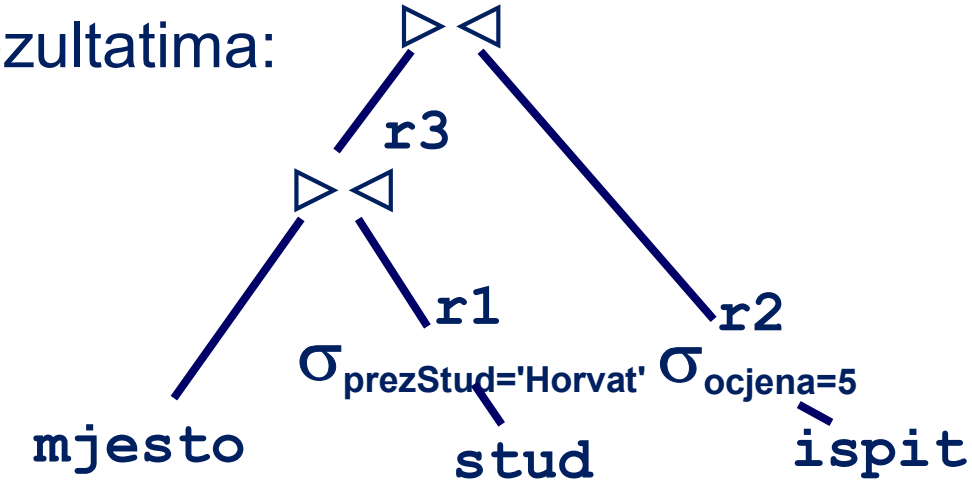
Primjer – b)

Kombiniranje operacije selekcije i Kartezijevog produkta



Primjer – b)

Procjena broja n-torki u međurezultatima:



$r_1 = \sigma_{prezStud='Horvat'}(stud)$

$N(r_1) = N(stud) / V(prezStud, stud) = 10000/1000 = 10$

$r_2 = \sigma_{ocjena=5}(ispit)$

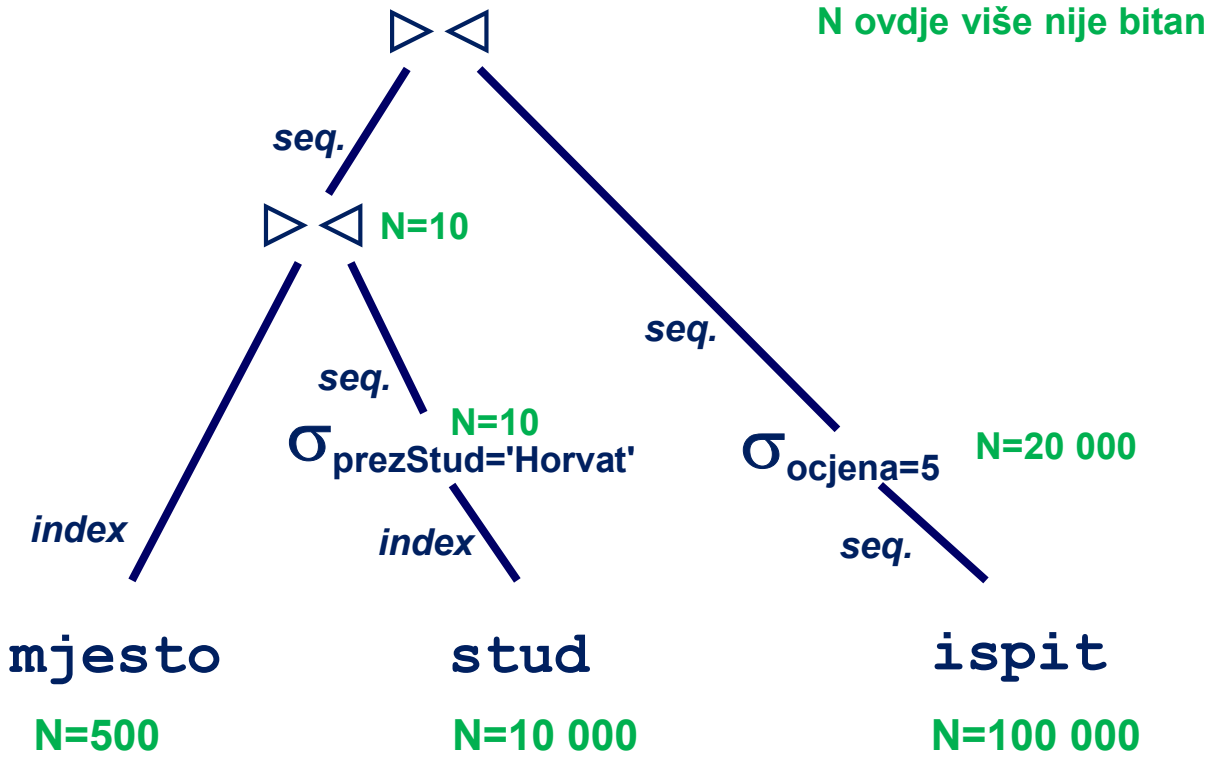
$N(r_2) = N(ispit) / V(ocjena, ispit) = 100000/5 = 20000$

$r_3 = mjesto \triangleright\!\triangleleft r_1$

$N(r_3) = N(r_1) = 10$ (mjesto $\cap$ r_1 je ključ tablice mjesto)

Primjer – b)

Korištene metode pristupa podacima



Primjer – c)

Procjena broja n-torki u međurezultatu za različite redoslijede spajanja

$r_1 = \sigma_{prezStud='Horvat'}(stud)$ $\quad N(r_1) = 10$

$r_2 = \sigma_{ocjena=5}(ispit)$ $\quad N(r_2) = 20000$

N (mjesto $\triangleright\triangleleft$ r_1) $\leq N(stud1) = 10$ (mjesto $\cap$ r_1 je ključ tablice mjesto)

N (mjesto $\triangleright\triangleleft$ r_2) $= 500*20\ 000$ (Kartezijev produkt)

N ($r_1 \triangleright\triangleleft r_2$) $= N(r_2) = 20\ 000$ ($r_1 \cap r_2$ je ključ tablice stud)

Kriterij za određivanje redosljeda spajanja: veličina međurezultata

$\Rightarrow$ (mjesto $\triangleright\triangleleft$ r_1) $\triangleright\triangleleft$ r_2

Optimiranje upita - analiza plana obavljanja PostgreSQL

Svaki korisnik koji pokrene SQL naredbu može doznati koji je plan obavljanja upotrijebljen pri obavljanju naredbe.

Korisnik postavlja zahtjev SUBP-u da za SQL naredbu ispiše plan obavljanja naredbom:

`EXPLAIN [ ANALYZE ] [ VERBOSE ]` ***`statement`***

`ANALYZE` – pored procjene troška i odabira plana izvođenja, obavlja naredbu, prikazuje vrijeme izvođenja i druge statističke podatke (procijenjeni broj n-torki, planirano i stvarno vrijeme obavljanja u ms,...)

`VERBOSE` – prikazuje dodatne informacije o planu izvođenja (npr. listu izlaznih stupaca za svaki čvor u stablu izvođenja)

Optimiranje upita - analiza plana obavljanja PostgreSQL

Analiziraju se **troškovi obavljanja upita** (u apstraktnim jedinicama koje mogu poslužiti za procjenu relativne uspješnosti jednog plana obavljanja u odnosu na drugi plan):

– početni trošak (*start-up*) – ostvaren prije dohvata prve n-torke (npr. kod pretraživanja po indeksu to je trošak čitanja indeksnih stranica da bi se dohvatila prva n-torka)

- trošak izvođenja (*run*) – trošak dohvaćanja svih n-torki
- ukupan trošak (*total*) – suma početnog i troška izvođenja

Ispisuje se:

- Procijenjeni i stvarni trošak obavljanja upita – početni i ukupni trošak
- Procjenjeni i stvarni broj n-torki koje se evaluiraju kao rezultat
- Redoslijed pristupa tablicama, način pristupa pojedinoj tablici
  - **Seq Scan** bez upotrebe indeksa
  - **Index Scan** pomoću indeksa, dohvaćaju se stranice s podacima
  - **Index Only Scan** pomoću indeksa, ne dohvaćaju se stranice s podacima
- Uvjeti selekcije (Filter), uz informaciju da li se selekcija obavlja uz pomoć indeksa
- Planirano i stvarno vrijeme obavljanja upita

1. primjer plana obavljanja – PostgreSQL

Za tablicu student je definiran primarni ključ i pripadni indeks (automatski).

```
ALTER TABLE student ADD CONSTRAINT pkStudent PRIMARY KEY(JMBAG);
EXPLAIN ANALYZE VERBOSE
SELECT *
  FROM student
 WHERE JMBAG BETWEEN '0555000443' AND '0555000950'
```

```
QUERY PLAN
Index Scan using pkstudent on public.student  (cost=0.28..11.14 rows=46 width=51)
          (actual time=0.039..0.045 rows=46 loops=1)
 Output: jmbag, imestudent, prezimestudent, oib, spol, datumrod, pbrrodstudent,
          pbrstanstudent
 Index Cond: (((student.jmbag)::text >= '0555000443'::text) AND ((student.jmbag)::text
          <= '0555000950'::text))
Planning Time: 0.097 ms
Execution Time: 0.058 ms
```

1. primjer plana obavljanja – PostgreSQL

```
Index Scan using pkstudent on public.student  (cost=0.28..11.14 rows=46 width=51)
         (actual time=0.039..0.045 rows=46 loops=1)
```

- Procjena troška: (cost=0.28..11.14 rows=46 width=51).
  - cost=0.28..11.14
    procjenjuje da će inicijalni trošak za obavljanje ove operacije iznositi 0.28,
    a da će ukupni trošak iznositi 11.14 apstraktnih jedinica.
  - rows=46
    procjenjuje da će u rezultatu upita biti 46 n-torki
  - width=51
    procjenjuje veličinu n-torke rezultata u bajtima
- Stvarni trošak (u ms): (actual time=0.039..0.045 rows=46 loops=1)
  - actual time=0.039..0.045 stvarno vrijeme izvođenja upita (0.045/11.14= 0.004039 ms/jedinici troška
  - loops=1 znači da je Index Scan obavljen jedan put
  - rows=46 znači da rezultat sadrži 46 n-torki

2. primjer plana obavljanja – PostgreSQL

Za tablice ***predmetGrupa*** i ***predmet*** nisu definirana integritetska ograničenja niti indeksi.

```
EXPLAIN ANALYZE VERBOSE
 SELECT *
   FROM predmetGrupa
   NATURAL JOIN predmet
  WHERE akGodina = 2016
```

```
QUERY PLAN
Hash Join  (cost=2.49..19.09 rows=171 width=54) (actual time=0.040..0.120 rows=171
           loops=1)
 Hash Cond: (predmetgrupa.sifpredmet = predmet.sifpredmet)
 ->  Seq Scan on public.predmetgrupa  (cost=0.00..16.13 rows=171 width=18) (actual
     time=0.013..0.063 rows=171 loops=1)
        Filter: (predmetgrupa.akgodina = 2016)
        Rows Removed by Filter: 639
 ->  Hash  (cost=1.66..1.66 rows=66 width=40) (actual time=0.023..0.023 rows=66 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 13kB
          ->  Seq Scan on predmet  (cost=0.00..1.66 rows=66 width=40) (actual
              time=0.005..0.011 rows=66 loops=1)
Planning time: 0.159 ms
Execution time: 0.139 ms
```

2. primjer plana obavljanja – PostgreSQL

- PostgreSQL gradi stablastu strukturu u kojoj za svaki čvor prikazuje plan izvođenja i (eventualno) statističke podatke. U planu se uz čvor prikazuje strelica (**->**) . Korijen je izuzetak.
- Stablo za plan izvođenja prethodnog upita:

2	`->  Hash  (cost=1.66..1.66 rows=66 width=40) (actual time=0.023..0.023 rows=66 loops=1)` `Buckets: 1024  Batches: 1  Memory Usage: 13kB`
1	`->  Seq Scan on predmet  (cost=0.00..1.66 rows=66 width=40) (actual time=0.005..0.011 rows=66 loops=1)`

- Plan treba čitati od dna (listova stabla) prema vrhu (korijenu):

1. Slijedno (Seq Scan) se jedan put (loops=1) čita tablica predmet. Procjenjuje se da će se u međurezultatu nalaziti 66 n-torki (rows=66) što se doista i dogodi.
2. Rezultat prethodnog koraka se pohrani u Hash tablicu u memoriji. Ako je Batches > 1 koristi se i sekundarna memorija (magnetski disk) jer nema dovoljno primarne memorije za međurezultat.

2. primjer plana obavljanja – PostgreSQL

4	Hash Join (cost=2.49..19.09 rows=171 width=54) (actual time=0.040..0.120 rows=171 loops=1) Hash Cond: (predmetgrupa.sifpredmet = predmet.sifpredmet)
3	-> Seq Scan on public.predmetgrupa (cost=0.00..16.13 rows=171 width=18) (actual time=0.013..0.063 rows=171 loops=1) Filter: (predmetgrupa.akgodina = 2016) Rows Removed by Filter: 639

3. Slijedno (Seq Scan) se jedan put (loops=1) čita tablica predmetGrupa. Primjenjuje se filter iz upita.
4. Rezultat prethodnog koraka se korištenjem Hash Join metode "spaja" s Hash tablicom u memoriji koja je izgrađena za n-torke iz predmet. Obavlja se faza ispitivanja.
   Uvjet spajanja je prikazan pod Hash Cond.
   Primijetite da je za vrijeme obavljanja Hash Join-a procijenjena veličina rezultata u bajtima width=54 dok je u koraku 3 iznosila 18 bajta, a u koraku 1 je iznosila 40 bajta. Zbog čega nije 58 bajta nego 54?