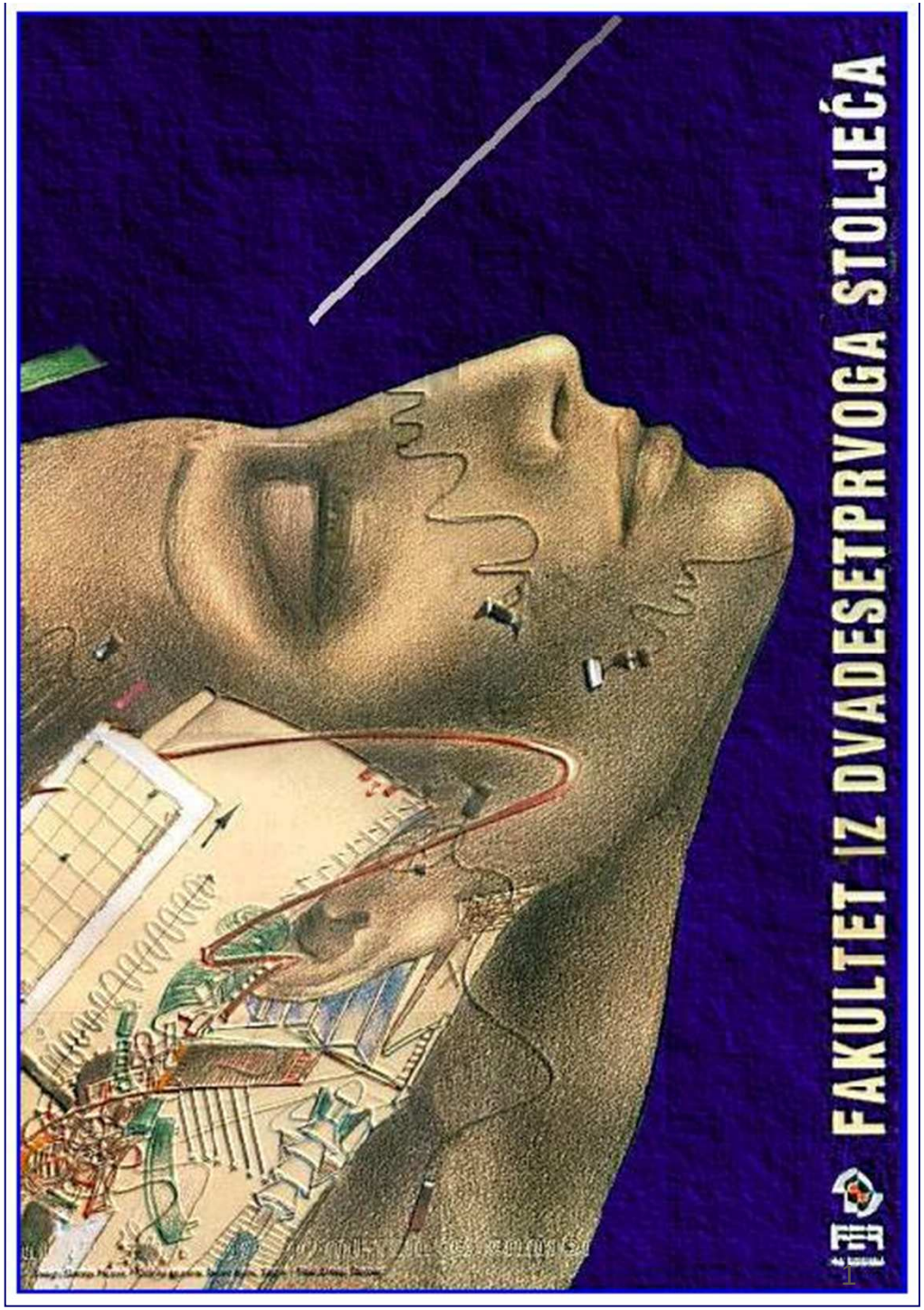
Baze podataka

Predavanja

Svibanj 2013.

13. Optimiranje upita



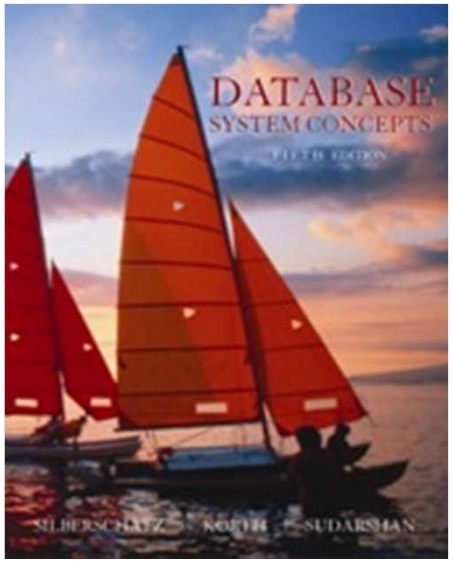
Obavljanje upita (*engl. Query Processing*)

- 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:

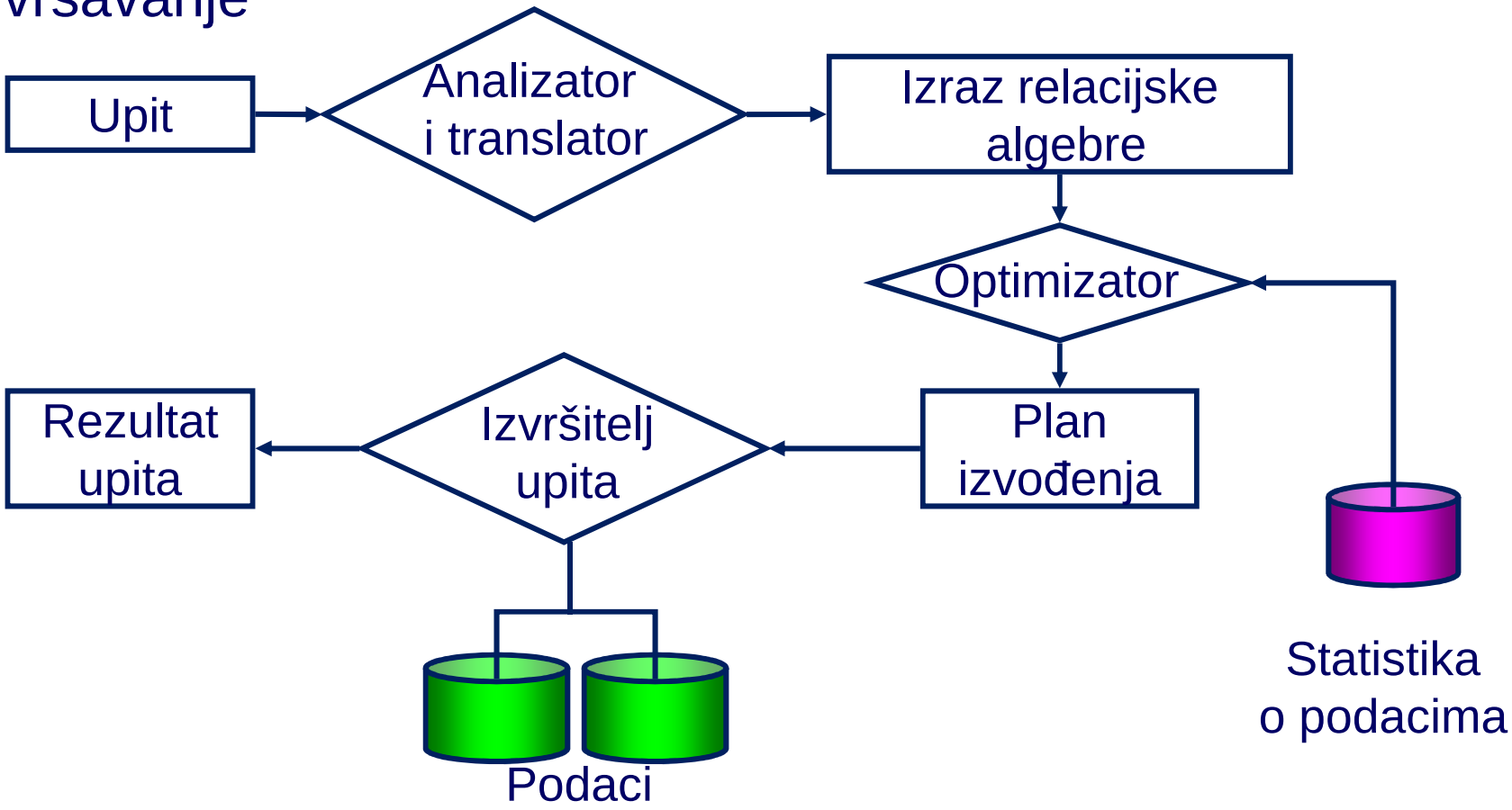
A. Silberschatz, H. Korth, S. Sudarshan,

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 relacijama koje sudjeluju u upitu.

- To su npr.
  - N(r) - broj n-torki u relaciji
  - informacija o tome da li atribut sadrži jedinstvene vrijednosti
  - koji indeksi su izgrađeni nad relacijom, 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 relaciji r
    - $V(A, r) = \mathcal{G}_{COUNT(*)}(\pi_A(r))$
    - A može biti skup atributa
  - broj jedinstvenih vrijednosti u indeksu
  - distribucija vrijednosti u pojedinim atributima relacije

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
    - IBM IDS: administrator obavlja naredbu: **`UPDATE STATISTICS`**
- Odgovornost je administratora baze podataka da uvijek kada se bitno promijene podaci koji bi mogli utjecati na rad optimizatora podataka (npr. u relaciju 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

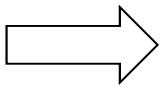
- Pretvorba upita pisanog u upitnom jeziku (SQL) u izraz relacijske algebre
- Zamjena virtualnih relacija s temeljnim relacijama koje proizlaze iz definicije virtualne relacije

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}

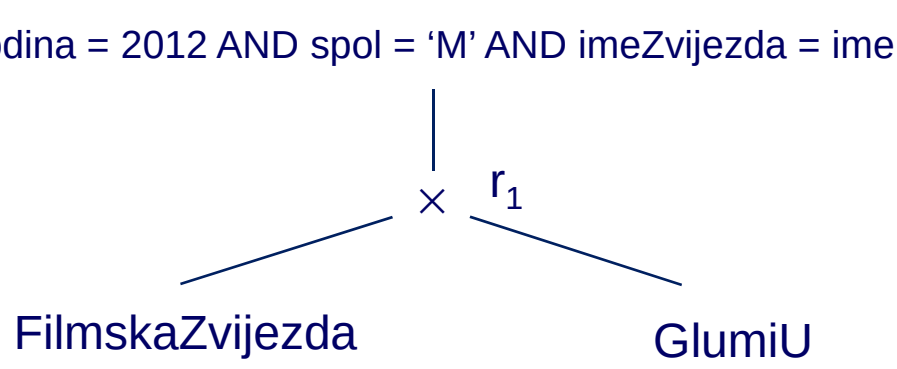
GLUMIU = {naslov, godina, imeZvijezda} K_{GLUMIU} = {naslov, godina, imeZvijezda}

```
SELECT *
  FROM FilmskaZvijezda, GlumiU
 WHERE imeZvijezda = ime
   AND godina = 2012
   AND spol = 'M'
```

izraz relacijske algebre:

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

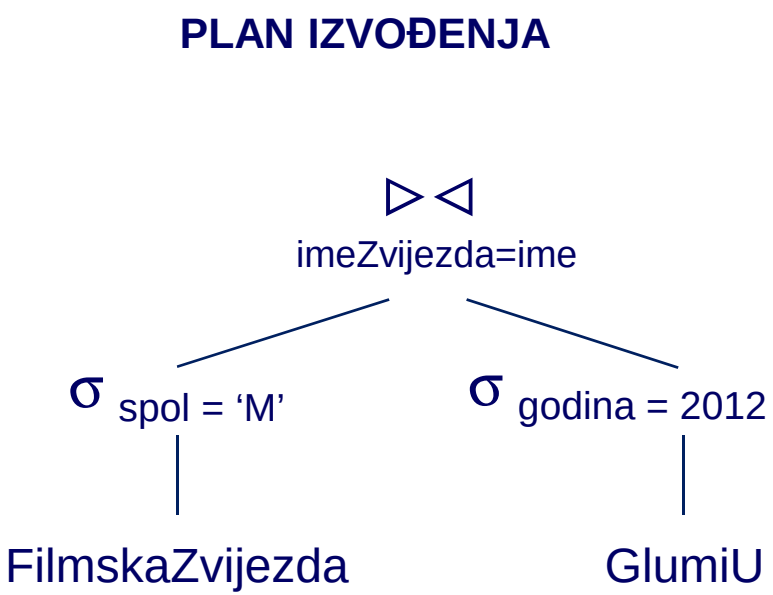
- stablo upita (*query tree*) je način prikaza izraza relacijske algebre
- listovi stabla su relacije, a ostali čvorovi su operacije relacijske algebre



card (r_1) = card(FilmskaZvijezda)*card(GlumiU)

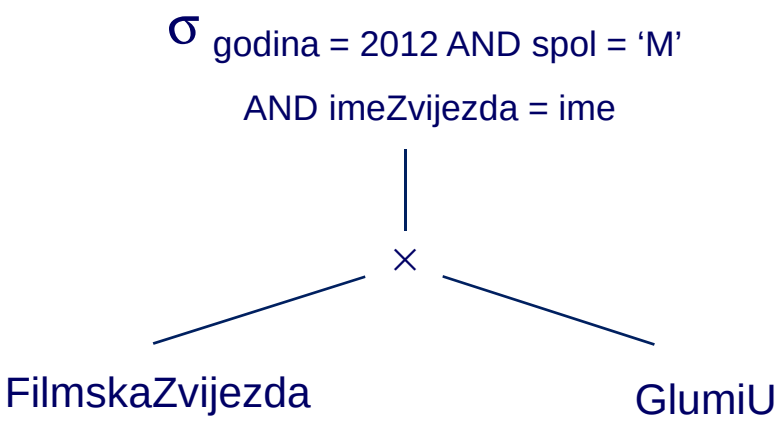
Plan izvođenja (engl. Execution Plan)

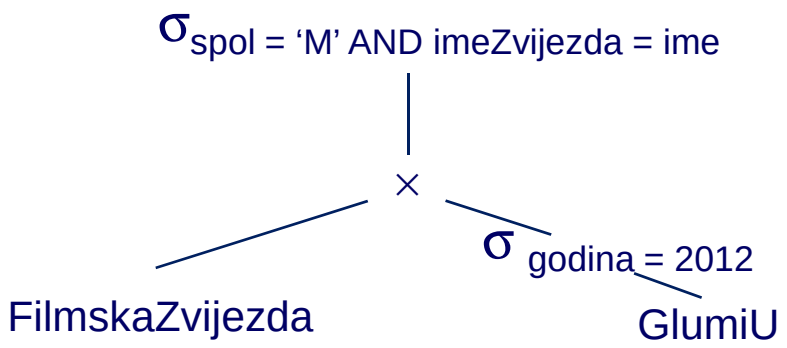
- Odabir operanada i operacija (σ, ⋈, …)
- Redoslijed izvođenja operacija
- Detaljni plan izvođenja operacija
  - metode pristupa podacima
  - redoslijed spajanja
  - metode spajanja
  - stvaranje privremenih indeksa?
  - sortiranje privremenih rezultata
  - ….

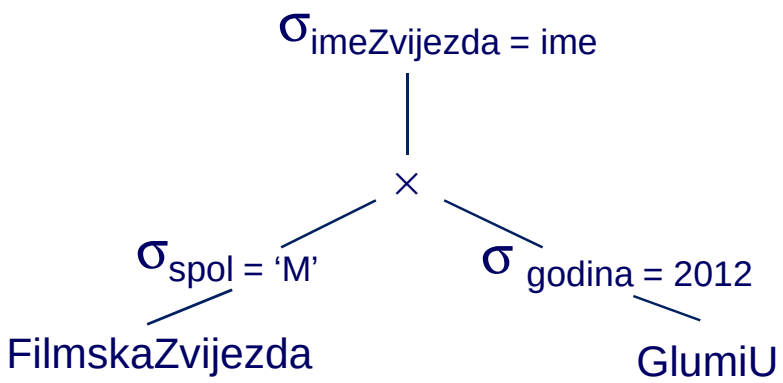


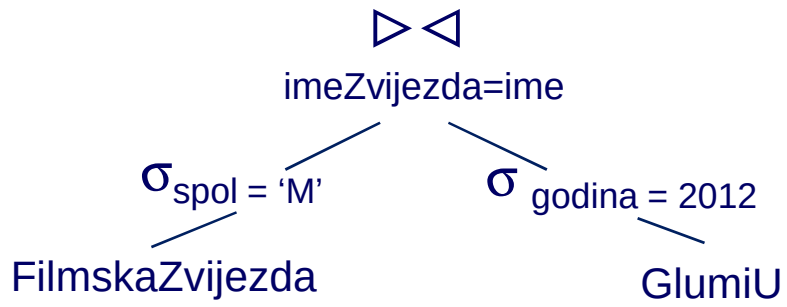
Plan izvođenja nije jednoznačno određen upitom

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



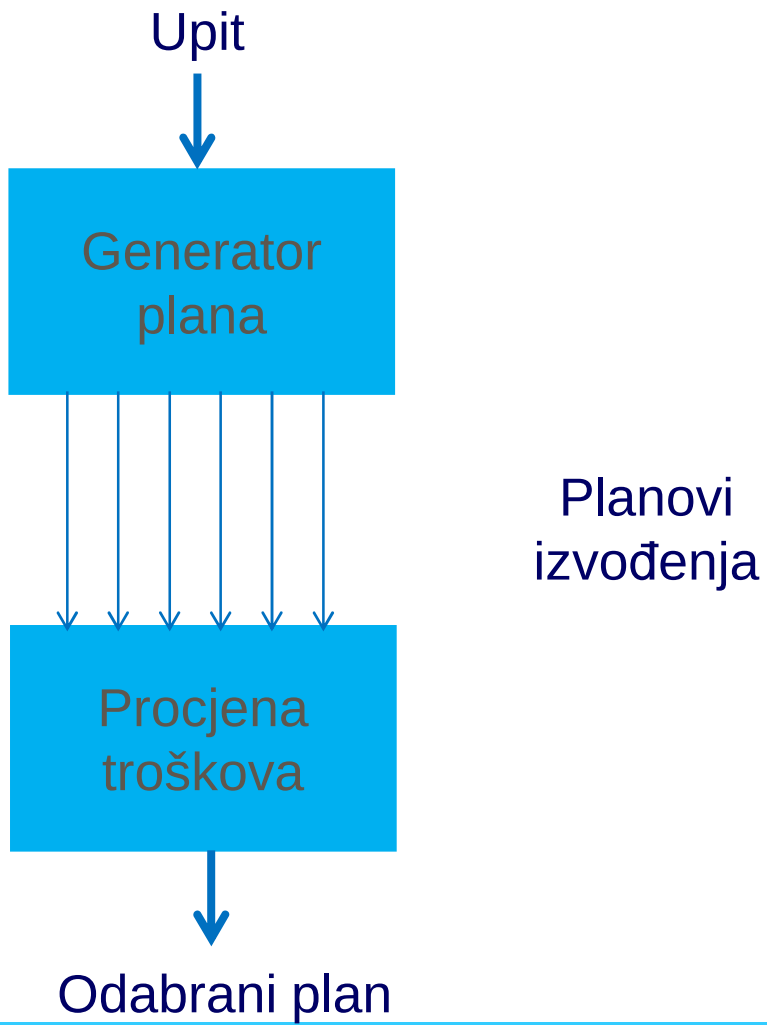






Optimiranje upita

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 ekvivalentni ako primijenjeni 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: relacije **r(A,B)**, **s(B,C)**, **t(C,D)**

  $(r \underset{r.B > s.B}{\bowtie} s) \underset{A < D}{\bowtie} t \neq r \underset{r.B > s.B}{\bowtie} (s \underset{A < D}{\bowtie} t)$ ➔ **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:**

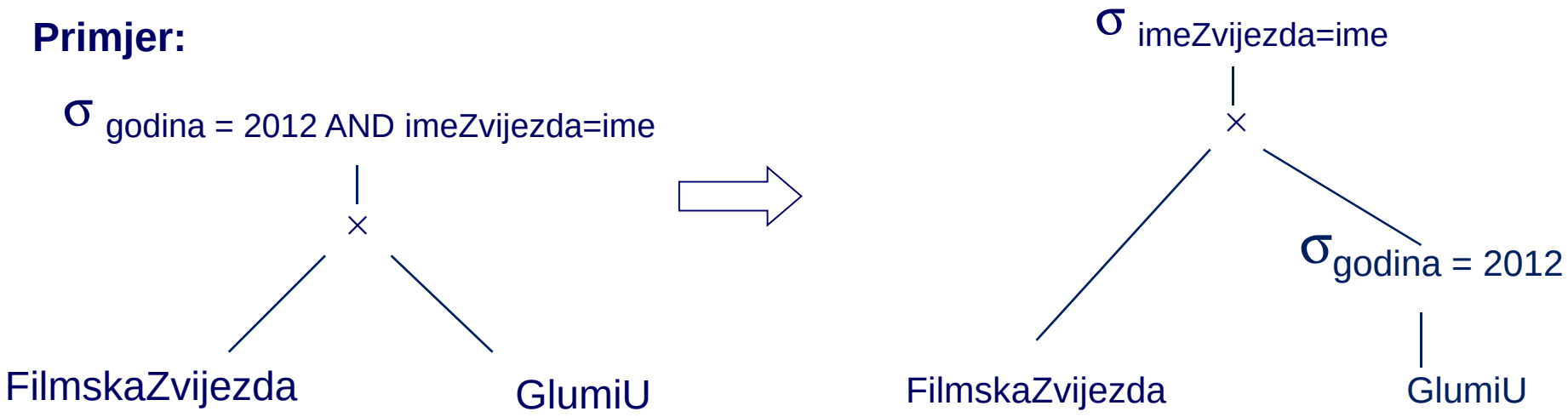
    $\sigma_C(r \bowtie s) = (\sigma_C(r)) \bowtie (\sigma_C(s))$
- Na isti se način selekcija može potiskivati u odnosu na Kartezijev produkt i spajanje uz uvjet
- U ovim predavanjima su navedena samo neka od pravila algebarskih transformacija (projekcija, grupiranje,…)

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:

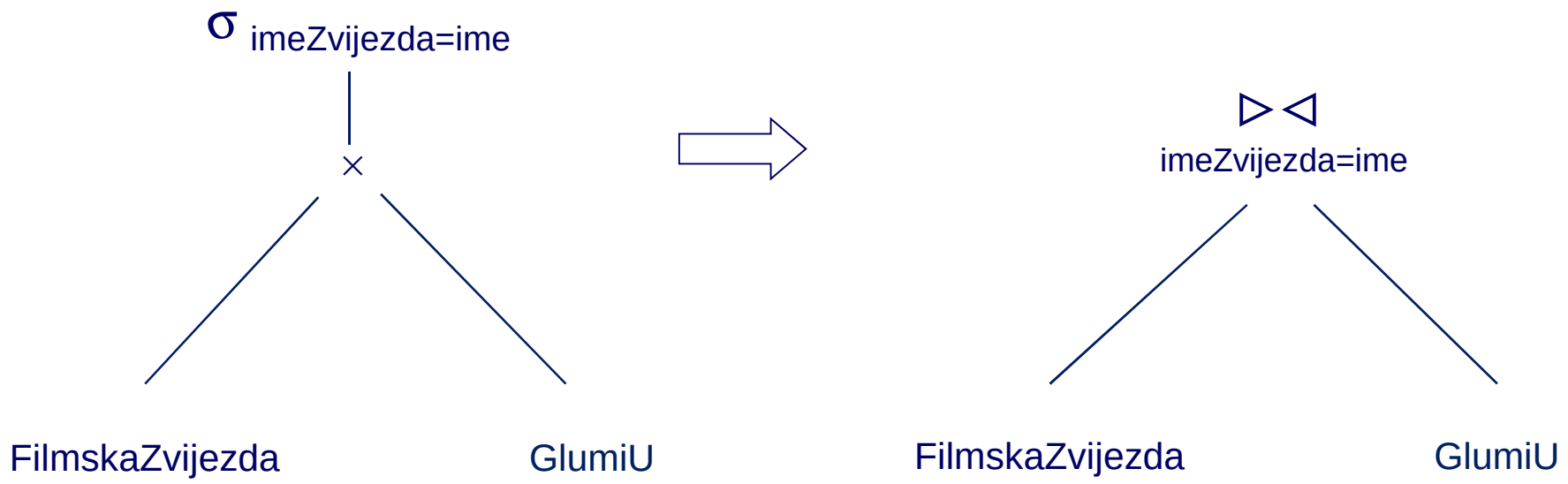


- **Potiskivanje selekcije je jedno od heurističkih pravila koja se koriste pri optimizaciji.**
- **Heuristička pravila**
  - **proizlaze iz iskustva, temelje se na ekvivalentnim izrazima relacijske algebre**
  - **njihova primjena u većini slučajeva rezultira efikasnijim planom izvršavanja**

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 ⋈ s ⋈ t

- optimizator ne raspolaže informacijom o ključevima relacija

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

   N (r ⋈ s) = 1000 × 2000 / max(20, 50) = 40.000

   V (C, r ⋈ s) = 100

   N ((r ⋈ s) ⋈ t) = 40.000 × 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 relaciji (*access plan*)

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

Metode pristupa podacima u relaciji

Slijedno čitanje podataka iz relacije (*table 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 relacije

```
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)
```

- **key-only index scan**
  ako su svi podaci koji se čitaju iz relacije 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 relacija (*join*)

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

Spajanje ugniježđenim petljama

Postupak:

- relacije koje se spajaju se nazivaju vanjska relacija (outer table) i unutarnja relacija (inner table). (Ovi pojmovi nemaju veze s vanjskim spajanjem (outer join) u relacijskoj algebri!)
- iz vanjske relacije 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 relacije koristi se index scan ili key-only index scan
- za svaku pročitanu n-torku iz vanjske relacije traže se n-torke iz unutarnje relacije koje zadovoljavaju uvjet spajanja

Spajanje ugniježđenim petljama

SUBP mora za svaku n-torku iz vanjske relacije po jednom pretražiti cijelu unutarnju relaciju

Način pristupa podacima iz unutarnje relacije:

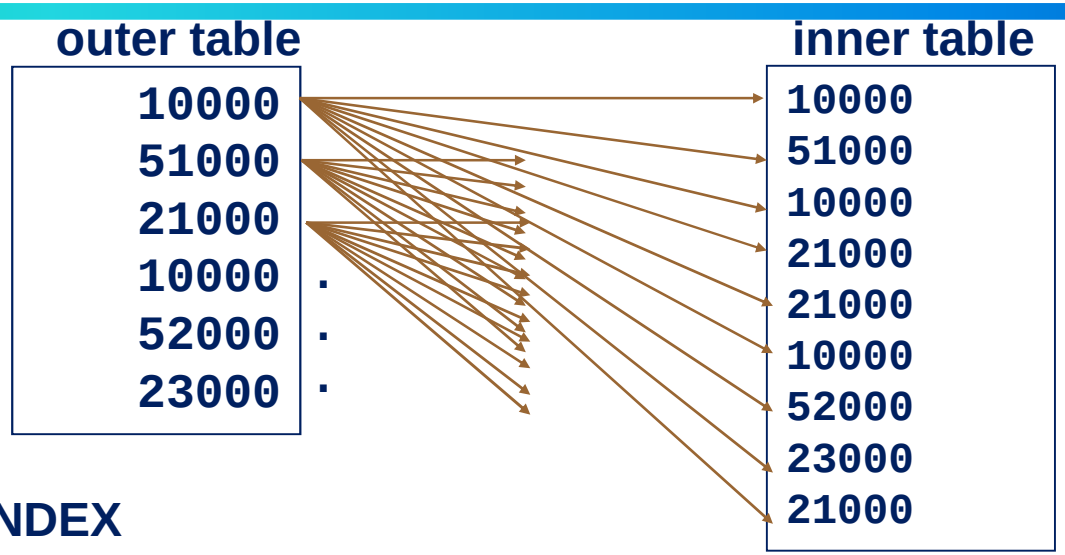
a) ako se u unutarnjoj relaciji nalazi mali broj n-torki može se koristiti slijedno pretraživanje

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

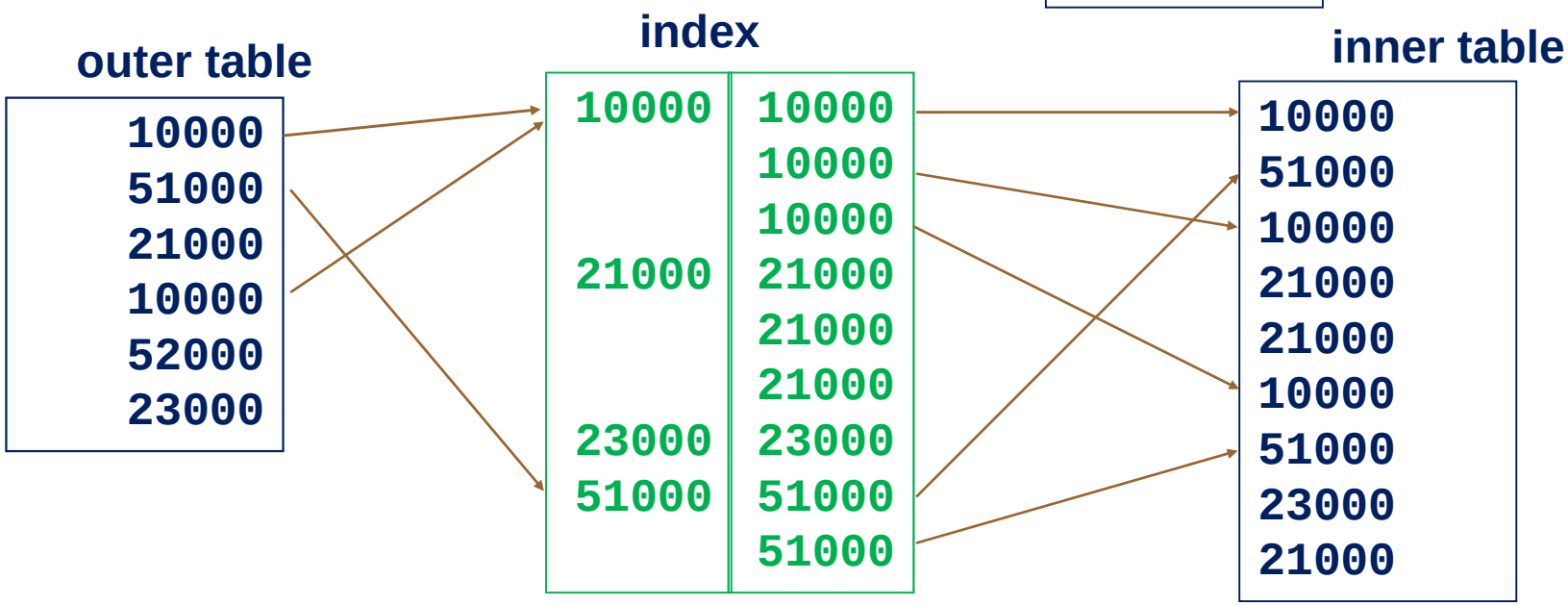
c) ukoliko upotrebljivi indeks za pretragu unutarnje relacije ne postoji, a relacija sadrži veliki broj n-torki, optimizator može procijeniti da je isplativije potrošiti određeno vrijeme na izgradnju privremenog indeksa nego veliki broj puta slijedno pretraživati relaciju. U planu obavljanja se takav način pristupa označava kao **autoindex-path**

Spajanje ugniježđenim petljama

SLIJEDNO



INDEX ili AUTO-INDEX



Raspršeno spajanje

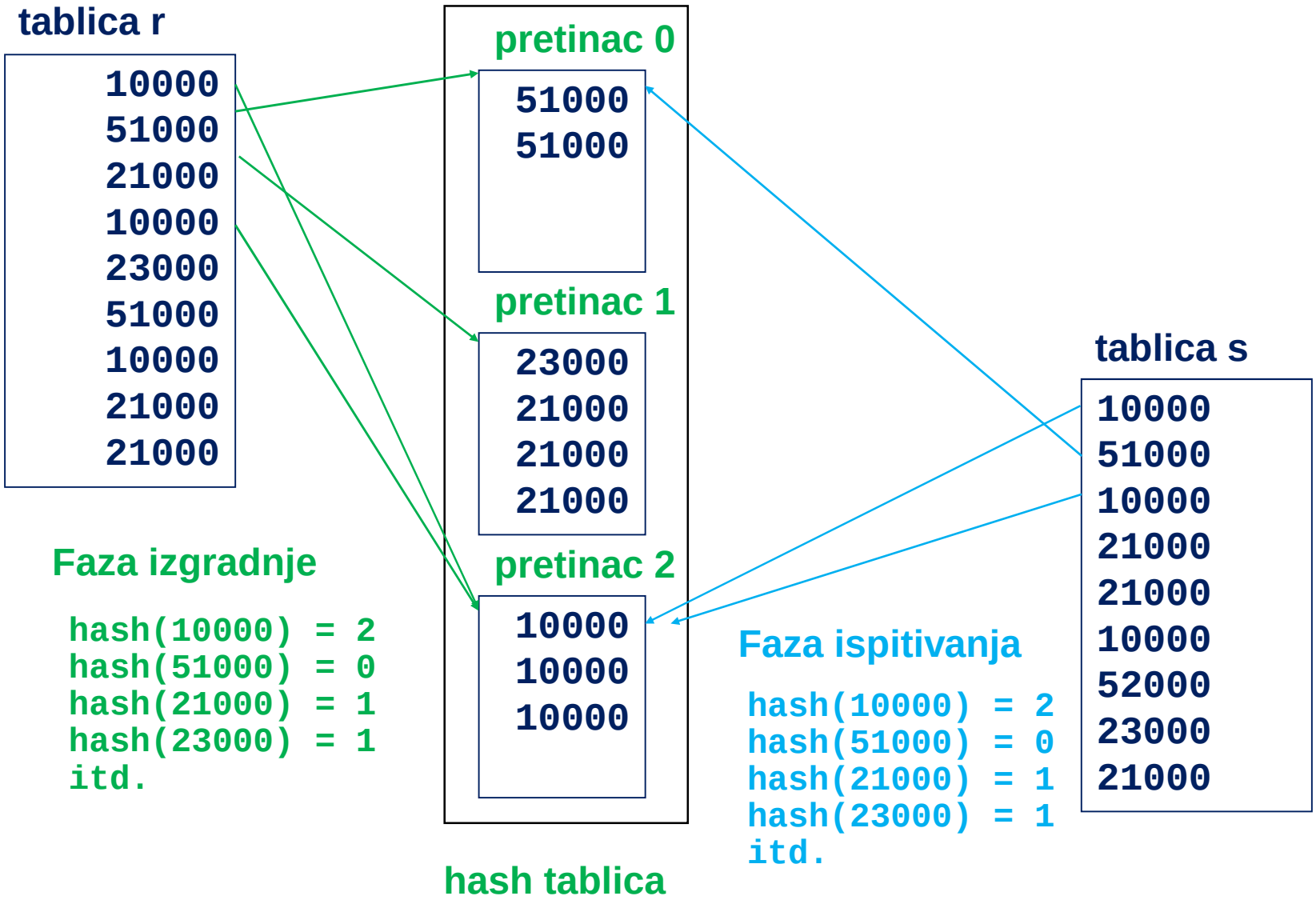
Faza izgradnje:

- Raspršena tablica se izgrađuje za samo jednu relaciju iz para kojeg treba spojiti (npr. relaciju r iz para relacija 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 relacije 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 relacije kod spajanja s ugniježđenom petljom.

Faza ispitivanja:

- Čita sadržaj relacije **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 relacije **s** prema kojima se obavlja spajanje, izračuna se u kojem se džepu raspršene tablice nalaze zapisi iz relacije **r** s kojima se ta n-torka treba spojiti.

Raspršeno spajanje



Primjer

Zadane su relacije *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

Indeks je kreiran samo nad atributom *prezStud* relacije *stud*.

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

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

Primjer

Za upit

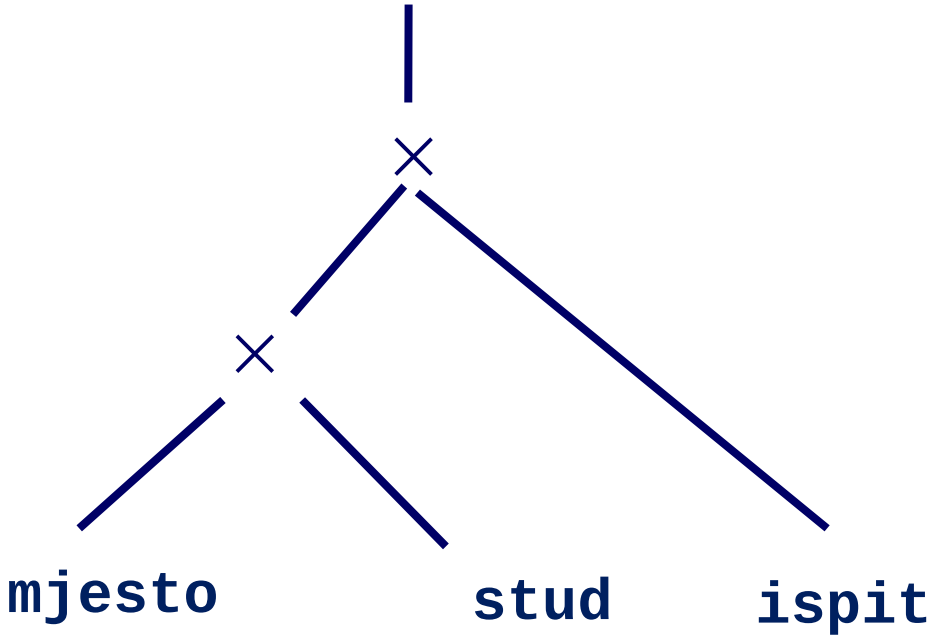
```
SELECT * FROM mjesto, stud, ispit
   WHERE mjesto.pbr = stud.pbr
     AND stud.mbrStud = ispit.mbrStud
     AND stud.prezStud = 'Horvat'
     AND ispit.ocjena = 5
```

a) Nacrtati stablo upita za početni plan izvođenja upita pri čemu je redoslijed spajanja relacija određen redoslijedom kojim su relacije 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)

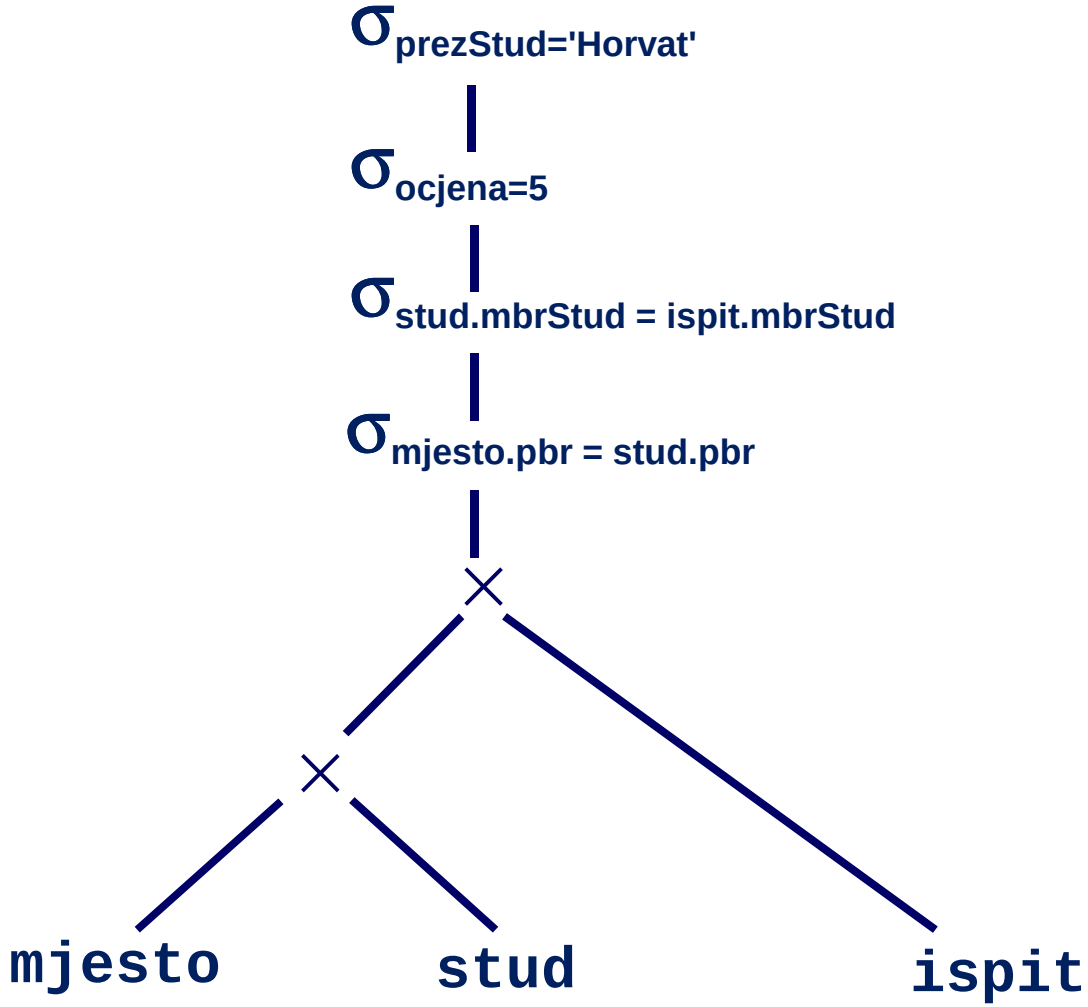
```
SELECT * FROM mjesto, stud, ispit
   WHERE mjesto.pbr = stud.pbr
     AND stud.mbrStud = ispit.mbrStud
     AND stud.prezStud = 'Horvat'
     AND ispit.ocjena = 5
```

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



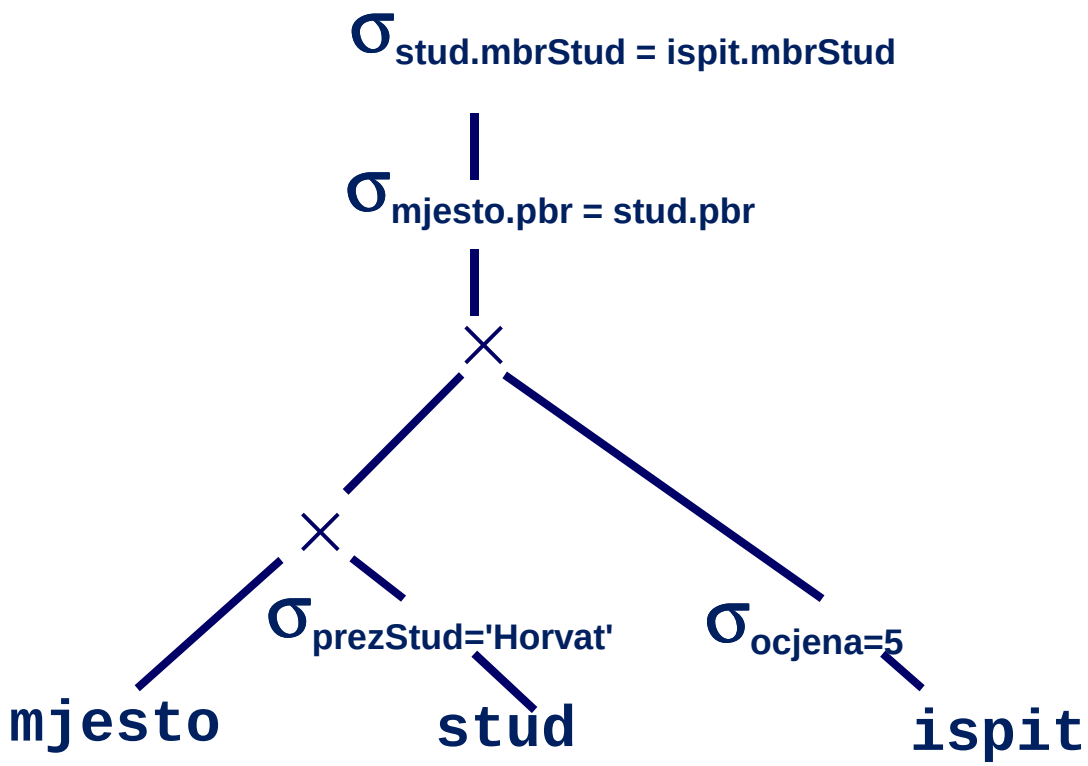
Primjer – b)

Rastavljanje uvjeta selekcije:



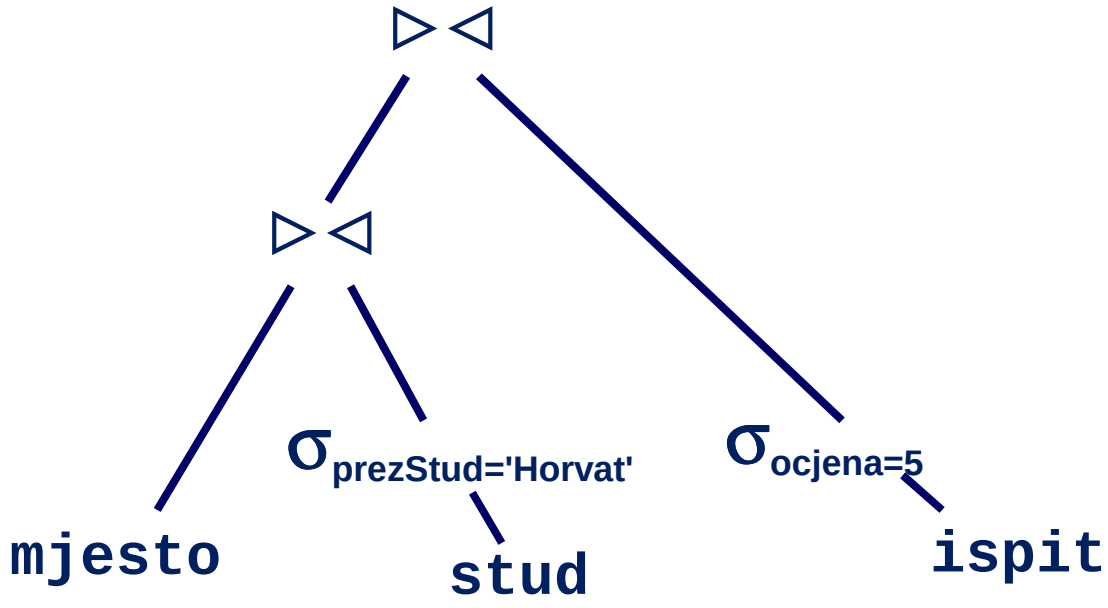
Primjer – b)

Potiskivanje selekcije:



Primjer – b)

Kombiniranje operacije selekcije i Kartezijevog produkta



Primjer – b)

Procjena broja n-torki u međurezultatima:

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

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

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

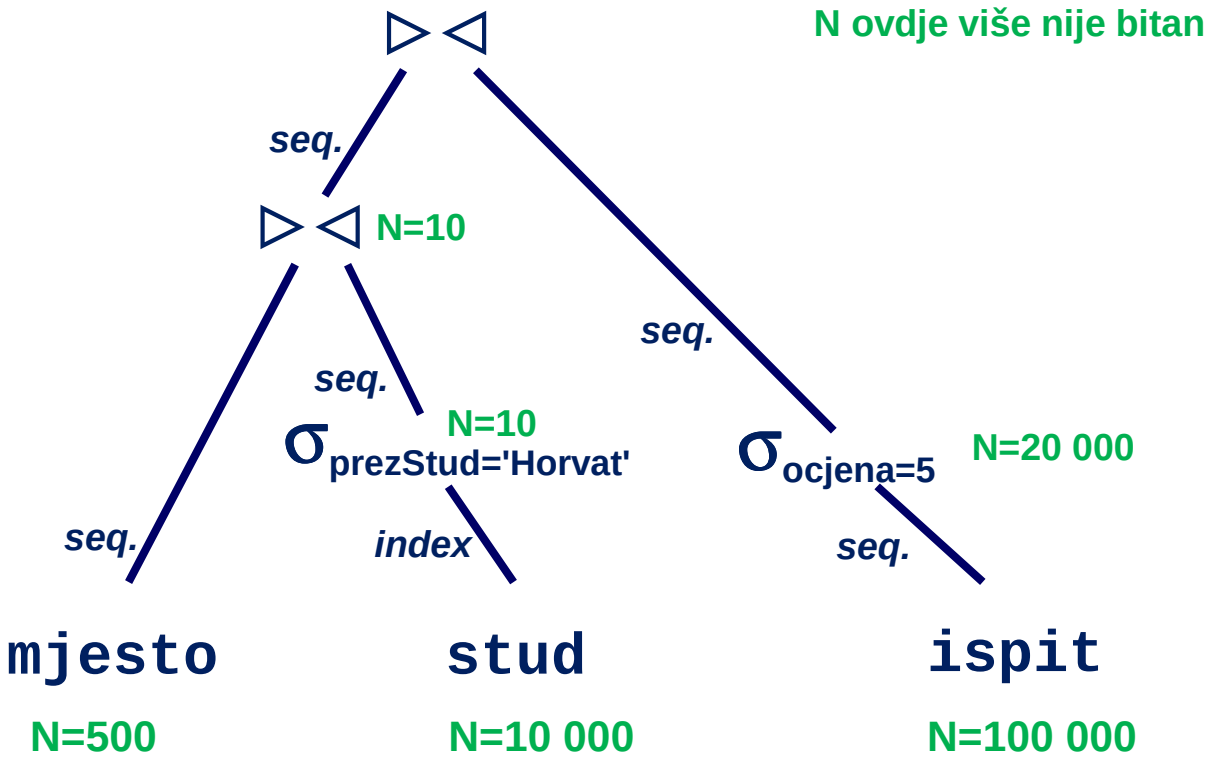
$N(r_2) = N(ispit) / V(ocjena, ispit) = 100000/5 = 20000$

$r_3 = mjesto \triangleright\triangleleft r_1$

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

Primjer – b)

Korištene metode pristupa podacima



Primjer – c)

Procjena broja n-torki u međurezultatu za različite redoslijede spajanja

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

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

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

Kriterij za određivanje redosljeda spajanja: veličina međurezultata

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

Optimiranje upita - analiza plana obavljanja IBM Informix

Svaki korisnik koji pokrene SQL naredbu može doznati koji je plan obavljanja upotrijebljen za obavljanje naredbe.

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

SET EXPLAIN ON

SUBP će prestati ispisivati plan obavljanja nakon što se obavi naredba:

SET EXPLAIN OFF

Optimiranje upita - analiza plana obavljanja
IBM Informix

Nakon što se obavi naredba SET EXPLAIN ON, plan obavljanja svake sljedeće naredbe će biti zapisan u datoteku **sqexplain.out** u *home* kazalu korisnika na računalu na kojem se nalazi SUBP.

Npr. ako ste na bazu **studadmin** spojeni kao korisnik **bpadmin**, datoteka će se nalaziti u kazalu **/usr/bpadmin/** (home kazalo korisnika **bpadmin** na virtualnom linux računalu)

sqexplain.out je obična tekstualna datoteka čiji sadržaj možete pregledati upotrebom naredbe **cat**, npr. "cat sqexplain.out | more" ili u nekom od instaliranih editora (npr. **vi** ili **joe**)

Zapisi se u ovu datoteku nadopunjuju, tako da opisi starih planova obavljanja ostaju sačuvani.

Optimiranje upita - analiza plana obavljanja IBM Informix

- Sadržaj SQL naredbe (prijepis zadane SQL naredbe)
- Procjenu troška obavljanja upita (u apstraktnim jedinicama koje mogu poslužiti za procjenu relativne uspješnosti jednog plana obavljanja u odnosu na drugi plan)
- Procjenu broja n-torki koje se evaluiraju kao rezultat
- Redoslijed pristupa relacijama
- Način pristupa pojedinoj relaciji
  - **SEQUENTIAL SCAN** - bez upotrebe indeksa
  - **INDEX PATH** - pomoću jednog ili više indeksa
  - **AUTOINDEX PATH** - kreira se privremeni indeks
- Popis atributa koji se koriste za uvjet selekcije, uz informaciju da li se selekcija obavlja uz pomoć indeksa

Sadržaj plana obavljanja – IBM Informix

Oznaka **DYNAMIC HASH JOIN** označava da se na prethodno ispisanom paru relacija (ili paru prethodno dobivenih međurezultata) obavlja raspršeno spajanje.

Oznaka (Build outer) u dijelu specifikacije koja se odnosi na **DYNAMIC HASH JOIN** označava da se raspršena tablica gradi za n-torke iz prve relacije (ili međurezultata) iz para. U suprotnom, raspršena tablica se gradi za n-torke iz druge relacije (ili međurezultata) u paru.

Ako se ne spominje **HASH JOIN**, radi se o spajanju ugniježdenim petljama (nested loop join). Ako se za pristup drugoj relaciji koristi:

- **INDEX PATH** - za spajanje se koristi postojeći indeks,
- **AUTO-INDEX PATH** - treba razmisliti treba li kreirati takav indeks, kako se privremeni indeks ne bi kreirao svaki put kad se obavlja ovaj upit.

Primjer plana obavljanja – IBM Informix

```
QUERY:
------
SELECT * FROM ispit
             , stud
             , mjesto
       WHERE mjesto.pbr = stud.pbrStan
         AND stud.mbrStud = ispit.mbrStud

Estimated Cost: 482
Estimated # of Rows Returned: 2960

1) stud: SEQUENTIAL SCAN

2) mjesto: INDEX PATH
    (1) Index Keys: pbr
        Lower Index Filter: mjesto.pbr = stud.pbrStan

3) ispit: SEQUENTIAL SCAN
   DYNAMIC HASH JOIN (Build Outer)
   Dynamic Hash Filters: stud.mbrStud = ispit.mbrStud
```

Primjer plana obavljanja – IBM Informix

Procijenjeni trošak: 482
Procijenjeni broj n-torki u rezultatu: 2960

1) stud: SEQUENTIAL SCAN - slijedno se čitaju sve n-torke iz relacije stud, jer ne postoji nikakav uvjet selekcije

2) mjesto: INDEX PATH
(1) Index Keys: pbr
Lower Index Filter:mjesto.pbr = stud.pbrStan

relacije stud i mjesto spajaju se metodom ugniježđene petlje. Pri tome se za svaku n-torku iz stud, pomoću indeksa dohvaćaju odgovarajuće n-torke u relaciji mjesto

3) ispit: SEQUENTIAL SCAN
DYNAMIC HASH JOIN (Build Outer)
Dynamic Hash Filters:stud.mbrStud=ispit.mbrStud

Za rezultat iz koraka 2) se napravi raspršena tablica. Slijedno se čitaju zapisi iz relacije ispit (jer ne postoji nikakav uvjet) te se preko hash funkcije dolazi do odgovarajućih zapisa iz koraka 2)