Optimalizácia dopytov

Kapitola 13

Optimalizácia dopytov

- Aj pri jednoduchých dopytoch máme viacero možností ako dostať výsledok.
- Z dopytu sa vytvorí výraz relačnej algebry / kalkulu (System R)
- Vygenerujú sa viaceré **alternatívne plány** výpočtu dopytu
- Odhadne sa cena každého vypočítaného plánu a vyberie sa ten s najlepšou cenou
    - Pre odhad ceny používajú informácie zo systémového katalógu a znalosť fungovania algoritmov výpočtu operátorov
- Túto činnosť vykonáva **optimalizátor** dopytov
- Ideál: Chceme nájsť najlepší plán.
- Prakticky: Chceme sa vyhnúť najhorším plánom

Plán vyhodnotenia dopytov

- Skladá sa z rozšíreného stromu relačnej algebry.
- Tento strom naznačuje v akom poradí budú vykonané operácie.
- Musí sa rozhodnúť ako budú implementované tieto operácie.
- Ak je to možné, tak výsledok jednej operácie je posunutý ďalšej operácii bez uloženia medzivýsledkov.

Odhad ceny a veľkosti výsledku

- Pre každú implementáciu operácie v pláne aj pre celý uvažovaný plán:
  - Musíme odhadnúť *cenu* každej operácie v strome plánu.
    - Závisí od veľkosti vstupu a spôsobu uloženia, ale aj od distribúcie uložených hodnôt.
    - Predpokladáme nezávislosť stĺpcov
    - Známa funkčná závislosť by mohla výrazne spresniť odhad
    - Dá sa to niekedy zapísať cez CHECK obmedzenie, ale zďaleka nie všetky závislosti
  - Musíme odhadnúť *veľkosť výsledku* pre každú operáciu v pláne
    - Využitie informácii pre nasledujúce relácie.

Odhad veľkosti výsledku – Redukčný faktor

- Redukčný faktor je pomer odhadovanej výslednej veľkosti k vstupnej veľkosti
- Pre rôzne prípady sa počíta rôzne:

- stĺpec = hodnota
  - $\frac{1}{\#kľúčov(I)}$ * N; ak máme index s kľúčom stĺpec
  - $\frac{1}{10}$ * N; ak nevieme nič
- stĺpec1 = stĺpec2
  - $\frac{1}{\max(\#kľúčov(I1),\#kľúčov(I2))}$ * N; ak máme indexy na oba stĺpce
  - $\frac{1}{\#kľúčov(I1\ alebo\ I2)}$ * N; index iba na jednom zo stĺpcov
  - $\frac{1}{10}$ * N; inak
- stĺpec > hodnota
  - $\frac{max_I - hodnota}{max_I - min_I}$ * N; ak vieme index
  - $\frac{1}{2}$ * N; inak
- Stĺpec IN (zoznam hodnôt) – súčet odhadov typu stĺpec=hodnota, ale max $\frac{1}{2}$ * N
- Projekcie - pomer veľkosti záznamov po a pred => z typov atribútov

Odhad veľkosti výsledku – odhad selektivity (redukčneho faktoru)

- **Histogram** - odhad veľkosti výsledku pre vek>13

vek	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	sum
počet výskytov	2	3	3	1	2	1	2	8	4	2	0	1	3	4	9	45

bez histogramu – ((14-13)/15) *45 = 1/15 *45=3

vek	0-2	3 až 5	6 až 8	5 až 11	12 až 14
Priemerná početnosť	2,67	1,33	2,8	1,0	5,33
počet výskytov	8	4	14	3	16

ekvidištančný - (16/3) = 5,33

vek	0 až 4	5 až 7	8 až 11	12 až 13	14
Priemerná početnosť	2,2	3,66	1,75	3,5	9
počet výskytov	11	11	7	7	9

ekvipotenčný (podobný počet výskytov) - 9

Ekvivalencie relačnej algebry

- Dva výrazy relačnej algebry sú ekvivalentné, ak dajú rovnaký výsledok nad rovnakou množinou
- Umožňujú dosiahnuť lepšie evaluačné plány
- Umožňujú nám vybrať rôzne poradie joinov a vybrať selekcie a projekcie pred joiny.

• Selekcie:	$\sigma_{p_1 \wedge \ldots \wedge p_n}(R) = \sigma_{p_1}\left(\ldots\left(\sigma_{p_n}(R)\right)\right)$	Kaskádové
	$\sigma_{p_1}\left(\sigma_{p_2}(R)\right) = \sigma_{p_2}\left(\sigma_{p_1}(R)\right)$	Komutatívne
• Projekcie:	$\pi_{a_1,\ldots,a_n}(R) \equiv \pi_{a_1}\left(\ldots\left(\pi_{a_n}(R)\right)\right)$	Kaskádové
• Joiny:	$R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$	Asociatívne
	$(R \bowtie S) \equiv (S \bowtie R)$	Komutatívne
	$R \bowtie (S \bowtie T) \equiv (T \bowtie R) \bowtie S$	Ekvivalencia

Výpočet alternatívnych plánov

- Sú to rôzne vylepšenia pôvodného plánu
- Sú dva hlavné prípady:
  - Plány s jednou reláciou
  - Plány s viacerými reláciami
- Dopyty nad jednou reláciou pozostávajú z kombinácie selekcií, projekcií a agregačných operácií (join nie)
  - Uvažuje sa každá dostupná prístupová cesta (prechod súborom alebo indexom) a vyberie sa tá s najnižšou odhadovanou cenou.
  - Niekedy sa viaceré operácie sa dajú realizovať spolu (postupnosť operátorov prepojených ako producent a konzument)

Odhad cien pre plány s jednou reláciou

- Selekcia (už vieme spraviť odhad selektivity):
  - Použitie indexu I na unique stĺpci (alebo unique množine stĺpcov) :
    - *Cena je Hĺbka(I) pre B+strom, 1,2 pre hash index.*
  - Klastrovaný index I zhodujúci sa s jednou alebo viacerými časťami podmienky v konjunktívnom normálnom tvare:
    - *(NStránok(I)+NStránok (R)) * súčin selektivít zhodujúcich sa častí CNF podmienky.*
  - Neklastrovaný index I zhodujúci sa s jednou alebo viacerými časťami podmienky v konjunktívnom normálnom tvare:
    - *(NStránok(I)+NRiadkov(R)) * súčin selektivít zhodujúcich sa častí CNF podmienky.*
  - Sekvenčný prechod súboru:
    - *NStránok(R).*

Odhad cien pre plány s jednou reláciou

- Projekcia:
  - Typicky sekvenčný prechod riadkov alebo indexu, ak obsahuje všetky potrebné stĺpce
  - Cena sa typicky zvyšuje, ak máme DISTINCT a veľa riadkov na to, aby sa zmestili do pamäte a nezískali sme ich z indexu, kde na DISTINCT vieme použiť zotriedenie alebo delenie cez hash
    - Obvykle *NStránok(R) + 2 * NStránok(R) * redukčný faktor projekcie*
    - Podobný odhad je pre GROUP BY

Príklad

```
SELECT rating, count(x)
FROM Predavači
WHERE rating > 5 AND vek=20
GROUP BY rating
Having COUNT DISTINCT(meno)>2
```

- Plán dopytu:
  - $\pi_{rating,count}(x)(HAVING\ _{count\ distinct(meno)}>2(GROUP\ By_{rating}\ (\pi_{rating,meno}(\sigma_{rating>5\ \&\ vek=20}(Predavaci)))))$
- Predavači: 500 stránok, 80 riadkov na stránku, každý riadok má 50 bajtov
- Odhad selektivity 50% pre rating>5 a 10% pre vek=20, ak nemáme index
- Ak máme metadáta, nech odhadovaná selektivita pre rating>5 je 25% a pre vek=20 je 4%
- Nech projekcia $\pi_{rating,meno}$ má redukčný faktor 80%
- Odhadnime veľkosť výsledku a cenu plánu, ak máme
  - žiaden index,
  - (ne)klastrovaný index < vek, rating >,
  - (ne)klastrovaný index < rating, vek >,
  - (ne)klastrovaný index pre rating,
  - (ne)klastrovaný index pre vek,
  - dva neklastrované indexy pre rating a vek

Žiaden index

```
SELECT rating, count(x)
FROM Predavači
WHERE rating > 5 AND vek=20
GROUP BY rating
Having COUNT DISTINCT(meno)>2
```

- Selekcia cez table scan 500 I/O
  - Odhad výsledku: 50% rating, 10% vek, dokopy 5%, teda výledok: 25 stránok
- Projekcia na rating a meno spracúva priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku 80% zo vstupu: 20 stránok
- GROUP BY napríklad cez sort (podľa <rating, meno>), predpokladajme, že 20 stránok utriedime na 2 prechody 20W + 20R = 40 I/O
  - Odhad výsledku: na rating nemám index, predpokladám 10 rôznych hodnôt v tabuľke, 10/2 rôznych hodnôt pre rating >5 : 5 riadkov teda <1 stránka
- Agregácia priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku: <1 stránka

- Celková cena 540 I/O

Klastrovaný index <vek, rating>

```
SELECT rating, count(x)
FROM Predavači
WHERE rating > 5 AND vek=20
GROUP BY rating
Having COUNT DISTINCT(meno)>2
```

- Selekcia cez hľadanie v indexe
  - Odhad výsledku: 25% rating, 4% vek, dokopy 1%, teda výledok: 5 stránok (5*80 riadkov = 400 riadkov)
  - B+ strom: výška 3: 3 zhora dolu + 5*1,5 susedných listov = 11 I/O
  - Hash index: nepoužiteľný
- Projekcia na rating a meno spracúva priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku 80% zo vstupu: 4 stránky
- GROUP BY napríklad cez sort (podľa <rating, meno>) v RAM 0 I/O
  - Odhad výsledku: <1 stránka
- Agregácia priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku: <1 stránka

- Celková cena 11 I/O

```
SELECT rating, count(x)
FROM Predavači
WHERE rating > 5 AND vek=20
GROUP BY rating
Having COUNT DISTINCT(meno)>2
```

Neklastrovaný index <vek, rating>

- Selekcia cez index a externú štruktúru
  - Odhad výsledku: 25% rating, 4% vek, dokopy 1%, teda výledok: 5 stránok (5*80 riadkov = 400 riadkov)
  - B+ strom: výška 3: 3 zhora dolu + ~2 susedné listy + 400 prístupov do externej štruktúry (potrebujeme stĺpec meno) = 405 I/O
  - Hash index: nepoužiteľný
- Projekcia na rating a meno spracúva priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku 80% zo vstupu: 4 stránky
- GROUP BY napríklad cez sort (podľa <rating, meno>) v RAM 0 I/O
  - Odhad výsledku: <1 stránka
- Agregácia priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku: <1 stránka

- Celková cena 405 I/O

Klastrovaný index <rating, vek>

```
SELECT rating, count(x)
FROM Predavači
WHERE rating > 5 AND vek=20
GROUP BY rating
Having COUNT DISTINCT(meno)>2
```

- Selekcia cez index
  - Odhad výsledku: 25% rating, 4% vek, dokopy 1%, teda výledok: 5 stránok (5*80 riadkov = 400 riadkov)
  - B+ strom: výška 3: 3 zhora dolu + 25% listov (188 stránok) = 201 I/O
    - Listov je pri 67% zaplnení 750, z toho štvrtina je 188
  - Hash index nepoužiteľný
- Projekcia na rating a meno spracúva priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku 80% zo vstupu: 4 stránky
- GROUP BY napríklad cez sort (podľa <rating, meno>) v RAM 0 I/O
  - Odhad výsledku: <1 stránka
- Agregácia priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku: <1 stránka

- Celková cena 201 I/O

```
SELECT rating, count(x)
FROM Predavači
WHERE rating > 5 AND vek=20
GROUP BY rating
Having COUNT DISTINCT(meno)>2
```

Neklastrovaný index < rating, vek >

- Selekcia cez index a externú štruktúru
  - Odhad výsledku: 25% rating, 4% vek, dokopy 1%, teda výledok: 5 stránok (5*80 riadkov = 400 riadkov)
  - B+ strom: výška 3: 3 zhora dolu + 25% listov cca. pätinovej veľkosti (38 stránok) + 400 prístupov do externej štruktúry (potrebujeme stĺpec meno) = 441 I/O
  - Hash index: nepoužiteľný
- Projekcia na rating a meno spracúva priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku 80% zo vstupu: 4 stránky
- GROUP BY napríklad cez sort (podľa <rating, meno>) v RAM 0 I/O
  - Odhad výsledku: <1 stránka
- Agregácia priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku: <1 stránka

- Celková cena 441 I/O

Klastrovaný index pre rating

```
SELECT rating, count(x)
FROM Predavači
WHERE rating > 5 AND vek=20
GROUP BY rating
Having COUNT DISTINCT(meno)>2
```

- Selekcia cez index
  - Odhad výsledku: 25% rating, 10% vek, dokopy 2,5%, teda výledok: 12,5 stránok (12,5*80 riadkov = 1000 riadkov)
  - B+ strom: výška 3: 3 zhora dolu + 25% listov (188 stránok) = 201 I/O
    - Listov je pri 67% zaplnení 750, z toho štvrtina je 188
  - Hash index nepoužiteľný
- Projekcia na rating a meno spracúva priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku 80% zo vstupu: 10 stránok
- GROUP BY napríklad cez sort (podľa <rating, meno>) v RAM 0 I/O
  - Odhad výsledku: <1 stránka
- Agregácia priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku: <1 stránka

- Celková cena 201 I/O

Neklastrovaný index pre rating

```
SELECT rating, count(x)
FROM Predavači
WHERE rating > 5 AND vek=20
GROUP BY rating
Having COUNT DISTINCT(meno)>2
```

- Selekcia cez index a externú štruktúru
  - Odhad výsledku: 25% rating, 10% vek, dokopy 2,5%, teda výledok: 12,5 stránok (12,5*80 riadkov = 1000 riadkov)
  - B+ strom: výška 3: 3 zhora dolu + 25% listov cca. pätinovej veľkosti (38 stránok) + 1000 prístupov do externej štruktúry (potrebujeme stĺpec meno) = 1041 I/O
  - Hash index: nepoužiteľný
- Projekcia na rating a meno spracúva priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku 80% zo vstupu: 8 stránok
- GROUP BY napríklad cez sort (podľa <rating, meno>) v RAM 0 I/O
  - Odhad výsledku: <1 stránka
- Agregácia priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku: <1 stránka

- Celková cena 1041 I/O

Klastrovaný index pre vek

```
SELECT rating, count(x)
FROM Predavači
WHERE rating > 5 AND vek=20
GROUP BY rating
Having COUNT DISTINCT(meno)>2
```

- Selekcia cez index
  - Odhad výsledku: 50% rating, 4% vek, dokopy 2%, teda výledok: 10 stránok (10*80 riadkov = 800 riadkov)
  - B+ strom: výška 3: 3 zhora dolu + 4% listov (30 stránok) = 33 I/O
    - Listov je pri 67% zaplnení 750, z toho 4% je 30
  - Hash index: nájdenie 1 I/0, na vytiahnutie dát 500*0,04 = 20 I/O
- Projekcia na rating a meno spracúva priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku 80% zo vstupu: 8 stránok
- GROUP BY napríklad cez sort (podľa <rating, meno>) v RAM 0 I/O
  - Odhad výsledku: <1 stránka
- Agregácia priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku: <1 stránka

- Celková cena 20 - 33 I/O

Neklastrovaný index pre vek

```
SELECT rating, count(x)
FROM Predavači
WHERE rating > 5 AND vek=20
GROUP BY rating
Having COUNT DISTINCT(meno)>2
```

- Selekcia cez index a externú štruktúru
  - Odhad výsledku: 50% rating, 4% vek, dokopy 2%, teda výledok: 10 stránok (10*80 riadkov = 800 riadkov)
  - B+ strom: výška 3: 3 zhora dolu + 4% listov cca. pätinovej veľkosti (6 stránok) + 800 prístupov do externej štruktúry (potrebujeme stĺpec meno) = 806 I/O
  - Hash index: nájdenie 1 I/O + cca 3-4 stránky(malé záznamy) pretečenia + 800 prístupov do externej štruktúry (potrebujeme stĺpec meno) = 804 I/O
- Projekcia na rating a meno spracúva priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku 80% zo vstupu: 8 stránok
- GROUP BY napríklad cez sort (podľa <rating, meno>) v RAM 0 I/O
  - Odhad výsledku: <1 stránka
- Agregácia priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku: <1 stránka

- Celková cena 804 I/O

```
SELECT rating, count(x)
FROM Predavači
WHERE rating > 5 AND vek=20
GROUP BY rating
Having COUNT DISTINCT(meno)>2
```

Dva neklastrované indexy pre rating a vek

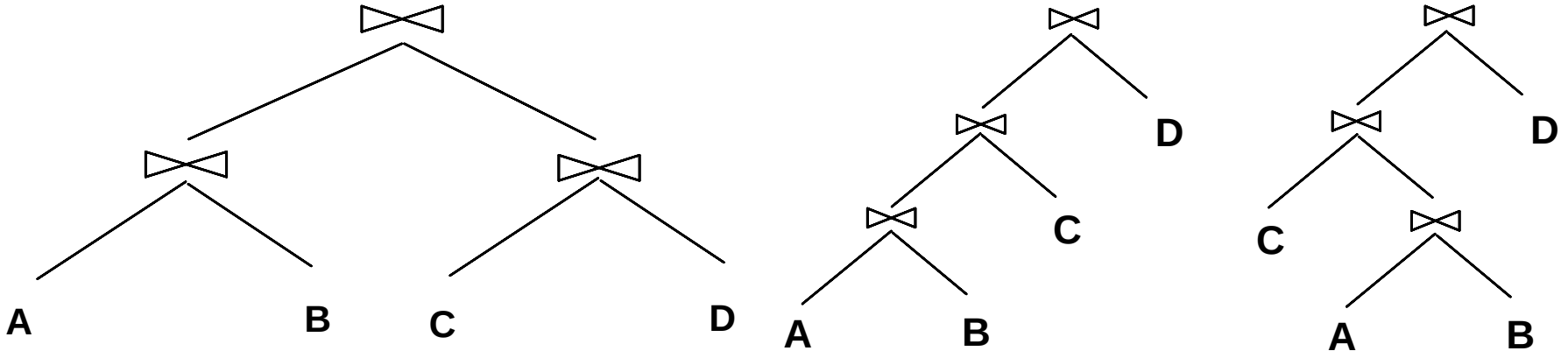
- Selekcia cez index a externú štruktúru
  - Odhad výsledku: 25% rating, 4% vek, dokopy 1%, teda výledok: 5 stránok (5*80 riadkov = 400 riadkov)
  - B+ stromy: výška 3: 3 zhora dolu + 4% listov cca. pätinovej veľkosti (6 stránok) + 3 zhora dolu + 25% listov cca. pätinovej veľkosti (38 stránok) + 400 prístupov do externej štruktúry (potrebujeme stĺpec meno) = 450 I/O
  - Hash index pre vek: nájdenie 1 I/O + cca 3-4 stránky(malé záznamy) – úspora 5 stránok = 445 I/O
- Projekcia na rating a meno spracúva priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku 80% zo vstupu: 4 stránky
- GROUP BY napríklad cez sort (podľa <rating, meno>) v RAM 0 I/O
  - Odhad výsledku: <1 stránka
- Agregácia priebežne bez prístupu na disk 0 I/O
  - Odhad výsledku: <1 stránka

- Celková cena 445 - 450 I/O

Indexy	cena	Odhad veľkosti prvého medzivýsledku
Bez indexov	540	25
Klastrovaný index <vek, rating>	11	5
Neklastrovaný index <vek, rating>	405	5
Klastrovaný index <rating, vek>	201	5
Neklastrovaný index < rating, vek >	441	5
Klastrovaný index pre rating	201	12.5
Neklastrovaný index pre rating	1041	12.5
Klastrovaný index pre vek	22 -33	10
Neklastrovaný index pre vek	804	10
Dva neklastrované indexy pre rating a vek	445 - 450	5

Dopyty nad viacerými reláciami

- Dopyty s 2 alebo viac reláciami potrebujú aj join
- Systém R uvažuje ***<u>iba left-deep join stromy</u>***
  - Ako sa počet joinov zvyšuje, počet alternatívnych plánov narastá veľmi rýchlo; *potrebujeme orezať priestor prehľadávania.*
  - Left-deep stromy nám dovoľujú generovať všetky ***plne prepojené plány*** bez ukladania výsledkov predchádzajúcich joinov
    - Vieme použiť Index nested loops join na každú vnútornú reláciu
    - Nie všetky left-deep stromy musia byť plne prepojené (napr. Sort merge join).



Výpočet Left-Deep plánov

- Left-deep plány sa líšia iba v poradí relácií, prístupových metódach pre každú reláciu a join metóde pre každý join.
- Ak spájame N relácií, rátame N prechodov:
  - Prechod 1: nájdi najlepší 1-relačný plán pre každú reláciu.
  - Prechod 2: nájdi najlepší spôsob ako spojiť výsledok každého 1-relačného plánu (ako outer join) s ďalšou reláciou.
  - Prechod N: nájdi najlepší spôsob ako spojiť výsledok (N-1)-relačného plánu (ako outer join) s N-tou reláciou.
- Pre každú podmnožinu relácií, zachovaj iba:
  - Celkovo najlacnejší plán a
  - Najlacnejší plán pre každú *zaujímavú postupnosť* riadkov.

Odhad ceny pre multirelačné plány

- Uvažujme blok dopytu:

```
SELECT  zoznam atribútov
FROM  zoznam relácií
WHERE  term1 AND ... AND termk
```

- Maximálny # riadkov vo výsledku je výsledkom kardinalít relácií z FROM klauzuly.
- Redukčný faktor (RF) každého termu má vplyv na veľkosť výsledku.
  - Výsledný počet riadkov = Maximálny počet riadkov * produkt všetkých RF.
- Multirelačné plány sú vybudované pomocov joinovania pomocou jednej novej relácie v jednom kroku.
  - Cena join metód, plus odhad veľkosti výsledku joinu nám dáva aj odhad ceny aj veľkosti výsledku

Stratégie vytvárania plánu

- Ak máme zložitejšiu selekciu, ktorej časť sa dá aplikovať skôr, aplikujeme ju
- Projekciou sa snažíme orezať atribúty najskôr ako to ide
    - Niekedy musíme do medzivýsledku dať viac stĺpcov, ako ide na výstup ak sú potrebné na použitie v nejakých podmienkach, agregáciách alebo v GROUP BY
- Snažíme sa zmeniť kartézske súčiny na joiny.
- ORDER BY, GROUP BY, agregácie sú spracované ako posledný krok, použitím buď užitočne usporiadaného plánu alebo najprv nasadíme operátor usporiadania alebo delenia cez hash.
- (i-1)-relačný plán nie je spájaný s ďalšou reláciou pokým neexistuje join podmienka medzi nimi (okrem prípadu, že už všetky predikáty vo WHERE boli použité).
- Napriek orezávaniu priestoru plánu, tento prístup je stále exponenciálny vzhľadom na počet tabuliek.

Jednoduché vnorené dopyty

- Ak nezávislý vnútorný dopyt vráti jednu hodnotu, ktorá sa dosadí do vonkajšieho dopytu
  - SELECT p.meno FROM Predacači p
    WHERE p.rating = (SELECT max(p2.rating) FROM Predavači p2)
- Ak nezávislý vnútorný dopyt vráti viac riadkov
  - SELECT p.meno FROM Predacači p
    WHERE p.pid IN (SELECT o.pid FROM Objednávky o
    WHERE o.bid = 100)
    - Tu ide reálne o join podľa pid, mohli by sa použiť všetky
    - Väčšina systémov spraví nested loops s použitím výsledku vnútorného selektu ako vnútornej relácie

Prepojené vnorené dopyty

- Vnútorný blok je typicky vypočítaný pre každý riadok vonkajšieho bloku
- Ak sa opakuje hodnota o.sid vnorený blok sa ráta pre každú hodnotu vždy odznova
- Nevieme použiť Index nested loops napr. ak máme index nad Objednávky.sid
- Optimalizátor nikdy neuvažuje, že by reláciu z vnútorného bloku uvažoval ako vonkajšiu pre join

```
SELECT  p.sname
FROM  Predavači p
WHERE EXISTS
   (SELECT  *
    FROM  Objednávky o
     WHERE  o.bid=103
     AND  p.sid=o.sid)
```

Vnorený blok pre optimalizáciu:

```
SELECT  *
FROM Objednávky o
WHERE  o.bid=103
    AND  o.sid= outer value
```

Ekvivaletný nevnorený dopyt:

```
SELECT  S.sname
FROM Sailors S, Reserves R
WHERE  S.sid=R.sid
   AND R.bid=103
```

Prepojené vnútorné dopyty

- Nevnorené dopyty sú umožnia optimalizátoru takmer vždy nájsť lepšiu stratégiu výpočtu.
- Súčasné optimalizátory nie sú schopné zistiť, že existuje aj ekvivalentný nevnorený dopyt
  - Ostáva to na múdrosti pisateľa dopytu