

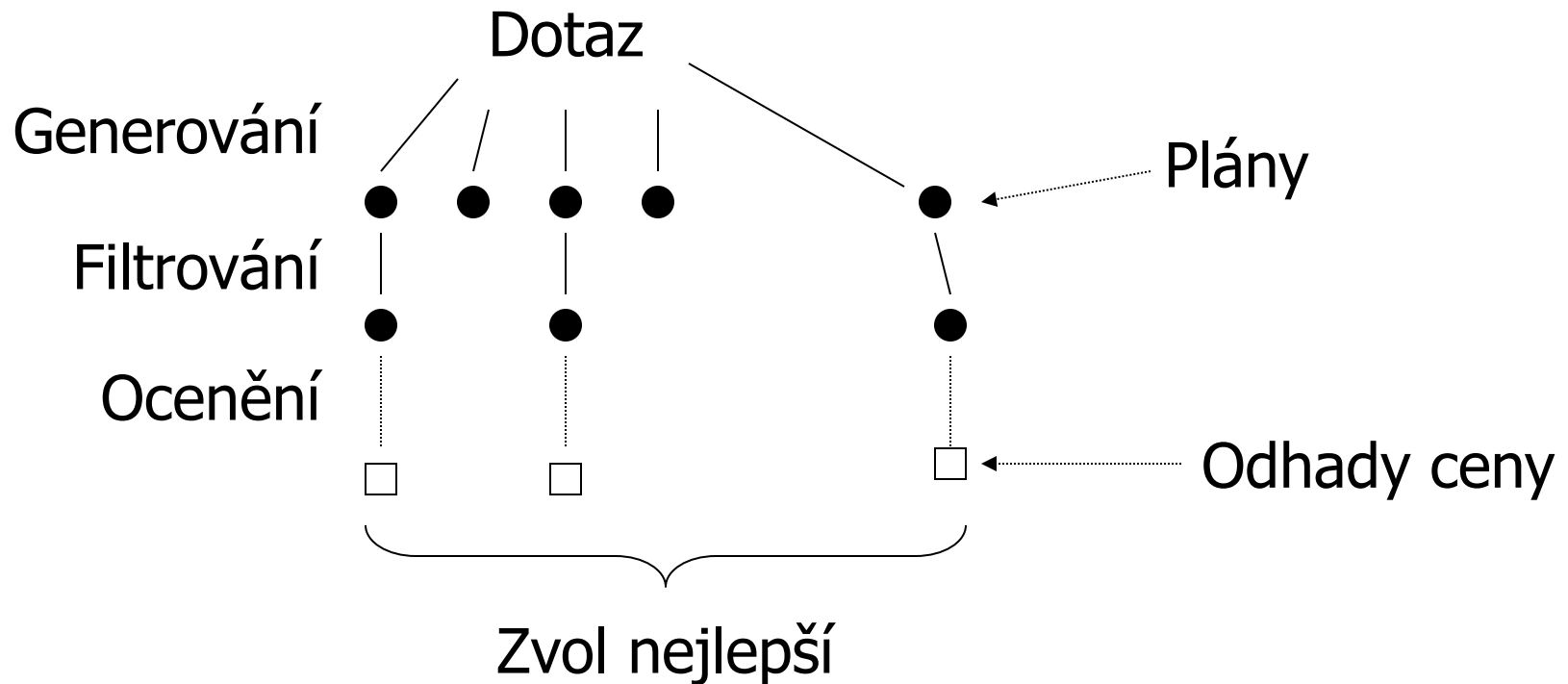


PA152: Efektivní využívání DB  
8. Optimalizace dotazu

Vlastislav Dohnal

# Optimalizace dotazu

- Generování a porovnávání plánů dotazu



# Generování plánu dotazu

- Zvážit používání:
  - Transformační pravidla rel. algebry
  - Implementace operací rel. algebry
  - Použití existujících indexů
  - Vytváření indexů a třídění podle potřeb

# Odhad ceny plánu

- Závisí na ceně provedení každé operace
  - Tj. její implementaci
- Předpoklady ceny operace
  - Vstup se čte z disku
  - Výstup zůstává v operační paměti
  - Operace na CPU
    - CPU stačí počítat během čtení z disku
    - často zanedbány nebo zjednodušeny
  - Komunikace po síti
    - Počítat u distribuovaných databází
  - Ignorování vyrovnávacích pamětí
- Odhad ceny operace
  - = počet čtení a zápisů z disku

# Odhad ceny plánu

## ■ Příklad nastavení PostgreSQL

<http://www.postgresql.org/docs/9.1/static/runtime-config-query.html#GUC-CPU-OPERATOR-COST>

- seq\_page\_cost (1.0)
- random\_page\_cost (4.0)
- cpu\_tuple\_cost (0.01)
- cpu\_index\_tuple\_cost (0.005)
- cpu\_operator\_cost (0.0025)
- work\_mem (1MB)
  - Memory available to an operation
- effective\_cache\_size (128MB)

# Odhad ceny plánu

## ■ Parametry

- $B(R)$  – velikost relace  $R$  v blocích
- $f(R)$  – max. počet záznamů relace v bloku
- $M$  – max. dostupná RAM v blocích
  
- $HT(i)$  – počet úrovní indexu  $i$
- $LB(i)$  – celkový počet listových bloků indexu

# Operace čtení relace: **table scan**

- Relace je shlukovaná (clustered)

R1 R2 R3 R4	R5 R6 R7 R8	...
-------------	-------------	-----

- Čtení je  $B(R)$
- TwoPhase-SortMerge =  $3B(R)$  čtení a zápisů
  - Finální zápis ignorujeme
- Relace není shlukovaná (non-clustered)

R1 R2 <i>S1 S2</i>	R3 R4 <i>S3 S4</i>	...
--------------------	--------------------	-----

- Čtení je až  $T(R)$  bloků!
- TwoPhase-SortMerge
  - $T(R) + 2B(R)$  čtení a zápisů

# Operace čtení relace: **index scan**

- Čtení relace s použitím indexu
  - Na libovolném atributu
  - Procházíme index → čteme záznamy
    - Čteme bloky indexu ( $\ll B(R)$ )
    - Čteme záznamy relace
  - Náklady:
    - (max.  $B(R)$  až  $T(R)$  čtení) + (max.  $\frac{m^{HT}-1}{m-1}$ )
      - Where  $m$  is an index arity ( $LB = m^{HT}$ )
  - Výhoda
    - Lze omezit pouze na interval záznamů
      - Min. náklady: 0 čtení



# Implementace operace

## ■ Použití konceptu **iterátor**

- *Open* – inicializace operace

  - příprava před vrácením řádků výsledku

- *GetNext* – vrácení dalšího řádku výsledku

- *Close* – ukončení operace

  - uvolnění dočasné paměti, ...

## ■ Výhody

- Výsledek nemusí být vygenerován „naráz“

  - Nezabírá paměť, nemusí být ukládán

- Operace lze řetězit (pipelining)

# Jednoprůchodové algoritmy

## ■ Implementace:

- Čtení relace → zpracování → výstupní paměť
- Zpracování záznam po záznamu

## ■ Operace

- Projekce, selekce, rušení duplicit (DISTINCT)
  - Náklady  $B(R)$
- Agregáční funkce (GROUP BY)
  - Náklady  $B(R)$
- Množinové operace, kartézský součin
  - Náklady  $B(R) + B(S)$

# Rušení duplicit – distinct

## ■ Postup zpracování

- Otestuj, zda je již záznam ve výstupu
- Ne, přidej na výstup

## ■ Testování existence ve výstupu

- Pamatovat si v paměti již vypsané záznamy
  - Lze použít  $M-1$  bloků
- Testování sekvenčně je pomalé ( $n^2$  porovnání)
- Použití hašování
- Omezení  $B(R) < M$

## ■ Lze realizovat pomocí iterátorů?

# Agregační funkce

## ■ Postup zpracování

- Vytváření skupin pro group-by atributy
- Ukládání hodnot atributů pro agregační funkce

## ■ Interní struktura

- Organizace skupin – např. hašování
- Agregační funkce
  - MIN, MAX, COUNT, SUM – pouze jedno „číslo“
  - AVG – dvě čísla (SUM a COUNT)
- Ukládaná informace je malá:  $M-1$  bloků bývá dostatečné
- Iterátory:
  - Vše je vypočteno v *Open*
  - Výhoda proudového zpracování mizí

# Množinové operace

- Požadavek  $\min(B(R), B(S)) < M-1$ 
  - Menší relace se načte celá, větší se čte postupně
  - Výjimka
    - Multimnožinové sjednocení s duplicitami
      - Obě relace se čtou postupně
- Předpoklad
  - R je větší relace, tj. S je celá v paměti
- Implementace
  - Obvykle pomocná vyhledávací struktura
    - Např. hašování

# Množinové operace

## ■ Množinové sjednocení a průnik

□ Pozor: *Ne multimnožinové verze*, tj. bez ALL v SQL

□ Načti S, vybuduj vyhledávací strukturu

■ Eliminuj duplicitní řádky

□ Při čtení R ověřuj přítomnost záznamu v S

■ Sjednocení:

□ Je, pak nic.

□ Není, pak přidej do struktury

■ Průnik:

□ Je, pak vypiš a smaž z interní struktury.

□ Není, pak nic.

□ Nakonec vypiš interní strukturu

■ Pouze při sjednocení!

# Množinové operace

- Množinový rozdíl ( $R-S \neq S-R$ )
- $R-S$ 
  - Načti  $S$ , vybuduj vyhledávací strukturu
    - Eliminuj případné duplicity v  $S$
  - Při čtení  $R$  ověřuj přítomnost záznamu v  $S$
  - Pokud není, dej na výstup
    - a přidej i do interní struktury
- $S-R$ 
  - Načti  $S$ , vybuduj vyhledávací strukturu
    - eliminuj duplicity
  - Při čtení  $R$  ověřuj přítomnost záznamu v  $S$
  - Pokud je, smaž záznam z  $S$
  - Nakonec vypiš zbylý obsah  $S$

# Multimnožinové operace

## ■ Multimnožinový průnik $R \cap_B S$

- Načti  $S$ , vybuduj vyhledávací strukturu
  - Místo ukládání duplicitních řádků ukládej jejich počet
- Při čtení  $R$  ověřuj přítomnost záznamu v  $S$
- Záznam byl nalezen, pak dej na výstup
  - A sniž počet záznamů!
  - Pokud je počet již nula, pak zruš z interní struktury.
- Záznam nebyl nalezen, pak nic



# Multimnožinové operace

## ■ Multimnožinový rozdíl $S -_B R$

- Používá stejný trik
- Záznam z  $R$  byl nalezen, sniž počet záznamů
- Nakonec vypiš pouze záznamy z  $S$ 
  - které mají kladný počet.

## ■ Multimnožinový rozdíl $R -_B S$

- Analogicky...
- Záznam z  $R$  nebyl v  $S$  nalezen  $\rightarrow$  výstup
- Záznam z  $R$  byl v  $S$  nalezen
  - $\rightarrow$  pokud je počet nula, dej na výstup.
  - $\rightarrow$  jinak sniž počet a nic

# Operace spojení

## ■ Kartézský součin

- Snadné cvičení...

## ■ Přirozené spojení (NATURAL JOIN v SQL)

- Předpoklad  $R(X, Y)$ ,  $S(Y, Z)$

- $X$  – atributy unikátní v  $R$ ,  $Z$  – atributy unikátní v  $S$
- $Y$  – atributy společné v  $R$  a  $S$

- Načti  $S$ , vybuduj vyhledávací strukturu pro  $Y$

- Pro záznam z  $R$ , najdi v  $S$  všechny odpovídající

- Na výstup dej jejich kombinace (eliminuj opakování  $Y$ )

# Jednoprůchodové algoritmy

## ■ Shrnutí

□ Unární operace:  $op(R)$

- $B(R) \leq M-1$ , 1 blok pro výstup

□ Binární operace:  $R \ op \ S$

- $B(S) \leq M-2$ , 1 blok pro  $R$ , 1 blok pro výstup

□ Cena =  $B(R) + B(S)$

## ■ Založeno na volné paměti $M$

□ Je známo  $\rightarrow$  ok

□ Není známo  $\rightarrow$  odhadnout

- Chyba  $\rightarrow$  swapování, výměna jednoprůchodového za dvouprůchodový algoritmus

# Algoritmy pro spojení

- Relace se nevejdou do paměti
- Základ – vnořené cykly (*nested-loop join*)
  - **for** each  $s$  in  $S$  **do**
    - **for** each  $r$  in  $R$  **do**
      - **if**  $r$  a  $s$  se shodují **then** output spojení  $r$  a  $s$ .

## ■ Příklad

□  $T(R) = 10\ 000$        $T(S) = 5\ 000$

□ Náklady =  $5\ 000 \cdot (1 + 10\ 000) = 50\ 005\ 000$  čtení

Čtení záznamu z  $S$

Čtení celé  $R$

# Algoritmy pro spojení

- Relace uloženy v blocích
- Blokované vnořené cykly ( $M=3$ )
  - *block-based nested-loop join*
    - R – vnitřní relace, S – vnější relace
- Příklad:
  - $B(R) = 1000$        $B(S) = 500$
  - Náklady =  $500 \cdot (1 + 1000) = 500\,500$  čtení

# Algoritmy pro spojení

- Využití vyrovnávací paměti (M bloků)
  - Načti M-2 bloků relace S naráz
    - Načítej relaci R po 1 bloku
      - Spojuj záznamy
  - Náklady:  $B(S)/(M-2) \cdot (M-2 + B(R))$  čtení
- Příklad  $R \bowtie S$ :
  - $M=102$
  - Náklady:  $5 \cdot (100 + 1000) = 5\,500$  čtení
  - Změna pořadí relací
    - Náklady:  $10 \cdot (100 + 500) = 6\,000$  čtení

# Algoritmy pro spojení – hodnocení

## ■ Vnořené cykly

- Vždy blokovávaná varianta
- Do paměti načítat dávky menší relace

## ■ Způsob uložení relace

- Důležité pro výslednou cenu
  - Nesouvislé → každý záznam jedno čtení
  - Souvislé → každý záznam  $B(R)/T(R)$  čtení

# Dvouprůchodové algoritmy

## ■ Princip:

- Předzpracování vstupu → uložení
- Zpracování

## ■ Předzpracování:

- Třídění (Víceprůchodový SortMerge)
- Hašování

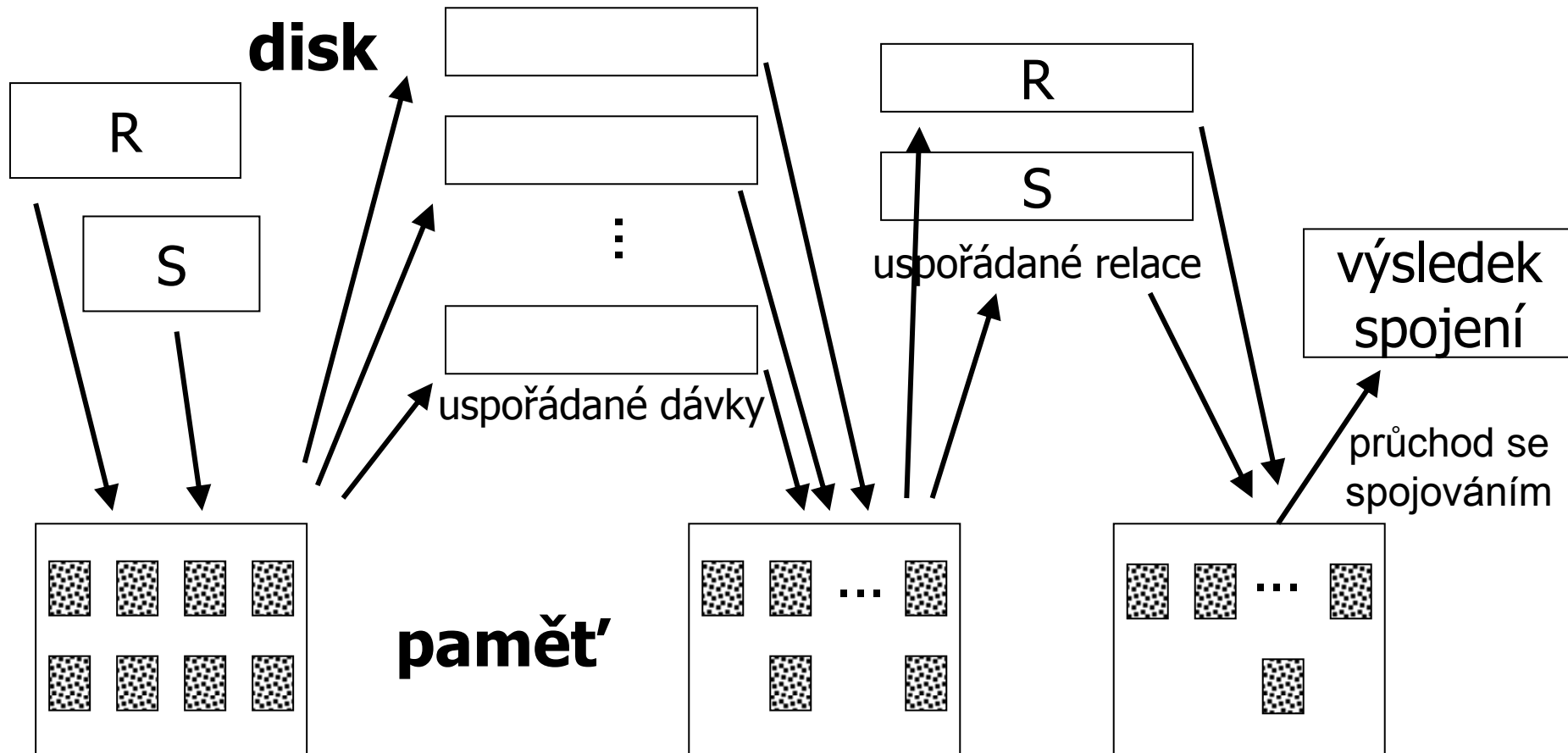
## ■ Operace:

- Rušení duplicit (DISTINCT)
- Agregáčn  funkce (GROUP BY)
- Spojen  relac , mnoŹinov  operace



# Algoritmy pro spojení – MergeJoin

■  $R \bowtie S$      $R(X,Y), S(Y,Z)$



# Algoritmy pro spojení – MergeJoin

- $R \bowtie S$      $R(X,Y), S(Y,Z)$
- Algoritmus:
  - Setříd' R a S
  - $i = 1; j = 1;$
  - **while**  $(i \leq T(R)) \wedge (j \leq T(S))$  **do**
    - **if**  $R[i].Y = S[j].Y$  **then** doJoin()
    - **else if**  $R[i].Y > S[j].Y$  **then**  $j = j+1$
    - **else if**  $R[i].Y < S[j].Y$  **then**  $i = i+1$

# Algoritmy pro spojení – MergeJoin

## ■ Funkce doJoin():

□ Proved' nested-loop join pro řádky se stejným Y

□ **while**  $(R[i].Y = S[j].Y) \wedge (i \leq T(R))$  **do**

■  $j_2 = j$

■ **while**  $(R[i].Y = S[j_2].Y) \wedge (j_2 \leq T(S))$  **do**

□ Vypiš spojení  $R[i]$  a  $S[j_2]$

□  $j_2 = j_2 + 1$

■  $i = i + 1$

□  $j = j_2$

# Algoritmy pro spojení – MergeJoin

<b>i</b>	<b>R[i].Y</b>	<b>S[j].Y</b>	<b>j</b>
1	10	5	1
2	20	20	2
3	20	20	3
4	30	30	4
5	40	30	5
		50	6
		52	7

# Algoritmy pro spojení – MergeJoin

## ■ Cena

□ SortMerge R a S  $\rightarrow 4 \cdot (B(R) + B(S))$

□ MergeJoin  $\rightarrow B(R) + B(S)$

## ■ Příklad (M=102)

□ MergeJoin

■ Uspořádání:  $4 \cdot (1000 + 500) = 6000$  čtení/zápisů

■ Spojení:  $1000 + 500 = 1500$  čtení

■ Celkem: 7500 čtení/zápisů

□ Původní block-based nested-loop join

■ 5500 čtení

# Algoritmy pro spojení – MergeJoin

## ■ MergeJoin

- Předzpracování je drahé

- Pokud jsou relace upořádané podle Y, lze vynechat.

## ■ Náklady – analýza

- MergeJoin – lineární

- Block-based Nested-loop Join – kvadratické

- → od jisté velikosti relací je MergeJoin lepší

# Algoritmy pro spojení – MergeJoin

## ■ Jiný příklad

□  $B(R) = 10000$

$B(S) = 5000$

□  $M = 102$  bloků

### □ Block-based Nested-loop Join

■  $(5\ 000/100) \cdot (100 + 10\ 000) = 505\ 000$  čtení

### □ MergeJoin

■  $5 \cdot (10\ 000 + 5\ 000) = 75\ 000$  čtení a zápisů

# Algoritmy pro spojení – MergeJoin

## ■ Paměťové nároky

- Omezení na  $B(R) < M^2$  a  $B(S) < M^2$

## ■ Optimální paměť

- Používáme SortMerge na relaci R

- Počet dávek =  $B(R) / M$ , Délka dávky = M

- Omezení: počet dávek  $\leq M-1$

- $B(R) / M < M \rightarrow B(R) < M^2 \rightarrow M > \sqrt{B(R)}$

## ■ Příklad

- $B(R) = 1000 \rightarrow M > 31,62$

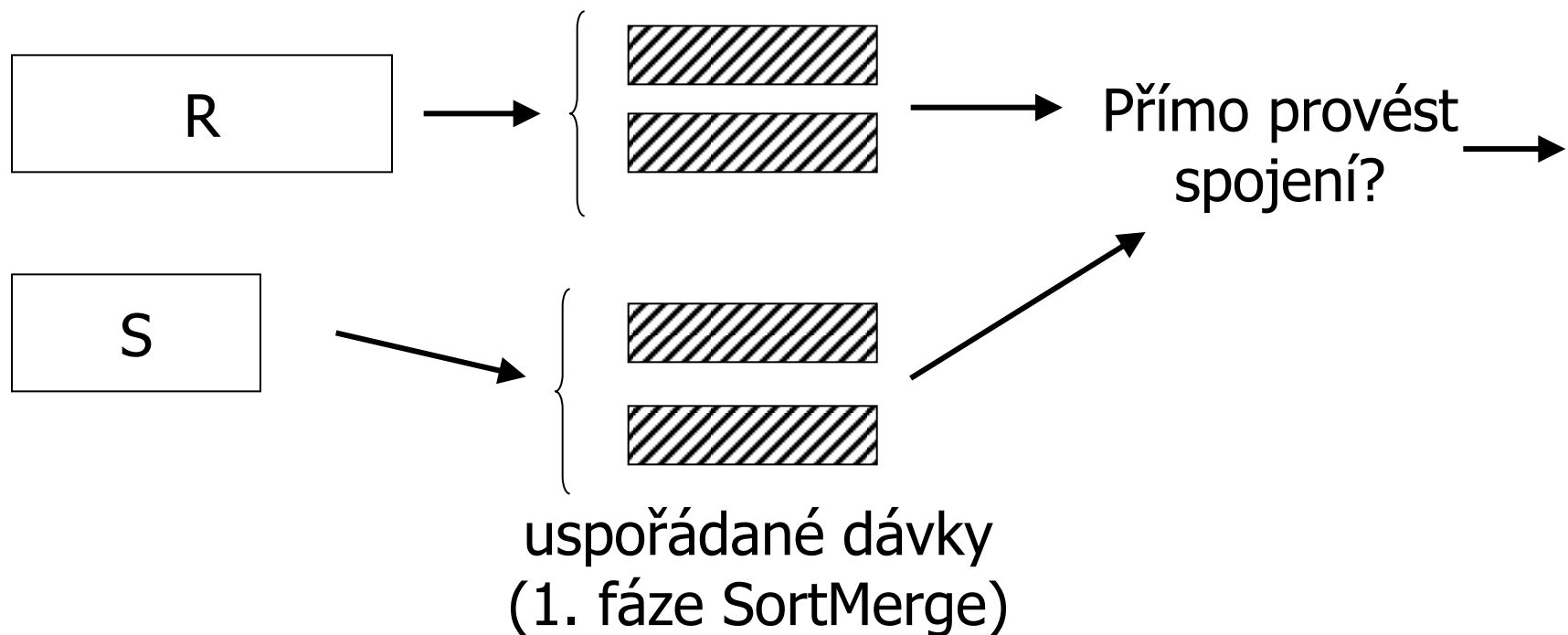
- $B(S) = 500 \rightarrow M > 22,36$



# Algoritmy pro spojení – MergeJoin

- Vylepšení:

- není potřeba mít relace zcela uspořádané



# Algoritmy pro spojení – SortJoin

## ■ Vylepšení

- Vytvoř seříděné dávky R a S
- Načti první blok z každé dávky
- Zjisti minimální hodnotu v Y
  - Najdi odpovídající záznamy z ostatních dávek
  - Proveď spojení

## ■ Pokud je hodně řádků se stejným y

- Aplikuj block-nested-loop join ve zbytku paměti

# Algoritmy pro spojení – SortJoin

## ■ Náklady

- Uspořádání dávek:  $2 \cdot (B(R) + B(S))$
- Provedení spojení:  $B(R) + B(S)$

## ■ Omezení

- Délka dávek  $M$ , počet dávek  $M$
- $\rightarrow B(R) + B(S) \leq M^2$

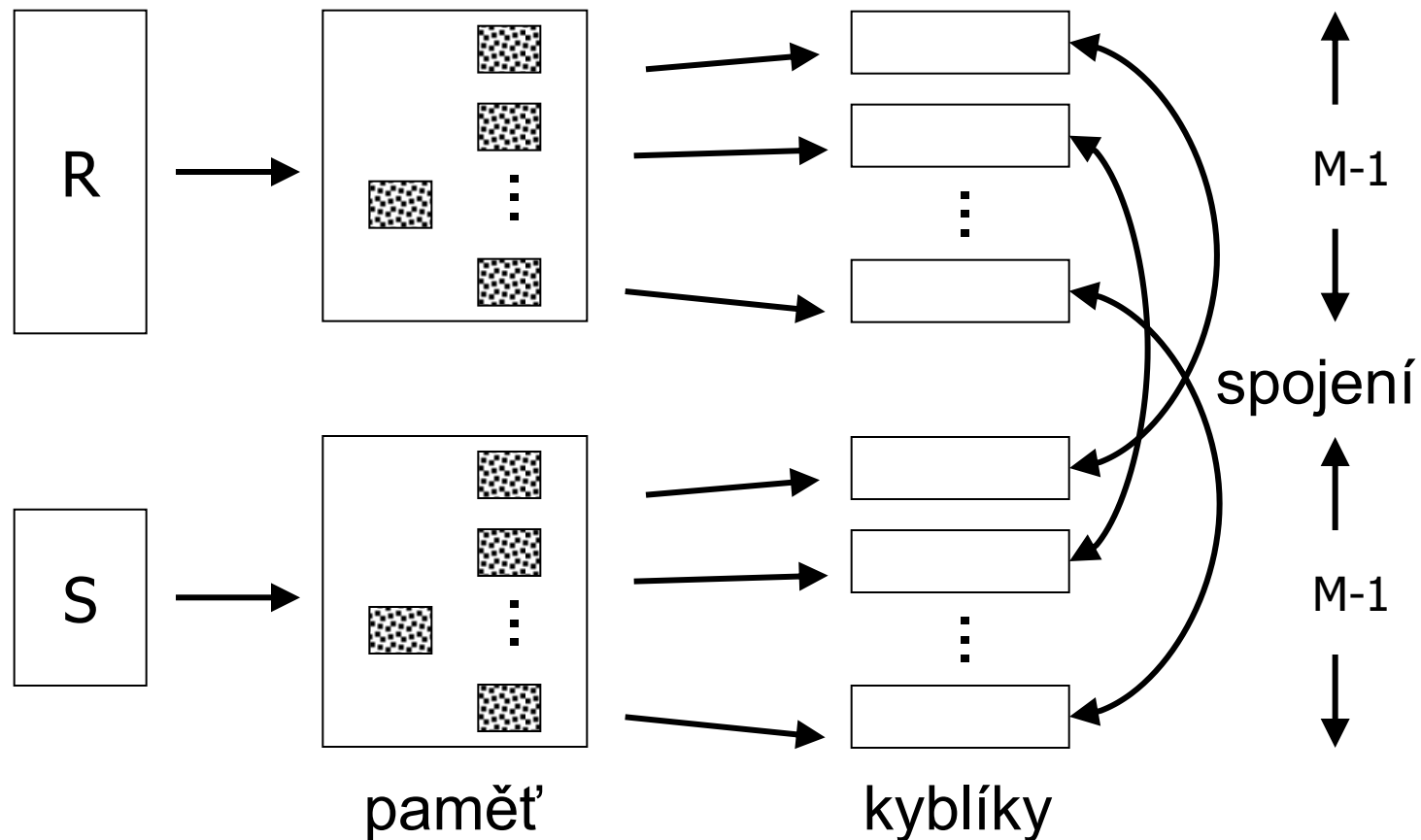
## ■ Příklad ( $M=102$ )

- Uspořádání dávek:  $2 \cdot (1000 + 500)$
- Spojení:  $1000 + 500$
- Celkem: 4 500 čtení/zápisů

- $\rightarrow$  lepší než block-based nested-loop join

# Algoritmy pro spojení – HashJoin

■  $R \bowtie S$      $R(X,Y), S(Y,Z)$



# Algoritmy pro spojení – HashJoin

- $R \bowtie S$       $R(X,Y), S(Y,Z)$ 
  - Pro atributy  $Y$  vytvoř hašovací funkci
  - Vytvoř hašovaný index pro  $R$  i  $S$ 
    - Počet kyblíků je  $M-1$
  - Pro každé  $i \in [1, M-1]$ 
    - Načti kyblík  $i$  pro  $R$  a  $S$
    - Proveď vyhledání odpovídajících si záznamů a jejich spojení

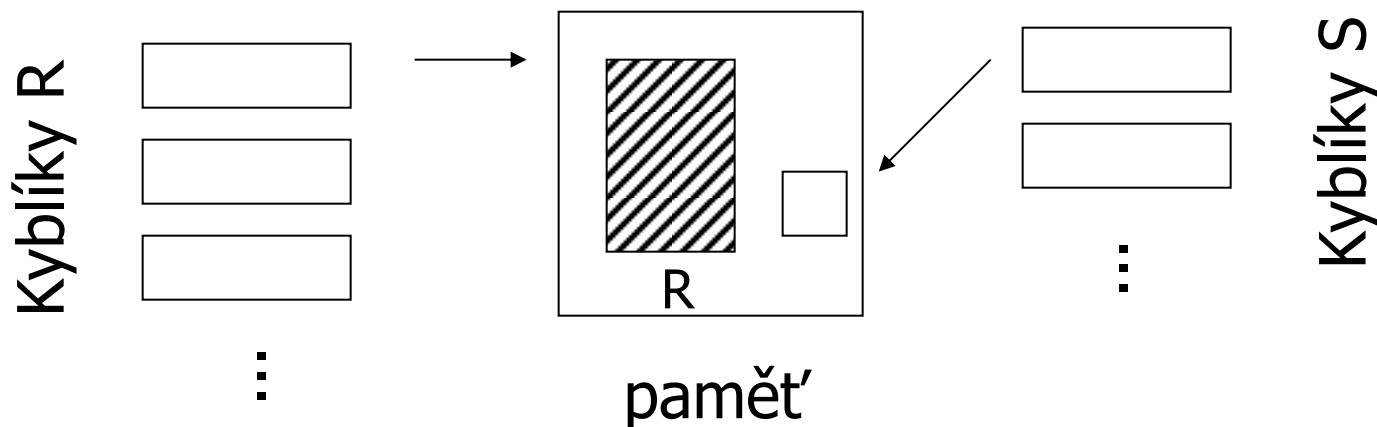
# Algoritmy pro spojení – HashJoin

## ■ Spojování kyblíků

□ Kyblík R načti celý ( $\leq M-2$ )

■ Pro zrychlení si vytvoř paměťové hašování

□ Kyblík S čti po blocích



# Algoritmy pro spojení – HashJoin

## ■ Náklady:

- Vytvoření hašovaného indexu:  $2 \cdot (B(R) + B(S))$
- Provedení spojení:  $B(R) + B(S)$

## ■ Omezení:

- Velikost každého kyblíku  $R$  (nebo  $S$ )  $\leq M-2$ 
  - Odhad:  $\min(B(R), B(S)) \leq M^2$

## ■ Příklad:

- Hašování:  $2 \cdot (1000 + 500)$
- Spojení:  $1000 + 500$
- Celkem: 4 500 čtení/zápisů

# Algoritmy pro spojení – HashJoin

## ■ Minimální paměťové nároky

### □ Hašování R, optimální naplnění kyblíků

#### ■ Celkem máme M paměťových bloků

#### ■ Velikost kyblíku

□  $B(R) / (M-1)$

□ Musí být menší než M (kvůli spojení)

□  $\rightarrow B(R) / (M-1) < M-2$

■  $\rightarrow M-2 > \sqrt{B(R)}$



# Algoritmy pro spojení – HashJoin

## ■ Optimalizace

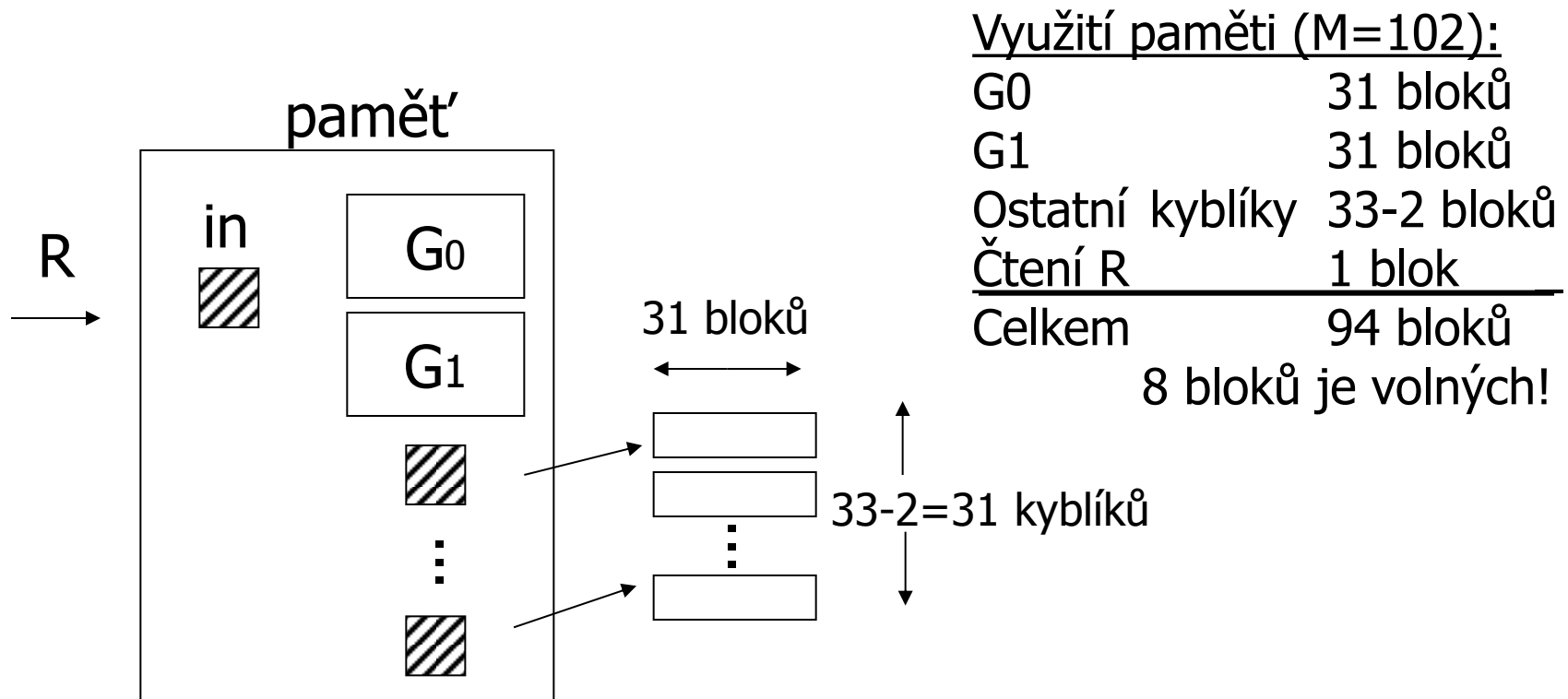
- ponech některé kyblíky v paměti
- Hybrid HashJoin

## ■ Optimum počtu kyblíků pro R

- $\sqrt{B(R)} \approx 33$
- Tj. 31 bloků má každý kyblík
- $M=102$ 
  - → ponechej 2 kyblíky v paměti (62 bloků)
  - → zbývá 40 bloků paměti

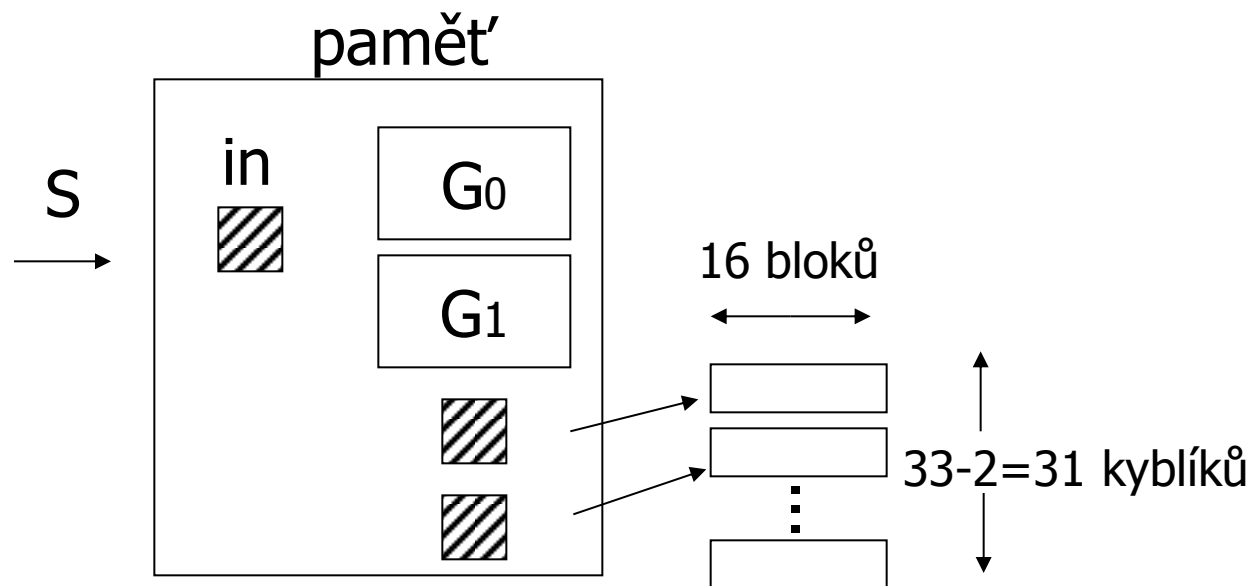
# Alg. pro spojení – Hybrid HashJoin

## ■ Paměť pro vytvoření hašovaného indexu R



# Alg. pro spojení – Hybrid HashJoin

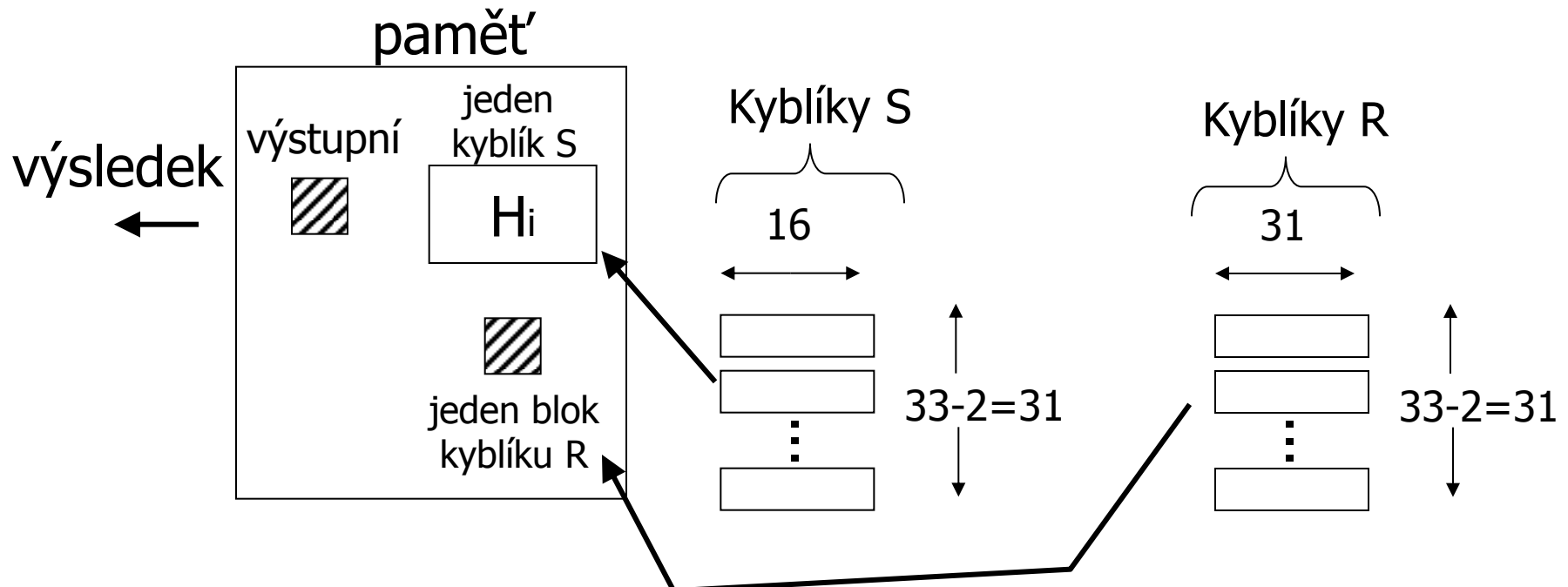
- Paměť pro vytvoření hašovaného indexu S
  - $500/33 = 16$  bloků na kyblík
  - Záznamy hašované do kyblíku 0 a 1
    - Vyřešit hned (R je v paměti) → výstup



# Alg. pro spojení – Hybrid HashJoin

## ■ Spojení kyblíků

- Pouze pro kyblíky s id 2-32
- Načti jeden kyblík celý do paměti, druhý procházej po blocích



# Alg. pro spojení – Hybrid HashJoin

## ■ Náklady:

- Hašování R:  $1000 + 31 \cdot 31 = 1961$  čtení/zápisů
- Hašování S:  $500 + 31 \cdot 16 = 996$  čtení/zápisů
  - Pouze 31 kyblíků!
- Spojení:  $31 \cdot 31 + 31 \cdot 16 = 1457$  čtení
  - 2 kyblíky jsou již zpracovány
- Celkem: 4414 čtení/zápisů

# Algoritmy pro spojení

## ■ Hybrid HashJoin

- Kolik kyblíků ponechat v paměti?

- Empiricky: 1 kyblík

## ■ Hašování ukazatelů

- Hašování ne záznamů, ale ukazatelů

- Do kyblíků ukládej dvojice [hodnota, ukazatel]

- Spojování

- Při shodě hodnot si musíme záznam načíst

# Alg. pro spojení – Hašování ukazatelů

## ■ Příklad

- Do bloku se vejde 100 dvojic hodnota-klíč
- Odhadovaný výsledek je 100 záznamů
- Náklady:
  - Hašování S do paměti
    - 5000 záznamů →  $5000/100$  bloků = 50 bloků
  - Spojení – čti R a spojuj
    - Při shodě načti záznam S → 100 čtení
  - Celkem:  $500 + 1000 + 100 = 1600$  čtení

# Algoritmy pro spojení – IndexJoin

- $R \bowtie S$      $R(X,Y), S(Y,Z)$
- Předpoklad:  $R$  má index nad atributy  $Y$
- Postup:
  - Pro každý záznam  $s \in S$
  - Prohledej index na shodu  $\rightarrow$  záznamy  $A$ 
    - Pro každý záznam  $r \in A$
    - Vypiš kombinaci  $r$  a  $s$



# Algoritmy pro spojení – IndexJoin

## ■ Příklad

### □ Předpoklady

- Index na Y pro relaci R:  $HT=2$ ,  $LB=200$

## ■ Situace 1

### □ Index se vejde do paměti

### □ Náklady:

- Průchod S: 500 čtení ( $B(S)=500$ ,  $T(S)=5000$ )
- Prohledání indexu: zdarma
  - Pokud shoda, načti záznam R → 1 čtení

# Algoritmy pro spojení – IndexJoin

## ■ Náklady

□ Závisí na počtu shod v indexu

□ Případy:

- A) Y je v R primární klíč, v S je cizí klíč → 1 záznam  
Výsledek:  $500 + 5000 \cdot 1 \cdot 1 = 5500$  čtení
- B)  $V(R, Y) = 5000$   $T(R) = 10\ 000$   
rovnoměrné rozložení → 2 záznamy  
Výsledek:  $500 + 5000 \cdot 2 \cdot 1 = 10500$  čtení
- C)  $DOM(R, Y) = 1\ 000\ 000$   $T(R) = 10\ 000$   
→  $10k/1m = 1/100$  záznamu  
Výsledek:  $500 + 5000 \cdot (1/100) \cdot 1 = 550$  čtení

# Algoritmy pro spojení – IndexJoin

## ■ Situace 2

- Index se nevejde do paměti
- Index na Y pro R má 201 bloků
  - V paměti udržuj kořen a 99 listů
- Náklady pro vyhledání
  - $0 \cdot (99/200) + 1 \cdot (101/200) = 0.505$  čtení

# Algoritmy pro spojení – IndexJoin

## ■ Situace 2

### □ Náklady

- $B(S) + T(S) \cdot (\text{prohledání indexu} + \text{čtení záznamů})$

### □ Případy:

- A)  $\rightarrow$  1 záznam

Výsledek:  $500 + 5000 \cdot (0.5 + 1) = 8000$  čtení

- B)  $\rightarrow$  2 záznamy

Výsledek:  $500 + 5000 \cdot (0.5 + 2) = 13000$  čtení

- C)  $\rightarrow$  1/100 záznamu

Výsledek:  $500 + 5000 \cdot (0.5 + 1/100) = 3050$  čtení

# Algoritmy pro spojení – shrnutí

$R \bowtie S$   
 $B(R) = 1000$   
 $B(S) = 500$

Algorithm	Costs
Block-based Nested-loop Join	5500
Merge Join (w/o sorting)	1500
Merge Join (with sorting)	7500
Sort Join	4500
Index Join (R.Y index)	5500 → 550
Hash Join	4500
Hybrid	4414
Pointers	1600

# Algoritmy pro spojení – doporučení

- Block-based Nested-loop Join
  - Vhodné pro malé relace (vzhledem k paměti)
- Pro spojení na rovnost (*equi-join*)
  - Relace nejsou uspořádané a nejsou indexy → HashJoin
- SortJoin
  - Vhodný pro spojení s nerovností (*non-equi-join*)
  - Např.  $R.Y > S.Y$
- MergeJoin
  - Pokud jsou relace již uspořádané
- IndexJoin
  - Pokud jsou indexy, může být vhodná volba
  - Závisí na velikosti odpovědi

# Dvouprůchodové algoritmy

- Pomocí třídění

- Rušení duplicit
- Agregáčn  funkce (GROUP BY)
- MnoŹinov  operace

# Rušení duplicit

## ■ Postup zpracování

- Proved' první fázi SortMerge

  - → uspořádané dávky na disku

- Z každé dávky načítej postupně bloky

  - Vezmi nejmenší záznam a dej na výstup

  - Přeskoč všechny duplicitní záznamy

## ■ Vlastnosti

- Náklady:  $3B(R)$

- Omezení:  $B(R) \leq M^2$

  - Optimální  $M \geq \sqrt{B(R)}$



# Agregační funkce

- Postup (podobný předchozímu)
  - Uspořádej dávky  $R$  (podle group-by atributů)
  - Z každé dávky načítej postupně bloky
    - Vezmi nejmenší záznam  $\rightarrow$  nová skupina
      - Počítej agregační funkce pro všechny stejné záznamy
      - Žádný další není  $\rightarrow$  vypiš výsledky na výstup
- Vlastnosti
  - Náklady:  $3B(R)$
  - Omezení:  $B(R) \leq M^2$ 
    - Optimální  $M \geq \sqrt{B(R)}$

# Množinové sjednocení

- Pro multimnožiny není třeba dvou průchodů
- Množinové sjednocení
  - Proveď první fázi SortMerge pro  $R$  a  $S$ 
    - → uspořádané dávky na disku
  - Z každé dávky ( $R$  i  $S$ ) načítej postupně bloky
    - Vezmi nejmenší záznam a dej na výstup
    - Přeskoč všechny duplicitní záznamy (z  $R$  i  $S$ )
- Vlastnosti
  - Náklady:  $3(B(R) + B(S))$
  - Omezení:  $\sqrt{B(R)} + \sqrt{B(S)} \leq M$

# Množinový průnik a rozdíl

- $R \cap S$ ,  $R - S$ ,  $R \cap_B S$ ,  $R -_B S$

- Postup

- Proveď první fázi SortMerge pro  $R$  a  $S$

- Z každé dávky ( $R$  i  $S$ ) načítej postupně bloky

- Vezmi nejmenší záznam  $t$

- Spočítej jeho všechny výskyty v  $R$  a  $S$  (odděleně)

- $\#_R$ ,  $\#_S$

- Vypiš na výstup (respektuj danou operaci)

# Množinový průnik a rozdíl

## ■ Ad výpis

□  $R \cap S$ : vypiš  $t$ ,

■ pokud  $\#_R > 0 \wedge \#_S > 0$

□  $R \cap_B S$ : vypiš  $t$   $\min(\#_R, \#_S)$ -krát

□  $R - S$ : vypiš  $t$ ,

■ pokud  $\#_R > 0 \wedge \#_S = 0$

□  $R -_B S$ : vypiš  $t$   $\max(\#_R - \#_S, 0)$ -krát

## ■ Vlastnosti

□ Náklady:  $3(B(R) + B(S))$

□ Omezení:  $\sqrt{B(R)} + \sqrt{B(S)} \leq M$

# Dvouprůchodové algoritmy

## ■ Hašování

- Rušení duplicit
- Agregáčn  funkce (GROUP BY)
- MnoŹinov  operace

# Rušení duplicit

## ■ Postup zpracování

□ Proved' hašování  $R$  do  $M-1$  kyblíků

■ → kyblíky ulož na disk

□ Pro každý kyblík

■ Načti do paměti (velikost kyblíku max.  $M-1$ )

■ Zruš duplicity → zbytek na výstup

## ■ Vlastnosti

□ Náklady:  $3B(R)$

□ Omezení:  $B(R) \leq (M-1)^2$

# Agregační funkce

- Postup (podobný předchozímu)
  - Proved' hašování  $R$  do  $M-1$  kyblíků
    - podle group-by atributů
  - Pro každý kyblík
    - Načti do paměti (velikost kyblíku max.  $M-1$ )
    - Vytvoř skupiny a spočítej agregační funkce
    - Vypiš výsledky na výstup
- Vlastnosti
  - Náklady:  $3B(R)$
  - Omezení:  $B(R) \leq (M-1)^2$

# Množinové sjednocení, průnik, rozdíl

## ■ Postup

- Proveď hašování pro  $R$  a  $S$  (stejnou haš. funkcí!)
  - Vždy  $M-1$  kyblíků
- Zpracuj vždy dvojici kyblíků  $R_i$  a  $S_i$ 
  - Jeden z kyblíků načti do paměti
    - velikost kyblíku max.  $M-2$
  - Druhý zpracuj postupně

## ■ Vlastnosti

- Náklady:  $3(B(R) + B(S))$
- Omezení:  $\min(B(R), B(S)) \leq (M-2)^2$



# Summary

## ■ Operations

- distinct, group by, set operations, joins

## ■ Algorithm type

- one-pass, two-pass

## ■ Implementation

- Sorting
- Hashing
- Exploiting indexes

## ■ Costs

- blocks to read/write
- memory footprint