evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

DISTRIBUOVANÉ DOTAZY

2016/17

Zdroj: Özsu M., T., Valduriez P.: Principles of Distributed Database Systems, Springer, 2011

Dotazy v centralizovaných systémech

Příklad: **EMP**(ENO, ENAME, TITLE)
ASG(ENO, PNO, RESP, DUR)

SELECT ENAME
FROM EMP, ASG
WHERE EMP.ENO = ASG.ENO
AND RESP = 'Manager'

Procesor dotazů vyjádří dotaz prostřednictvím RA.

Strategie 1

$$\Pi_{ENAME}(\sigma_{RESP='Manager' \wedge EMP.ENO=ASG.ENO}(EMP \times ASG))$$

Strategie 2

$$\Pi_{ENAME}(EMP \bowtie_{ENO} (\sigma_{RESP='Manager'}(ASG))$$

Zpracování dotazů v distribuovaném prostředí

- V centralizovaném systému stačí využít transformační pravidla relační algebry a transformovat dotaz do ekvivalentního tvaru o kterém víme, že je efektivnější.
- V distribuovaném prostředí musíme ještě určit, na kterém místě budou data zpracována a případně v jakém pořadí budou přenášena.

Příklad:

Předpokládejme, že tabulky EMP a ASG jsou horizontálně fragmentovány takto:

- $EMP_1 = \sigma_{ENO \leq \text{“E3“}} (EMP)$ $\quad\quad$ $ASG_1 = \sigma_{ENO \leq \text{“E3“}} (ASG)$
- $EMP_2 = \sigma_{ENO > \text{“E3“}} (EMP)$ $\quad\quad$ $ASG_2 = \sigma_{ENO > \text{“E3“}} (ASG)$

Fragmenty EMP_1, EMP_2, jsou uloženy v místech 1, 2

a fragmenty ASG_1, ASG_2 v místech 3, 4.

Výsledek se očekává v místě 5.

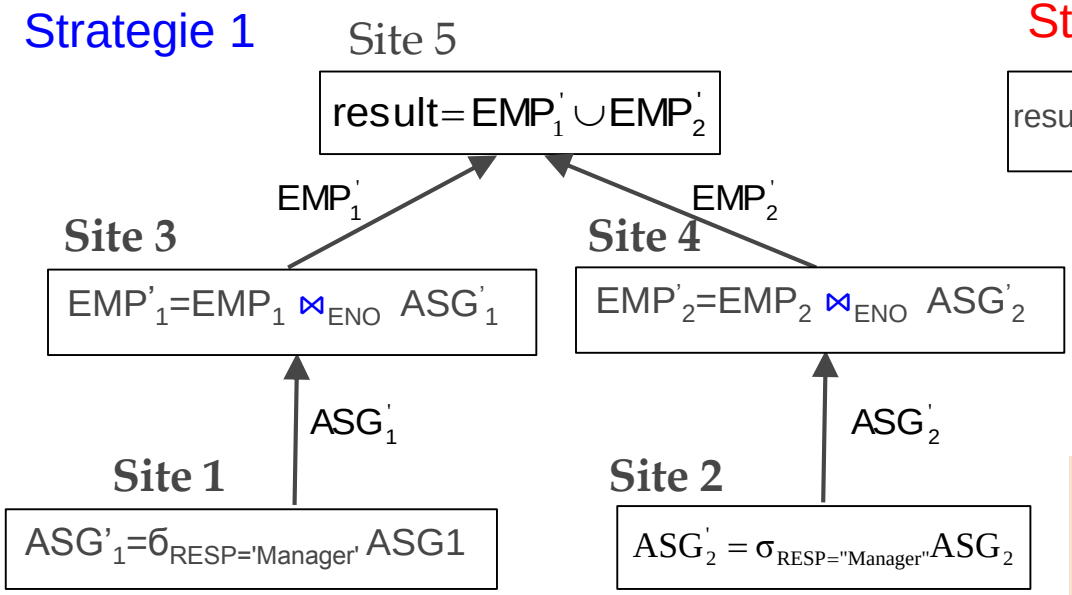
Možné strategie vykonání dotazu

Site 1 | Site 2 | Site 3 | Site 4 | Site 5

$EMP_1 = \sigma_{ENO \leq 'E3'}(EMP)$ | $EMP_2 = \sigma_{ENO > 'E3'}(EMP)$ | $ASG_1 = \sigma_{ENO \leq 'E3'}(ASG)$ | $ASG_2 = \sigma_{ENO > 'E3'}(ASG)$ | Výsledek

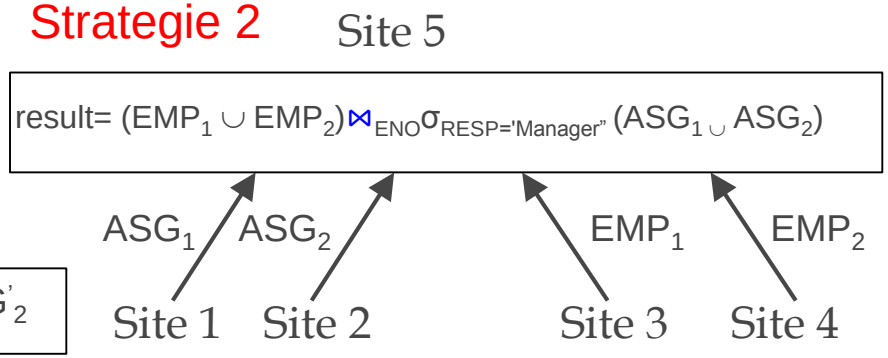
DOTAZ: $EMP \bowtie_{ENO} (\sigma_{RESP=\text{“Manager”}} (ASG))$ nad distribuovanou databází

Strategie 1



Nejdříve zmenšíme relace a potom realizujeme přenos.

Strategie 2



Nejdříve přeneseme všechny relace na místo kde bude výsledek a potom realizujeme dotaz.

Náklady na realizaci dotazů

- Předpokládejme že data jsou *rovnoměrně* rozdělena mezi místa v síti, relace ASSG a EMP jsou *lokálně klastrovaná* dle atributů RESP a ENO a lze tak přistoupit k řádkům přímo.
- Dále platí:
  - Počet řádků (EMP) = 400, počet řádků (ASG) = 1000
  - Počet manažerů v tabulce ASG = 20
  - Cena přístupu k řádku = 1 jednotka
  - Cena přenosu 1 řádku = 10 jednotek
- Strategie 1
  - vytvoření ASG': (10+10) ∗ *cena přístupu k řádku* — 20
  - přenos ASG' na místo kde je EMP: (10+10) ∗ *cena přenosu řádku* — 200
  - vytvoření EMP': (10+10) ∗ *cena přístupu k řádku* ∗ 2 — 40
  - přenos EMP' na místo 5: (10+10) ∗ *cena přenosu řádku* — 200

  Cena celkem — 460
- Strategie 2
  - přenos EMP na místo 5: 400 ∗ *cena přenosu řádku* — 4 000
  - přenos ASG na místo 5: 1000 ∗ *cena přenosu řádku* — 10 000
  - vytvoření ASG': 1000 ∗ *cena přístupu k řádku* — 1 000
  - Vytvoření spojení EMP a ASG': 400 ∗ 20 ∗ *cena přístupu k řádku* — 8 000

  Cena celkem — 23 000

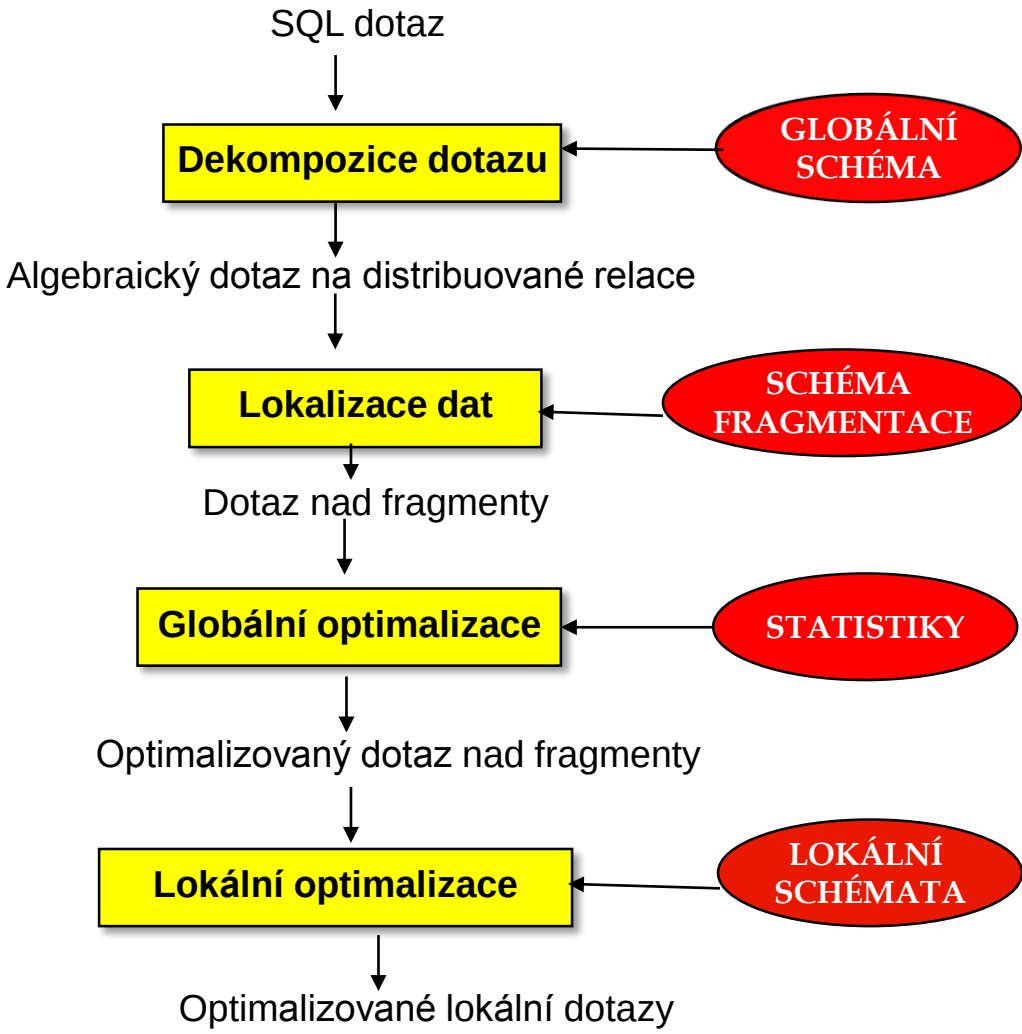
Závěr: Strategie 1 je 50 krát lepší než strategie 2.

Metodologie zpracování distribuovaných dotazů

Cíl:

Transformovat SQL dotaz nad globálními relacemi do posloupnosti databázových operací nad fragmenty.

Optimalizátor musí nalézt nejlepší místo pro zpracování dat a určit, v jakém pořadí se budou výsledky v síti přenášet.

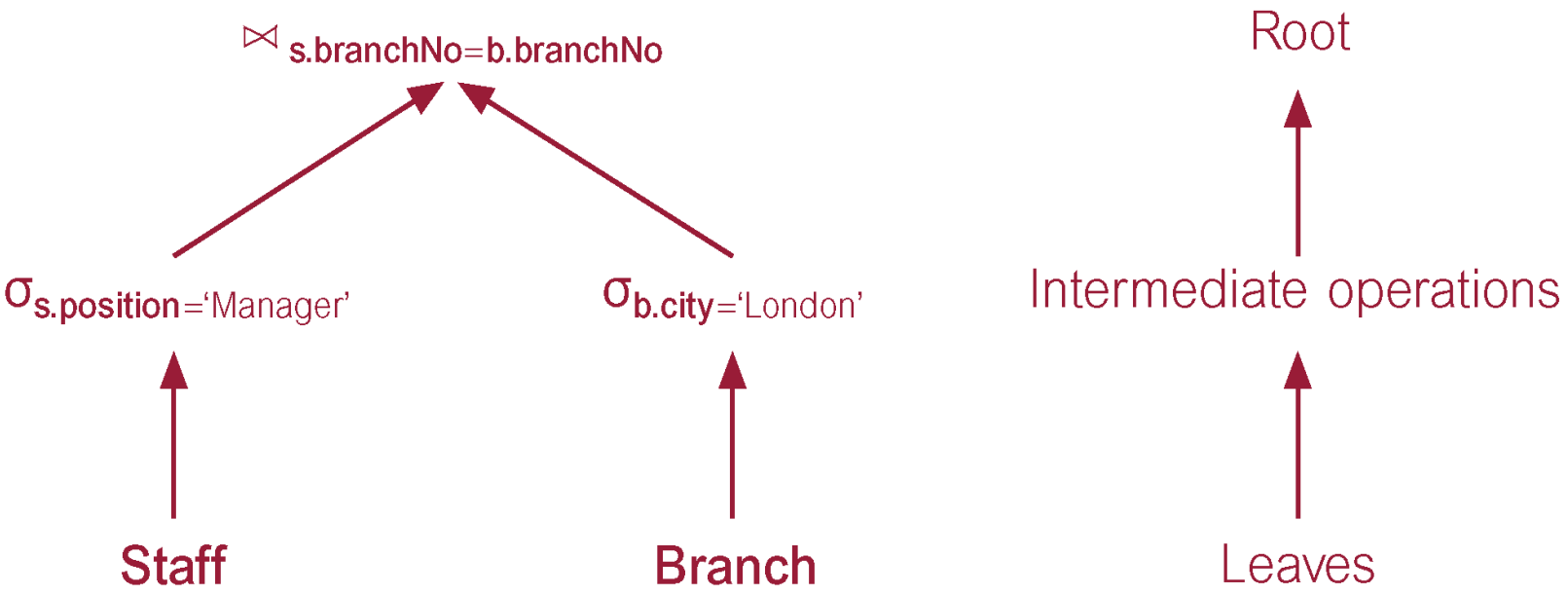


Dekompozice dotazu

- V této vrstvě se aplikují optimalizační metody používané v centralizovaných systémech, jejichž základem je vyjádření dotazu ve formě stromu dotazu.
- Postup při vytváření stromu dotazu:
  - Pro každou relaci v dotazu se vytvoří list.
  - Pro každou relaci vytvořenou operací relační algebry se vytvoří uzel.
  - Kořen reprezentuje výsledek dotazu.
  - Posloupnost vykonávání kroků je od listů ke kořeni.

Příklad: strom dotazu


```
SELECT *
FROM Staff S, Branch B
WHERE S.BranchNo = B.BranchNo
AND (S.Position = 'manager' AND B.City = 'London')
```



Poznámka: Symbol ⋈ představuje operátor spojení

Proces dekompozice

- V procesu dekompozice se dotaz
  - normalizuje kvůli jednodušší manipulaci,
  - sémanticky analyzuje,
  - zjednoduší (eliminují se redundantní podmínky),
  - vyjádří jako algebraický dotaz.
- Výsledkem je částečně optimalizovaný dotaz, který lze vyjádřit v určité formě stromu dotazu nad globálními relacemi.

Normalizace dotazu

- Konvertuje dotaz do normalizovaného tvaru. Predikát (v SQL podmínka WHERE) může být konvertován do konjunktivní nebo disjunktivní normální formy.
- Konjunktivní normální forma je konjunkce ($\wedge$) disjunkcí ($\vee$)

  $(p_{11} \vee p_{12} \vee \ldots \vee p_{1n}) \wedge \ldots \wedge (p_{m1} \vee p_{m2} \vee \ldots \vee p_{mn})$
- Disjunktivní normální forma je disjunkce ($\vee$) konjunkcí ($\wedge$)

  $(p_{11} \wedge p_{12} \wedge \ldots \wedge p_{1n}) \vee \ldots \vee (p_{m1} \wedge p_{m2} \wedge \ldots \wedge p_{mn})$
- Konjunktivní normální forma je „praktičtější“ – podmínky obvykle obsahují více AND než OR.

Příklad

- Konjunktivní normální forma

  (Position = ‘Manager’$\vee$Salary>50000) $\wedge$
  BranchNo=‘B003’

- Disjunktivní normální forma

  (Position = ‘Manager’ $\wedge$ BranchNo = ‘B003’) $\vee$
  (Salary>50000 $\wedge$ BranchNo = ‘B003’)

Zjednodušení

- Cílem zjednodušení dotazu je transformovat dotaz do sémanticky ekvivalentního tvaru, se kterým se lépe manipuluje.
- Na normální formy lze aplikovat známá pravidla idempotence boolovské algebry:

$p \wedge p \equiv p$	$p \vee p \equiv p$
$p \wedge \text{false} \equiv \text{false}$	$p \vee \text{false} \equiv p$
$p \wedge \text{true} \equiv p$	$p \vee \text{true} \equiv \text{true}$
$p \wedge (p \vee q) \equiv p$	$p \vee (p \wedge q) \equiv p$

  - Tato pravidla lze využít k eliminaci redundantních podmínek.

Příklad: Zjednodušení dotazu

Zjednodušte následující dotaz použitím pravidel idempotence.

```
SELECT ENO
FROM ASG
WHERE RESP = 'Analyst'
AND NOT(PNO='P2' OR DUR=12)
AND PNO =  'P2'
AND DUR=12
```

$\Pi_{ENO}(\sigma_{RESP='Analyst' \wedge \neg(PNO='P2' \vee DUR=12) \wedge PNO='P2' \wedge DUR=12}) ASG$

Negaci vnitřní závorky můžeme rozepsat:

$\neg(PNO='P2' \vee DUR=12) \equiv \neg (PNO='P2') \wedge \neg (DUR=12)$ a dostaneme:

$\neg (PNO='P2') \wedge \neg (DUR=12) \wedge PNO='P2' \wedge DUR=12$ – výsledek je FALSE

To znamená, že výsledek je prázdná množina.

Optimalizace dotazu

- Cílem optimalizace dotazu je transformovat dotaz do sémanticky ekvivalentního, ale efektivnějšího tvaru.
- Pro optimalizaci se používají transformační pravidla relační algebry, která optimalizátoru umožní transformovat výraz relační algebry do tvaru, o němž je známo, že je efektivnější.

Složitost operátorů relační algebry

Operation	Complexity
Select Project (without duplicate elimination)	$O(n)$
Project (with duplicate elimination) Group by	$O(n*\log n)$
Join Semijoin Division Set Operators	$O(n*\log n)$
Cartesian Product	$O(n^2)$

Složitost operátorů je relativní k počtu řádků relací *n*. Proto by operátory, redukující počet řádků měly být realizovány co nejdříve. Operátory by měly být vykonávány v pořadí od jednoduššího ke složitějšímu a kartézskému součinu bychom se měli vyhnout.
V distribuovaném prostředí lze využít SEMIJOIN na zmenšení velikosti přenášených dat.

Užitečná pravidla relační algebry (1)

Použitím vhodných transformačních pravidel relační algebry lze strom relační algebry restrukturalizovat.

V dalším předpokládejme, že R, S a T jsou relace kde

R má atributy $A=\{A_1, A_2, \ldots A_n\}$, S má atributy $B=\{B_1, B_2, \ldots B_n\}$

1. Komutativita binárních operací

$$R \times S \Leftrightarrow S \times R$$

$$R \bowtie S \Leftrightarrow S \bowtie R$$

$$R \cup S \Leftrightarrow S \cup R$$

2. Asociativita binárních operací

$$(R \times S) \times T \Leftrightarrow R \times (S \times T)$$

$$(R \bowtie S) \bowtie T \Leftrightarrow R \bowtie (S \bowtie T)$$

Užitečná pravidla relační algebry (2)

3. V posloupnosti projekcí se vyžaduje pouze poslední v pořadí

$$\Pi_{A'}(\Pi_{A''}(R)) \Leftrightarrow \Pi_{A'}(R)$$

kde $A' \subseteq A$, $A'' \subseteq A$ a $A' \subseteq A''$

4. Konjunktivní selekci lze transformovat na kaskádu selekcí

$$\sigma_{p \wedge q \wedge r}(R) \Leftrightarrow \sigma_p(\sigma_q(\sigma_r(R)))$$

5. Komutativita selekce a projekce

Obsahuje-li predikát selekce pouze atributy vyskytující se v projekci, tak operátory selekce a projekce jsou komutativní.

Užitečná pravidla relační algebry (3)

6. Distribuce selekce a binárních operací

$$\sigma_{p(A)}(R \times S) \Leftrightarrow (\sigma_{p(A)}(R)) \times S$$

$$\sigma_{p(A_i)}(R \bowtie_{(A_j,B_k)} S) \Leftrightarrow (\sigma_{p(A_i)}(R)) \bowtie_{(A_j,B_k)} S$$

$$\sigma_{p(A_i)}(R \cup T) \Leftrightarrow \sigma_{p(A_i)}(R) \cup \sigma_{p(A_i)}(T)$$

kde A_i patří do R i do T.

7. Distribuce projekce a binárních operací

$$\Pi_C(R \times S) \Leftrightarrow \Pi_{A'}(R) \times \Pi_{B'}(S)$$

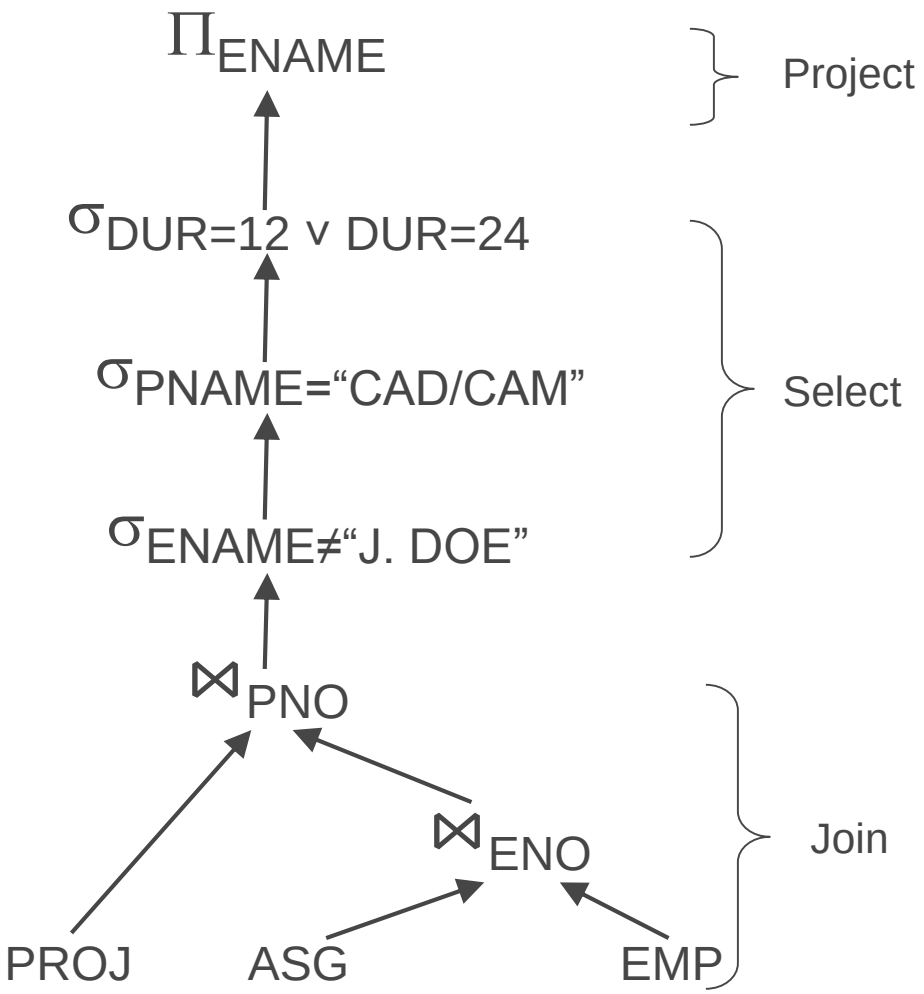
$$\Pi_C(R \bowtie_{(A_j,B_k)} S) \Leftrightarrow \Pi_{A'}(R) \bowtie_{(A_j,B_k)} \Pi_{B'}(S)$$

$$\Pi_C(R \cup S) \Leftrightarrow \Pi_C(R) \cup \Pi_C(S)$$

kde $C = A' \cup B'$, $A' \subseteq A, B' \subseteq B$

Příklad

SELECT ENAME
FROM PROJ, ASG, EMP
WHERE
ASG.ENO=EMP.ENO
AND ASG.PNO=PROJ.PNO
AND ENAME ≠ "J. Doe"
AND PROJ.PNAME="CAD/CAM"
AND (DUR=12 **OR** DUR=24)

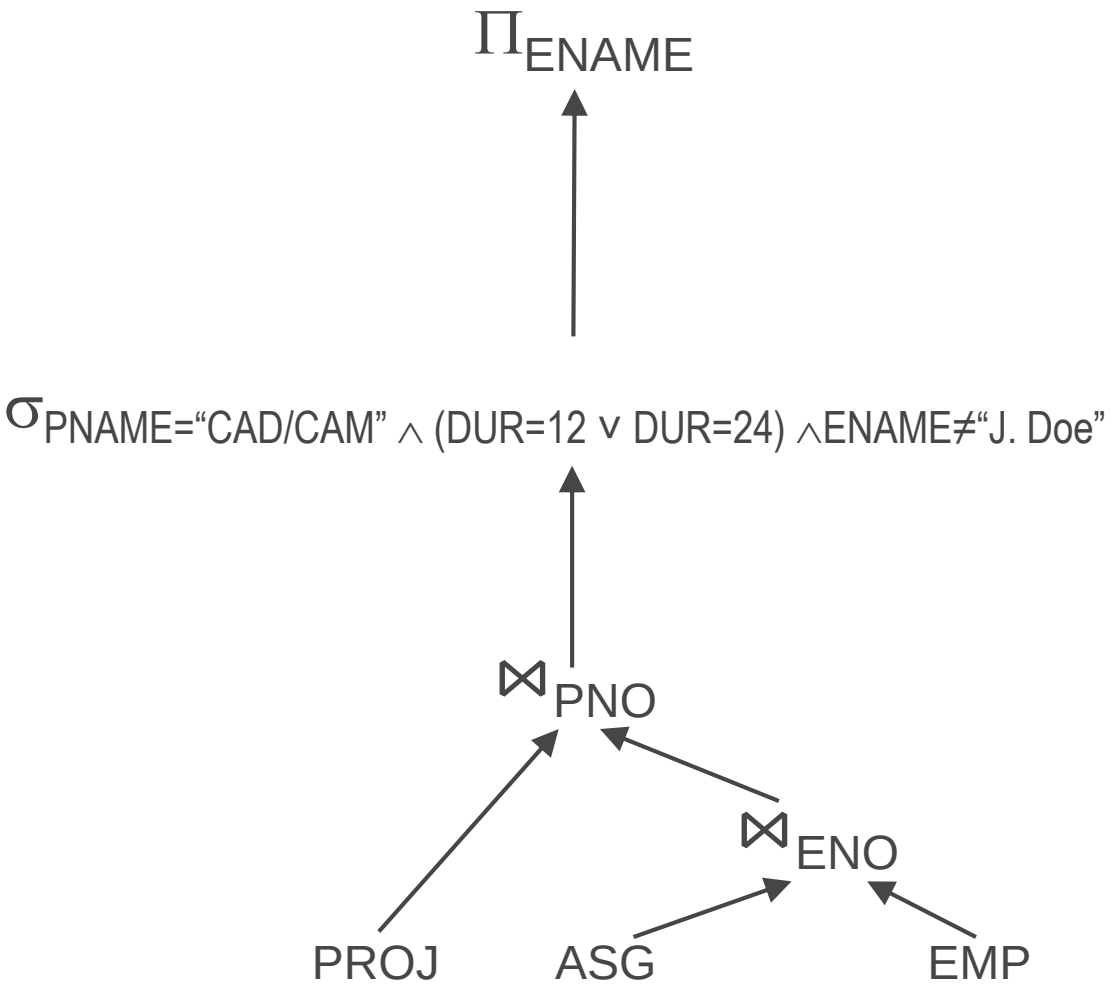


Příklad – ekvivalentní tvar dotazu

- Kaskáda selekcí je nahrazena konjunktivní selekcí.
- V dalším kroku lze „posunout selekce dolů“, aby se zmenšily kardinality relací.
- Abychom posunuli selekce dolů, budeme aplikovat pravidlo o distributivnosti selekce a spojení:

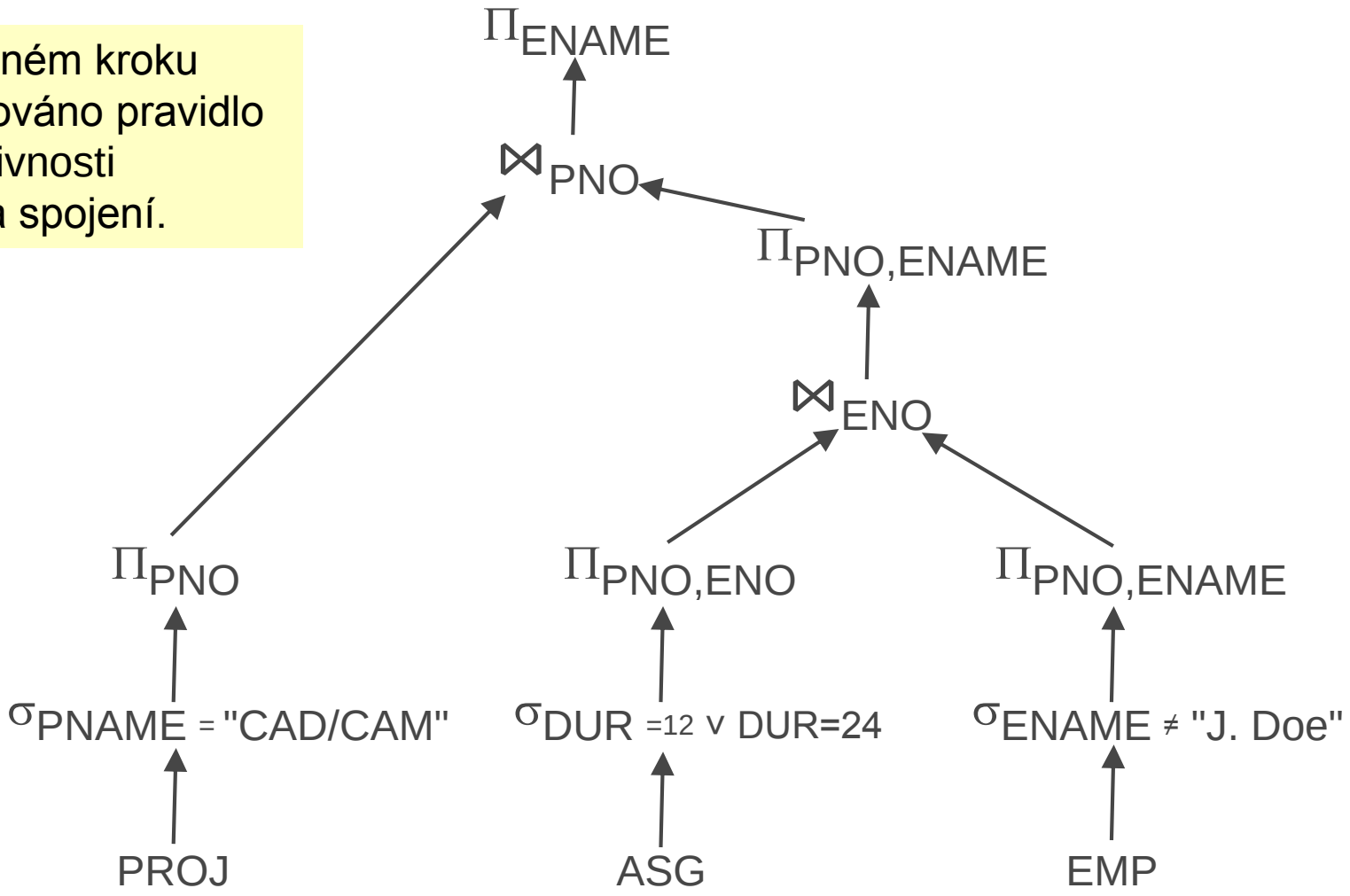
  Pokud je predikát selekce tvaru $p \wedge q$, tak operace selekce a spojení je distributivní, tj.

  $\sigma_{p \wedge q}(R \bowtie S) = (\sigma_p(R)) \bowtie (\sigma_q(S))$



Restrukturalizovaný dotaz

V závěrečném kroku bylo aplikováno pravidlo o distributivnosti projekce a spojení.



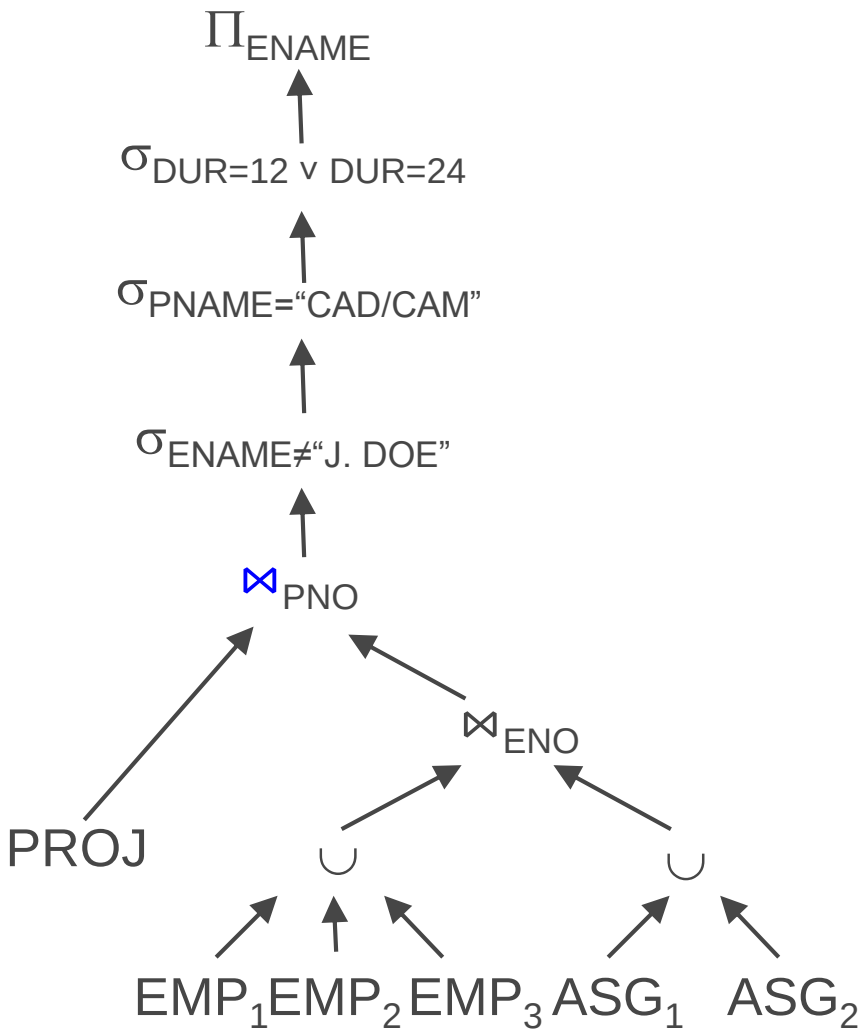
Vrstva lokalizace dat

- Tato vrstva bere v úvahu distribuci dat. Postará se o další iteraci optimalizace tím, že nahradí globální relace na listech stromu jejich vyjádřením pomocí fragmentů.
  - Pro horizontální fragmentaci na rekonstrukci globální relace použijeme operátor UNION.
  - Pro vertikální fragmentaci použijeme na rekonstrukci operátor JOIN.
- Poté použijeme redukci, abychom vytvořili jednodušší a optimalizovaný dotaz.

Příklad

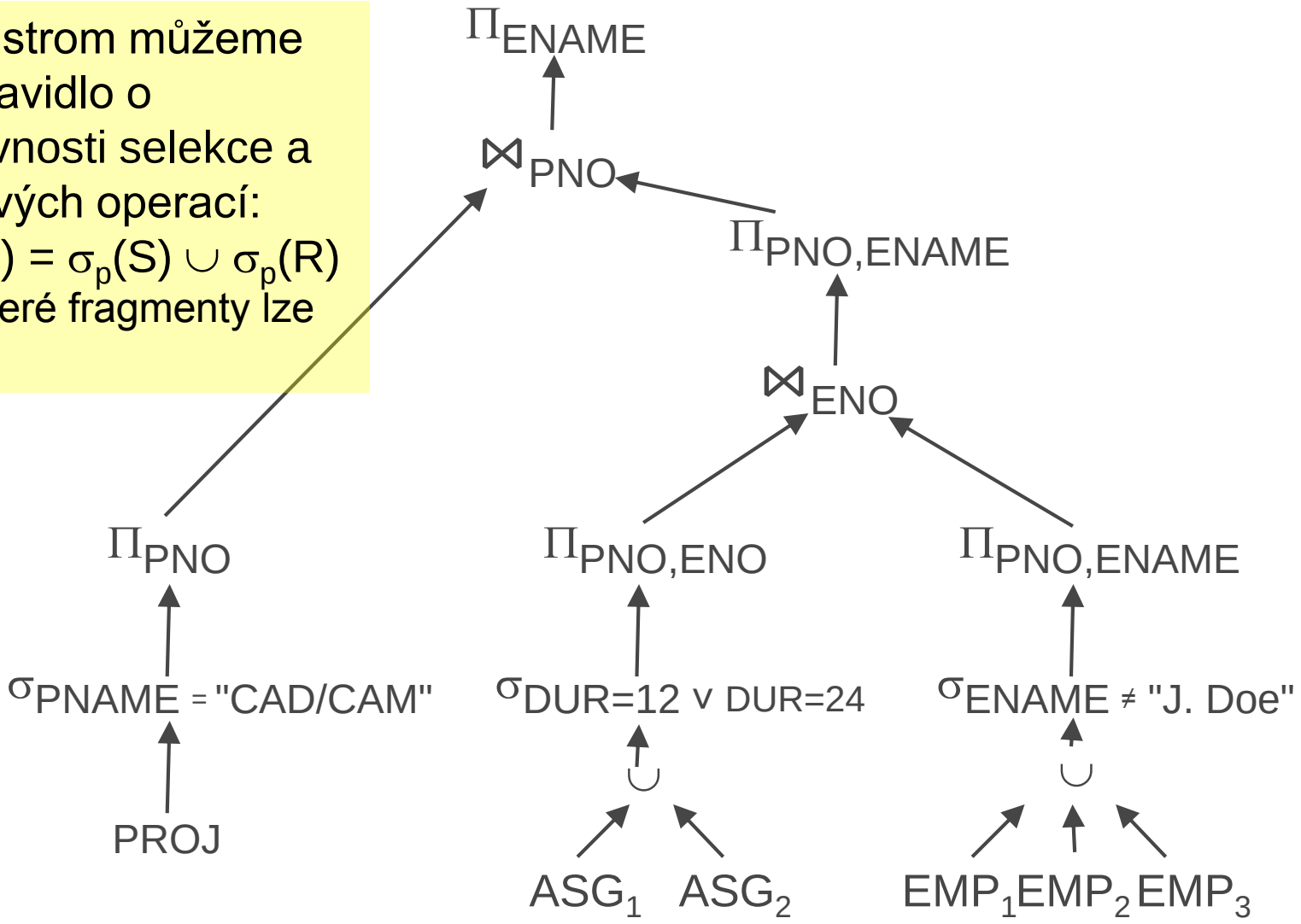
Předpokládejme, že relace z předchozího příkladu jsou fragmentované.

- EMP je fragmentována:
  - $EMP_1 = \sigma_{ENO \leq \text{“E3”}}(EMP)$
  - $EMP_2 = \sigma_{\text{“E3”} < ENO \leq \text{“E6”}}(EMP)$
  - $EMP_3 = \sigma_{ENO > \text{“E6”}}(EMP)$
- ASG je fragmentována:
  - $ASG_1 = \sigma_{ENO \leq \text{“E3”}}(ASG)$
  - $ASG_2 = \sigma_{ENO > \text{“E3”}}(ASG)$
- Pro každý typ fragmentace existují redukční techniky, které vytváří jednodušší a optimalizované dotazy.

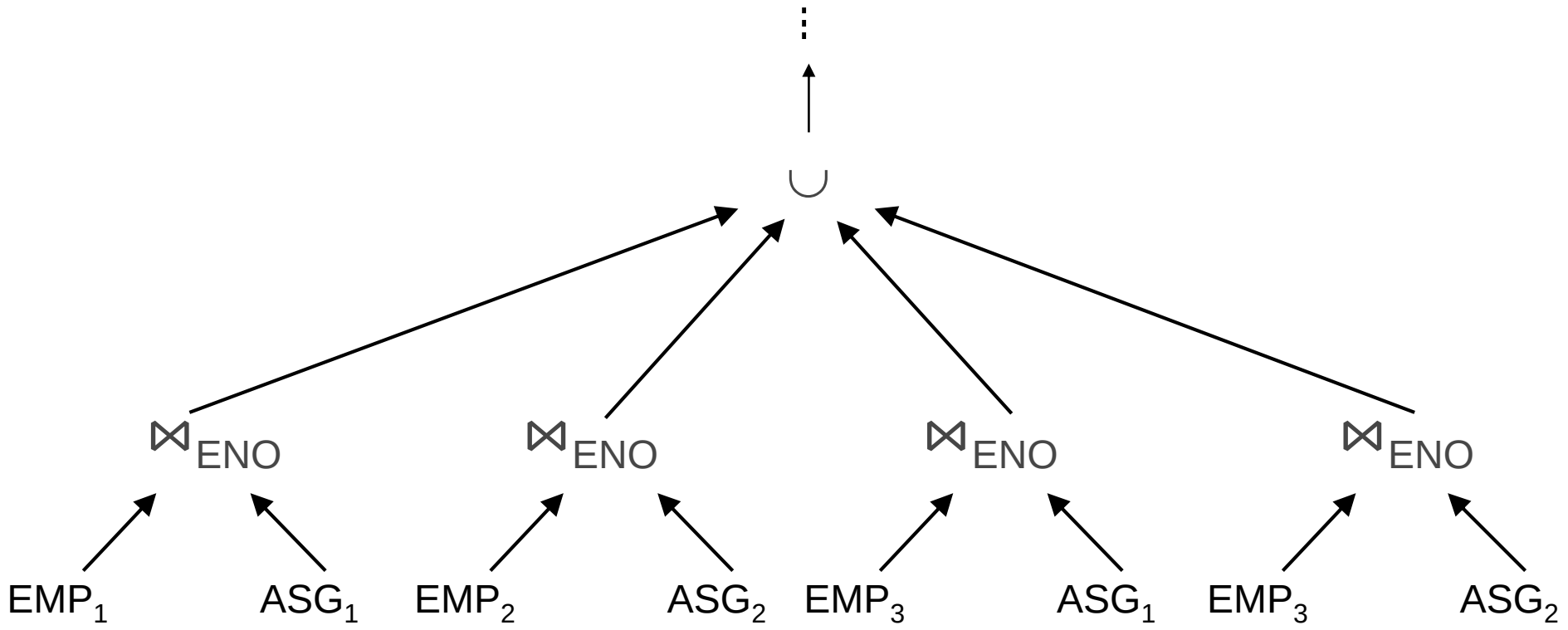


Restrukturalizovaný dotaz

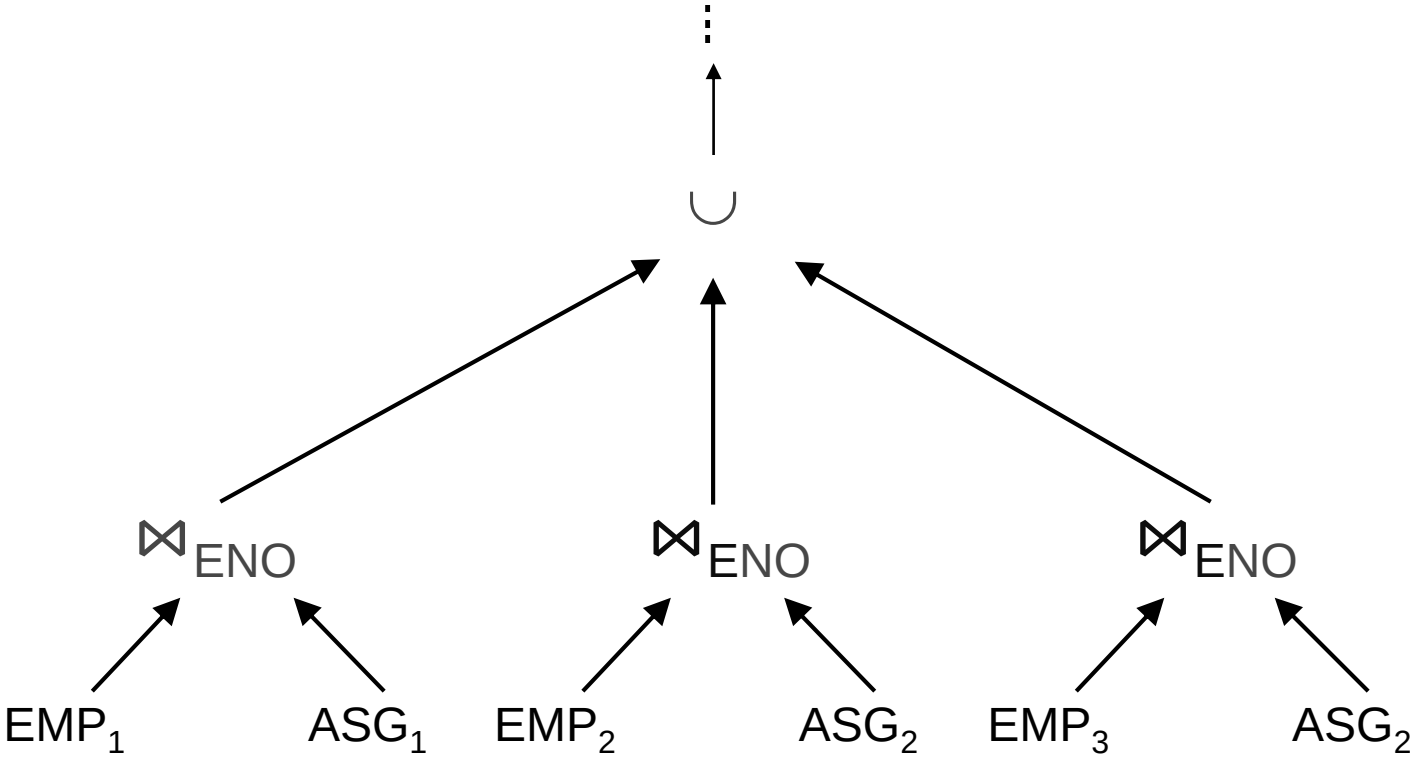
Na tento strom můžeme použít pravidlo o distributivnosti selekce a množinových operací:
$\sigma_p(R \cup S) = \sigma_p(S) \cup \sigma_p(R)$
a zjistit, které fragmenty lze vynechat.



Poskytuje paralelizmus



Eliminuje



Typy redukce

- Typ redukce, který je vhodné použít je závislý na typu fragmentace:
  - redukce pro primární horizontální fragmentaci
  - redukce pro vertikální fragmentaci
  - redukce pro odvozenou fragmentaci
  - redukce pro hybridní fragmentaci

Redukce pro primární horizontální fragmentaci

Horizontální fragmentace je vytvořena pomocí predikátu selekce.

Redukce dotazů nad horizontálně fragmentovanými relacemi je založená hlavně na restrukturalizaci dotazu a určení, které fragmenty jsou nadbytečné a lze je vynechat. Horizontální fragmentaci lze využít pro na zjednodušení selekce i spojení.

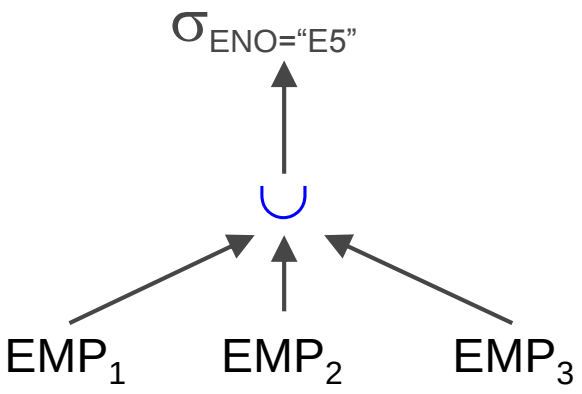
- Redukce pro operátor selekce
  - Když je operátor selekce v kontradikci s definicí fragmentu, tak získáme jako mezivýsledek prázdnou relaci a tak můžeme tyto operace vynechat.
- Redukce pro operátor spojení.
  - Použitelné v případě fragmentace na základě společného atributu
  - Nejdříve použijeme transformační pravidlo o distributivnosti spojení a sjednocení: $(R_1 \cup R_2) \bowtie R_3 = (R_1 \bowtie R_3) \cup (R_2 \bowtie R_3)$.
  - Potom prověříme každé spojení, zda tam nejsou redundantní operace které lze eliminovat.
  - Redundantní spojení existují v případě, když se predikáty fragmentů nepřekrývají.

Redukce pro PHF - příklad

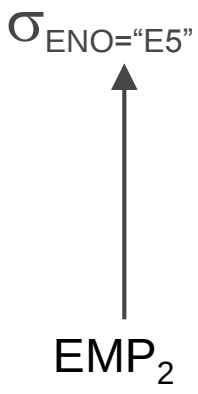
- Redukce na základě selekce
  - Příklad

```
SELECT *
FROM EMP
WHERE  ENO="E5"
```

Lokalizovaný dotaz:

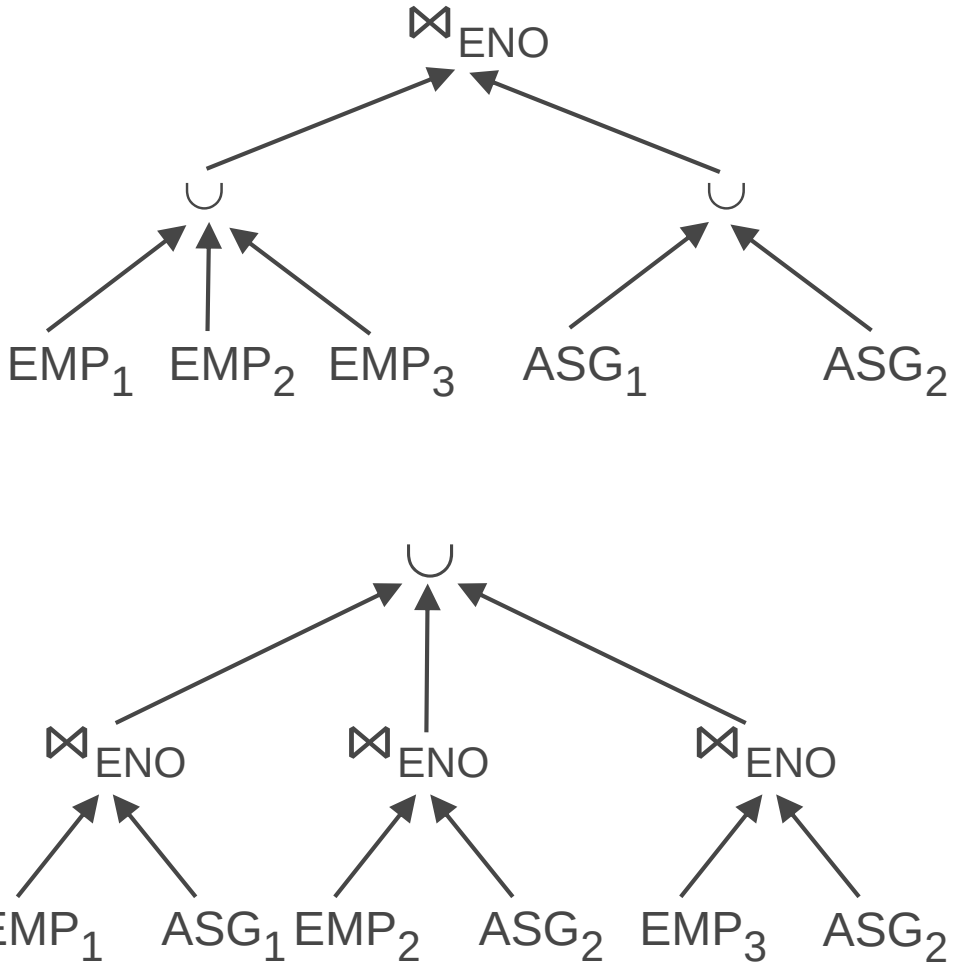


Redukovaný dotaz:



Redukce pro PHF - příklad

- Redukce na základě spojení
  - EMP fragmentováno jako předtím
  - ASG také fragmentováno
    - ASG_1: $\sigma_{ENO \leq "E3"}(ASG)$
    - ASG_2: $\sigma_{ENO > "E3"}(ASG)$
  - **SELECT** *
    FROM EMP,ASG
    WHERE
    EMP.ENO=ASG.ENO



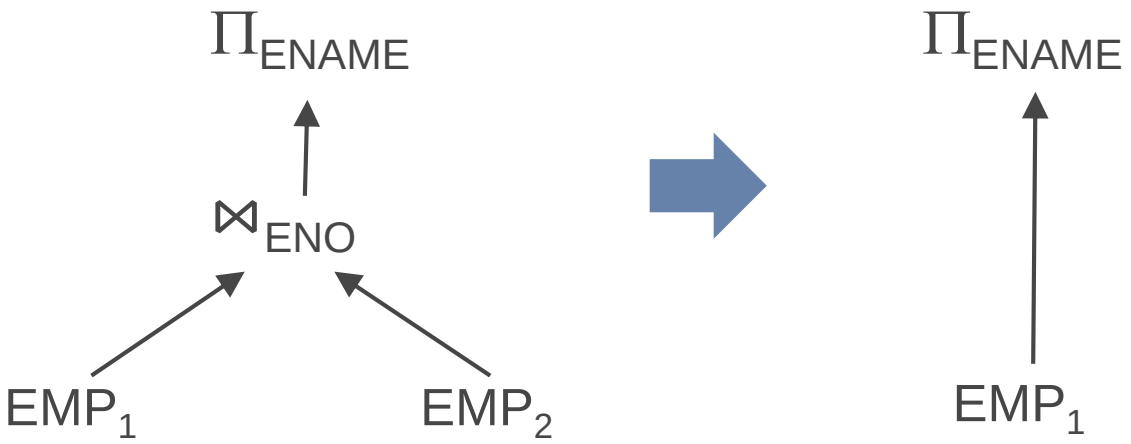
Redukce pro vertikální fragmentaci

Redukce pro vertikální fragmentaci spočívá v tom, že odstraní ty vertikální fragmenty, které kromě primárního klíče nemají žádné společné atributy s atributy projekce.

Příklad:

$EMP_1 = \Pi_{ENO,ENAME}(EMP)$
$EMP_2 = \Pi_{ENO,TITLE}(EMP)$

SELECT ENAME
FROM EMP



Redukce pro odvozenou HF

- Redukce pro odvozenou horizontální fragmentaci používá pravidlo o distribuci spojení nad sjednocením.
- V tomto případě využíváme znalost toho, že fragmentace jedné relace je založená na jiné relaci a při výměně pořadí operací některé z parciálních sjednocení mohou být redundantní.
- Postup :
  - Distribuce spojení nad sjednocením
  - Aplikace join redukce pro horizontální fragmentaci

Redukce pro DHF - příklad

- Příklad

  ASG_1: $ASG \ltimes_{ENO} EMP_1$

  ASG_2: $ASG \ltimes_{ENO} EMP_2$

  EMP_1: $\sigma_{TITLE=\text{“Programmer”}}(EMP)$

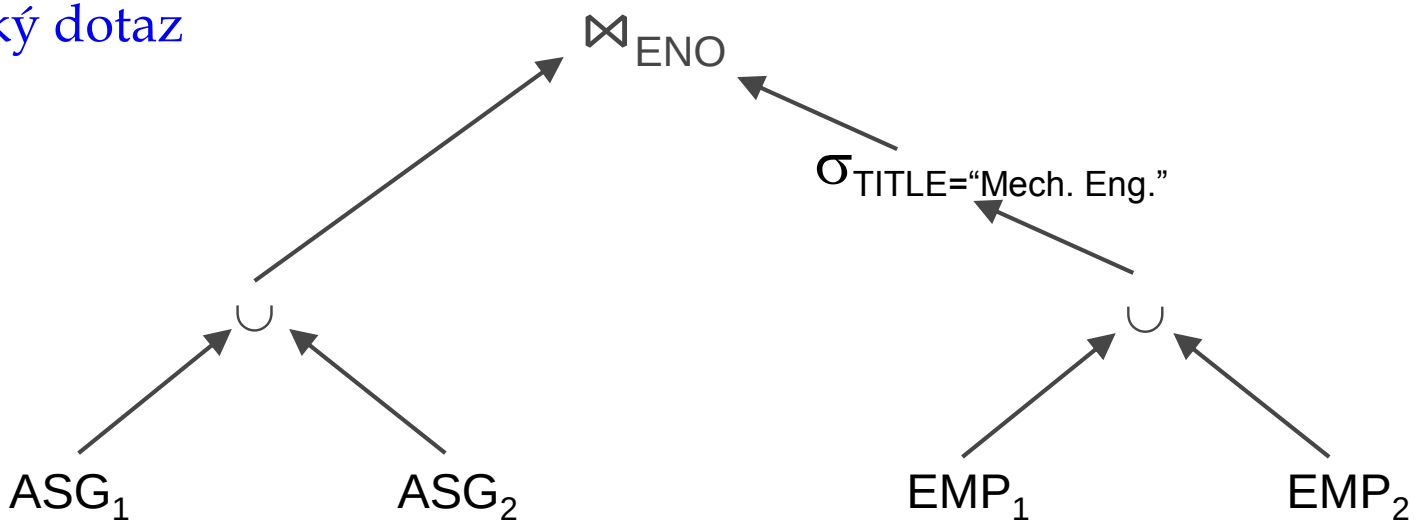
  EMP_2: $\sigma_{TITLE\neq\text{“Programmer”}}(EMP)$

- Dotaz

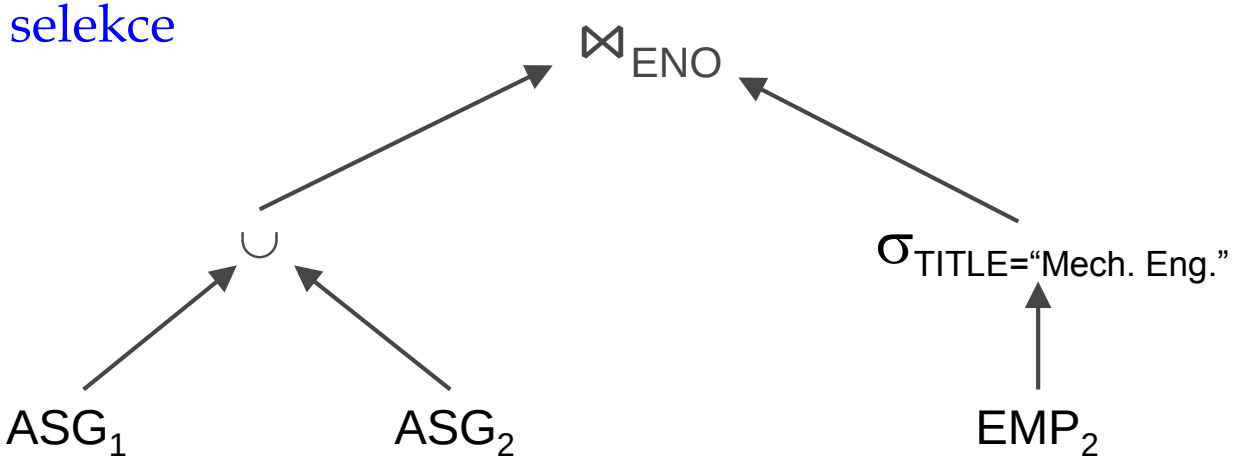
```
SELECT   *
FROM     EMP, ASG
WHERE    ASG.ENO = EMP.ENO
AND      EMP.TITLE = "Mech. Eng."
```

Redukce pro DHF – řešení příkladu

Generický dotaz

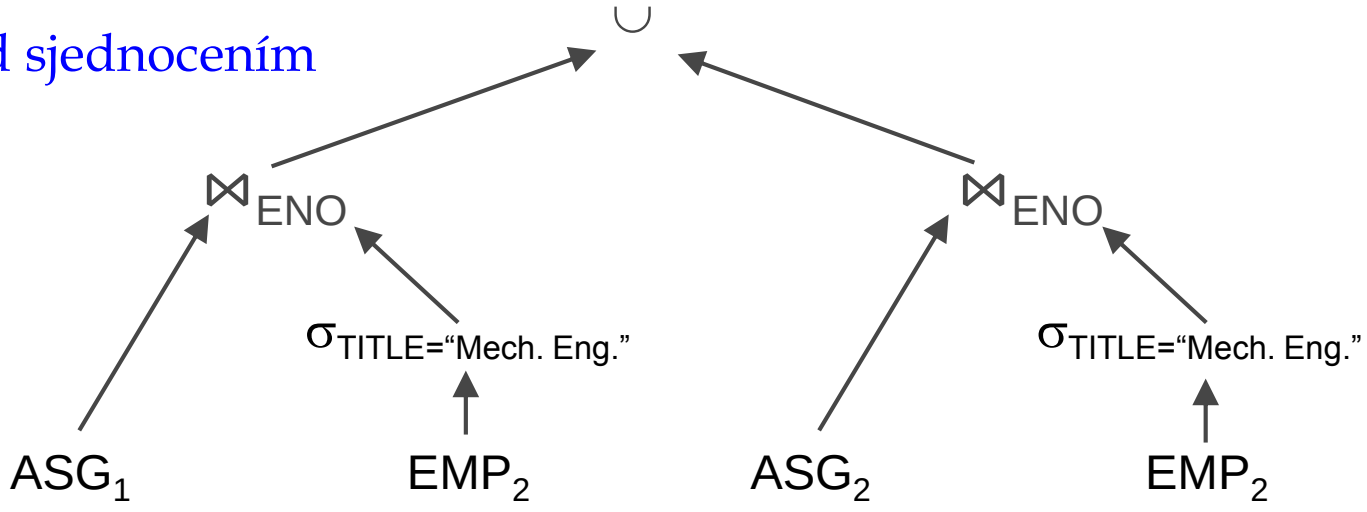


Nejdříve selekce

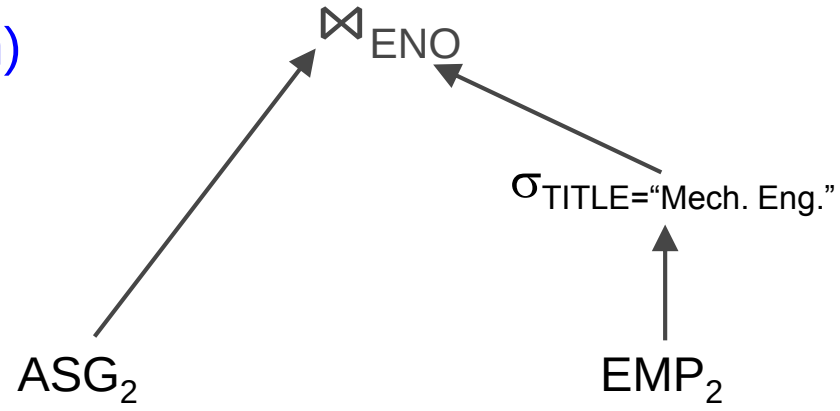


Redukce pro DHF

Spojení nad sjednocením



Vynecháme prázdné relace (levý podstrom)

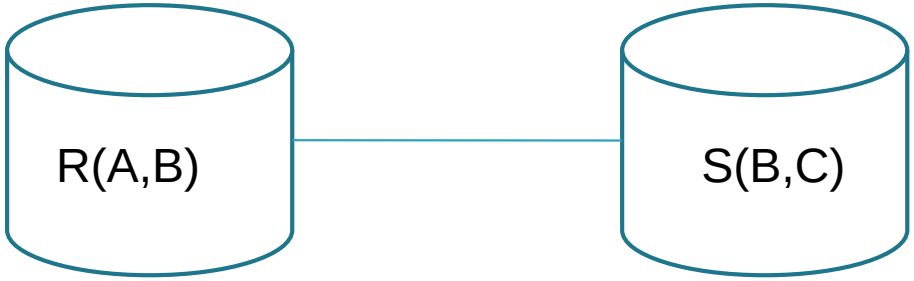


Spojení v distribuované databázi

- Operace spojení (join) je nejnákladnější z operací relační algebry.
- Jeden z přístupů k optimalizaci distribuovaných dotazů nahrazuje spojení kombinací polospojení (semi-join).
- Polospojení má důležitou vlastnost, že redukuje velikost operandu a snižuje objem přenášených dat.

Výpočet přirozeného spojení

- Máme vypočítat R(A,B) ⋈ S(B,C)



Máme 2 možnosti:

- poslat kopii R na místo, kde je S a tam vypočítat přirozené spojení,
- poslat kopii S na místo, kde je R a tam vypočítat přirozené spojení.

Pro mnoho situací obě možnosti jsou přijatelné.

Problémy

- Co když propojení má malou kapacitu?
- I když kapacita propojení je dostatečná, co když B obsahuje například pouze identifikátory a názvy videí a C obsahuje samotná videa?
- V obou případech lze použít takový postup vyhodnocení dotazu, při kterém se pošle pouze relevantní část příslušné relace.

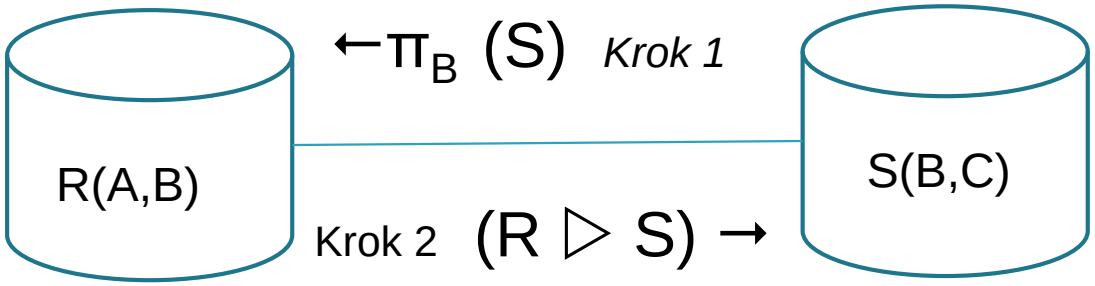
Polospojení relací (semijoin ▷)

- Polospojení relací R a S (tj. R ▷S) je množina n-tic relace R, pro které platí, že existuje alespoň jedna n-tice v S, která má stejné hodnoty společných atributů.
- R ▷ S je relace, která obsahuje n-tice z R, které lze spojit s S.
- $R \triangleright S = \pi_R (R \bowtie S)$
- R ▷ S eliminuje n-tice R, které nelze spojit s S

Redukce použitím polospojení (1)

- Máme R(A,B), S(B,C)
- $R \triangleright S = \pi_{A,B} (R \bowtie S) = R \bowtie (\pi_B (S))$
- To znamená, že vytvoříme projekci S na společné atributy a potom vytvoříme přirozené spojení této projekce s R
- Pokud pošleme $\pi_B (S)$ na místo, kde je R, můžeme tam vypočítat $R \triangleright S$. Víme, že ty řádky R, které nejsou v $R \triangleright S$ nemohou participovat v $(R \bowtie S)$. Proto stačí poslat $R \triangleright S$ na místo kde je S a tam vypočítat přirozené spojení.

Redukce použitím polospojení (2)



Zda je tento postup výhodnější, závisí na několika faktorech. Pokud projekce S na atributy B vytvoří relaci mnohem menší než je S, tak je levnější poslat $\pi_B(S)$ na R než posílat celé S. Toto platí v případě, že

- Mnoho řádků S má stejné hodnoty B
- Komponenty C jsou mnohem větší než komponenty B.

Abychom mohli tvrdit, že použití polospojení je výhodnější, tak R musí mít mnoho řádků, které nelze s ničím spojit.

Příklad

Předpokládejme, že potřebujeme vytvořit $R(A,B) \bowtie S(B,C)$, přičemž R a S jsou na různých místech sítě a cena za přenos dat po síti je dominantní složkou nákladů na vytvoření. Dále předpokládáme:

1. velikost R je v_R
2. velikost S je v_S
3. velikost $\pi_B(R)$ je p_R a je to pouze zlomek v_R
4. velikost $\pi_B(S)$ je p_S a je to pouze zlomek v_S
5. počet řádků relace R, které nejde spojit s S je n_R
6. počet řádků relace S, které nejde spojit s R je n_S

Příklad (pokrač.)

- Máme vyjádřit (použitím výše zmíněných parametrů) cenu každé z následujících možných strategií a určit, za jakých podmínek bude strategie vhodná.
  a) Poslat R na místo kde je S.
  b) Poslat S na místo kde je R.
  c) Poslat $\pi_B(S)$ na místo kde je R a potom poslat R $\triangleright$ S na místo kde je S.
  d) Poslat $\pi_B(R)$ na místo kde je S a potom poslat S $\triangleright$R na místo kde je R.

Řešení

a) v_R
- když $v_R < v_S$ a $v_R < p_S + (v_R - n_R)$ je vhodné provést spojení na místě kde je S

b) v_S
- Když $v_R > v_S$ a $v_S < p_R + (v_S - n_S)$ je vhodné provést spojení na místě, kde je R

c) $p_S + (v_R - n_R)$
- Když $v_R < v_S$ a $v_R > p_S + (v_R - n_R)$

d) $p_R + (v_S - n_S)$
- když $v_R > v_S$ and $v_S > p_R + (v_S - n_S)$