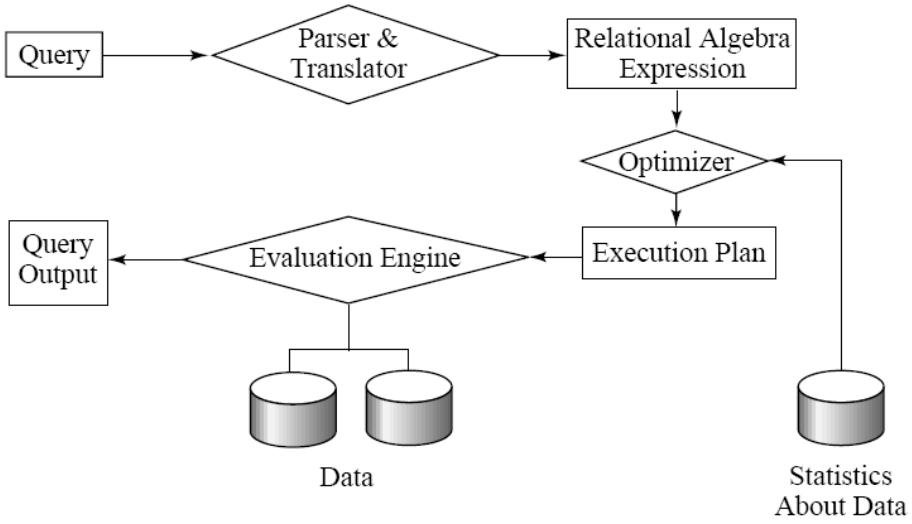
Kapitola 12: Zpracování dotazů

- Přehled
- Informace pro odhad nákladů
- Míry pro náklady dotazu
- Operace výběru
- Řazení
- Operace spojení
- Vyhodnocování výrazů
- Transformace relačních výrazů
- Výběr plánu pro vyhodnocení

Základní kroky ve zpracování dotazů

1. analýza dotazu (parsing) a překlad
2. optimalizace
3. vyhodnocení



Analýza dotazu (parsing) a překlad:

- Přelož dotaz do interní reprezentace, která je následně přeložena do relační algebry
- Analyzátor ověří správnou syntaxi a ověří existenci relací

Vyhodnocení:

- **Stroj pro vyhodnocení dotazu** (query-execution/evaluation engine) bere na vstupu plán pro vyhodnocení dotazu, spustí plán a vrátí výsledky dotazu.

Optimalizace – nalezení nejlevnějšího plánu pro vykonání dotazu:

- Mějme výraz v relační algebře, tento výraz může mít několik ekvivalentních (produkují stejný výsledek) výrazů
  Např. $\sigma_{\text{zůstatek<2500}}(\Pi_{\text{zůstatek}}(\textit{účet}))$ je ekvivalentní výrazu:
  $\Pi_{\text{zůstatek}}(\sigma_{\text{zůstatek<2500}}(\textit{účet}))$
- Jakýkoli výraz v relační algebře může být vyhodnocen mnoha způsoby. Komentovaný výraz určující detailně postup vyhodnocení se nazývá *plán pro vyhodnocení.*

Např. má se použít index na atributu *zůstatek* k nalezení účtů se zůstatkem < 2500, nebo se má použít sekvenční průchod celého souboru a vynechat všechny účty s zůstatkem ≥ 2500?

- Mezi všemi možnými výrazy se snažíme najít ten, který má nejlevnější plán pro vyhodnocení. Odhad ceny plánu pro vyhodnocení je založený na *statistických informacích* v databázovém katalogu.

Informace v DB katalogu pro odhad nákladů

- n_r: počet n-tic relaci *r*
- b_r: počet bloků obsahující n-tice z relace *r*
- s_r: velikost n-tice z *r* v bajtech
- f_r: blokovací faktor relace *r* – tj. počet n-tic z *r*, které se vejdou do jednoho bloku
- *V(A,r)*: počet jedinečných hodnot, které se vyskytují v relaci *r* v atributu *A*, což je stejné jako velikost $\Pi_A(r)$
- *SC(A,r)*: (selection cardinality) průměrný počet záznamů, které mají stejnou hodnotu na atributu *A*
- Pokud jsou n-tice relace *r* uloženy společně v jednom souboru, potom:

$$b_r = \lceil n_r / f_r \rceil$$

- f_i: průměrný počet potomků vnitřního uzlu ve stromovém indexu *i* jako např. B^+-strom
- HT_i: počet úrovní stromového indexu *i*, tj. hloubka stromu
  - Pro vyvážený strom na atributu *A* relace *r*: $HT_i = \lceil log_{fi}(V(A,r)) \rceil$
  - Pro hešovací index je $HT_i = 1$
- LB_i: počet bloků na nejnižší úrovni indexu *i*, tj. počet bloků v listech stromu

Míry nákladů dotazů

- Mnoho způsobů jak měřit a odhadovat náklady (cenu), např. počet diskových přístupů, CPU čas nebo dokonce komunikační režie v distribuovaných nebo paralelních systémech.
- Přístupy na disk typicky tvoří převládající náklady a také jsou relativně snadno měřitelné. Tudíž, počet přenosů bloků z disku je používán jako míra pro aktuální cenu vyhodnocení. Předpokládá se, že všechny přístupy mají stejnou cenu.
- Náklady algoritmů závisí na velikosti vyrovnávacích pamětí v operační paměti, protože více paměti snižuje náklady na čtení z disku. Tedy velikost paměti by měl být parametr pro odhad ceny dotazu; často se používá nejhorší odhad.
- Odhad nákladů algoritmu *A* je značen jako E_A. Náklady pro zápis na disk nejsou uvažovány.

Operace výběru

- **Sekvenční průchod souborem** – vyhledávací algoritmus, který hledá a vrací záznamy vyhovující podmínce
- Algoritmus **A1** (lineární hledání) – postupně projdi každý blok v souboru a otestuj všechny záznamy v bloku, jestli splňují podmínku výběru.
  - Odhad nákladů (počet čtení bloku z disku) $E_{A1} = b_r$

  - Jestli je podmínka na atributu, který je vyhledávacím klíčem, pak $E_{A1}=(b_r/2)$ (skonči při nalezení záznamu)
  - Lineární hledání může být aplikováno bez ohledu na
    * Výběrovou podmínku
    * Pořadí záznamů v souboru
    * Dostupnosti indexů
- Algoritmus **A2** (binární hledání) – použitelné, pokud výběrová podmínka je operátor rovnosti na atributu, který je klíčem.
  - Předpokládejme, že bloky souboru jsou uloženy spojitě za sebou
  - Odhad ceny (počet bloků přečtených z disku):

$$E_{A2} = \lceil log_2(b_r) \rceil + \lceil SC(A,r) / f_r \rceil - 1$$

    * $\lceil log_2(b_r) \rceil$ - náklady na nalezení první n-tice binárním hledáním
    * $SC(A,r)$ – počet záznamů splňující podmínku
    * $\lceil SC(A,r) / f_r \rceil$ - počet bloků, které obsahují tyto záznamy
  - podmínka rovnosti na atributu, který je klíč: $SC(A,r) = 1$; odhad je pak $E_{A2}=\lceil log_2(b_r) \rceil$

Statistické informace použité v příkladech

- $f_{účet} = 20$ (20 n-tic relace *účet* v jednom bloku)
- $V(jméno\text{-}pobočky, účet) = 50$ (50 poboček)
- $V(zůstatek, účet) = 500$ (500 různých hodnot *zůstatku*)
- $n_{účet} = 10000$ (10000 účtů)
- nechť existují následující indexy na relaci *účet:*
  - primární: B^+-strom na atributu *jméno-pobočky*
  - sekundární: B^+-strom na atributu *zůstatek*

Příklad dohadu operace výběru

$$\sigma_{jméno\text{-}pobočky=\text{“}Perryridge\text{“}}(účet)$$

- Počet bloků je $b_{účet} = 500$; 10000 n-tic v relaci; každý blok obsahuje 20 záznamů
- Předpokládejme, že relace *účet* je seřazena na atributu *jméno-pobočky*
  - $V(jméno\text{-}pobočky, účet) = 50$
  - 10000/50 = 200 n-tic v relaci *účet*, které splňují podmínku *jméno-pobočky=“Perryridge“*
  - 200/20 = 10 bloků, které obsahují tyto záznamy
  - binárním hledáním najdeme první záznam s $\lceil log_2(500) \rceil=9$ přístupy na disk
- Celkové náklady binárního hledání jsou 9+10-1 = 18 čtení bloku (v porovnání s 500 při lineárním průchodu souboru)

Výběr s použitím indexů

- **Indexové hledání** – vyhledávací algoritmus používající index; podmínka je na vyhledávacím klíči indexu
- **A3** (primární index na kandidátním klíči, rovnost) – vybírá jediný záznam splňující podmínku rovnosti: $E_{A3} = HT_i + 1$
- **A4** (primární index na neklíčovém atributu, rovnost) – vybírá více záznamů, vyhledávací klíč je A: $E_{A4} = HT_i + \lceil SC(A,r) / f_r \rceil$

- **A5** (rovnost na vyhledávacím klíči se sekundárním indexem)
  - Vrací jediný záznam, pokud je vyhledávací klíč kandidátním klíčem $E_{A5}=HT_i + 1$
  - Vrací více záznamů (každý může být v jiném bloku), pokud vyhledávací klíč není klíčem relace: $E_{A5}=HT_i + SC(A,r)$

Implementace složitých výběrů

- **Selektivita** podmínky θ_i je pravděpodobnost, že n-tice v relaci *r* splňuje θ_i. Pokud s_i je počet záznamů splňující tuto podmínku, potom selektivita je s_i / n_r.
- **Konjunkce:** $\sigma_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \theta n}(r)$. Odhad počtu n-tic ve výsledku:

$$n_r * \frac{s_1 * s_2 * \ldots * s_n}{n_r^n}$$

- **Disjunkce:** $\sigma_{\theta 1 \vee \theta 2 \vee \ldots \vee \theta n}(r)$. Odhad počtu n-tic ve výsledku:

$$n_r * \left(1 - \left(1 - \frac{s_1}{n_r}\right) * \left(1 - \frac{s_2}{n_r}\right) * \ldots * \left(1 - \frac{s_n}{n_r}\right)\right)$$

- **Negace:** $\sigma_{\neg\theta}(r)$. Odhad počtu n-tic ve výsledku:

$$n_r - size\left(\sigma_\theta(r)\right)$$

Řazení

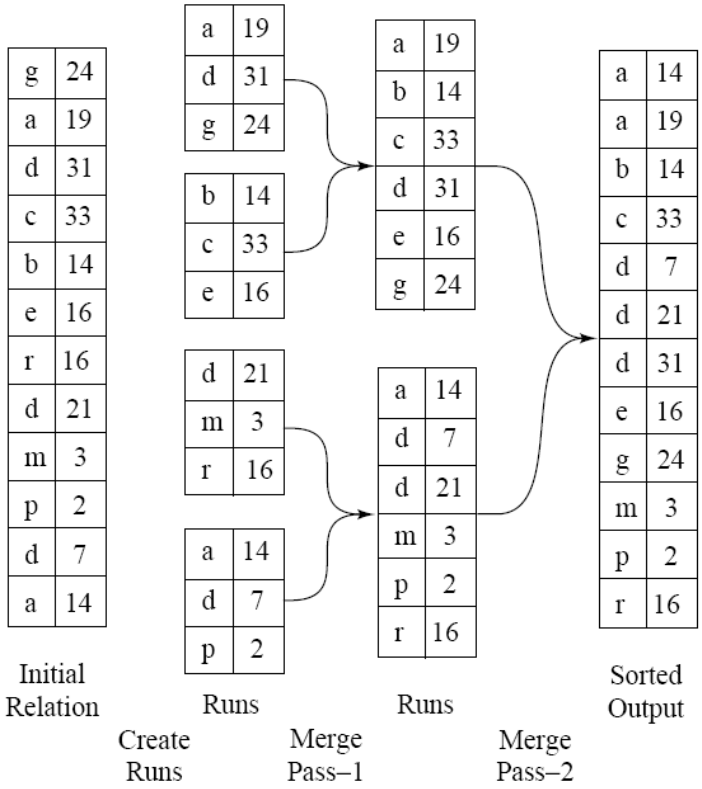
- Můžeme vytvořit index na relaci a poté jej využít ke čtení relace v uspořádaném pořadí. Protože index uspořádává záznamy „logicky“ a ne fyzicky (kde pořadí záznamů může být velmi odlišné), potom vytvoření indexu a jeho procházení může vést k jednomu čtení bloku z disku pro každý záznam.
- Pro relace, které se vejdou do paměti, techniky jako quicksort mohou být použity. Pro relace, které nelze celé načíst do paměti, je vhodnou volbou **externí sort-merge** (seřaď a spoj).

Externí sort-merge

Nechť *M* je velikost paměti ve stránkách (odpovídají blokům):

1. Vytvoř dávky (runs) pro řazení. Nechť $i = 0$. Opakuj, dokud není zpracována celá relace:
   a. Načti *M* bloků relace do paměti
   b. Setřiď načtené bloky
   c. Ulož seřazená data do dávky R_i a zvyš *i*
2. Spoj dávky; pro jednoduchost předpokládejme, že $i < M$. V jednom spojovacím kroku, použij *i* bloků paměti pro načtení vstupních dávek a *1* blok jako výstup. Opakuj následující, dokud nejsou všechny vstupní vyrovnávací paměti prázdné:
   a. Z jednotlivých seřazených bloků v paměti vyber nejmenší záznam
   b. Ulož vybraný záznam do výstupního bloku

c. Smaž vybraný záznam ze vstupního bloku; pokud je blok již prázdný, načti nový blok dávky z disku.



- Je-li $i \geq M$, je nutné provést několik spojovacích kroků.
  - V každém průchodu je *M-1* blízkých skupin dávek spojeno
  - Jeden průchod sníží počet dávek poměrem *M-1* a vytvoří nové dávky delší o stejný poměr
  - Opakujeme dokud nejsou všechny dávky spojeny do jedné
- Analýza nákladů
  - Počáteční vytvoření dávek stejně jako každý průchod slučování vyžaduje $2b_r$ diskových přístupů (kromě posledního průchodu, který nezapisuje výsledek na disk)
  - Celkový počet průchodů je: $\lceil \log_{M-1}(b_r / M) \rceil$

Celkový počet diskový přístupů je: $b_r (2\lceil \log_{M-1}(b_r / M) \rceil + 1)$

Operace spojení

- Několik různých algoritmů pro implementaci spojení (join)
  - Vnořené cykly (nested loops join)
  - Blokované vnořené cykly (block nested loops join)
  - Indexované vnořené cykly
  - Slučované spojení (Merge-join)
  - Hešované spojení
- Volba je založena na odhadu nákladů

Příklad

$$vkladatel \bowtie zákazník$$

Databázový katalog nám poskytuje údaje:

- $n_{zákazník}=10000$
- $f_{zákazník}=25,$ což implikuje, že $b_{zákazník}=10000/25=400$
- $n_{vkladatel}=5000$
- $f_{vkladatel}=50,$ což implikuje, že $b_{vkladatel}=5000/50=100$
- $V(jméno\text{-}zákazníka,vkladatel)=2500,$ což implikuje, že v průměru má každý zákazník dva účty.

Zde předpokládáme, že *jméno-zákazníka* v relaci *vkladatel* je cizí klíč do relace *zákazník.*

Odhady ceny spojení

- Kartézský součin $r \times s$ obsahuje $n_r n_s$ n-tic, každá n-tice je s_r+s_s bajtů dlouhá.
- Pokud $R \cap S = \varnothing$, potom $r \bowtie s$ je stejný jako $r \times s$
- Pokud $R \cap S$ obsahuje klíč schématu R, potom n-tice z relace s je spojena s nejvýše jednou n-ticí relace r; tudíž, počet n-tic ve spojení $r \bowtie s$ není větší než počet n-tic v s.
  Pokud $R \cap S$ v S je cizí klíč odkazující do R, potom počet n-tic ve spojení $r \bowtie s$ je přesně počet n-tic v s.
  Opačný případ, kdy $R \cap S$ je cizí klíč odkazující do S je symetrický.
- V příkladu dotazu *vkladatel* $\bowtie$ *zákazník, jméno-zákazníka* v relaci *vkladatel* je cizí klíč do relace *zákazník,* tedy výsledek obsahuje přesně $n_{vkladatel}=5000$ n-tic
- Pokud $R \cap S = \{A\}$ není klíč ani v R nebo v S.
  Pokud předpokládáme, že každá n-tice v r vytváří n-tice v $r \bowtie s$, počet n-tic v $r \bowtie s$ je odhadován jako:

$$\frac{n_r * n_s}{V(A,s)}$$

  Pokud platí opačný příklad, odhad je následující:

$$\frac{n_r * n_s}{V(A,r)}$$

  Nižší z obou výrazů je pravděpodobně ten přesnější.
- Vypočítej odhady velikostí spojení *vkladatel* $\bowtie$ *zákazník* bez užití informace o cizích klíčích:
  - $V(jméno\text{-}zákazníka,vkladatel)=2500$ a $V(jméno\text{-}zákazníka,zákazník)=10000$
  - Dva odhady jsou $5000*10000/2500=20000$ a $5000*10000/10000=5000.$
  - Zvolíme nižší odhad, který je v tomto případě stejný jako dřívější výsledek pomocí cizích klíčů.

Spojení - vnořené cykly

- Spočítej theta spojení $r \bowtie_\theta s$

```
for each n-tici t_r z r do
    for each n-tici t_s z s do
            testuj dvojici (t_r,t_s), jestli splňuje podmínku θ spojení
            pokud ano, přidej t_r t_s do výsledku
    enddo
```

enddo

- *r* je nazývána **vnější** relace a *s* je **vnitřní** relací spojení.
- Nevyžaduje žádné indexy a může být použit s jakoukoli spojovací podmínkou
- Velmi drahý, protože prochází každou dvojici n-tic z obou relací. Pokud se menší relace vejde do paměti, použij takovou relaci jako vnitřní.
- V nejhorším případě, pokud je dostatek paměti pouze na jeden blok každé relace, odhad ceny spojení je $n_r*b_s + b_r$ diskových přístupů.
- Když se menší relace vejde celá do paměti, použije se jako vnitřní relace a odhad ceny je $b_r + b_s$
- Předpokládejme nejhorší případ, náklady budou *5000*400 + 100=2 000 100* diskových přístupů s vnější relací *vkladatel* a *1000*100 + 400=1 000 400* diskových přístupů s vnější relací *zákazník.*
- Když se menší relace *(vkladatel)* vejde celá do paměti, odhad je *500* diskových přístupů.
- Vhodnější jsou blokované vnořené cykly.

Spojení – blokované vnořené cykly

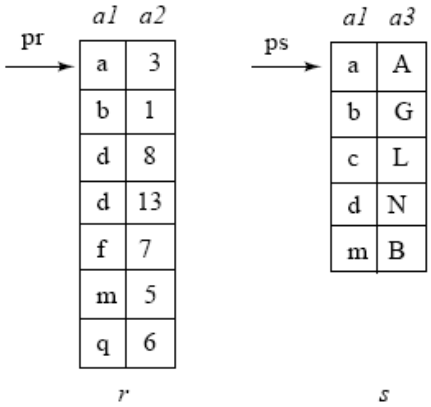
- Varianta vnořených cyklů, která spojuje každý blok vnitřní relace s každým blokem vnější relace.

for each blok B_r z *r* **do**
for each blok B_s z *s* **do**
for each n-tici t_r z B_r **do**
for each n-tici t_s z B_s **do**
testuj dvojici (t_r,t_s), jestli splňuje podmínku θ spojení
pokud ano, přidej t_rt_s do výsledku
enddo
enddo
enddo
enddo

- Nejhorší případ: každý blok vnitřní relace je čten pouze jednou pro každý blok vnější relace, místo jednoho čtení pro každý záznam vnější relace.
- Nejhorší případ odhadujeme: $b_r*b_s + b_r$ diskových přístupů, nejlepší je b_r+b_s, což v našem případě znamená *100*400+100=40 100* čtení bloku z disku.

Slučované spojení

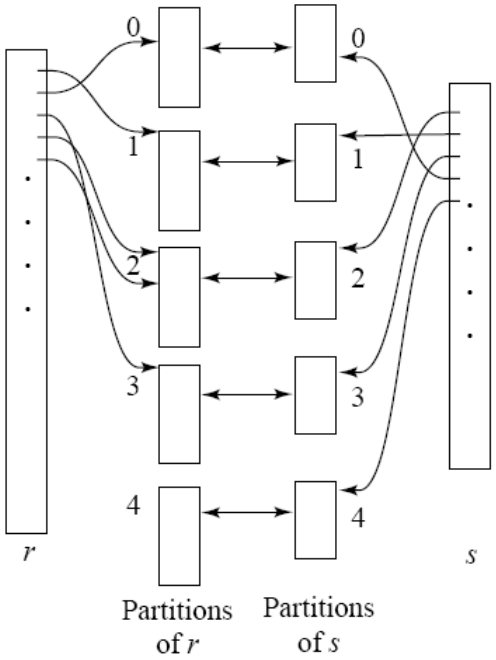
1. Nejprve uspořádej obě relace na jejich atributech pro spojení (pokud již nejsou podle spojovacího atributu uspořádány)
2. Krok spojení je podobný spojovací fázi sort-merge (seřaď a spoj) algoritmu. Hlavní rozdíl je v zacházení s duplikáty v spojovacích atributech – každý pár se stejnou hodnotou na spojovacích atributech musí být vypsán.



- Každá n-tice je čtená pouze jednou, a proto je každý blok také čtený jen jednou. Tedy, počet čtení bloků je b_r+b_s plus náklady na uspořádání relací.

Hešované spojení

- Hešovací funkce *h* je použita pro rozdělení n-tic obou spojovaných relací do množin, které mají stejnou hodnotu hešovací funkce na spojovacích atributech.
  - *h* mapuje spojovací atributy A_s na hodnoty *{0, 1, … max}*, kde A_s označuje společné atributy relací *r* a *s* použité přirozeným spojením.
  - $H_{r0}, H_{r1}, \ldots H_{rmax}$ označují oblasti relace *r* (na počátku prázdné). Každá n-tice $t_r \in r$ je vložena do oblasti H_{ri}, pokud $i = h(t_r[A_s])$.
  - $H_{s0}, H_{s1}, \ldots H_{smax}$ označují oblasti relace *s* (na počátku prázdné). Každá n-tice $t_s \in s$ je vložena do oblasti H_{si}, pokud $i = h(t_s[A_s])$.
- n-tice v H_{ri} jsou porovnávány pouze s n-ticemi v H_{si} – nemusí být porovnávány s jinými oblastmi, protože:
  - n-tice z *r* a n-tice z *s*, které splňují spojovací podmínku, mají stejnou hodnotu na spojovacích atributech.
  - Když je na takovou hodnotu aplikována hešovací funkce, dostaneme hodnotu *i* a n-tice z *r* je uložena do H_{ri} a n-tice z *s* je uložena do H_{si}.



Spojení pomocí hešování je vypočítáno následovně:

1. Pomocí hešovací funkce *h* rozděl relaci *s* na oblasti, jeden blok paměti je použit jako vyrovnávací paměť pro jednu oblast.
2. Podobně rozděl *r.*
3. Pro každé *i*:
   a. Nahraj H_{si} do paměti
   b. Po jedné čti n-tice v H_{ri} z disku a pro každou n-tici t_r najdi odpovídající n-tici t_s v H_{si}. Na výstup dej spojení těchto dvou n-tic.

- Například spojení relací *zákazník* a *vkladatel* s $b_{zákazník}=400$ a $b_{vkladatel}=100$ pomocí hešování vyžaduje *1500* čtení bloků z disku.

Pravidla ekvivalence výrazů

1. Konjunktivní operace výběru (AND) může být převedena na posloupnost jednotlivých operací výběru:
$$\sigma_{\theta 1 \wedge \theta 2}(E) = \sigma_{\theta 1}(\sigma_{\theta 2}(E))$$
2. Operace výběru jsou komutativní:
$$\sigma_{\theta 1}(\sigma_{\theta 2}(E)) = \sigma_{\theta 2}(\sigma_{\theta 1}(E))$$
3. Pouze poslední projekce z posloupnosti operací projekce je nutná, ostatní mohou být vynechána:
$$\Pi_{L1}(\Pi_{L2}(\ldots(\Pi_{Ln}()))) = \Pi_{L1}(E)$$
4. Operace výběru mohou být kombinovány s operací spojení s podmínkou θ (kartézský součin s podmínkou):
   a. $\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$
   b. $\sigma_{\theta 1}(E_1 \bowtie_{\theta 2} E_2) = E_1 \bowtie_{\theta 1 \wedge \theta 2} E_2$
5. Spojení s podmínkou θ je komutativní:
$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$
6. Operace přirozeného spojení jsou asociativní:
$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$
7. a další…

- Příklad
  - Dotaz: najdi všechny účty se zůstatkem větším než 1200 vedené na pobočce v Brně.
$$\sigma_{zůstatek>1200}(účet \bowtie_{pobočka\text{-}jméno=\text{“}Brno\text{“}} pobočka)$$
  - Transformace pomocí pravidla 4b
$$účet \bowtie_{pobočka\text{-}jméno=\text{“}Brno\text{“} \wedge zůstatek>1200} pobočka$$

Kroky v optimalizaci pomocí heuristiky

1. Rozlož konjunktivní operace výběru do posloupnosti jednoduchých výběrů
2. Přesuň výběr co nejblíže k relacím, pro urychlení vyhodnocení dotazu
3. Nejdříve vyhodnoť ty výběry a spojení, které vrací nejmenší výsledek
4. Nahraď kartézské součiny, které jsou následované operací výběru, operací spojení
5. Rozlož seznam atributů projekce na jednotlivé atributy a přesuň je co nejblíže k relacím
6. Najdi místa výrazu, která lze provádět souběžně