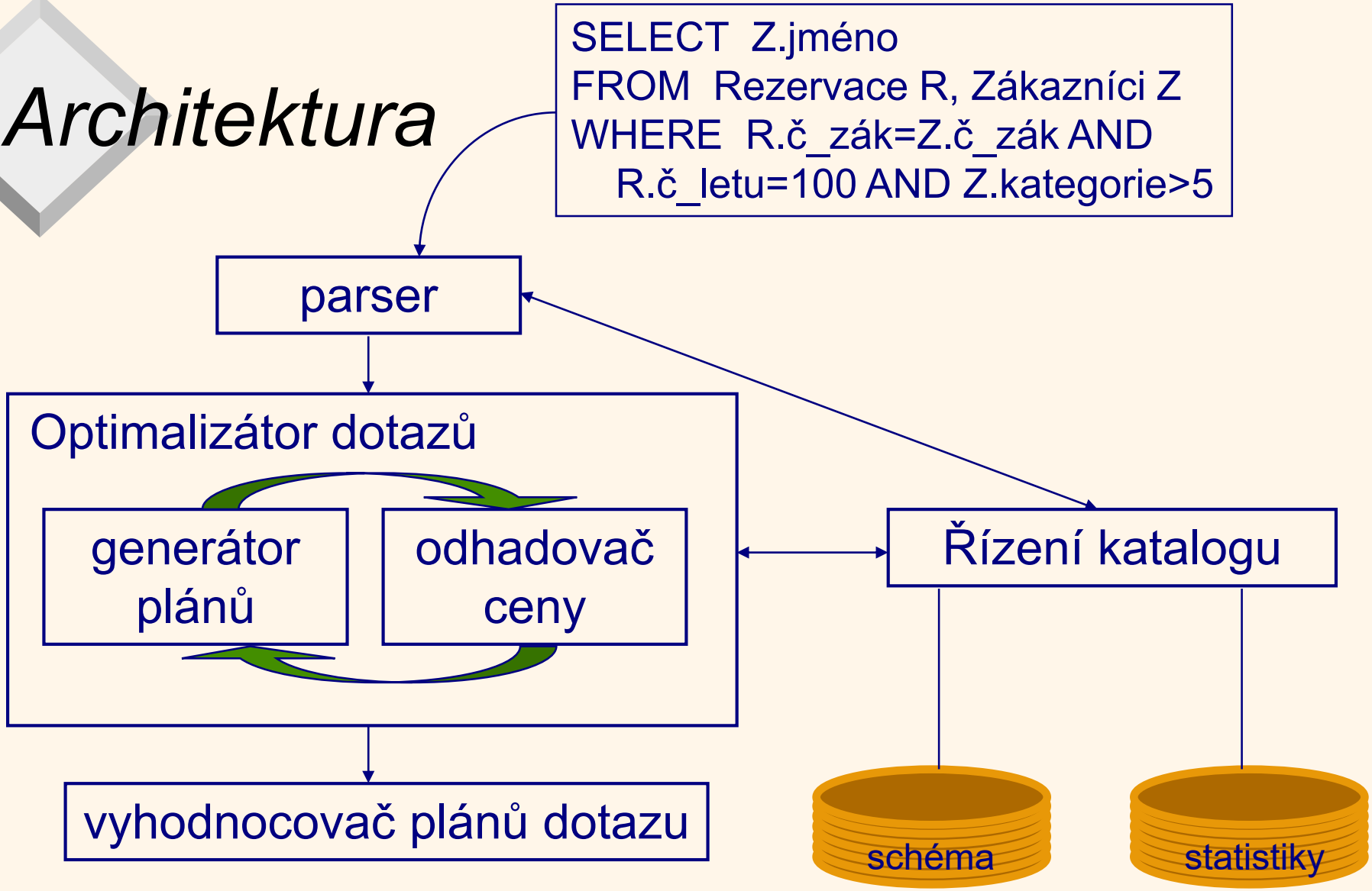
Optimalizace dotazů
slajdy k přednášce NDBI001

Jaroslav Pokorný
MFF UK, Praha
pokorny@ksi.mff.cuni.cz

Kontext v SŘBD

- Jde o klíčový modul SŘBD
- cíl: učinit optimalizaci nezávislou na strategii zápisu dotazu
  - protipříklady: navigační jazyky, interprety SQL
- paralela s vyhodnocováním aritmetických výrazů
  - zde: časová složitost operací A_R pomocí I/O operací
  - rozhodující: velikost relací, velikost aktivních domén, indexy, hašování, bitmapy atd.

Architektura



Optimalizátor

- Fáze zpracování dotazu
  - převod do vnitřní formy
    - SQL $\rightarrow$ A_R
    - lineární výraz $\rightarrow$ strom

    Pz.: kalkul $\leftarrow$ $\rightarrow$ A_R v polynomiálním čase v závislosti na délce výrazu
  - konverze do kanonického tvaru
  - optimalizace
  - plán vyhodnocení
  - generování kódu

Přehled problému

- *Plán vyhodnocení*: strom dotazu + algoritmus pro každou operaci.
- Dvě hlavní myšlenky:
  - jaké plány jsou uvažovány pro daný dotaz?
  - jak se odhaduje cena plánu?
- Z uvažovaných plánů se vybere ten s nejmenší cenou.

Př.: System R

- použití statistických dat pro odhad ceny,
- použití ekvivalentních algebraických výrazů,
- omezení na plány *doleva-do-hloubky*.

Schéma příkladu

Zákazníci (*č_zák*: int, *jméno*: string, *kategorie*: int, *věk*: real)
Rezervace(*č_zák*: int, *č_letu*: int, *datum*: date, *pozn*: string)

Sémantika: Zákazníci si rezervují lety do daného data.

Parametry: B = 4 KByte

- Rezervace:

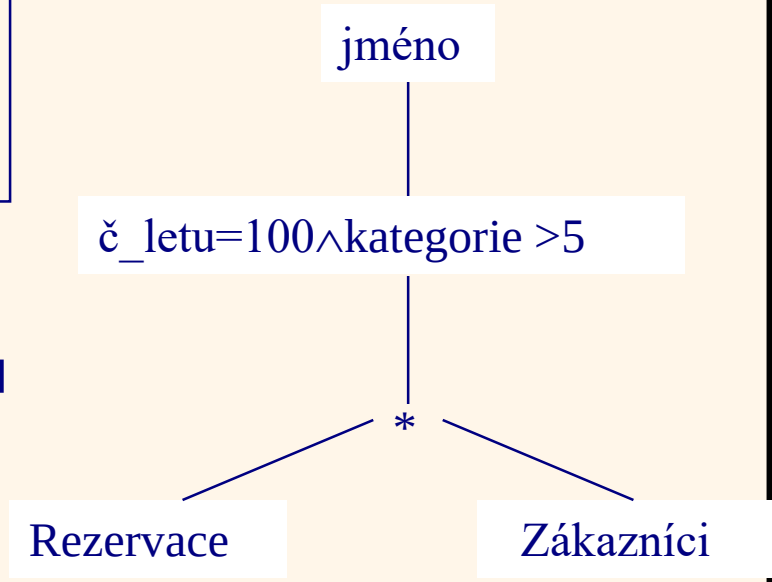
  R = 40 Byte, b = 100, p_R = 1000 stránek.

- Zákazníci:

  R = 50 Bytes, b = 80, p_Z = 500 stránek.

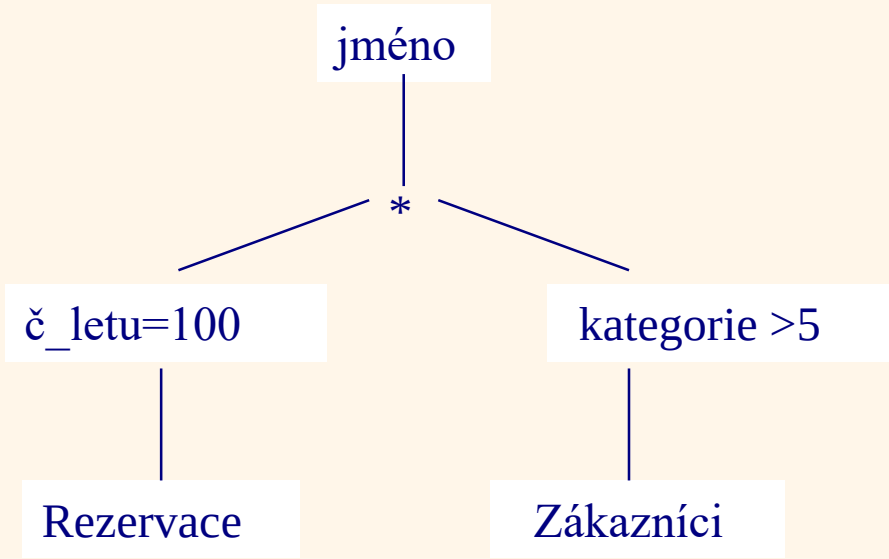
Alternativa 1

```
SELECT  Z.jméno
FROM  Rezervace R, Zákazníci Z
WHERE  R.č_zák=Z.č_zák AND
   R.č_letu=100 AND Z.kategorie>5
```



- Plán: spojení hnízděnými cykly, selekce+projekce při generování výsledku
- Cena: 500+500*1000 I/Os
- zřejmě nejhorší plán!
- Využít možnosti: selekce by měly být vyhodnoceny dříve, dostupné indexy, atd.
- Cíl optimalizace: Nalézt nejefektivnější plány, které vedou ke stejnému výsledku (odpovědi)

Alternativa 2 (bez indexů)



- Hlavní rozdíl: selekce nejdříve.
- Předpoklad: M=5 (5 bufferů). Výpočet ceny plánu:
  - Prohlídka Rezervace (1000) + write do T1 (10 stránek, máme-li 100 letů a rovnoměrné rozložení).
  - Prohlídka Zákazníci (500) + write do T2 (250 stránek, máme-li 10 kategorií rovnoměrné rozložení).
  - Sort(T1) (2*2*10), Sort(T2) (2*4*250), Merge(T1,T2) (10+250)
  - Součet: 1000+10+500+250+40+2000+260= 4060 I/O operací.

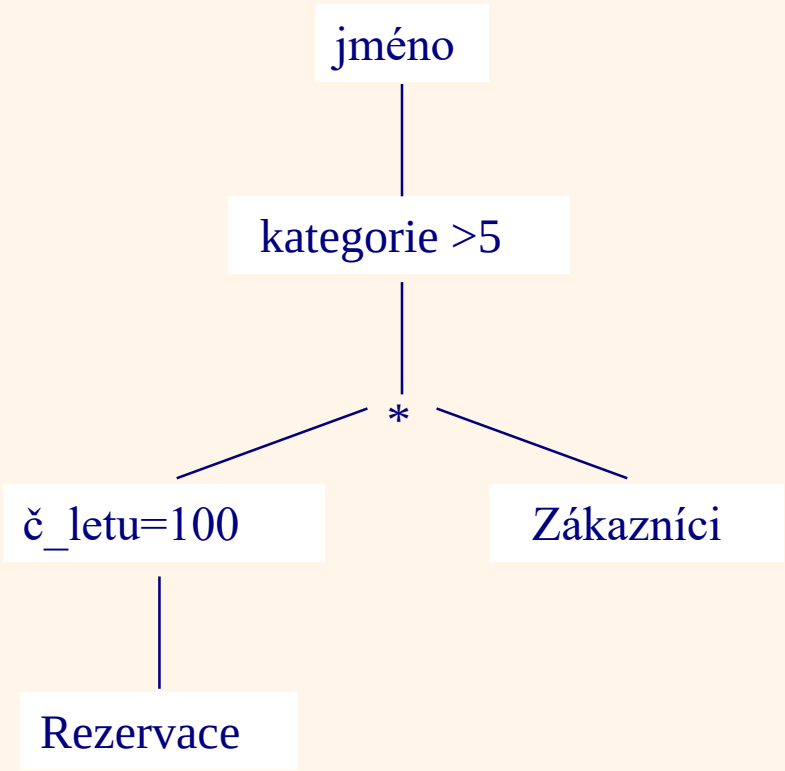
Pz.: třídění - n-cestným algoritmem třídění (T1 na 2 průchody, T2 na 4 průchody)

$2p_T$*#průchodů

Zlepšení: projekce před tříděním - T1[č_zák], T2[č_zák,jméno]:

- T1 (1 stránka), T2 (166 stránek),
- Součet: 1000+1+500+166 + 2*1*1 + 2*4*166 + 1 + 167 = 3027 I/O operací.

Alternativa 3 (s indexy)



- s klastrovaným indexem *č_letu* v Rezervace, obdržíme 100,000/100 = 1000 n-tic na 1000/100 = 10 stránkách.
- Spojovací atribut je klíčem v Zákazníci
  - nejvýše jedna n-tice, neklastrovaný index na *č_zák* OK.
- Rozhodnutí nepropagovat *kategorie >5* před spojením je založeno na dostupnosti indexu *č_zák* v tabulce Zákazníci.
- Cena: čtení stránek z Rezervace (10); pro každou rezervační n-tici se čte 1 stránka ze Zákazníci (1000 $\times$); součet: 1010 I/O operací

Algebraická optimalizace

Dovolují použít různé strategie pro spojení a propagovat selekce a projekce před operaci spojení.

- Komutativita spojení a kartézského součinu

$$E_1 [\theta] E_2 \approx E_2 [\theta] E_1$$

$$E_1 * E_2 \approx E_2 * E_1$$

$$E_1 \times E_2 \approx E_2 \times E_1$$

- Asociativita spojení a kartézského součinu

$$(E_1 [\theta_1] E_2) [\theta_2 \wedge \theta_3] E_3 \approx E_1 [\theta_1 \wedge \theta_3] (E_2 [\theta_2] E_3),$$

kde θ_2 zahrnuje atributy pouze z E_2 a E_3

$$(E_1 * E_2) * E_3 \approx E1 * (E_2 * E_3)$$

$$(E_1 \times E_2) \times E_3 \approx E_1 \times (E_2 \times E_3)$$

Algebraická optimalizace

- Komutativita selekce a projekce

Jsou-li všechny atributy z ϕ v $\{A_1,...,A_k\}$, pak

$$E_1[A_1...A_k](\phi) \approx E_1(\phi)[A_1...A_k]$$

Nejsou-li $B_1,...,B_s$ ve ϕ, pak

$$E_1(\phi)[A_1...A_k] \approx E_1[A_1...A_kB_1...B_s](\phi)[A_1...A_k]$$

Pz.:Propagaci selekce k (základním) relacím lze použít i u operací $\cup$, -, $\times$.

- Komutativita selekce a kartézského součinu

Jestliže všechny atributy ve ϕ jsou současně v E_1, pak

$$(E_1 \times E_2)(\phi) \approx E_1(\phi) \times E_2$$

Algebraická optimalizace

- Komutativita selekce a sjednocení

$$(E_1 \cup E_2)(\phi) \approx E_1(\phi) \cup E_2(\phi)$$

- Komutativita selekce a rozdílu

$$(E_1 - E_2)(\phi) \approx E_1(\phi) - E_2(\phi)$$

Pz.: Podobně lze použít i projekci.

- Komutativita projekce a kartézského součinu

$$(E_1 \times E_2)[A_1 \ldots A_n] \approx E_1[B_1 \ldots B_k] \times E_2[C_1 \ldots C_m]$$

kde $\cup_i B_i \cup \cup_i C_i = \cup_i A_i$, B_i se týkají E_1 a C_j se týkají E_2.

- Komutativita projekce a sjednocení

$$(E_1 \cup E_2)[A_1 \ldots A_n] \approx E_1[A_1 \ldots A_n] \cup E_2[A_1 \ldots A_n]$$

Heuristiky pro optimalizaci dotazů

1. Selekce co nejdříve. Použij kaskád selekcí, komutativnost selekcí s projekcemi a $\times$, distributivnost selekce nad $\cup$, $\cap$, - tak, aby se selekce dostaly co nejvíce k listům.
2. Projekce co nejdříve. Použij kaskád projekcí, distributivnost projekce nad $\times$, $\cup$, $\cap$, - a komutativnost selekce a projekce tak, aby se projekce dostaly co nejvíce k listům. Odstraň zbytečné projekce.
3. Je-li to možné, transformuj $\times$ na *. Selekce na 1 argument v $\times$ aplikuj dříve.
4. Posloupnost selekcí a/nebo projekcí nahraď jednou selekcí, jednou projekcí. Využij možnosti více operací najednou!

   (pipeline: např. následuje-li *, generuj n-tice spojení)

Heuristiky pro optimalizaci dotazů

5. Použij asociativity *, $\times$, $\cup$, $\cap$ k přeskupení relací ve stromu dotazu tak, aby selekce produkující menší relace byly volány dříve.
6. Ukládej výsledky společných poddotazů (nejsou-li příliš velké).

   Pz.: vhodné u dotazů na pohledech

Algebraická optimalizace - příklad

D: Nalezni tituly knih, u nichž existují exempláře, které mají být vráceny do 30.9.2015.

D_{RA}:

(VÝPŮJČKA * ČTENÁŘ * EXEMPLÁŘ * KNIHA) [TITUL, AUTOR, ISBN, INV_Č, JMÉNO, ADRESA, Č_ČT, D_ZPĚT]

(D_ZPĚT < 30.9.2015) [TITUL]

Pz.: D mohl vzniknout jako dotaz na pohled VÝPŮJČÁK

```
SELECT TITUL
FROM VÝPŮJČÁK
WHERE D_ZPĚT < 30.9.2015
```

Algebraická optimalizace - příklad

Transformace:

(1) 4 spojení po dvou nahradíme ×

((VÝPŮJČKA × ČTENÁŘ)(V.Č_ČT = Č.Č_ČT) [INV_Č, Č_ČT, D_ZPĚT, JMÉNO, ADRESA]

* ((EXEMPLÁŘ × KNIHA)(E.ISBN = K.ISBN) [TITUL, AUTOR, ISBN, INV_Č, D_NÁKUP])) [TITUL, AUTOR, ISBN, INV_Č, JMÉNO, ADRESA, Č_ČT, D_ZPĚT]
(D_ZPĚT < 30.9.2015) [TITUL]

(2) odstraníme * a z [ ] vynecháme D_NÁKUP

(A×B)(INV_Č = INV_Č) [TITUL, AUTOR, ISBN, INV_Č, JMÉNO, ADRESA, Č_ČT, D_ZPĚT]
(D_ZPĚT < 30.9.2015) [TITUL]

Algebraická optimalizace - příklad

(3) protože D_ZPĚT je v [] a podmínky selekcí komutují ⇒

(A×B)(D_ZPĚT < 30.9.2015)(INV_Č = INV_Č)[TITUL]

Pz.: odstranily se zbytečné projekce

(4) protože D_ZPĚT je jen v A v relaci VÝPŮJČKA ⇒

((VÝPŮJČKA(D_ZPĚT < 30.9.2015) × ČTENÁŘ)(V. Č_ČT = Č. Č_ČT)[INV_Č, Č_ČT, D_ZPĚT, JMÉNO, ADRESA] × B)
(INV_Č = INV_Č)[TITUL]

(5) redukce projekcí v () na [INV_Č] a na [INV_Č,TITUL] ⇒

(VÝPŮJČKA(D_ZPĚT < 30.9.2015)[INV_Č] × (EXEMPLÁŘ × KNIHA)(E.ISBN = K.ISBN)[INV_Č, TITUL])
(INV_Č = INV_Č)[TITUL] ⇒ relace ČTENÁŘ zmizí

Algebraická optimalizace - příklad

(6) výsledek v operacích selekce, projekce a * ⇒

(VÝPŮJČKA(D_ZPĚT < 30.9.2015)[INV_Č] * (EXEMPLÁŘ * KNIHA) [INV_Č, TITUL])[TITUL]

Dotaz patří do třídy SPJ-dotazů.

Lze je optimalizovat ve smyslu minimalizace počtu spojení.

(Jde o NP-úplný problém.)

Statisticky řízená optimalizace

- Odhad ceny pro každý plán: pro každou operaci se
  - provádí odhad *ceny* a *velikosti výsledku*
- Je třeba informace o velikosti R* a indexů.
- *Katalogy dat* typicky obsahují popis relace R a indexů:
  - n_R (# n-tic) a p_R (# stránek)
  - $V(A,R) = |R[A]|$ (tj. $|adom_A|$) prostřednictvím indexu
  - $p_{R.A}$ (# listových stránek B^+-stromu indexu pro R.A).
  - l(A,R) hloubku B^+-stromu pro index R.A, min/max hodnoty pro každý B^+-strom indexu, $2min_A$, $2max_A$ (druhá minimální, resp. maximální v $adom_A$)
- podrobnější informace (např. histogramy pro $adom_A$)

Odhad velikosti výsledku a redukční faktory

SELECT seznam atributů
FROM seznam relací
WHERE $atom_1$ AND ... AND $atom_k$

- Maximální # n-tic ve výsledku je dán součinem kardinalit relací za FROM.
- *Redukční faktor (RF)* asociovaný s každým *atomem* odráží vliv atomu na redukci velikosti výsledku. *Kardinalita výsledku* = Max # n-tic * součin všech RF.
- Implicitní předpoklad, že termy jsou nezávislé!

Odhad velikosti výsledku a redukční faktory

SELECT seznam atributů
FROM seznam relací
WHERE $atom_1$ AND ... AND $atom_k$

- atom A=k
  - RF = 1/V(A.R) existuje-li index
  - RF = 1/10 neexistuje-li index
- atom A=B
  - RF = 1/MAX(V(A,R), V(B,S)) existují-li index na A i B
  - RF = 1/V(A,R) existují-li index na A
  - RF = 1/10 neexistuje-li žádný index
- atom A>k
  - RF = (2max-k)/(2max-2min) existuje-li index
  - RF < ½ není-li A číselného typu nebo neexistuje index

Optimalizace pomocí hrubého odhadu redukčních faktorů

Strategie: odhady RF operátorů

Př.: hrubé odhady konstantami

$RF_{=} = 20\%$, $RF_{>} = 40\%$

$\Rightarrow$ LET.cena > 26.000 (1)

LET.cena > 7.000 (2)

mají stejné RF, přestože zřejmě

$RF1_{>skut} < RF2_{>skut}$

Příklad: Informix Online

Předpoklady: i-atribut je atribut s indexem, *k* je konstanta, *m* je odhad velikosti poddotazu, 2max, 2min je druhá nejmenší, resp. největší hodnota z adom(A)

selekční podmínka	RF
R.i-atribut = k	
R.i-atribut IS NULL	1/V(R.i-atribut, R)
R.i-atribut = S.i-atribut	1/max(V(R.i-atribut,R), V(S.i-atribut,S))
i-atribut > k	(2max - k)/(2max -2min)
i-atribut < k	(k -2min)/(2max -2min)
atribut = výraz	
atribut = NULL	1/10
atribut LIKE výraz	1/5

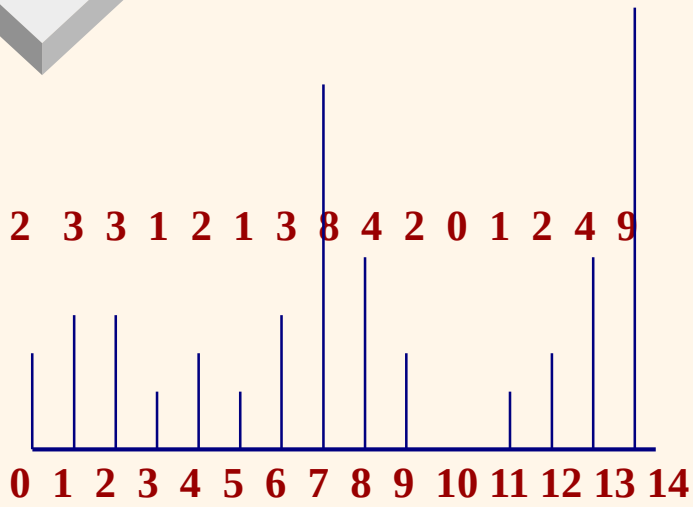
Příklad: Informix Online

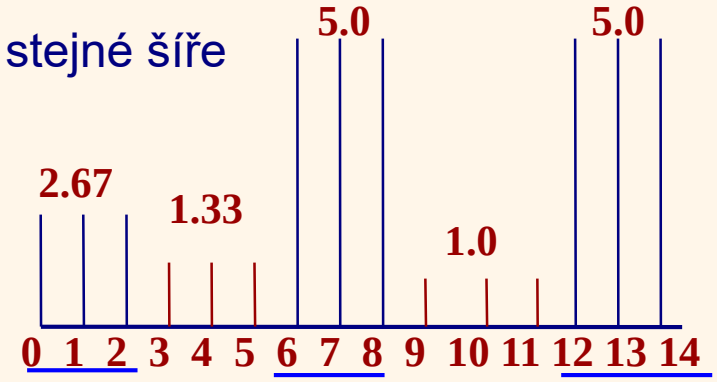
selekční podmínka	RF
atribut > výraz	
atribut < výraz	1/3
EXISTS poddotaz	1, je-li odhad, že TRUE 0, v opačném případě
NOT selekce	$1-RF_{selekce}$
$selekce_1$ AND $selekce_2$	$RF_{selekce1} * RF_{selekce2}$
$selekce_1$ OR $selekce_2$	$RF_{selekce1} + RF_{selekce2} - RF_{selekce1} * RF_{selekce2}$
atribut IN seznam-hodnot	$\Leftrightarrow$ atribut = k_1 OR … atribut = k_m
atribut θ ANY poddotaz	$\Leftrightarrow$ atribut θ k_1 OR … atribut θ k_m

Statisticky řízená optimalizace

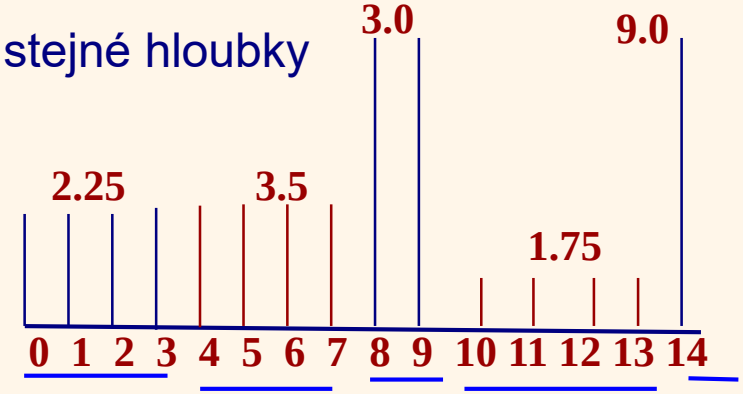
- *Histogramy*
  - v aplikacích není předpoklad rovnoměrného rozdělení reálný
  - histogramy aproximují reálné rozdělení dat je udržována SŘBD
  - jsou udržovány SŘBD
- Druhy histogramů
  - stejné šíře: rozdělují obor hodnot sloupce do intervalů s předpokladem, že rozdělení hodnot v intervalu je rovnoměrné
  - stejné hloubky: počet n-tic v intervalu je přibližně stejně velký

Statisticky řízená optimalizace





interval	1	2	3	4	5
počet	8	4	15	3	15



interval	1	2	3	4	5
počet	9	14	6	7	9

Výčet alternativních cest

- Dva hlavní případy:
  - plány pro jednotlivou relaci
  - plány pro více relací
- Dotazy nad jednotlivou relací se skládají s operací selekce, projekce (a agregačních operací):
  - každá dostupná přístupová cesta (prohlížení souboru/ index) je uvažována a ta s nejmenší odhadnutou cenou je zvolena.
  - Dvě různé operace mohou být provedeny pohromadě (např. je-li pro selekci zvolen index, projekce se udělá pro každou vybranou n-tici a výsledné n-tice jsou posouvány (pipeline) do výpočtu agregace).

Příklad: Systém R

Předpoklady: jednoduchý dotaz q nad relací R, některé atributy s indexem, V(A, R)

- indexy:
  - klastrované (R(A=c) je $\approx$ v minimálním množství stránek)
  - neklastrované (R(A=c) je $\approx$ v n_R/V(A,R) stránek)

Metoda: vybere se nejlevnější strategie z (1)-(8) a na výsledek s použijí zbývající podmínky z *q*

(1) A = c, kde pro A existuje klastrovaný index

Cena: p_R/V(A,R)

(2) A θ c, kde $\theta \in \{\geq, \leq, <, >\}$ a pro A existuje klastrovaný index

Cena: $p_R/2$

Pz.: pro $\neq$ je třeba číst $\approx$ celou R $\Rightarrow$ (5)

Příklad: Systém R

(3) A = c, kde pro A existuje neklastrovaný index

Cena: $n_R/V(A,R)$

(4) Je-li R sekvenční soubor, pak se čte celá.

Cena: p_R

(5) Je-li R „smíchaná“ s jinými relacemi a existuje klastrovaný index na libovolný atribut (skupinu atributů), pak se čte celá R „přes“ něj.

Cena: p_R

(6) A θ c, kde $\theta \in \{\geq, \leq, <, >\}$ a pro A existuje neklastrovaný index

Cena: $n_R/2$

Příklad: Systém R

(7) Existuje-li libovolný neklastrovaný index, čte se celá R „přes“ něj.

Cena: n_R

(8) Nelze použít (1)-(7). Čtou se všechny stránky eventuálně obsahující R.

Cena: $\geq n_R$

Pz: A = c AND B=d a existuje index pro A i B.

Pak lepší strategie by byla „přes oba indexy“ $\Rightarrow$ průnik dvou seznamů ukazatelů

Odhad ceny plánu pro jednotlivou relaci – přesněji s RF

- Index na primárním klíči A vyhovuje rovnosti:
  Cena: I(A,R)+1 pro B^+strom, okolo 1.2 hašovaný index.
- Klastrovaný index I vyhovuje 1 či více porovnání:
  $(p_{R.A} + p_R)$ * součin RF vyhovujících selekcí.
- Neklastrovaný index I vyhovuje 1 či více selekcí:
  $(p_{R.A} + n_R)$ * součin RF vyhovujících selekcí.

- projekce se prováděly bez eliminace duplicit!

Příklad

```
SELECT  Z.č_zák
FROM Zákazníci Z
WHERE Z.kategorie=8
```

- existuje index na *kategorie*:

Má být vybráno $(1/V(A,R))*n_R = (1/10) * 40000$ n-tic.

  - klastrovaný index: $(1/V(A,R)) * (p_{Z.kategorie} + p_R)=$ $(1/10) * (50+500)$ stránek je vybráno.
  - neklastrovaný index: $(1/V(A,R)) * (p_{Z.kategorie} + n_R) = (1/10) * (50+40000)$ stránek je vybráno.

- existuje index na *č_zák*:
  - musely by se vybrat všechny n-tice/stránky. Index není použitelný. Prohlíží se celý soubor Z (500).

Dotazy nad více relacemi

- protože počet spojení se zvyšuje, rychle roste počet alternativních plánů; je třeba omezit vyhledávací prostor.
  - Pro n relací $R_1, \ldots, R_n$ je počet plánů $(2(n-1))!/(n-1)!$, např. pro $n=7$ je to 665280
- Řešení: pomocí dynamického programování; S obsahuje n relací

procedure findbestplan(*S*)
  if (*bestplan*[*S*].*cost* $\neq \infty$)
    return *bestplan*[*S*] //else *bestplan*[*S*] ještě nebyl spočítán, spočítej ho nyní
  if (*S* obsahuje pouze 1 relaci)
    nastav *bestplan*[*S*].*plán* a *bestplan*[*S*].*cena* podle nejlepšího přístupu k *S*
  else for each $S1 \subseteq S$ taková, že $S1 \neq \emptyset$ a $S1 \neq S$
    P1= findbestplan(*S*1)
    P2= findbestplan(*S* - *S*1)
    A = nejlepší algoritmus pro spojení výsledků *P*1 a *P*2
    cena = *P*1.*cena* + *P*2.*cena* + cena *A*
    if *cena* < *bestplan*[*S*].*cena*
      bestplan[*S*].*cena* = cena
      bestplan[*S*].*plán* = "vyvolej *P*1.*plán*; vyvolej *P*2.*plán*;
        spoj výsledky *P*1 a *P*2 algoritmem *A*"
  return *bestplan*[*S*]

Dotazy nad více relacemi

- zásadní rozhodnutí v System R: pro * uvažovány pouze *lineární stromy*, které jsou typu *doleva-do-hloubky*.

  Df.: lineární: každý nelistový uzel má alespoň jednoho následníka z **R**

  Df.: doleva-do-hloubky: každý pravý následník je z **R**

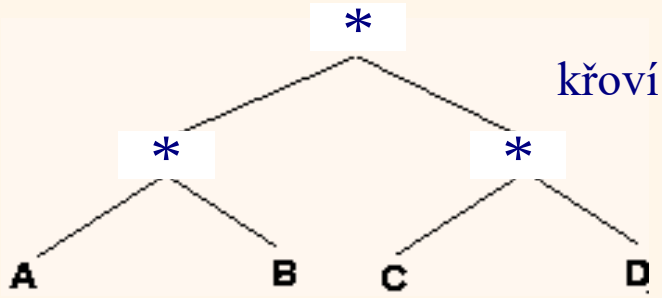
- spojení doleva-do-hloubky umožňují generovat *plně posouvatelné* plány (piplined).
  - mezivýsledky není nutné zapisovat do pomocných souborů

  Pz.: ne všechny plány doleva-do-hloubky jsou plně posouvatelné (závisí na algoritmu operace spojení, např. třídění-slévání)

- Není třeba generovat všechna uspořádání spojení. Pomocí dynamického programování, nejlevnější uspořádání pro každou podmnožinu $\{R_1, \ldots, R_n\}$ je generováno pouze jednou a uloženo.

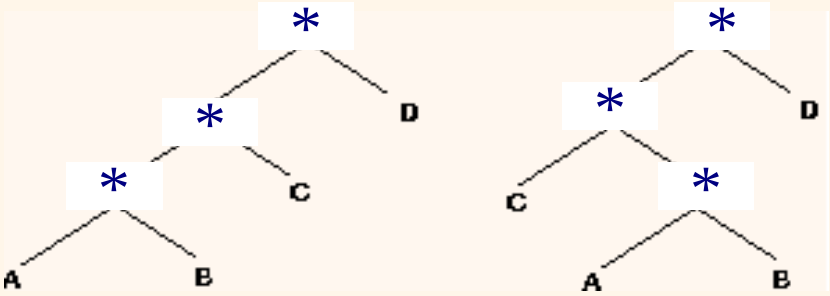
  Pz.: je $O(n*2^n)$ plánů doleva do hloubky

Dotazy nad více relacemi



křoví

nelineární stromy



doleva-do-hloubky

lineární stromy

Výčet plánů doleva-do-hloubky

Plány doleva-do hloubky se liší pouze v pořadí relací, přístupové metodě pro každou relaci a metodě spojení pro každou relaci.

- Modifikace algoritmu:
  - nahraď "**for each** $S1 \subseteq S$ taková, že $S1 \neq \emptyset$ a $S1 \neq S$"
  - výrazem „**for** r $\in$ S
    let S1 = S – r"
- Očíslování pomocí n průchodů (spojuje-li se n relací):
  - průchod 1: najdi nejlepší 1-relační plán pro každou relaci.
  - průchod 2: najdi nejlepší způsob ke spojení výsledku každého 1-relačního plánu (vnější) k další relaci (všechny 2-relační plány)
  - průchod n: najdi nejlepší způsob ke spojení výsledku (n-1)-relačního plánu (vnějšího) k n-té relaci. (všechny n-relační plány)

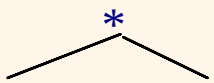
Hledání „nejlepšího“ plánu doleva-do-hloubky

```
procedure findbestplan(S)
    if (bestplan[S].cost ≠ ∞) return bestplan[S]
                    //else bestplan[S] ještě nebyl spočítán, spočítej ho nyní
    if (S obsahuje pouze 1 relaci)
        nastav bestplan[S].plán a bestplan[S].cena podle nejlepšího
        přístupu k S
    else for r ∈ S
        let S1 = S – r
        P1= findbestplan(S1)
        P2= findbestplan(S - S1)
        A = nejlepší algoritmus pro spojení výsledků P1 a P2
        cena = P1.cena + P2.cena + cena A
        if cena < bestplan[S].cena
                bestplan[S].cena = cena
                bestplan[S].plán = “vyvolej P1.plán; vyvolej P2.plán;
                                spoj výsledky P1 a P2 algoritmem A”
    return bestplan[S]
```

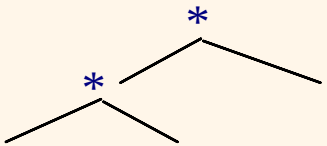
Výčet plánů doleva-do-hloubky

průchod 1 **A** **B** **C** **D**

průchod 2



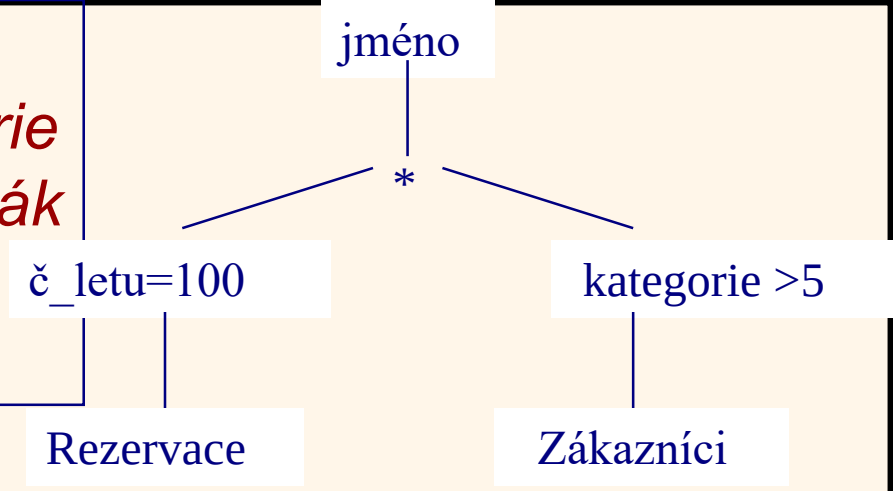
průchod n



- pro každou podmnožinu relací je ponechán pouze nejlevnější plán (pro každé *zajímavé uspořádání* n-tic - viz třídění, slévání, group by).

Příklad

Zákazníci:
B$^+$-strom na *kategorie*
hašování podle *č-zák*
Rezervace:
B$^+$-strom na *č_letu*



Průchod 1:

- *Zákazníci*: B$^+$-strom se shoduje na *kategorie>5*, a je asi nejlevnější. Výsledkem je ale množina n-tic, index je neklastrovaný, prohlížení souboru může být levnější.
  - podrží se plán s B$^+$-stromem (setřídění podle *kategorie*)
- *Rezervace*: B$^+$-strom se shoduje na *č_letu*=100; nejlevnější.

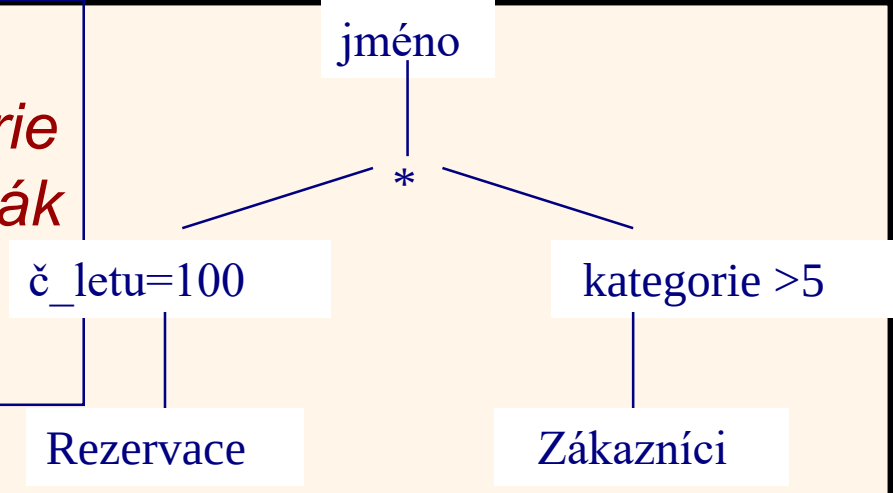
Průchod 2:

uvažujeme každý plán daný z průchodu 1 jako vnější a zkoumáme, jak ho připojit k další relaci

- *Rezervace* jako vnější: hašováním k n-ticím Zákazníci, které vyhovují *č-zák* = hodnota *č-zák* vnější n-tice.

Příklad

Zákazníci:
B⁺-strom na *kategorie*
hašování podle *č-zák*
Rezervace:
B⁺-strom na *č_letu*



Průchod 1:

- *Zákazníci*: B⁺-strom se shoduje na *kategorie>5*, a je asi nejlevnější. Výsledkem je ale množina n-tic, index je neklastrovaný, prohlížení souboru může být levnější.
  - podrží se plán s B⁺-stromem (setřídění podle *kategorie*)
- *Rezervace*: B⁺-strom se shoduje na *č_letu*=100; nejlevnější.

Průchod 2:

uvažujeme každý plán daný z průchodu 1 jako vnější a zkoumáme, jak ho připojit k další relaci

- *Rezervace* jako vnější: hašováním k n-ticím Zákazníci, které vyhovují *č-zák* = hodnota *č-zák* vnější n-tice.

Bloky dotazu: jednotky optimalizace

```
SELECT  Z.jméno
FROM  Zákazníci S
WHERE  Z.věk IN
     (SELECT  MAX (S2.věk)
       FROM  Zákazníci S2
       GROUP BY  S2.kategorie)
```

Vnější blok *Hnízděný blok*

- Dotaz v SQL je rozložen na kolekci *bloků dotazu*, které jsou optimalizovány vždy 1 blok v čase.
- Hnízděnému bloku odpovídá (zjednodušeně) volání procedury - pro každou n-tici z vnějšího bloku pro každý blok jsou uvažovány plány:
  - ∀ dostupné přístupové metody pro ∀ relaci za FROM.
  - ∀ stromy pro spojení doleva do hloubky (jak spojovat s relacemi za vnitřním FROM (uvažují se permutace a metody spojení)

Hnízděné dotazy

- Hnízděný blok je optimalizován nezávisle, s vnější n-ticí považovanou za selekční podmínku.
- Vnější blok je optimalizován s cenou, která bere do úvahy ‘volání’ výpočtu hnízděného bloku.
- Implicitní uspořádání těchto bloků znamená, že některé dobré strategie nejsou uvažovány. Nehnízděná verze dotazu je obvykle optimalizována lépe.

```
SELECT  Z.jméno
FROM  Zákazníci Z
WHERE EXISTS
   (SELECT  *
    FROM  Rezervace R
    WHERE  R.č_letu=103
    AND  R.č_zák=Z.č_zák)
```

Hnízděný blok k optimalizaci:

```
SELECT  *
FROM  Rezervuje R
WHERE R.č_letu=103 AND
Z.č_zák=vnější hodnota
```

Ekvivalentní ne-hnízděný dotaz:

```
SELECT  Z.jméno
FROM Zákazníci Z, Rezervace R
WHERE  Z. č_zák =R. č_zák
AND R.č_letu=103
```

Syntaxí řízená optimalizace

Př: SELECT * FROM EXEMPLÁŘ (1)
WHERE CENA >’40’ AND ZEMĚ_VYDÁNÍ = ’GB’

SELECT * FROM EXEMPLÁŘ (2)
WHERE ZEMĚ_VYDÁNÍ = ’GB’ AND CENA >’40’

v některých SŘBD závisí vyhodnocení na pořadí podmínek:

ta s nejnižším RF se vyhodnocuje jako první

$\Rightarrow$ varianta (2) je efektivnější než (1).

Syntaxí řízená optimalizace

Jak obejít optimalizátor?

Př:

```
SELECT * FROM EXEMPLÁŘ                                  (1)
WHERE (D_NÁKUP >'23.04.99' AND ZEMĚ_VYDÁNÍ = 'GB')
          OR  ISBN = '486';

(SELECT * FROM EXEMPLÁŘ                                 (2)
WHERE D_NÁKUP >'23.04.99' AND ZEMĚ_VYDÁNÍ = 'GB')
UNION
(SELECT * FROM EXEMPLÁŘ WHERE ISBN = '486');
```

Tendence optimalizátoru: (1) sekvenčně, (2) s indexy pro poddotazy

Shrnutí

- Optimalizace dotazů je důležitý úkol řešený SŘBD.
- Další přístupy:
  - založené na pravidlech
  - pravděpodobnostní algoritmy
  - parametrická optimalizace
- Je třeba rozumět optimalizaci, aby bylo možné porozumět vlivu návrhu databáze (relace, indexy) na zátěž (množina dotazů).

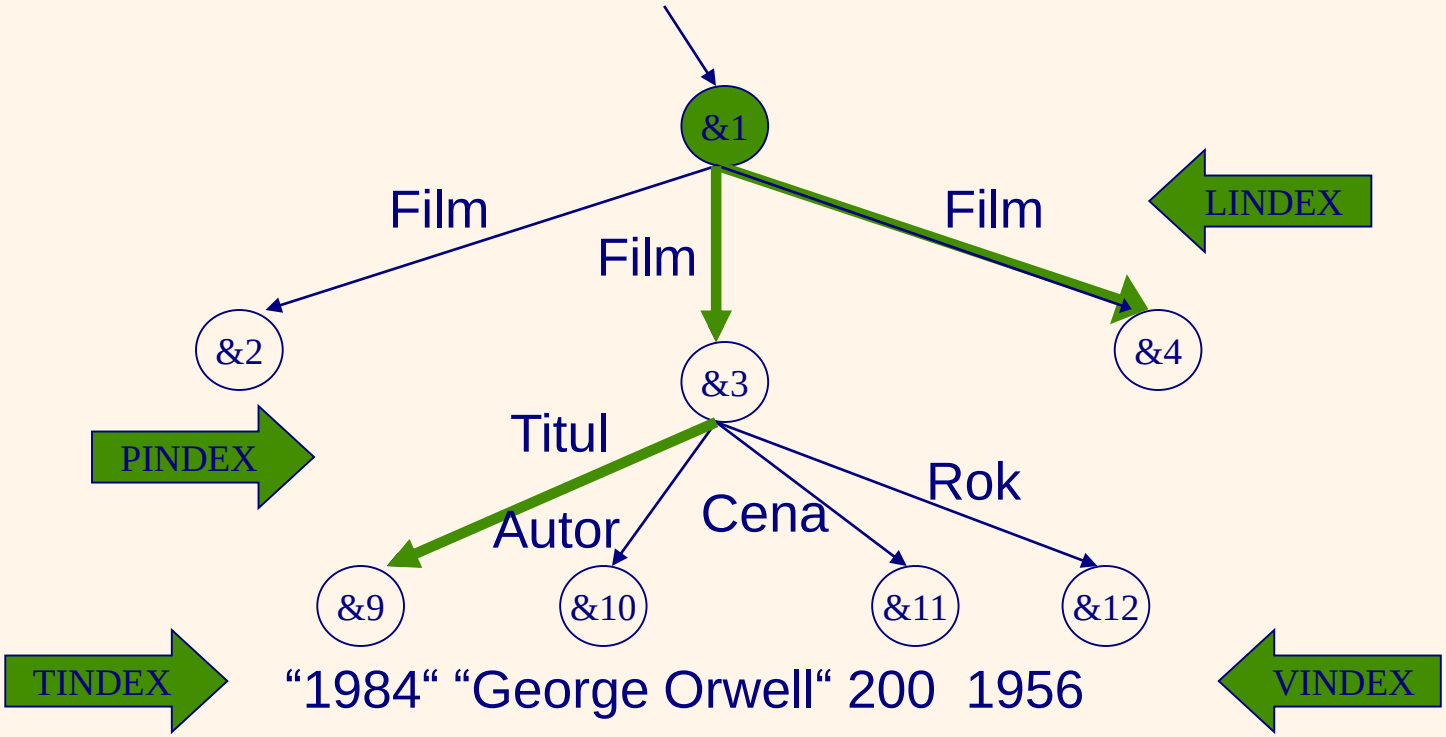
Metody indexace semistrukturovaných dat

- Indexace úplného textu

Nevýhoda: nelze dotazovat podle struktury

- Indexace relací klasicky (Lore)
- Indexování založené na pozicích
  - Použití absolutních resp. relativních adres pro reprezentaci pozic slov a značek v XML dokumentu
- Indexování založené na cestách
  - kódují se cesty podle typu průchodu stromem
  - kóduje všechny cesty vedoucí ke všem slovům
    - je možné se dotazovat na obsah i strukturu
  - …

Indexace v Lore



Value Index (VINDEX)

- **Vstup:** značka *T*, porovnávací operátor Θ, hodnota *v*
- **Výstup:** všechny atomické objekty mající vstupující hranu *T* a hodnotu *v'* vyhovující Θ a hodnotě *v*.
- **Př.**: (*Cena*, >, 150)

  Výsledek je {&11, &15}
- *Vindex* lze implementovat např. B+-stromů;

Link Index (LINDEX)

- **Vstup:** oid *o* a značka *T*
- **Výstup:** všechny rodičovské objekty mající hranu *T* vstupující do *o*.
  - Je-li *T* vynechána, vrací všechny rodiče a jejich značky.
- **Př.**: nalezení rodiče pomocí *Lindex* pro objekt &4 podle značky Film;

  vrací rodičovský objekt &1
- *Lindex* lze implementovat např. lineárním hašováním

Link Index (LINDEX)

Př.: db/A/B[C=5]

Využije Vindex i Lindex:

- C komponenta pomocí Vindex a (C, =, 5)
- Zkouší, zdali existuje cesta A/B z db do tohoto objektu pomocí dvou volání Lindex
- Zpětné vyhodnocení

Text Index (TINDEX)

- **Vstup:** *Tindex* provádí hledání podle klíčových slov na hodnotách typu text.
- **Výstup:** seznam souřadnic tvaru <*o*, *n*>
- Lze implementovat pomocí invertovaných seznamů, mapujících slovo *w* a značku *T* do seznamu atomických hodnot *v* se vstupující hranou *T*, přičemž *v* obsahuje *w*
- Značka může být vynechána
- **Př.**: nahlédnutí pomocí *Tindex* na všechny objekty s atomickou hodnotou obsahující slovo “Ford“ a mající vstupující hranu *Jméno*

  Result:{<&17, 2><&21, 2>}

Path Index (PINDEX)

- **Vstup:** objekt *o* a výraz *p* označující cestu
- **Výstup:** všechny objekty dosažitelné z *o* po cestě *p*
- omezení: obvykle pouze *jednoduché cesty*, tj. ty začínající v pojmenovaných objektech a neobsahující žádné regulární výrazy
- **Př.**: DB/Film/Titul

  Pindex k nalezení všech objektů dosažitelných pomocí DB/Film/Titul

  Result: {&5, &9, &14}

Vyhodnocování shora dolů přímé

Př.: DB.Film[Cena < 200]

- prohledají se všechny podelementy elementu Film v DB a pro každý nahlédnout na obsah podelementu Cena, jestli je jeho hodnota menší než 200
- To vede na prohledávání stromu do hloubky (depth-first) podle hran, které se vyskytují ve výrazech cest

Vyhodnocování zdola-nahoru s indexy

Př.: DB.Film[Cena < 200]

- Nejprve se naleznou všechny objekty vyhovující podmínce použitím vhodného Vindex indexu
- Pro každý z těchto objektů se prochází zpět ve stromě pomocí Lindex směrem k jejich rodičům

  Výhoda: vyhrne se cestám, které nesplňují podmínku.

Vyhodnocování hybridní s indexy

Př.: DB.Film[Cena < 200]

Vyhodnotí se něco (ne nutně všechno) co se týká podmínky pomocí shora-dolů

- Pak se naleznou přímo ty objekty, které vyhovují podmínce pomocí Videx a prochází se dál pomocí Lindex do stejného bodu jako při shora-dolů přístupu
- Výsledek dotazu se nalezne jako průnik množiny objektů a kombinace procházení cest.