Optimalizace SQL dotazů

Roman Dušek
http://dusek.vysokeskoly.cz

Motivace

- SQL je velmi flexibilní jazyk.
- Dvěma či více různými dotazy je možno obdržet stejná data.
- Rychlost různých dotazů ovšem nemusí být stejná i přesto, že vracejí stejná data.

Rozvrh přednášky

- Proč optimalizujeme
- Obecná pravidla pro psaní SQL dotazů
- Oracle: zpracování SQL dotazů

Proč optimalizujeme? 1/2

- Jedním z hlavních důvodů provádění optimalizace v databázových (DB) prostředcích je minimalizace nákladů.

Proč optimalizujeme? 2/2

- Jedná se především o minimalizaci nákladů na:
  - zdrojový čas,
  - kapacitu paměti (prostor),
  - programátorskou práci.

  (= snažíme dosáhnout maximálního výkonu se stávajícími prostředky)

Kdo se na ladění výkonu podílí

- Návrhář databáze (designer)
- Vývojář (developer)
- Správce databáze (DBA)
- Uživatel

Obecná pravidla pro psaní SQL dotazů

- Vyjmenovat sloupce
- Používat co nejméně klauzuli LIKE
- Používat co nejméně klauzule IN, NOT IN
- Používat klauzule typu LIMIT
- Na začátek dávat obecnější podmínky
- Výběr vhodného pořadí spojení
- Používat hinty
- Nastavit indexy

Vyjmenovat sloupce 1/2

- V *SELECT* dotazech nepoužívat v seznamu sloupců hvězdičku *(*)*
- Ve většině případů nepracujeme se všemi sloupci výsledku
- ***SELECT * FROM Lide***
- ***SELECT Jmeno, Prijmeni FROM Lide***

Vyjmenovat sloupce 2/2

- Používáte-li v *SELECT* dotazu všechny sloupce, používejte také výpis jednotlivých sloupců
- Databáze nemusí zjišťovat seznam sloupců tabulky

Používat co nejméně klauzuli LIKE

- Nedoporučuje se používat pro vyhledávání ve velkých textových polích (můžou obsahovat až několik GB textu)
- Zamyslet se, zda nejde vyhledávání provést jinou metodou

Používat co nejméně klauzuli IN, NOT IN

- *Vhodnější je použití příkazů WHERE a WHERE NOT EXISTS*

... WHERE Doprava IN ('Ford', 'Octavia', 'Seat', 'Peugeot');

... WHERE Typ_Dopravy = 'Automobil';

Používat klauzule typu LIMIT 1/2

- V případech, kdy vybíráme např. nejstaršího člověka, můžeme použít dotaz:

  SELECT Jmeno, Prijmeni FROM Lide ORDER BY Vek DESC

- Dotaz vybere všechny záznamy, které následně sestupně setřídí

Používat klauzule typu LIMIT 2/2

- Lepší řešení:

SELECT Jmeno, Prijmeni FROM Lide ORDER BY Vek DESC LIMIT 0,1

Na začátek dávat obecnější podmínky 1/3

- V klauzuli *WHERE* dávat na začátek podmínky, po kterých vypadne ze seznamu nejvíce záznamů

  :-/

Na začátek dávat obecnější podmínky 2/3

- Příklad: V tabulce *Lide* hledáme ženy starší 18 let

```
SELECT Jmeno, Prijmeni FROM Lide
WHERE Pohlavi = 'Z' AND Vek > 18
```

Na začátek dávat obecnější podmínky 3/3

- DS nejprve vyhledá záznamy, vyhovující první podmínce, z nich pak vybírá záznamy vyhovující druhé podmínce

- Snažíme se, aby systém vyřadil na začátku co nejvíce řádků; ty se pak již při další podmínce nezkoumají…

Výběr vhodného pořadí spojení

1) vyhnout se plnému prohledávání tabulky (pokud možno využít index)

2) efektivně vybírat takové indexy, které načtou z tabulky co nejméně záznamů

3) vybrat takové pořadí spojení ze všech možných pořadí, aby bylo spojeno co nejméně položek

Další rady

- Použití UNION ALL místo UNION
- Spojování tabulek s využitím indexů
- Vytváření indexů pro atributy podle nichž se třídí v klauzili ORDER BY
- Provádění analýzy na indexovaných sloupcích

Používat hintů 1/3

- **Hint** = podnět, kterým optimalizátoru určíme, jaký má použít plán vykonávání dotazu

- Hinty se aplikují na blok dotazu, ve kterém se vyskytují.

Používat hintů 2/3

- SELECT jmeno, prijmeni, plat FROM ucitel WHERE pohlavi='M';

- Optimalizátor by v takovémto případě zřejmě zvolil full table scan, protože pohlaví může obsahovat pouze dvě hodnoty, tedy vrácených řádků by měla být velká část ze všech možných.

Používat hintů 3/3

- Pokud však víme, že učitelů - mužů je hodně málo (například ukládáme pouze učitele z mateřských školek), pak si můžeme pomocí hintu vynutit rychlejší přístup - index scan.

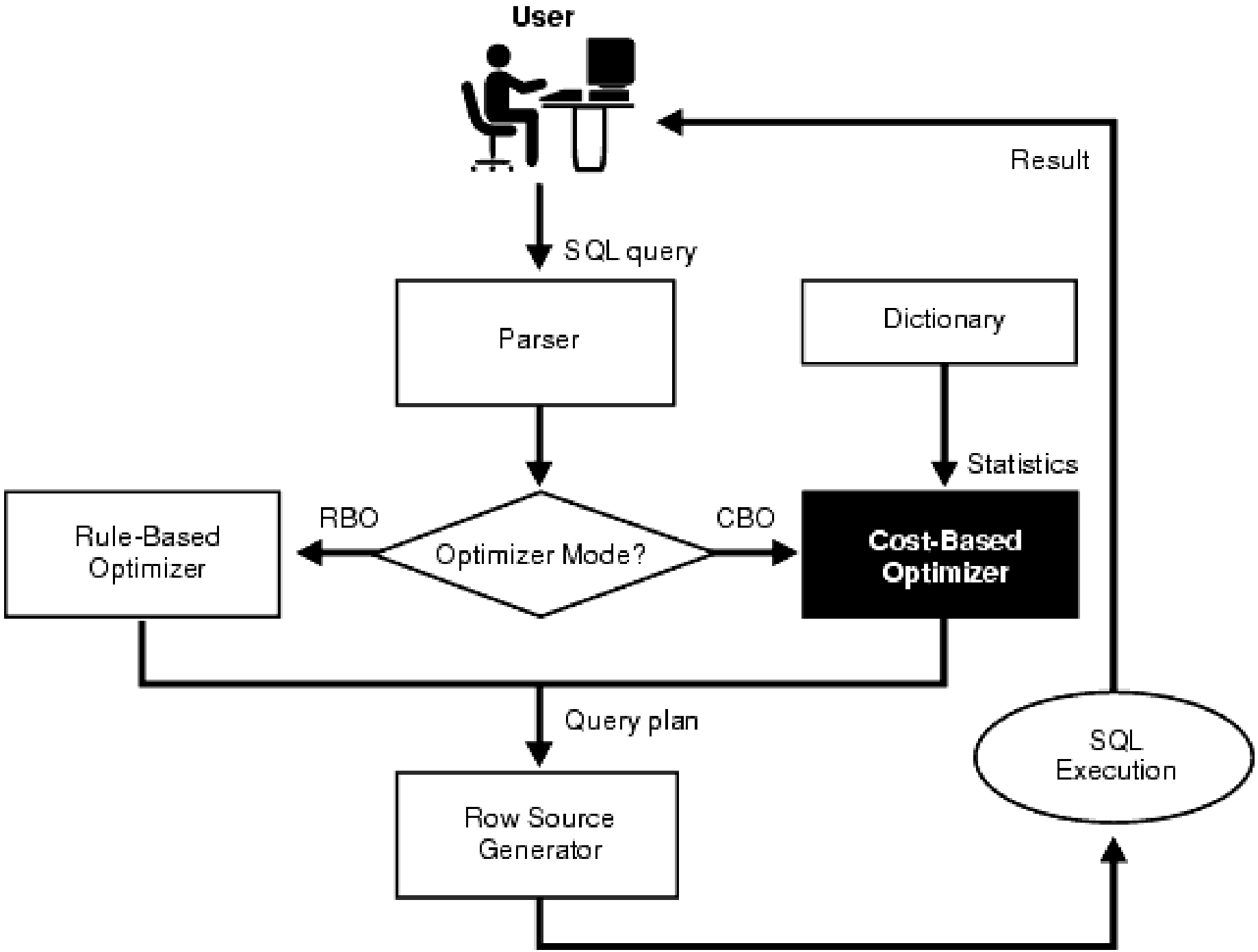
- SELECT /*+ INDEX(ucitel pohlavi_index) */ jmeno, prijmeni, plat FROM ucitele WHERE pohlavi='M';

Nastavit indexy

- Procházení tabulky pomocí indexu trvá mnohem kratší dobu než procházení tabulky bez jeho použití.
- Změna indexů se zdá být nejlepším řešením pro optimalizaci, jelikož má větší sílu než změna SQL dotazu či změna dat.
- Samotné vytvoření indexů však nelze brát v úvahu jako univerzální řešení problému.

Oracle: Zpracování SQL dotazů

- Zpracování SQL příkazů se sestává z následujících komponent:
  - Parser
  - Optimalizátor
  - Generátor řádkových zdrojů (row source generator)
  - Vlastní provádění (SQL execution)


User
Result
SQL query
Parser
Dictionary
Statistics
RBO
Optimizer Mode?
CBO
Rule-Based Optimizer
Cost-Based Optimizer
Query plan
Row Source Generator
SQL Execution

Optimalizátor 1/3

- Jádro celého zpracování
- Analyzuje sémantiku dotazu
- Hledá optimální způsob jeho provádění
- V Oracle rozeznáváme:
  - Rule-based optimizer (RBO)
  - Cost-based optimizer (CBO)
- Liší se v přístupu, jakým hledají optimální plán vykonávání

Optimalizátor 2/3

- **rule-based optimizer** vyhodnocuje jednotlivé přístupové cesty pomocí předem daného systému pravidel
- **cost-based optimizer** hledá plán s nejmenšími "náklady" (využívá statistiky)
- Oracle doporučuje používat pouze CBO

Optimalizátor 3/3

- Výstupem optimalizátoru je plán vykonávání (execution plan), který určuje:
  - přístupové cesty k jednotlivým tabulkám používaným dotazem,
  - pořadí jejich spojování (join order).

Statistiky 1/2

- Statistiky tvoří celá řada údajů o databázových objektech (tabulkách, indexech)
- Některé z těchto údajů jsou přístupné prostřednictvím tabulek a pohledů slovníku dat a může je tedy využívat i uživatel databáze.
- Aktualizují se výpočtem nebo odhadem.

Statistiky 2/2

- **údaje o tabulkách** (počet řádků, počet bloků, počet nevyužitých bloků, průměrnou délku záznamu)
- **údaje o sloupcích** (počet unikátních hodnot, počet prázdných (NULL) hodnot, histogram popisující distribuci dat)
- **údaje o indexech** (počet listových bloků, počet úrovní, clusterovací faktor)

Možnosti ladění 1/2

- **OPTIMIZER_MODE** – pro dosažení maximální propustnosti (s co nejmenším využitím zdrojů), nebo dosažení co nejlepší odezvy (co nejdříve vrátit první výsledky)

- **SORT_AREA_SIZE** - Určuje velikost paměti využívané při třídění a nepřímou úměrou ovlivňuje cenu spojení

Možnosti ladění 2/2

- **CURSOR_SHARING** – Tento parametr určuje, zda se bude dotaz vyhodnocovat přesně jak byl zadán nebo se literály nahradí vázanými proměnnými

- **HASH_AREA_SIZE** - Určuje velikost paměti využívané při hašovaném spojování a nepřímou úměrou ovlivňuje cenu hašovaného spojení

Minimalizace reparsingu dotazů

- Toho dosáhneme používáním jednotného zápisu dotazů a používáním vazebních proměnných místo konstant

–

„Dobrý návrh databáze a aplikace má daleko větší vliv na výkon, než sebelepší nastavení parametrů instance.“

Odkazy na zdroje dat

http://www.oracle.com
http://www.pcsvet.cz/art/article.php?id=197
http://www.dbs-intro.com/dbplus/ch01.html
http://www.sweb.cz/nidrle.vaclav/oracle2/optimalizace.html

–

Děkuji za pozornost.