Optimalizace dotazu, jednotlivé přístupy (např. Cost Based optimalizace (CBO)), podstata optimalizátoru, přínos optimalizace.

Thursday, May 30, 2013 8:17 AM

- SQL velmi flexibilní – dvěma i více různými dotazy je možné obdržet stejná data, ovšem rychlost dotazů nemusí být stejná
- důvodem optimalizace je minimalizace nákladů na:
  - strojový čas
  - kapacitu paměti či prostoru
  - programátorskou práci
  - přenesená data
- u malých databází optimalizace nepatrná, projeví se až u objemných, nebo u často navštěvovaných webů může při špatně formulovaných dotazech vzrůst trafic v obou směrech

zpracování příkazů se skládá z následujících komponent:

- parser
- optimalizátor
- generátor řádkových zdrojů
- vlastní provádění SQL

SŘBD Oracle již sám používá optimalizačních technik pro vyhodnocení jakéhokoli dotazu. Těmito technikami jsou:

- Rule based optimaliaztion (RBO)
- Cost based optimalization (CBO) – od verze Oracle 9i je preferovaný

Jak zjistit způsob provedení příkazu? Abychom mohli zjistit, jak ve skutečnosti daná optimalizace vyhodnocení dotazu funguje, potřebujeme vytvořit tabulku **PLAN_TABLE** (podle skriptu *utlxplan.sql),* kam optimalizátor ukládá své vítězné plány právě vyhodnoceného dotazu. Pro vysvětlení (tj. zjištění optimálního plánu) vyhodnocení dotazu použijeme příkaz **EXPLAIN_PLAN.**

```
explain plan for select p.NAZEV,m.ZKRATKA, ...
```

Vykonáním tohoto příkazu se vítězný plán uloží do tabulky PLAN_TABLE v podobě několika záznamů. Informace vyčteme s použitím dotazu:

```
select plan_table_output from table(dbms_xplan.display());
```

CBO/RBO

Oracle doporučuje používat pouze CBO, který je stále vylepšován a RBO je implementován hlavně kvůli zpětné kompatibilitě.

RBO (Rule Based Optimization)

- Starší přístup, dnes často deprecated (Oracle),
- Odvozuje plán ze syntaxe příkazu a existence indexů
  řídí se předem sestavenou sadou pravidel, která nezohledňují např:
  - velikost tabulky -- ***Malá tabulka*** (obsahuje 5 řádků a vejde se do jednoho datového bloku), je rychlejší jí přečíst celou než hledat podle indexu (1 IO operace vs. čtení bloku indexů a pak dat)
  - Možnost špatného výběru použitého indexu - Pokud existuje více neunikátních indexů na jedné tabulce, nemusí optimalizátor vybrat ten nejlepší. Použití určitého indexu je možné optimalizátoru znemožnit použitím výrazu v dotazu.

Ceny přístupu k podmnožině řádek v tabulce v klesajícím pořadí:

- *Plný přístup (Full scan)* – prochází se celá tabulka – všechny záznamy, u každé řádky se ověří podmínka. Vhodné, pokud procento vyhovujících řádek bude velké.
- *Index-Range-Scan* – vyhledání intervalu v indexu. Ověření ostatních podmínek v odkazovaných řádcích
- *Unique-Index-Scan* – vyhledání jediné možné vyhovující řádky podle unikátního indexu
- *ROWID-Scan* – vyhledání řádky na základě známé hodnoty jejího fyzického identifikátoru v DB.

- Z důvodu kompatibility je možné jej však aktivovat odpovídajícím hintem RULE.

CBO (Cost Based Optimization)

- Hledá plán s nejmenšími náklady pomocí statistik. Optimalizace je založena na vyhodnocení kvantitativního využívání zdrojů v průběhu zpracování dotazu (CPU, paměť, I/O,…). Využívá **statistiky** o tabulkách a datech, které jsou uloženy v Data Dictionary:

- Počet různých hodnot ve sloupci, histogramy rozložení hodnot ve sloupci, počet řádek v tabulce, průměrná délka jedné řádky.

- Některé statistiky jsou přístupné i pro uživatele db pomocí pohledů, či tabulek

- Aktualizují se výpočtem nebo odhadem

- **Typy statistik**:

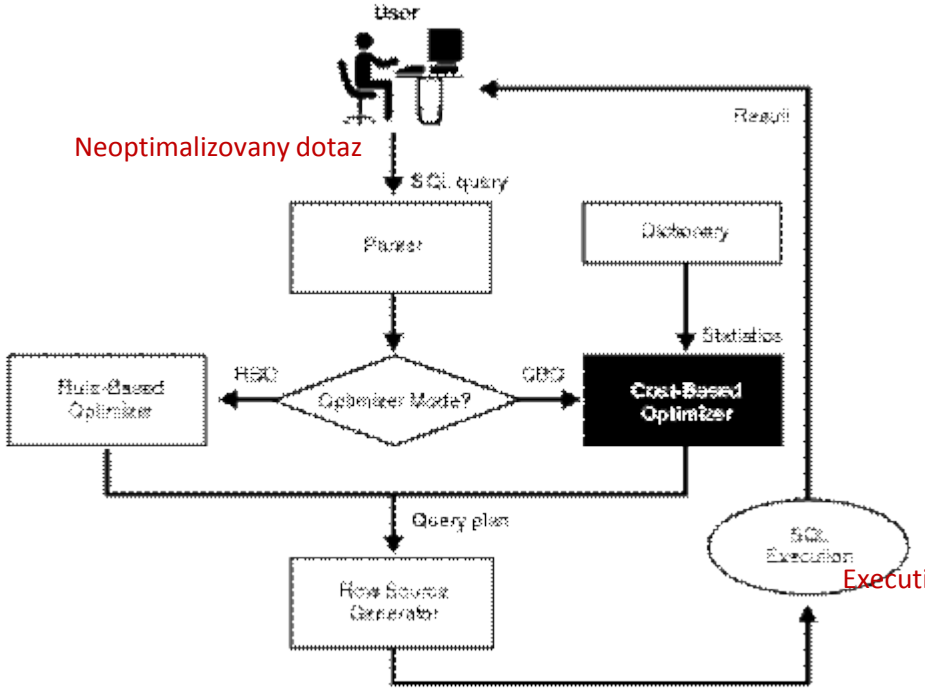
  - **Údaje o tabulkách** – počet řádků, bloků, délka záznamu
  - **Údaje o sloupcích** – počet unikátních hodnot a NULL, histogram
  - **Údaje o indexech** – počet listových bloků, clustering

- Dokáže rozlišit plány i pro různé typy konstant v dotazu.

- Použití CBO je doporučeno firmou Oracle. Vybírá se plán s nejnižší váženou cenou.

Podstata optimalizátoru a přínos

Podstata: stejná data lze z databáze získat různými dotazy (SELECT * vs. SELECT col1, col2…), výsledek bude stejný, ovšem zpracování se může lišit potřebným časem a systémovými nároky.



Execution plan - stromova struktura

Výstupem optimalizátoru je plán vykonávání (execution plan), který určije:

- Přístupové cestu k jednotlivým tabulkám používaných dotazem a pořadí jejich spojování (JOIN order)

Hints

Hint = podnět, kterým optimalizátoru určíme, jaký má použít plán vykonávání dotazu. Hinty se aplikují na blok dotazu, ve kterém se vyskytují.

V CBO lze využít pro optimalizaci i nápovědu – Hints. Prostřednictvím ní mohu optimalizátoru vnutit některou operaci, protože si myslím, že její užití přispěje k lepší optimalizaci. Tato nápověda se zapisuje jako komentář specifického tvaru.

```
explain plan for select /*+ INDEX (predmety predmety_index1) */
p.NAZEV,…
```

Indexy

- Tvorba indexů není v SQL-92 standardizována
- Jednotlivé databázové systémy řeší tvorbu indexů svými prostředky, které jsou navzájem více či méně podobné
  - Může se lišit syntaxe, podpora různých typů indexů, jejich použití/nepoužití pro daný dotaz
- **B-tree indexy**
  - Obvykle redundantní B+ stromy
    - Hodnoty v listech
    - Listy oboustranně linkované pro snadný sekvenční průchod
    - Vhodné pro sloupce s vysokou selektivitou (počtem různých hodnot ve sloupci)
    - Vícesloupcové (složené) indexy mohou zvýšit selektivitu
    - Nad jednou tabulkou v jednom dotazu nelze obvykle kombinovat více B-tree indexů. Dotaz se vyhodnocuje s použitím jednoho z indexů a ostatní podmínky se dopočítávají.
- **Bitmapové indexy**
  - Pro každou hodnotu sloupce/výrazu vytvořen binární řetězec obsahující 1 právě pro řádky s danou hodnotou
    - Vhodné pro sloupce s nízkou selektivitou
    - Lze kombinovat více bitmapových indexů nad jednou tabulkou pro zvýšení selektivity
    - Kombinací více bitmap se zvyšuje selektivita indexu
- Indexy nepomohou
  - Pokud je procento vyhovujících záznamů velké (zvýšená režie s přístupem k řádkům v nesekvenčním pořadí daném indexem
  - Při dotazech na hodnotu null
    - V indexech se běžně neukládá
- Indexy pomohou
  - V dotazech na rovnost sloupce s konstantou
  - V dotazech na to, zda je hodnota v intervalu
- Indexy jsou automaticky vytvářeny
  - Pro primární klíče
  - Pro sloupce s UNIQUE (kandidátní klíče)
- **Vždy vytvářet indexy pro cizí klíče!!!**
  - Zrychlení odezvy při manipulaci s nadřízenou tabulkou
  - Průchod přes index najde efektivně všechny existující závislé řádky bez nutnosti čtení celé tabulky

Výběr typu optimalizace

Je dalším krokem, kterým můžeme ovlivnit optimalizaci. Jedná se o změnu optimalizačního typu optimalizátoru SŘBD Oracle. Typy optimalizace jsou:

- CHOOSE – výběr podle (ne)přítomnosti statistik nejsou-li k dispozici, potom RBO, jinak CBO.
- ALL_ROWS – vždy CBO, minimalizuje se cena za získání všech řádek odpovědi. Vhodné pro dávkové zpracování.
- FIRST_ROWS – vždy CBO, minimalizuje se cena za získání prvních řádek odpovědi. Vhodné pro interaktivní zpracování.

- RULE – vždy RBO.

Typ optimalizace se vybírá příkazem:

```
ALTER SESSION SET OPTIMIZER_GOAL = ALL_ROWS;
```

Další možnost ladění je změna módu optimalizátoru. Dotazy mohou být optimalizovány na:

- Nejlepší průchodnost – ALL_ROWS
- Nejrychlejší odezvu – FIRST_ROWS_1 – zkrátí čas vyhodnocení dotazu.

Př. nastavení módu:

```
ALTER SESSION SET optimizer_mode = all_rows;
```

Obecná pravidla pro psaní SQL dotazů

Vyplývají z technik optimalizace:

- V selectu nepoužívat v seznamu sloupců *, protože ve většině případů nepracujeme se všemi
- Používat co nejméně klauzuli LIKE, IN, NOT IN (vhodnější je WHERE a WHERE NOT EXISTS)
- Používat klauzule typu LIMIT
- Používat hinty (podnět, kterým optimalizátoru určíme, jaký má použít plán vykonávání dotazu)
- Na začátek dávat obecnější podmínky (takové, po kterých vypadne co nejvíc záznamů)
- Výběr vhodného pořadí spojení
- Nastavit indexy

Přínos:

- zdrojový čas
- kapacitu paměti či prostoru
- programátorskou práci
- přenesená data