Optimalizace SQL dotazů

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

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.