CIDUG

Sdružení uživatelů Informix & DB2

IDS optimalizátor

Ing. Jan Musil, IBM ČR

Community of Practice for CEEMEA

Agenda

- Optimalizační plán dotazu
- Typy přístupových plánů
- Metody pro spojení tabulek
- Určení optimalizačního plánu
- Vyhodnocení přístupových plánů k tabulkám
- Určení pořadí spojení tabulek
- Distribuční statistiky
- Ovlivnění činnosti optimalizátoru
- Doba zpracování dotazu
- Příklady z praxe

Optimalizační plán dotazu

- Postup, který optimalizátor určil pro efektivní způsob výběru požadovaných dat
- Optimalizační plán dotazu (query plan)
  - Přístupové plány (access plans)
  - Plány pro spojení tabulek (join plans)
- Cílem je
  - Redukce počtu čtení stránek z disku
  - Eliminování operací třídění

Typy přístupových plánů

- Sekvenční čtení (sequence scan)
- Indexové čtení (index scan)
- Key-only indexové čtení (key-only index scan)
- Key-first indexové čtení (key-first index scan)
- Auto-index čtení (auto-index scan)

Metody pro spojení tabulek

- Nested-loop join
  - Každému záznamu z první tabulky je vyhledán záznam z druhé tabulky
  - V případě existence indexu, je použit index
- Hash join
  - Vytvoření hash tabulky z jedné, obvykle menší tabulky (build) a sekvenční čtení záznamů z druhé tabulky (probe)
  - Porovnání záznamů z druhé tabulky proti hash tabulce

Určení optimalizačního plánu

- Vyhodnocení přístupových plánů k tabulkám
- Určení pořadí spojení tabulek a metody spojení

- Klíčové informace jsou uloženy v distribučních statistikách

Vyhodnocení přístupových plánů k tabulkám

- Určení selektivity každého filtru a pořadí aplikace filtrů
- Rozhodnutí, zda lze použít index
  - Filtry, ORDER BY, GROUP BY
- Určení nejlepšího způsobu, jak vyhledat požadovaná data v tabulce
  - Sekvenční čtení
  - Použití indexu

Filtry a selektivita

- Filtr
  - každý jednotlivý výraz ve WHERE podmínce SQL dotazu
- Selektivita
  - Odhad optimalizátoru, kolik dat bude muset být přečteno při použití určitého filtru
  - Hodnota blízká nule – velmi selektivní filtr
  - Hodnota rovná 1 – filtru odpovídají všechny záznamy
  - Určuje se buď z distribucí nebo aplikací funkce
  - Určuje pořadí aplikace filtrů (od nejvíce po nejméně selektivní filtry)

Určení pořadí spojení tabulek

- S použitím přístupových plánů a metod spojení se vyhodnotí váhy všech možných dvojic tabulek
- Z ekvivalentních dvojic (např. AB, BA) se vybere pouze dvojice s nižší váhou
- U spojení více než dvou tabulek se pokračuje vytvořím trojic, čtveřic, atd. a vybere se spojení s nejnižší váhou

Distribuční statistiky

- Používají se pro určení selektivity filtrů, přístupových plánů a metod spojení tabulek
- UPDATE STATISTICS
  - LOW
    - Bez distribučních statistik
  - MEDIUM
    - Distribuční statistiky jsou vytvořeny na základě vzorku dat
  - HIGH
    - Distribuční statistiky vytvořené ze všech dat

Ovlivnění činnosti optimalizátoru

- OPTCOMPIND
  - Konfigurační parametr
  - Proměnná prostředí
  - SQL příkaz pro dynamické nastavení
- SET OPTIMIZATION
  - LOW/HIGH
  - All rows/First rows
- Optimalizační direktivy
  - Možné někdy jindy ... 

OPTCOMPIND

0	- přístupový plán: index, pokud existuje - spojení tabulek: nested-loop join - nevyhodnocuje váhy !!!
1	- úroveň izolace Repeatable Read: 0 - jiná úroveň izolace než Repeatable Read: 2
2	- vyhodnocuje váhy - vybere optimalizační plán s nejnižší váhou - přístupový plán: index nebo sekvenční čtení - spojení tabulek: nested-loop nebo hash join

- ONCONFIG konfigurační parametr
- Proměnná prostředí: OPTCOMPIND
- SQL: set environment optcompind “0|1|2|default“

SET OPTIMIZATION

- HIGH (default)
- LOW
  - Vybere pouze kombinaci s nejnižší váhou v každé úrovni, ostatní možnosti již nevyhodnocuje

SELECT .... FROM a,b,c,d WHERE …

ab ac ad bc …

acb acd

acdb

SET OPTIMIZATION LOW

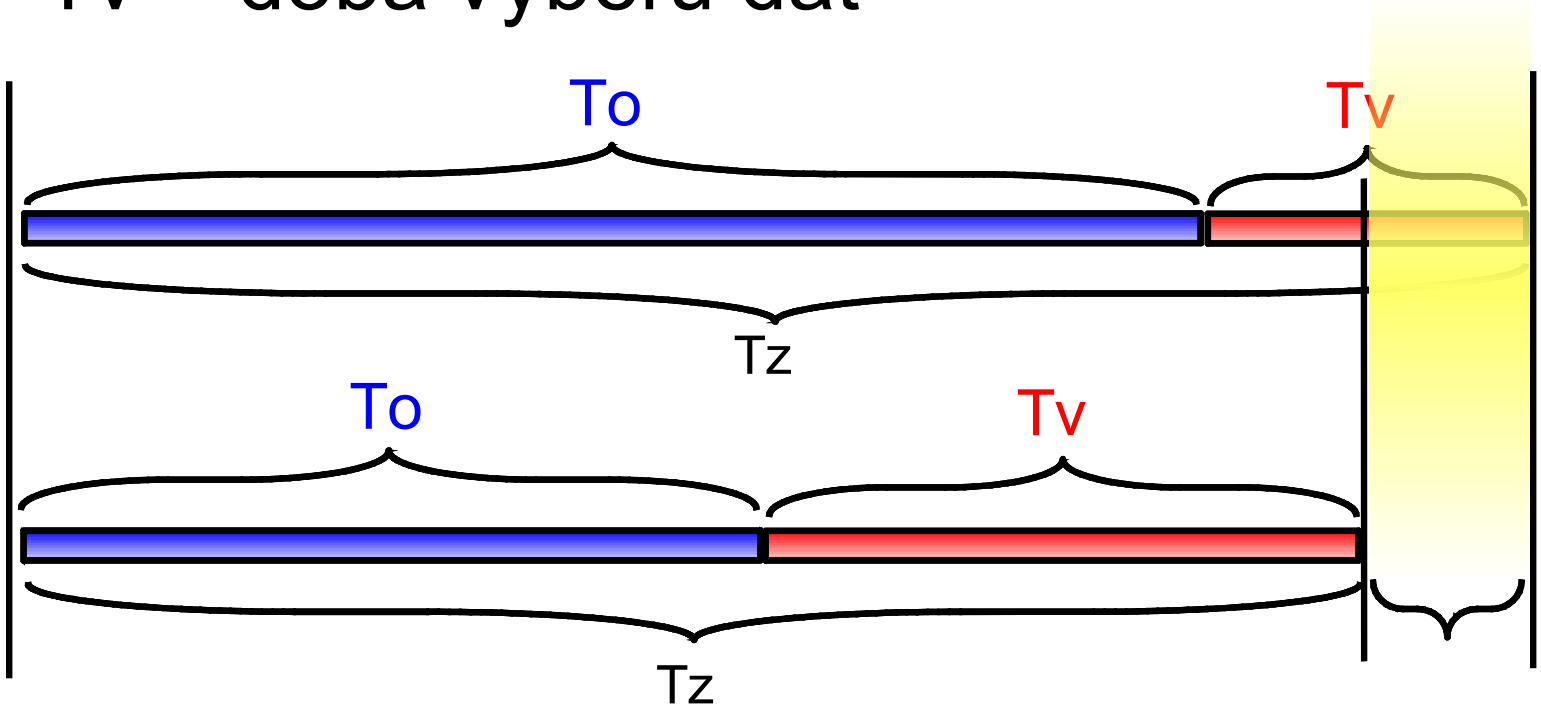
- Doba provedení dotazu je velmi dlouhá
- Spojení většího počtu tabulek (>=5)
- Všechny join položky jsou indexované
- Join jedné tabulky s ostatními

SET OPTIMIZATION

- ALL_ROWS (default)
- FIRST_ROWS
  - Dotaz je optimalizovaný tak, aby vrátil co nejrychleji první výsledky
  - Celková doba provedení dotazu může být delší než v případě standardní optimalizace ALL_ROWS
  - Např.: hash join může být nahrazen nested-loop joinem

Doba zpracování dotazu

- Tz = ceková doba zpracování dotazu
- To = doba optimalizace
- Tv = doba výběru dat



Příklady z praxe: případ první

```
create table tabulka (
  id1 integer,
  id2 smallint,
  kod integer,
  .....
);

create index i_tab1 on tabulka(id1,id2);
create index i_tab2 on tabulka(kod);

create tabulka ciselnik (
  kod integer,
  txt char(30),
);

create index i_cis1 on ciselnik(kod);
create index i_cis2 on ciselnik(txt);
```

SELECT ....
FROM tabulka,ciselnik, …
WHERE
tabulka.kod=ciselnik.kod AND
tabulka.id1=123456 AND
ciselnik.txt="TEXT" AND
.....

Zadání:
Provedení optimalizace a vysvětlení odlišnosti chování optimalizátoru pro různé hodnoty OPTCOMPIND

Příklady z praxe: případ první

OPTCOMPIND=0	Estimated Cost: 50 Estimated # of Rows Returned: 2 Temporary Files Required For: Order By 1) ciselnik: INDEX PATH (1) Index Keys: txt (Serial, fragments: ALL) Lower Index Filter: ciselnik.txt = 'TEXT' 2) tabulka: INDEX PATH Filters: tabulka.id1 = 123456 (1) Index Keys: kod (Serial, fragments: ALL) Lower Index Filter: tabulka.kod = ciselnik.kod NESTED LOOP JOIN
OPTCOMPIND=2	Estimated Cost: 29 Estimated # of Rows Returned: 2 Temporary Files Required For: Order By 1) ciselnik: INDEX PATH (1) Index Keys: txt (Serial, fragments: ALL) Lower Index Filter: ciselnik.txt = 'TEXT' 2) tabulka: INDEX PATH Filters: tabulka.kod = ciselnik.kod (1) Index Keys: id1 id2 (Serial, fragments: ALL) Lower Index Filter: tabulka.id1 = 123456 NESTED LOOP JOIN

Příklady z praxe: případ druhý

```
create table tabulka (
   .....
   zacatek date,
   konec date,
   .....
);
```

```
SELECT ....
FROM tabulka
WHERE
konec >= '07.06.2005' AND
zacatek <= '07.07.2005'
.....
```

Popis problému:

- OPTCOMPIND=0
- Pokus 1
  - indexy pro položky *zacatek* a *konec* nejsou vytvořeny
  - trvání dotazu je přibližně stejné bez ohledu na distribuce
  - *K čemu jsou tedy distribuce, když nepomohou ???*
- Pokus 2
  - indexy na položkách *zacatek* a *konec* jsou vytvořeny
  - distribuce jsou korektně vytvořeny
  - Doba provedení dotaz je nejdelší !!!
  - *Proč indexy a distribuce tak zhoršily situaci ?*

Příklady z praxe: případ druhý

Indexy a distribuce nejsou vytvořeny

```
Estimated Cost: 437022
Estimated # of Rows Returned: 41877

 1) poji.k_ziv: SEQUENTIAL SCAN

        Filters: (poji.k_ziv.datpoc <= 07.07.2005 AND poji.k_ziv.datkon >= 07.06.2005 )
```

Byly vytvořeny distribuce, ale bez indexů

```
Estimated Cost: 2446748
Estimated # of Rows Returned: 368539

  1) poji.k_ziv: SEQUENTIAL SCAN

        Filters: (poji.k_ziv.datkon >= 07.06.2005 AND poji.k_ziv.datpoc <= 07.07.2005 )
```

Byly vytvořeny distribuce i indexy

```
Estimated Cost: 2494646
Estimated # of Rows Returned: 368539
  1) poji.k_ziv: INDEX PATH
        Filters: poji.k_ziv.datkon >= 07.06.2005
    (1) Index Keys: datpoc   (Serial, fragments: ALL)
        Upper Index Filter: poji.k_ziv.datpoc <= 07.07.2005
```

Děkuji za pozornost

jan_musil@cz.ibm.com