Vysoké učení technické v Brně
Fakulta informačních technologií

Databázové Systémy

Dokumentace ke 4. části projektu
Zadání č. 28 - Nemocnice

Pavel Heřmann (xherma34)
Alexander Sila (xsilaa00)
2. května 2022

1 Triggery

1.1 personal_number

Tento trigger využívá sekvence, která se inkrementuje a byl vytvořen za účelem generování primárního klíče tabulce `Personel`.

1.2 distribute_evenly

Trigger byl vytvořen za účelem rovnoměrné rozložení práce mezi doktory se stejnou specializací.

Na začátek se načte specializace daného doktora do proměnné `spec`, podle které se najdou všichni dostupní doktoři. Následuje jejich porovnání podle počtu hospitalizací, to je prováděno příkazem `SELECT`, který spočítá počet hospitalizací doktorů dané specializace a následně je seřadí od nejmenšího po nejvyšší. Doktorovi s nejnižším počtem hospitalizací je následně přidělen nový pacient.

2 Procedury

2.1 who_has_medicament

Tato procedura příjma jako parametr ID léku a ID doktora, podle kterých vyfiltruje všechny pacienty, kteří berou tento lék a byli hospitalizování doktorem s daným ID.

Využívá cursoru `medicines`, který nám "propojí"potřebné tabulky pro provedení procedury. Následně se z kurzoru vytahují jednotlivé řádky do proměnné `cursor_row` a požadované informace jsou vypisovány na výstup.

V případě, že některá z informací neexistuje, je vyvolána vyjímka.

2.2 doctor_prescribed_stats

Využívá cursoru `doctors`, přes který přístupuje k informacím o všech doktorech. Postupně se prochází tabulkou po řádcích. Nejdříve se pomocí příkazu `SELECT` uloží do proměnných `first_name, last_name` jméno a příjmenní doktora a dalším příkazem `SELECT` se do proměnné `number_of_prescripted_meds` uloží počet jím předepsných léků. Informace uložené v porměnných jsou následně tisknuty na výstup.

3 Přidělení práv

Přidělení práv bylo uděleno příkazem `GRANT ALL ON`, který byl použit na každou z tabulek databáze.

4 Explain Plan

Pomocí Explain Plan získáme informace důležité pro optimalizaci jako např.: pořadí vykonaných instrukcí, čas a zatížení cpu v podobě "ceny".

Rozhodli jsme se optimalizovat příkaz SELECT, který zjišťuje kolikrát byl pacient s určitým jménem hospitalizován v jednotlivých letech.

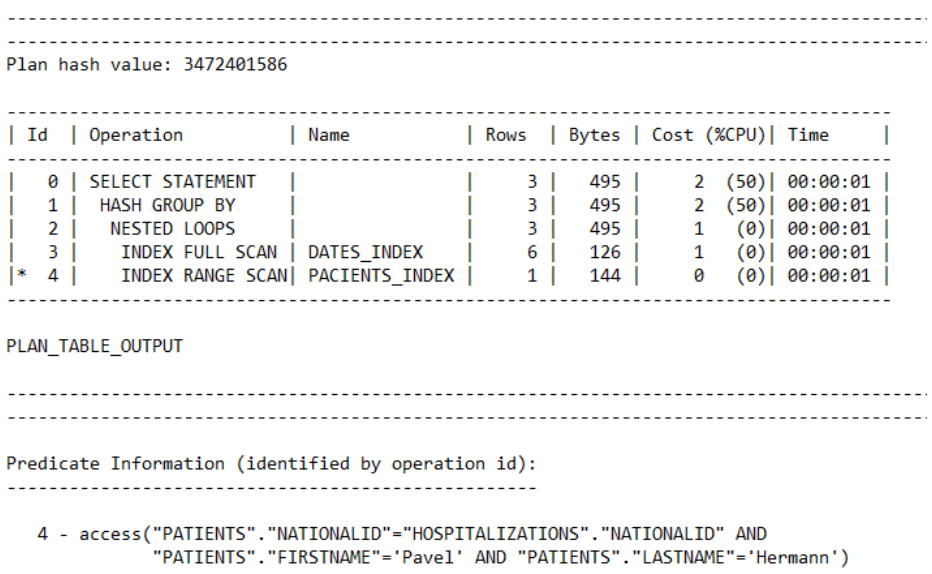
Pro optimalizaci zmíněného příkazu je nutno vytvořit indexy pro tabulky, které příkaz využívá. První index spojuje sloupce `NationalID, firstname, lastname` z tabulky `Patient`, druhý potom sloupce `NationalID, HosDate` z tabulky `Hospitalizations`. Jelikož jsou hodnoty v daném příkazu zaindexovány, tak může jednoduše SQL engine přistoupit k daným informacím tabulek, za pomocí reference

```sql
--Kolikrat byl pacient Pavel Hermann hospitalizovan v jednotlivych letech
SELECT count(*), TO_CHAR(HosDate, 'YYYY') FROM Patients NATURAL JOIN Hospitalizations
    WHERE FirstName = 'Pavel' AND LastName = 'Hermann' GROUP BY TO_CHAR(HosDate, 'YYYY');
```

Obrázek 1: Příkaz SELECT

```
--------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------
Plan hash value: 2613028984

-------------------------------------------------------------------------------------------
| Id  | Operation           | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                  |     3 |   495 |     7  (15)| 00:00:01 |
|   1 |  HASH GROUP BY      |                  |     3 |   495 |     7  (15)| 00:00:01 |
|*  2 |   HASH JOIN         |                  |     3 |   495 |     6   (0)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL| PATIENTS         |     1 |   144 |     3   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| HOSPITALIZATIONS |     6 |   126 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

PLAN_TABLE_OUTPUT

--------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("PATIENTS"."NATIONALID"="HOSPITALIZATIONS"."NATIONALID")
   3 - filter("PATIENTS"."FIRSTNAME"='Pavel' AND
              "PATIENTS"."LASTNAME"='Hermann')
```

Obrázek 2: Explain Plan před použitím indexů



Obrázek 3: Explain Plan po použití indexů

5 Materializovaný Pohled

Materializovaný pohled byl vytvořen nad příkazem `SELECT`, který vypíše doktora který udělal nejvíce hospitalizací za daný kalendářní rok.

```
SELECT * FROM (SELECT firstname, lastname, count(*) pocet
    FROM xsilaa00.Personel NATURAL JOIN xsilaa00.Doctors NATURAL JOIN xsilaa00.Hospitalizations
    WHERE '2020' = TO_CHAR(HosDate, 'yyyy')
    GROUP BY firstname, lastname ORDER BY pocet DESC) WHERE rownum = 1;
```

Obrázek 4: Materializovaný pohled