Vysoké učení technické v Brně

Fakulta informačních technologií

Dokumentace projektu z předmětu IDS 2014/2015

SQL skript pro vytvoření pokročilých objektů schématu databáze - obchod s čajem

Autoři: Jan Pawlus [xpawlu00]
Zdeněk Studený [xstude21]

Obsah

1 Popis implementace specializace 2

2 Triggery 2

3 Procedury 2

4 Index a explain plan 2

5 Přístupová práva a materializovaný pohled 3

1 Popis implementace specializace

Specializace je implementována tím způsobem, že tabulka `Odberatel` má další podtabulky - `Fyzicka_osoba` a `Firma`. Obě tyto podtabulky obsahují jako cizí klíč `id_odberatel`, který je primárním klíčem v tabulce `Odberatel`. Tímto je dosaženo specializace.

2 Triggery

První `TRIGGER` slouží společně se `SEQUENCE` pro auto increment primárního klíče při vkládání do tabulky `Varka`. `TRIGGER` aktivuje před vkládáním do tabulky a zavolá se `SEQUENCE`, která nastaví hodnotu klíči o jednu větší.

Druhý `TRIGGER` slouží k ověřování správnosti formátu IČO. Jsou definovány dvě funkce, které ověřují, zda je IČO opravdu ve správném formátu. Volání těchto funkcí zajišťuje právě `TRIGGER`, jenž se spustí před vkládáním hodnoty IČO.

3 Procedury

Jako proceduru s využitím `CURSOR` jsme zvolili hledání nejdražší várky. Jsou deklarovány dvě výstupní proměnné, které budou uchovávat ID nejdražší várky a samotnou cenu. Nejprve je potřeba definovat dotaz pro `CURSOR`, ten vybere všechny várky. `CURSOR` poté slouží k tomu, abychom byli schopní porovnávat ceny jednotlivých záznamů v cyklu.

Další procedury slouží k mazání sekvencí, tabulek a indexů. Hodí se, pokud je třeba obsluhovat výjimku (jako například chybu, pokud mazaný prvek v databázi neexistuje).

4 Index a explain plan

`EXPLAIN PLAN` Popisuje postup provedení dotazu nad databází. Pro dotaz:

```
SELECT Varka.id_caj, COUNT (Varka.id_varka) as pocet_varek
FROM Varka
INNER JOIN Caj ON Caj.id_caj = Varka.id_caj
WHERE cena > 2500
GROUP BY Varka.id_caj;
```

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |       |     3 |    21 |     4  (25)| 00:00:01 |
|   1 |  HASH GROUP BY     |       |     3 |    21 |     4  (25)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| VARKA |     3 |    21 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Provedení je následující: Po načtení `SELECT` statementu se provede hashovaní funkce `GROUP BY` a poté se přistoupí k celé tabulce `Varka`. Zde vidíme možnosti optimalizace, provedení `GROUP BY` je poměrně náročné na paměť i procesor. Dále se také přistupuje k celé tabulce `Varka`, což by šlo optimalizovat zavedením `INDEX` nad cenou várky. V následujícím přikladu je uveden výsledek právě po použití `INDEX`. Tabulka je zkrácena o sloupce rows a bytes.

```
 ------------------------------------------------------------------------------
| Id  | Operation                           | Name      | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |           |     3  (34)| 00:00:01 |
|   1 |  HASH GROUP BY                      |           |     3  (34)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| VARKA     |     2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN                 | INDEXCENA |     1   (0)| 00:00:01 |
------------------------------------------------------------------------------
```

5 Přístupová práva a materializovaný pohled

Pro nastavení přístupových práv jinému uživateli je třeba projít všechny tabulky provést `GRANT SELECT ON TABULKA TO UZIVATEL`. Nabízí se tedy opět využití `CURSOR` stejně, jako v proceduře hledající nejdražší čaj - `SELECT` vybere všechny tabulky a v cyklu se poté nastavuje pro každou oprávnění.

Materializovaný pohled je kopie cílové databáze v časový moment. V tomto případě uživatel `XPAWLU00` přidělil práva uživateli `XSTUDE21`. Ten si poté může udělat pohled příkazem `CREATE MATERIALIZED VIEW` a později tak může přistoupit k těmto datům, i když nebude připojen k databázím uživatele `XPAWLU00`. Příklad použití `MATERIALIZED VIEW` a jednoduchého dotazu `SELECT` z něj.

```
CREATE MATERIALIZED VIEW guestTable FOR UPDATE AS
SELECT * FROM XPAWLU00.caj;
SELECT id_caj, nazev_caj FROM guestTable;
```