

Dokumentácia k projektu, predmet IDS

Zadanie: Truhlářství

26.4.2020

Lukáš Javorský – xjavor20

Patrik Ondriga – xondri08

Obsah:

ZADANIE	1
TABUĽKY V DATABÁZE	2
IMPLEMENTÁCIA	3
Trigre (triggers)	3
<i>login_update_zakazka.....</i>	<i>3</i>
<i>generate_poradove_cislo.....</i>	<i>3</i>
Procedúry (procedures)	3
<i>zakaznik_nespravny_email().....</i>	<i>3</i>
<i>zmazat_zamestnanca(Vymazat_RC)</i>	<i>4</i>
Explain plan a index.....	4
<i>Výstup z Explain plan bez indexu.....</i>	<i>4</i>
<i>Vytvorenie indexu.....</i>	<i>5</i>
<i>Výstup z Explain plan s indexom.....</i>	<i>5</i>
Materializovaný pohľad	5
ZÁVER	7

Zadanie

Navrhňte informační systém malého truhlářství, které přijímá zakázky od zákazníků. Každá zakázka má specifické požadavky. Každý kus nábytku má určen materiál, barvu, rozměry, příslušenství, atd. Systém uchovává informace o využitém materiálu na zakázce - materiál, množství, cena, atd. Firma odebírá materiál od více dodavatelů, přičemž každý z dodavatelů může mít jiné ceny. Každá zakázka má zodpovědného zaměstnance, který řídí celou zakázku, přiděluje další zaměstnance na zakázku, určuje jaký materiál se použije apod. Na zakázce může pracovat více zaměstnanců; každý zaměstnanec má svoji specializaci. Firma může na zakázku najmout i externího zaměstnance, který má opět svoji specializaci a hodinovou mzdu. Jednotlivé položky na zakázce jsou fakturovány (použitý materiál, odpracované hodiny zaměstnanců, ...), z těchto položek je pak vyhotovena celková faktura zakázky.

Tabuľky v databáze

- Material_doda_dodavatel
- Nabytok_mozne_vyrobit_z_materialu
- Kus_vyrobeny_z_materialu
- Nabytok_ma_prislusenstvo
- Kus_ma_prislusenstvo
- Material
- Dodavatel
- Kus_ma_farbu
- Prislusenstvo
- Nabytok_ma_farbu
- Kus
- Nabytok
- Farba
- Zamestnanec_pracuje_na_zakazke
- Zakazka
- Zakaznik
- Zamestnanec

Kvôli implementácii väzieb N ku N z ERD diagramu, sme museli vytvoriť nasledovné tabuľky.

- Material_doda_dodavatel
- Nabytok_mozne_vyrobit_z_materialu
- Kus_vyrobeny_z_materialu
- Nabytok_ma_prislusenstvo
- Kus_ma_prislusenstvo
- Kus_ma_farbu
- Nabytok_ma_farbu
- Zamestnanec_pracuje_na_zakazke

Implementácia

Nasledovný opis je určený implementácii projektovej časti 4.

Trigre (triggers)

V našom sql skripte sa nachádzajú 2 trigre:

login_update_zakazka

- Trigger má za úlohu sledovať zmenu zákazníckeho loginu, a kvôli konzistencii dát túto zmenu aplikuje aj na login, v rámci zákazky, v ktorej splňuje úlohu cudzieho kľúča.

generate_poradove_cislo

- Trigger, s využitím *sekvencie* (*zakazka_poradove_cislo_automat*). Tento trigger je navrhnutý pre automatické vytváranie poradového čísla pre zákazky. Poradové čísla začínajú od 1 a postupnou inkrementáciou sú priradované k novým zákazkám.

Procedúry (procedures)

Procedúry, ktoré obsahuje náš projekt majú slúžiť užívateľovi, aby mu zjednodušili prácu s databázovými objektami.

zakaznik_nespravny_email()

- Procedúra, ktorá prechádza každého zákazníka pomocou *kurzora* a kontroluje jeho emailovú adresu. Emailová adresa musí byť vo formáte *%@%.%*, kde *%* znamená neprázdna sekvencia znakov.
- Procedúra taktiež počíta, koľko krát sa vyskytol zákazník s nesprávnym emailom, ukladá tento počet do premennej *pocet_nespravnych* a tento počet je potom vypísaný pomocou *dbms_output.put_line*.
- Pri každom výpise nesprávneho emailu je vypísaný aj *Login*, *Meno* a *Priezvisko* a *nespravny Email* tejto osoby.
- Pokiaľ sa nenašli žiadne dáta (*NO_DATA_FOUND*), procedúra vyvolá error s kódom -20003 a priliehajúcou správou.

zmazat_zamestnanca(Vymazat_RC)

- Procedúra vykonáva vymazanie zamestnanca z databáze na základe zadaného rodného čísla.
- Prechádza sa záznam po zázname v tabuľke *Zamestnanec* a porovnáva sa zadané rodné číslo s hodnotou v stĺpci *RC*.
- V prípade, že hľadaný zamestnanec má pridelenú zákazku, je procedúra ukončená s chybovou hláškou „*Zamestnanec ma priradenú zakazku, neda sa odstranit*“ a vyvolá sa error kód -20081.
- Pokiaľ nie je priradený k žiadnej zákazke, vypíše sa jeho meno, priezvisko a následne je jeho záznam odstránený.

Explain plan a index

Pomocou tohto výrazu si vieme zobrazit' časovú, výpočtovú a pamäťovú náročnosť príkazov. My sme si zobrazili údaje pri vykonávaní príkazu *SELECT*.

```
EXPLAIN PLAN FOR
SELECT nazov_farby as Farba, count(nazov_farby) as Pocet_nabytku FROM Nabytok_ma_farbu
GROUP BY (nazov_farby)
ORDER BY count(nazov_farby) desc;
```

Príkaz *SELECT* vo výraze *EXPLAIN PLAN* vypíše farby a počet rôzneho nábytku ktorý môže byť vyrobený v danej farby.

Výstup z Explain plan bez indexu

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		9	108	5 (40)	00:00:01
1	SORT ORDER BY		9	108	5 (40)	00:00:01
2	HASH GROUP BY		9	108	5 (40)	00:00:01
3	TABLE ACCESS FULL	NABYTOK_MA_FARBU	9	108	3 (0)	00:00:01

Note

- dynamic statistics used: dynamic sampling (level=2)

Z výstupu môžeme vidieť, že sa vykonajú 4 operácie v následovnom poradí.

1. Výber
2. Zoradenie skupiny farieb zostupne podľa ich názvu
3. Interné zoradenie pomocou hashu

4. Prístup do tabuľky „NABYTOK_MA_FARBU“

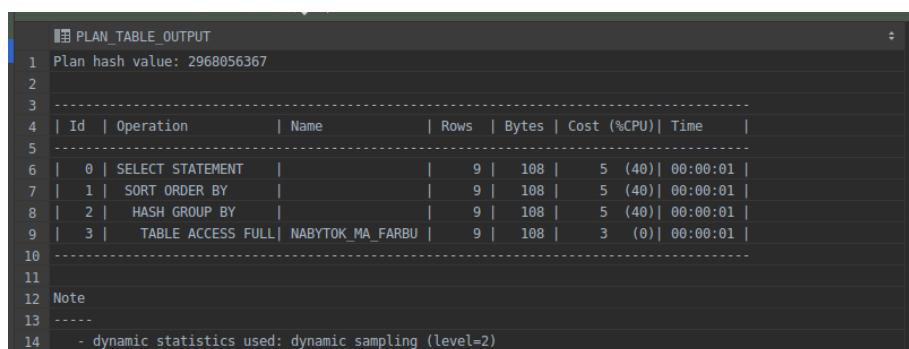
Vytvorenie indexu

Index vytvára odkazy na rovnaké položky v stĺpci pre ktorý sme index vytvorili. Tým pádom je prehľadávanie tohto stĺpca rýchlejšie. Nevýhodou je, že tvorenie indexu zaberá dosť času a pri vložení nových záznamov sa index musí znova vytvoriť.

V našom prípade tabuľku prehľadávame a zoradujeme podľa stĺpca *nazov_farby*. Preto sme sa rozhodli, že práve pre tento stĺpec je najlepšie vytvoriť index.

```
-- Vytvorenie indexu
CREATE INDEX farba_index ON Nabytok_ma_farbu (nazov_farby);
```

Výstup z Explain plan s indexom



PLAN_TABLE_OUTPUT

1 Plan hash value: 2968056367

2

3

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		9	108	5 (40)	00:00:01
1	SORT ORDER BY		9	108	5 (40)	00:00:01
2	HASH GROUP BY		9	108	5 (40)	00:00:01
3	TABLE ACCESS FULL	NABYTOK_MA_FARBU	9	108	3 (0)	00:00:01

10

11

12 Note

13 -----

14 - dynamic statistics used: dynamic sampling (level=2)

Aj napriek použitiu indexu, nie je na výstupe vidieť žiadne zrýchlenie. Nie je to však dôsledkom vytvorenia zlého indexu, ale počtom záznamov. V našej databáze je pokusne uložených 9 záznamov v tabuľke *Nabytok_ma_farbu*. Na to, aby sa prejavilo zrýchlenie by sme tých záznamov tam museli mať omnoho viac. Preto je vytvorenie indexu dobrým krokom pre budúce využívanie databáz.

Materializovaný pohľad

Materializovaný pohľad vytvára kópiu výsledku dotazu, čím zrýchľuje prístup k týmto dátam pre neskoršie spracovanie. V prípade vloženia nových záznamov sa pohľad musí aktualizovať.

Najprv sme vytvorili *LOG* pre tabuľku *Kus*.

```
-- Zmeny v logoch tabuľke Kus
CREATE MATERIALIZED VIEW LOG ON Kus WITH PRIMARY KEY, ROWID;
```

Potom sme vytvorili materializovaný pohľad pre výpis všetkých *zákaziek*, zoradené podľa *datum_vytvorenia* zostupne.

```
-- Vypis zakaziek podľa vytvorenia od najnovších
CREATE MATERIALIZED VIEW prehlad_zakaziek_podla_datumu_vytvorenia
  BUILD IMMEDIATE
  REFRESH COMPLETE
AS
SELECT * FROM Zakazka ORDER BY datum_vytvorenia desc;
```


Záver

Počas projektu sme mali možnosť vytvoriť návrh v podobe ER diagramu, ktorý sme následne implementovali ako SQL databázu. Prešli sme si procesom vytvárania tabuliek a vkladania záznamov. V ďalšom kroku sme si vyskúšali, akým spôsobom sa dajú vyťahovať určité informácie pomocou preambule SELECT. Nakoniec sme vytvárali možnosti pre rýchlejší prístup k informáciám nachádzajúcich sa v databáze.