



Databázové systémy
Dátový model a model případu užitia
2018/2019

Zadanie č. 58 - Fitness centrum

29. apríla 2019

Adrián Boros (xboros03)
Tomáš Žigo (xzigot00)

Obsah

| | | |
|----------|---|----------|
| 1 | Zadanie | 2 |
| 2 | Výsledná schéma databáze | 3 |
| 2.1 | Generalizácia / špecializácia | 3 |
| 3 | Implementácia SQL skriptu | 4 |
| 3.1 | Triggery | 4 |
| 3.2 | Procedúry | 4 |
| 3.3 | Index a explain plan | 5 |
| 3.4 | Prístupové práva | 6 |
| 3.5 | Materializovaný pohľad | 6 |

1 Zadanie

Navrhňte jednoduchý IS fitness centra, ktoré organizuje rôzne kurzy skupinových lekcí (zumba, TRX, kruhový tréning, atď.). Ve fitness centru pracujú inštruktori, ktorí vedú jednotlivé skupinové lekcie, a ľudia na recepcii, ktorí sa musia okrem víťania príchodých klientů a mixovania proteinových koktejlů zapojiť do práce s IS fitness centra prostredníctvom vytvárania členských kariet pre jednotlivé klienty, ktorí sa rozhodli pravidelne trápiť svá tela ve fitness centru a chčujú využíť členskú výhodu. Aby karta nebola využívana jinými klienty než jejím vlastníkem, musí být v IS uloženy základní informace o klientech, jejich rodná čísla a adresy. Zákazník si může vypsat kurzy, které navštěvuje a informace o jednotlivých lekcích. Navíc si může zobrazit rozvrh vypisovaných kurzů a zjistit počet volných míst na jednotlivých lekcích a jejich cenu. Zákazník se může registrovat buď na jednu lekci nebo na celý kurz. Kurzy mají svou délku trvání, obtížnost a popis. Skupinové lekce probíhají v různých sálech fitness centra, které mají konkrétní název, umístění a maximální kapacitu. Lekce jsou vedené jedním instruktorem, mají maximální kapacitu účastníků a odehrávají se v daném sále v určitý čas a den v týdnu. Předpokládejte, že jeden instruktor může být vyškolen pro vedení různých kurzů, toto modelujte. Kromě pravidelných skupinových lekcí nabízí fitness centrum i individuální lekce, na kterých se instruktor věnuje pouze jednomu klientovi. Tyto lekce jsou podobného charakteru jako ty skupinové, jen je konkrétnímu klientovi věnováno více pozornosti. Instruktor má možnost vložit do systému nové typy kurzů a konkrétní lekce (a to jak skupinové, tak i individuální) a měnit čas a sál, ve kterém se lekce konají. Systém musí být na požádání schopný vypsat rozvrh pro jednotlivé místnosti.

3 Implementácia SQL skriptu

Skript na začiatku zahodí databázové objekty príkazom `DROP`, aby sa predišlo možným konfliktom. Následne sa vytvoria tabuľky a nastaví sa primárne a cudzie kľúče. Následne sa tabuľky naplnia testovacími údajmi nad ktorými prevádzame niekoľko `SELECT` príkazov podľa požiadavok zo zadania. Skript taktiež obsahuje trigger, procedúry, materializovaný pohľad, použitie indexu s `EXPLAIN PLAN` a pridelenie prístupových práv druhému členovi.

3.1 Triggery

V projekte sme implementovali 3 databázové trigger, pričom všetky sa spúšťajú pred vložením alebo aktualizácií dát do tabuľky (`BEFORE INSERT OR UPDATE`). Skript taktiež obsahuje ukážkové príkazy manipulácie dát ktoré demonštrujú prevedenie jednotlivých triggerov.

Prvý trigger, vyplývajúci zo zadania, slúži na automatické generovanie hodnôt primárneho kľúča - `ID_saly_trig`, ktorý sme použili s tabuľkou `Sála`. Trigger je realizovaný pomocou sekvencie pre uchovanie poslednej hodnoty ktorá sa získa pomocou `nextval` a vloží sa do stĺpca `ID`.

Úlohou druhého triggeru, (`Prevod_meny_na_euro`) je previesť cenu kurzu z českých korún na eurá, pričom pre jednoduchosť sa uvažuje kurz 25.

Posledný trigger je kontrolný, slúži na kontrolu či zadaná maximálna kapacita lekcie nepresahuje maximálnu kapacitu sály v ktorej prebieha. Na začiatku je deklarovaná premenná `kapacita` typu `INTEGER` do ktorej si pomocou `SELECT INTO` priradíme kapacitu sály v ktorej daná lekcia prebieha. Nakoniec pomocou podmienky `IF` sa kontroluje či zadávaná kapacita lekcie nepresahuje maximálnu kapacitu sály, ak áno, zavolá sa výnimka `raise_application_error` s chybovým kódom a hlásením.

3.2 Procedúry

Skript obsahuje 3 procedúry v ktorých je využitý kurzor a takisto aj premenná s dátovým typom odkazujúcim sa na riadok či typ stĺpca tabuľky (`table_name.column_name%TYPE` alebo `table_name%ROWTYPE`).

Procedúra `vyuzitie_saly` vypočíta na koľko percent sú využité všetky sály dokopy v zadaný deň. Argumentom procedúry je deň ktorý je zadaný ako dátum. Uvažuje sa čas od 6:00 do 22:00. Výstupom procedúry je číslo využitia v percentách. Táto procedúra je využiteľná v prípade že by si nejaká firma chcela prenajať viac miestností vo fitness centre, tak aby vedela ktorý deň je najvýhodnejší, čiže kedy sú sály najmenej využívané. Táto procedúra obsahuje aj ošetrenie **výnimiek**, nakoľko sa môže stať že pre zadaný deň neexistuje žiadna lekcja. V tomto prípade sa vyvolá výnimka `NO_DATA_FOUND`.

Druhá procedúra s názvom `priemerna_cena_lekcie` vypočíta priemernú cenu lekcií pre dátum, ktorý je argumentom tejto procedúry. V tejto procedúre využívame kurzor do ktorého sa vyberú lekcje a následne v cykle zvyšujeme celkovú cenu a počet lekcií ak sa zadaný deň zhoduje s dňom v tabuľke. Tieto pomocné premenné (počet lekcií a celková cena) sú definované na začiatku procedúry. Na konci cyklu vydáme celkovú cenu počtom lekcií. Tento výsledok je uložený v premennej `priemerna_cena` ktorého typ je **dátovým typom odkazujúcim sa na stĺpec tabuľky**. V prípade že by bol počet rovný 0, volá sa výnimka `ZERO_DIVIDE`, v opačnom prípade sa výsledok vypíše na výstup.

Procedúra `vekove_rozdelenie_aktivnych_zakaznikov` vypíše vekové rozdelenie aktívnych zákazníkov. Na začiatku je opäť vytvorený kurzor do ktorého sa vyberú informácie o aktívnych zákazníkoch. Nasleduje deklarácia premenných. Cyklom sa potom prechádzajú aktívny zákazníci o ktorých sa potom pomocou funkcií `SUBSTR`, `TO_DATE` získa dátum narodenia. Ak je dátum narodenia `NULL`, čiže ak by bola tabuľka `zakaznik` prázdna, príkazom `RAISE` sa volá výnimka. Následne sa príkazom `TRUNC (MONTHS_BETWEEN (SYSDATE, datum_narodenia) / 12` získa vek. V tomto prípade sa môže stať že vek vyjde záporný, v tom prípade iba pripočítame hodnotu 100 nakoľko dátum je vo formáte `YY-MM-DD`. Potom nasleduje príkaz `CASE` kde sa porovnáva získaný vek a zvyšuje sa pomocná premenná. Na konci sa vypíše informácia o tom, aký je počet zákazníkov v daných vekových skupinách.

3.3 Index a explain plan

Použitie indexu môže byť výhodné v prípade častého vyhľadávania v určitej tabuľke. Explain plan sme demonštrovali na `SELECT` dotaze ktorý vybral názov, umiestnenie sály a spočítal počet lekcí ktoré v jednotlivých sálach prebiehajú. Najprv sme spustili explain plan bez použitia indexu, teda tak ako databáza daný dotaz spracováva. Výstupom bolo:

| PLAN_TABLE_OUTPUT | | | | | | | |
|---|-----------------------------|---------|------|-------|-------------|----------|--|
| Plan hash value: 348358987 | | | | | | | |
| ----- | | | | | | | |
| Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time | |
| ----- | | | | | | | |
| 0 | SELECT STATEMENT | | 5 | 205 | 5 (40) | 00:00:01 | |
| 1 | SORT ORDER BY | | 5 | 205 | 5 (40) | 00:00:01 | |
| 2 | HASH GROUP BY | | 5 | 205 | 5 (40) | 00:00:01 | |
| 3 | NESTED LOOPS | | 5 | 205 | 3 (0) | 00:00:01 | |
| 4 | NESTED LOOPS | | 5 | 205 | 3 (0) | 00:00:01 | |
| 5 | TABLE ACCESS FULL | LEKCIA | 5 | 65 | 3 (0) | 00:00:01 | |
| * 6 | INDEX UNIQUE SCAN | SALA_PK | 1 | | 0 (0) | 00:00:01 | |
| 7 | TABLE ACCESS BY INDEX ROWID | SALA | 1 | 28 | 0 (0) | 00:00:01 | |
| ----- | | | | | | | |
| Predicate Information (identified by operation id): | | | | | | | |
| ----- | | | | | | | |
| 6 - access("LEKCIA"."SALA_FK"="SALA"."ID_SALY") | | | | | | | |
| Note | | | | | | | |
| ----- | | | | | | | |
| - dynamic statistics used: dynamic sampling (level=2) | | | | | | | |

Vo výpise je vidno že bolo potrebné pristúpiť ku všetkým údajom stĺpca v tabuľke `Lekcia` (riadok 5) bez použitia indexu. `NESTED LOOPS` znamená, že sa tabuľky spájajú naivne, čiže každý riadok z prvej tabuľky sa porovnáva so všetkými riadkami druhej tabuľky. `HASH GROUP BY` značí, že sa

zokupuje podľa hashovacieho kľúča. Taktiež sa použil INDEX UNIQUE SCAN ktorý pristupuje k tabuľkám cez B-strom a vracia jeden jedinečný riadok podľa primárneho kľúča SALA_PK.

Následne sme si zadefinovali index a opätovne spustili explain plan. Kedže naša databáza obsahuje málo riadkov, tak sa najprv index nepoužil lebo databáza usúdila že by to nebolo výhodné, napriek tomu sme mu zadali aby sa náš index použil. Dostali sme tento výstup:

| PLAN_TABLE_OUTPUT | | | | | | | |
|---|-------------------------------------|------------|------|-------|-------------|----------|--|
| Plan hash value: 4122391771 | | | | | | | |
| ----- | | | | | | | |
| Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time | |
| ----- | | | | | | | |
| 0 | SELECT STATEMENT | | 5 | 205 | 7 (29) | 00:00:01 | |
| 1 | SORT ORDER BY | | 5 | 205 | 7 (29) | 00:00:01 | |
| 2 | HASH GROUP BY | | 5 | 205 | 7 (29) | 00:00:01 | |
| * 3 | HASH JOIN | | 5 | 205 | 5 (0) | 00:00:01 | |
| 4 | TABLE ACCESS BY INDEX ROWID BATCHED | SALA | 5 | 140 | 2 (0) | 00:00:01 | |
| 5 | INDEX FULL SCAN | INDEX_SALA | 5 | | 1 (0) | 00:00:01 | |
| 6 | TABLE ACCESS FULL | LEKCIA | 5 | 65 | 3 (0) | 00:00:01 | |
| ----- | | | | | | | |
| Predicate Information (identified by operation id): | | | | | | | |
| ----- | | | | | | | |
| 3 - access("LEKCIA"."SALA_FK"="SALA"."ID_SALY") | | | | | | | |
| Note | | | | | | | |
| ----- | | | | | | | |
| - dynamic statistics used: dynamic sampling (level=2) | | | | | | | |

HASH JOIN znamená že záznamy spojovaných tabuliek sa spárovali cez hash kľúče spojenia, čiže sa spočítal hash pre stĺpec kľúča spojenia v každom riadku menšej tabuľky a potom sa prechádza väčšia tabuľka a pre každý jej riadok sa opäť spočíta hash kľúča spojenia, pomocou ktorého sa nájde odpovedajúci riadok prvej tabuľky ktorý má rovnaký hash. TABLE ACCESS BY INDEX ROWID BATCHED značí že sa do tabuľky pristupuje cez náš vytvorený index. Taktiež sa vykonal INDEX FULL SCAN s našim indexom, čo znamená že sa použil výpis hodnôt z indexovaných stĺpcov.

3.4 Prístupové práva

Prístupové práva sú v skripte nastavené tak aby simulovali činnosť inštruktora. Inštruktor má obmedzený prístup k niektorým tabuľkám. Má plne povolený prístup k tabuľkám Sala, Lekcia, Kurz ktoré sa týkajú jeho činnosti, čiže má možnosť meniť údaje o lekciách, vkladať nové lekcie a kurzy a taktiež meniť sálu v ktorej lekcia prebieha. Taktiež má práva na spúšťanie procedúr. Naopak z tabuliek Instruktor, Zakaznik má možnosť iba čítať údaje, ale nemá možnosť modifikácie údajov svojich kolegov ani údajov zákazníkov.

3.5 Materializovaný pohľad

V projekte sme implementovali materializovaný pohľad patriaci druhému členovi tímu, ktorý používa tabuľky nadefinované prvým členom. Pre materializovaný pohľad bol vytvorený materializované logy,

v ktorých sa uchovávajú zmeny v tabuľkách, aby bolo možné pri zmenách použiť `REFRESH FAST ON COMMIT` namiesto `COMPLETE REFRESH` kedy by sa celý dotaz musel znova spúšťať čo by trvalo dlhšie. Po vytvorení logu sme si vytvorili samotný materializovaný pohľad s týmito vlastnosťami:

- `CACHE` - optimalizácia čítania z pohľadu, často vyhľadávané dáta sa uložia do vyrovnávacej pamäte, kde sú potom ihneď dostupné
- `BUILD IMMEDIATE` - ihneď ako sa pohľad vytvorí tak sa aj naplní
- `ENABLE QUERY REWRITE` - použitie materializovaného pohľadu optimalizátorom

Následne po vytvorení samotného materializovaného pohľadu sme predviedli jeho funkčnosť, najprv sme si vypísali čo pohľad obsahoval, potom sme pridali nové údaje do tabuľky a príkazom `COMMIT` sme potvrdili zmeny a opäť sme si vypísali čo materializovaný pohľad obsahuje.