Vysoké učení technické v Brně
Fakulta informačných technológií

Databázové systémy
2017/2018

Projekt IDS – Dokumentace popisující finální schéma databáze

Správa ZOO

Igor Ignác(xignac00) a Filip Kočica(xkocic01) Brno 27.4.2018

1. Zadání

Navrhněte jednoduchý informační systém pro ZOO, v které pracují vyškolení ošetřovatelé, kteří se starají o zvířata v ZOO. Informační systém umožňuje plánovat krmení a čištění klecí/výběhů pro konkrétní zvířátka. Každý ošetřovatel je vyškolen pro krmení jednoho či více druhů zvěře a pro určité typy klecí/výběhů. Systém uchovává informace o zvířatech (datum narození, země původu, rodiče oblast výskytu ve volné přírodě atd.) i o samotných ošetřovatelích (jméno, příjmení, datum narození, vzdělání, titul, atd.). Systém umožňuje efektivní plánování čištění výběhu, kdy každý typ výběhu (vodní nádrž, klec, venkovní výběh atd.) má specifické nároky: potřebný čas, pomůcky, počet ošetřovatelů, atd. Systém uchovává informace, kdo čištění provedl, ale zároveň umožňuje kontrolu, zda jsou ošetřovatelé pro dané čištění vyškoleni. Stejně tak systém umožňuje kontrolu při plánování krmení zvěře, konkrétní ošetřovatel musí být kompetentní ke krmení zvířete a musí mít v danou dobu čas (nemůže krmit na více místech, nebo krmit a zároveň čistit klec na druhé straně ZOO). U krmení je zaznamenáno množství žrádla, čas krmení, ošetřovatel, zvíře. Při úmrtí zvířete se zaznamená datum úmrtí a další krmení se pro zvíře již neplánuje, ale záznam v systému zůstává.

2. Řešení

2.1 Realizace vztahu generalizace/specializace

V našom návrhu systému sme tento vzťah uplatnili pri entitnej množine Vedúci ZOO a Ošetrovateľ. Vedúci ZOO dedí všetky atribúty z entitnej množiny Ošetrovateľ a sám má ako ďalší atribút uvedenú prax. V databázovom systéme sme túto skutočnosť riešili vytvorením samostatnej tabuľky pre ošetrovateľa a vedúceho zoo. Tabuľka Vedúci ZOO obsahovala všetky atribúty tabuľky Ošetrovateľ(samozrejme okrem jeho primárneho kľúča) a navyše ešte požadovaný atribút prax.

2.2 Vytvoření alespoň dvou netriviálních uložených procedur

Prvá procedúra počíta percentuálne koľko času z celkovej doby čistenia výbehov bolo potrebné venovať čisteniu zadaného výbehu, na základe jeho typu. Na začiatku procedúry sa vytvorí požadovaný kurzor, ktorý odkazuje na spojenie tabuliek Vybeh a ZaznamCisteni. Následne sa deklarujú premenné pre celkovú dobu čistenia `celkova_doba_cisteni` , hľadanú dobu čistenia `hledana_doba_cisteni` a `meno_výbehu`, ktoré získame pomocou kurzoru ako `Vybeh.typ_vybehu%TYPE`. Následne, za podmienky, že sa v zázname čistenia nachádza rovnaké id výbehu ako v tabuľke Vybeh sa tento typ výbehu uloží premennej `meno_výbehu`. Toto spojenie nám zaručí, že v danom výbehu prebehlo čistenie aspoň raz. Po tejto inicializácii nasleduje cyklus, počíta dobu čistenia zadaného výbehu a taktiež celkovú dobu čistenia všetkých výbehov. Po prejdení všetkých položiek vypisuje pre zadaný typ výbehu jeho percentuálny podiel na celkovom čistení. Ak by jeho zastúpenie na čistení bolo nulové je vytvorená EXCEPTION, ktorá zachytáva tento stav a vypisuje príslušné hlásenie. Taktiež ak nastane iná chyba skript končí s chybou -20006 a chybovým hlásením.

Druhá procedúra počíta odchylku množstva krmiva pre zviera od bežnej priemernej hodnoty pre všetky zvieratá. Na začiatku je opäť vytvorení kurzor ktorý vytvára odkaz na spojenie tabuliek Zvire a ZaznamKrmeni. Nasleduje deklarácia premenných aj s požadovanou položkou odkazujúcou sa na riadok tabuľky, konkrétne `polozka_mnozstvi odchylka%ROWTYPE`. Cyklom prejdeme všetky záznamy kŕmenia a pri zhodnom id zvierata z tabuľky Zvire a id zvierata z tabuľky ZaznamKrmeni postupne sčítavame hodnotu pre hľadané množstvo kŕmenia. Taktiež uchovávame celkové množstvo krmiva. Po ukončení cyklu jednoduchou matematickou operáciou dostávame hľadanú odchylku. Rovnako ako v prvej procedúre ošetrujeme delenie nulou a prípadne nenájdenie Zvierata pomocou exceptions.

2.3 Vytvoření indexu spojeného s explain plan

Explain plan sme sa rozhodli použiť pre dotaz select, ktorý v jednotlivých typoch výbehov spočíta počet zvierat, ktoré tento typ obývajú. Následne sme si vytvorili index, ktorý odkazoval na položku typ výbehu z tabuľky výbeh a explain plan sme zopakovali s vytvoreným indexom.

Výsledok bol takýto :

```
| Id  | Operation                     | Name        | Rows  | Bytes | Cost (%CPU)| Time|
-------------------------------------------------------------------------------------------
|   0  | SELECT STATEMENT      |                 |     5 |   190 |     4  (25)| 00:00:01 |
|   1  |  HASH GROUP BY        |                 |     5 |   190 |     4  (25)| 00:00:01 |
|   2  |   NESTED LOOPS        |                 |     5 |   190 |      3   (0)| 00:00:01 |
|   3  |    NESTED LOOPS       |                 |     5 |   190 |      3   (0)| 00:00:01 |
|   4  |     TABLE ACCESS FULL   | ZVIRE       |     5 |    65 |      3   (0)| 00:00:01 |
|*  5  |     INDEX UNIQUE SCAN  | PK_VYBEH |     1 |       |      0   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
|   6 |    TABLE ACCESS BY INDEX ROWID| VYBEH    |     1 |    25 |     0   (0)| 00:00:01 |
```

Môžeme vidieť, že k stĺpcu tabuľky Vybeh bolo pristúpené pomocou indexu. Žiadne zrýchlenie však nenastalo ani s použitým indexom. Príčinou môže byť to, že aj bez použitia indexu sa daná operácia vykonala za 1 sekundu. Preto by index mal väčšie využitie pri väčšom objeme dát v jednotlivých tabuľkách.

2.4 Triggery

Triger na autoinkrementaci identifikačních čísel jsme použili u tabulky zvíře. První jsme vytvořili sekvenci IDcounter, jejíž položku nexval jsme při průchodu tabulkou přiřazovali jednotlivým primárním klíčům, tedy identifikačním číslům zvířete.

Další triger jsme zvolili jako kontrolní. Konkrétně pro kontrolu rodných čísel ošetřovatelů. Před úpravou /vložením rč do tabulky se se deklarují proměnné pro uložení jednotlivých položek rč, jež pomocí funkce SUBSTR získáme a uložíme do příslušných proměnných. Poté pomocí podmínek IF/END IF kontrolujeme délku rč, zda jsou čísla ve správném rozsahu (např. měsíc větší než 0 a zároveň menší než 13), dělitelnost 11 pomocí funkce MOD a zda obsahuje pouze čísla. Pokud je některá z podmínek porušena, vyhodí se vyjímka Raise_Application_Error s příslušnou chybovou hláškou a číslem chyby.

2.5 Práva

Provádíme pomocí příkazu GRANT na tabulky, poté následuje výčet práv (např. INSERT – právo vkládat záznamy, DELETE – parvo rušit záznamy, apod.), který jsme vybrali ALL, tedy všechna práva. Poté náslaedují jména tabulek/záznamů tabulek, pro které se aplikuje toto přidělení práv. Opět jsme vybrali všechny tabulky. V poslední řadě se vybírá, komu se tato práva přiřazují, tedy mému kolegovi xignac00.

2.6 Materializovaný pohled

Je to objekt, jež poskytuje data ve stejné podobě jako tabulka. Místo tabulky, která obsahuje přímo data, pohled obsahuje pouze předpis, jakým způsobem data získat z jiných tabulek/pohledů. Materializovaný pohled se na rozdíl od nematerializovaného někam uloží, proto je dotaz rychlejší. Náš materializovaný pohled jsme vytvořili na tabulku zvíře s tím, že vytváří data okamžitě a obnovuje je při commitu. Poté povolíme přepis obsahu pohledu, seřadíme zvířata podle počtu výskutů, pomocí INSERT vložíme nový řádek a provedeme commit. Pri dalsim SELECTu uvidíme změněná data v pohledu.

3. Závěr

Vývoj projektu probíhal po celou dobu na školním serveru pomocí aplikace SQL Developer. Části projektu byly relativně jednoduché a ne příliš časově náročné, ale i přesto jsme se naučili spoustu nových věcí a hlavně prakticky vyzkoušeli látku probíranou na přednáškách.

Jsme rádi, že jsme projekt dokončili poměrně brzy a vyhnuli se každoročnímu zatížení serveru těsně před odevzdáním.

4. Zdroje

[1] SQL Syntax: W3Schools [online]. [cit. 2018-04-28]. Dostupné z: https://www.w3schools.com/sql/sql_syntax.asp

[2] SQL: Wikipedia [online]. [cit. 2018-04-28]. Dostupné z: https://cs.wikipedia.org/wiki/SQL

[3] SQL docs: Oracle [online]. [cit. 2018-04-28]. Dostupné z: https://docs.oracle.com/cd/B19306_01/server.102/b14200/toc.htm