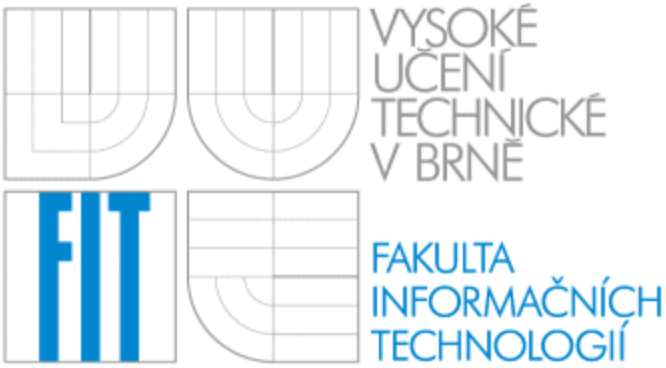


Databázové systémy
2014/2015

Projekt - SQL skript

Zadanie č.34 - Ordinace praktického lékaře

Filip Ježovica (xjezov01)
Marek Marušic (xmarus05)

Brno, 2. 5.2015

Zadanie

34. Ordinace praktického lékaře
V ordinaci praktického lékaře jsou ošetřováni objednaní pacienti i akutní případy cizích pacientů. Uvažujte ordinaci jednoho lékaře a jedné sestry. Systém umožní sestře objednávat pacienty na daný termín a případně i specifikovat jaké výkony jsou v plánu během tohoto termínu provést. Dále eviduje návštěvy pacienta (ať již byly předem naplánovány anebo šlo o akutní případ). U každé návštěvy je třeba také evidovat jaké výkony byly provedeny a jaké léky byly předepsány. Systém bude zvát pacienty na pravidelné prohlídky a očkování, které mají různou dobu účinnosti. Pacient je vždy pojištěncem jediné pojišťovny, předpokládejte, že nás nezajímají změny pojišťovny, které pacient v průběhu času provedl. Systém musí evidovat faktury pro pojišťovnu a umožnit lékaři je vytisknout. V průběhu jedné návštěvy mohl lékař zažádat o (jedno či více) vyšetření, na které musel pacient dojít na jiné pracoviště.

1. Mazanie a vytváranie tabuliek

Pri spustení skriptu je nutné na začiatku zmazať už vytvorené tabuľky, pre správne vytvorenie nových tabuliek a ostatných databázových objektov. Následne sa vytvoria prázdne tabuľky a sekvencie pre auto increment do našej databázy. Postupne sa do všetkých tabuliek pridávajú primárne kľúče a cudzie kľúče na základe vzťahov medzi danými tabuľkami.

2. Luhnov algoritmus

Pred vložením nového externého pracoviska sa spustí trigger, ktorý kontroluje správnosť vkladaného IČPE, či spĺňa luhnov algoritmus. Vstupné číslo sa prevedie na postupnosť znakov, a po jednom, sa znaky naspäť prevádzajú na čísla. Zároveň sa zisťuje či daný znak sa nachádza na párnej alebo nepárnej pozícii. Každé párne číslo sa vynásobí číslom 2 a následne priráta do dočasne vytvorenej premennej, ktorá udržiava tento súčet. Ak je výsledný číselný súčet deliteľný číslom 10, vkladané číslo spĺňa podmienku a môžeme vložiť záznam do tabuľky. V prípade že číslo nespĺňa dané podmienky, vypíše sa chyba č. -20203.

3. Automatická inkrementácia

Automatickú inkrementáciu sme použili pri vkladaní do tabuliek *Navsteva*, *Pacient*, *Liek*, *PredpisanyLiek*. Každá z týchto tabuliek má vytvorenú svoju vlastnú sekvenciu, ktorá sa používa v triggery pred vkladaním do danej tabuľky. Pri vkladaní do týchto tabuliek nie je potrebné vkladať primárny kľúč, pretože bude generovaný automaticky.

4. Generalizácia

Pre tabuľku výkon sme použili generalizáciu, kde bolo treba rozlíšiť či je daný výkon očkovanie, alebo vyšetrenie, pretože tabuľka očkovanie musí obsahovať dobu účinku a tabuľka vyšetrenie musí obsahovať časovú náročnosť.

5. Procedúry

V demonštrovaných procedúrach je použité zachytávanie chybových stavov pomocou exception, ďalej využitie explicitných i implicitných kurzorov.

5.1. *show_plan()*

Prvá vytvorená procedúra *show_plan()* má za úlohu vypísať plán ordinácie na zadaný počet dní. Tento plán vypisuje deň, pacienta a výkon, ktorý má prebehnúť počas návštevy ordinácie. Daná procedúra demonštruje použitie explicitného a implicitného kurzoru. Zároveň

je tu využitá funkcia *datum()* pre jednoduchšie získavanie rozsahu od aktuálneho dňa po požadovaný deň.

5.2. *stat_Poist()*

Ďalšia procedúra, zobrazuje štatistiku o tom koľko percent našich pacientov je zaregistrovaných u ktorej poisťovni. Znova je tu využitý implicitný kurzor (priamo v cykle for) v ktorom sa vypíše číslo zaokrúhlené na 2 desatinné čísla a názov poisťovne.

6. Vkladanie záznamov

Pre vkladanie záznamov sme použili jednoduché príkazy *INSERT*. Tak isto sme demonštrovali optimalizované vkladanie záznamov pomocou príkazu *INSERT ALL*, ktorý pošle záznamy v jednej dávke na server, čím by sa tento proces mal urýchliť, kedže sa vkladanie odľahčí od sieťovej komunikácie pri každom vložení jedného záznamu.

7. Explain plan

Pri využití explain plan sa daný dotaz nevykonáva, len je volaný optimalizátor. Výsledky tohoto plánu sú uložené do systémovej tabuľky, odkiaľ je ich možné vypísať s rôznou úrovňou podrobností (napríklad “basic”, “typical”, “all”). Optimalizátor si volí najjednoduchší plán, nie vždy ale môže byť najsprávnejší. Preto máme možnosť optimalizátor ovplivniť pomocou tzv. “hints”, kde môžme zvoliť index alebo metódu spojenia na vyžiadanie.

Pri využití explain plan pre určitý SELECT statement (bez využitia explicitného vytvorenia indexu) nám optimalizátor poskytne tieto výsledky plánovaného dotazu:

```
----------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                     |    24 | 1584  |     5  (40)| 00:00:01 |
|   1 |  SORT ORDER BY                 |                     |    24 | 1584  |     5  (40)| 00:00:01 |
|   2 |   HASH GROUP BY                |                     |    24 | 1584  |     5  (40)| 00:00:01 |
|   3 |    NESTED LOOPS                |                     |    24 | 1584  |     3  (00)| 00:00:01 |
|   4 |     NESTED LOOPS               |                     |    24 | 1584  |     3  (00)| 00:00:01 |
|   5 |      TABLE ACCESS FULL         | PREDPISANYLIEK      |    24 |  624  |     3  (00)| 00:00:01 |
|*  6 |      INDEX UNIQUE SCAN         | PK_LIEK             |     1 |       |     0  (00)| 00:00:01 |
|   7 |     TABLE ACCESS BY INDEX ROWID| LIEK                |     1 |   40  |     0  (00)| 00:00:01 |
----------------------------------------------------------------------------------------------------------------------------
```

Tabuľky sa začnú spájať vo vnorených cykloch, kde sa každý riadok prvej tabuľky porovnáva s každým riadkom druhej tabuľky. Pri spájaní sa používa priamy prístup k tabuľke *PredpisanyLiek* bez zmysluplne použiteľného indexu (*TABLE ACCESS FULL*). K tabuľke *Liek* sa pristupuje cez binárny strom, kde sa použije *INDEX UNIQUE SCAN*. V tomto kroku (6) sa spájajú tabuľkuy podľa definovaného *JOIN ON*. Následne po úspešnom spojení sa dohľadá *Nazov* z tabuľky *Liek* po prechode indexom.

Pri využití explain plan pre rovnaký určitý SELECT statement s využitím explicitného vytvorenia indexu, ktorý sme sa rozhodli vytvoriť na tabuľku s väčším množstvom dát (*PredpisanyLiek*) za účelom zníženia prístupov na disk , nám optimalizátor poskytne tieto výsledky plánovaného dotazu:

```
-----------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name                | Rows  | Bytes | Cost      | Time       |
-----------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                     |    24 |  1584 |    4  (50)| 00:00:01 |
|   1 |  SORT ORDER BY                 |                     |    24 |  1584 |    4  (50)| 00:00:01 |
|   2 |   HASH GROUP BY                |                     |    24 |  1584 |    4  (50)| 00:00:01 |
|   3 |    NESTED LOOPS                |                     |    24 |  1584 |    2  (00)| 00:00:01 |
|   4 |     NESTED LOOPS               |                     |    24 |  1584 |    2  (00)| 00:00:01 |
|   5 |      VIEW                      | index$_join$_001    |    24 |   624 |    2  (00)| 00:00:01 |
|*  6 |       HASH JOIN                |                     |       |       |           |          |
|   7 |        INDEX FAST FULL SCAN    | PK_PREDPISANYLIEK   |    24 |   624 |    1  (00)| 00:00:01 |
|   8 |        INDEX FAST FULL SCAN    | INDEX_LIEK          |    24 |   624 |    1  (00)| 00:00:01 |
|*  9 |      INDEX UNIQUE SCAN         | PK_LIEK             |     1 |       |    0  (00)| 00:00:01 |
|  10 |    TABLE ACCESS BY INDEX ROWID | LIEK                |     1 |    40 |    0  (00)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------------------------------------
```

Tabuľky sa začnú spájať vo vnorených cykloch podobne ako pri predchádzajúcom pláne. Rozdiel je vo vnorenom cykle, kde sa spájajú tabuľky. Spájanie prebieha cez hash spojenia. Vytvorí sa hash pre stĺpce spojenia menšej tabuľky a väčšej tabuľky a ďalej sa spájajú na základe rovnakého hash kľúča (*HASH JOIN*). Pre výber dát z týchto tabuliek je využitý výpis hodnôt z predom vytvoreného indexu (*INDEX FAST FULL SCAN*). Zvyšná časť plánu sa vykoná rovnako ako v predchádzajúcom pláne.

8. Materializovaný pohľad

Uchováva informácie o obľúbenosti poisťovne. Obsahuje názov poisťovne a ku každej, číslo koľko pacientov ju používa. Pohľad je zoradený od najobľúbenejšej poisťovne. Najskôr je potrebné vytvoriť log pre obe využité tabuľky. Ďalej pohľad vytvoríme, ten sa naplní ihneď po jeho vytvorení kvôli použitému príkazu *BUILD IMMEDIATE*. Tak isto sme použili príkaz *REFRESH FAST ON COMMIT,* ktorý zabezpečí že sa nám pohľad aktualizuje podľa logov priradených tabuliek ihneď po zavolaní *COMMIT*. Následne sme použili príkaz *ENABLE QUERY REWRITE*, ktorý povolí optimalizátoru použiť materializovaný pohľad.

9. Pridelenie práv

Bolo možné prideliť rôzne práva (*ALL*, *INSERT*, *SELECT*, *UPDATE*, *EXECUTE,* …) k určitému objektu databázy. Tieto práva sme prideľovali druhému členovi týmu - ako “sestričke v ambulancii”. Práva k objektom ako triggery pre automatickú inkrementáciu pri vkladaní záznamov alebo trigger na overovanie *IČPE* neboli pridelené, pretože pre sestričku nie sú potrebné a manipulovať snimi môže iba administrátor.

Sestrička môže vložiť, upraviť, zobraziť zoznam pacientov a návštev v ordinácii ale zadané dáta nemôže mazať. Ďalej môže spustiť metódu *stat_Poist()*, s ktorou zobrazí štatistiku zastúpenia poisťovní u našich pacientov v percentách. Lieky môže predpísať iba doktor, sestričke je umožnené lieky iba prehliadať, takisto i odporučenie na externé pracovisko. Faktúry môže vložiť alebo zobraziť. Nie je ich možné upravovať ani mazať. Sestrička má i právo spúšťať metódu ako *show_plan()*, s ktorou si zobrazí plán ordinácie na určitý počet dní. Ďalej má právo na zobrazenie materializovaného pohľadu obľúbenosti poisťovní u pacientov. Ako posledné dopĺňa poskytnuté informácie o liekoch, extrených pracoviskách, poisťovniach , popisu výkonov,... Na tieto objekty má plné práva.

10. Databázové transakcie

Rozhodli sme sa demonštrovať atomicitu transakcií pri súbežnom prístupe viac spojení k jedným dátam. Pred vykonaním transakcií, sme vypli automatický commit aby sa transakcie neukončovali automaticky. Transakcia začína prvým SQL príkazom alebo dotazom a potom končí príkazom *COMMIT* alebo *ROLLBACK*. COMMIT pri správnom ukončení, ROLLBACK zasa naopak - pri neúspešnom. Je možné použiť *SAVEPOINT*, ktorý vytvorí miesto v transakcii, na ktoré je možnosť sa vrátiť.

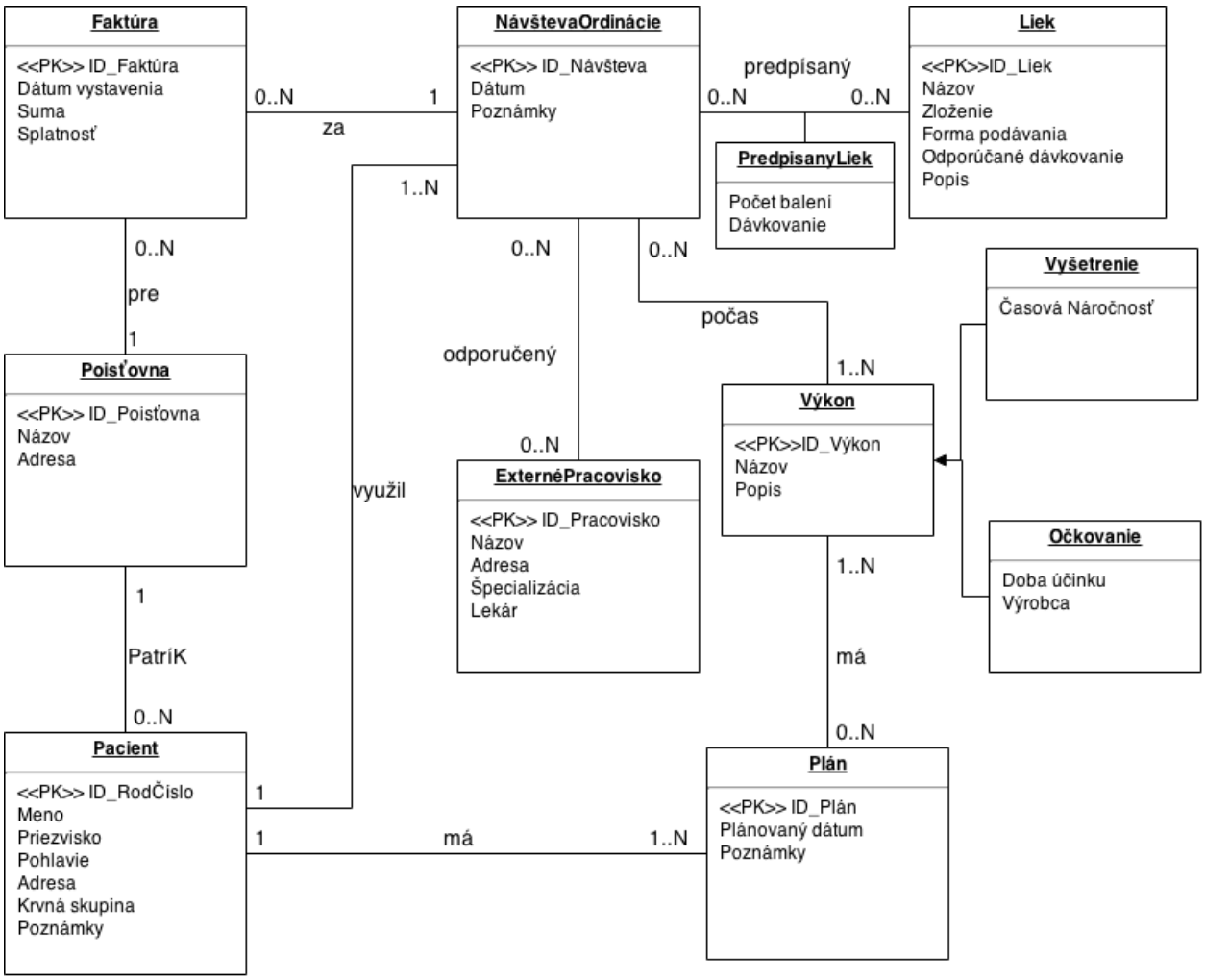
Pre príklad si môžme predstaviť tri oddelené spojenia pre tri transakcie. Prvá transakcia začne vlastniť zámok, ktorým zamkne prístup k práve upravovaným objektom. Zámok vlastní až do skončenia transakcie (*COMMIT/ROLLBACK*). Medzitým sa môže začínať iná transakcia (2) z iného spojenia, ktorá je ihneď blokovaná. Po tomto zablokovaní môže nastať situácia, že transakcia 1 využije *ROLLBACK TO SAVEPOINT*, ktorý odvolá všetko po uloženom mieste vrátane zámku ale transakciu neukončí. Preto zostáva transakcia 2 stále blokovaná a čaká na už uvoľnený zámok preto napríklad transakcia 3 z ďalšieho spojenia blokovaná nie je a môže ju predbehnúť, získa zámok vykoná zmeny a ukončí sa - *COMMIT*. Pre uvoľnenie transakcie 2 je potrebné ukončiť transakciu 1. Potom je transakcia 2 uvoľnená získa zámok a môže uzamknúť upravovaný objekt a následne sa ukončiť.

Záver

Skript sme vyvíjali spoločne na portáli stypi.com, ktorý podporuje real-time online collaboration. Ďalej sme skript testovali na školskom servery *Oracle Database 12c* za využitia prostredia *Oracle SQL Developer* a *Oracle SQL*Plus*. Pre úspešné vypracovanie skrtiptu nám pomohli materiály dostupné k predmetu IDS, oficiálna dokumentácia na weboch Oracle a taktiež demonštračné cvičenia. Projekt sme chceli vytvoriť zaujímavejším, preto sme sa rozhodli i demonštrovať transakčné spracovania, ukážku zamykania. Demoštrovali sme i optimalizáciu použitím materializovaného pohľadu s prevedením dôkazu pomocou *EXPLAIN PLAN*. Ďalej sme vyskúšali prácu s funkciami - vytvorenie funkcie *datum()*, ktorá prijíma parameter, ktorým je možné sa posunúť na iný deň. Práca na tomto projekte nám priniesla nový pohľad do problematiky databáz, ktorý môže byť v budúcnosti prínosom.

Prílohy

1. Návrh databázy - ERD diagram



2. Finálne schéma databázy