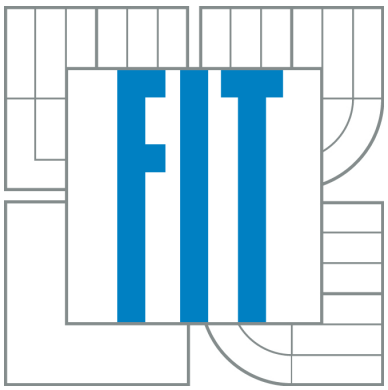
VYSOKÉ UČENIE TECHNICKÉ V BRNE

Fakulta informačných technológií



Databázové systémy

2014/2015

Konceptuálny model

Zadanie č. 41 – Zoologická záhrada

Filip Gulán (xgulan00)

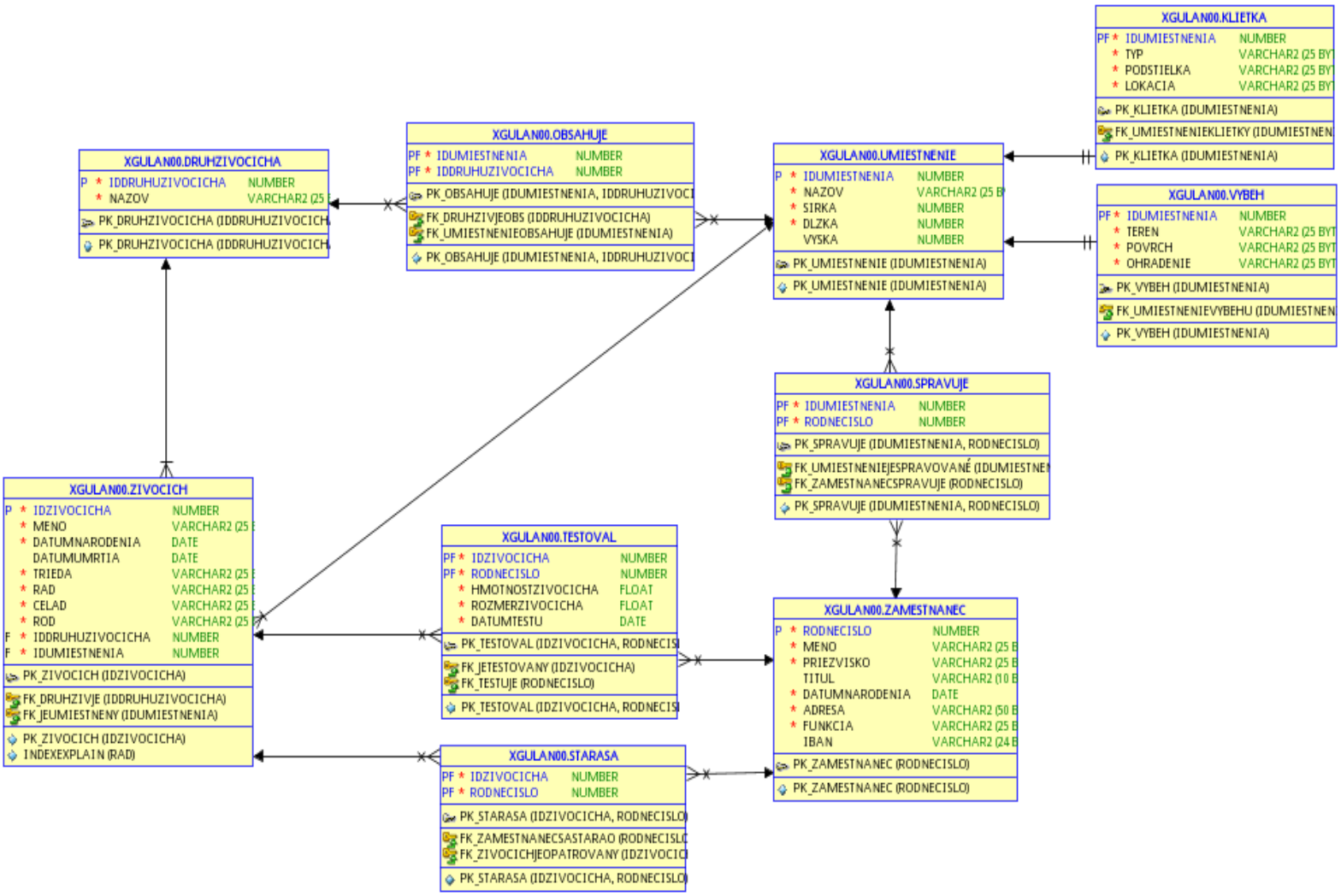
Eduard Rybár (xrybar04)

22.4.2015

1. Zadanie

Navrhnite informačný systém pre zoologickú záhradu. V zoologickej záhrade sú živočíchy umiestnené do klietok, výbehov alebo do klietok umiestnených v pavilónoch. Živočíchy sú delení podľa tried, radu, čeľade , rodu a druhu (napr. lama alpaka je v triede cicavcov, radu párnokopytníkov, v čeľadi ťavovitých, v rodu lama a druh alpaka). Pre zjednodušenie predpokladajme striktne hierarchické delenie živočíchov a to, že každý živočích je príslušníkom práve jedného druhu. Jeden druh živočícha môže byť v niekoľkých výbehoch alebo klietkach, a naopak, v jednom výbehu alebo klietke môže byť viac rôznych druhov. Systém musí byť schopný vyhľadávať živočíchy podľa ich príslušnosti do jednotlivých kategórií. Pre každého živočícha je treba uchovávať informácie o dátume narodenia (a prípadne úmrtia), meno, históriu výsledkov merania (hmotnosti, rozmerov, ...), apod.

2 Finálne schéma databáze



2.1 Generalizácia/Špecializácia

Generalizáciu/Špecializáciu sme v našej databázy využili pri tabuľke UMIESTNENIA, a to tak, že umiestnenia môžu byť typu buď klietka, alebo výbeh, avšak nie obidve naraz. Klietka aj výbeh obsahujú vo svojich tabuľkách rozdielne položky, teda sme nemohli do tabuľky umiestnenia pridať tieto položky.

3. Implementácia

Skript má najprv vytvoriť základné objekty a naplniť tabuľky ukážkovými hodnotami. Potom musí zadefinovať pokročilé obmedzenia a objekty. Následne skript povinne obsahuje ukážkové príklady manipulácie s dátami a požiadavkami demonštrujúce použitie vyššie spomínaných obmedzení a objektov tohoto skriptu. .

3.1 Triggery

Ako prvé sme implementovali databázové triggery. V našom prípade to bola automatická inkrementácia primárneho kľúča v tabuľke UMIESTNENIA. Tento trigger sme realizovali pomocou sekvencie, pre pamätanie si čísla posledného pridaného primárneho kľúča. Ďalším triggerom bolo overovanie správnosti zadaného IBAN čísla účtu v tabuľke ZAMESTNANEC. Tento trigger presnejšie kontroluje či sa v danom čísle účtu nachádzajú iba platné znaky, či je po vynásobení váhami daný účet deliteľný 11, či sa na začiatku nachádza platný identifikátor štátu a či je číslo presne 24 znakové. Obidva triggery sa spúšťajú vždy pred vkladaním dát do danej tabuľky (BEFORE INSERT ON).

3.2 Procedúry

Pri vytváraní dvoch ne-triviálnych procedúr sme podľa zadania museli využiť kurzor, aby sme boli schopný pracovať s viacerými riadkami databáze, v prípade, že nám ich daný dotaz vráti viacej ako jeden, ošetrenie výnimok a premennú s dátovým typom odkazujúcim sa na riadok či typ stĺpca tabuľky. V procedúre *percentZastup()* využívame: ošetrenie výnimky pri delení nulou, kde nevypisujeme chybu, ale iba vypíšeme, že sa daný živočích v danom umiestnení nachádza 0%. Argumenty tejto procedúry sú id druhu živočícha, ktorého percentuálne zastúpenie počítame a id umiestnenia, v ktorom percentuálne zastúpenie počítame. Výstupom tejto procedúry je výpis percentuálneho vyjadrenia výskytu vybraného živočíšneho druhu vo vybranom umiestnení.

Ďalšou procedúrou bola *prostredZiv()*. Zmyslom tejto procedúry je po zadaní mena nejakého zvieraťa, vypísať plochu umiestnenia v metroch štvorcových, v ktorom sa dané

zviera nachádza a menný zoznam ďalších zvierat ktoré sa nachádzajú v tom istom umiestnení.

3.3 Explain plan a vytvorenie indexu

Explain plan sme demonštrovali na jednoduchom SELECT dotaze. Najprv sme spustili explain plan, alebo teda plán ako databáza spracováva daný select dotaz, na SELECTe bez použitia indexu, potom sme zadefinovali index a spustili explain plan znovu. Index sme vytvorili na tabuľke s viacerými dátami. Výstupom explain planov bolo:

Bez použitia indexu

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		6	318	4 (25)	00:00:01
1	HASH GROUP BY		6	318	4 (25)	00:00:01
2	NESTED LOOPS		6	318	3 (0)	00:00:01
3	NESTED LOOPS		6	318	3 (0)	00:00:01
4	TABLE ACCESS FULL	ZIVOCICH	6	162	3 (0)	00:00:01
5	INDEX UNIQUE SCAN	PK_UMIESTNENIE	1		0 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	UMIESTNENIE	1	26	0 (0)	00:00:01

S použitím indexu

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		6	318	3 (34)	00:00:01
1	HASH GROUP BY		6	318	3 (34)	00:00:01
2	NESTED LOOPS		6	318	2 (0)	00:00:01
3	NESTED LOOPS		6	318	2 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID BATCHED	ZIVOCICH	6	162	2 (0)	00:00:01
5	INDEX FULL SCAN	INDEXEXPLAIN	6		1 (0)	00:00:01
6	INDEX UNIQUE SCAN	PK_UMIESTNENIE	1		0 (0)	00:00:01

7	TABLE ACCESS BY INDEX ROWID	UMIESTNENIE	1	26	0 (0)	00:00:01

Databáza najprv nami definovaný index nepoužila, keďže usúdila, že pri takom malom počte riadkov, aký naša databáza obsahuje, by to bola príliš drahá operácia. Avšak my sme databáze povedali, že aby aj napriek tomu index použila. Ako môžeme vidieť, znížil sa cost, t.j. prístupy na disk, ale naopak CPU sa zvýšil. Rozoberme si ale samotný explain plan. SELECT STATEMENT bol uskutočnený dotaz select, HASH GROUP BY znamená, že sa zgrupuje podľa hashovacieho kľúča, ďalej tam sú 2X NESTED LOOPS, čo znamená, že sa tabuľky spájajú naivne, tak, že sa prechádza po jednej tabuľke a pre každý položku z prvej tabuľky sa prejdú všetky riadky z druhej tabuľky. Ďalej sa nám naše plány líšia, podľa toho, či sa použil nami definovaný index, alebo nie. Bez použitia indexu sa dotaz ďalej vykonával tak, že v tabuľke ZIVOCICH sa prechádzalo pomocou TABLE ACCESS FULL, kedy sa prechádza tabuľka celá od začiatku bez žiadnych indexov. Ďalej sa použil INDEX UNIQUE SCAN, ktorý pristupuje k tabuľkám cez B-strom, kedy nám vypadne jeden jedinečný riadok podľa primárneho kľúča PK_UMIESTNENIE. Na druhej strane, s použitím indexu sa vykonával TABLE ACCESS BY INDEX ROWID v tabuľke ZIVOCICH, kedy sa pristupuje do tabuľky cez konkrétny riadok (použil sa náš index). Ďalej sa vykonal INDEX FULL SCAN s našim indexom INDEXEXPLAIN, čo znamená, že sa použil index na výpis, bez toho aby sme sa museli pozerať do tabuľky. Posledné 2 kroky explain plan s indexom sú totožné s poslednými 2 krokmi z výpisu explain plan bez indexu.

3.4 Prístupové práva

Prístupové práva v úlohe simulujeme vytvorením práv pre zamestnanca. Tento zamestnanec má obmedzený prístup k niektorým tabuľkám. Ako príklad možno uviesť tabuľky ZAMESTNANEC, STARASA alebo SPRAVUJE, ku ktorý má zamestnanec prístup absolútne odoprený, pretože prirodzene by radový zamestnanec nemal právo rozhodovať o svojich kolegoch. Naopak, k ostatným tabuľkám, ktoré sa týkajú hlavne umiestnení a živočíchov, ktorý dané umiestnenia obývajú má prístup plne povolený a taktiež k procedúram *percentZastup()* a *prostredZiv()* má práva na spúšťanie týchto procedúr.

3.5 Materializovaný pohľad

Pri implementácii materializovaného pohľadu patriacemu druhému členovi týmu, ktorý používa tabuľky definované prvým členom týmu sme si najprv vytvorili materializované logy (kde sa uchovávajú zmeny hlavnej tabuľky), aby sa mohol pri zmenách tabuľky využívať fast refresh on commit, namiesto complete refresh, kedy by sa musel znovu spúšťať celý dotaz materializovaného pohľadu, čo by prirodzene trvalo dlhšie. Po vytvorení logov, sme vytvorili samotný materializovaný pohľad a predviedli jeho funkčnosť na príklade, kedy sme si najprv dali vypísať všetko čo materializovaný pohľad obsahoval, potom sme pridali dáta do tabuľky z ktorej sme materiliazovaný pohľad vytvorili, potvrdili zmeny COMMIT a nakoniec sme si znovu dali vypísať všetko čo materializovaný pohľad obsahoval. Materializovanému pohľadu sme nastavili okrem REFRESH FAST ON COMIT tieto vlastnosti: CATCHE (snaží sa optimalizovať, to čo sa načítalo z pohľadu), BUILD IMMEDIATE (hneď ako sa pohľad vytvorí, tak sa aj naplní), ENEBLE QUERY REWRITE (materializovaný pohľad bude používaný optimalizátorom, keď budeme chcieť vyhodnotiť dotaz, na ktorom je pohľad založený).

4. Záver

Skript sme riadne otestovali v prostredí Oracle na školskom servery Oracle12c. Skript sme tvorili v jednoduchom textovom editore Sumblime Text3 a následne testovali cez Oracle SQL developer. K úspešnému vypracovaniu tejto úlohy sme si naštudovali učebnú látku z projektu IDS a značne nám pomohli demonštračné cvičenia a oficiálna dokumentácia k Oracle SQL Developer. Ostatné informácie sme získavali z rôznych neoficiálnych internetových zdrojov, hlavne z StackOverflow.