VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Databázové systémy

Dokumentácia projektu

26. dubna 2020

Michal Koval(xkoval17)
Norbet Pócs(xpocsn00)

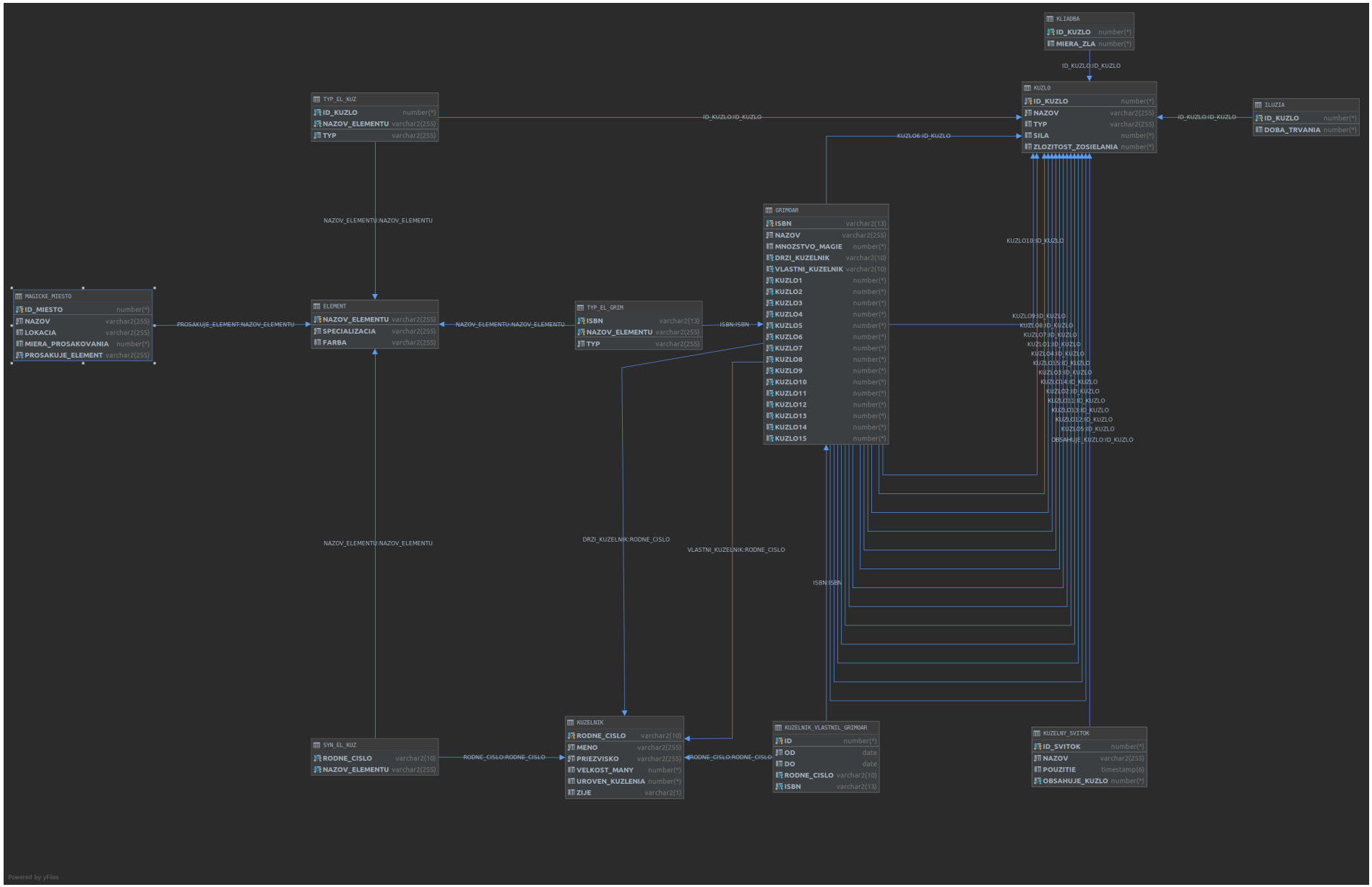
Obsah

1 Zadanie

47. Svět Magie

Kouzelnický svět vytváří informační systém pro evidence kouzel a kouzelníků. Magie v kouzelnickém světě je členěna podle elementů (např. voda, oheň, vzduch,.), které mají různé specializace (obrana, útok, podpora,.) a různé, ale pevně dané, barvy magie (např. ohnivá magie je pomerančově oranžová). Každé kouzlo má pak jeden hlavní element a může mít několik vedlejších elementů (např. voda a led), přičemž každý kouzelník má positivní synergii s určitými elementy. U kouzelníků rovněž evidujeme velikost many, jeho dosaženou úroveň v kouzlení (předpokládáme .klasickou stupnici. E, D, C, B, A, S, SS,.). U jednotlivých kouzel pak jejich úroveň složitosti seslání, typ (útočné, obranné) a sílu. Kouzla však nemohou být samovolně sesílána, pouze s využitím kouzelnických knih, tzv. grimoárů. Grimoáry v sobě seskupují více připravených kouzel a uchováváme veškerou historii jejich vlastnictví. Grimoáry mohou obsahovat kouzla různých elementů, nicméně jeden z elementů je pro ně primární, přičemž může obsahovat přibližně 10-15 kouzel. S postupem času však grimoáry ztrácejí nabitou magii (přibližně po měsíci ztratí veškerou magii) a je nutno je znovu dobít, ale pouze na dedikovaných místech, kde prosakuje magie (míra prosakování magie daného místa je evidována) určitých typů element (předpokládejte, že na daném místě prosakuje právě jeden typ). Toto nabití však nemusí být provedeno vlastníkem, ale I jiným kouzelníkem. V případě blížícího se vypršení magie grimoáru, systém zašle upozornění vlastníkovi. Alternativním způsobem sesílání magie je pak s využitím svítku, který obsahuje právě jedno kouzlo a po jeho použití se rozpadne.

2 Schéma databáze



3 Triggery

Implementovali sme 4 triggery:
Trigger `magicke_miesto_auto_pk` slúži na automaticke vypĺňanie primárneho kľúča položiek v tabuľke `magicke_miesto`. Využiva pri tom sekvenciu `magicke_miesto_seq`, ktorá sa začína na 100 a inkrementuje sa štandardne po 1.
Trigger `rodne_cislo_check` kontroluje správny formát rodného čísla, ktoré slúži ako primárny kľúč v tabuľke `kuzelnik`. Kontrola je definovaná vo fukncii `is_rc`, ktorá je volaná z tela triggeru. Je kompatibilná s 10 aj 9 miestnym formátom rodného čísla. V prípade nevalidného rodného čísla zavolá RAISE_APPLICATION_ERROR s chybovým kódom -20100.
Trigger `got_grimoar` automatizuje pridávanie hodnôt do tabuľky `kuzelnik_vlastnil_grimoar`. Ak pri pridávaní do tabuľky `grimoar` atribút `vlastni_kuzelnik` nie je NULL, tak je do tabuľky `kuzelnik_vlastnil_grimoar` vložený nový riadok s daným kúzelníkom a grimoárom. Dátum začiatku vlastníctva je pri tom nastavený na CURRENT_DATE.
Trigger `kuzelnik_died` slúži na automatizáciu upratania dat v pripade smrti kúzelníka. Samotné príkazy na prevedenie upratania sú zadefinované v procedúre `clean_up_kuzelnik`.

4 Procedury

Procedúra `priemerne_presakovanie_elementu` vyráta a vypíše pomocou DBMS_OUTPUT.PUT_LINE priemerné presakovanie na magické miesto zadaného ako parameter typu element.nazov_elementu%TYPE. V procedúre su použité: kurzor, ošetrenie vynimiek a premenné s dátovým typom odkazujúcim sa na typ stĺpca tabuľky. Kurzor využívame na prechod v LOOPe všetkými mierami prosakovania magických miest s rovnakým elementom ako je zadaný argument. Keďže rátame priemer potrebujeme deliť a tu môže nastať delenie nulou. To odchytávame výnimkou tak, že vypíšeme s DBMS_OUTPUT.PUT_LINE, že element vôbec nepresakuje.
Procedúra `clean_up_kuzelnik` v tabuľke `kuzelnik_vlastnil_grimoar` nastaví koniec vlastníctva na aktuálny dátum. Zároveň v tabuľke `grimoar` zruší vlastníctvo a držanie grimoárov zosnulým kúzelníkom.

5 Explain plan a index

EXPLAIN PLAN vysvetľuje ako SRBD spracuje dotaz. Použili sme dotaz z 3. časti projektu, ktorý z databázy zistí, ktorý element je najsilnejší. Využívame v ňom JOIN a 2x GROUP BY.

5.1 Pred indexom

```
-------------------------------------------------------------------------------------------------
| Id  | Operation                              | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |               |     7 |   287 |     7  (43)| 00:00:01 |
|   1 |  SORT ORDER BY                         |               |     7 |   287 |     7  (43)| 00:00:01 |
|   2 |   HASH GROUP BY                        |               |     7 |   287 |     7  (43)| 00:00:01 |
|   3 |    MERGE JOIN                          |               |    15 |   615 |     5  (20)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID        | KUZLO         |    15 |    90 |     2   (0)| 00:00:01 |
|   5 |      INDEX FULL SCAN                   | SYS_C001230237 |    15 |       |     1   (0)| 00:00:01 |
|*  6 |     SORT JOIN                          |               |    15 |   525 |     3  (34)| 00:00:01 |
|   7 |      VIEW                              | VW_GBF_5      |    15 |   525 |     2   (0)| 00:00:01 |
|   8 |       HASH GROUP BY                    |               |    15 |   240 |     2   (0)| 00:00:01 |
|*  9 |        TABLE ACCESS BY INDEX ROWID BATCHED| TYP_EL_KUZ    |    15 |   240 |     2   (0)| 00:00:01 |
|  10 |         INDEX FULL SCAN                | SYS_C001230293 |    19 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

Náhodne trvanie vykonania dotazu:

```
[2020-04-25 20:59:29] 7 rows retrieved starting from 1 in 255 ms (execution: 47 ms, fetching: 208 ms)
```

5.2 Index

Vytvorili sme index na tabuľku `typ_el_kuz` podľa atributu `typ` na základe toho, že server teraz musí prechádzať celú tabuľku a kontrolovať typ. Keďže v dotaze vyberáme len primárne kúzla, tým pádom vytvorením indexu podľa typu urýchlime prístup k primárnym typom a vyhneme sa sekvenčnému prechádzaniu celej tabuľky.

5.3 Po indexe

```
-------------------------------------------------------------------------------------------------------
| Id  | Operation                              | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |                |     7 |   154 |     7  (43)| 00:00:01 |
|   1 |  SORT ORDER BY                         |                |     7 |   154 |     7  (43)| 00:00:01 |
|   2 |   HASH GROUP BY                        |                |     7 |   154 |     7  (43)| 00:00:01 |
|   3 |    MERGE JOIN                          |                |    15 |   330 |     5  (20)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID        | KUZLO          |    15 |    90 |     2   (0)| 00:00:01 |
|   5 |      INDEX FULL SCAN                   | SYS_C001230237 |    15 |       |     1   (0)| 00:00:01 |
|*  6 |     SORT JOIN                          |                |    15 |   240 |     3  (34)| 00:00:01 |
|   7 |      TABLE ACCESS BY INDEX ROWID BATCHED| TYP_EL_KUZ     |    15 |   240 |     2   (0)| 00:00:01 |
|*  8 |       INDEX RANGE SCAN                 | TYP_TEK_IDX    |    15 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------
```

Náhodne trvanie vykonania dotazu:

```
[2020-04-25 20:59:53] 7 rows retrieved starting from 1 in 102 ms (execution: 30 ms, fetching: 72 ms)
```

PLAN po zavedení indexu potrvdzuje našu úvahu. Z plánu vidno, že INDEX FULL SCAN nahradil INDEX RANGE SCAN a aj sa vynechali dva kroky VIEW a HASH GROUP BY. Ušetrili sme 4 časové jednotky, čo sa môže zdať málo, ale doba vykonania dotazu sa pri testovaní preukázala výrazne kratšia. Dosiahli sme 3-10 násobné zrýchlenie vykonania dotazu.

6 Práva a materializovany pohľad

6.1 Práva

Prístupové práva k databázovým objektom boli všetky udelené príkazom GRANT ALL PRIVILEGES ON. Toto riešenie funguje na tabuľky ako aj na procedúry.

6.2 Materializovany pohľad

Jeho účelom je lokálne uloženie často využívaného pohľadu. Tým sa dosiahne rýchlejší prístup. Význam to má najmä ak sa jedná o často žiadaný dotaz.

Zvolili sme si dotaz z 3. časti riešenia projektu. Zobrazuje aké elementy prepúšťajú miesta v Londýne. Na vytvorenie materialized view musí mať používateľ už udelené práva k objektom užovateľa, z ktorých chce spraviť materializovny pohľad.

Použili sme následovné možnosti:
CACHE - optimalizácia čítania
BUILD IMMEDIATE - naplnenie pohľadu hneď po vytvoreni
REFRESH ON COMMIT - aktualizacia pohľadu po commite
ENABLE QUERY REWRITE - pohľad bude používaný optimalizátorom

V demonštrácii znázorňujeme, že pohľad sa aktualizuje až po prevedení príkazu COMMIT. Prvé dva selecty nezobrazujú ground element. Po commite sa už nachádza v pohľade.