Vysoké učení technické v Brně

Fakulta informačních technologií

Databázové systémy

2019/2020

Projekt IDS - Konceptuální model

Zadání č. 47 – Magický svět

Tomáš Dvořáček (xdvora3d)

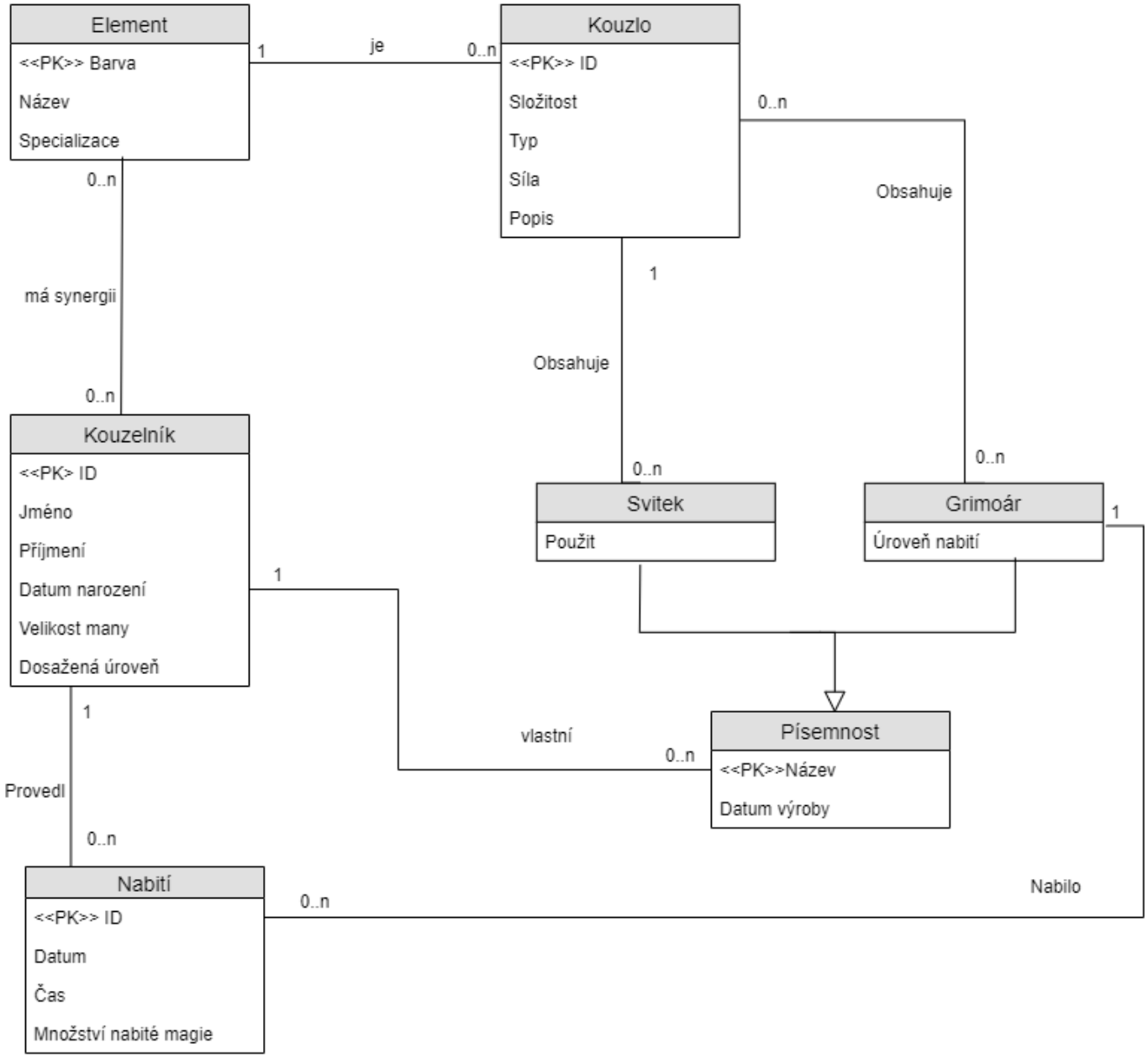
Michal Zobaník (xzoban01)

1. Zadání

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. Finální ER diagram



2.1 Generalizace

Jako generalizaci jsme zvolili tabulku Písemnost. Jako písemnost se počítá Grimoár a Svitek. Jejich hlavním rozdílem je to, že svitek může obsahovat jen jedno kouzlo a může být použit jen jednou, kdežto grimoár obsahuje více kouzel a také může být použit opakovaně.

3. Implementace

Skript nejprve vytvoří základní tabulky databáze, poté následuje část v podobě nastavení primárních a cizích klíčů. Následuje implementace triggerů, procedur a naplnění tabulek ukázkovými daty. Nakonec je vytvořen Explainplan, přidělení práv druhému členovi týmu a demonstrativní ukázka rozdílu mezi obyčejným a materializovaným pohledem.

3.1 Triggery

První z triggerů pojmenovaný *auto_id* slouží k automatickému přiřazení identifikátoru kouzelníkovi. První kouzelník bude mít identifikátor 0 a u dalších kouzelníků se tento identifikátor automaticky inkrementuje.

Druhý trigger *kontrola_urovne* kontroluje správnost vkládané úrovně kouzelníka. Pokud je vložená hodnota jiná než: SS, S, A, B, C, D, E je vyvolána výjimka a záznam o kouzelníkovy se do tabulky nepřidá.

3.2 Procedury

Úkolem procedury nazvané *procent_zastoupeni_kouzelniku* je výpočet procentuálního zastoupení kouzelníků zadané úrovně v synergii zadané barvy. Výpočet probíhá tak, že se do dvou kurzorů *data_1* a *data_2* uloží obsahy tabulek Synergie a Kouzelník.

Následně se v prvním cyklu počítá celkový počet kouzelníků, kteří jsou v synergii se zadanou barvou a zároveň se v dalším cyklu, který je vnořený do cyklu předchozího počítá kolik je kouzelníků, jež mají synergii se zadanou barvou a jejich úroveň zároveň odpovídá zadané úrovni.

Nakonec proběhne prostý matematický výpočet a je vypsán výsledek. Vzhledem k tomu, že procentuální počet kouzelníků může být nulový, bylo nutné ošetřit tento stav výjimkou ZERO_DIVIDE.

Procedura *prumerna_uroven* slouží k zjištění průměrné dosažené úrovně kouzelníků v databázi. Procedura využívá jeden kurzor, do kterého se uloží obsah tabulky Kouzelník. V cyklu se potom jednotlivé textové reprezentace úrovně převedou na číselnou reprezentaci, ze které se snáze vypočítá průměr. Spočítaný průměr se potom zaokrouhlí a zpátky převede na text reprezentující dosaženou úroveň.

3.3 Explain plan

Příkaz explain plan ukazuje jak se vykoná zadaný dotaz. V tomto případě je to dotaz na zjištění s kolika barvami má každý kouzelník synergii, seřazený podle jména kouzelníků.

```
-------------------------------------------------------------------------------------------
| Id  | Operation                      | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |               |     7 |   161 |     8  (38)| 00:00:01 |
|   1 |  SORT ORDER BY                 |               |     7 |   161 |     8  (38)| 00:00:01 |
|   2 |   HASH GROUP BY                |               |     7 |   161 |     8  (38)| 00:00:01 |
|   3 |    MERGE JOIN                  |               |     7 |   161 |     6  (17)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID| KOUZELNIK     |     4 |    80 |     2   (0)| 00:00:01 |
|   5 |      INDEX FULL SCAN           | PK_KOUZELNIKA |     4 |       |     1   (0)| 00:00:01 |
|*  6 |     SORT JOIN                  |               |     7 |    21 |     4  (25)| 00:00:01 |
|   7 |      TABLE ACCESS FULL         | SYNERGIE      |     7 |    21 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Výsledek explain plan ukazuje, že nejdřív dojde k operaci *TABLE ACCES FULL* tedy k úplnému přístupu do tabulky *Synergie* bez jakékoliv optimalizace a operaci *SORT JOIN*, která položky tabulky setřídí. *INDEX FULL SCAN*, postupně přečte celý index primárního klíče v tabulce *Kouzelník* a potom pomocí něj přistupuje k tabulce *Kouzelník*. Následuje operace *MERGE JOIN*, která spojí seřazené listy s daty z tabulek *Synergie* a *Kouzelník*. Operace *HASH GROUP BY* seskupí data pomocí hashovacího klíče. *SORT ORDER BY* položky seřadí (v našem případě podle jména Kouzelníka) a operace *SELECT STATEMENT* je potom samotný výsledek dotazu *SELECT*

Když vytvoříme nový index v tabulce *Synergie* pro sloupeček *ID_kouzelníka* a znovu zavoláme explain plan můžeme vidět, že dojte ke změně některých operací a snížení ceny dotazu.

```
-------------------------------------------------------------------------------------------
| Id  | Operation            | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |               |     7 |   161 |     5  (40)| 00:00:01 |
|   1 |  SORT ORDER BY       |               |     7 |   161 |     5  (40)| 00:00:01 |
|   2 |   HASH GROUP BY      |               |     7 |   161 |     5  (40)| 00:00:01 |
|   3 |    NESTED LOOPS      |               |     7 |   161 |     3   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL| KOUZELNIK     |     4 |    80 |     3   (0)| 00:00:01 |
|*  5 |     INDEX RANGE SCAN | IDX_SYNERGIE  |     2 |     6 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Z výstupu explain plan po vytvoření indexu je vidět, že se nejdříve provede *INDEX RANGE SCAN*, dojde tedy k použití definovaného indexu a přístupu k tabulce *Synergie* přes B-strom a *TABLE ACCES FULL*, kde se přistupuje k tabulce *Kouzelník* bez jakýchkoliv optimalizací. Následuje operace *NESTED LOOPS*, která znamená, že dojde ke spojení tabulek *Synergie* a *Kouzelník*, tak že se postupně prochází jedna tabulka a pro každý záznam se projde postupně druhá tabulka. Zbytek dotazu je stejný jako při nepoužití indexu.

3.4 Materializovaný pohled

Vytvořili jsme dva pohledy, obyčejný jménem *kniha_obsahuje* a materializovaný *kniha_obsahuje_m.* Následně jsme do tabulky, ze které oba pohledy vycházejí, přidali novou položku. Po provedení dotazu SELECT nad oběma pohledy jest vidno že v materializovaném pohledu na rozdíl od obyčejného žádná položka nepřibila. To je způsobeno tím, že materializovaný pohled vytvořil novou tabulku, kam se také nově přidaná položka uložila.

Pokud bychom chtěli, aby se položka v materializovaném pohledu promítla i do stávající tabulky, musíme nastavit refresh dat, např. příkazem REFRESH ON COMMIT. Obyčejný pohled žádnou novou tabulku nevytváří, takže každá provedená změna je po dotazu okamžitě vidět.