Slovenská Technická Univerzita v Bratislave
Fakulta Informatiky a Informačných Technológii

Databázové systémy
Druhé odovzdanie

1.5.2016
Branislav Pecher

1 Zadanie

Vo vami zvolenom prostredí vytvorte databázovú aplikáciu, **ktorá komplexne rieši minimálne 6 scenárov** vo vami zvolenej doméne. Presný rozsah a konkretizáciu scenárov si dohodnete s Vašim cvičiacim na cvičení. Aplikáciu vytvoríte v dvoch iteráciach. V prvej iterácii, postavenej nad relačnou databázou, musí aplikácia realizovať tieto všeobecné scenáre:

- Vytvorenie nového záznamu,
- Aktualizácia existujúceho záznamu,
- Vymazanie záznamu,
- Zobrazenie prehľadu viacerých záznamov (spolu vybranou základnou štatistikou),
- Zobrazenie konkrétneho záznamu,
- Filtrovanie záznamov spĺňajúcich určité kritériá zadané používateľom.

Aplikácia môže mať konzolové alebo grafické rozhranie. Je dôležité aby scenáre boli realizované realisticky - teda aby aplikácia (a teda aj jej používateľské rozhranie) naozaj poskytovala časť funkcionality tak, ako by ju očakával zákazník v danej doméne.

Scenáre, ktoré menia dáta musia byť realizované **s použitím transakcií** a aspoň jeden z nich musí zahŕňať **prácu s viacerými tabuľkami** (typicky vytvorenie záznamu a naviazanie cudzieho kľúča).

V druhej iterácii do aplikácie pridáte min. 1 scenár postavený na nerelačnej databáze Redis alebo Elasticsearch (dohoda s cvičiacim na inom type nerelačnej databázy je samozrejme možná). Konkrétny scenár si dohodnete s vašim cvičiacim v závislosti od použitej databázy a domény vašej aplikácie (napr. štatistiky o interakciách s jednotlivými záznamami aplikácie v Redise alebo vyhľadávanie záznamov cez Elasticsearch).

2 Stručný popis implementácie

Tento projekte je súčasne aj projektom na predmet Výskumne orientovaný seminár (VOS) a aj projektom na Databázové systémy (DBS), preto som zvolil ako programovacie prostredie ruby s frameworkom rails. Z tohto dôvodu sa jedná o webovú aplikáciu. Využívam relačnú databázu postgresql, pričom na prístup do nej používam z väčšej časti funkcie poskytované frameworkom Rails avšak aj zopár vlastných čistých SQL cez funkciu find_by_sql. Zároveň využívam aj NoSQL databázu Redis na realizovanie scenárov pre druhé odovzdanie. Táto NoSQL databáza je využitá na cachovanie jednotlivých udalostí aby sa zrýchlila činnosť v prípade, že je ich veľké množstvo. Zároveň sa aj využíva na sledovanie počtu unikátnych používateľov, ktorí navštívili jednotlivé udalosti.

3 Realizovanie jednotlivých SQL scenárov

Keďže tento dokument sa týka odovzdania prvej verzie pre predmet Databázové systémy uvádzam tu aj časti projektu, ktoré realizujú požadované scenáre.

Vytvorenie nového záznamu: Tento scenár je realizovaný hneď na dvoch miestach. Ako prvé umožňuje aplikácia registráciu nového používateľa do systému spolu s jeho autentifikáciou. Zároveň každý prihlásený používateľ je schopný si vytvoriť ľubovoľný počet udalostí.

Aktualizácia existujúceho záznamu: Tento scenár je implementovaný v rámci toho, že každý prihlásený používateľ je schopný si upraviť svoje zadané údaje (meno, email, heslo).

Vymazanie záznamu: Tento scenár je rovnako ako vytvorenie realizovaný hneď na dvoch miestach. Aplikácia obsahuje jedného používateľa, ktorý slúži ako admin pre ňu, ktorý vie zmazať každého vytvoreného používateľa okrem seba. Zároveň aj každý prihlásený používateľ vie zmazať sám seba v svojom profile ak sa práve nejedná o admina. Taktiež vie každý používateľ zmazať udalosť ktorú vytvoril.

Zobrazenie prehľadu viacerých záznamov: Tento scenár je realizovaný v rámci toho, že pri každom používateľovi je zobrazený zoznam jeho vytvorených udalostí. Zároveň je možné si pozrieť prehľad všetkých vytvorených používateľov v aplikácii.

Zobrazenie konkrétneho záznamu: Tento scenár je realizovaný v rámci toho, že je možné si zobraziť konkrétneho používateľa, či konkrétnu udalosť kde sú zobrazené jednotlivé informácie.

Filtrovanie záznamov: Tento záznam je momentálne realizovaný v rámci toho, že pri prehľade používateľov sa filtrujú iba tí, ktorí už majú aktivovaný účet. Avšak v budúcnosti bude pridané vyhľadávanie udalostí podľa mena udalosti, autora alebo predmetu.

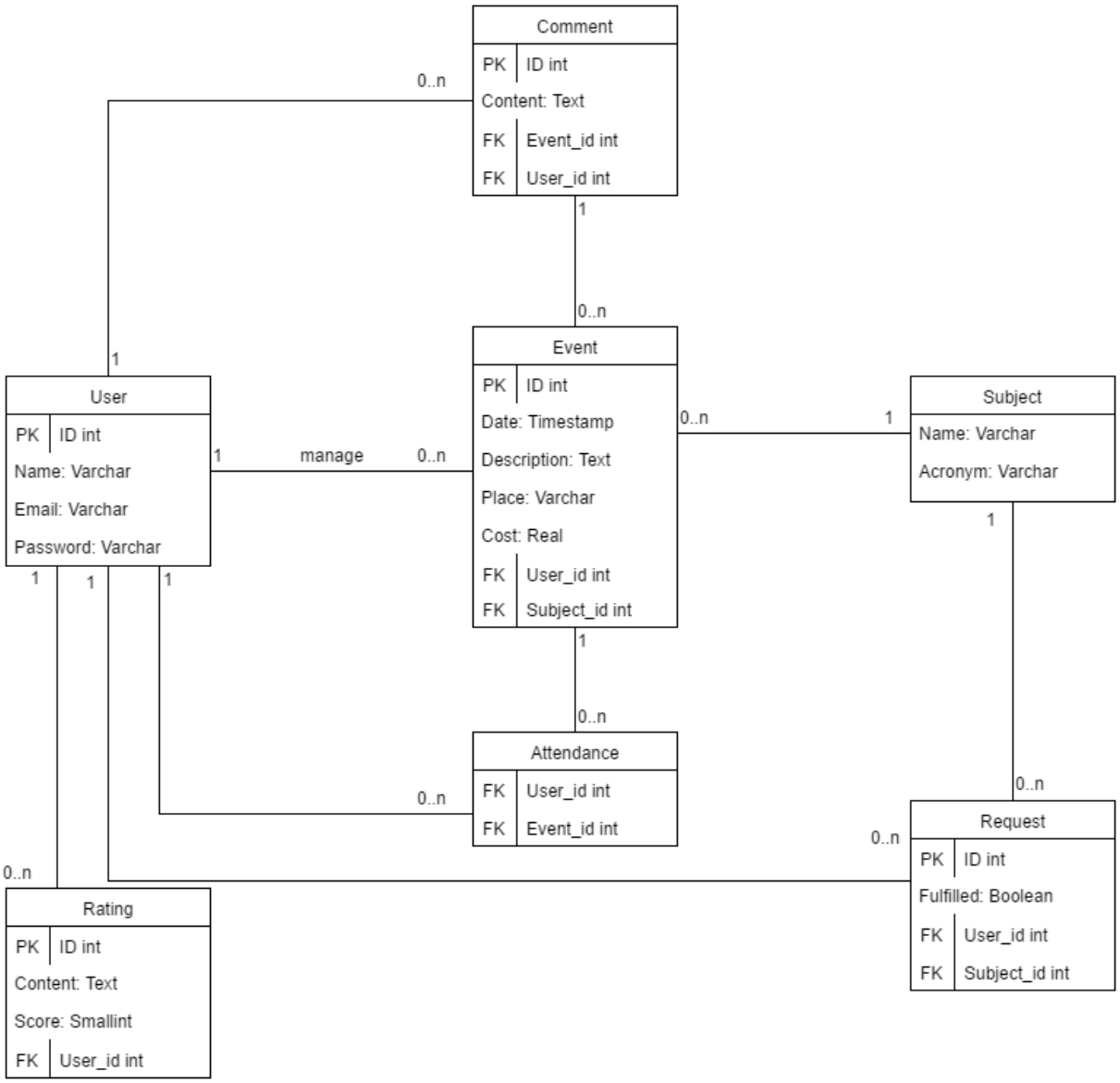
Transakcia: Keďže som zatiaľ v aplikácií nenašiel miesto v ktorom by bolo potrebné naraz vložiť do databázy 2 na seba závislé záznamy, je transakcia implementovaná iba nad vytváraním jednotlivých udalostí, čiže iba nad jedným insertom do databázy.

JOIN, GROUP BY, agregačná funkcia: Tieto 3 prípady uvádzam spolu preto, lebo sú realizované jedným selectom v mojej aplikácii. Jedná sa o nasledujúci select v users_controller:

```
User.find_by_sql("SELECT u.id, u.name, email, COUNT(e.id) AS count FROM users u LEFT JOIN events e ON u.id = e.user_id WHERE activated = 't' GROUP BY u.name, u.id, email ORDER BY u.id")
```

Ďalšie selecty sa nachádzajú aj v ostatných triedach typu Controller, avšak nie všetky využívajú čisté SQL, ale používajú funkcie poskytované používaným frameworkom RubyOnRails.

4 Diagram fyzického model spolu s vysvetlením



Na implementáciu môjho projektu som zvolil nasledujúci fyzický model. Hlavnými časťami je používateľ (User) a udalosť (Event). Každý jeden user môže vytvoriť ľubovoľný počet eventov, ktoré obsahujú aj informácie o tomto evente – čas konania, popis, z akého predmetu je organizovaný, miesto a cenu za osobu na jednu hodinu. Na každý takýto event sa môže prihlásiť ľubovoľný (v budúcnosti možno aj obmedzený) počet userov, pričom každý user môže byť zahlásený aj na viacerých eventoch. Zároveň každý event môže mať určitý počet komentárov (Comment) k nemu od usera, pričom user zase môže mať ľubovoľný počet komentárov. Zároveň aj každý user má určité ohodnotenie ostatnými pričom sa však neeviduje od koho bolo toto ohodnotenie pridané (Rating). Keďže práca ešte nie je finálna a nie je potreba využitia všetkých tabuliek, v projekte sa nachádzajú iba niektoré.

5 Realizovanie NoSQL scenárov

5.1 Cache-ovanie

V programe sa cache-ujú jednotlivé udalosti, ktoré vytvárajú používatelia. Pri otvorení prehľadu jednotlivých udalostí sa najprv zistí či je v cache niečo uložené. Ak je niečo uložené tak sa to iba vytiahne z Redis databázy a to sa použije. Ak tam ešte nič nebolo uložené tak sa vyberú všetky udalosti z postgresql databázy, prevedú sa na JSON a v takom formáte sa uložia do Redis databázy. Zároveň sa táto cache premazáva každý deň a taktiež vždy pri vytvorení a zmazaní novej udalosti.

5.2 Unikátne návštevy

Pri každom otvorení určitej udalosti sa zistí IP adresa z ktorej požiadavka prišla a taktiež aj ID používateľa, ktorý túto požiadavku odoslal (ak používateľ nie je prihlásený tak nie je možné otvoriť žiadnu udalosť). Táto IP adresa a ID sa skombinujú a uložia sa do Redis databázy do množiny, pričom ako meno premennej slúži ID udalosti, ktorá je zobrazovaná. Následne sa zistí kardinalita tejto množiny, aby bolo možné zobraziť tieto unikátne návštevy.

6 Záver

Keďže projekt je súčasne vyvíjaný aj na predmet Výskumne orientovaný seminár, ešte nie je vo finálnom štádiu a teda jeho grafické rozhranie je vo veľmi zlom stave. K projektu je zároveň pripojená aj táto dokumentácia, dump z databázy Redis(dump.rdb) a SQL dump postgresql databázy (psql_dump.sql). Dá sa však využiť aj možnosť seedovania, ktorú poskytuje framework Rails.