

Slovenská Technická Univerzita v Bratislave
Fakulta Informatiky a Informačných Technológií

Databázové systémy
Prvé odovzdanie

8.4.2016
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 Vaším 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).

2 Stručný popis implementácie

Tento projekt 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`.

3 Realizovanie jednotlivých 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 používateli 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.

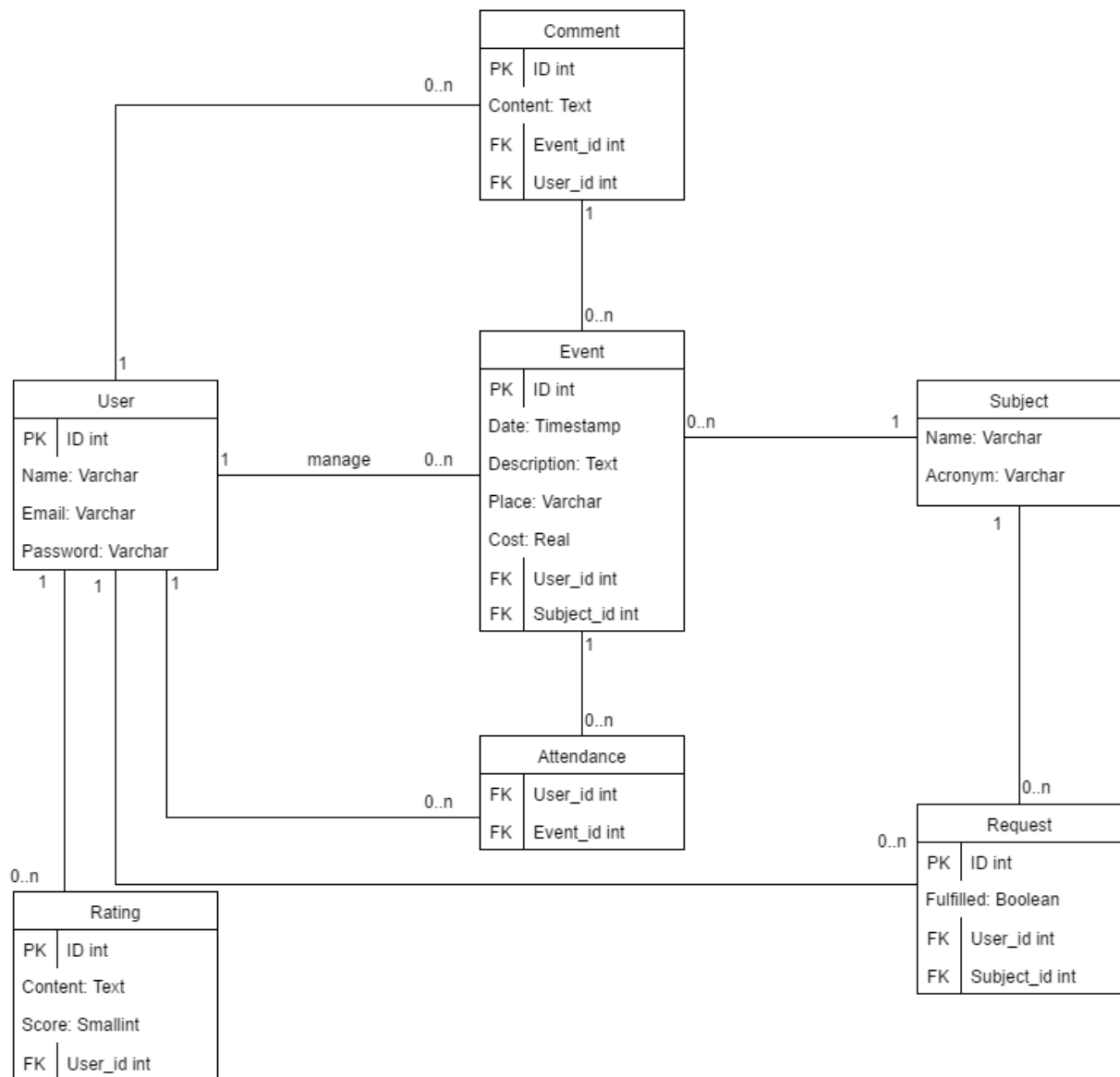
Transakcia: Keďže som zatiaľ v aplikácii nenašiel miesto v ktorom by bolo potrebné naraz vložiť do databázy 2 na seba závislé záznamy, je transakcia implementovaná iba nad vytváraním jednotlivých udalostí, čiže iba nad jedným insertom do databázy.

JOIN, GROUP BY, agregáčná 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 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 a SQL dump z mojej použitej databázy. Dá sa však využiť možnosť seedovania, ktorú poskytuje framework Rails.