Elektrotehnicki fakultet Sarajevo
Univerzitet u Sarajevu

Analiza SQL transakcija i upi u ETF Online Competition System aplikaciji

BAZE PODATAKA 2017/2018

Studenti:

Adnan Alibegovic
Anisa Hadzibulic
Edin Ceric

SQL Transakcije

1. Snimanje studentovih odgovora na pitanja testa

Polaganje testa je glavna i jedina kompleksnija funkcionalnost aplikacije, te ce zbog toga biti jedina opisana, dok su sve druge sporedne i ne zahtijevaju posebnu teoretsku obradu. U nastavku ce biti dat i opisan listing svih SQL iskaza unutar transakcije koja se izvrsava prilikom snimanja jednog polaganja u bazu. Ova funkcionalnost je odlican primjer gdje je neophodno koristenje transakcije u bazama podataka. Koristenje transakcije ce zakljucati tabele s kojima se radi unutar bloka naredbi, te tako osigurati da se ne dese citanja fantomskih redova, kao i da se radnja ne izvrsi do kraja. Dakle, ili ce sve naredbe unutar bloka biti uspjesno izvrsene ili nece nijedna.

Pocetak transakcije se oznacava naredbom **BEGIN**.

```
BEGIN;
```

Zatim slijedi niz INSERT i UPDATE iskaza koji ubacuju i azuriraju nove redove u tabele SCHEDULEDTESTRESULT, ANSWER i ANSWER_PREDEFINED_ANSWERS. Prvo je potrebno napraviti prazan TestResult kako bi se kasnije mogli referencirati rekordi u ostalim tabelama pri unosenju odgovora. SELECT iskazi su izostavljeni jer je njihova uloga iskljucivo namijenjena dobavljanju potrebnih ID-eva.

```
INSERT INTO "api_scheduledtestresult" ("comment",
                                       "scheduled_test_id",
                                       "student_id",
                                       "reviewer_id")
VALUES (NULL, 4, 4, NULL);
-----------------------------------------------------
INSERT INTO "api_answer" ("text",
                          "comment",
                          "points",
                          "question_id",
                          "scheduled_test_result_id")
VALUES ('Amdhal is great :D',
        NULL,
        NULL,
        4,
        10);
-----------------------------------------------------
INSERT INTO "api_answer" ("text", "comment", "points", "question_id",
                          "scheduled_test_result_id")
VALUES ('',NULL,NULL,5,10);
-----------------------------------------------------
INSERT INTO "api_answer_predefined_answers" ("answer_id",
                                             "predefinedanswer_id")
SELECT 25, 5;
-----------------------------------------------------
UPDATE "api_answer"
```

```
SET "text" = '',
  "comment" = NULL,
  "points" = NULL,
  "question_id" = 5,
  "scheduled_test_result_id" = 10
WHERE "api_answer"."id" = 25;
--------------------------------------------------
INSERT INTO "api_answer" ("text",
                          "comment",
                          "points",
                          "question_id",
                          "scheduled_test_result_id")
VALUES ('I don''t know',
        NULL,
        NULL,
        6,
        10);
--------------------------------------------------
UPDATE "api_scheduledtestresult"
SET "comment" = NULL,
  "scheduled_test_id" = 4,
  "student_id" = 4,
  "reviewer_id" = NULL
WHERE "api_scheduledtestresult"."id" = 10;
--------------------------------------------------
```

Optimizacija upita

Zbog lakseg objasnjenja, uzmimo isti primjer funkcionalnosti kao i u prethodnom dijelu. Dakle, pri snimanju odgovora na test, potrebno je dobaviti IDeve iz vise tabela kako bi se mogli ispravno referenciti na njih, novi rekordi vezani za odgovore. Django ORM koji se koristi u projektu pri svakom pristupanju polju objekta, koje referencira drugi entitet iz baze podataka, ce napraviti novi upit na bazu podataka. Tako da bismo dobili ID TestSetupa za odredjeni ScheduledTest, Django izgenerise sljedeca 2 upita:

```
SELECT "api_scheduledtest"."id",
FROM   "api_scheduledtest"
WHERE  "api_scheduledtest"."id" = 3;

SELECT "api_testsetup"."id",
FROM   "api_testsetup"
WHERE  "api_testsetup"."id" IN (1);
```

Lako se uocava da se upiti mogu napisati spojiti u jedan, te je to moguce postici koristeci Django ORM metode za povecanje performansi izvrsavanja SQL upita. Nakon koristenja **`select_related`** funkcije, Django generise sljedeci upit:

```
SELECT "api_scheduledtest"."id",
       "api_testsetup"."id",
FROM   "api_scheduledtest"
       INNER JOIN "api_testsetup"
               ON ( "api_scheduledtest"."test_setup_id" = "api_testset
up"."id" )
WHERE  "api_scheduledtest"."id" = 3;
```