TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Robert Sepp

Automaattestide loomine andmebaasile Edu

Bakalaureusetöö (9 EAP)

Juhendaja: Vambola Leping, MSc

Tartu 2018

Automaattestide loomine andmebaasile Edu

Lühikokkuvõte:

Töös on kirjeldatud automaattestide loomist Tartu Ülikoolis õpetatavale ainele Andmebaasid. Selgitatakse loodavate automaattestide vajalikkust ning koostatakse automaattestid ning neid käitav programm.

Võtmesõnad:

Andmebaas, automaattestimine

CERCS: P175 Informatics, systems theory

Creation of Automated Tests for Edu Database

Abstract:

The aim of this bachelor's thesis is to describe creation of automated tests for the Database course in the University of Tartu. In this thesis a program and automated tests are developed to be used on database. The thesis also explains the importance of automated testing.

Keywords:

Database, test automation

CERCS: P175 Informatics, systems theory

Sisukord

1. Sissejuhatus 4
1.1 Seotud töö 4
2. Taustinformatsioon 5
2.1 Andmebaas Edu 5
2.2 Ülesande vajalikkus 5
3. Andmebaaside automaattestimine 6
3.1 Automaattestimine üldiselt 6
3.2 Andmebaaside automaattestimise vahendid 6
4. Testimisplaan 7
4.1 Loodud testimisvahend 7
4.2 Testimiseks ettevalmistus 7
4.3 Andmete kontrollimine 7
4.3.1 Metaandmete kontrollimine 7
4.3.2 Tabeliandmete kontrollimine 8
4.4 Tulemustest teavitamine 9
5. Edasine arendus ja kokkuvõte 10
5.1 Edasine arendus 10
5.2 Kokkuvõte 10
6. Viidatud kirjandus 11
Litsents 12

1. Sissejuhatus

Andmebaaside testimine on tarkvara arenduses väga tähtsal kohal. Samas andmebaaside käsitsi kontrollimine on aga väga ajamahukas.

Antud bakalaureusetöö keskendub aines Andmebaasid tehtavale õppeotstarbelisele andmebaasile Edu. Töö raames koostatakse automaattestid, et kontrollida üliõpilaste tehtud tööde õigsust. Teste on vaja, sest igal aastal võtab seda ainet üle kahesaja tudengi ning aine läbimiseks on vaja esitada ka andmebaas Edu. Hetkel toimub andmebaaside ülevaatus käsitsi, mis võtab õppejõududelt palju energiat ning aega.

Teises peatükis kirjeldatakse automaattestimise vajalikkust ning tutvustatakse teisi võimalusi testimise jaoks.

Kolmandas peatükis kirjutatakse töö raames koostatud testimisvahendi struktuurist ning kirjeldatakse selle tööd.

1.1 Seotud töö

Varasemalt on sarnast tööd teinud bakalaureusetöö raames Marten Siiber. Töö kannab pealkirja „Aine „Andmebaasid“ automaattestimise vahend“ ja on mõeldud andmebaas Ope jaoks. Sellest tööst on saadud antud töö lahendamiseks mõtteid, kuid antud lahendus ei sobi selle bakalaureusetöö eesmärkide täitmiseks.

Ainet „Andmebaasid“ õppivad tudengid koostavad mitmeid andmebaase ning eelnevalt valminud töö ei sobi andmebaas Edu jaoks. Järgnevalt on välja toodud paar tähtsamat muudatust andmebaasi testimisel.

Esiteks, testimiseks on vajalikud ainult testitav ja korrektne andmebaas. Seega, kui andmebaasi koostamise ülesanne peaks muutuma, siis saab õiget andmebaasi korrastada mõne SQL-päringuga.

Teiseks, testimiseks kasutatakse andmetüüpi *LinkedHashMap*, sest andmete lisamise ja kontrollimise keerukus on sellel konstante kiirus ehk O(1).[1] Viimased on peamised operatsioonid, mida antud lahenduses kasutatakse.

2. Taustinformatsioon

Käesolevas peatükis seletatakse andmebaas Edu olemust ja ülesande vajalikkust.

2.1 Andmebaas Edu

Andmebaas Edu on Tartu Ülikooli õppeaines Andmebaasid loodav õppeotstarbeline andmebaas, millega kontrollitakse üliõpilase teadmisi. Selles andmebaasis tehakse läbi tabelite loomine, andmete sisestamine failist, vaadete loomine ja välisvõtmete (ingl *foreign key*) loomine. Andmebaasi loomiseks kasutatakse haldamissüsteemi SAP SQL *Anywhere*.

2.2 Ülesande vajalikkus

Andmebaaside ainet läbib aastas üle kahesaja tudengi. Tööde arv on suur ning hetkel on probleem, et kõiki töösid kontrollitakse käsitsi. Kuigi on tehtud automaatseid andmebaasi testimisvahendeid, siis pole otstarbekas neid antud andmebaasi jaoks kasutada, sest enamasti on nad liiga kallid ning võimsad.

Lisaks võib hiljem olla plaan alustada Tartu Ülikooli Andmebaaside MOOC'i (ingl *massive open online course*) kursustega. MOOC tähendab üldiselt veebipõhist massikursust, mille tõttu oleks vaja veelgi rohkem kontrollida.

3. Andmebaaside automaattestimine

3.1 Automaattestimine üldiselt

Automaattestimine on ühe programmi testimine teise tarkvara abil. Automaatselt testimine on võrreldes manuaalse testimisega:

- kiirem;
- vähem ressursse nõudvam ja
- puuduvad inimlikud vead, mis võivad samasugust ülesannet tehes tekkida.[2]

Seega automaatsete testidega on võimalik inimressurssi mujale suunata. Samas leidub ka puudusi, mis võivad automaattestimise hoopis ebaefektiivseks teha. Mõned puudused on näiteks:

- lähtekoodi pidev muutumine, kuid teste ei muudeta;
- testimisel ei ole arvestatud kõikide võimalustega, mille tõttu on vale arusaam, et testid läbitakse edukalt.[2]

Seega programmi kirjutamisel ei saa täielikult loota automaattestide peale, sest need võivad vananeda ning anda ebatäpseid vastuseid.

3.2 Andmebaaside automaattestimise vahendid

Andmebaaside automaattestimiseks on saadaval nii mõnedki vahendid (nt TestingWhiz, tSQLT), kuid need ei ole sobivad SAP SQL *Anywhere* jaoks, sest seal kasutatakse teisi SQL versioone. Lisaks on enamik automaattestimise vahendeid tasulised ning sisaldavad funktsioone, mida antud bakalaureusetöö ülesande lahendamiseks ei ole vaja kasutada.

Lisaks on olemas vahend nimega SQLZOO[3]. Selles antakse kasutajale väike ülesanne, mis tuleb SQL päringu abil ära lahendada. Lahenduse saatmisel antakse kasutajale tagasisidet tehtud töö kohta. Oma lihtsuse poolest sobib see ka päringute harjutamiseks, kuid mitte andmebaaside testimiseks.

4. Testimisplaan

Käesolevas peatükis kirjeldatakse programmi üldist arhitektuuri. Peatükis põhjendatakse tehtud valikuid ning selgitatakse nende tööpõhimõtet. Testimine koosneb kahest osast: metaandmete ning tabelis olevate andmete testimisest.

4.1 Loodud testimisvahend

Testimisvahendi tegemisel otsustati teisi raamistike mitte kasutada. Seda põhjusel, et tulevaste edasi arendamiste käigus ei tuleks peale Java programmeerimisoskuse ning SQL teadmiste midagi juurde õppida.

Testimisvahendis kasutatakse andmetüüpi *LinkedHashMap*. Seda põhjusel, et andmebaasis esineb päringuid, millel peab olema kindel järjekord. Tavalist *HashMap*'i kasutades ei ole õige järjekorra tagasi saamine garanteeritud ja seega võivad saadud tulemused valeks osutuda.[4]

Lisaks on testimisvahendi käigus kohendatud SQL-päringutes kasutatavaid tabeleid, sest SAP SQL *Anywhere* on oma uuendustega vanemaid tabeleid märkinud iganenuks (ingl *deprecated*).

4.2 Testimiseks ettevalmistus

Testimise alustamiseks on vaja korrektselt koostatud Edu andmebaasi, mille pealt kontrollitakse õigeid tulemusi, ja testitavat andmebaasi. Klassile BaseTest.java antakse ette kaks kataloogiteed, esimesel kohal peab olema õige ning teisel testitav andmebaas. Kõigepealt luuakse ühendus mõlema andmebaasiga. Seejärel tehakse SQL-päringud andmebaasidele, et kätte saada metaandmed ning tabelites olevad väärtused.

Päringutest saadav metaandmete *ResultSet* tüüpi muutuja ebamugavuse (andmete võrdlemine erinevate tabelite vahel on keeruline) tõttu kasutatakse siin kohal meetodit *resultSetToListOfHashMap*(), mis muudab *ResultSet*'i *LinkedHashMap*'iks. Tabeli andmete *ResultSet*'is tehakse lisaks veel üks muutus – see pannakse veel kord *LinkedHashMap*'i, kus võtmeväärtuseks jääb tabeli nimi, et hiljem oleks lihtsam kontrollida tabeli andmete õigsust. Lõpuks lõpetatakse andmebaasiga ühendus.

4.3 Andmete kontrollimine

Saadud andmetega viiakse läbi testid. Parimal juhul tuleb testitava andmebaasi andmeid korrektsega võrrelda kasutades meetodit *equals*(). See tähendab, et andmebaas saab kontrollitud kuue testiga. Muudel juhtudel testitakse andmebaaside metaandmeid ja tabeliandmeid.

4.3.1 Metaandmete kontrollimine

Andmebaasis on olemas kõikide loodud objektide metaandmed. Need andmed on lihtsasti kättesaadavad – selleks tuleb teha päring andmebaasi süsteemitabelitesse. Saadud päringu vastuseks tuleb palju testimise jaoks ebavajalikku (nt tabeli loomise kellaaeg), seega tuleb seda *SELECT*-päringu abil sorteerida ja vajadusel kasutada *REPLACE* käsku, et tulemust oleks võimalik loetavamaks teha. Sealjuures, kuna metaandmete järjekord ei ole tähtis, siis päringutes on kasutusel käsk *ORDER BY*, mis paneb vastused kas kasvavasse või kahanevasse järjekorda, et andmebaaside vaheline võrdlus oleks täpsem.

```
SELECT tname table_name,
REPLACE(colnames,' ASC', '') unique_columns
FROM SYS.sysindexes
WHERE creator = 'DBA' AND indextype = 'UNIQUE constraint'
ORDER BY tname
```

Joonis 1. Näide päringust, mis võtab andmeid *SYS.sysindexes* tabelist.

Jooniselt 1 on näha kuidas tehakse päring tabelile *SYS.sysindexes*. Kuna seal tabelis on palju ebavajalikke tulpi ning ridu, siis sorteeritakse seda nii, et jääks ainult tabelinimi (*table_name)* ja unikaalsed väärtused (*unique_columns*).

Joonisel 1 olevaga ja lisaks veel kolme päringuga saadakse teada mõlemast andmebaasist vajalikud metaandmed, milleks on:

1. tabelite nimed;
2. tulpade arv;
3. ridade arv;
4. tabelitüüp (vaade või tabel);
5. tulpade nimed;
6. tulpadele antud maksimaalne pikkus;
7. tulbatüüp;
8. tulba vaikimisi väärtus;
9. välisvõtme nimi;
10. mida teha, kui tabelite vahel on välisvõti ning ülemtabelis väärtused muutuvad ja
11. unikaalsete tulpade nimed.

Pärast päringute teostamist läbitakse saadud vastustega teste. Kui mõlemas tabelis on samad metaandmed, siis loetakse metaandmete testid õnnestunuks ning liigutakse edasi tabeliandmete testimisele. Kui mõni test ebaõnnestub, siis otsitakse vigane koht üles ning teavitatakse sellest testide käivitajat.

4.3.2 Tabeliandmete kontrollimine

Tabeliandmetega testimine toimub sarnaselt metaandmete omale. Siin kontrollitakse objekte, mida metaandmetega kätte ei saa - tabelites ja vaadetes olevaid väärtusi.

Andmebaaside kõikidest tabelitest võetakse andmed muutujasse vastava päringuga: *SELECT * FROM tabel,* kus *tabel* on muutujanimi. Tulemuseks saadakse *LinkedHashMap*, mille võti on tabelinimi ning väärtus on tabeli andmetest koosnev *LinkedHashMap*.

Saadud tulemusi võrreldakse testitava ja korrektse andmebaasi vahel. Vea esinemisel otsitakse tabelist vigane rida ning teavitatakse sellest kasutajat.

Mõnes ülesandes on nõutud käsu *ORDER BY* kasutamist. Antud testimisvahend katsetab ka seda. Nimelt nähakse testides, et andmed on erinevad. Erinevuste tekkel võetakse mõlema andmebaasi sama tabel ja proovitakse leida nende erinevus. Luuakse kaks uut muutujat, mille väärtused on esimene tabel ilma teise tabeli elementideta ning teine tabel ilma esimese tabeli elementideta. Kui mõlema muutuja väärtuste pikkused on 0, siis saame järeldada, et tabelites olid õiged andmed, kuid need olid vales järjekorras. Seega jäi andmebaasi koostajal kas *ORDER BY* tegemata või kasutati seda valesti.

4.4 Tulemustest teavitamine

Testimisvahend annab kasutajale teada, kas läbitud testid õnnestusid või kukkusid läbi. Selleks kirjutatakse ekraanile vastav tekst sõltuvalt testimise tulemustest.

Testi läbikukkumisel kuvatakse kasutajale veaga koht ning oodatav õige vastus. Kui probleem juhtus metaandmete testimisel, siis tabeliandmeid edasi ei kontrollita, sest võib järeldada, et tabelitesse on vale andmebaasi ülesehituse tõttu pandud ka valed andmed.

```
Vastuseks oodati: [{foreign_key_name=fk_faculty_person_dean, on_event=D, do_action=N}]
Vastuseks saadi: [{foreign_key_name=fk_faculty_person_dean, on_event=D, do_action=C}]
```

Joonis 2. Tulemuse ebaõnnestumine.

Joonisel 2 on toodud näide illustreerimaks juhtu, kui tabel on valesti koostatud. Antud näites on välisvõtmele omistatud vale väärtus. Tabeli koostaja on oma võtme loomisel öelnud, et kirje tuleb kustutada, kui ülemtabelis vastav väärtus kustutatakse. Oodati aga, et kustutamise asemel pandaks väärtuseks NULL.

Testide õnnestumisest annab kasutajale teada ekraanile kuvatav tekst, mille sisu on „Andmebaas on korrektselt koostatud!“. Positiivne tagasiside antakse selleks, et kasutaja saaks aru, et testimisvahend on töö edukalt ära teinud.

5. Edasine arendus ja kokkuvõte

5.1 Edasine arendus

Antud bakalaureusetöö jooksul koostatud testimisvahendit on nii mõnelgi moel võimalik edasi arendada.

Üks võimalus on see integreerida Moodle'i keskkonda, kuhu tudengid ka oma tööd esitavad. Seal võiks toimuda testimine automaatselt, et kiirendada kontrollimise protsessi. Sealjuures õppur saaks ka kohest tagasisidet oma töö kohta.

Teiseks võiks uurida ühendatud serverite (ingl *linked server*) võimalusi. Kui tudengi loodud ning korrektne andmebaas kokku ühendada, siis loodetavasti on võimalik kõik testid andmebaasi siseselt läbi viia. Praeguses olukorras puudub andmebaasidel omavahelise informatsiooni edastuse luba.

Programm loodi nii, et võimalikult vähe oleks koodi sisse kodeeritud (ingl *hard coded*). Seega võib otsida võimalusi sama programmi kasutada ka teiste andmebaaside testimisel. Üks tingimus (tulenevalt andmebaas Edu realiseeringust) oleks, et andmebaasis loodud vaadete nimed algaksid algusega „v_".

5.2 Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli arendada andmebaasi Edu automaatselt kontrolliv programm, mille abil saab õppejõud testida tudengite poolt esitatavaid andmebaase.

Töö eesmärk sai enamjaolt täidetud ning programmi on võimalik kasutada mõne hiireklõpsuga. Programm edastab kasutajale veaga rea ning teavitab, mida tegelikkuses andmebaasilt oodati.

Loodud programmi automaattesti kvaliteet saab kontrollitud ja parandatud tulevikus Andmebaaside kursusel. Töö mahu suurenemise tõttu jäi tööst välja algselt püstitatud eesmärk testide integreerimiseks Moodle'sse.

6. Viidatud kirjandus

[1] Java Documentation

https://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashMap.html (14.05.2018)

[2] Colantonio, J. „Automation Testing Resources & Best Practices“

https://www.joecolantonio.com/2017/03/02/automation-testing/ (14.05.2018)

[3] SQLZOO

http://sqlzoo.net (14.05.2018)

[4] Java Documentation

https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html (14.05.2018)

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Robert Sepp**,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose **Automaattestide loomine andmebaasile Edu**,
(*lõputöö pealkiri*)

mille juhendaja on Vambola Leping,
(*juhendaja nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **14.05.2018**