TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

Andmebaasi tiražeerimise kooskõlalisuse jälgimise vahend Äriregistri näitel

Bakalaureusetöö

Üliõpilane: Rauno Kulla
Üliõpilaskood: 112606IABB
Juhendaja: dotsent Erki Eessaar

Tallinn
2014

Autori deklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

---------- ----------

(*kuupäev*) (*allkiri*)

Annotatsioon

Andmebaasi tiražeerimise kooskõlalisuse jälgimise vahend Äriregistri näitel

Käesoleva töö tekkis vajadusest leida sobivaim tarkvara, mis oleks suuteline kontrollima Eesti Äriregistri (edaspidi Äriregistri) andmebaasi tiražeerimise käigus ühest andmebaasist teise kantavate ridade kooskõla. Kaaludes alternatiive, jõuti selgusele, et ühel või teisel põhjusel ei sobi ükski leitud lahendus. Seega sai selgeks, et sobiv tarkvara tuleb alles projekteerida ning realiseerida ning seda tehti käesoleva töö tulemusena. Kuna kõik Äriregistri tarkvarasüsteemid on arendatud Python'i programmeerimiskeeles, realiseeriti tiražeerimise kooskõla jälgimise vahend samuti Python'is. Realiseeritud lahenduse testimisel jõuti järeldusele, et selle aluseks oleva PostgreSQL *dblink* mooduli kasutamine teiste andmebaaside tabelitest andmete küsimiseks on Äriregistri andmemahtu arvestades piisava töökiirusega.

Tänu tarkvaras realiseeritud dünaamilisusele ning tarkvara rahuldavale töökiirusele, on võimalik see kasutusele võtta ka teistes süsteemides peale Äriregistri, kus on kasutusel andmete tiražeerimine välisesse andmebaasi tingimusel, et andmebaas on realiseeritud PostgreSQL andmebaasisüsteemi kasutades.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 68 leheküljel, 6 peatükki, 16 joonist, 13 tabelit.

Abstract

A Tool for Monitoring Consistency of Database Replication Based on the Example of Business Register

The necessity of this work raised from the need to find the most suitable software that would be able to verify consistency of rows that are replicated from one database of Estonian Business Register (Business Register in short) to another. After considering alternatives, it became clear that for one or the other reason no available solution is sufficient. Therefore, suitable software had yet to be designed and implemented. This was done as the result of this thesis. Because all of the Business Register's software systems have been developed in Python programming language, the tool for monitoring consistency was also developed in Python. While testing developed software, it became clear that PostgreSQL's *dblink* module that allows us to ask data from tables of other databases and is a basis of this tool offers good enough performance considering the amount of data in the Business Register.

Due to the dynamism of the developed software and its adequate performance, it can be used besides Business Register in other systems where master database replication to slave database is used. The prerequisite is that the database has to be a PostgreSQL database.

The thesis is in Estonian and contains 68 pages of text, 6 chapters, 16 figures, 13 tables, etc.

Lühendite ja mõistete sõnastik

SQL ***Structured Query Language***

„Enimkasutatav päringukeel, mida toetavad kõik klient-server keskkonnale projekteeritud relatsioonandmebaasid. Päringukeeled kujutavad endast reeglite kogumit, mille alusel konstrueeritakse päringuid andmete otsimiseks andmebaasist. Erinevad andmebaasihaldurid toetavad erinevaid päringukeeli, kusjuures ainult SQL on pooleldi standardiseeritud.“ [1]

Python ***Python***

„Interpreteeritav objektorienteeritud programmikeel, mille lõi Guido van Rossum 1990. aastal. Python on väga hästi porditav, sest Pythoni interpretaatorid on saadaval peaaegu kõigi opsüsteemide jaoks. Pythoni edasiarendamisega tegeleb Python Software Foundation.“ [1]

PostgreSQL ***PostgreSQL***

„PostgreSQL on SQL andmebaasihaldur (SQL-andmebaasisüsteem) mõningase objektorienteeritud kallakuga. PostgreSQL on vabavara, kuid sellele vaatamata on see tehniliselt ja funktsionaalselt tõsine konkurent parimatele kommertsanaloogidele.“ [1]

MD5 ***MD5***

„1991. aastal prof. Ronald Rivest'i poolt loodud algoritm, mida kasutatakse digitaalallkirjade loomiseks. MD5 on mõeldud 32-bitistele masinatele ja on turvalisem kui vanem MD4 algoritm. MD5 on ühesuunaline räsifunktsioon, st. ta teeb sõnumist teatud numbrite jada, mida kutsutakse ka sõnumi kokkuvõtteks. Ühesuunalise räsifunktsiooni abil arvutatud sõnumikokkuvõtet võrdleb sõnumi saaja avaliku võtmega dekrüpteeritud sõnumikokkuvõttega, et kindlaks teha, kas keegi pole vahepeal sõnumit muukinud. Sellist võrdlust nimetatakse räsikontrolliks." [1]

Andmetüüp ***Data type***

„Väärtuste hulk, mille seast muutuja, konstant, funktsioon või muu avaldis võib võtta oma väärtuse." [1] Enamik programmikeeli toetab selliseid tüüpe nagu täisarvu tüübid, loogikatüüp, reaaltüübid ning ka stringitüübid, millesse kuuluvaid väärtuseid käsitletakse tähejadadena. [1]

Tiražeerimine ***Replication***

„Andmebaasihalduses hajusandmebaaside hoidmine sünkroniseerituna sel viisil, et kogu andmebaasi tervikuna või selle osi kopeeritakse regulaarselt teistesse võrgus olevatesse serveritesse." [1]

Londiste ***Londiste***

Londiste on PostgreSQL tiražeerimise tarkvara, mis on osa SkyTools komplektist. Londiste on realiseeritud Python programmeerimiskeeles Skype poolt. Londiste on asünkroonne ülem-alluv tiražeerimise süsteem. Asünkroonne tähendab, ei ole garanteeritud, et operatsioon, mis on tehtud ülembaasis on jõudnud ühessegi alluvbaasi ülembaasi operatsiooni täitmise hetkeks. Ülem-alluv tiražeerimise süsteem tähendab, et andmete muudatused, mis toimuvad alluv- baasis, ei kanta tagasi ülembaasi. Liiklus toimub ainult ülembaasist alluv- baasi. [2]

Muutmälu ***Random-Access Memory***

Muutmälu, süvapöördusmälu. Arvuti keskne mäluseade, kuhu saab andmeid kirjutada ja kust saab neid lugeda. „Süvapöördus (Random Access) tähendab seda, et igal mälupesal on oma aadress ning nii lugemiseks kui ka kirjutamiseks on võimalik pöörduda suvalise aadressi poole. Enamik muutmälusid pole säilmälud, see tähendab toite väljalülitamisel mälus olevad andmed hävivad.“ [1]

Linux ***Linux***

„Tasuta levitatav UNIX-i laadne operatsioonisüsteem, mis jookseb tervel real riistvaraplatvormidel, sh Intel’i ja Motorola mikroprotsessoritel. Linuxi kerneli töötas välja soomlane Linus Torvalds. Kuna Linux on tasuta ja jookseb nii IBM PC, Macintosh’i kui ka Amiga arvutites, siis muutub see viimasel ajal üha populaarsemaks.“ [1]

Jooniste nimekiri

Tabelite nimekiri

Sisukord

Sissejuhatus

Andmebaasi tiražeerimine toob andmebaasi administreerimise protsessi kaasa mitmeid hüvesid. Kopeerides kõik andmebaasi read teise andmebaasi ning kasutades teist andmebaasi välistele kasutajatele andmete näitamiseks, saab vähendada võrgu koormust põhiandmebaasis ning seeläbi vähendada aega, mis kulub serveritel päringute töötlemiseks.

Kuid andmete ühest andmebaasist teise kopeerimisel on ka ohud. Ei saa loota tiražeerimise protsessi läbi viiva tarkvara 100-protsendilisele töökindlusele, sest tarkvara võib ootamatult jätta mõne rea teise andmebaasi kopeerimata. Sellest kirjutatakse täpsemalt töö esimeses peatükis. Juhul kui kõik read ei ole vajalikesse andmebaasidesse kopeeritud, ei ole andmebaas enam terviklik. Kirjeldatud stsenaarium leidis aset Eesti Äriregistris (edaspidi Äriregister) ning tulevikus selliste olukordade vältimiseks on hakatud otsima sõltumatut lahendust mõne tarkvara näol, mis kontrolliks pidevalt tiražeerimise protsessi edukust ning teavitaks võimalikult kiiresti, kui on jäänud mõni rida teise andmebaasi üle kandmata. Puudu oleva rea kohta teavituse saamisel oleks vea parandamine juba Äriregistri töötaja ülesanne.

Käesoleva töö esimeses peatükis tutvustatakse lähemalt probleemi tausta ning selgitatakse olemasolevaid lahendusi ning põhjuseid, miks need Äriregistri vajadusi ei rahulda. Tehakse järeldus, et ainus võimalus on luua ise sobiv tarkvara. Teises peatükis tutvustatakse teoreetilisi aluseid, mida tarkvara arendamise käigus kasutatakse ning mida arvesse võetakse. Kolmas peatükk kirjeldab süsteemi kavandamist ja realiseerimist. Seal tuuakse välja eesmärgid, mille saavutamiseni töö tulemusena soovitakse jõuda ning süsteemi loomuseks vajalikud mudelid. Neljandas peatükis kirjeldatakse loodud kasutajaliidese kavandit, mille alusel on võimalik arendada välja kasutajaliides. Viiendas peatükis testitakse loodud tarkvara (taustaprotsessi) ning esitatakse järeldused testimisel saadud tulemuste kohta. Kuuendas peatükis tehakse kogu tööst kokkuvõtlikud järeldused ning tutvustatakse võimalikke edasiarendusi.

1. Probleemi detailne kirjeldus

Äriregistri tarkvarasüsteemid kasutavad tiražeerimisel lahendust, mille kohaselt kopeeritakse ridu põhiandmebaasist välisesse andmebaasi. Põhiandmebaas, edaspidi nimetatud keskbaas, on ühendatud Äriregistri menetlustarkvaraga, kus kohtute registriosakondade töötajad teevad vastavalt sisse tulnud kandeavaldustele toiminguid Äriregistri andmetega. Andmed jõuavad pärast keskbaasi lisamist tiražeerimise protsessi kaudu teatud aja möödudes välisesse baasi, (edaspidi nimetatud veebibaas). Keskbaasi tabelitesse lisatakse ridu, kuid olemasolevaid ridu ei kustutata. Leidub ridu, mida keskbaasis muudetakse, kuid seda tehakse harva.

Veebibaas on ühendatud Äriregistri ettevõtjaportaaliga, milles saavad ettevõttega seotud isikud ettevõtte andmetega tutvuda ning kandeavaldusi menetlusse lisada. Samuti on veebibaas ühendatud Äriregistri teabesüsteemiga, kus andmetest huvitatud isikutel on võimalik andmetele tasuliselt ligi pääseda.

Äriregistris on kasutusel avatud lähtekoodiga ja tasuta pakutav PostgreSQL andmebaasihaldur. Andmete keskbaasist veebibaasi saatmiseks on valitud Londiste tiražeerimise lahendus. Tiražeerimine toimub automaatselt teatud aja tagant, millega kantakse kõik keskbaasis toimunud muudatused veebibaasi üle. Lisaks on võimalik andmebaasi administraatoril käivitada käsku ’repair <tabeli nimi>’, mis üritab tabelid sünkroniseerida, kirjutades välja SQL laused, mis parandavad erinevused.

Äriregistris on arvatud, et Londiste tarkvara ning selle lisavõimalused on piisavad, et andmebaasi terviklikkus oleks igal ajal tagatud. Seda seni, kuni 2014. aasta alguses pöördus meie poole ettevõtjaportaali kasutaja murega: „Miks ei ole tema muutmiskandele kohtunikuabi poolt tehtud puuduste kõrvaldamise määrus ettevõtjaportaali jõudnud?“

Pärast kasutaja pöördumist hakati otsima probleemi põhjust ning ilmnes, et kõnealuse määruse rida on olemas keskbaasis, kuid mingil põhjusel ei ole sellist rida veebibaasis. Esialgne arvamus oli, et tiražeerimise protsess on seisma jäänud, kuna määrus oli tehtud samal kuupäeval ning arvasime, et tegu on esimese pöördumisega ja sarnaseid pöördumisi tuleb veelgi. Seda võimalust kontrollides jõudsime järeldusele, et antud

variant ei ole tõene, kuna järgmised kohtunikuabide poolt tehtud määrused olid jõudnud keskbaasist veebibaasi.

Edasiste määruste ridade veebibaasis olekule selguse saamise järel jäi ainsaks võimalikuks selgituseks, et andmebaasid ei ole sünkroonis. Andmebaasi administraator käivitas käsureal baaside sünkroniseerimise käsu, mis pidanuks lisama puuduva rea ka veebibaasi. Sünkroniseerimise protsessi lõppedes ei leidnud puuduolevat rida endiselt veebibaasist.

Seejärel hakkasid olukorda uurima arendajad. Kuna see oli täielik juhus, et kasutaja probleemile peale sattus, oli selge, et see ei pruugi olla ainus veebibaasist puuduv rida. Arendajad kontrollisid andmebaasides olevaid andmeid. Tuli välja, et veebibaasist on erinevatest tabelitest puudu 15 rida. Leiti, et Äriregistri baaside tiražeerimise protsessi täidab vana Londiste tarkvara versioon. Londiste tarkvara uuendati ning käivitati sünkroniseerimine, mis lisas kõik puuduolevad read veebibaasi.

Juhtum tõi Äriregistrisse arusaama, et olukorras, kus andmebaaside tiražeerimine ei tööta korralikult, jättes ridu vahele, ei teavita Äriregistrit sellest miski. Sellest ajast peale soovib Äriregister leida lahenduse, kuidas jälgida andmebaasi tiražeerimise tulemust. Kiire lahendusena hakati pärast tööaega käivitama andmebaaside kriitilistes tabelites ridade loendamise päringuid. Esmalt tehakse ridade loendamise päring keskbaasi tabelist ning veebibaasi tabelist ja kontrollitakse kas ridu on samapalju.

Praegune lahendus on mitmes mõttes puudulik.

1. Kooskõla saab kontrollida ainult pärast tööaega. Keskbaasis lisandub tööajal tabelitesse pidevalt uusi ridu, mida koheselt veebibaasi ei kanta. Tehes tööajal ridade loendamise päringu mõlemast baasist, on ridade arvu erinevus ette teada tulemus. Pärast tööaega keskbaasi tabelitesse uusi ridu ei lisandu, seega erinevate baaside tabelite ridade arvu võrdlemisel on tulemuse põhjal võimalik järeldusi teha.

2. Ei võrrelda ridade sisu. Kuigi probleemi aluseks oli erinevates baasides oleva tabeli ridade arvu võrdlus, tuleks tiražeerimise protsessi tulemuslikkust hinnata ka tiražeeritud ridade sisu vastavuse põhjal. Kõikide võrdlemist vajavate tabelite

ridade sisu võrdlus ei ole ajakulu mõttes mõistlik. Seega ei pärita tabelite ridade sisusid, et neid omavahel võrrelda.

3. Äriregistri haldur peab praeguse kasutusel oleva lahendusena igapäevaselt ise võrdlusi käivitama, mis tõstab halduri töömahtu.

Kuna ajutiseks lahendusena kasutusele võetud päringud ei taga piisavat kontrolli, on vaja leida tõhusam lahendus.

1.1 Kaalutud variandid

Esialgne plaan oli leida olemasolev tiražeerimise jälgimise (monitooringu) lahendus, mida hakata kasutama Äriregistri andmebaasi tiražeerimise tulemuste jälgimiseks. Leitud võimalikud lahendused olid järgmised.

- Lisada tabelis olevad väärtused kõik esmalt massiivi, kasutades selleks PostgreSQL funktsiooni array_agg(). Tekkiv massiiv muuta seejärel String'iks PostgreSQL funktsiooniga array_to_string(). Kõiki tabeli ridade väärtusi sisaldavast String'st leida viimaks MD5 räsiväärtus. Kirjeldatud päring tuleb koostada mõlemast baasist ning võrreldes saadud MD5 räsiväärtusi on võimalik määrata kas tabelid on võrdsed või mitte. Lahendusel on mitmeid puudujääke. Päringud on väga ressursi-mahukad, sest terve tabeli kõikidele veergudele mitme SQL funktsiooni rakendamine võtab liiga kaua aega. Lisaks ei pruugi suuremad tabelid muutmälusse mahtuda. Kõigi tabeli ridade põhjal leitud räsiväärtuste võrdlemisel ei oleks võimalik leida, millises reas on viga tekkinud. String'ide võrdluseks kasutatav MD5 räsiväärtuse leidmine ei ole 100% täpne, kuna on leitud, et kahel erineval väärtusel võib olla sama MD5 räsiväärtus. Selle avastasid 2005. aastal Xiaoyun Wang ja Hongbo Yu, avaldades artikli, milles kirjeldasid algoritmi, mis leiab kaks erinevat 128 bitist jada, mis omavad sama MD5 räsiväärtust. [3]

- Alluvbaasi transaktsioonide järje teadasaamiseks on PostgreSQL olemas funktsioon pg_xlog_location_diff(). Esmalt tuleb see käivitada ülembaasis ning seejärel alluvbaasis. Kui tulemus on alluvbaasis võrdne või suurem, siis on baasid samas seisus. Äriregistris on kasutusel käesoleva töö kirjutamise ajal

PostgreSQL versioon 9.1.13, kuid pg_xlog_location_diff() funktsioon on toetatud alates PostgreSQL versioonist 9.2, seega antud variant ei ole sobilik.

Kuna mõlemal variandil on puudused, miks neid ei saa Äriregistri andmebaaside tiražeerimise jälgimiseks kasutusele võtta, on vaja ise sobiv lahendus realiseerida.

2. Teoreetiline taust

2.1 Andmebaasi tiražeerimine

Andmebaasi tiražeerimine on sagedane elektrooniline andmete kopeerimine ühest andmebaasist teise, eesmärgiga tagada kõikidele kasutajatele sama teabe tase. Tulemuseks on hajutatud andmebaas, milles kasutajad saavad juurdepääsu oma ülesannete täitmiseks asjakohastele andmetele, ilma, et see häiriks teiste kasutajate tööd. [4]

Andmebaasioperatsioonide hajutamiseks on Äriregistris kasutusel kaks andmebaasi. Keskbaas on ühenduses menetlustarkvaraga, kus registriosakondade töötajad teevad Äriregistri andmetega toiminguid. Veebibaas on ühenduses Äriregistri ettevõtjaportaaliga ning teabesüsteemiga, kus saavad andmeid vaadata huvi tundvad isikud. Sellise andmebaaside jaotusega on koondatud ühte andmebaasi kõik andmete sisestused ning teise andmebaasi väliste kasutajate poolt tehtavad päringud. Tiražeerimist kasutatakse andmete saatmiseks keskbaasist veebibaasi, et tagada veebibaasiga ühenduses olevas ettevõtjaportaalis ja teabesüsteemis andmete päringute vastuste korrektsus.

2.2 Tiražeerimise õnnestumise määramine

Tiražeerimise protsessi tulemuste kooskõlalisust saab määrata, võrreldes iga tiražeeritava tabeli ridu erinevates baasides. Võrrelda ei tohiks kõiki tabelis olevaid ridu, sest iga rida võrreldes läheb jälgimine ajaliselt liiga pikaks ning koostatud päringud oleksid andmebaasile liiga koormavad. Seetõttu tuleks andmebaasist iga tiražeerimise kooskõla kontrollimise seansi ajal pärida ainult kindlasse vahemikku kuuluvad read ning võrrelda neid ridu erinevates baasides. Ridade omavahelise kontrolli jaoks ei ole PostgreSQL-is korralikku aega kokkuhoidvat funktsiooni. Siiski oleks võimalik teha reast massiiv ning massiivist omakorda sõne. Selline lahendus koosneks mitme funktsiooni väljakutsest, mis ei annaks kindlasti aja kokkuhoidu. Seega tuleks erinevate baaside tabelite ridu omavahel võrrelda rakenduses.

Tiražeerimine toimub väikese ajalise viitega. Tänu sellele võib peamises andmebaasis, keskbaasis, olla tööajal, mil tabelitesse lisatakse registriosakondade töötajate poolt ridu koguaeg juurde, tabelites veebibaasi kopeerimata ridu. Seetõttu tuleks kooskõla kontrollimise ajal viimase kontrollitava reana arvestada veebibaasi tabelis olevat suurima primaarvõtme väärtusega rida (primaarvõtmete väärtused kasvavad monotoonselt).

Viimase kontrollitud rea primaarvõtme väärtus tuleks salvestada selleks ettenähtud tabelisse, et järgmine kord, kui tiražeerimise kooskõla jälgimise vahend käivitub, oleks olemas punkt, kust kontrolli alustada. Seega ei teeks kontrollimise programm üleliigset tööd, kontrollides juba kord üle vaadatud ridu.

2.3 Päring teisest baasist

PostgreSQL 9.1 pakutav lahendus teisest andmebaasist päringu tegemiseks on moodulis *dblink* sisalduv funktsioon dblink(). Äriregistris on hetkel (2014. aasta mai) kasutusel PostgreSQL versiooniga 9.1.13 ning juhul kui tulevikus plaanib Äriregister minna üle uuema versiooni, siis ka uuemates versioonides on *dblink* toetatud.

Dblink moodul on osa PostgreSQL *contrib* paketist, mis peab dblink() funktsiooni käivitamiseks olema installitud. Linuxi käsurealt käib installeerimise käsk järgnevalt:

```
yum install postgres*contrib
```

Pärast paketi installeerimist tuleb andmebaasi siseselt lisada *dblink* laiendus:

```
CREATE EXTENSION dblink;
```
[5]

dblink() funktsioon väljastab read, mis leiti funktsioonile argumendina etteantud päringu tulemusena. Igas päringus tuleb täpsustada oodatavad veeru tüübid. Muidu *dblink* ei tea, mida oodata. Näide:

```
SELECT * FROM ('dbname=andmebaasinimi', 'SELECT tabel_id,
nimi FROM tabel') AS t(tabel_id bigint, nimi varchar(30));
```
[6]

2.4 Automaatne käivitamine

Loodav taustaprotsess peab käivituma päevas kindel arv kordi. Äriregistri serverites kasutusel olevas Linuxi operatsioonisüsteemis on võimalik kasutada regulaarse käivitamise tööriista – *cron. Cron*'i saab seadistada käivitamaks vajalikku programmi kindlatel soovitud aegadel. Käivitades Linuxi käsurealt *cron*'i alustamise käsu, tuleb määrata käivitamise aeg ning lause, milleks on käivitatava programmi asukoht.

Käivitamise aja määramine käib järgneva formaadi kohaselt:

MI HH DD MM DW lause

Tabel 1. Linuxi käsureal *Cron*'i käivitamise formaadi kirjeldus.

Väli	Kirjeldus	Võimalikud väärtused
MI	Minuti väli	00-59
HH	Tunni väli	00-23
DD	Kuupäeva väli	1-31
MM	Kuu väli	1-12
DW	Nädalapäeva väli	0-6

Käesolevas töös on vaja käivitada tiražeerimise kooskõla jälgimise taustaprotsess tööaegadel, mil ridu tuleb keskbaasi koguaeg juurde. Kõige mõistlikumad käivitamise ajad oleks 6:00, 10:00, 14:00 ning 18:00. Kirjeldatud aegadel taustaprotsessi käivitamiseks tuleb kasutada järgnevat lauset:

```
00 6,10,14,18 * * 1-5 /home/arireg/bin/kontrollialgoritm.py
```

Ülal toodud käsus tähendab '*', et seda aega ei täpsustata ehk igal võimalikul ajal. [9]

Kuna *cron*'iga sarnane funktsionaalsus on võimalik lisada ka Windows Server operatsioonisüsteemile, näiteks *CRONw [10]* või *pycron [11]*, on võimalik käesolevas töös realiseeritavat kontrolli algoritmi kasutada automaatselt käivitatava taustaprotsessina ka Windows Serveri operatsioonisüsteemiga.

3. Süsteemi kavandamine ja realiseerimine

3.1 Ülesande püstitus

Järgnevalt on välja toodud neli eesmärki, mida töö tulemusena soovitakse saavutada.

Peamiseks eesmärgiks on töötada välja vahend Äriregistri andmebaasi tiražeerimise tulemuste kooskõlalisuse jälgimiseks, mis pakuks tõhusamat kontrolli andmete ühest andmebaasist teise kopeerimise üle, kui töö kirjutamise ajal kasutusel olevad ridade lugemise päringud. Kontrollides iga kord ainult uusi juurde tulnud ridu, on päringute täitmine ajasäästlikum ning seega on võimalik võrrelda ka erinevas baasides olevate ridade vastavust.

Teiseks eesmärgiks on realiseerida kontrollivahend automaatse taustaprotsessina, mis käivitub päevas teatud arv kordi. Juhul kui taustaprotsess leiab andmetest ebakõla, siis teavitab see Äriregistri haldureid e-meiliga. Automaatselt käivituv taustaprotsess võtaks haldurilt vajaduse igapäevaselt tiražeerimise tulemuste kontrolli läbi viia.

Kuna Äriregister on pidevas arendamises, on vaja taustaprotsess realiseerida selliselt, et kui tekib soov panna jälgimise alla uus tabel, siis seda peaks olema võimalik teha kasutajaliidese kaudu. See tähendab, et taustaprotsessi koodi ei tohi sisse kirjutada kontrollitavate tabelite põhjal tehtavaid täpseid SQL lauseid, vaid taustaprotsess peaks iga kontrollitava tabeli kohta ise SQL lause koostama.

Neljandaks eesmärgiks on jõuda selgusele, kas PostgreSQL päringu dblink vahendusel teise baasi täitmiseks saatmine on Äriregistri andmemahtu arvestades piisavalt kiire, et seda oleks sobilik kasutada lahenduses, kus iga kontrollitava rea kohta tehakse dblink vahendusel teisest baasist päring.

Realiseeritav tarkvara tuleb realiseerida kasutades Python'i programmeerimiskeelt, et oleks võimalik lisada kirjeldatav rakendus menetlustarkvarasse. Lisaks tuleb luua konfiguratsiooni fail, kuhu talletatakse andmebaasi aadressid ning info, millise

kasutajanimega ning parooliga andmebaasi sisenetakse. Samuti loetakse konfiguratsiooni failist emaili aadress, kuhu vea leidmise korral teavitus saata.

3.2 Funktsionaalsed nõuded

- **Kontrollitavate tabelite lisamine** – Peab olema võimalik vajadusel lisada kontrolli uusi tabeleid ilma, et peaks selleks koodi täiendama.
- **Kontrollitavate veergude lisamine** – Peab olema võimalik lisada kontrollitava tabeli veerge, mida tiražeerimise tulemuste kooskõla jälgimise vahend peab kontrollima, ilma, et peaks selleks koodi täiendama.
- **Kontrollitava veeru kustutamine** – Peab olema võimalik kontrollitavat veergu vajadusel kustutada ilma, et peaks selleks koodi muutma.
- **Kontrollitava tabeli kontrollist eemaldamine/kontrolli lisamine** – Peab olema võimalik ilma koodi muutmata võtta kontrollis olevat tabelit mõneks ajaks kontrollist maha ning hiljem vajadusel uuesti kontrolli lisada.
- **Kontrollitava tabeli muutmine** – Kontrollis oleva tabeli andmeid (tabelis olevaid ridu) peab olema võimalik muuta ilma, et peaks selle jaoks programmi koodi muutma.
- **Viimase kontrollitud rea muutmine** – Viidet tabelis viimati kontrollitud reale peab olema võimalik muuta ilma programmi koodi muutmiseta.
- **Kontrolli otsimine** – Kontrolle peab olema võimalik otsida kontrolli numbrilise identifikaatori või algusaja ning lõppaja alusel.
- **Kontrollide nimekirja järjestamine** – Vaikimisi peab kontrollide nimekiri olema sorteeritud kontrolli numbrilise identifikaatori järgi kahanevas järjekorras. See tagab, et kõige viimaste kontrollide andmed on kõige eespool. Peab olema võimalus sorteerida ka vigade arvu järgi.

- **Tabeli otsimine** – Peab olema võimalus otsida kontrollitavaid tabeleid tabeli nime järgi.

- **Kontrollitavate tabelite nimekirja järjestamine** – Vaikimisi peavad kontrollitavad tabelid olema järjestatud tabeli numbrilise identifikaatori järgi kasvavalt, kuid peab ka olema võimalus sorteerida kontrolli oleku, skeemi nime ja tabeli nime järgi.

- **Vea kuvamine** – Kontrollide nimekirjas peab eristama kontrolle, kus leitud vigade arv on suurem kui null. Samuti peab vigade nimekirjas eristama vigu, mille staatus on „parandamata“. Eristades peab kasutama sellistel ridadel punast kirjavärvi ning ühe punkti võrra suuremat teksti kui tavaridades.

- **Vigade otsimine** – Vigu peab olema võimalik otsida ajavahemiku järgi, kontrolli numbrilise identifikaatori järgi ning vea numbrilise identifikaatori järgi. Lisaks peab olema võimalik valida kas soovitakse tulemusse kõiki vigu või ainult kindla staatusega vigu. Vaikimisi peab kuvama kõiki vigu.

- **Vea kirjelduse detailsus** – Süsteem ei pea andma infot, millistes rea väljades on mittekooskõla erinevate andmebaaside vahel. Piisab, kui süsteem annab infot selle kohta, milline on mittekooskõlaliste ridade primaarvõtme väärtus. Täpsem vea sisu tehakse kindlaks süsteemi väliselt.

- **Vigade nimekirja järjestamine** – Vigade nimekirja järjestus peab olema staatuse järgi, et esimesena esitataks vead, mille staatus on „parandamata“. Vaikimisi peab olema teiseks järjestamise tingimuseks vea numbriline identifikaator, mille järgi on tulemus järjestatud kahanevalt.

- **Vea staatuse muutmine** – Tiražeerimise kooskõla jälgimise vahendi kasutaja peab saama muuta vea staatust. Kõik vea staatuse muudatused peab logima, sh ka süsteemi poolt tehtud muudatused. Logisse peab salvestama töötaja identifikaatori, kes vea staatusega muudatuse tegi.

- **Sisenemise õigused** – Tiražeerimise jälgimise vahendisse sisenemise õigused võivad olla ainult Äriregistri menetlustarkvara halduri rollil.

- **Kontrolli käivitamine** – Kontroll peab päevas automaatselt käivituma kellaaegadel: 06:00, 10:00, 14:00 ning 18:00. Lisaks peab kasutajal olema võimalik vajadusel nupuvajutusega kontrolli käivitada. Kui juba üks kontroll tegutseb, siis tohi olla võimalik käivitada uut kontrolli.

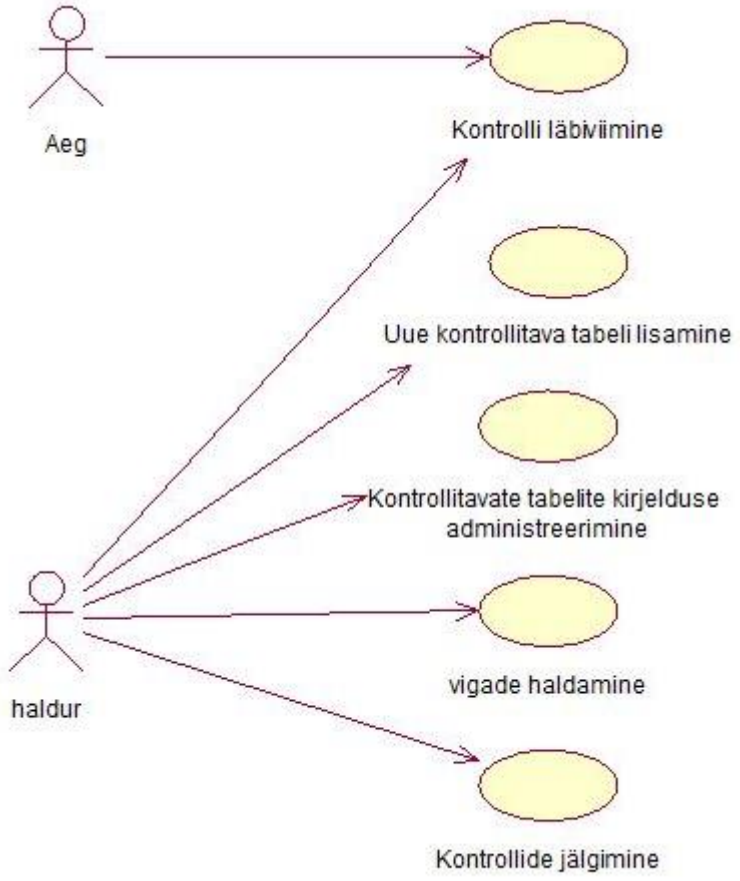
- **Vigaste sisendandmete sisestamise keelamine** – Peab olema välistatud võimalus sisestada sellist skeemi nime ning tabeli nime ja veeru nime, millist kontrollitavas andmebaasis ei eksisteeri.

- **Vigade salvestamine** – Vea leidmisel peab süsteem leitud vea asukoha andmed salvestama ning määrama vea staatuseks „parandamata“.

- **Vaikimisi väärtused** – Tabeli puhul peab olema vaikimisi määrang, et ei kontrollita. Vaikimisi ei kuulu veerg primaarvõtmesse.

3.3 Kasutusjuhtude mudel

Järgnevalt on kirjeldatud süsteemi kasutusjuhte. Selgitav diagramm on esitatud joonisel 1.



Joonis 1. Kasutusjuhtude diagramm.

Kasutusjuht: Kontrolli läbiviimine

Primaarne tegutseja: Aeg, haldur.

Osapooled ja nende huvid:

- Subjekt: Kontrollib tiražeeritud ridade kooskõla erinevates andmebaasides.
- Äriregistri veebisüsteemi kasutajad soovivad näha korrektseid andmeid.
- Äriregister soovib vältida probleemidest tulenevat negatiivset tähelepanu ja pakkuda kasutajatele head teenust.

Käivitav sündmus: Haldur annab kontrolli käivitamise korralduse või saabub üks automaatse käivitumise aegadest: 06:00, 10:00, 14:00 või 18:00.

Eeltingimused: On salvestatud kontrolli vajavate tabelite nimed ja veerud, mida tabelites kontrollida. Iga kontrollitava tabeli kohta on määratud vastav primaarvõtme väärtus, kust kontrollimist alustada.

Järeltingimused: Kontroll on läbi viidud. Iga tabeli viimane kontrollitud rea primaarvõtme väärtus on salvestatud, et kontrolli järgmisel käivitusel oleks võtta alguspunkt. Leitud vea korral on viga salvestatud.

Stsenaarium (tüüpiline sündmuste käik):

1. Subjekt käivitab kontrolli.
2. Süsteem registreerib kontrolli. OP 1.0
3. Süsteem otsib „parandamata“ staatuses olevaid vigu.
4. Süsteem alustab järjekorras iga tabeli kontrollimist.
5. Süsteem otsib viimase kontrollitud rea, mille alusel saab määrata esimese kontrollitava rea primaarvõtme väärtuse (alguspunkti).
6. Süsteem pärib veebibaasist kontrollitava tabeli kõige suurema primaarvõtme väärtuse, mis määrab viimase kontrollitava rea (lõpp-punkti).
7. Süsteem pärib keskbaasist alguspunkti ja lõpp-punkti vahelise primaarvõtme väärtusega read. Alguspunkt on viimati vaadatud rea primaarvõtme väärtus + 1 ning lõpp-punkt on veebibaasis olev kõige suurem rea primaarvõtme väärtus.
8. Süsteem pärib veebibaasist iga saadud reaga sama primaarvõtme väärtusega rea ning võrdleb ridu veergude kaupa.
9. Süsteem leiab, et read on kooskõlas ja liigub edasi järgmist rida võrdlema.
10. Kui kõik tabelid on kontrollitud, siis registreerib süsteem kontrolli lõpu aja, leitud vigade arvu ning tulemuse. OP 1.1
11. Kui leiti vigu, siis süsteem saadab välja emaili teavitamaks leitud vigadest. Kõigi leitud vigade kohta saadetakse üks e-meil (mitte iga vea kohta eraldi e-meili).

Laiendused (või alternatiivne sündmuste käik):

2a. Kui kontrolli protsess parajasti toimub, siis kontrolli ei alustata ning kasutajale väljastatakse vastav teade.

3b. Süsteem otsib iga sellises staatuses oleva veaga rea keskbaasist ning veebibaasist. Seejärel võrdleb süsteem saadud ridu veergude kaupa.

3c. Kui viga on parandatud, siis süsteem muudab vea staatust. OP 2.1

9b. Süsteem leiab ridu võrreldes vea ja registreerib viga sisaldava rea primaarvõtme väärtuse. OP 2.0

9c. Süsteem registreerib vea OP 2.2 ning liigub edasi järgmist rida võrdlema.

Kasutusjuht: Uue kontrollitava tabeli lisamine

Primaarne tegutseja: Haldur - (edaspidi subjekt).

Osapooled ja nende huvid:

- Subjekt: soovib, et kõik vajalikud tabelid oleksid kontrolli kaasatud.

Käivitav sündmus: Äriregistri arendusega on tekitatud andmebaasi juurde tabel, mille ridu kopeeritakse välisesse andmebaasi. Subjektil tuleb lisada uus tabel kontrollitavate tabelite nimekirja.

Eeltingimused: Andmebaasis eksisteerib lisatav tabel. See tähendab, et kui lisatakse kontrolli tabel *ar.digitaliseerimine*, siis peab skeemis *ar* olema olemas tabel nimega *digitaliseerimine*. Samuti peavad kontrolli lisatavas tabelis olema olemas kõik kontrollitavad veerud.

Järeltingimused: Kontrollitav tabel ja selle veerud on süsteemis registreeritud.

Stsenaarium (tüüpiline sündmuste käik):

1. Subjekt annab korralduse uue kontrollitava tabeli lisamiseks.
2. Süsteem avab kontrolli uue tabeli lisamise akna.
3. Subjekt registreerib skeemi nime, tabeli nime ning salvestab.
4. Süsteem registreerib andmed. OP 4.0
5. Subjekt lisab järjest kõik kontrolli vajavad veerud.
6. Süsteem registreerib andmed. OP 5.0
7. Subjekt määrab, et tabelit tuleb hakata kontrollima.
8. Süsteem salvestab andmed. OP 4.1

Kasutusjuht: Kontrollitavate tabelite kirjelduse administreerimine

Primaarne tegutseja: Haldur - (edaspidi subjekt).

Osapooled ja nende huvid:

- Subjekt: soovib, et kõik vajalikud tabelid oleksid kontrolli kaasatud.

Käivitav sündmus: Soovib muuta kontrollis oleva tabeli kirjeldust, et kõik vajalikud tabelid/veerud oleksid kontrolli hõlmatud.

Eeltingimused: On registreeritud kontrollimist vajava tabeli nimi ja skeem.

Järeltingimused: Kontrollimist vajava tabeli kirjeldus on muudetud.

Stsenaarium (tüüpiline sündmuste käik):

1. Subjekt soovib alustada kontrollitava tabeli kirjelduse administreerimist.
2. Süsteem väljastab kõikide kontrollitavate tabelite nimekirja kus on näha nii tabeli nimi kui skeemi nimi, milles tabel paikneb.
3. Subjekt valib tabeli, mille kirjeldust muuta.
4. Süsteem avab kontrollitava tabeli kirjelduse administreerimise akna.
5. Subjekt lisab kontrollitavale tabelile uue veeru, mida kontrollida.
6. Süsteem salvestab uue kontrollitava veeru OP 5.0

Laiendused (või alternatiivne sündmuste käik):

5b. Subjekt kustutab kontrollis oleva tabeli kontrollitavate veergude nimekirjast veeru, milles olevaid andmeid ei soovita kontrollida.

1. Süsteem kustutab tabeli kirjeldusest kontrollitava veeru. OP 5.1

5c. Subjekt soovib võtta mingil põhjusel tabeli kontrollist maha.

1. Süsteem salvestab andmed. OP 4.1

5d. Subjekt muudab kontrollitava tabeli viimase kontrollitud rea primaarvõtme väärtust, et kontroll algaks mingis teisest algpunktist.

1. Süsteem salvestab andmed. OP 3.0

5e. Subjekt muudab kontrollitava tabeli nime või skeemi nime

1. Süsteem muudab kontrollitava tabeli andmeid. OP 4.1

Kasutusjuht: Vigade haldamine

Primaarne tegutseja: Haldur - (edaspidi subjekt).

Osapooled ja nende huvid:

- Subjekt: Soovib, et oleks õieti määratud, millised vead on parandatud ja millised mitte. Kui viga on parandatud kuid süsteemis pole see märgitud, siis teeb süsteem sellise rea kontrollimiseks asjatud tööd ning kontrollimine võtab rohkem aega kui peaks ning koormab süsteemi rohkem kui peaks.

Käivitav sündmus: Viga lahendati.

Eeltingimused: Tiražeerimise jälgimise vahend on leidnud vea ning selle kohta käivad andmed salvestatud.

Järeltingimused: On muudetud vea staatust.

Stsenaarium (tüüpiline sündmuste käik):

1. Subjekt annab korralduse alustada vigade haldamist.
2. Süsteem avab vigade halduse akna, kus kuvab leitud ja parandamata vead.
3. Subjekt valib huvipakkuva vea.
4. Süsteem avab valitud vea halduse akna.
5. Subjekt muudab vea staatust.
6. Süsteem salvestab tehtud muudatuse. OP 2.2
7. Subjekt sisestab vea märkuse väljale põhjenduse, miks vea staatust muudeti ning salvestab.
8. Süsteem salvestab tehtud muudatuse. OP 2.1

Kasutusjuht: Kontrollide jälgimine

Primaarne tegutseja: Haldur - (edaspidi subjekt).

Osapooled ja nende huvid:

- Subjekt: Soovib, et tiražeeritud read oleksid erinevates andmebaasides kooskõlas.

Käivitav sündmus: Subjekt soovib vaadata kindla kontrolli tulemust.

Eeltingimused: Tiražeerimise jälgimise vahend on käivitunud vähemalt ühe korra ning salvestanud kontrolli tulemuse.

Järeltingimused: Süsteem kuvab subjektile soovitud kontrolli andmed.

Stsenaarium (tüüpiline sündmuste käik):

1. Subjekt avab tiražeerimise jälgimise vahendi akna
2. Süsteem kuvab kontrolli vaate akna.
3. Subjekt valib kontrolli, mille andmeid soovib vaadata.
4. Süsteem kuvab järgmised kontrolli andmed: kontrolli identifikaatori, kontrolli alguse aja, kontrolli lõpu aja ning leitud vigade arvu. Leitud vigade arv suurem kui null, kuvab süsteem ka leitud vigade info: vea identifikaatori ja skeemi nime, tabeli nime ning võtmeväärtuse kus viga asub. Lisaks info vea staatuse muutmiste kohta: millal muudeti, kes muutis ning millisesse seisundisse viga viidi.
5. Subjekt tutvub kontrolli andmetega.

Laiendused (või alternatiivne sündmuste käik):

4b. Kui kontrollis leitud vigade arv on null, siis infot vigade kohta ei kuvata.

3.4 Mittefunktsionaalsed nõuded

- **Veebilehitsejate tugi** – Rakendus peab töötama tõrgeteta Äriregistris kasutusel olevates veebilehitsejates, milleks on 2014. aasta mai seisuga IE10 ning Mozilla Firefox 27.

- **Jõudlus** – Tiražeerimise tulemuste kooskõla jälgimise vahend peab olema suuteline toime tulema uute ridade kontrollimisega. Hinnanguliselt tiražeeritakse päevas keskmiselt 3000 rida. Kontroll peab olema suuteline kontrollima 3000 rida viie minuti jooksul.

- **Teisaldatavus** – Tiražeerimise jälgimise vahendit peab olema võimalikult lihtne vajadusel integreerida teiste süsteemidega.

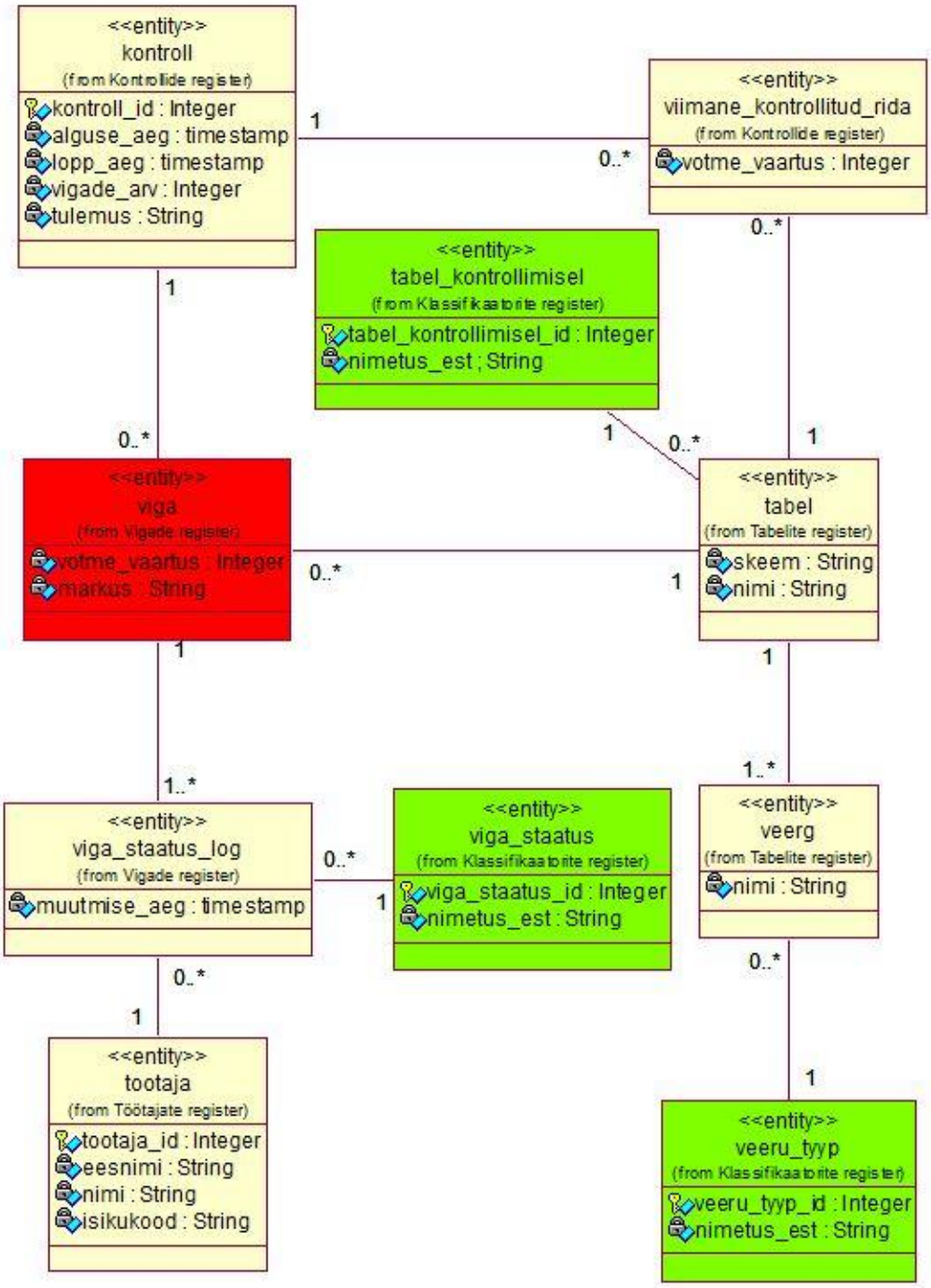
- **Dokumenteeritus** – Kavandatav süsteem peab olema dokumenteeritud, et kasutajatel oleks süsteemist parem arusaam ning võimalike edasiarenduste jaoks oleks arendajatel võimalik saada ülevaade süsteemist.

- **Kasutajaliides** – Peab jälgima Äriregistri menetlustarkvara disaini joont.

- **Turvalisus** – Kuna rakendus peab koostama etteantud sisendi alusel dünaamiliselt SQL lauseid, siis tuleb hoolitseda selle eest, et rakenduse kaudu ei saaks Äriregistri süsteemi rünnata SQL süstimise (*SQL injection*) rünnakuga. Selle üheks komponendiks on kasutajate ligipääsu piiramine rakendusele ning teiseks komponendiks programmide kirjutamine turvaliselt, et SQL süstimise rünnak poleks võimalik. [12]

3.5 Kontseptuaalne andmemudel

3.5.1 Olemi suhte diagramm



Joonis 2. Andmebaasi kontseptuaalset struktuuri kirjeldav olemi-suhte diagramm

3.5.2 Olemitüüpide definitsioonid

Kuigi andmebaas pole väga suur saab ka selle jaotada andmekesksteteks alamosadeks e registriteks ning Tabelis 1 on muuhulgas näha olemitüüpide jaotus erinevate registrite vahel.

Tabel 2. Olemitüüpide definitsioonide kirjeldused.

Olemitüübi nimi (aliased)	Kuuluvus registrisse	Definitsioon
kontroll	Kontrollide register	Iga kord, kui tiražeerimise jälgimise vahend käivitub ning töö lõpetab, salvestatakse andmed kontrolli kohta.
viimane_kontrollitud_rida	Kontrollide register	Igas kontrollitavas tabelis olev järjekoht, millest järgmise reaga tiražeerimise jälgimise vahend ridade kontrollimist alustab. Iga kontrolli lõpus salvestatakse info viimase kontrollitud rea kohta.
tabel	Tabelite register	Tiražeeritav baastabel, milles andmete kopeerimise tulemusi tiražeerimise jälgimise vahend kontrollib.
veerg	Tabelite register	Kirjeldab tehtud valikut, milliste veergude puhul tuleb igas jälgitavas tabelis kontrolli läbi viia. Kontrollima ei pea kõiki tabeli veerge, vaid ainult vajalikke veerge.
viga	Vigade register	Kindla tabeliga seotud viga kirjeldab asukohta, kus tabelis on kontroll leidnud ebakõla.
viga_staatus_log	Vigade register	Sisaldab informatsiooni, millal ning kes on vea staatust muutnud.
viga_staatus	Klassifikaatorite register	Võimalikud staatused millesse viga on kas lisamisega või muutmisega võimalik viia. Need staatused on: parandamata, lahendatud, mitte kontrollida
veeru_tyyp	Klassifikaatorite register	Kirjeldab, kas veeru puhul on tegemist tavaveeruga või primaarvõtme veeruga.
tabel_kontrollimisel	Klassifikaatorite register	Kirjeldab, kas kontrolli käivitamisel kontrollitakse ka seotud tabelit.
tootaja	Töötajate register	Menetlustarkvara kasutaja, mida tiražeerimise jälgimise vahend kasutab vea seisundi muutja tuvastamiseks.

3.5.3 Atribuutide definitsioonid

Tabel 3. Atribuutide definitsioonide kirjeldused.

Olemitüübi nimi	Atribuudi nimi	Atribuudi definitsioon	Näiteväärtus
kontroll	kontroll_id	Andmebaasis olevate kontrollide järjekorranumber, mis on igal kontrollil erinev.	1121
kontroll	alguse_aeg	Kuupäev ja kellaaeg, millal tiražeerimise jälgimise vahend tööprotsessi alustas.	2014-05-17 16:11:20
kontroll	lopp_aeg	Kuupäev ja kellaaeg, millal tiražeerimise jälgimise vahend tööprotsessi lõpetas.	2014-05-17 16:15:10
kontroll	vigade_arv	Kokku loetud arv, kui mitu viga on iga kontroll leidnud.	3
kontroll	tulemus	Tulemus, millele süsteem kirjete kontrollimine jõudis. Kui leidis mõne vea, siis väljastab tabeli skeemi ja nime ning viga sisaldava rea võtme väärtuse; kui jäi sisse mõni lahendamata viga, siis väljastab ka need lahendamata vead. Kui lahendamata vigu ei ole, ega uusi vigu ei leidnud, siis väljastab teate: read kooskõlas. Kui leiti vigu, siis saadeti sellise tekstiga e-meil halduritele.	’Leitud uued vead: 1. Skeemi nimi: ar Tabeli nimi: menetlused Võtme väärtus: 44232002’
viimane_kontrollitud_rida	votme_vaartus	Võtme väärtus, mis on iga kontrollitava tabeli viimase üle vaadatud rea primaarvõtme väärtus. Eeldatakse, et kõikides kontrollitavates tabelites on primaarvõti lihtvõti (sinna kuulub üks veerg) ja surrogaatvõti, mille väärtused moodustavad monotoonselt kasvava täisarvude jada.	44232022
viga	votme_vaartus	Tiražeerimise jälgimise vahendi poolt vigaseks määratud rea primaarvõtme väärtus. Eeldatakse, et kõikides kontrollitavates tabelites on primaarvõti lihtvõti (sinna kuulub üks veerg) ja surrogaatvõti, mille väärtused moodustavad monotoonselt kasvava täisarvude jada.	44232002
viga	markus	Märkus on kasutaja poolt vajadusel sisestatav selgitus, mis põhjusel on vea staatusega muudatus tehtud. Märkuse sisestab töötaja vastavalt enda soovile.	Vastavalt töörühma otsusele, viga jääb sisse.
tabel	skeem	Kontrollitavat tabelit sisaldava skeemi	Ar

Olemitüübi nimi	Atribuudi nimi	Atribuudi definitsioon	Näiteväärtus
		nimi, mida kasutatakse päringu lausete ülesehitamisel.	
tabel	nimi	Kontrollitava tabeli nimi, mida kasutatakse päringu lausete ülesehitamisel.	Menetlused
veerg	nimi	Tabeli veeru nimi, milles olevate väärtuste kooskõla on süsteemil vaja kontrollida.	ettevotte_id
veeru_tyyp	veeru_tyyp_id	Andmebaasis olevate veeru tüüpide järjekorra number, mis on igal veeru tüübil erinev.	1
veeru_tyyp	nimetus_est	Selgitab, kas veerg on tavaveerg või primaarvõtmesse kuuluv veerg.	Tavaveerg
tabel_kontrollimisel	tabel_kontrollimisel_id	Andmebaasis olevate tabel_kontrollimisel väärtuste järjekorra number, mis on igal kontrollimisel oleku puhul erinev.	0
tabel_kontrollimisel	nimetus_est	Selgitab, kas kontrolli käivitusel tehakse ridade võrdlemist ka seotud tabelis	Kontrollimisel
viga_staatus_log	muutmise_aeg	Kuupäeva ning kellaaeg, mis ajal on vea staatust muudetud.	24.05.2014 13:20
viga_staatus	Nimetus_est	Selgitab, millises olekus viga on. Tiražeerimise jälgimise vahendi puhul saab leitud viga automaatselt staatuse „parandamata“.	Parandamata
viga_staatus	viga_staatus_id	Andmebaasis olevate vigade staatuste järjekorranumber, mis on igal vea staatusel erinev.	
tootaja	tootaja_id	Andmebaasis olevate töötajate unikaalne identifikaator.	131
tootaja	eesnimi	"Lapsele pärast sündi (registreerimisel) pandav nimi, osa isikunimest. Eesnimi asetseb harilikult perekonnanime ees, harva järel (nt Ungari pruugis)." [7]	Raiko
tootaja	nimi	"Nimi, mis on isikul ühine teiste tema perekonna liikmetega". [7]	Tamm
tootaja	isikukood	Isiku isikukood on talle riigi, mille kodakondne isik on, poolt antud. "Oluline on teada, et isikukoodid ei kuulu delikaatsete isikuandmete alla. Isikukoodid on tavalised isikuandmed ja nende kasutamisele ei ole seatud rohkem piiranguid kui näiteks inimese nime või sünniaja kasutamisele." [8]	39105226012

3.6 Andmeoperatsioonide lepingud

OP 1.0 Uue kontrolli alustamine (alguse_aeg)

Eeltingimused:

- Peab olema sisestatud vähemalt üks kontrollitav tabel
- Kontrollitaval tabel peab olema seotud kontrollitavate veergudega.

Järeltingimused:

- On loodud uus Kontroll
- Kontroll.alguse_aeg = alguse_aeg

Kasutus kasutusjuhtude poolt: Kontrolli läbiviimine

OP 1.1 Kontrolli muutmine (kontrolli identifikaator, lopp_aeg, vigade arv, tulemus)

Eeltingimused:

- Kontroll on registreeritud

Järeltingimused:

- Kontroll.lopp_aeg = lopp_aeg
- Kontroll.vigade_arv = vigade_arv
- Kontroll.tulemus = tulemus

Kasutus kasutusjuhtude poolt: Kontrolli läbiviimine

OP 2.0 Uue vea sisestamine (võtme_väärtus, viide kontrollile, viide tabelile)

Eeltingimused:

- Kontroll on registreeritud
- Tabel on registreeritud
- võtme_väärtus on olemas kontrollitavas tabelis

Järeltingimused:

- Viga on registreeritud
- Viga.võtme_väärtus = võtme_väärtus
- Viga on seotud Kontroll’iga
- Viga on seotud Tabel’iga

Kasutus kasutusjuhtude poolt: Kontrolli läbiviimine

OP 2.1 Vea muutmine (vea identifikaator, markus)

Eeltingimused:

- Viga on registreeritud

Järeltingimused:

- Viga.markus = markus

Kasutus kasutusjuhtude poolt: Vigade haldamine

OP 2.2 Vea staatuse logi sisestamine (viide veale, muutmise_aeg, viide töötajale, viide viga_staatusele)

Eeltingimused:

- Viga on registreeritud
- Viga_staatus on registreeritud
- Tootaja on registreeritud

Järeltingimused:

- Viga_staatus_log on registreeritud
- Viga_staatus_log.muutmise_aeg = muutmise_aeg
- Viga_staatus_log on seotud Viga_staatus'ega
- Viga_staatus_log on seotud Viga'ga.
- Viga_staatus_log on seotud Tootaja'ga

Kasutus kasutusjuhtude poolt: Kontrolli läbiviimine, Vigade haldamine

OP 3.0 Viimase kontrollitud rea lisamine (võtme_väärtus, viide tabelile, viide kontrollile)

Eeltingimused:

- Kontroll on registreeritud
- Tabel on registreeritud
- võtme_väärtus on olemas kontrollitavas tabelis

Järeltingimused:

- Viimane_kontrollitud_rida on registreeritud

- Viimane_kontrollitud_rida.võtme_väärtus = võtme_väärtus
- Viimane_kontrollitud_rida on seotud Kontroll'iga
- Viimane_kontrollitud_rida on seotud Tabel'iga

Kasutus kasutusjuhtude poolt: Kontrollitavate tabelite kirjelduse administreerimine

OP 4.0 Uue kontrollitava tabeli sisestamine (tabeli_skeem, tabeli_nimi, viide tabel_kontrollimisele)

Eeltingimused:

- Andmebaasis eksisteerib skeem nimega tabeli_skeem
- Skeemis skeem eksisteerib tabel nimega tabeli_nimi
- Tabel_kontrollimisel on registreeritud

Järeltingimused:

- On loodud uus Tabel
- Tabel.skeem = tabeli_skeem
- Tabel.nimi = tabeli_nimi
- Tabel on seotud Tabel_kontrollimisel'iga

Kasutus kasutusjuhtude poolt: Uue kontrollitava tabeli lisamine

OP 4.1 Kontrollitava tabeli muutmine (tabeli identifikaator, tabeli_skeem, tabeli_nimi, viide tabel_kontrollimisele)

Eeltingimused:

- Tabel on registreeritud
- Andmebaasis eksisteerib skeem nimega tabeli_skeem
- Skeemis skeem eksisteerib tabel nimega tabeli_nimi
- Tabel_kontrollimisel on registreeritud

Järeltingimused:

- Tabel.skeem = tabeli_skeem
- Tabel.nimi = tabeli_nimi
- Tabel on seotud Tabel_kontrollimisel'iga

Kasutus kasutusjuhtude poolt: Kontrollitavate tabelite kirjelduse administreerimine, uue kontrollitava tabeli lisamine

OP 5.0 Kontrollitava veeru lisamine (veeru_nimi, viide veerg_tyyp'ile, viide tabelile)

Eeltingimused:

- Tabel on registreeritud
- Veerg veeru_nimi on olemas kontrollitavas tabelis

Järeltingimused:

- On loodud uus Veerg
- Veerg.nimi = veeru_nimi
- Veerg on seotud Veeru_tyyp'iga
- Veerg on seotud Tabel'iga

Kasutus kasutusjuhtude poolt: Kontrollitavate tabelite kirjelduse administreerimine, uue kontrollitava tabeli lisamine

OP 5.1 Kontrollitava veeru kustutamine (veeru identifikaator)

Eeltingimused:

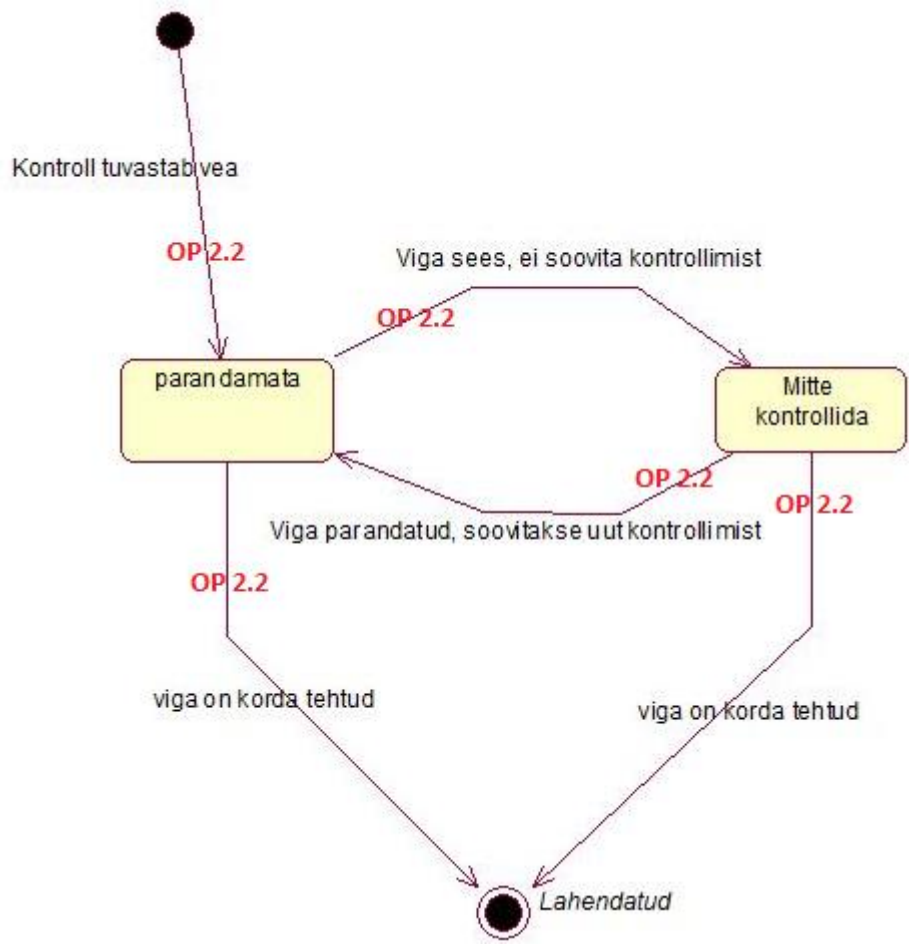
- Veerg on registreeritud

Järeltingimused:

- Veerg on kustutatud andmebaasist
- Veeru seos Tabel'iga on kustutatud
- Veeru seos Veeru_tyyp'iga on kustutatud.

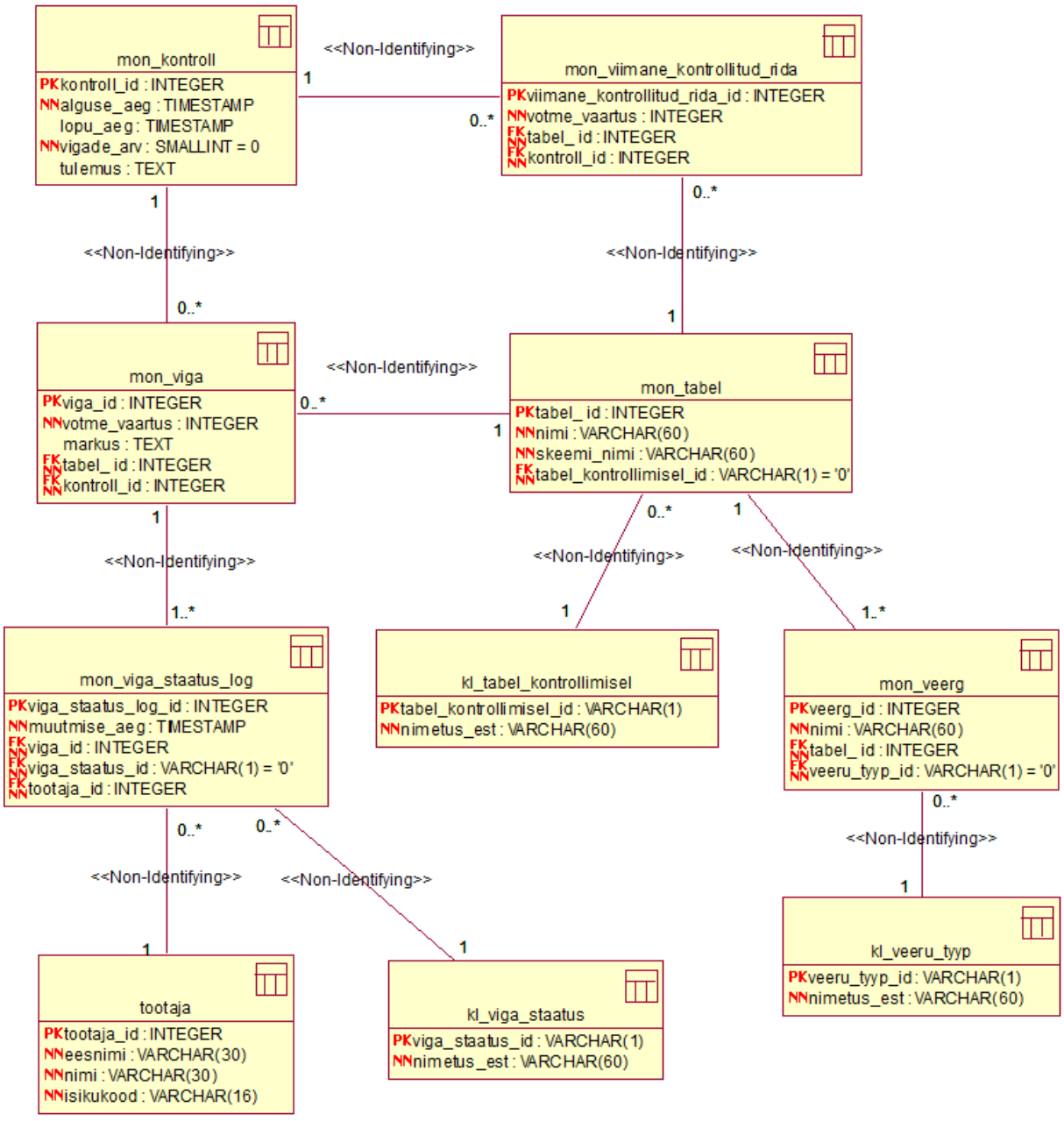
Kasutus kasutusjuhtude poolt: Kontrollitavate tabelite kirjelduse administreerimine

3.7 Vea seisundidiagramm



Joonis 3. Vea seisundidiagramm

3.8 Andmebaasi diagramm



Joonis 4. Loogilise disaini andmebaasi diagramm (UML notatsioonis).

3.8.1 Tabelite kirjeldus:

Eesliide *mon_* viitab monitoorimisele ja sellega tõstetakse esile kõik selle süsteemi tabelid peale klassifikaatorite tabelite, millel on eesliide *kl_*, mis viitab klassifikaatorile. Eesliited on kasulikud kuna aitavad tabeleid kasutajatele suunatud nimekirjades grupeerida. Selle süsteemi tabelid on skeemis, kus on ka muudeks haldustöödeks mõeldud süsteemide tabeleid.

Tabel 4. Tabelite kirjeldused.

Tabeli nimi	**Millise olemitüübi, atribuudi või seosetüübi põhjal on loodud?**
Mon_kontroll	Olemitüüp: kontroll
Mon_viimane_kontrollitud_rida	Olemitüüp: viimane_kontrollitud_rida
Mon_viga	Olemitüüp: viga
Mon_tabel	Olemitüüp: tabel
Tootaja	Olemitüüp: tootaja
Mon_viga_staatus_log	Olemitüüp: viga_staatus_log
Kl_viga_staatus	Olemitüüp: viga_staatus
Kl_tabel_kontrollimisel	Olemitüüp: tabel_kontrollimisel
Kl_veeru_tyyp	Olemitüüp: veeru_tyyp
Mon_veerg	Olemitüüp: veerg

3.8.2 Tabelite detailsed kirjeldused

Tabel 5. Tabelite detailsed kirjeldused.

Mon_kontroll						
Veeru nimi	**Tüüp/ domeen**	**Tüübi/ domeeni nimi**	**Pikkus**	**Võimalikud väärtused**	**Vaike-väärtus**	**Kohustuslik**
kontroll_id	T	suur täisarv		Unikaalsed väärtused. Automaatselt genereeritavad väärtused. Samm=1		jah
alguse_aeg	T	kuupäev/kell aaeg		Unikaalsed väärtused		jah
lopu_aeg	T	kuupäev/kell		Unikaalsed		ei

		aaeg		väärtused		
vigade_arv	T	väike täisarv			0	jah
tulemus	T	tekst				ei

Primary Key (kontroll_id)
Alternate Key (alguse_aeg)
Alternate Key (lopu_aeg)

Mon_viimane_kontrollitud_rida

Veeru nimi	Tüüp/ domeen	Tüübi/ domeeni nimi	Pikkus	Võimalikud väärtused	Vaike-väärtus	Kohustuslik
viimane_kontrollitud_rida_id	T	suur täisarv		Unikaalsed väärtused. Automaatselt genereeritavad väärtused. Samm=1		jah
votme_vaartus	T	suur täisarv				jah
tabel_id	T	suur täisarv				jah
kontroll_id	T	suur täisarv				ei

Primary Key (viimane_kontrollitud_rida_id)
Foreign Key (tabel_id) REFERENCES mon_tabel(tabel_id) ON UPDATE CASCADE
Foreign Key (kontroll_id) REFERENCES mon_kontroll(kontroll_id) ON UPDATE CASCADE

Mon_viga

Veeru nimi	Tüüp/ domeen	Tüübi/ domeeni nimi	Pikkus	Võimalikud väärtused	Vaike-väärtus	Kohustuslik
viga_id	T	suur täisarv		Unikaalsed väärtused. Automaatselt genereeritavad väärtused. Samm=1		jah
votme_vaartus	T	suur täisarv				jah
markus	T	tekst				ei
tabel_id	T	suur täisarv				jah
kontroll_id	T	suur täisarv				jah

Primary Key (viga_id)
Alternate Key (votme_vaartus, tabel_id)
Foreign Key (tabel_id) REFERENCES mon_tabel(tabel_id) ON UPDATE CASCADE
Foreign Key (kontroll_id) REFERENCES mon_kontroll(kontroll_id) ON UPDATE CASCADE

Mon_tabel

Veeru nimi	Tüüp/ domeen	Tüübi/ domeeni nimi	Pikkus	Võimalikud väärtused	Vaike-väärtus	Kohustuslik

tabel_id	T	suur täisarv		Unikaalsed väärtused. Automaatselt genereeritavad väärtused. Samm=1		jah
nimi	T	tekst	60			jah
skeemi_nimi	T	tekst	60			jah
Tabel_kontrollimisel_id	T	tekst	1		’0’	jah

Primary Key (kontroll_id)
Alternate Key (nimi, skeemi_nimi)
Foreign Key (tabel_kontrollimisel_id) REFERENCES kl_tabel_kontrollimisel(tabel_kontrollmisel_id) ON UPDATE CASCADE

Mon_veerg

Veeru nimi	**Tüüp/ domeen**	**Tüübi/ domeeni nimi**	**Pikkus**	**Võimalikud väärtused**	**Vaike-väärtus**	**Kohustuslik**
veerg_id	T	suur täisarv		Unikaalsed väärtused. Automaatselt genereeritavad väärtused. Samm=1		jah
nimi	T	tekst	60			jah
Veeru_tyyp_id	T	tekst	1		’0’	jah
tabel_id	T	suur täisarv				jah

Primary Key (veerg_id)
Alternate Key (nimi, tabel_id)
Foreign Key (tabel_id) REFERENCES mon_tabel(tabel_id) ON UPDATE CASCADE
Foreign Key (veeru_tyyp_id) REFERENCES kl_veeru_tyyp(veeru_tyyp_id) ON UPDATE CASCADE

Mon_viga_staatus_log

Veeru nimi	**Tüüp/ domeen**	**Tüübi/ domeeni nimi**	**Pikkus**	**Võimalikud väärtused**	**Vaike-väärtus**	**Kohustuslik**
Viga_staatus_log_id	T	suur täisarv		Unikaalsed väärtused. Automaatselt genereeritavad väärtused. Samm=1		jah
Muutmise_aeg	T	kuupäev/kell aaeg				jah
Viga_id	T	suur täisarv				jah
Viga_staatus_id	T	tekst	1		’0’	jah
Tootaja_id	T	suur täisarv				jah

Primary Key (viga_staatus_log_id)
Foreign Key (viga_id) REFERENCES mon_viga(viga_id) ON UPDATE CASCADE
Foreign Key (viga_staatus_id) REFERENCES kl_viga_staatus(viga_staatus_id) ON

UPDATE CASCADE
Foreign Key (tootaja_id) REFERENCES tootaja(tootaja_id) ON UPDATE CASCADE

Kl_viga_staatus

Veeru nimi	Tüüp/ domeen	Tüübi/ domeeni nimi	Pikkus	Võimalikud väärtused	Vaike-väärtus	Kohustuslik
Viga_staatus_id	T	tekst	1			jah
Nimetus_est	T	tekst	60			jah

Primary Key (viga_staatus_id)
Alternate Key (nimetus_est)

Kl_tabel_kontrollimisel

Veeru nimi	Tüüp/ domeen	Tüübi/ domeeni nimi	Pikkus	Võimalikud väärtused	Vaike-väärtus	Kohustuslik
Tabel_kontrollimisel_id	T	tekst	1			jah
Nimetus_est	T	tekst	60			jah

Primary Key (tabel_kontrollimisel_id)
Alternate Key (nimetus_est)

Kl_veeru_tyyp

Veeru nimi	Tüüp/ domeen	Tüübi/ domeeni nimi	Pikkus	Võimalikud väärtused	Vaike-väärtus	Kohustuslik
Veeru_tyyp_id	T	tekst	1		’0’	jah
Nimetus_est	T	tekst	60			jah

Primary Key (veeru_tyyp_id)
Alternate Key (nimetus_est)

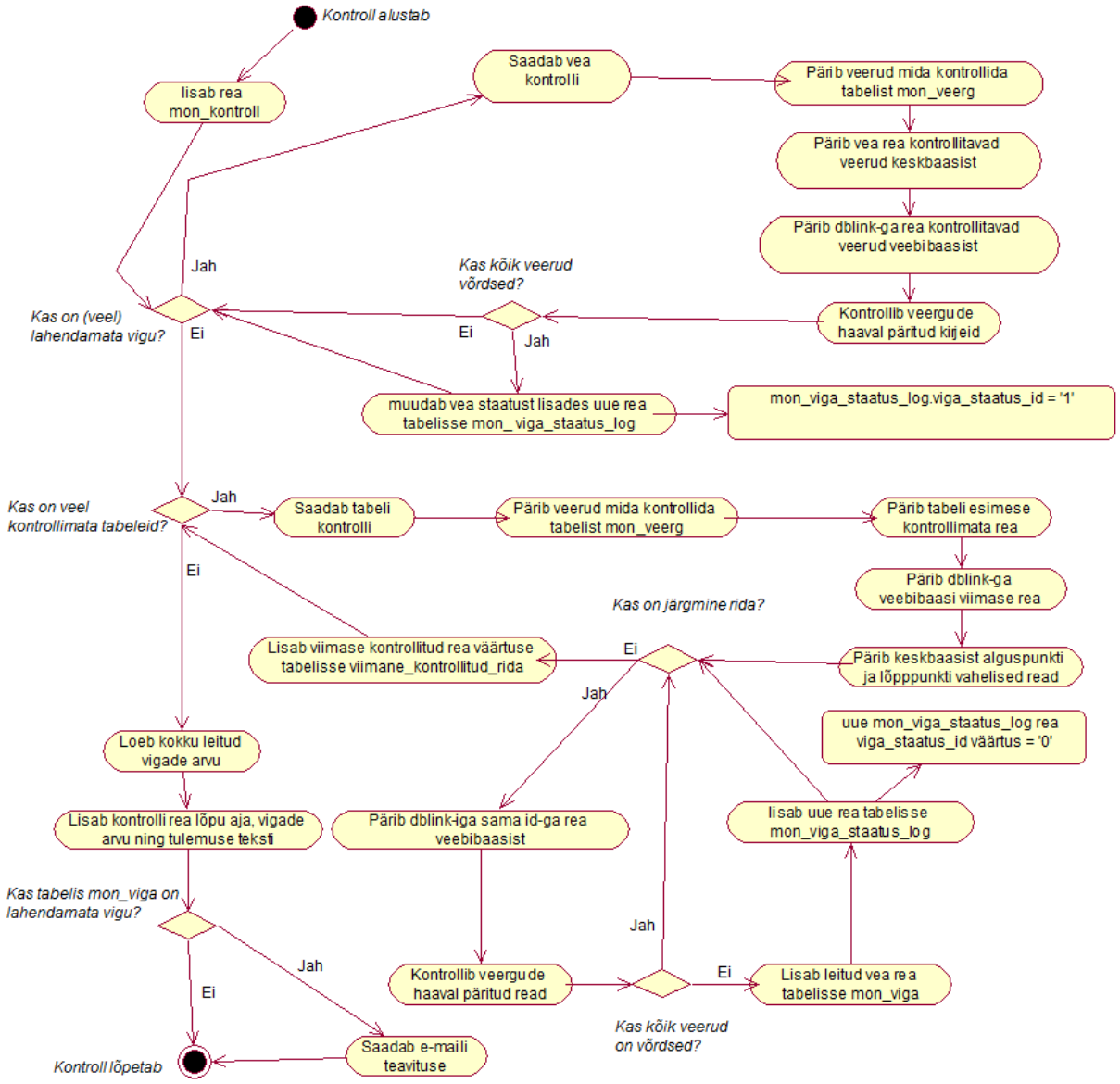
tootaja

Veeru nimi	Tüüp/ domeen	Tüübi/ domeeni nimi	Pikkus	Võimalikud väärtused	Vaike-väärtus	Kohustuslik
Tootaja_id	T	suur täisarv		Unikaalsed väärtused. Automaatselt genereeritavad väärtused. Samm=1		jah
Eesnimi	T	tekst	30			jah
Nimi	T	tekst	30			jah
Isikukood	T	tekst	16			jah

Primary Key (Tootaja_id)

3.9 Kontrolli algoritmi tegevusdiagramm

Joonis 3 esitab tegevusdiagrammi kasutades kontrolli algoritmi, mille juures viidatakse juba konkreetsetele andmebaasi tabelitele, mida algoritm oma töö korraldamiseks sisemiselt kasutab. Kontrolli alustamise eeltingimus on, et kontroll juba parajasti ei toimu (ei saa käivitada paralleelseid kontrollimise protsesse).



Joonis 5. Kontrolli algoritmi tegevusdiagramm.

Kontrolli algoritm lisab esmalt tabelisse *mon_kontroll* kontrolli alustamise aja. Seejärel pärib vigade hulgast, kas on mõni viga, mille staatus on „parandamata“ (teiste sõnadega, viga on lahendamata). Juhul, kui saab vastuseks, et leidub vigaseid ridu, siis saadab vea kontrollimisele, mille eesmärgiks on uurida, kas viga on vahepeal parandatud.

Kontrollimisel koostatakse kõigepealt vea kohta käiv päring, millega küsitakse andmeid ainult veaga seotud tabelis olevatest kontrollitavatest veergudest. Päring koostatud, küsitakse rida keskbaasist ning kasutades *dblinki* küsitakse sama rida veebibaasist. Seejärel kontrollitakse leitud read veergude haaval läbi ning kui read on võrdsed, siis lisatakse vea staatuse logisse staatuse muutmise kohta käiv info. Kui tulemuseks on, et viga on endiselt olemas, siis liigutakse edasi järgmist vigast rida kontrollima.

Kui kõik vigased read on üle vaadatud või ei ole registreeritud ühtegi parandamata viga, liigutakse edasi kontrollitavate tabelite uusi ridu kontrollima. Otsitakse kõik kontrollis olevad tabelid, mis saadetakse ükshaaval kontrolli. Kontrolli käigus otsitakse tabeli veerud, mida on vaja kontrollida, seejärel otsitakse rida millest kontrolli alustada. Viide esimesele reale leitakse tabelist *mon_viimane_kontrollitud_rida*, kuhu on salvestatud eelmise kontrolli käigus viimasena kontrollitud rea primaarvõtme väärtus. Seejärel küsitakse *dblinki* kasutades viimane rida veebibaasist. Alguspunkt on viimati vaadatud rea primaarvõtme väärtus + 1 ning lõpp-punkt on veebibaasis olev kõige suurem rea primaarvõtme väärtus. Alguspunkt ning lõpp-punkt käes, küsitakse keskbaasi tabelist kõik read, milles olev primaarvõtme väärtus jääb leitud vahemikku, piiripunktid kaasaarvatud.

Iga saadud rea kohta otsitakse *dblink'i* kasutades sama primaarvõtme väärtusega rida veebibaasist ning asutakse veergude kaupa keskbaasi rida ja veebibaasi rida omavahel võrdlema. Kui read on võrdsed, siis liigutakse edasi järgmist rida võrdlema, kuid erinevuse leidmisel lisatakse leitud vea informatsioon tabelisse *mon_viga*. Lisaks lisatakse vea staatuse kohta uus rida tabelisse *mon_viga_staatus_log*, milles viga_staatus_id = '0', ehk vea staatuseks saab „parandamata". Vea informatsioon talletatud, liigutakse edasi järgmist rida kontrollima. Kõik leitud read võrreldud, asub kontroll järgmist tabelit kontrollima.

Kui kõik kontrollitavad tabelid on läbi käidud, siis loetakse kokku selle kontrolli käigus registreeritud vigade arv ning uuendatakse kontrolli kohta käivat rida tabelis *mon_kontroll*, kuhu lisatakse kontrolli lõppemise aeg, leitud vigade arv ning tulemuse sõnaline kirjeldus. See kirjeldab, kas leiti uusi vigu ning leidub lahendamata vigu.

Lahendamata vigade olemasolu korral saadetakse konfiguratsioonifaili seatud e-meili aadressile teavitus, et andmebaasi tiražeerimises esineb ebakõla ning kontrolliprotseduur lõpetab töö. Kui lahendamata vigu ei eksisteeri, siis sellisel juhul e-meili teavitust välja ei saadeta, sest ainult vea leidmise korral saadetud e-meil on saajale erakorraline ning paremini esilekerkiv, võrreldes olukorraga, kui iga kontrolli lõpus saadetaks e-meil.

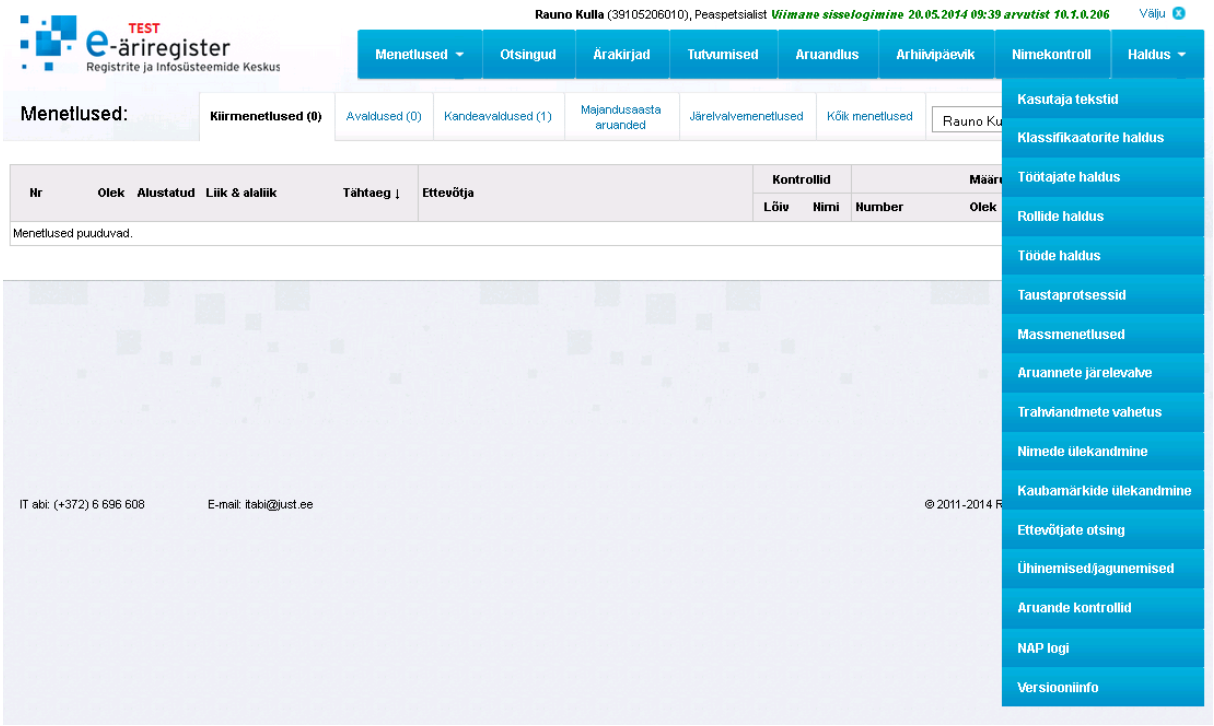
4. Kasutajaliidese kavand

Järgnevas peatükis on kirjeldatud loodavat kasutajaliidest piltide ning selgituste abil. Kuna töö on koostatud Äriregistri näitel, hakkab kasutajaliides olema osa Äriregistri menetlustarkvarast. Kasutajaliidese disainimisel on jälgitud menetlustarkvara disaini joont. Kasutajaliidesesse sisselogimist ei ole vaja realiseerida, kuna menetlustarkvaras tiražeerimise jälgimise vahendi avamise hetkeks peab olema kasutaja ennast autentinud Eesti ID-kaardi või Eesti mobiil-ID vahendusel.

Nende ridade kirjutamise hetkeks on realiseeritud kontrollimise süsteemi andmebaasi tabelid ning kontrolli teostav taustaprotseduur. Selles peatükis esitatav kasutajaliides pole töö kirjutamise ajal veel valminud, kuid plaan on see realiseerida lähemal ajal. Töös esitatud kasutajaliidese pildid on tehtud MS Paint joonistusvahendiga.

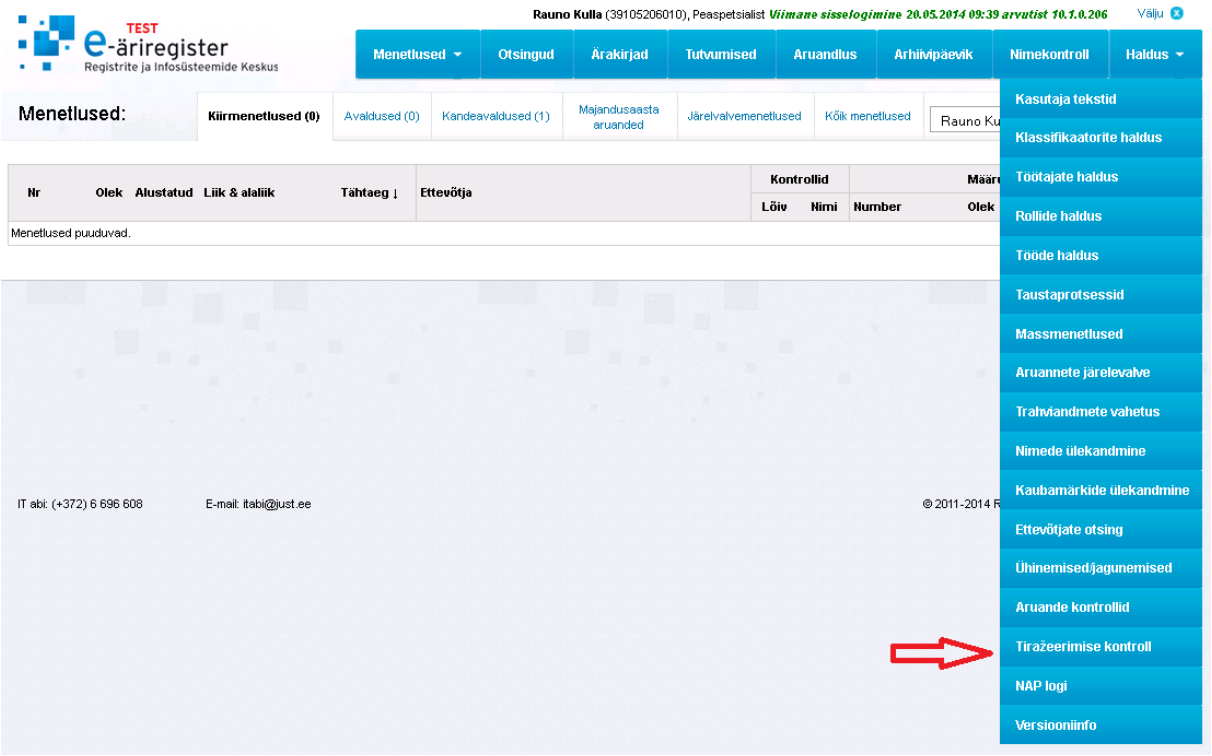
4.1 Tiražeerimise kooskõla jälgimise vahendi avamine

Tiražeerimise jälgimise vahendit on võimalik avada ainult menetlustarkvara kasutajal, kes omab halduri õiguseid. Joonisel 5 on toodud ekraanipilt töö kirjutamise ajal kasutusel olevast menetlustarkvara kasutajaliidese avaekraanilt avanevast halduse sakist.



Joonis 6. Praegune menetlustarkvara halduse sakk.

Tiražeerimise kooskõla jälgimise vahendit peab saama avada halduse saki kaudu, vajutades ’Tiražeerimise kontroll’ peale nagu näidatud joonisel 6.



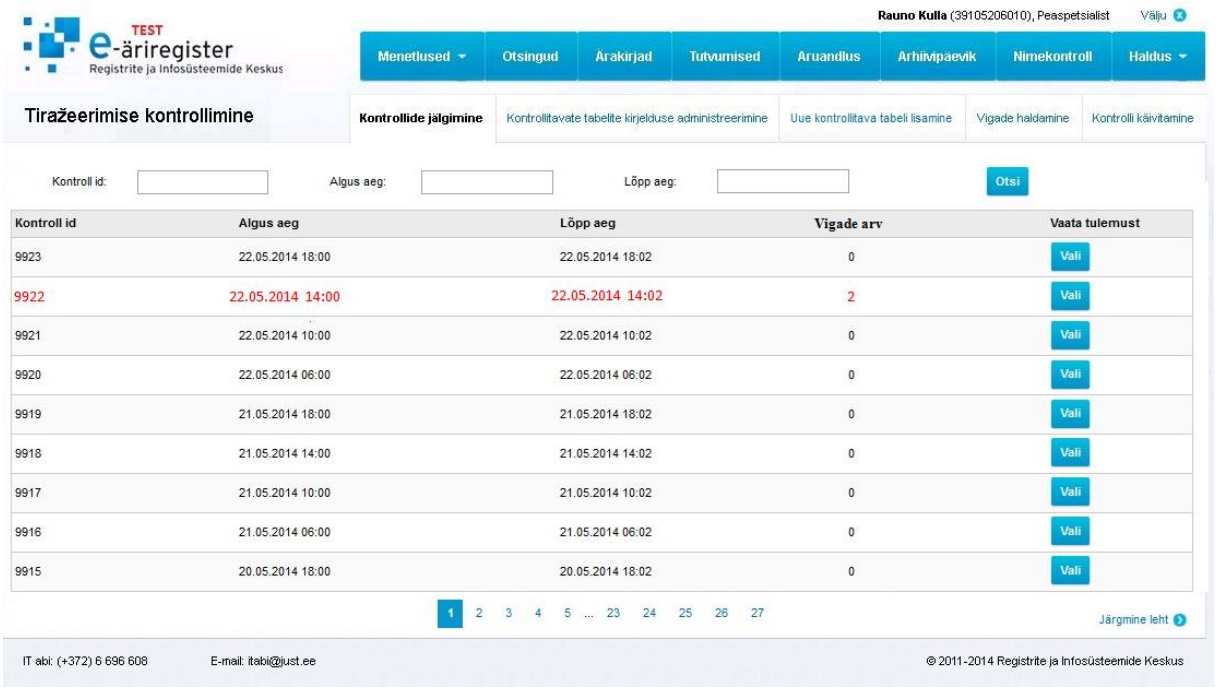
Joonis 7. Planeeritav menetlustarkvara halduse saki muudatus.

4.2 Kontrollide jälgimine

Joonisel 6 näidatud ’Tiražeerimise kontroll’ nupule vajutades avaneb tiražeerimise kontrollimise vaade, mis on näidatud joonisel 7. Vaates on võimalik otsida kontrolli selle numbrilise identifikaatori järgi ning kontrolli alustamise ja lõpetamise aja järgi.

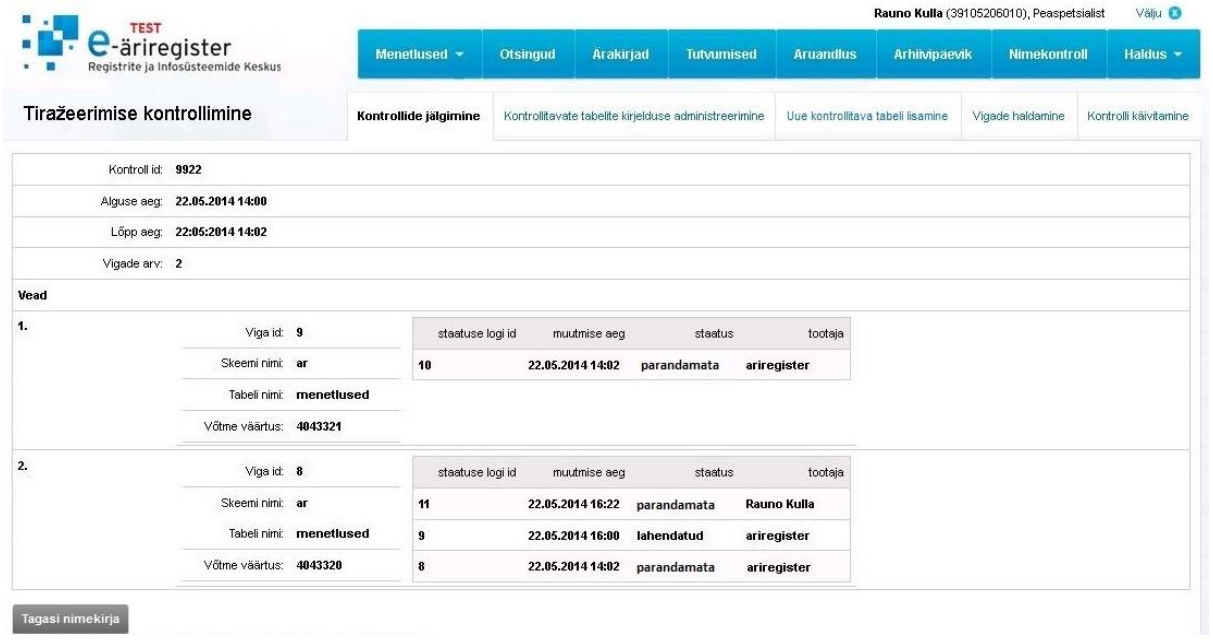
Algselt on kontrollid järjestatud identifikaatori kahanemise järjekorras, kuid saab järjestada ka vigade arvu järgi, vajutades veeru nime ’Vigade arv’ peale. Korraga kuvatakse üheksa viimast kontrolli ning kui kontrolle on salvestatud rohkem kui üheksa, siis eelmisi saab vaadata liikudes lehe jaluses olevatel numbritel.

Juhul, kui kontroll on leidnud vea ehk kontrolli vigade arv on suurem kui 0, siis kuvatakse kontrolli kohta käiv rida märgatavuse parandamiseks punases kirjas ning suuremalt kui tavakiri. Joonisel 7 on selliselt kujutatud rida kontroll id-ga 9922.



Joonis 8. Tiražeerimise kontrollide loetelu vaade.

Vajutades mõnel real nuppu ’Vali’, kuvatakse selle rea kohta käiva kontrolli tulemus. Kontrolli tulemuse vaade on näidatud joonisel 8.



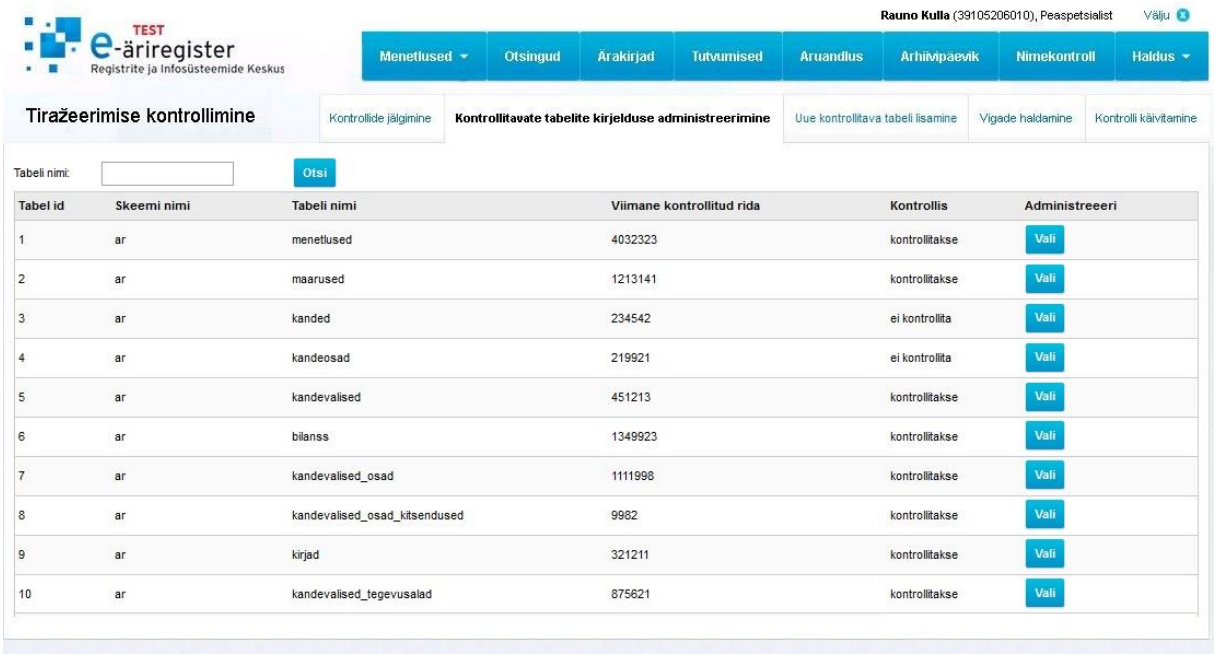
Joonis 9. Kontrolli tulemuse andmete vaatamise vaade.

Kontrolli tulemuse vaates kuvatakse andmed vigade kohta, mida valitud kontroll leidis ja registreeris. Samuti kuvatakse info vea staatuse logi kohta, kust on võimalik kindlaks teha, kes on vea staatust muutnud. Antud vaates ei ole muudatusi teha võimalik, kogu info on ainult tutvumiseks.

Juhul, kui valitud kontroll vigu ei tuvastanud, siis kuvatakse sektsioonis ’Vead’ tekst: „Kontroll vigu ei tuvastanud“.

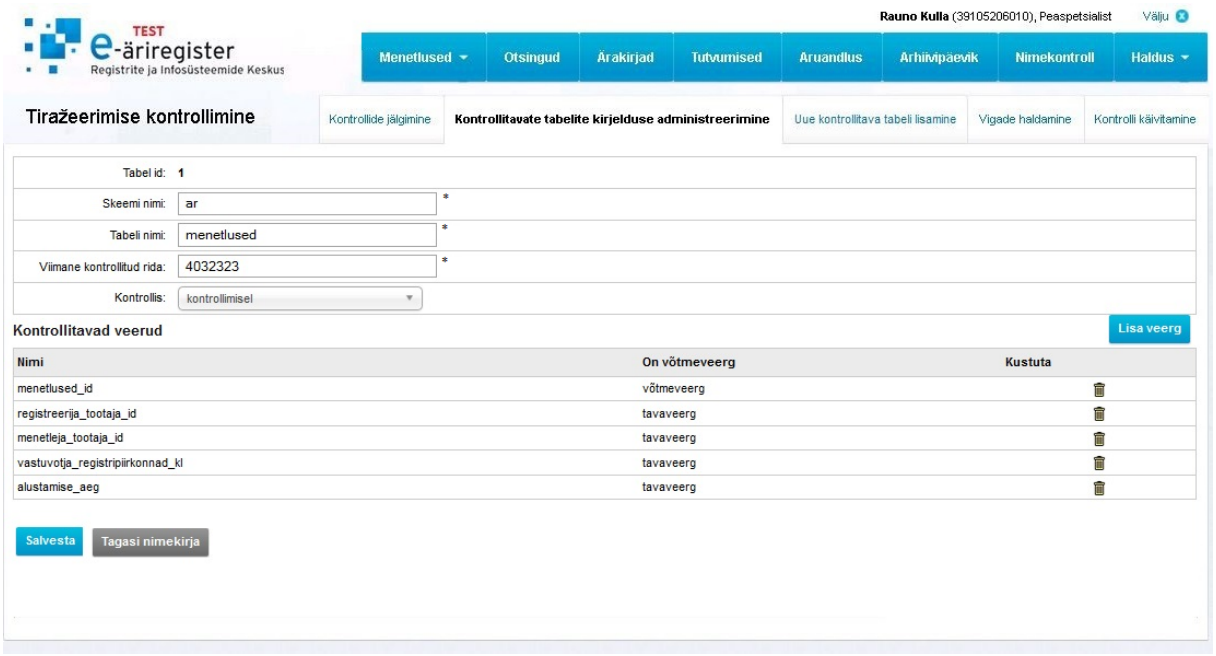
4.3 Kontrollitavate tabelite kirjelduse administreerimine

Valides tiražeerimise kontrollimise sakkide hulgast ’Kontrollitavate tabelite kirjelduse administreerimine’, avaneb kontrollis olevate tabelite loetelu vaade. Vaade on kujutatud joonisel 9.



Joonis 10. Kontrollitavate tabelite loetelu vaade.

Nimekirjast on võimalik otsida kindlat rida tabeli nime järgi. Kui vaade avatakse, on read sorteeritud tabeli numbrilise identifikaatori järgi, kuid vajutades veerunimedele: ’Skeemi nimi’, ’Tabeli nimi’ või ’Kontrollis’, siis järjestatakse read ümber valitud veeru järgi. Vajutades valitud real nupule ’Vali’, avatakse kontrollitava tabeli kirjelduse administreerimise vaade, mis on kujutatud joonisel 10.



Joonis 11. Kontrollitava tabeli kirjelduse administreerimise vaade.

Antud vaates on halduril võimalik muuta kontrollitava tabeli skeemi nime, tabeli nime ja viimase kontrollitud rea primaarvõtme väärtust. Lisaks on võimalik eemaldada tabel kontrollist või lisada tabel kontrolli. Kontrollist eemaldamine tähendab, et kontrolli käivitusel ei jälgita antud tabelit. Salvestamise nupule vajutades peavad kasutajal olema täidetud kõik nõutavad väljad, milleks on skeemi nimi, tabeli nimi ning viimane kontrollitud rida. Kui mõni nõutav andmesisestusväli on jäetud tühjaks, siis väljastab süsteem veateate „Täitke nõutud väljad!“.

Samuti on tabeli kirjelduse administreerimise vaates võimalik hallata kontrollitavaid veerge. Veerge on võimalik kas kustutada või lisada. Kui mõne veeru andmeid on vaja muuta, siis tuleb selleks veeru andmed kustutada, vajutades selle veeru real prügikasti märgile. Seejärel saab lisada õigete andmetega veeru, vajutades nupule ’Lisa veerg’.

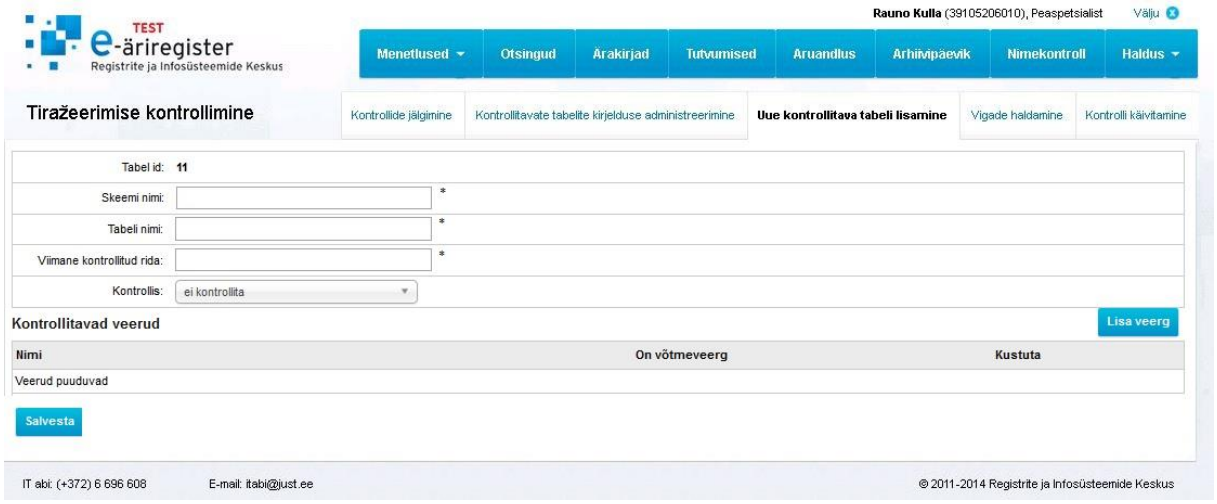
’Lisa veerg’ nupule vajutades avaneb hüpikaken, mis on kujutatud joonisel 11.

Joonis 12. Uue kontrollitava veeru lisamise hüpikakna vaade.

Täites ära veeru nime ning määrates, kas lisatava veeru puhul on tegu võtmeveeruga või mitte ning seejärel salvestades, ilmub kontrollitavate veergude nimekirja uus veerg.

4.4 Uue kontrollitava tabeli lisamine

Valides tiražeerimise kontrollimise sakkide hulgast ’Uue kontrollitava tabeli lisamine’, avaneb uue kontrollitava tabeli lisamise vaade. Vaade on kujutatud joonisel 12.



Joonis 13. Uue kontrollitava tabeli lisamise vaade.

Vaates on nõutud täidetavad väljad ’Skeemi nimi’, ’Tabeli nimi’ ja ’Viimane kontrollitud rida’. Väljade väärtused on vaja kasutajal tekstina sisestada. Skeemi nime ning tabeli nime salvestamise järgselt kontrollib andmebaasi triger, kas kasutaja sisestatud tabel on andmebaasis olemas, üritades küsida kasutaja sisestatud tabelist ühe rea. Kui tulemuseks on viga, siis kuvatakse kasutajale veateade: „Kontrollige skeemi või tabeli nime!“

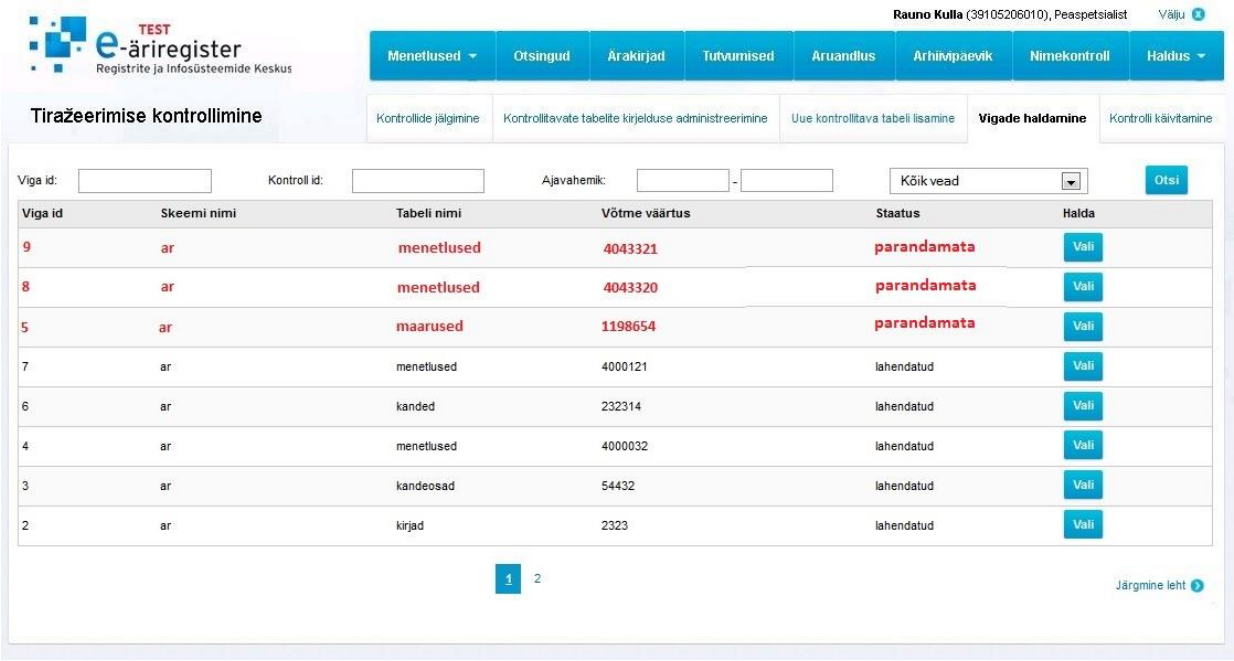
Kaalumise all oli ka variant otsida andmebaasi *information_schema* vaadete vahendusel kõik skeemide ning tabelite nimed ning lasta kasutajal valida andmebaasis salvestatud tabelite ning skeemide nimede hulgast kontrollitav skeem ja tabel. Varianti ei valitud, kuna suuremates skeemides on Äriregistri andmebaasis kuni 180 tabelit ning rippmenüüst soovitava tabeli valimine oleks 180 hulgast keeruline.

Lisaks peab kasutaja sisestama kõik kontrolli vajavad veerud. Selleks vajutab kasutaja nupule ’Lisa veerg’, mis avab hüpikakna, mida näidatakse joonisel 11. Uue kontrollitava veeru salvestamisel üritab andmebaasi triger jällegi päringut lisatud veeru põhjal. Kui tulemuseks on viga, siis kuvatakse hüpikakna päises veateade, mis palub kontrollida veeru nime sisestust.

Vaikimisi on määratud, et tabelit ei kontrollida. Kasutajal on võimalus ’Kontrollis’ välja muutmisega lisada tabel kontrolli ning eemaldada tabel kontrollist.

4.5 Vigade haldamine

Valides tiražeerimise kontrollimise sakkide hulgast ’Vigade haldamine’, avaneb registreeritud vigade loetelu vaade. Vaade on kujutatud joonisel 13.



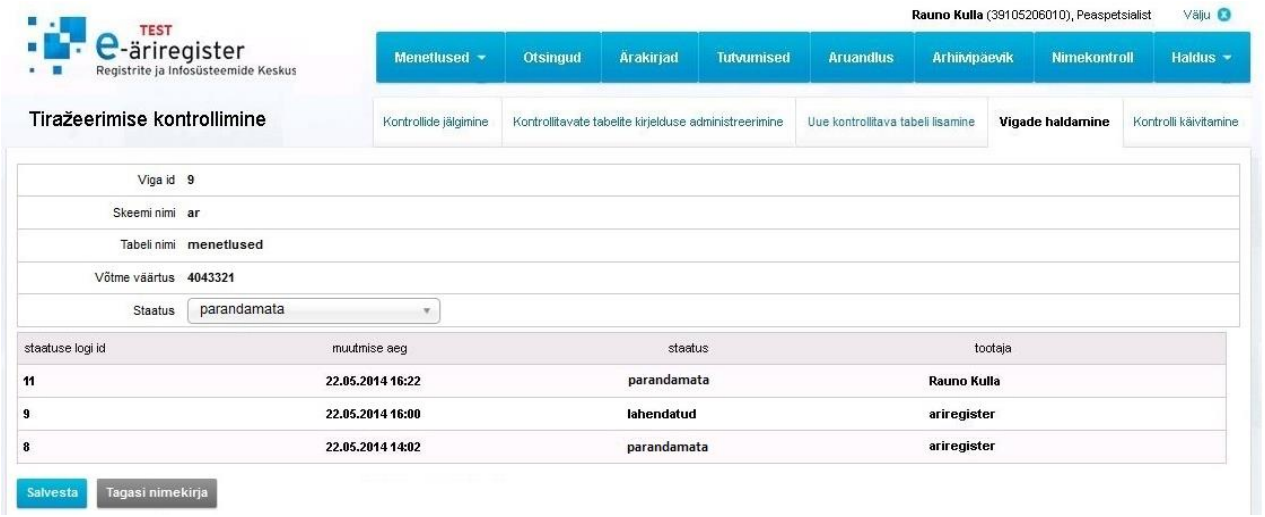
Joonis 14. Vigade loetelu vaade.

Kasutajal on võimalik vigu otsida vea numbrilise identifikaatori järgi, kontrolli numbrilise identifikaatori järgi ning vea avastamise ajavahemiku järgi. Samuti on võimalik staatuse järgi valida, milliseid vigu tulemuses kuvatakse. Vaikimisi kuvatakse kõik leitud vead.

Algselt on vead järjestatud staatuse järjekorras, mis tagab, et esmalt kuvatakse parandamata read. Seejärel on vead järjestatud vea numbrilise identifikaatori järgi kahanevalt. Korraga kuvatakse kaheksa viimast viga ning kui vigu on salvestatud rohkem kui kaheksa, siis eelmisi saab vaadata liikudes lehe jaluses olevatel numbritel.

Juhul, kui vea staatus on „parandamata“ ehk viga on endiselt probleemiks, siis kuvatakse märgatavuse suurendamiseks vea kohta käiv rida punases kirjas ning suuremalt kui tavakiri. Joonisel 13 on selliselt kujutatud read vea identifikaatoriga 9, 8 ning 5.

Vajutades vea real veerus halda nupule ’Vali’, avaneb vea halduse vaade. Vaade on kujutatud joonisel 14.



Joonis 15. Vea halduse vaade.

Vaates kuvatakse kasutajale info vea kohta – millises skeemis olevas tabelis viga leiti ning viga sisaldava rea primaarvõtme väärtus. Loodav lahendus ei võimalda kasutajal

süsteemist teada saada vea täpset sisu (millistele veergudele vastavates väljades pole väärtused kooskõlas). Täpse vea sisu leidmiseks peab haldur tegema päringud.

Samuti kuvatakse kasutajale logi, kust on näha, milline töötaja on vea staatust muutnud. Kui töötaja nimeks on „äriregister“, siis on staatuse muudatuse teinud süsteem. Logi tabeli read on väljundis järjestatud logi numbrilise identifikaatori järgi kahanevalt ning kehtivaks staatuseks on kõige suurema identifikaatoriga, ehk viimasena lisatud rea staatus.

Kui kasutaja muudab vea staatust ning salvestab, lisatakse staatuse logisse uus rida. Uues reas märgitakse töötajaks muudatuse teinud kasutaja.

4.6 Kontrolli käivitamine

Valides tiražeerimise kontrollimise sakkide hulgast ’Kontrolli käivitamine’, avaneb kontrolli käivitamise hüpikaken, mis on näidatud joonisel 15.

Joonis 16. Tiražeerimise kontrolli käivitamise hüpikakna vaade.

Vajutades nupule ’Käivita’ alustab tiražeerimise jälgimise vahend tööd. Seejärel suunatakse kasutaja tagasi tiražeerimise kontrollide loetelu vaatesse, mis on kujutatud joonisel 7, kuhu on lisatud uus käimasoleva kontrolli kohta käiv rida, millel on kontrolli numbrilise identifikaatori väärtus ning algus aeg. Kontroll on töö lõpetanud, kui uue kontrolli kohta käivale reale on lisatud lõpu aeg.

5. Testimine

Antud peatükis testitakse realiseeritud taustaprotsessi tööd. Testimisel mängitakse läbi kolm võimalikku stsenaariumit.

1. Mittevõrdsete ridade leidmine kolme tuhande kontrollitud rea hulgast.
2. Vea staatuse muutmine lahendatuks pärast raporteeritud ridade võrdsustamist.
3. Veebibaasist puuduolevate ridade leidmine kuue tuhande kontrollitud rea hulgast.

Igas kontrollis võrdleb taustaprotsess kokku kolm tuhat rida, kuna see on hinnanguliselt keskmine Äriregistris päevas tiražeeritavate ridade arv ning kui taustaprotsess tuleb toime korraga kolme tuhande rea võrdlemisega, siis käivitades seda kolm korda päevas, tuleb taustaprotsess toime ka ekstreemolukordadega kui vaja näiteks kontrollida kogu päeva jooksul tiražeeritud read. Selle põhjuseks võib olla, et taustaprotsess ei käivitunud tehnilistel põhjustel ettenähtud aegadel.

Testimiseks on piisav kui jälgitavate tabelite hulka lisatakse kaks tabelit. Seda põhjusel, et kui taustaprotsessi algoritm oskab kahe tabeli korral ehitada ridade pärimise lauseid, siis oskab algoritm ehitada lauseid ka juhul kui kontrollis olevate tabelite arv on suurem kui kaks.

5.1 Ettevalmistus

Kogu kontroll algoritmi testimine viiakse läbi Äriregistri testandmebaasis salvestatud andmetega, mis on toodangukeskkonna andmebaasist 2014. aasta alguses tehtud koopia. Veebibaasis tabelis *menetlused* viimasena lisatud rea primaarvõtme väärtus on töö kirjutamise ajal 40652600. Algselt võetakse kontrolli read vahemikus 40649600 – 40651100 ning seejärel read vahemikus 40651101 – 40652600. Tabelis *maarused* on töö kirjutamise ajal viimasena lisatud rea primaarvõtme väärtus 9000410200. Algselt võetakse kontrolli read vahemikus 9000407200 – 9000408700 ning seejärel read vahemikus 9000408701 – 9000410200.

Toodud vahemikesse kuuluvad read on vaja mõlemas andmebaasis kopeerida skeemis *haldurid* olevatesse tabelitesse: *ar.menetlused* tabeli read kopeeritakse tabelisse *haldurid.testime_menetluseid* ning *ar.maarused* tabeli read kopeeritakse tabelisse *haldurid.testime_maaruseid*. Kopeerimine teise tabelisse on vajalik testi käigus ridade muutmise võimaluse olemasolu tagamiseks. *Ar* nimelises skeemis toimub andmete tiražeerimine nii kiiresti, et kui viia sisse mõne reaga muudatus, siis jõuab see kontrolli käivitamise ajaks kindlasti ka veebibaasi. Haldurid nimelises skeemis tiražeerimist ei toimu, mis tähendab, et kui muuta kindlat rida keskbaasis, siis muudatus ei kandu automaatselt veebibaasi.

Lisaks lisan tabelisse *mon_viimane_kontrollitud_rida* mõlema kontrollitava tabeli kohta pidepunkti, kust algoritm teab kontrollimist alustada. Tabelis *menetlused* on see võtme väärtus 40649600 ning tabelis *maarused* on võtme väärtus 9000407200.

5.2 Esimene stsenaarium

Esialgu kontrollitakse terved seatud vahemikud, et olla kindel, kas vahemiku read on mõlemas andmebaasis võrdsed. Kahe minutiga on tekkinud kontrolli reale tulemus: „Read kontrollitud. Vigu ei ole.“ Seega on vaja käsitsi teha ainult ühes andmebaasis seatud ridade vahemikes mõned andmemuudatused, et kontrollil oleks võimalik neid ebakõlasid leida. Lisaks kustutatakse andmebaasist käivitatud kontrolli lisatud viimase kontrollitud rea primaarvõtme väärtus, et järgmine kontroll asuks uuesti esimesse vahemikku kuuluvaid ridu võrdlema.

Teen veebibaasis tabelis *haldurid.testime_menetluseid* esimeses ridade vahemikus kaks muudatust järgmiste menetlus numbritega ridades: 40651111, 40651932. Tabelis *haldurid.testime_maaruseid* tehakse kaks muudatust järgmiste määruste numbritega ridades: 9000407312, 9000407882. Seejärel käivitatakse kontroll.

```
SELECT * FROM haldurid.mon_kontroll WHERE kontroll_id = 2;
```

Ülal toodud lausega saame info läbiviidud kontrolli kohta.

Tabel 6. Esimese stsenaariumi kontrolli andmed

kontroll_id	alguse_aeg	lopu_aeg	vigade_arv	tulemus
2	26.05.2014 14:15	26.05.2014 14:17	4	’Leitud uued vead…’

LISA 1 oleva päringuga saame tulemuseks kõik registreeritud vead; skeemi ja tabeli nime, kus viga esineb ning vea viimasena lisatud staatuse.

Tabel 7. Esimese stsenaariumi vigade andmed.

viga_id	skeemi_nimi	nimi	votme_vaartus	staatus
1	haldurid	testime_menetluseid	40651111	parandamata
2	haldurid	testime_menetluseid	40651932	parandamata
3	haldurid	testime_maaruseid	9000407312	parandamata
4	haldurid	testime_maaruseid	9000407882	parandamata

Taustaprotsess leidis kahe minutiga 3000 rida kontrollides üles kõik neli testija poolt tehtud viga. Seega algoritm töötab esimese stsenaariumi puhul väga hästi.

5.3 Teine stsenaarium

Parandame kõik neli veebibaasis tehtud muudatust nii, et keskbaasi ning veebibaasi esialgselt muudetud read oleksid jälle võrdsed. Lisame tehtud tabelitesse *haldurid.testime_menetluseid* ja *haldurid.testime_maaruseid* teise mainitud ridade vahemiku ning käivitame kontrolli.

```
SELECT * FROM haldurid.mon_kontroll WHERE kontroll_id = 3;
```

Ülal toodud lausega saame info käivitunud kontrolli kohta

Tabel 8. Teise stsenaariumi kontrolli andmed.

kontroll_id	alguse_aeg	lopu_aeg	vigade_arv	tulemus
3	26.05.2014 15:31	26.05.2014 15:33	0	’Read kontrollitud. Vigu ei ole.’

Nagu tulemusest näha, siis uusi vigu ei registreeritud ning peaks olema kontrollitud ja staatust muudetud ka eelmise kontrolli käigus avastatud vigadel. Seda näeme jällegi LISA 1 oleva päringut käivitades.

Tabel 9. Teise stsenaariumi vigade andmed.

viga_id	skeemi_nimi	nimi	votme_vaartus	staatus
1	haldurid	testime_menetluseid	40651111	lahendatud
2	haldurid	testime_menetluseid	40651932	lahendatud
3	haldurid	testime_maaruseid	9000407312	lahendatud
4	haldurid	testime_maaruseid	9000407882	lahendatud

Algoritm kontrollis esmalt uuesti üle vigu sisaldavad read ja märkis parandatud vead lahendatuks. Seejärel jätkas algoritm uute ridade kontrollimist. Seega võib väita, et ka teise stsenaariumiga tuleb algoritm hästi toime.

5.4 Kolmas stsenaarium

Veebibaasis ning keskbaasis on mõlemasse loodud tabelisse *haldurid.testime_maaruseid* ning *haldurid.testime_menetluseid* lisatud nüüdseks 3000 rida, mis teeb kokku 6000 rida. Kustutame tabelist *mon_viimane_kontrollitud_rida* kõik read peale algsete pidepunktide, mis tabelis *menetlused* on 40649600 ning tabelis *maarused* on 9000407200. Seega peab algoritm uuesti kõik read üle kontrollima.

Kontrollil on tööd alustades vaja kontrollida 6000 rida. Eemaldame veebibaasi *haldurid.testime_maarusid* tabelist ühe rea, määruse numbriga 9000407983. Seejärel käivitame kontrolli ja uurime, kas kontroll suudab veebibaasist puudu oleva rea tuvastada.

```
SELECT * FROM haldurid.mon_kontroll WHERE kontroll_id = 4;
```

Ülal toodud lausega saame info käivitunud kontrolli kohta.

Tabel 10. Kolmanda stsenaariumi esimese kontrolli andmed.

kontroll_id	alguse_aeg	lopu_aeg	vigade_arv	tulemus
4	26.05.2014 15:52	26.05.2014 15:57	1	’Leitud uued vead…’

LISA 1 asuva päringu käivitamisel vaatame kontrolli leitud vea infot.

Tabel 11. Kolmanda stsenaariumi esimese kontrolli järgsete vigade andmed.

viga_id	skeemi_nimi	nimi	votme_vaartus	staatus
1	haldurid	testime_menetluseid	40651111	lahendatud

2	haldurid	testime_menetluseid	40651932	lahendatud
3	haldurid	testime_maaruseid	9000407312	lahendatud
4	haldurid	testime_maaruseid	9000407882	lahendatud
5	haldurid	testime_maaruseid	9000407983	parandamata

Selgub, et viie minuti jooksul on algoritm suutnud kontrollida 6000 rida ning leidnud kuue tuhande rea hulgast ühe, mis on veebibaasis puudu. Nüüd uuesti kontrolli käivitades peaks algoritm asuma kohe ainukest registreeritud „parandamata“ staatuses olevat rida kontrollima. Kuna uusi ridu juurde tulnud ei ole, siis vigase rea kontrollimise järel algoritm lõpetab.

Lisame puuduoleva määruse rea ka veebibaasi ning käivitame kontrolli.

```
SELECT * FROM haldurid.mon_kontroll WHERE kontroll_id = 5;
```

Ülal toodud lausega saame info käivitunud kontrolli kohta.

Tabel 12. Kolmanda stsenaariumi teise kontrolli andmed.

kontroll_id	alguse_aeg	lopu_aeg	vigade_arv	tulemus
5	26.05.2014 16:03	26.05.2014 16:03	0	’Read kontrollitud.

LISA 1 asuva päringu käivitamisel vaatame, kas kontroll parandas ainukese vigase vea.

Tabel 13. Kolmanda stsenaariumi teise kontrolli järgsete vigade andmed.

viga_id	skeemi_nimi	nimi	votme_vaartus	staatus
1	haldurid	testime_menetluseid	40651111	lahendatud
2	haldurid	testime_menetluseid	40651932	lahendatud
3	haldurid	testime_maaruseid	9000407312	lahendatud
4	haldurid	testime_maaruseid	9000407882	lahendatud
5	haldurid	testime_maaruseid	9000407983	lahendatud

Tabelite väljavõtetest näeme, et kontroll on vähem kui minuti jooksul uuendanud vigase rea staatust. Järelikult ka kolmanda stsenaariumi puhul leiab algoritm puuduoleva rea ülesse ning kui puuduolev rida süsteemi väliselt lisatakse ka veebibaasi, märgib algoritm vea staatuseks „lahendatud“.

5.5 Testimise järeldused

Läbi viidud testimise käigus mängiti läbi kolm stsenaariumit, mis võivad esineda ridade tiražeerimise tulemusena. Algoritm suutis leida read, mille sisu ei olnud erinevates andmebaasides kooskõlas või read, mis olid veebibaasist puudu. Vigade leidmisel saatis taustaprotsess konfiguratsiooni failist loetud adressaadile e-meili, kus teavitas leitud vigadest. Antud ridade süsteemivälise võrdsustamise järgselt käivitati uus kontroll, mis registreeritud vigu kontrollides sai aru, et read on nüüd võrdsed ning märkis vead lahendatuks.

Äriregistri päevas tiražeeritavate ridade arvuga suudab algoritm mõistliku aja piires toime tulla. Läbi viidud ekstreemolukorra testis, kus kontrollitavate ridade hulk oli kuus tuhat, tuli algoritm samuti toime leides ühe rea, mis oli puudu veebibaasis.

6. Kokkuvõte

Võrreldes PostgreSQL andmebaasi tiražeerimise kooskõla jälgimise olemasolevaid võimalusi Äriregistri soovide ning kasutusel oleva tarkvaraga jõuti selgusele, et sobivat lahendust ei ole. Seega tuleb kooskõla jälgimise vahend ise realiseerida. Töö käigus arendati välja kontrolli algoritmil põhinev erinevate andmebaaside ridade võrdlemist teostav tiražeerimise kooskõla jälgimise tarkvara ja realiseeriti see teatud aegadel automaatselt käivituva taustaprotsessina. Tarkvara realiseeriti programmeerimiskeeles Python.

Tarkvara teostab andmebaasi lisatud ridade kontrolli automaatselt neli korda päevas. Automaatne kontroll eemaldab vajaduse Äriregistri töötajal erinevates andmebaasides tabeli ridade arvu võrrelda, vähendades seeläbi töötaja töökoormust. Kuna kontroll on ainult uusi lisatud ridu jälgides suuteline võrdlema ka tiražeeritud ridade väljade väärtuseid, siis on loodud lahendus kahtlemata efektiivsem kui töö kirjutamise ajal kasutusel olev ridade loendamise päring, mis teeb ainult kindlaks, kas rida on teise andmebaasi jõudnud.

Loodud tarkvara on piisavalt dünaamiline, sest suudab iga andmebaasis oleva tabeli kohta, mille info on lisatud spetsiaalsesse tabelisse, koostada andmete otsimise laused. Tänu sellele omadusele on vajadusel piisavalt lihtne lisada tiražeerimise kooskõla jälgimise alla uusi tabeleid. Selleks tuleb ainult süsteemis registreerida lisatava tabeli nimi, skeemi nimi ning jälgitavate veergude andmed.

Kirjeldatud prototüübi põhjal on võimalik arendada välja kasutajaliides, mida saab ühildada Äriregistri menetlustarkvaraga. Kasutajaliidesest kasutades saab haldur hea ülevaate tiražeerimise kontrolli vahendi tööst ja tulemustest. Lisaks on halduril võimalik hallata vahendi tööd, lisades kontrolli uusi tabelid või muutes leitud vea staatust. Töö kirjutamise hetkel ei ole kasutajaliides veel valminud, kuid see on plaanis lähiajal valmis saada. Töös on esitatud joonistusvahendiga loodud kasutajaliidese ülesehitust kirjeldavad pildid.

Testimisel jõuti järeldusele, et loodud taustaprotsess, mis pärib iga võrreldava rea eraldi teisest andmebaasist PostgreSQL *dblink* funktsiooni kasutades, on piisavalt kiire, et suudaks kontrollida tiražeeritavate ridade kooskõla Äriregistri andmebaasi päevas lisanduvate ridade hulga korral. Ekstreemolukorra testis jõudis tarkvara kontrollida 6000 rida viie minuti jooksul.

Tänu tarkvaras realiseeritud dünaamilisusele ning tarkvara rahuldavale töökiirusele, on võimalik see kasutusele võtta ka teistes süsteemides peale Äriregistri, kus on kasutusel andmete tiražeerimine välisesse andmebaasi, tingimusel, et andmebaas on realiseeritud PostgreSQL andmebaasisüsteemi kasutades.

Kuigi Äriregistris on enamus tabelid loodud logimise põhimõttel, et lisatakse muudatus uue reana, eksisteerib siiski veerge, kus tehakse ka muudatusi. Loodud süsteemi on võimalik edasi arendada, muutes kontrolli algoritmi selliseks, et hakatakse kontrollima ka ridu mida on pärast eelmise kontrolli käivitamist muudetud. Kuna Äriregistri andmebaasis leidub tabeleid, kus muutmise kuupäeva või muud väärtus, mida saaks arvestada pidepunktina rea muudatuse leidmiseks ei eksisteeri, siis tuleb süsteemi edasiarenduseks täiendada ka Äriregistri andmebaasi ning rakendust.

Summary

While comparing available possibilities for monitoring consistency of PostgreSQL database replication with Business Register's wishes and software, it became clear that suitable solution does not exist. Therefore, I had to develop it by myself. In the course of the thesis, I designed a tool for comparing rows in different databases and implemented it as a background process by using Python programming language.

Developed software is based on an algorithm that was described in the thesis. The tool performs automatic checking of consistency of replicated rows. It does so four times a day. Automatic checking removes necessity for the employees of Business Register to compare the number of rows in tables in different databases, thereby reducing their workload. The tool checks only new rows that are added to tables as well as rows where inconsistencies have been previously identified. It is able to compare values in the fields of replicated rows, making the solution more efficient than row count queries that are being used at the time of writing this thesis.

The software is dynamic enough to be able to construct SELECT statements for each table in a database. To make it possible, one has to register data about these tables and their columns. Due to that quality, it is simple enough to add new tables for checking the consistency of replication results.

A user interface can be developed that can be joined with Business Register's other software. From that interface, user can get a good overview of the work the developed tool and its results. In addition, the user can manage work of the tool, by adding new tables under checking or by changing the status of detected mistake.

While testing, I made a conclusion that the developed background process is quick enough to be able to check the consistency of replicated rows by inquiring each row that needs comparing separately from other database using PostgreSQL function *dblink*. The tool works quickly enough, considering the number of new rows that are daily added to Business Register's database. At extreme situation, the software was able to check 6000 rows in five minutes.

Due to the dynamism of the developed software and its adequate performance, it can be used besides Business Register in other systems where master database replication to slave database is used. The prerequisite is that the database has to be a PostgreSQL database.

Even though most of the tables in Business Register have been created on the principle of logging, so that a change is added into table as a new row, there still exists columns where values are updated in existing rows. One can extend the system by changing the control algorithm so that it would start checking rows that have been updated after the last checking session. There are tables in the Business Register's database that do not have a timestamp column for the time of the last change or some other column that could be used to identify that the row has been changed. Hence, one has to make changes in the Business Register's database and user interfaces to make such extended consistency checking possible.

Kasutatud kirjandus

1. E-teatmik [WWW] http://www.vallaste.ee (09.05.2014)
2. Londiste [WWW] http://skytools.projects.pgfoundry.org/doc/londiste.cmdline.html#toc0 (11.05.2014)
3. Wang, X., Yu, H. How to Break MD5 and Other Hash Functions [WWW] http://www.infosec.sdu.edu.cn/uploadfile/papers/How%20to%20Break%20MD5%20and%20Other%20Hash%20Functions.pdf (11.05.2014)
4. Database replication [WWW] http://searchsqlserver.techtarget.com/definition/database-replication (14.05.2014)
5. Installing DBLink for Postgres 9 [WWW] http://stackoverflow.com/questions/5075193/installing-dblink-for-postgres-9 (14.05.2014)
6. Dblink [WWW] http://www.postgresql.org/docs/9.1/static/dblink.html (14.05.2014)
7. ESTERM [WWW] http://mt.legaltext.ee/esterm/ (24.05.2014)
8. Andmekaitse inspektsiooni isikukoodi kasutamise juhend [WWW] http://www.aki.ee/sites/www.aki.ee/files/elfinder/article_files/Isikukoodi%20kasutamise%20juhend_2.rtf (24.05.2014)
9. Linux Crontab: 15 Awesome Cron Job Examples [WWW] http://www.thegeekstuff.com/2009/06/15-practical-crontab-examples/ (02.06.2014)
10. CRONw – CRON for Windows [WWW] http://cronw.sourceforge.net/ (02.06.2014)
11. Pycron [WWW] http://www.kalab.com/freeware/pycron/pycron.htm (02.06.2014)
12. Roman, E. SQL Injection Defense in Python [WWW] http://www.slideshare.net/openpbs/sql-injection-defense-in-python (02.06.2014)

Lisa 1

```
SELECT
  haldurid.mon_viga.viga_id,
  haldurid.mon_tabel.skeemi_nimi,
  haldurid.mon_tabel.nimi,
  haldurid.mon_viga.votme_vaartus,
  haldurid.kl_viga_staatus AS staatus
FROM
  haldurid.mon_viga
  INNER JOIN haldurid.mon_tabel ON (haldurid.mon_viga.tabel_id =
haldurid.mon_tabel.tabel_id)
  INNER JOIN haldurid.mon_viga_staatus_log ON (haldurid.mon_viga =
haldurid.mon_viga_staatus_log)
  INNER JOIN haldurid.kl_viga_staatus ON
(haldurid.mon_viga_staatus_log.viga_staatus_id =
haldurid.mon_viga_staatus.viga_staatus_id)
WHERE
  haldurid.mon_viga_staatus_log.id IN (
  SELECT
    max(haldurid.mon_viga_staatus_log.viga_staatus_log_id)
  FROM
    haldurid.mon_viga_staatus_log
  GROUP BY
    haldurid.mon_viga_staatus_log.viga_id
  );
```