TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Andres Eelma 179027IAIB

LÄHTEKOODIREDAKTORIS NOTEPAD++ SQL PROGRAMMEERIMIST LIHTSUSTAVA PISTIKPROGRAMMI EDASIARENDAMINE MS ACCESSI JA POSTGRESQL JAOKS

Bakalaureusetöö

Juhendaja: Erki Eessaar
PhD

Tallinn 2024

Autorideklaratsioon

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.

Autor: Andres Eelma

10.01.2024

Abstract

Further Development of a Plugin for Source Code Editor Notepad++ that Simplifies SQL Programming in MS Access and PostgreSQL

The aim of the bachelor's thesis is to further develop a plugin called NppDB for the text and source code editor Notepad++. The previous version of the plugin was also developed in TalTech. It supported executing SQL statements in MS Access databases and checking the existence of various errors in the SQL statements. Firstly, the MS Access module's SQL code error predetection functionality has to be further improved to handle even more error types. Secondly, the plugin must make it possible to work with PostgreSQL databases from within Notepad++. With the added PostgreSQL support, it must be possible to execute SQL statements against a database and see the results of the executed statements. Moreover, the users should be able to explore the structure elements of the PostgreSQL database they have connected to. Furthermore, it must also be possible to analyze PostgreSQL SQL statements before executing these to identify possible errors and report these to the user.

While improving the functionality of detecting errors in SQL code, the author examined the existing functionality and checks which were added during the previous work with the program. In addition, he also considered studies about frequent problems in SQL statements. The improvements to the error detection functionality that were added to the plugin were selected based on those works.

Further developments of the plugin were made in C# programming language using the .NET framework. For parsing and analyzing SQL statements ANLTR4 was used to create a parser generator for C#. Npgsql was used to communicate with PostgreSQL.

The new functionalities of the plugin were manually tested by both the author and the supervisor. Moreover, the plugin was also compared against other similar programs that are able to detect errors in PostgreSQL SQL statements or allow its users to execute SQL statements in PostgreSQL databases. Further improvements of the functionality of the program were made based on the results of these validations.

Compared with the existing programs that allow users to execute or check PostgreSQL

SQL statements, the software developed during this thesis is more capable because the developed plugin can both execute and analyze PostgreSQL statements. Out of 38 different errors detected by the developed plugin, SQLFluff was able to detect 9 errors and DataGrip 12 errors. The number of different types of errors in SQL code that this plugin can detect is bigger than in these programs.

The resulting plugin is able to predetect 38 different types of errors in PostgreSQL SQL statements and 40 different types of errors in MS Access SQL statements. Out of 40 different error types in MS Access, 12 were added by the author.

The result of this work is open source. It is published under the MIT license and is publicly available on GitHub at: https://github.com/aneelm/NppDB

The thesis is in Estonian and contains 47 pages of text, 7 chapters, 76 figures, 1 tables.

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks oli täiendada tekstiredaktorile Notepad++ loodud pistikprogrammi NppDB. Pistikprogrammi edasiarendus peab esiteks täiendama MS Accessi moodulis olevat SQL lausetes vigade otsimise funktsionaalsust uute vigade esinemise eelkontrollimisega. Teiseks peab pistikprogrammi abil saama ühenduda PostgreSQL andmebaasidega, vaadata nende struktuurielemente, käivitada seal SQL lauseid, vaadata tulemusi ning käivitatavaid lauseid vigade suhtes eelkontrollima.

SQL lausete kontrollide lisamisel vaadeldi eelneva magistritöö raames lisatud kontrolle ja käsitleti olemasolevaid uuringuid, et leida täiendavaid tüüpilisi vigu, mille esinemist programm võiks ka kontrollida.

Töö käigus loodud tarkvara testiti nii autori kui ka juhendaja poolt. Lisaks valideeriti programmi funktsionaalsust võrreldes seda teiste sarnaste programmidega, milles oli samuti PostgreSQL SQL lausete vigade tuvastamise funktsionaalsus ja/või lausete käivitamise funktsionaalsus. Saadud tagasiside põhjal tehti pistikprogrammi täiendusi ja parandusi. Võrdlus näitas, et loodud programm on konkurentidest parem, sest seal on nii koodi käivitamise kui kontrolli võimekus. Muuhulgas suudab pistikprogramm tuvastada suuremat hulka erinevat tüüpi vigu kui võrreldud programmid.

Käesoleva töö tulemusena valminud tarkvara on avatud lähtekoodiga. See avaldati MIT litsentsiga ja on avalikult kättesaadav GitHub'ist aadressilt: https://github.com/aneelm/NppDB

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 47 leheküljel, 7 peatükki, 76 joonist, 1 tabel.

Lühendite ja mõistete sõnastik

.NET	Avatud lähtekoodiga raamistik, millega saab erinevates programmeerimiskeeltes nagu C# kirjutada tarkvaralahendusi
Alfa staadium	Programmi algeline seis, kuhu tehakse edaspidi veel palju muudatusi, kuid mis on siiski kasutajatele kättesaadav
ANTLR4	Parseri generaator (ANother Tool for Language Recognition)[1]
C#	Objektorienteeritud programmeerimiskeel
IDE	Tarkvara arendamise tarkvara (Integrated Development Environment)
Lexer	Tööriist, mis tükeldab vastavalt ette antud reeglitele lause sõnadeks ja märkideks
Linter	Programm, mis analüüsib koodi ja tuvastab seal probleeme
ODBC	Rakendusliides, mis võimaldab ühenduda erinevate andmebaasisüsteemidega (Open Database Connectivity)
OLEDB	Rakendusliides, mis võimaldab ühenduda erinevate rakendustega (Object Linking and Embedding, Database)
Parser	Tööriist, mis võtab lexeri poolt tükeldatud sõnad ning loob nendest grammatikareeglite alusel süntaksipuu
Pistikprogramm	Väiksem programm, mis lisab suuremale programmile uut funktsionaalsust
SQL	Relatsioonilise andmemudeli alusel väljatöötatud andmebaasikeel (Structured Query Language)

Sisukord

Jooniste loetelu

Tabelite loetelu

1. Sissejuhatus

PostgreSQL [2] on 2023. aasta sügise seisuga populaarsuselt neljas andmebaasihaldussüsteem[3]. Lõputöö lähtub soovist avardada võimalusi PostgreSQL andmebaasidega töötamiseks.

On olemas palju erinevaid tööriistu SQL lausete andmebaasides käivitamiseks, mille abil on võimalik andmebaase vaadelda ja näha käivitatud lausete tulemusi.

Need programmid võivad olla graafilise kasutajaliidesega nagu DbSchema [4], DBeaver [5] ja DbVisualizer [6], mis toetavad muuhulgas ka PostgreSQL andmebaasisüsteemi. Kuid PostgreSQL jaoks on olemas ka käsureapõhise kasutajaliidesega tööriistu nagu psql [7], pgcli [8] ja odbc-cli[9]. Viimane nendest on lõputöö kirjutamise ajahetkel (2023. aasta sügisel) alfa staadiumis, kuid peaks toetama andmebaasisüsteeme, mille jaoks on saadaval ODBC draiverid, ehk siis ka PostgreSQLi.

1.1 Taust ja probleem

Mida rohkem on erinevaid tööriistu, seda suurem on kasutajate valikuvõimalus. Nii nagu on palju erinevaid andmebaasisüsteeme, programmeerimiskeeli ja arenduskeskkondi võib ka olla palju erinevaid keskkondi andmebaaside kasutamiseks. Kasutaja võiks saada töötada talle tuntud ja mugavas keskkonnas, ilma vajaduseta mitmeid erinevaid programme oma töökeskkonda installeerida ja neid seal avada. Enamik programmeerimiskeeli on tekstilised keeled ja nendega töötamiseks on loodud tekstiredaktoreid nagu näiteks Notepad++ [10] ja Visual Studio Code [11]. Oleks väga mugav seal kirjutatud SQL koodi otse andmebaasis käivitada. Visual Studio Code jaoks leidub laiendusi PostgreSQL andmebaasisüsteemiga (aga ka teiste andmebaasisüsteemidega) ühendumiseks ja seal koodi käivitamiseks [12]. Teisalt on Notepad++ Visual Studio Code'ga võrreldes palju kiirem ja väiksema kettamahuga programm. Kui Visual Studio Code ver 1.85.1 paigaldaja kettamaht on 90.4MB ja paigaldatult kasutatud ketta maht on 353MB, siis Notepad++ ver 8.5.7 paigaldaja kettamaht on 4.6MB ja paigaldatult kasutatud ketta maht on 19.5MB[13], [14]. Lisaks saab Notepad++'is avada ajutisi faile ilma neid ise kettale salvestamata ning määrata faili keele ja süntaksi esiletõstmise ilma faililaiendita. Lisaks on võimalik vaadata Notepad++ abil peidetud tähemärke, mis on eriti kasulik üksikute SQL lausete käivitamisel, kuhu võivad lause programmide vahel kopeerimisel nähtamatud tähemärgid tekkida.

Notepad++ on mugav vahend väiksemate koodilõikude (sh SQL koodi lõikude) kirjutamiseks. Näiteks kasutajal, kes ei realiseeri suurt ja mitut keelt hõlmavat rakendust (nagu näiteks õppejõud, õppija, analüütik, andmekvaliteedi parandaja, andmebaasi administraator) võibki oma ülesannete täitmiseks just selline keskkond olla vajalik ja piisav.

Kui tahta lisada Notepad++'i PostgreSQLi andmebaasis SQL koodi käivitamise võimalus, siis ei pea arendust alustama tühjalt kohalt, vaid saab jätkata juba loodud pistikprogrammi arendamisega. Pistikprogrammi NppDB esimene versioon loodi Sangkyu Jungi[15] poolt 2014. aastal. Priit Post [16] täiendas seda oma magistritöö raames 2023. aastal, realiseerides võimaluse käivitada SQL lauseid MS Accessi [17] andmebaasis ning neid lauseid enne käivitamist vigade suhtes kontrollida [18].

Kui MS Accessi andmebaasis SQL lauseid käivitada võimaldavaid programme leidub küllaltki palju, nagu näiteks DbVisualizer [6], DBeaver [5] ja DbGate [19], siis SQL koodi eelkontrollimise funktsionaalsus on üpris unikaalne.

Magistritöös realiseeriti hulk kontrolle, kuid tegelikkuses on lausetes võimalike vigade hulk palju suurem[20]–[22]. Enamik kontrollitavatest vigadest on universaalsed mistahes SQL-andmebaasisüsteemi jaoks, kuid kuna igal andmebaasisüsteemil on oma SQLi dialekt e mägimurrak, siis leidub ka andmebaasisüsteemi-spetsiifilisi vigu. See pistikprogramm on muuhulgas kasulik SQL programmeerimise õpetajatele, kes saavad seda kasutada nii koodi kirjutamise demonstreerimiseks kui ka õppurite poolt kirjutatud lahenduste kontrollimiseks. See pistikprogramm on samuti kasulik SQLi õppijatele, sest nad saavad mugavas ja tuttavas keskkonnas SQL lauseid käivitada ja saavad juba enne koodi käivitamist tagasisidet võimalike vigade kohta. Vigade kontrolli funktsionaalsus on kasulik tegelikult mistahes tasemel SQL lausete kirjutajale, sest ka ekspert võib teha hooletusvigu ning see, kes koodi igapäevaselt ei kirjuta, võib asju ajapikku unustada.

1.2 Ülesandepüstitus

Töö eesmärgiks on edasi arendada teksti- ja lähtekoodiredaktorile Notepad++ pistikprogrammi NppDB.

Esmalt tuleb täiendada MS Accessi lausete vigade kontrolli, et programm suudaks hoiatada veel suurema hulga vigade eest. Kontrollitavad vead peaksid olema sellised, mille kontrollimiseks piisab kontrollitava lause töötlemisest ja ei peaks lugema andmeid kontrollitava andmebaasi kohta. Sellised vead on universaalsed ja ei ole tingitud näiteks sellest, et tabeli või veeru nimi on valesti kirjutatud. Sellised vead kuuluvad süntaktiliste, semantiliste või keerukusega seotud vigade klassi.

Seejärel tuleb pistikprogrammi täiendada, et võimaldada PostgreSQL andmebaasidega ühenduse loomist, SQL lausete käivitamist nendes andmebaasides ning ka SQL lausete vigade kontrollimist, nagu on tehtud juba eksisteerivas MS Accessi jaoks loodud lahenduses. PostgreSQLi jaoks tuleb realiseerida kontrollid kõigi universaalsete vigade jaoks (ühised erinevatele andmebaasisüsteemidele), mille esinemist kontrollitakse juba MS Accessi puhul ning lisaks ka mõned spetsiifilised kontrollid.

Pistikprogrammi täiendus peab olema avatud lähtekoodiga ja peab olema avaldatud koodihoidlas GitHub[23].

1.3 Töö struktuur

Bakalaureusetöö koosneb seitsmest peatükist.

Esimeses peatükis on töö sissejuhatus, kus kirjeldatakse töö tausta ja lahendatavat probleemi ning esitatakse ka ülesandepüstitus.

Teises peatükis antakse ülevaade töö tegemise protsessist ning töö tegemisel kasutatud tööriistadest.

Kolmandas peatükis antakse ülevaade töö teoreetilisest taustast. Kirjeldatakse SQL lausetes esinevate vigade liigitust ning tutvustatakse lühidalt erinevaid programme, mis võimaldavad PostgreSQL andmebaasisüsteemis SQL lauseid käivitada ning ka programme, mis suudavad SQL lausetes sarnaselt NppDB pistikprogrammile vigu tuvastada.

Neljandas peatükis esitletakse töö tulemust. Selleks kirjeldatakse kõigepealt pistikprogrammi eesmärke ning funktsionaalseid ja mittefunktsionaalseid nõudeid PostgreSQLi kasutamise moodulile. Lisaks kirjeldatakse ka tagarakenduse tööpõhimõtteid, kasutajaliidesesse tehtud muudatusi ja programmi testimist. Veel kirjeldatakse programmis teadaolevaid puuduseid, antakse juhis pistikprogrammi paigaldamiseks ning esitatakse info arendatud tarkvara litsentsi kohta.

Viiendas peatükis antakse ülevaade arendatud tarkvara valideerimisest. Tuuakse välja võrdlused käesoleva töö raames arendatud pistikprogrammi ja programmide vahel, mis suudavad PostgreSQL andmebaasis SQL lauseid käivitada ja/või probleeme tuvastada. Samuti kirjeldatakse programmi kasutajate poolset testimist.

Kuuendas peatükis tuuakse välja pistikprogrammi täiendavad arendusvõimalused, mida lõputöö raames teostada ei jõutud.

Seitsmendas peatükis on bakalaureusetöö kokkuvõte.

2. Metoodika

Käesolevas peatükis kirjeldab autor täpsemalt töö tegemise protsessi ning töö käigus kasutatud vahendeid.

2.1 Ülevaade töö protsessist

Töö käigus laadis autor pidevalt GitHubi hoidlasse üles uusi versioone pistikprogrammist, mida juhendaja sai katsetada. Juhendaja edastas autorile infot tarkvarast leitud vigade kohta ning soove, kuidas hõlbustada tarkvara kasutamist PostgreSQL andmebaasisüsteemiga suhtlemiseks. Töö käigus üritas autor parandada vigu kohe kui nendest teadlikuks sai, et juba leitud vead ei takistaks järgnevate viga leidmist ja programmi kasutamist.

SQL lausete vigade kontrolli lisamisel ja täiendamisel uuris autor nii eelnevalt tehtud magistritöös lisatud kontrolle[16], kui ka eraldiseisvaid uuringuid.

MS Accessi vigade kontrolli täiendamisel ja PostgreSQL vigade kontrolli lisamisel lisas autor programmi ka ühiktestid, mis kindlustas, et programmi jooksvalt arendades ei läheks eelnevalt lisatud kontrollid katki. Veel lisas autor testid, et lisaks üksikute lausete kontrollimise funktsionaalsusele jääks töökorda ka programmi funktsionaalsus, mis võimaldab jooksutada mitu SQL lauset korraga ning ei tekiks vigu, kui PostgreSQL lauseid käivitatakse päris andmebaasis.

2.2 Kasutatud tööriistad

Käesoleva pistikprogrammi edasiarenduse realiseerimiseks kasutatavateks vahenditeks olid varasemalt loodud pistikprogramm NppDB[18] ja lähtekoodi redaktor Notepad++[10].

PostgreSQL andmebaasisüsteemiga suhtluseks on kasutusel Npgsql[24], mis ei kasuta ODBC[25] ega OLEDB[26] draivereid, vaid on otse integreeritud .NET raamistikku. Seega pakub see sujuvat integratsiooni .NET rakenduste ja PostgreSQL andmebaaside vahel.

SQL lausete töötlemiseks kasutati parseri generaatorit ANTLR4[1], mis kasutas vastavalt MS Accessi või PostgreSQL SQL süntaksi[27] grammatikareegleid, et luua puu SQL lausete osadest, mille erinevaid osi analüüsides sai tuvastada probleemseid kohti.

Tarkvara arendamisel kasutatavaks keeleks oli C#[28], raamistikuks .NET[28] ja arendamisel kasutatud tarkvara oli Visual Studio[29] ja Visual Studio Code[11].

Tarkvara testimiseks kasutati xUnit.net[30] ühiktestimise tööriista ning Testcontainers[31] raamistikku, et testide jaoks nullist alati sama PostgreSQL andmebaas üles seada.

Tarkvara arendamisel ja testimisel kasutati andmebaasisüsteemi PostgreSQL versioone 9.2, 12, 15 ja 16 ning Notepad++ versiooni 8.5.7.

3. Teoreetiline taust

Selles peatükis kirjutatakse erinevatest vigade kategooriatest, millesse kuuluvaid vigu SQL lausete kirjutamisel tehakse. Veel käsitletakse erinevaid programme, mis võimaldavad kasutajal suhelda PostgreSQL andmebaasisüsteemiga ja mis võimaldavad PostgreSQL SQL lauseid analüüsida ja seal vigu tuvastada.

3.1 Vead SQL lausetes

Käesolevas jaotises vaadeldakse erinevaid vigade kategooriaid, mida SQL lausete kirjutamisel tehakse. Sellised vigade kategooriad ning nende alla kuuluvad vead on ka välja toodud erinevates eelnevalt tehtud uuringutes [20], [22].

- Süntaktiliste vigade puhul ei saa SQL lauset käivitada, kuna lause ei ole loodud vastavalt andmebaasisüsteemi nõutud SQL lausete struktuurile. SQL lause käivitamisel annab andmebaasisüsteem veateate, mis on lause ehituses valesti. Selliseid vigu saab analüüsida, kui programmile on teada, milline on andmebaasisüsteemi poolt oodatav lausete struktuur.
- Semantiliste vigade puhul on SQL lause süntaktiliselt korrektne ja seda saab andmebaasisüsteemis käitada, kuid saadud tulemus on vale, sest lause kirjutamisel on tehtud vigu. Selliste vigade puhul ei pea vea tuvastamiseks teadma, milline on andmebaasist oodatav tulemus. Semantiline viga on näiteks see kui ühele veerule pannakse mitu tingimust, mis ei saa olla samaaegselt tõesed või see kui lauses nähakse ette, et tulemuses peab olema mitu sama nimega veergu.
- Loogiliste vigade puhul on SQL lause süntaktiliselt korrektne ja seda saab andmebaasisüsteemis käitada, kuid saadud tulemus on vale, sest lause kirjutamisel on tehtud vigu. Selliseid vigu ei saa analüüsida ja kontrollida, kuna ei ole teada, mis tulemust kasutaja ootab. Loogiline viga on näiteks see kui kasutaja tahab leida andmeid, mis vastavad mingile tingimusele, kuid tingimuses kasutatavad väärtused on valesti kirja pandud - näiteks tahtis kasutaja leida kaupu, mille hind on vahemikus 10 kuni 100, kuid kirjutas BETWEEN 10 AND 1000.
- Keerukusega seotud vigade puhul on SQL lause süntaktiliselt korrektne ja selle täitmisel tagastatakse ka õiged tulemused, kuid SQL lause ehitusse on pandud üleliigseid osi. Keerukusega seotud vigade tuvastamiseks peab sageli teadma, mis on SQL lause oodatud tulemus ning ka andmebaasi struktuuri, kus lauset täidetakse. Keerukuse seotud viga on näiteks lauses, kus tahetakse andmeid küsida üle mitme

tabeli, kuid SQL lausesse ühendati ebavajalikke tabeleid, mida lause täitmiseks tegelikult vaja ei lähe.

3.2 Programmid PostgreSQLis SQL lausete käivitamiseks

Selles jaotises vaadatakse lühidalt erinevaid andmebaasi haldusprogramme, millega on võimalik suhelda PostgreSQL andmebaasisüsteemiga. Nendega saab nii andmebaasi struktuuri näha kui ka SQL lauseid käivitada. See nimekiri pole mõeldud lõplikuna, vaid demonstreerib, et leidub päris palju erinevaid programme.

- pgAdmin[32] on andmebaasi haldusprogramm PostgreSQL andmebaaside jaoks ja PostgreSQL andmebaasist tuletatud SQL-andmebaasisüsteemidele nagu EDB Advanced Server[33]. Seda saab käitada kas veebirakendusena või töölauarakendusena. pgAdmin on avatud lähtekoodiga tarkvara, kuhu võivad kõik panustada ning PostgreSQL paigaldajaga pakutakse ka võimalus pgAdmin4 paigaldada.
- JetBrains' DataGrip[34] on Jetbrainsi arendatud andmebaasisüsteemide spetsiifiline integreeritud programmeerimiskeskkond (IDE), mis toetab palju erinevaid andmebaasisüsteeme, sealhulgas ka PostgreSQLi. Andmebaasisüsteemidega suhtlemist võimaldavad JetBrains tooted on tasulised.
- DBeaver[5] on tasuta ja avatud lähtekoodiga universaalne andmebaaside tööriist tarkvara arendajatele ja andmebaasi administraatoritele, mis võimaldab suhelda mitmete erinevate andmebaasisüsteemidega. DBeaveril on olemas ka tasuline versioon.
- DbVisualizer[6] on tasuta mitmete erinevate andmebaasisüsteemidega suhtlust ja nende andmebaaside haldust võimaldav tööriist. Tasuta versioonis saab ühest tabelist visuaalselt korraga elementaarseid lauseid moodustada, tasulises versioonis on võimalik visuaalselt ka keerulisemaid SQL lauseid koostada.
- Beekeeper Studio[35] on avatud lähtekoodiga tasuline andmebaaside haldussüsteem, mis keskendub eelkõige kasutusmugavusele ja lihtsusele.
- DbGate[19] on tasuta ja avatud lähtekoodiga andmebaaside haldussüsteem, mis toetab palju erinevaid andmebaasisüsteeme. Programmile on võimalik ka pistikprogramme paigaldada ning DbGate toetab ka visuaalselt SQL lausete koostamist.
- Postgres Visual Query Builder[36] pakub veebipõhise graafilise kasutajaliidese PostgreSQL andmebaasis käivitatavate SELECT lausete graafiliseks koostamiseks ning koostatud lausete käivitamiseks. Tegemist on avatud lähtekoodiga ja tasuta pakutava programmiga, mille on loonud Tallinna Tehnikaülikooli üliõpilased.

3.3 Programmid, mis kontrollivad PostgreSQLi SQL lauseid

On erinevaid programme, nagu näiteks JetBrains'i arenduskeskkonna IDEd [37], käsureapõhine tööriist SQLFluff[38] ja veebikeskkonna töövahend EverSQL[39], mis suudavad sarnaselt käesoleva lõputöö raames edasi arendatud pistikprogrammile kontrollida PostgreSQL SQL lausete valiidsust e reeglipärasust. Tegelikult on SQL lausetes programmi- ja stiilivigade esinemist kontrollivaid lintimise programme nagu SQLFluff ka teisi [40]. See vahend valiti võrdlemiseks, kuna on sarnaste tööriistade seas populaarseim ja kõige rohkemate kasutajatega.

Üheski eelnimetatud tööriistas ei pea SQL lausetes vigade kontrollimiseks olema loodud ühendus andmebaasiga, vaid peab olema määratud, millise andmebaasiüsteemi SQL dialektiga on tegemist.

JetBrainsi toodetel on ka erinevaid variante, kas siis täiemahuline IDE, nt IntelliJ Ultimate[41], kuhu lisaks programmeerimist lihtsustavale funktsionaalsusele on külge ehitatud ka andmebaasisüsteemides SQL lausete käivitamise ja analüüsimise funktsionaalsus või puhas andmebaasidele keskendunud IDE DataGrip[34], kus ei ole üleliigset funktsionaalsust. Mõlemal juhul toetavad programmid andmebaasiga suhtlust, kuid nõuavad ka arvutilt rohkelt ressursse ning kettaruumi võtab värskelt paigaldatult DataGrip (ver 2023.2.3) 1.2GB ja IntelliJ Ultimate (ver 2023.2.5) 3.7GB.

SQLFluff on linttöövahend (linter), mis töötab käsurealt programmeerimiskeele Python keskkonnas. Kuigi SQLFluff on käsurea tööriist, siis on olemas Visual Studio Code pistikprogramm, mis võimaldab kahte programmi omavahel siduda[42]. Sellega saab faili salvestamisel automaatselt kasutada SQLFluffi erinevaid funktsionaalsusi. Pistikprogrammiga aga SQLFluff eraldi kaasa ei tule ja selle peab ise arvutisse paigaldama. Nende kahe programmi sidumisel on võimalik Visual Studio Code tekstiredaktoris visuaalselt näha SQL lausete vigaseid kohti, samamoodi nagu on neid võimalik näha Jetbrains IDE-s kui ka lõputöö raames edasi arendatud pistikprogrammis. Küll aga ei toeta SQLFluff andmebaasisüsteemidega suhtlust.

EverSQL on tehisintellektil põhinev peamiselt SQL lausete optimeerija, mis suudab SQL lauseid kiiremaks ja efektiivsemaks teha [39]. Sellel on ka SQL lausete süntaksi kontrolli ja valideerimise funktsionaalsus, mida saab veebikeskkonnas kasutada. Süntaksi kontrolli ja valideerimist saab kasutada sisestades programmi ainult SQL lause, kuid lausete optimiseerimiseks peab programm teadma andmebaasi ülesehitust. EverSQL suudab analüüsida ühte SQL lauset korraga ning ei toeta suhtlust andmebaasisüsteemidega.

4. Notepad++ pistikprogrammi täiendused

Käesolevas peatükis kirjeldatakse bakalaureusetöö raames edasi arendatud pistikprogrammi eesmärke ja funktsionaalseid ning mittefunktsionaalseid nõudeid. Samuti kirjeldatakse pistikprogrammi SQL lausete kontrollide lisamist ja täiendamist nii MS Accessi kui ka PostgreSQL jaoks. Lisaks antakse ülevaade kasutajaliidese täiendustest, tagarakendusest ja andmebaasisüsteemiga suhtlemiseks tehtud draiveri valikust, programmi testimisest, teadaolevatest puudustest ja ka installeerimisest.

4.1 Eesmärgid

Käesoleva töö raames täiendatud pistikprogramm peab olema võimeline probleemideta käivitama kõiki erinevaid PostgreSQL SQL lauseid. Programm peab olema suuteline eristama kommentaare SQL lausetest ning mitme SQL lause korraga käivitamisel SQL lauseid üksteisest. Samuti peab programm suutma täita transaktsiooni plokke nii, et on täidetud isoleerituse omadus ja transaktsiooni saab lõpuks soovi korral tagasi rullida. Lisaks peab programm olema võimeline tuvastama vigu, mis võivad esineda PostgreSQL SQL lausetes ja neid kasutajale kuvama.

4.2 Funktsionaalsed nõuded PostgreSQL kasutamise moodulile

Käesolevas jaotises esitatakse funktsionaalsed nõuded edasiarendatavale pistikprogrammile.

4.2.1 PostgreSQL andmebaasis koodi käivitamise funktsionaalsus

Käesolevas jaotises esitatakse funktsionaalsed nõuded PostgreSQL koodi käivitamisele tekstiredaktoris Notepad++.

- Peab olema võimalik luua ühendus Notepad++ tekstiredaktori ja andmebaasisüsteemi PostgreSQL vahel.
- Skeemide kaupa peab olema võimalik näha ühendatud andmebaasi baastabeleid e tabeleid, väliseid tabeleid, vaateid ja materialiseeritud vaateid e hetktõmmiseid ning funktsioone.
- Peab olema võimalik näha ühendatud andmebaasi tabelite, väliste tabelite, vaadete ja materialiseeritud vaadete struktuuri, et oleks hea ülevaade veergudest, veergude

tüüpidest, võtmetest ja indeksitest.
- Peab olema võimalik näha ühendatud andmebaasi funktsioonide sisendparameetreid, et saada nendest ülevaade ja eristada üksteisest samanimelisi funktsioone.
- Peab olema võimalik ühendatud andmebaasis, mis on Notepad++ redaktori aknaga seotud, käivitada SQL lauseid ning näha nende lausete tulemusi.
- Programm peab olema võimeline töötlema ja üksteisest eristama keerulisi SQL lauseid.

4.2.2 SQL lausete kontrollimise funktsionaalsus

Käesolevas jaotises esitatakse PostgreSQL SQL lausete kontrollimise funktsionaalsed nõuded.

- Peab olema võimalik kontrollida PostgreSQL SQL lausete vigu, mis ei eelda andmebaasi struktuuri teadmist.
- PostgreSQL SQL lausetes peab olema võimalik kontrollida kõiki vigu, mida pistikprogramm kontrollib ka MS Access SQL lausetes ja mis ei ole andmebaasisüsteemispetsiifilised.

4.3 Mittefunktsionaalsed nõuded PostgreSQL kasutamise moodulile

Käesolevas jaotises esitatakse edasiarendatavale pistikprogrammile mittefunktsionaalsed nõuded.

- Peab olema võimalik töötada PostgreSQL andmebaasisüsteemide erinevate versioonidega.
- Peab säilima pistikprogrammi teiste andmebaasisüsteemide jaoks laiendamise võimalus.

4.4 Täiendavad SQL lausete kontrollid MS Accessi jaoks

Pistikprogrammis täiendati ka olemasolevaid SQL lausete kontrolle MS Accessi mooduli jaoks. Järgnevalt on toodud loetelu kontrollidest, mis MS Accessi jaoks lisati. Iga vea tüübi kohta esitatakse lühikommentaar ja näide lausest, kus selline viga esineb. Need vead võivad esineda erinevat tüüpi SQL lausetes (sh INSERT, UPDATE, DELETE, SELECT ... INTO) ja ka alampäringutes. Vigade kontrolli testiti suurema hulga lausetega kui selles jaotises on välja toodud.

Magistritöös [16], mida käesolev töö edasi arendas, realiseeriti 28 kontrolli. Käesolev töö lisas 12 kontrolli, mis valiti välja uurides varasemalt koostatud uuringuid [20]–[22] ning MS Accessis ja MS SQL Serveris kasutatavat TOP predikaati [43].

1. **TOP predikaat ilma ORDER BY klauslita.** [43], [44] TOP predikaat tagastab teatud arvu ridu päringutulemuse esimeste ridade hulgast. Kui päringus puudub ridade sorteerimine, siis on tulemus määramatu. Tegemist on semantilise veaga. Joonisel 1 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT TOP 100 * FROM Hotell;
```

Joonis 1. *Näide SELECT lausest, kus on TOP predikaat ilma ORDER BY klauslita.*

2. **Puuduv veerg GROUP BY klauslist.** [22] SELECT klauslis nimetatud veerg (millele ei rakendata kokkuvõttefunktsiooni), mis puudub GROUP BY klauslist, on MS Access andmebaasisüsteemi puhul süntaktiline viga. PostgreSQL oskab vastavalt SQL standardile [45] *osadel* juhtudel tabelites deklareeritud võtmete järgi aru saada, et GROUP BY klauslist võib veerg ilma tulemust mõjutamata puududa. Joonisel 2 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT
    ruum_tüüp,
    hind,
    Count(hotelli_nr) AS hotell_arv
FROM Ruum
GROUP BY hind;
```

Joonis 2. *Näide SELECT lausest, kus GROUP BY klauslist puudub veerg.*

3. **Puuduv ühendamise tingimus** [20], [22]. Ühendamise tingimuse puudumine tähendab, et tulemuseks on otsekorrutis, mida iseloomustab lähtetabelitega võrreldes suur ridade hulk ning mis *enamasti* pole lauselt oodatav lõpptulemus. Tegemist võib olla loogilise veaga. Joonisel 3 on näide SELECT lausest, kus vastav probleem esineb.

```
SELECT DISTINCT
    Hotell.*,
    Ruum.ruumi_nr
FROM Hotell, Ruum;
```

Joonis 3. *Näide SELECT lausest, kus puudub ühendamise tingimus.*

4. **Üleliigne semikoolon** [22]. Üleliigne semikoolon lause lõpus SQL lause tulemust ega käitumist ei mõjuta. Küll aga võib see tekitada probleeme, kui SQL lauset kasutada teistes programmides. Selle näol, et lauset lõpetav märk esitatakse mitmekordselt, on tegemist süntaksiveaga. Joonisel 4 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT * FROM Hotell;;
```

Joonis 4. *Näide SELECT lausest, kus on üleliigne semikoolon.*

5. **WHERE või HAVING klauslis puuduv otsingutingimus.** [20] WHERE või HAVING klauslis otsingutingimuse puudumine on süntaktiline viga, kuna lause on poolik. Joonisel 5 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT * FROM Hotell WHERE;
```

Joonis 5. *Näide SELECT lausest, kus puudub WHERE klauslis otsingutingimus.*

6. **TOP predikaat võib alampäringus ebasobivalt tagastada mitu rida.** [43], [44] Kui TOP predikaati kasutatakse alampäringus koos võrdluse operaatoriga põhipäringus, siis lause ei pruugi töötada, kuna TOP predikaat piiranguga 1 võib tagastada rohkem kui ühe rea. Tegemist on semantilise veaga. Joonisel 6 on näide SELECT lausest, kus vastav viga esineb. Antud juhul võib leiduda mitu külalist, kellel on kõige rohkem reserveerimisi. Lause oleks vigane ka siis, kui = operaatori asemel kasutatakse <>, >, <, >=, <= operaatoreid. Samas kui = asemel oleks näiteks IN, NOT IN, = SOME, = ANY, <> ALL, siis viga ei oleks.

```
SELECT * FROM Reserveerimine
WHERE külalise_nr = (
    SELECT TOP 1 külalise_nr
    FROM Reserveerimine
    GROUP BY külalise_nr
    ORDER BY Count(*) DESC
);
```

Joonis 6. *Näide SELECT lausest, kus TOP'i kasutav alampäring võib tagastada ebasobivalt üle ühe rea.*

7. **TOP n PERCENT predikaadis kasutatakse ebaõiget arvulist piirangut.** TOP PERCENT predikaat võimaldab määrata väljastatavate ridade protsendi, mis saab olla täisarv vahemikus 1-100. Kui vastavatest piiridest välja minna, siis ei saa

andmebaasisüsteem tulemust leida. Tegemist on semantilise veaga. Joonisel 7 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT TOP 101 PERCENT *
FROM Reserveerimine
ORDER BY hotelli_nr;
```

Joonis 7. *Näide SELECT lausest, kus TOP PERCENT piirang on ebaõige täisarv.*

8. **TOP predikaadiga kasutatakse ebakorrektset arvulist piirangut.** Kui TOP predikaadis kasutada piirangut, mis on väiksem kui 1 , siis ei saa andmebaasisüsteem tulemust leida. Tegemist on semantilise veaga. Joonisel 8 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT TOP 0 *
FROM Reserveerimine
ORDER BY hotelli_nr;
```

Joonis 8. *Näide SELECT lausest, kus TOP piirang on ebaõige täisarv.*

9. **TOP predikaadis ei kasutata täisarvu.** TOP predikaadi tingimuses peab kasutama täisarvulist väärtust. Mitte-täisarvulise või mittearvulise väärtuse korral või ka väärtuse puudumise korral on lause süntaktiliselt ebakorrektne. Joonisel 9 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT TOP 5.5 *
FROM Reserveerimine
ORDER BY hotelli_nr;
```

Joonis 9. *Näide SELECT lausest, kus TOP piirangus kasutatav väärtus pole täisarv.*

10. **FROM klausli alampäringul puudub alias.** [44] FROM klausli alampäringul peab olema alias, kuna muidu määrab andmebaasisüsteem aliase ise, ning hiljem võib olla keeruline tuvastada alampäringu nimest, mida alampäring täpsemalt tagastab. Tegemist on keerukusega seotud veaga. Joonisel 10 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT Count(*) AS arv
FROM (
    SELECT DISTINCT kommentaar
    FROM Reserveerimine
);
```

Joonis 10. *Näide SELECT lausest, kus FROM klauslis oleval alampäringul puudub alias.*

11. **TOP predikaati kasutatakse päringus, mis tagastab maksimaalselt ühe rea.** [44] TOP predikaati kasutatakse päringus, et leida mitme rea hulgast nende pärisalamhulk. Kui päring aga tagastab alati maksimaalselt ühe rea, näiteks kokkuvõttefunktsiooni kasutamise korral, siis on TOP kasutamine ebavajalik. Tegemist on semantilise veaga. Joonisel 11 on näide SELECT lausest, kus vastav viga esineb - maksimaalse hotelli numbri leidmise lause tulemuses on üks rida (kui on registreeritud vähemalt üks reserveerimine) või null rida (kui ei ole registreeritud ühtegi reserveerimist).

```
SELECT TOP 1 Max(hotelli_nr) AS suurim
FROM Reserveerimine;
```

Joonis 11. *Näide SELECT lausest, kus TOP predikaati tagastatakse päringus, mis tagastab maksimaalselt ühe rea.*

12. **WHERE või HAVING võtmesõna kasutatakse lauses samal tasemel mitu korda.** [22]. SQL lauses kasutatakse samal tasemel WHERE või HAVING võtmesõna mitmekordselt, mis on süntaktiline viga. Joonisel 12 on näide SELECT lausest, kus vastav viga esineb. Samas tuleb öelda, et WHERE või HAVING klauslis võib olla alamtingimusi, kus kasutatakse alampäringuid, milles on omakorda WHERE või HAVING klauslid ja see on lubatud.

```
SELECT ruumi_tüüp
FROM Ruum
GROUP BY ruumi_tüüp
HAVING  Count(*)=1
OR HAVING Count(*)=2;
```

Joonis 12. *Näide SELECT lausest, kus on üleliigne HAVING võtmesõna.*

4.5 SQL lausete kontrollid PostgreSQL jaoks

Pistikprogrammi jaoks lisati peaaegu kõik samad kontrollid, mis eelnevas magistritöös [16] olid MS Accessi jaoks lisatud. Ei realiseeritud kontrolli *"SELECT alampäringu tulemuses on mitu veergu juhul kui alampäringut kasutatakse IN, NOT IN, =ANY, =SOME ja <>ALL operaatoritega tingimuses"*, sest PostgreSQLi puhul võib seal olla mitu veergu. Selle asemel realiseeriti kontroll, et põhipäringu tingimuses olev veergude arv ja alampäringus leitud veergude arv peab olema võrdne.

Samuti ei realiseeritud PostgreSQL jaoks käesolevas töös MS Accessi jaoks lisatud kontrolle:

- *"TOP n PERCENT predikaadis kasutatakse ebaõiget arvulist piirangut"*, sest PostgreSQLis ei saa sellist konstruktsiooni kasutada.
- *"TOP predikaadis ei kasutata täisarvu."*, sest PostgreSQLis saab LIMIT/FETCH klauslis kasutada ka püsikomaarvu.

Kokku realiseeriti 38 kontrolli. Nendest kontrollidest on 37 sellised, mis on realiseeritud *mingil kujul* ka MS Accessi korral ning üks selline, mis on spetsiifiline PostgreSQLile.

Tuleb rõhutada, et osade kontrollide puhul, mis olid realiseeritud MS Accessis, tehti PostgreSQLis realiseerimisel muudatusi ja täiendusi, et arvestada PostgreSQL erisustega võrreldes MS Accessiga. Näiteks PostgreSQLis ei saa kasutada TOP predikaati, kuid saab kasutada LIMIT või FETCH klauslit. Samuti saab PostgreSQLis kasutada lisaks LIKE operaatorile SIMILAR TO operaatorit ning regulaaravaldise mustrile vastavuse kontrollimist, mida MS Accessis ei saa kasutada. Samuti saab PostgreSQLis kasutada lisaks hulgateoreetilisele UNION operaatorile ka EXCEPT ja INTERSECT operaatoreid, mida MS Accessis ei saa kasutada.

Järgnevalt on toodud loetelu kontrollidest, mis PostgreSQL jaoks lisati. Esitatakse vea nimi koos viitega allikale, kust selle vea kohta saab lähemalt lugeda, viga sisaldav näitelause ja mõnikord lisakommentaare. Need vead võivad esineda erinevat tüüpi SQL lausetes (sh INSERT, UPDATE, DELETE, SELECT ... INTO, CREATE TABLE AS) ja ka alampäringutes, sh ühistes tabeli avaldistes. Vigade kontrolli testiti suurema hulga lausetega kui selles jaotises on välja toodud.

Kõigepealt tuuakse välja kontrollid, mis oli tehtud eelnevas magistritöös MS Accessi jaoks ning mis nüüd realiseeritakse ka PostgreSQLi jaoks. Seejärel tuuakse välja kontrollid, mis käesolevas töös realiseeriti MS Accessi jaoks ning mis realiseeriti ka PostgreSQLi jaoks. Kõige lõpuks esitatakse kontrollid, mis on tehtud spetsiifiliselt PostgreSQLi jaoks.

4.5.1 Eelnevas magistritöös realiseeritud kontrollid MS Accessi jaoks, mis realiseeriti mingil kujul ka PostgreSQL jaoks

1. **{LIKE|SIMILAR TO} operaator ilma metamärkideta.** [16] Probleem esineb kui SQL lauses kasutatakse LIKE, või SIMILAR TO operaatoreid, mis on mõeldud tekstilise väärtuse mustriga võrdlemiseks, kuid mustris puuduvad vastavad metamärgid. LIKE operaatori puhul on need "%" ja "_" ning SIMILAR TO puhul "|", "*", "+", "?", "{", "}", "(", ")", "[", "]". Ilma metamärkideta võrreldakse tekstilisi väärtusi üks ühele, ning tegu võib olla kas loogilise veaga või keerukusega seotud veaga.

Selline kontroll tehti ka MS Accessi jaoks, kuid PostgreSQLi puhul lisati võimalus, et kasutatakse SIMILAR TO operaatorit, mida MS Access ei toeta. PostgreSQLi tõstutundetu LIKE operaatori ILIKE kasutamisel seda viga välja ei tooda, sest seda võidakse kasutada tõstutundetuks otsinguks. Joonisel 13 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT *
FROM Hotell
WHERE linn LIKE 'Tallinn';
```

Joonis 13. *Näide SELECT lausest, kus on LIKE operaator ilma metamärkideta.*

2. **Otsingutingimuses: =NULL.** [16] Joonisel 14 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT *
FROM Hotell
WHERE linn = NULL;
```

Joonis 14. *Näide SELECT lausest, kus on ebaõige väärtuse puudumise kontroll.*

3. **Otsingutingimuses: <>NULL.** [16] Joonisel 15 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT *
FROM Hotell
WHERE linn <> NULL;
```

Joonis 15. *Näide SELECT lausest, kus otsingutingimuses on ebaõige väärtuse olemasolu kontroll.*

4. **Otsingutingimus: >NULL, >=NULL,<NULL, <=NULL.** [16] Joonisel 16 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT *
FROM Hotell
WHERE linn >= NULL;
```

Joonis 16. *Näide SELECT lausest, kus on NULLiga võrdlemine.*

5. **Otsingutingimus pole loogikaavaldis (nt hinne=1 OR 2).** [16] Joonisel 17 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT *
FROM Hotell
WHERE
        linn = 'Tallinn'
        OR 'Tartu'
        OR hind BETWEEN 8 AND 9;
```

Joonis 17. *Näide SELECT lausest, kus otsingutingimus pole loogikaavaldis.*

6. **Otsingutingimuses: <veerg> (LIKE|SIMILAR TO) <veerg>.**[16] Kui SQL lause otsingutingimuses soovitakse võrrelda ühte veergu teisega, peaks kasutama õiget operaatorit (=). LIKE ja SIMILAR TO operaatorid on mõeldud tekstilise väärtusega võrdlemiseks, kus on ka metamärkidest muster. Tegu on süntaktiliselt korrektse lausega, kuid tegu võib olla kas loogilise veaga või keerukusega seotud veaga. Selline kontroll tehti ka MS Accessi jaoks, kuid PostgreSQLi puhul lisati võimalus, et kasutatakse SIMILAR TO operaatorit, mida MS Access ei toeta. PostgreSQLi tõstutundetu LIKE operaatori ILIKE kasutamisel seda viga välja ei tooda, sest seda võidakse kasutada tõstutundetuks otsinguks. Joonisel 18 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT *
FROM Hotell
WHERE linn SIMILAR TO nimi;
```

Joonis 18. *Näide SELECT lausest, kus tekstiliste väärtuste võrdlemiseks võib olla kasutusel vale operaator.*

7. **Päringu tulemuses on rohkem kui üks sama nimega veerg.** [16] Joonisel 19 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT
        linn,
        nimi AS linn
FROM Hotell;
```

Joonis 19. *Näide SELECT lausest, kus päringu tulemuses on mitu sama nimega veergu.*

8. **SELECT klauslis on avaldis (sh konstant või funktsiooni poole pöördumine), kuid vastavale veerule pole ise antud nime.** [16] Joonisel 20 on näide SELECT lausest, kus vastav viga esineb mitu korda.

```
SELECT 'nimi', (
        SELECT Count(*)
        FROM Ruum WHERE
    Hotell.hotelli_nr=Ruum.hotelli_nr
)
FROM Hotell;
```

Joonis 20. *Näide SELECT lausest, kus tulemuses olevatele veergudele ei anta ise nimesid.*

9. **GROUP BY ja DISTINCT samal lause tasemel.** [16] Joonisel 21 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT DISTINCT nimi
FROM Hotell
GROUP BY nimi;
```

Joonis 21. *Näide SELECT lausest, kus on mitmekordne korduvate ridade eemaldamine.*

10. **ORDER BY <arv>.** [16] Joonisel 22 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT eesnimi
FROM Külaline
ORDER BY 1;
```

Joonis 22. *Näide SELECT lausest, kus sorteerimine toimub veeru positsiooni järgi, mitte nime järgi.*

11. **Kokkuvõttefunktsioone kasutava avaldise SUM()/COUNT() kasutamine kokkuvõttefunktsiooni AVG asemel.** [16] Joonisel 23 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT Sum(hind)/Count(*) AS keskmine
FROM Ruum;
```

Joonis 23. *Näide SELECT lausest, kus aritmeetilist keskmist ei arvutata vastavat kokkuvõttefunktsiooni kasutades.*

12. **SUM(1) kasutamine COUNT(*) asemel.** [16] Joonisel 24 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT Sum(1) AS arv
FROM Hotell;
```

Joonis 24. *Näide SELECT lausest, kus ridu loendatakse valesti.*

13. **INSERT lause INSERT klauslis puuduvad veerunimed.** [16] Joonisel 25 on näide INSERT lausest, kus vastav viga esineb.

```
INSERT INTO Hotell
VALUES (999, 'TOP', 'Tallinn');
```

Joonis 25. *Näide INSERT lausest, mis on liigselt tundlik sihttabeli struktuurimuudatuste suhtes.*

14. **SELECT * alampäringus, mille tulemuse põhipäringuga võrdlemiseks: IN/NOT IN/=ANY/<>ALL/=/<>/>/>=/</<= operaatorit.** [16] Joonisel 26 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT * FROM Külaline
WHERE külalise_nr IN (
        SELECT * FROM Reserveerimine
);
```

Joonis 26. *Näide SELECT lausest, kus alampäring ei leia andmeid kindlatest veergudest.*

15. **FROM klauslis rohkem kui üks tabel ja SELECT klauslis * (ilma täpsustava tabelita).** [16] Joonisel 27 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT * FROM hotell h
INNER JOIN reserveerimine r
    ON r.hotelli_nr = h.hotelli_nr
INNER JOIN külaline k
    ON k.külalise_nr = r.külalise_nr;
```

Joonis 27. *Näide SELECT lausest, kus SELECT klauselt ei täpsusta veerge.*

16. **OUTER JOIN + COUNT(*)** [16]. Joonisel 28 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT
        Hotell.hotelli_nr,
        nimi,
        Count(*) AS arv
FROM Hotell
LEFT JOIN Reserveerimine
    ON Hotell.hotelli_nr = Reserveerimine.hotelli_nr
GROUP BY Hotell.hotelli_nr, nimi;
```

Joonis 28. *Näide SELECT lausest, kus seotud ridade arvu loendamine teises tabelis toimub valesti.*

17. **{UNION|EXCEPT|INTERSECT} [ALL|DISTINCT] lauses SELECT *** [16] Joonisel 29 on näide SELECT lausest, kus vastav viga esineb. Selline kontroll tehti ka MS Accessi jaoks, kuid PostgreSQLi puhul lisati võimalus, et kasutatakse INTERSECT ja EXCEPT operaatoreid, mida MS Access ei toeta.

```
SELECT * FROM Ruum
UNION
SELECT * FROM Ruum_koopia;
```

Joonis 29. *Näide SELECT lausest, kus ühendi leidmise lauses pole veerud selgelt välja toodud.*

18. **Alampäringus ORDER BY (välja arvatud siis, kui seal on LIMIT või FETCH piirang).** [16] Kuna PostgreSQL ei toeta TOP predikaati, siis PostgreSQL jaoks rakendati kontroll LIMIT ja FETCH piiranguid arvestades. Joonisel 30 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT * FROM Hotell
WHERE hotelli_nr IN (
    SELECT hotelli_nr
    FROM Reserveerimine
    ORDER BY hotelli_nr
);
```

Joonis 30. *Näide SELECT lausest, kus alampäringus on ebavajalik sorteerimine.*

19. **INSERT .. SELECT *.** [16] Joonisel 31 on näide INSERT lausest, kus vastav viga esineb.

```
INSERT INTO Külaline_koopia
SELECT * FROM Külaline;
```

Joonis 31. *Näide INSERT lausest, mis on liigselt tundlik lähtetabeli struktuurimuudatuste suhtes.*

20. **Tekstilised väärtused jutumärkides (aga äkki identifikaator?).** [16] Joonisel 32 on näide SELECT lausest, kus vastav viga esineb. Samas ei pruugi selline lause olla vigane (vt joonis 33 - antud juhul on jutumärkides veeru nimi), kuid selliste olukordade eristamiseks peaks kontrollprogramm oskama arvestada tabelite struktuuriga, mida see praegu ei tee. Igal juhul on kasulik kasutajat võimaliku probleemi eest hoiatada.

```
SELECT * FROM Hotell
WHERE nimi = "Viru";
```

Joonis 32. *Näide SELECT lausest, kus tekstiline väärtus on jutumärkides.*

```
SELECT * FROM Hotell
WHERE "nimi" = "linn";
```

Joonis 33. *Näide SELECT lausest, kus veergude nimed on jutumärkides.*

21. **Otsingutingimuses nii AND (BETWEEN x AND y ei lähe arvesse) kui OR, kuid pole sulge.** [16] Joonisel 34 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT * FROM ruum
WHERE (
        ruumi_tüüp='Äriklassi tuba'
        AND ruumi_tüüp='Luksusnumber'
        OR hotelli_nr=2
);
```

Joonis 34. *Näide SELECT lausest, kus võibolla pole arvestatud tingimuste kontrollimise järjekorraga.*

22. **HAVING klauslis tingimus, mis ei sisalda kokkuvõttefunktsiooni tulemusega võrdlemist.** [16] Joonisel 35 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT
    ruumi_tüüp,
    hind
FROM ruum
GROUP BY ruumi_tüüp
HAVING hind > 10;
```

Joonis 35. *Näide SELECT lausest, kus HAVING klauslis on piirang, mis peaks olema WHERE klauslis.*

23. **{=|<>} <tekst kus on LIKE või regulaaravaldise muster>.** [22] Vastupidiselt probleemile nr 1, võib olla ka juhus, kus kasutaja kasutab tekstilise väärtusega võrdlusel metamärke, kuid operaator ei ole LIKE või regulaaravaldise mustrile vastavuse kontroll. Sellisel juhul võrreldakse teksti üks üheselt ning tegu võib olla nii loogilise kui ka keerukusega seotud veaga. Kuna PostgreSQL toetab erinevalt MS Accessist regulaaravaldiste kasutamist, siis täiendati PostgreSQLi kontrolli vastavalt. Joonisel 36 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT *
FROM Hotell
WHERE linn = '%linn';
```

Joonis 36. *Näide SELECT lausest, kus otsingutingimuses võib olla vale operaator.*

24. **SELECT klauslis kokkuvõttefunktsioon + veerud, kuid puudub GROUP BY.** [16] Joonisel 37 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT
        hotelli_nr,
        ruumi_nr,
        Count(külalise_nr) AS külalineCount
FROM Reserveerimine;
```

Joonis 37. *Näide SELECT lausest, kus puudub grupeerimine.*

25. **Kokkuvõttefunktsioon WHERE klauslis või GROUP BY klauslis.** [16] Joonisel 38 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT
        hotelli_nr,
        Count(*) AS arv
FROM Reserveerimine
WHERE Count(*) > 1;
```

Joonis 38. *Näide SELECT lausest, kus otsingutingimus on pandud valesse klauslisse.*

26. **=ALL kasutamine =ANY asemel.** [16] Joonisel 39 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT *
FROM Hotell
WHERE hotelli_nr = ALL (
        SELECT hotelli_nr
        FROM Reserveerimine
);
```

Joonis 39. *Näide SELECT lausest, kus otsingutingimuse kirjutamisel on ilmselt tehtud viga.*

27. **<>ANY|SOME kasutamine <>ALL asemel.** [16] Joonisel 40 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT *
FROM Hotell
WHERE hotelli_nr <>ANY (
        SELECT hotelli_nr
        FROM Reserveerimine
);
```

Joonis 40. *Näide SELECT lausest, kus otsingutingimuse kirjutamisel on ilmselt tehtud viga.*

4.5.2 Käesolevas töös realiseeritud kontrollid MS Accessi jaoks, mis realiseeriti ka PostgreSQL jaoks

1. **LIMIT/FETCH klausel ilma ORDER BY klauslita.** Jaotise 4.4 kontroll 1 juures on täpsem kirjeldus. Joonisel 41 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT *
FROM Hotell
LIMIT 5;
```

Joonis 41. *Näide SELECT lausest, kus on LIMIT klausel ilma ORDER BY klauslita.*

2. **Puuduv veerg GROUP BY klauslist.** Jaotise 4.4 kontroll 2 juures on täpsem kirjeldus. Joonisel 42 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT
    ruum_tüüp,
    hind,
    Count(hotelli_nr) as hotellCount
FROM Ruum
GROUP BY hind;
```

Joonis 42. *Näide SELECT lausest, kus GROUP BY klauslist on veerge puudu.*

3. **Puuduv ühendamise tingimus.** Jaotise 4.4 kontroll 3 juures on täpsem kirjeldus. Joonisel 43 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT
        Hotell.hotelli_nr,
        Reserveerimine.ruumi_nr
FROM Reserveerimine, Hotell
```

Joonis 43. *Näide SELECT lausest, kus puudub ühendamise tingimus.*

4. **Üleliigne semikoolon.** Jaotises 4.4 kontroll 4 juures on täpsem kirjeldus. Joonisel 44 on näide DELETE lausest, kus vastav viga esineb.

```
DELETE FROM Reserveerimine;;
```

Joonis 44. *Näide DELETE lausest, kus on üleliigne semikoolon.*

5. **WHERE või HAVING klauslis puuduv otsingutingimus.** Jaotises 4.4 kontroll 5 juures on täpsem kirjeldus. Joonisel 45 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT ruumi_tüüp
FROM Ruum
GROUP BY ruumi_tüüp
HAVING;
```

Joonis 45. *Näide SELECT lausest, kus HAVING klauslis puudub otsingutingimus.*

6. **LIMIT/FETCH klausel võib alampäringus ebasobivalt tagastada mitu rida.** Jaotises 4.4 kontroll 6 juures on täpsem kirjeldus. Joonisel 46 on näide SELECT lausest, kus vastav viga esineb. PostgreSQL lisanüanss, millega kontrollija ka arvestab, on selles, et kui alampäringus oleks FETCH FIRST ROW ONLY, FETCH FIRST 1 ROW ONLY või LIMIT 1, siis oleks see lause sobiv, sest alampäring tagastab siis maksimaalselt ühe rea. Antud näites kasutatav WITH TIES tagab, et tulemuses oleks kõigi nende külaliste numbrid, kelle reserveerimiste arv on kõige suurem (analoogiline tulemus MS Accessi TOP predikaadi kasutamisele).

```
SELECT * FROM Reserveerimine
WHERE külalise_nr = (
        SELECT külalise_nr
        FROM Reserveerimine
    GROUP BY külalise_nr
        ORDER BY Count(*) DESC
        FETCH FIRST 1 ROW WITH TIES
);
```

Joonis 46. *Näide SELECT lausest, kus FETCH'i kasutav alampäring võib tagastada ebasobivalt üle ühe rea.*

7. **LIMIT/FETCH klausliga kasutatakse ebakorrektset arvulist piirangut.** Jaotises 4.4 kontroll 8 juures on täpsem kirjeldus. Joonisel 47 on näide SELECT lausest, kus vastav viga esineb. PostgreSQL lisanüanss, millega kontrollija ka arvestab, on selles, et ridade arvu alapiir on 0. Sellisel juhul tagastatakse tühi ridade hulk.

Teine lisanüanss, millega kontrollija samuti arvestab, on see, et FETCH klauslis saab määrata ka nihke (*offset*) ja ka see peab olema suurem võrdne nullist.

```
SELECT külalise_nr
FROM Reserveerimine
ORDER BY külalise_nr DESC
LIMIT -1
```

Joonis 47. *Näide SELECT lausest, kus LIMIT piirang on ebaõige täisarv.*

8. **LIMIT/FETCH klauslit kasutatakse päringus, mis tagastab maksimaalselt ühe rea.** Jaotises 4.4 kontroll 11 juures on täpsem kirjeldus. Joonisel 48 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT Max(hotelli_nr) AS suurim
FROM Reserveerimine
LIMIT 1;
```

Joonis 48. *Näide SELECT lausest, kus LIMIT klauslit kasutatakse päringus, mis tagastab maksimaalselt ühe rea.*

9. **FROM klausli alampäringul puudub alias.** Jaotises 4.4 kontroll 10 juures on täpsem kirjeldus. Joonisel 49 on näide SELECT lausest, kus vastav viga esineb. Tuleb märkida, et kuni PostgreSQL 16 oli aliase puudumine süntaksi viga. Koodi loetavuse huvides peaks aliast ikkagi kasutama. [46]

```
SELECT Count(*) AS arv
FROM (
    SELECT DISTINCT kommentaar
    FROM Reserveerimine
);
```

Joonis 49. *Näide SELECT lausest, kus FROM klauslis oleval alampäringul puudub alias.*

10. **WHERE või HAVING võtmesõna kasutatakse päringus samal tasemel mitu korda.** Jaotises 4.4 kontroll 12 juures on täpsem kirjeldus. Joonisel 50 on näide SELECT lausest, kus vastav viga esineb.

```
SELECT ruumi_tüüp
FROM Ruum
GROUP BY ruumi_tüüp
HAVING Count(*)=1
OR HAVING Count(*)=2;
```

Joonis 50. *Näide SELECT lausest, kus on üleliigne HAVING võtmesõna.*

4.5.3 PostgreSQL-spetsiifilised kontrollid

1. **Alampäringus ja põhipäringus pole võrdluses sama arv veergusid.** [44] Tegemist on süntaksiveaga. Joonisel 51 on näide SELECT lausest, kus vastav viga esineb. Alampäring tagastab andmed kolmes veerus, kuid põhipäringus on võrdluses kaks veergu. MS Accessis saab võrdlust teha ainult ühe veeru põhjal.

```
SELECT * FROM Ruum
WHERE (hotelli_nr, ruumi_nr) IN (
    SELECT
        hotelli_nr,
        ruumi_nr,
        ruumi_tüüp
    FROM Reserveerimine
);
```

Joonis 51. *Näide SELECT lausest, kus otsingutingimus on valesti kirjutatud.*

4.6 Andmebaasisüsteemiga suhtlemiseks mõeldud draiveri valik

Andmebaasisüsteemiga suhtlemiseks on vajalik draiver või vahelüli, mis ühendaks omavahel kaks süsteemi. Töö raames uuris autor ODBC[25] ja OLEDB[26] draivereid, ning ka Npgsql-i[24], mis on otse integreeritud .NET raamistikku ja võimaldab samuti suhtlust PostgreSQL andmebaasisüsteemiga, ilma sealjuures draivereid kasutamata.

Kuna MS Accessi jaoks oli eelnevalt kasutusele võetud OLEDB draiver, siis alguses uuris autor ka selle kasutamist PostgreSQL andmebaasisüsteemiga suhtlemiseks. Paraku oli PostgreSQL OLEDB draiver tasuline, mistõttu otsustas autor esialgu realiseerida andmebaasisüsteemiga suhtluse ODBC draiverit kasutades.

ODBC draiverit kasutades esinesid aga mõningad probleemid. Suurim probleem tekkis sellest, kuidas vastendati andmebaasisüsteemist objektide tüübid C# objektideks. .NET raamistiku OLEDB teegis olid õigetele tüüpidele määratud õiged arvulised väärtused, mis olid vastavuses MS Accessi andmebaasist saadud väärtustega, kuid ODBC teegis need PostgreSQL andmebaasisüsteemist saadud väärtusetega vastavuses ei olnud. Lisaks tekkisid probleemid mõningate SQL lausete käivitamisega ODBC draiverit kasutades, mis olid siis kas draiveri põhjustatud või .NET ODBC teegi põhjustatud. Igal juhul ei leidnud autor sellele lihtsat lahendust.

Samuti oleks ODBC draiveri kasutamine olnud kasutaja jaoks ebamugav, kuna selle peaks

pistikprogrammist eraldiseisvalt arvutisse paigaldama

Eelnevalt toodud põhjustel otsustas autor ODBC draiveri välja vahetada Npgsql lahenduse vastu. Selle korral ei pea arvutisse paigaldama eraldi draiverit kuna kogu suhtlus toimub läbi .NET raamistiku. Seeläbi lahenesid ka lausete käivitamisel tekkinud vead ning ka objektide vastendamise probleem.

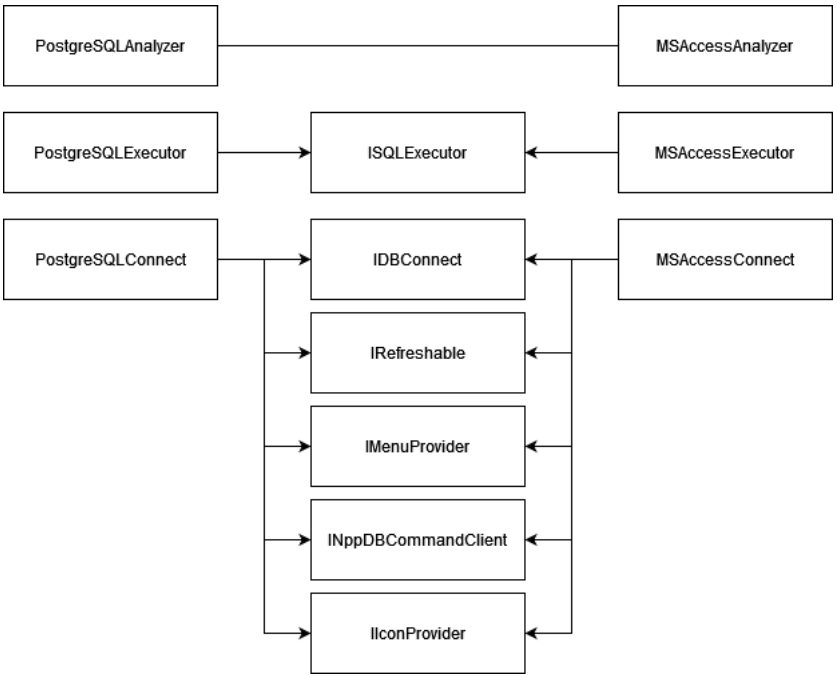
4.7 Tagarakenduse realisatsioon

Käesolevas jaotises kirjeldab autor täpsemalt, kuidas arendati tagarakendust ning kuidas see töötab.

4.7.1 PostgreSQL mooduli lisamise protsess

Enne uue andmebaasisüsteemi mooduli lisamist oli vaja olemasolev pistikprogramm kompileerida. Selleks oli vaja projekti lisada ANTLR generaatori .jar fail, millega projekt genereerib grammatikafailidest vastavad C# Lexer ja Parser klassid. Lisaks olid eelnevast arendusest jäänud projekti lähtekoodi lisamata osad andmebaasiühenduse alt avanevas vaates kasutatavad ikoonid. Seega alguses pidi need koodist välja kommenteerima. Hiljem saadi need ikoonid tarkvara eelmise versiooni arendaja käest, misjärel sai projekti algse seisu taastada.

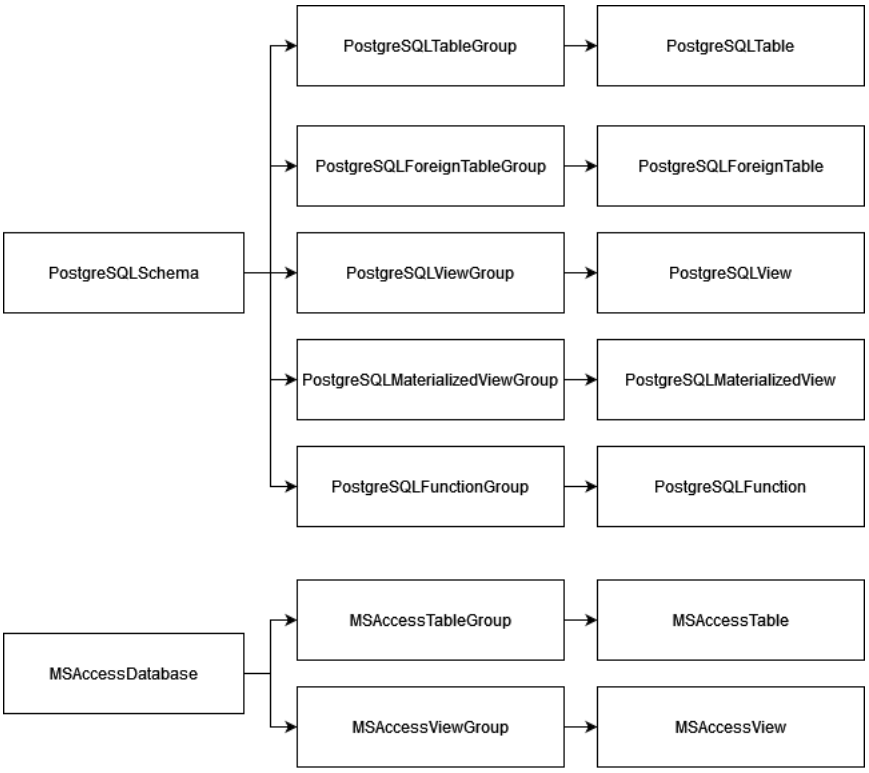
Notepad++ pistikprogramm NppDB on ehitatud modulaarselt. Seega uue andmebaasisüsteemi (antud juhul PostgreSQL) mooduli lisamiseks tuli luua õigete failide ja atribuutidega uus .NET projekt, et põhikomponendid oleksid võimelised tuvastama uut moodulit. Joonisel 52 on välja toodud suuremad komponendid, mis olid realiseeritud MS Accessi jaoks ja mille pidi PostgreSQL mooduli lisamiseks samuti realiseerima. Keskmises veerus on välja toodud liidesed, mida komponendid laiendavad. Küll aga võivad komponentide sisud erineda, sõltuvalt andmebaasisüsteemist ja kasutatud draiverist. Näiteks pidi *PostgreSQLExecutor* komponenti, mis vastutab ka SQL päringu vastusest andmete lugemise eest, kirjutama kohandatud loogika rahaliste väärtuste ja aja õiges vormingus kuvamiseks, mis hõlmas endas iga rea ja veeru eraldi töötlemist.



Joonis 52. *MS Access ja PostgreSQL komponendid, mille struktuur on sarnane ja mis on vajalikud uue mooduli lisamiseks.*

Pistikprogrammi abil peab olema võimalik töötada PostgreSQL andmebaasisüsteemiga, ehk tähtis on andmebaasiühenduse loomine, andmebaasi struktuurielementide nägemine ja SQL lausete käivitamine ning vigade esinemise suhtes analüüsimine.

Andmebaasi struktuurielementide esitamiseks pidi kirjutama MS Accessi moodulist erineva lahenduse, kuna raamistikesse sisseehitatud meetodid ei olnud piisavalt võimekad. Seetõttu oli vaja kirjutada eraldiseisvad SQL laused andmebaasiobjektide kohta info lugemiseks PostgreSQL andmebaasi süsteemikataloogist (vt lisa 3 joonis 69 ja 70). Veel pidi PostgreSQL andmebaasi struktuurielementide esitamiseks lisama rohkem komponente, kui MS Access mooduli korral, sest PostgreSQL moodulis kuvatakse rohkem erinevat tüüpi struktuurielemente. Joonisel 53 on välja toodud mõlema mooduli komponendid andmebaasi struktuurielementide esitamise.



Joonis 53. *MS Access ja PostgreSQL andmebaaside struktuurielementide esitamise komponendid.*

Andmebaasiga ühenduse loomiseks arendati välja uus vorm, kus saab PostgreSQL andmebaasiga ühenduse loomiseks täita vajalikud väljad (vt lisa 2 joonis 62).

Uue andmebaasi toe lisamine, ilma SQL lausete kontrollideta, ei olnud eriti keeruline, kuid keerukust lisas eraldi SQL lausete kirjutamine andmebaasi struktuurielementide esitamiseks ning SQL lause vastusest saadud andmete töötlus.

4.7.2 PostgreSQL SQL lausete kontrollide lisamise protsess

SQL lausete käivitamisel ja analüüsimisel kasutati lausete üksteisest eristamiseks ANTLR poolt genereeritud parserit. Lausete eristamiseks kasutati lauset lõpetavad märki ning kasutusel olevad Lexer ja Parser failid suutsid iseseisvalt välja filtreerida rea või ploki kommentaarides asuvad lauset lõpetavad märgid. Ainus probleem lausete eristamisel tekkis siis, kui lause sees oli rohkem kui üks lauset lõpetav märk, millest autor kirjutab lähemalt jaotises 4.9.

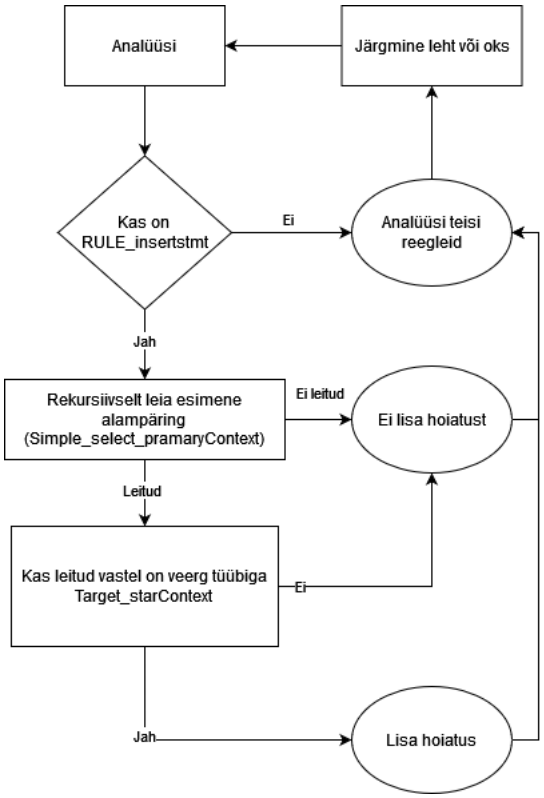
Lausete analüüsimiseks kasutatakse samuti ANTLR poolt genereeritud süntaksipuud. Kuna valitud PostgreSQL grammatikafailid on väga detailsed, siis on ka lihtsamatele SQL lausetele genereeritud süntaksipuud väga suured ja detailsed (vt lisa 4 joonis 73). Post-

greSQL grammatikafailid[27] võeti ametlikust ANTLR4 jaoks mõeldud grammatikafailide koodihoidlast. Selliste süntaksipuude programmiliselt analüüsimiseks ja sealt vigade leidmiseks pidi ehitama mitmeid väga sügavaid rekursiivseid meetodeid, mis käivad terve puu läbi ning erinevaid kohti analüüsides suudavad tuvastada, kas lauses on vigu või mitte. Selleks, et erinevate vigade tuvastamist hõlbustada, pidi autor ka pidevalt grammatikafaile täiendama. Lisas 4 joonisel 76 välja toodud grammatikafaili osal on näha, et avaldiste operaatoritele on ette lisatud *operands*+= ning avaldiste erinevatele pooltele on lisatud kas *lhs*=, mis tähistab avaldise vasakut poolt, või *rhs*=, mis tähistab avaldise paremat poolt. Selliste täiendustega, on programmiliselt lihtsustatud avaldiste operaatorite ja poolte leidmine, et neid analüüsida saaks.

PostgreSQL SQL lausete analüüsimiseks sai taaskasutada MS Access moodulist üleüldist loogikat, et genereeritud süntaksipuu lehti ja oksi rekursiivselt läbi käia, kuni jõutakse lauset lõpetava sümbolini. Küll aga pidi iga lisatud kontrolli jaoks kirjutama teistsuguse loogika, sest PostgreSQL ja MS Accessis kasutusel olevad grammatikafailid on väga erinevad.

INSERT SQL lausele **INSERT .. SELECT *** kontrolli lisamine on suhteliselt lihtne. Joonisel 31 on välja toodud kontrollitav SQL lause.

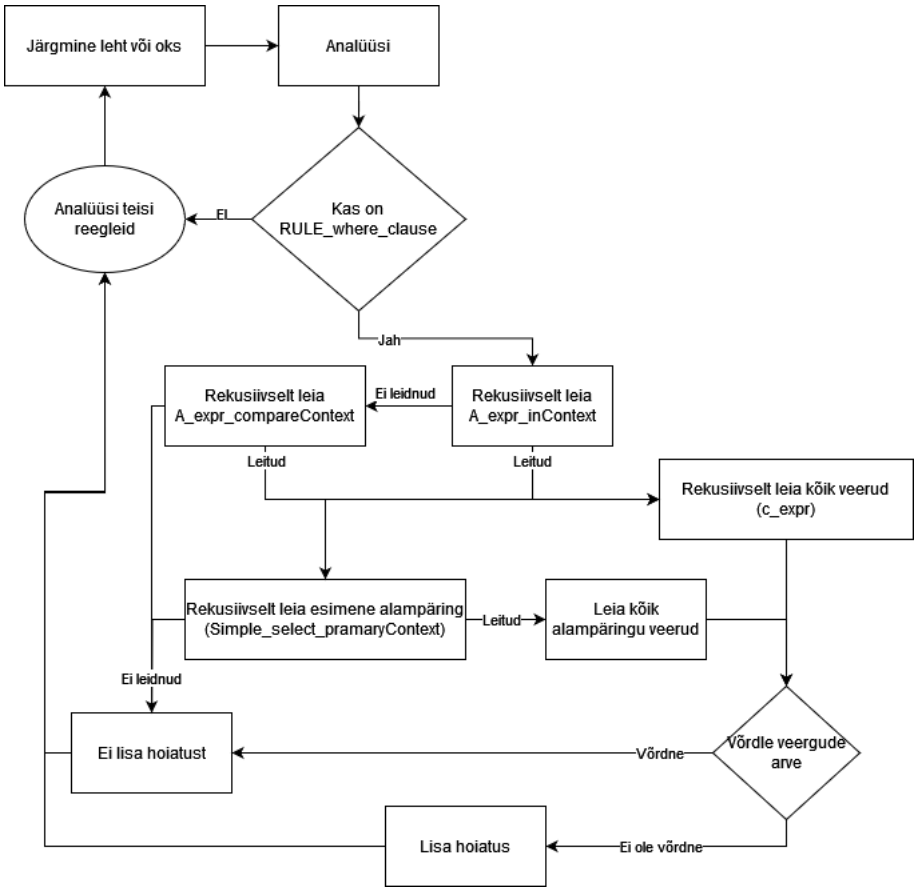
Kui SQL lauset analüüsides tuvastab programm, et tegemist on INSERT lausega, siis tuleb seal otsida üles SELECT lause. Kui SELECT lauset ei leita (lauses on ridade moodustamiseks VALUES lause), siis ei saa ka viga tuvastada. Kui SELECT lause on leitud, siis tuleb kindlaks teha, kas seal on SELECT klauslis kasutusel * sümbol veergude tähistamiseks. Kui selline sümbol on veergude tähistamiseks kasutusel, siis tuleb tagasta hoiatus selle SQL lause kohta. Kui * on kasutusel korrutamise operaatori nimena, siis ei tagastata kasutajale hoiatust. Joonisel 54 on näha diagramm, kuidas selle SQL lause kontroll töötab ning joonisel 74 on näha ANTLR poolt genereeritud süntaksipuu.



Joonis 54. ***INSERT .. SELECT **** *kontrolli realiseerimine.*

PostgreSQL spetsiifilise kontrolli **Alampäringus ja põhipäringus pole võrdluses sama arv veergusid** lisamine on aga juba keerulisem. Joonisel 51 on välja toodud kontrollitav SQL lause. Järgnevalt kirjeldatakse selle kontrolli protsessi.

Kui SQL lauset analüüsides, tuvastab programm, et lauses on kasutusel WHERE klausel, siis tuleb sealt leida avaldis IN operaatoriga. Kui IN operaatoriga avaldist ei leita, peab otsima avaldist, kus on kasutusel võrdlus. Peale avaldise leidmist peab rekursiivselt leidma kõik operaatorist vasakul pool olevad veerud, ning operaatorist paremal pool oleva alampäringu ja alampäringus kasutusel olevad veerud. Nende kahe veergude arve võrreldes, saab kasutajale kas hoiatuse kuvada või mitte. Joonisel 55 on näha diagramm, kuidas selle SQL lause kontroll töötab ning joonisel 75 lisas 4 on näha ANTLR poolt genereeritud süntaksipuu. Sellist viga suudab programm tuvastada sõltumata sellest, millist tüüpi SQL lauses on WHERE klausel kasutusel, kas SELECT, INSERT, DELETE, UPDATE või muud tüüpi laused. Viga võib olla ka mitme alampäringu sügavusel, kuid programm suudab seda ikka tuvastada.



Joonis 55. ***Alampäringus ja põhipäringus pole võrdluses sama arv veergusid*** *kontrolli realiseerimine.*

Kõikidest kontrollidest kõige keerulisem oli realiseerida kontroll veale **Otsingutingimus pole loogikaavaldis (nt hinne=1 OR 2)**. Kontrolli loomiseks pidi analüüsima ANTLR genereeritud süntaksipuu SQL lause otsingutingimuses iga elementi *a_expr_<tähistus>* rekursiivselt. Selliste süntaksipuu elementide ehitust on näha joonisel 73 lisas 4. Kuna iga elemendi jaoks ei kirjutatud eraldi kontrolli, vaid lahendus üldistati, siis peab leidma avaldisest operaatorid ning avaldise pooled dünaamiliselt, teadmata, millise objektiga tegu on, ning kas vastaval objektil on avaldis, mille pooli saame kontrollida. Lisaks pidi arvestama avaldistega, kus võib olla ühel tasemel mitu operaatorit, nagu tingimuses *BETWEEN x AND y*. Kui avaldise pooled ja operaatorid oli leitud, pidi nende seast tuvastama, kas avaldises on operaatorite ja poolte arv korrektne. Lisaks pidi enne võrdlusest eemaldama ebakorrektsed avaldised, kus ei ole kasutatud operaatorit.

Töö kõige keerulisem osa oli PostgreSQL SQL vigade kontrollide realiseerimine. See nõudis väga palju rekursiivset mõtlemist ning SQL lausete struktuuri analüüsimist, et oleks võimalik realiseerida SQL lause kontroll ka koodis. Kokku on 38 PostgreSQL kontrolli realiseerimiseks kirjutatud ligikaudu 1300 rida koodi, mis on mõeldud ainult SQL lausete

analüüsimiseks.

4.8 Kasutajaliides

Käesoleva töö käigus valminud pistikprogrammi täiendus kasutab olemasolevat Notepad++ ja NppDB kasutajaliidest, mida täiendati vastavalt vajadusele.

- PostgreSQL andmebaasisüsteemiga ühenduse loomiseks tehti uus aken, mida kasutatakse nii täiesti uue ühenduse loomisel kui ka eelnevalt salvestatud ühenduse taasavamiseks. Kui luuakse täiesti uus ühendus, siis on kõik lahtrid peale pordi tühjad. Pordi puhul pakutakse välja vakimisi PostgreSQL port - 5432, kuid ka seda saab soovi korral muuta. Valikuliselt saab määrata ühenduse nime. Kui ühenduse nimi puudub, siis kasutatakse ühenduse nimena andmebaasi nime (vt lisa 2 joonis 57).

  Eelneva ühenduse taastamisel on kõik lahtrid peale parooli täidetud eelnevate andmetega, valikuline ühenduse nime lahter on peidetud ning parooli lahter on kohe aktiivne - selle peal ei pea eraldi klõpsama (vt lisa 2 joonis 58).
- Käivitatud SQL lause täitmise ebaõnnestumise korral kuvatakse veateade logi sakis rasvases kirjes, et veateated oleksid seal lause õnnestumise teadetest kergemini eristatavad (vt lisa 2 joonis 59).
- Käivitatud SQL lause tulemuste saki päises olevad järjekorranumbrid olid iga päringu puhul peidetud. Alles esimese veeru lahtri suurust muutes oli näha, et seal on järjekorranumbreid. Parandustega on järjekorranumbrid alati näha ning veeru lahtri suurus muutub vastavalt sellele kui mitme kohaline on kõige suurem järjekorranumber. (vt lisa 2 joonis 60).
- Kui lisada mitu samanimelist andmebaasiühendust, siis üritas programm lisada ühenduse nimele sulgudesse järjekorranumbri, et ühendusi oleks võimalik eristada. Paraku see ei töötanud korrektselt, sest muster, millega üritati samanimeliste ühenduste lõpust välja lugeda järjekorranumber, ei olnud selleks suuteline ja tekkis ootamatu viga, mis kuvati ka kasutajale. Parandatud lahendus ei kasuta enam keerulist mustrit nimede analüüsimiseks, vaid analüüsitakse ühenduste nimede algust. Parandustega saavad samanimelised ühendused nime lõppu järjekorranumbri.
- Pistikprogramm suudab vastavalt Notepad++ programmis valitud keelele muuta SQL lausete veateadete tõlget. Toetatud on eesti- ja inglise keelsed tõlked (vt lisa 2 joonis 61).
- PostgreSQL andmebaasiobjektide kontekstmenüüde kaudu saab täita andmebaasis erinevaid lauseid. Kustutamisel kasutatakse RESTRICT omadust e kui leidub kustutatavast objektist sõltuvaid objekte, siis kustutamine ebaõnnestub.
  - Baastabel (vt lisa 2 joonis 62)

    * SELECT * FROM */tabeli_nimi/*
    * DROP TABLE
  - Väline tabel (vt lisa 2 joonis 63)
    * SELECT * FROM */tabeli_nimi/*
  - Vaade (vt lisa 2 joonis 64)
    * SELECT * FROM */vaate_nimi/*
    * DROP VIEW
  - Materialiseeritud vaade (vt lisa 2 joonis 65)
    * SELECT * FROM */materialiseeritud_vaate_nimi/*
    * REFRESH MATERIALIZED VIEW
    * DROP MATERIALIZED VIEW
  - Funktsioonid (vt lisa 2 joonis 66)
    * DROP FUNCTION

4.9 Testimine

Käesoleva töö raames valminud tarkvara testiti pidevalt töö autori ja juhendaja poolt. SQL lausete vigade kontrollide testimiseks oli kasutuses nii MS Accessi kui ka PostgreSQL jaoks eraldi failid, mis koosnesid erinevat tüüpi SQL lausetest, kus esines kõiki kontrollitavaid vigu ning ka SQL lauseid, kus programm võis vigu valesti tuvastada. Joonisel 72 lisas 3 on näide SQL failist, millega SQL lausetes vigade kontrolli kontrolliti.

Töö käigus ja valminud arendust testides ilmnesid mitmed lahendamist vajavad probleemid, millest järgnevalt esitatakse mõned näited. Need probleemid lahendati töö käigus jooksvalt.

- Leidub keerulisemad laused, kus võib lause sees olla lauset lõpetav märk, samas kui SQL lause on süntaktiliselt korrektne (näiteks funktsiooni loomise lause, mille alamosade lõpus on semikoolonid). Programm tuvastas esimese lause lõpetava märgi lause lõpuna, mistõttu tükeldati üks terviklik lause mitmeks erinevaks, ning üritati neid järjest käivitada. Selle jaoks pidi tegema erandid kindlate lause tüüpide jaoks, kus sellised olukorrad võivad tekkida, et lauset liiga varakult ei lõpetataks. Samas tuli tagada ka see, et kui lauset lõpetav märk ei ole süntaktiliselt õiges kohas lause sees, siis see tuvastatakse ikka kui viga. Samuti pidi ütlema Npgsql raamistikule, et see tagataustal lauseid uuesti ümber ei kirjutaks, kuna seal esines sama viga (vt lisa 3 joonis 67).
- Notepad++ aknas, mille taga loodi ühendus andmebaasiga, loodi igal lause käivitamisel uus sessioon. Seega ei saanud näiteks ühes aknas käivitada transaktsiooni plokki kuuluvaid lauseid nii, et transaktsiooni omadused oleks tagatud. Seetõttu pidi muutma ühenduste loomise loogikat, et ühes aknas kasutataks alati sama sessiooni.

- Rakenduses olev andmebaasist lahti ühendamise nupp ei sulgenud kõiki avatud ühendusi, mistõttu jäid tagataustal mitmed ühendused aktiivseks. See takistas andmebaasi kustutamist enne kui Notepad++ rakendus oli suletud. Sellega seoses muudeti lahtiühendamise loogikat. Siiski, kui ühe andmebaasiga on loodud mitu ühendust, siis ei suleta kõiki ühendusi sama andmebaasiga, kui need ühendused on andmebaasi ühenduse halduris üksteisest eraldi salvestatud. Kõikide ühenduste katkestamiseks peab kõik eraldi salvestatud ühendustest lahti ühendama.
- Kasutusel olevad ANTLR grammatika reeglid PostgreSQL jaoks, ei toetanud arvulisi väärtusi, mis olid kuueteistkümnendformaadis või kus olid alakriipsud, kuigi PostgreSQL versioon 16 lisandus sellisel viisil väärtuste esitamise tugi. Selle jaoks pidi grammatika faili lexer failis arvude definitsiooni muutma (vt lisa 3 joonis 68).
- Loodud lahendus kuvamaks andmebaasiühenduse halduris tabeleid, vaateid, hetktõmmiseid, funktsioone, veerge ja kitsendusi ei toetanud erilisi tabelite nimetusi (nt tabeli nimetus "12345678901234567890123456789012345678901234567890123456789012 34567890123"), ning andis veateateid, kui üritati tabeliga tööd teha. Seetõttu pidi lahenduse ümber kirjutama, et selliseid vigu ei tekiks ning vastavate andmebaasiobjektidega saaks vigadeta töötada.
- Pistikprogramm suudab vastavalt Notepad++ programmis valitud keelele muuta SQL lausete veateadete tõlget. Selles osas sõltus programm failist "nativeLang.xml", mille Notepad++ genereeris alles siis, kui kasutaja programmi keelt vahetas. Kui pistikprogramm paigaldada värskelt paigaldatud Notepad++-i korral, siis tekiks viga, mis ei luba pistikprogrammil käivituda, kuna "nativeLang.xml" faili ei ole veel tekitatud. Selle lahendamiseks on pistikprogrammi tõlkekeel vaikimisi inglise keel, kui "nativeLang.xml" faili ei leita.
- Andmebaasiühenduse halduri kontekstmenüüs salvestatud ühendusel parempoolse hiireklahvi vajutamisel ei muudetud valikut aktiivseks ja programm võis avada vale ühenduse menüüd. Peale parandusi muudetakse alati aktiivseks just valitud ühendus ning kuvatakse õige andmebaasi info.
- Andmebaasiühenduse halduri kontekstmenüüdes lõigati mõningatel juhtudel menüüelementidel kuvatud tekst ära liiga varakult. Peale parandust näidatakse iga menüüelemendil kuvatud teksti alati täies pikkuses.
- Kui avatud aknas oli mitu SQL lauset, aga vigade leidmiseks analüüsiti ainult ühte, siis viga näidati vales kohas. Nüüd arvestab programm vea kuvamisel üksinda analüüsitud lause asukohta.
- Funktsioonide kuvamisel ei näidatud nende parameetreid, kui parameetrid olid loodud ilma neile nime andmata.

Lisaks käsitsi testimisele lisati pistikprogrammi ka ühiktestid, mis kontrollivad programmi võimet SQL lausetest vigu tuvastada, ja integratsioonitestid, mis tagavad programmi

korrektse töötamise päris PostgreSQL andmebaasi vastu. Testide käivitamiseks kasutatakse xUnit.net [30] tarkvara.

Ühiktestides antakse programmile ette nimekiri SQL lausetest ning igal SQL lausel on ka nimekiri vigadest, mida programm peaks lauses tuvastama. Iga SQL lauset käivitatakse eraldiseisvalt pistikprogrammi SQL lauseid analüüsiva osa vastu, kust vastuseks antakse nimekiri programmi tuvastatud vigadest. Seejärel kontrollitakse, et sisendis olevad vead oleksid täpselt samad nende vigadega, mis programm ühiktestile tagastab. Kui mingi osa ei ole võrdne, siis loetakse ühiktest läbikukkunuks ja arendaja peab probleemse koha koodis tuvastama ning parandama. Sellised ühiktestid lisati iga lisatud või muudetud SQL lause kontrollile, nii MS Access moodulisse kui ka PostgreSQL moodulisse.

Integratsioonitestides seatakse alati Testcontainers abil üles uus PostgreSQLi andmebaas. Kui andmebaas on käivitunud, siis loetakse programmi sisse tekstifail, mis sisaldab suurel hulgal korrektseid PostgreSQL SQL lauseid. Kogu faili sisu saadetakse pistikprogrammi PostgreSQL mooduli samasse kohta, kuhu ka kasutaja poolt Notepad++ tekstiredaktoris käivitatud SQL laused jõuaksid. Kui vähemalt üks töödeldud ja käivitatud SQL lause tagastab vea, siis loetakse integratsioonitest läbikukkunuks ja arendaja peab probleemse koha koodis tuvastama ning parandama. Selliste testidega tagatakse programmi üleüldine PostgreSQL SQL lausete käivitamise ja ka andmebaasi ühenduvuse funktsionaalsus.

4.10 Teadaolevad puudused

Käesolevas jaotises tuuakse välja pistikprogrammis teadaolevad puudused.

Kuigi SQL lauseid saab analüüsida ilma andmebaasist päringut tegemata, siis praeguse lahendusega peab olema loodud ühendus vastava andmebaasisüsteemi mistahes andmebaasiga, kuna selle järgi saab programm aru, milliste reeglite kohaselt tuleb SQL lauseid analüüsida. Tulevaste arenduste käigus peaks ühenduse puudumisel veateate asemel pakkuma kasutajale valiku, millise andmebaasisüsteemi keelereeglite järgi tuleb analüüsi teha.

Pistikprogrammi arendatud PostgreSQL moodulis ja täiendatud MS Access moodulis pole realiseeritud kõikvõimalike vigade kontrolle. Kontrolle võiks rohkem olla ning andmebaasiühenduse olemasolu puhul saaks kontrollida näiteks ka tabelite ja veergude nimede korrektsust. Kui programm on teadlik andmebaasi struktuurist, siis saaks lisada ka abistava funktsionaalsuse, mis pakub SQL lausete kirjutamise ajal tabelite ja veergude nimesid, mida SQL lauses kasutada saaks.

4.11 Installeerimine

Käesoleva töö raames loodud pistikprogrammi PostgreSQL mooduli kasutamiseks peab arvutis olema järgmine tarkvara:

- Notepad++ 64-bitine versioon,
- Notepad++ pistikprogramm NppDB, ning NppDB kompileeritud failid peavad asetsema järgnevates kaustades, kuhu Notepad++ on paigaldatud:
  - "NppDB.Comm.dll" peab paiknema failiga "notepad++.exe" samas kaustas.
  - "NppDB.dll" ja "NppDB.PostgreSQL.dll" koos tõlkefailidega peavad olema kaustas ". . . /plugins/NppDB"

Käesoleva töö raames täiendatud pistikprogrammi MS Access mooduli kasutamiseks peab arvutis olema järgmine tarkvara:

- Notepad++ 64-bitine versioon
- MS Access Database Engine 2010 Redistributable (draiver) [47]
- Notepad++ pistikprogramm NppDB, ning NppDB kompileeritud failid peavad asetsema järgnevates kaustades, kuhu Notepad++ on paigaldatud:
  - "NppDB.Comm.dll" peab paiknema failiga "notepad++.exe" samas kaustas.
  - "NppDB.dll" ja "NppDB.MSAccess.dll" koos tõlkefailidega peavad olema kaustas ". . . /plugins/NppDB"

4.12 Litsenseerimine ja avalikkusega jagamine

Käesoleva lõputöö raames täiendatud tarkvara on tasuta ja avatud lähtekoodiga. Pistikprogramm NppDB on varasemate autorit poolt välja antud MIT litsentsiga [48]. Kuna käesoleva lõputöö raames arendatud tarkvara on pistikprogrammi NppDB edasiarendus, ning tarkvara võib vaja olla tulevaste autorite poolt samuti edasi arendada, siis on samamoodi mõistlik määrata arendatud tarkvara litsentsiks MIT.

MIT litsents annab igale isikule, kes saab MIT litsentsiga tarkvara lähtekoodi või dokumentatsiooni, õiguse kasutada tarkvara ilma piiranguteta.

Käesoleva bakalaureusetöö tulemus on avalikkusele kättesaadav GitHub koodihoidlast aadressitl https://github.com/aneelm/NppDB.

5. Tulemuste valideerimine

Käesolevas peatükis antakse ülevaade töö tulemuste valideerimisest. Töö valideerimiseks võrreldi pistikprogrammi funktsionaalsusi erinevate teiste sarnase funktsionaalsusega programmidega. Veel valideeriti programmi funktsionaalsust käsitsi nii autori kui ka juhendaja poolt.

5.1 PostgreSQLis SQL lausete käivitamise funktsionaalsuse võrdlemine olemasolevate vahenditega

Käesolevas jaotises on toodud välja mõningad programmid, mis on suutelised PostgreSQL andmebaasidega töötama. Programme, mis on suutelised PostgreSQL andmebaasidega töötada on väga palju, mistõttu valiti funktsionaalsuste võrdlemiseks välja üksikuid kõige populaarsemate seast [49]–[51].

Järgnevas loetelus toodud programmide korral käivitati igas programmis täpselt samu SQL lauseid, mille seal olid koos kommentaaridega CREATE, ALTER, DROP, DELETE, SELECT, INSERT, UPDATE, COMMENT ja SET laused. Joonisel 71 ja 67 on välja toodud üksikud laused, mida kasutati.

- pgAdmin 4 (v7) võimaldas käivitada kõiki ette antud süntaktiliselt korrektseid SQL lauseid, kuid mitme lause korraga käivitamisel näitas ainult viimase tulemusi.
- DbGate (5.2.7) ei võimalda käivitada keerulisi SQL lauseid, kus lause keskel võib esineda lauset lõpetav märk, kuid mis on süntaktiliselt korrektsed (vt lisa 3 joonis 67). Programm näitas iga käivitatud lause kohta detailset infot nagu muudetud ridade arv ja kui kaua lause täitmine aega võttis. Kui laused tagastasid andmeid, siis iga lause tulemust näidati eraldi sakis.
- JetBrains DataGrip (2023.2.5) võimaldas käivitada kõiki lauseid, isegi kui programm hoiatas, et lause pole süntaktiliselt korrektne, nt lause, kus arv on alakriipsudega esitatud. Igale käivitatud lause kohta anti detailne info ning iga andmeid tagastava lause jaoks tehti uus sakk, kus on võimalik tulemusi näha.
- DBeaver (23.3.0, tasuta versioon) oli võimeline käivitama kõiki SQL lauseid üksikult, kuid probleemid tekkisid kui üritati kõiki lauseid korraga käivitada. Programm ei ole võimeline mitut keerulisemat lauset korraga käivitama, kuid kui keerulisemaid lauseid ühekaupa käivitada, siis töötasid ka need. Programm iga käivitatud lause kohta täpset infot ei anna, vaid teeb kokkuvõtte kui mitut lauset käivitati ning kui

kaua see aega võttis. Andmeid tagastavate lausete korral avab programm samuti iga tulemuse jaoks eraldi saki.

- DbVisualizer (23.2.5, tasuta versioon) ei olnud võimeline käivitama keerulisemaid SQL lauseid, kus esines lause sees lauset lõpetav märk, isegi kui SQL lauseid käivitati ilma, et programm lauseid üksteisest eristada proovib ehk skriptina. Programm andis iga käivitatud lause kohta väga detailset infot ning avas ka iga andmeid tagastanud SQL lause jaoks eraldi saki. Võrreldes infoga, mida näitab käesoleva töö raames arendatud pistikprogrammi kasutajale, näitas DbVisualizer lisa infona ainult jooksutatud lause tüüpi, näiteks kas tegemist on SELECT, UPDATE või DELETE lausega.
- Beekeeper Studio (4.0.3) suutis kõiki lauseid probleemideta käivitada. Programm salvestas ka sessioonis kõikide käivitatud lausete tulemused, ehk rippmenüüst sai iga lause tagastatud andmeid vaadata. Samas ei andnud programm detailset infot üksikute lausete käivitamise kohta.

Käesolevas töös edasiarendatud pistikprogramm suutis käivitada kõiki neid lauseid - nii ükshaaval kui skriptina. Käivitamisel näidatakse käivitamise aega, vea tekkimisel veateadet ning kui lause tagastab andmeid, siis avatakse need eraldi sakis. Käivitamise aeg on näha ka saki nimes.

5.2 PostgreSQLi SQL lausete kontrollimise funktsionaalsuse võrdlemine olemasolevate vahenditega

SQL vigade kontrollimiseks otsiti programme, millel on sarnane lausete kontrollimise funktsionaalsus. Selles jaotises tuuakse välja leitud programmid ning milliseid vigu nad on võimelised kontrollima.

Tulemuste valideerimiseks kasutati samu lauseid, mida kasutati pistikprogrammi lausete kontrollide loomisel. Need SQL laused on välja toodud jaotistes 4.4 ja 4.5.

Jetbrains DataGrip (versioon 2023.2.5) suutis tuvastada järgnevaid probleeme:

- = NULL kasutamine IS NULL asemel.
- >= NULL ja <= NULL kasutamine otsingutingimustes - need pole kunagi täidetud e tulemuseks on 0 rida.
- <> NULL kasutamine IS NOT NULL asemel.
- HAVING klausel ilma kokkuvõttefunktsioonita.
- WHERE või GROUP BY klauslis kokkuvõttefunktsioon.

- Grupeerimist sisaldava lause SELECT klauslis on veerg, mis ei ole kokkuvõttefunktsiooni argument või ei esine GROUP BY klauslis.
- FROM klausli alampäringul puudub alias.
- Veel suutis programm tuvastada erinevaid süntaksivigu, näiteks topelt WHERE või HAVING klausel lauses ning HAVING ja WHERE klauslis puuduv otsingutingimus.

SQLFluff (versioon 2.3.5) ei suutnud testimiseks kasutatud lauseid lugeda, kuna ei toeta nimedes e identifikaatorites täpitähtede kasutamist samas kui käesolevas töös arendatud pistikprogramm nende kasutamist lubab. Kui täpitähed asendada, siis lause ehitus jääb siiski samaks ja SQLFluff suutis muudetud lausetes tuvastada järgnevaid vigu.

- Kasutatakse SELECT * ehk kasutajale ei ole teada, mitu veergu lause tulemusse tekib. SQLFluff aga ei suuda tuvastada sama viga, kui SELECT * on kasutuses alampäringus.
- DISTINCT võtmesõna kasutamine koos GROUP BY fraasiga SELECT lauses e korduste mitmekordne eemaldamine.
- Kasutatakse SELECT * lauses, kus päritakse andmeid mitmest tabelist, ilma SELECT klauslis täpsustamata, millisest tabelist kõiki andmeid soovitakse saada.
- SELECT klauslis konstandi või avaldise kasutamine ilma nendele aliase määramiseta e päringu tulemuses vastava veeru nime määramiseta.
- SELECT klauslis kasutatakse korduvalt sama aliast.
- = NULL või <> NULL kasutamine IS NULL või IS NOT NULL asemel.
- SQLFluff leidis ka süntaktilised vead nagu topelt WHERE või HAVING klausel lauses ning HAVING ja WHERE klauslis puuduv otsingutingimus.

Küll aga ei saa SQLFluff aru arvuliste väärtuste esitusest kuueteistkümnendformaadis või alamkriipse kasutades ning loeb neid vigadeks.

SQLFluff kontrollis ka selliste probleemide esinemist, mida käesolevas töös arendatud pistikprogramm praegu ei kontrolli. Enamik nendest on seotud SQL koodi loetavusega.

- SQL lausete võtmesõnad, nagu SELECT, UPDATE ja WHERE, ei ole suurtähtedega kirjutatud.
- Jutumärkideta identifikaatorid, nagu veergude nimed, ei ole väiketähtedega kirjutatud.
- SQL lause osa on liiga pikalt ühel real. Soovitavaks pikkuseks on kuni 80 tähemärki.
- SQL lauses pole tabelite ühendamise tingimuse vasakul pool tabel, millele lauses esimesena viidati.
- Tabelile aliase andmine ilma võtmesõnata AS.

- JOIN operatsioon ilma ühenduse tüüpi (INNER või OUTER) täpsustava võtmesõnata.
- Enne või pärast UNION, EXCEPT või INTERSECT võtmesõna puudub reavahetus.
- Kui SELECT klauslis on rohkem kui üks veerg, siis pole iga veerg esitatud eraldi real.
- Kui SELECT lauses päritakse kindlaid veerge, kuid on ühendatud mitu tabelit, pole täpsustatud, mis tabeli veergu soovitakse kasutada.
- Funktsioonide nimed pole *Pascal case* vormis, mille kohaselt algab nimi suurtähega ning iga järgnev sõna osa algab samuti suurtähega (nt SubString).
- Võrdlustes pole operaatori nime ees ja taga tühikut.
- Mittevõrdsuse kontrollis kasutatakse <> asemel !=.
- Sulgeva sulu ja operaatori nime vahel puudub tühik.
- Üleliigsed tühikud lauses.
- Funktsiooni nime ja sulu alguse vahel on tühik.

Tabelis 1 näidatakse iga pistikprogrammis kontrollitava SQL lause vea kohta, kas samasugust kontrolli oskavad teha ka SQLFluff ja DataGrip. Vigade korral, mida DataGrip tuvastas osaliselt, suutis programm neid vigu tuvastada, kui oli teada andmebaasi struktuur. 38-st veast, mille kontrollimine realiseeriseeriti pistikprogrammis, suutis SQLFluff tuvastada 9 viga ja DataGrip 11 viga, millest 3 oli osaliselt tuvastatud.

Tabel 1. *Tabel SQL pistikprogrammi kontrollide realiseerimisest võrreldes SQLFluff ja DataGrip programmidega*

Kontrolli nimi	**SQLFluff**	**DataGrip**
{LIKE|SIMILAR TO} operaator ilma metamärkideta	Ei tuvastanud	Ei tuvastanud
Otsingutingimuses: =NULL	**Tuvastas**	**Tuvastas**
Otsingutingimuses: <>NULL	**Tuvastas**	**Tuvastas**
Otsingutingimus: >NULL, >=NULL,<NULL, <=NULL	Ei tuvastanud	**Tuvastas**
Otsingutingimus pole loogikaavaldis (nt hinne=1 OR 2)	Ei tuvastanud	Ei tuvastanud
Otsingutingimuses: <veerg> (LIKE|SIMILAR TO) <veerg>	Ei tuvastanud	Ei tuvastanud
Päringu tulemuses on rohkem kui üks sama nimega veerg	**Tuvastas**	Ei tuvastanud

Jätkub...

Tabel 1 – *Jätkub...*

Kontrolli nimi	**SQLFluff**	**DataGrip**
SELECT klauslis on avaldis (sh konstant või funktsiooni poole pöördumine), kuid vastavale veerule pole ise antud nime	**Tuvastas**	Ei tuvastanud
GROUP BY ja DISTINCT samal lause tasemel	**Tuvastas**	Ei tuvastanud
ORDER BY <arv>	Ei tuvastanud	Ei tuvastanud
Kokkuvõttefunktsioone kasutava avaldise SUM()/COUNT() kasutamine kokkuvõttefunktsiooni AVG asemel	Ei tuvastanud	Ei tuvastanud
SUM(1) kasutamine COUNT(*) asemel	Ei tuvastanud	Ei tuvastanud
INSERT lause INSERT klauslis puuduvad veerunimed	Ei tuvastanud	Ei tuvastanud
SELECT * alampäringus, mille tulemuse põhipäringuga võrdlemiseks: IN/NOT IN/=ANY/<>ALL/=/<>/>/>=/</<= operaatorit	Ei tuvastanud	Ei tuvastanud
FROM klauslis rohkem kui üks tabel ja SELECT klauslis * (ilma täpsustava tabelita)	**Tuvastas**	Ei tuvastanud
OUTER JOIN + COUNT(*)	Ei tuvastanud	Ei tuvastanud
UNION|EXCEPT|INTERSECT [ALL] lauses SELECT *	Ei tuvastanud	Ei tuvastanud
Alampäringus ORDER BY (välja arvatud siis, kui seal on LIMIT või FETCH piirang)	Ei tuvastanud	Ei tuvastanud
INSERT .. SELECT *	**Tuvastas**	Ei tuvastanud
Tekstilised väärtused jutumärkides (aga äkki identifikaator?)	Ei tuvastanud	**Tuvastas osaliselt**
Otsingutingimuses nii AND (BETWEEN x AND y ei lähe arvesse) kui OR, kuid pole sulge	Ei tuvastanud	Ei tuvastanud
HAVING klauslis tingimus, mis ei sisalda kokkuvõttefunktsiooni tulemusega võrdlemist	Ei tuvastanud	**Tuvastas**
{=|<>} <tekst kus on LIKE või regulaaravaldise muster>	Ei tuvastanud	Ei tuvastanud
SELECT klauslis kokkuvõttefunktsioon + veerud, kuid puudub GROUP BY	Ei tuvastanud	**Tuvastas osaliselt**
Kokkuvõttefunktsioon WHERE klauslis või GROUP BY klauslis	Ei tuvastanud	**Tuvastas**
=ALL kasutamine =ANY asemel	Ei tuvastanud	Ei tuvastanud

Jätkub...

Tabel 1 – *Jätkub...*

Kontrolli nimi	**SQLFluff**	**DataGrip**
<>ANY kasutamine <>ALL asemel	Ei tuvastanud	Ei tuvastanud
LIMIT/FETCH klausel ilma ORDER BY klauslita	Ei tuvastanud	Ei tuvastanud
Puuduv veerg GROUP BY klauslist	Ei tuvastanud	**Tuvastas osaliselt**
Puuduv ühendamise tingimus	Ei tuvastanud	Ei tuvastanud
Üleliigne semikoolon	Ei tuvastanud	Ei tuvastanud
WHERE või HAVING klauslis puuduv otsingutingimus	**Tuvastas**	**Tuvastas**
LIMIT/FETCH klausel võib alampäringus ebasobivalt tagastada mitu rida	Ei toeta FETCH klauslit	Ei tuvastanud
LIMIT/FETCH klausliga kasutatakse ebakorrektset arvulist piirangut	Ei tuvastanud	Ei tuvastanud
FROM klausli alampäringul puudub alias	Ei tuvastanud	**Tuvastas**
LIMIT/FETCH klauslit kasutatakse päringus, mis tagastab maksimaalselt ühe rea	Ei tuvastanud	Ei tuvastanud
WHERE või HAVING võtmesõna kasutatakse päringus samal tasemel mitu korda	**Tuvastas**	**Tuvastas**
Alampäringus ja põhipäringus pole võrdluses sama arv veergusid	Ei tuvastanud	Ei tuvastanud

EverSQL programmis saab kõigile avalikus vaates infot, kas SQL lause on valiidne või mitte, ning osade lausete puhul pakub see ka erinevaid variante, kuidas saaks lauset optimiseerida. Optimiseerimise detailne info on peidetud, kuni kasutaja keskkonda sisse logib. Peale sisselogimist saab kasutada piiratud arvu tasuta analüüse, mida programm pakub. Seega töö autor kõiki kasutatud lauseid analüüsida ei saanud. Küll aga üksikute puhul sai autor programmilt detailset tagasisidet, kuidas saaks SQL lauseid optimiseerida. Samas EverSQL ei tuvastanud selliseid vigu nagu tuvastasid eeltoodud programmid.

Joonisel 17 välja toodud SQL lause puhul programm ei tuvastanud, et otsingutingimus ei ole korrektne loogikaavaldis. See-eest pakkus programm välja, et lause optimiseerimiseks saaks luua indeksi tabeli *Hotell* veerule *linn*. Samuti soovitas programm, et OR operaatori asemel tuleks kasutada UNION operaatorit ning soovitas veel kasutada UNION operaatorit UNION ALL asemel, kuna see aitab eemaldata lause tulemusest korduvad read. Joonisel 56 on näha EverSQL poolt optimiseeritud SQL lause.

```
1    SELECT
2            hotell_nimi
3        FROM
4            ((SELECT
5                Hotell.nimi AS hotell_nimi
6            FROM
7                Hotell
8            WHERE
9                Hotell.hind BETWEEN 8 AND 9)
10       UNION
11       DISTINCT (SELECT
12           Hotell.nimi AS hotell_nimi
13       FROM
14           Hotell
15       WHERE
16           Hotell.linn = 'Tallinn'
17           OR 'Tartu')
18   ) AS union1
```

Joonis 56. *EverSQL poolt optimeeritud lause*

5.3 Kasutajate poolne testimine

Lisaks valideeriti programmi funktsionaalsust ka autori ja juhendaja poolt testimisega.

Programmi testimiseks kasutati erinevaid andmebaase ja mitmeid erinevaid SQL lauseid, sh nii andmekirjelduskeele kui ka andmekäitluskeele lauseid. Suur osa kasutatud SQL lausetest olid sellised, mida võiks kohata igapäevases kasutuses, kuid oli ka keerulisemaid või ainulaadsemaid, mis panid pistikprogrammi PostgreSQL mooduli funktsionaalsuse eriti hästi proovile.

Lausete käivitamisel leitud vead edastas juhendaja autorile, mille autor üritas ka võimalikult kiirelt parandada, et see ei takistaks edaspidist SQL lausete testimist. Parandatud vigade kohta on täpsemalt kirjutatud jaotises 4.9.

Tellija (antud juhul juhendaja) tagasiside loodud programmi kohta on esitatud lisas 5.

6. Arendusvaade

Käesoleva töö raames valminud pistikprogrammi täiendus võimaldab kasutajatel käivitada Notepad++ tekstiredaktoris SQL lauseid PostgreSQL andmebaasis. Lisaks suudab pistikprogramm tuvastada mitmeid vigu, mis võivad esineda PostgreSQL SQL lausetes ja anda kasutajatele nende kohta tagasisidet. Pistikprogrammi võiks edasi arendada, et kasutajatel oleks sellest veel rohkem kasu. Järgnevalt on välja toodud mitteammendav loetelu võimalustest, millega saaks rakendust kasulikumaks teha.

- SQL lauseid peaks saama analüüsida ilma andmebaasi ühenduse avamiseta. Kui andmebaasiga pole ühendust loodud ja analüüsitakse lauseid, siis peaks pakkuma valiku toetatud andmebaasisüsteemidest.
- Pakkuda kasutajale võimalus ühenduse loomisel parool salvestada, et kasutaja ei peaks iga kord parooli sisestama.
- Taastada pistikprogrammis andmebaasisüsteemide MS SQL Server ja SQLite kasutamise võimalus, mis realiseeriti algses 2014. aasta versioonis.
- Realiseerida pistikprogrammis ka teiste andmebaasisüsteemide moodulid koos SQL lausete käivitamise ja kontrollimise funktsionaalsusega, nagu näiteks H2, MySQL ja MariaDB.
- Arendada välja SQL lausete kontrollid, mis sõltuvad andmebaasi struktuurist.
- Täiendada olemasolevaid SQL lausete kontrolle nii MS Accessi kui ka PostgreSQL andmebaasisüsteemidele.
- Pakkuda täiendavad informatsiooni SQL vigade kohta, mille esinemist programm kontrollib.
- Kopeerida analüüsi tulemusel leitud vigadega laused automaatselt uude ajutisse faili koos selle juures oleva tekstilise kommentaariga vea kohta, mida kasutaja saab hiljem ise muuta.
- Arendada välja süsteem, kus kasutaja saab erinevaid SQL lausete vigade kontrolle sisse ja välja lülitada.
- Arendada pistikprogrammile koodi lõpetamise funktsionaalsus.
- Arendada pistikprogrammis funktsionaalsus andmebaasiühenduse halduri kontekstmenüüs tabelite, vaadete, hetktõmmiste, funktsioonide ja veergude nimetusi kopeerida.
- Registreerida pistikprogamm Notepad++ ametliku pistikprogrammina, et kasutajad ei peaks pistikprogrammi eraldi sammudega arvutisse paigaldama.

7. Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli täiendada tekstiredaktori Notepad++ pistikprogrammi NppDB, mille eelmises versioonis realiseeriti MS Accessi andmebaasis SQL lausete käivitamine ja lausetest vigade otsimine. Täiendatud pistikprogramm pidi olema suuteline tuvastama MS Accessi SQL lausetest veel rohkem vigu kui enne ning tuli lisada uue andmebaasisüsteemi - PostgreSQL - tugi. PostgreSQL mooduliga pidi nägema ühendatud PostgreSQL andmebaasi struktuurielemente ning pidi olema võimalik käivitada tekstiredaktoris SQL lauseid PostgreSQL andmebaasis. SQL lauseid pidi olema võimalik analüüsida ilma neid lauseid käivitamata ning iga analüüsitud SQL lause kohta pidi andma kasutajale tagasisidet võimalike probleemide kohta. Analüüsitavad SQL lausete probleemid võivad olla universaalsed või spetsiifilised probleemid MS Accessi või PostgreSQL andmebaasisüsteemi jaoks. Need probleemid võivad olla nii koodi käivitamist takistavad vead, kui ka koodi loetavust ja arusaadavust halvendavad vead.

Probleemsete SQL lausete osas uuriti eelnevalt tehtud magistritöös lisatud vigade kontrolle ning varem läbiviidud uuringuid, et koguda infot vigade kohta, mille otsimisega saaks pistikprogrammi täiendada. Pistikprogrammi poolt täiendavalt kontrollitavad vead leiti koostöös juhendajaga.

Bakalaureusetöö tulemusena valmis tekstiredaktori Notepad++ pistikprogrammi NppDB edasiarendus, kuhu lisati PostgreSQL andmebaasisüsteemis SQL lausete käivitamise ja eelkontrollimise funktsionaalsus. Lisaks täiendati olemasolevat MS Access SQL lausete vigade kontrollimise funktsionaalsust, lisades juurde 12 vea esinemise kontrollimine. Pistikprogrammi laiendamisel kasutati programmeerimiskeelt C# ja raamistikku .NET, andmebaasisüsteemiga suhtlemiseks kasutatakse Npgsql-i ja lausete tuvastamiseks ning analüüsimiseks parseri generaatorina ANTLR4. Täiendatud pistikprogrammi abil saab käivitada kõiki PostgreSQL SQL lauseid, vaadelda andmebaasi struktuurielemente ning kontrollida SQL lausetes vigade esitamist. PostgreSQL korral on realiseeritud 38 vea esinemise kontrollimine.

Töö käigus testiti loodud tarkvara nii autori kui ka juhendaja poolt käsitsi ning võrreldi pistikprogrammi funktsionaalsust ka teiste sarnaste programmidega. Testimisest saadud tagasiside põhjal tehti programmi jooksvalt täiendusi ja parandusi. Võrreldes olemasolevate programmidega, mis võimaldavad PostgreSQL andmebaasis SQL lauseid käivitada ja kontrollida, on käesoleva töö raames valminud tarkvara võimekam, kui võtta arvesse, et

täiendatud pistikprogrammis on nii lausete käivitamise kui ka kontrollimise funktsionaalsus. Lisaks suudab pistikprogramm tuvastada suuremat hulka erinevat tüüpi vigu kui võrreldud programmid.

Bakalaureusetöö tulemusel valminud pistikprogrammi täiendus ei ole lõplikult valmis selles mõttes, et alati saab lisada pistikprogrammi täiendavaid kontrolle ja funktsionaalsust, sealhulgas ka teiste andmebaasisüsteemide tuge.

Käesoleva töö tulemus avaldati MIT litsentsiga ja on avalikult kättesaadav GitHub'ist aadressilt: https://github.com/aneelm/NppDB

Kasutatud kirjandus

[1] T. Parr, "ANTLR", Kasutatud: 21-08-2023. [Online]. Loetud aadressil: https://www.antlr.org/.

[2] The PostgreSQL Global Development Group, "PostgreSQL", Kasutatud: 01.09.2023. [Online]. Loetud aadressil: https://www.postgresql.org/.

[3] solid IT, "DB-Engines Ranking", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://db-engines.com/en/ranking.

[4] Wise Coders GmbH, "DbSchema", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://dbschema.com/.

[5] S. Rider, "DBeaver", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://dbeaver.io/.

[6] DbVis Software, "DbVisualizer", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://www.dbvis.com/.

[7] P. G. D. Group, "psql", Kasutatud: 01.10.2023. [Online]. Loetud aadressil: https://www.postgresql.org/docs/current/app-psql.html.

[8] DBCLI, "pgcli", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://github.com/dbcli/pgcli.

[9] DBCLI, "odbc-cli", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://github.com/dbcli/odbc-cli.

[10] D. Ho, "What is Notepad++", Kasutatud: 25-08-2023. [Online]. Loetud aadressil: https://notepad-plus-plus.org/.

[11] Microsoft, "Visual Studio Code", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://code.visualstudio.com/.

[12] Microsoft, "PostgreSQL for Visual Studio Code", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://marketplace.visualstudio.com/items?itemName=ms-ossdata.vscode-postgresql.

[13] Microsoft, "Download Visual Studio Code", Kasutatud: 21-08-2023. [Online]. Loetud aadressil: https://code.visualstudio.com/download.

[14] D. Ho, "Download Notepad++", Kasutatud: 12.11.2023. [Online]. Loetud aadressil: https://notepad-plus-plus.org/downloads/v8.5.8/.

[15] S. Jung, "NppDB", Kasutatud: 19-08-2023. [Online]. Loetud aadressil: https://github.com/gutkyu/NppDB.

[16] P. Post, "MS Accessi andmebaasides SQL programmeerimist lihtsustava pistikprogrammi loomine lähtekoodiredaktorile Notepad++", [Magistritöö], Infotehnoloogia teaduskond, TalTech, Tallinn, Eesti, 2023, Kasutatud: 25.08.2023. [Online]. Loetud aadressil: https://digikogu.taltech.ee/et/item/0ac16423-5434-450d-a961-26fa087f91d4.

[17] Microsoft, "Microsoft Access", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://www.microsoft.com/en-us/microsoft-365/access.

[18] P. Post, "NppDB", Kasutatud: 06-08-2023. [Online]. Loetud aadressil: https://github.com/pripost/NppDB.

[19] J. Prochazka, "DbGate | Open Source SQL+noSQL Database Client", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://dbgate.org/.

[20] Stefan Brass ja Christian Goldberg, "Semantic errors in SQL queries: A quite complete list", *Journal of Systems and Software*, vol. 79, no. 5, pp. 630–644, 2006, Kasutatud: 01.09.2023. DOI: 10.1016/j.jss.2005.06.028.

[21] D. Miedema, E. Aivaloglou ja G. Fletcher, "Identifying SQL Misconceptions of Novices: Findings from a Think-Aloud Study", ICER 2021, pp. 355–367, 2021, Kasutatud: 01.09.2023. DOI: 10.1145/3446871.3469759.

[22] T. Taipalus, M. Siponen ja T. Vartiainen, "Errors and Complications in SQL Query Formulation", *ACM Trans. Comput. Educ.*, vol. 18, no. 3, pp. 1–29, Aug. 2018, Kasutatud: 01.09.2023. DOI: 10.1145/3231712.

[23] GitHub Inc, "GitHub", Kasutatud: 31-08-2023. [Online]. Loetud aadressil: https://github.com/.

[24] The Npgsql Development Team, "Npgsql - .NET Access to PostgreSQL", Kasutatud: 07.10.2023. [Online]. Loetud aadressil: https://www.npgsql.org/.

[25] PostgreSQL Global Development Group, "psqlODBC - PostgreSQL ODBC driver", Kasutatud: 04.10.2023. [Online]. Loetud aadressil: https://odbc.postgresql.org/.

[26] Intellisoft LLC, "PostgreSQL Native OLEDB Provider", Kasutatud: 01.09.2023. [Online]. Loetud aadressil: https://www.pgoledb.com/.

[27] ANTLR, "antlr/grammars-v4", Kasutatud: 01.09.2023. [Online]. Loetud aadressil: https://github.com/antlr/grammars-v4/tree/master/sql/postgresql.

[28] Microsoft, "What is .NET?" Kasutatud: 25-08-2023. [Online]. Loetud aadressil: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet.

[29] Microsoft, "Visual Studio: IDE and Code Editor for Software Developers and Teams", Kasutatud: 21-08-2023. [Online]. Loetud aadressil: https://visualstudio.microsoft.com/.

[30] Microsoft, "xUnit.net", Kasutatud: 14.10.2023. [Online]. Loetud aadressil: https://xunit.net/.

[31] Testcontainers, "Testcontainers for .NET", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://dotnet.testcontainers.org/.

[32] pgAdmin, "pgAdmin - PostgreSQL Tools", Kasutatud: 01.10.2023. [Online]. Loetud aadressil: https://www.pgadmin.org/.

[33] EnterpriseDB, "EDB: Open-Source, Enterprise Postgres Database Management", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://www.enterprisedb.com/.

[34] JetBrains, "DataGrip: The Cross-Platform IDE for Databases & SQL by JetBrains", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://www.jetbrains.com/datagrip/.

[35] M. Rathbone, "The SQL Editor and Database Manager Of Your Dreams | Beekeeper Studio", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://www.beekeeperstudio.io/.

[36] R. Pärnamäe, "PostgreSQL veebipõhise visuaalse andmekäitluskeele lausete koostamise tarkvara edasiarendamine", [Magistritöö], Infotehnoloogia teaduskond, TalTech, Tallinn, Eesti, 2020, Kasutatud: 15.12.2023. [Online]. Loetud aadressil: https://digikogu.taltech.ee/et/Item/0f3e89c8-bf81-4353-8e83-3c83b1ebe698.

[37] JetBrains, "JetBrains: Essential tools for software developers and teams", Kasutatud: 14.10.2023. [Online]. Loetud aadressil: https://www.jetbrains.com/.

[38] SQLFluff, "SQLFluff - The SQL Linter for humans", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://sqlfluff.com/.

[39] EverSQL, "EverSQL | Automatic SQL Query Optimization for MySQL & PostgreSQL", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://www.eversql.com/.

[40] AnalysisTools, "The Best SQL Static Analysis Tools (Linters/Formatters)", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://analysis-tools.dev/tag/sql.

[41] JetBrains, "IntelliJ IDEA – the Leading Java and Kotlin IDE", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://www.jetbrains.com/idea/.

[42] P. Doran, "vscode-sqlfluff", Kasutatud: 11.11.2023. [Online]. Loetud aadressil: https://github.com/sqlfluff/vscode-sqlfluff.

[43] Microsoft, "ALL, DISTINCT, DISTINCTROW, TOP Predicates", Kasutatud: 31-08-2023. [Online]. Loetud aadressil: https://support.microsoft.com/en-us/office/all-distinct-distinctrow-top-predicates-24f2a47d-a803-4c7c-8e81-756fe298ce57.

[44] E. Eessaar, "SQL tüüpvead ja kommentaarid enne SQL kontrolltööd", Tallinn, 2023.

[45] R. Odone, "The love-hate relationship between SELECT and GROUP BY in SQL", Kasutatud: 15.12.2023. [Online]. Loetud aadressil: https://medium.com/@riccardoodone/the-love-hate-relationship-between-select-and-group-by-in-sql-4957b2a70229.

[46] CYBERTEC PostgreSQL International GmbH, "Aliases for sub-SELECTS in FROM clause", Kasutatud: 19.12.2023. [Online]. Loetud aadressil: https://www.cybertec-postgresql.com/en/aliases-for-sub-selects-in-from-clause/.

[47] Microsoft, "Microsoft Access Database Engine 2010 Redistributable", Kasutatud: 25-08-2023. [Online]. Loetud aadressil: https://www.microsoft.com/en-us/download/details.aspx?id=13255.

[48] The Open Source Initiative, "The MIT license", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://opensource.org/license/mit/.

[49] M. Wu, "Top 8 Free, Open Source SQL Clients to Make Database Management Easier 2023", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://www.bytebase.com/blog/top-open-source-sql-clients/.

[50] K. Ostrowska, "Best SQL IDEs for You", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://learnsql.com/blog/best-sql-editor/.

[51] ScaleGrid, "Which Is The Best PostgreSQL GUI? 2021 Comparison", Kasutatud: 04.12.2023. [Online]. Loetud aadressil: https://scalegrid.io/blog/which-is-the-best-postgresql-gui-2021-comparison/.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

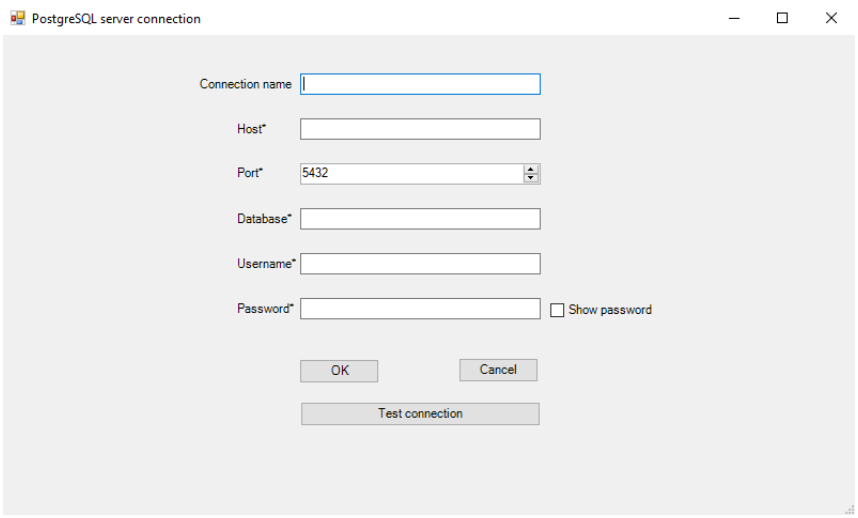
Mina, Andres Eelma

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Lähtekoodiredaktoris Notepad++ SQL programmeerimist lihtsustava pistikprogrammi edasiarendamine MS Accessi ja PostgreSQL jaoks", mille juhendaja on Erki Eessaar
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

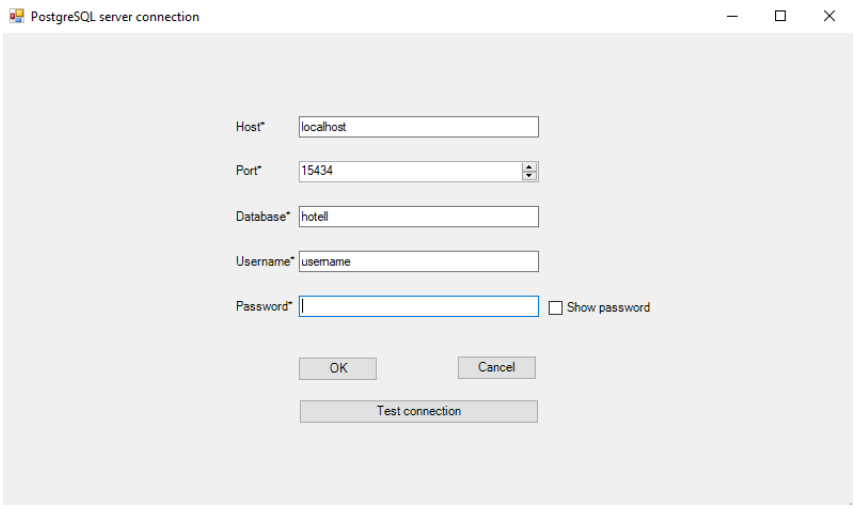
10.01.2024

[1] Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

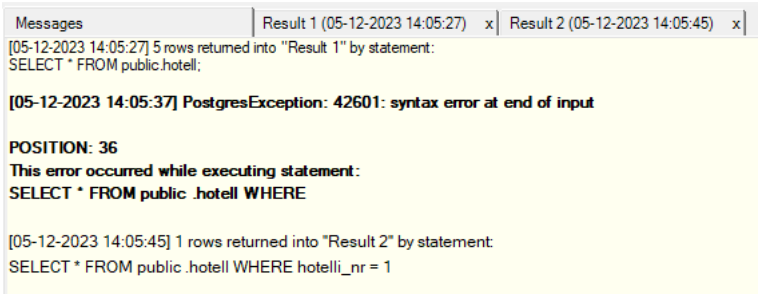
Lisa 2 – Kuvatõmmised rakendusest



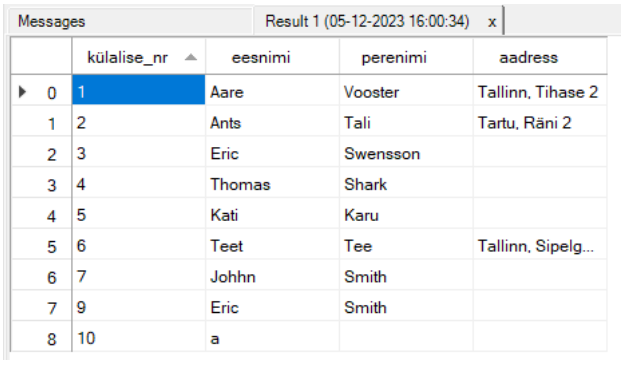
Joonis 57. *Uue PostgreSQL andmebaasiühenduse loomise aken.*



Joonis 58. *Eelnevalt ühendatud PostgreSQL andmebaasiga ühenduse taastamise aken.*

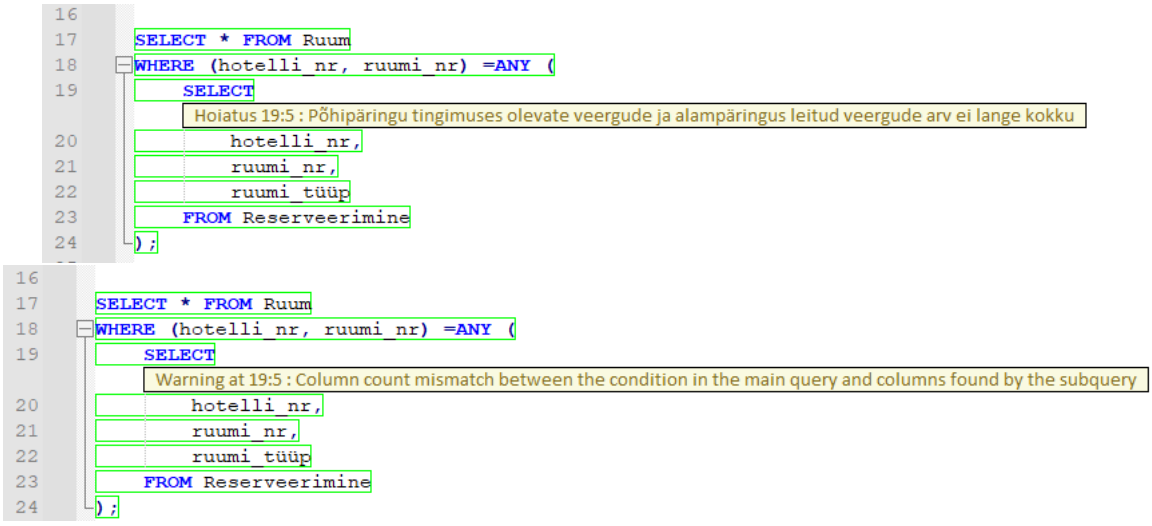


Joonis 59. *Rasvases kirjes veateade tulemuste logi sakis.*

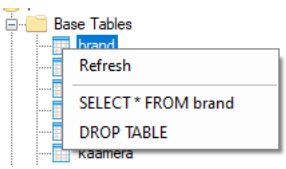


	külalise_nr	eesnimi	perenimi	aadress
0	1	Aare	Vooster	Tallinn, Tihase 2
1	2	Ants	Tali	Tartu, Räni 2
2	3	Eric	Swensson	
3	4	Thomas	Shark	
4	5	Kati	Karu	
5	6	Teet	Tee	Tallinn, Sipelg...
6	7	Johhn	Smith	
7	9	Eric	Smith	
8	10	a		

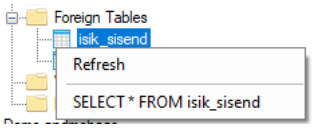
Joonis 60. *Tulemuste sakis olevad järjekorranumbrid.*



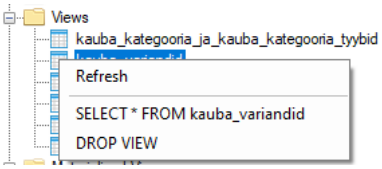
Joonis 61. *SQL lause veateade eesti- ja inglise keeles.*



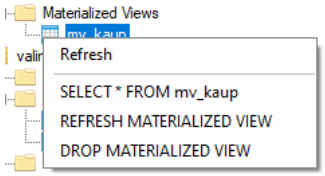
Joonis 62. *Baastabelite kontekstmenüü.*



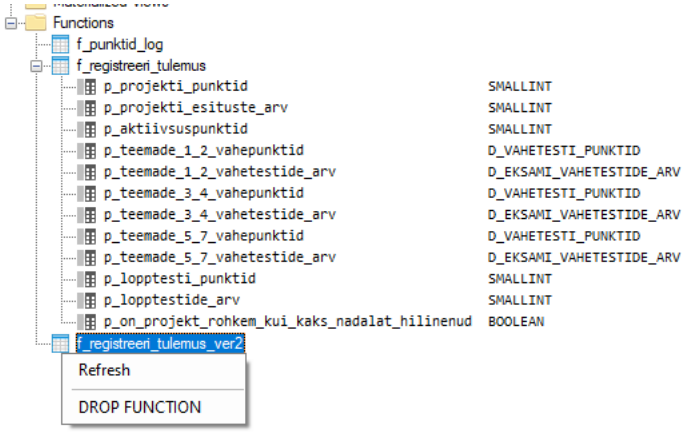
Joonis 63. *Väliste tabelite kontekstmenüü.*



Joonis 64. *Vaadete kontekstmenüü.*



Joonis 65. *Materialiseeritud vaadete kontekstmenüü.*



Joonis 66. *Funktsioonide kontekstmenüü.*

Lisa 3 – SQL laused

```
CREATE OR REPLACE FUNCTION f_registreeri_tulemus_ver2(
p_projekti_punktid Punktid.projekti_punktid%TYPE,
p_projekti_esituste_arv Punktid.projekti_esituste_arv%TYPE,
p_aktiivsuspunktid Punktid.aktiivsuspunktid%TYPE,
p_teemade_1_2_vahepunktid Punktid.teemade_1_2_vahepunktid%TYPE,
p_teemade_1_2_vahetestide_arv Punktid.teemade_1_2_vahetestide_arv%TYPE,
p_teemade_3_4_vahepunktid Punktid.teemade_3_4_vahepunktid%TYPE,
p_teemade_3_4_vahetestide_arv Punktid.teemade_3_4_vahetestide_arv%TYPE,
p_teemade_5_7_vahepunktid Punktid.teemade_5_7_vahepunktid%TYPE,
p_teemade_5_7_vahetestide_arv Punktid.teemade_5_7_vahetestide_arv%TYPE,
p_lopptesti_punktid Punktid.lopptesti_punktid%TYPE,
p_lopptestide_arv Punktid.lopptestide_arv%TYPE,
p_on_projekt_rohkem_kui_kaks_nadalat_hilinenud Punktid.on_projekt_rohkem_kui_kaks_nadalat_hilinenud%TYPE
) RETURNS VOID
LANGUAGE sql SECURITY DEFINER
SET search_path = public, pg_temp
BEGIN ATOMIC
MERGE INTO Punktid T
USING (SELECT
0 AS punktid_id,
p_projekti_punktid AS projekti_punktid,
p_projekti_esituste_arv AS projekti_esituste_arv,
p_aktiivsuspunktid AS aktiivsuspunktid,
p_teemade_1_2_vahepunktid AS teemade_1_2_vahepunktid,
p_teemade_1_2_vahetestide_arv AS teemade_1_2_vahetestide_arv,
p_teemade_3_4_vahepunktid AS teemade_3_4_vahepunktid,
p_teemade_3_4_vahetestide_arv AS teemade_3_4_vahetestide_arv,
p_teemade_5_7_vahepunktid AS teemade_5_7_vahepunktid,
p_teemade_5_7_vahetestide_arv AS teemade_5_7_vahetestide_arv,
p_lopptesti_punktid AS lopptesti_punktid,
p_lopptestide_arv AS lopptestide_arv,
p_on_projekt_rohkem_kui_kaks_nadalat_hilinenud AS on_projekt_rohkem_kui_kaks_nadalat_hilinenud) S
ON (T.punktid_id=S.punktid_id)
WHEN MATCHED THEN
UPDATE SET projekti_punktid = S.projekti_punktid,
projekti_esituste_arv = S.projekti_esituste_arv,
aktiivsuspunktid = S.aktiivsuspunktid,
teemade_1_2_vahepunktid = S.teemade_1_2_vahepunktid,
teemade_1_2_vahetestide_arv = S.teemade_1_2_vahetestide_arv,
teemade_3_4_vahepunktid = S.teemade_3_4_vahepunktid,
teemade_3_4_vahetestide_arv = S.teemade_3_4_vahetestide_arv,
teemade_5_7_vahepunktid = S.teemade_5_7_vahepunktid,
teemade_5_7_vahetestide_arv = S.teemade_5_7_vahetestide_arv,
lopptesti_punktid = S.lopptesti_punktid,
lopptestide_arv = S.lopptestide_arv,
on_projekt_rohkem_kui_kaks_nadalat_hilinenud = S.on_projekt_rohkem_kui_kaks_nadalat_hilinenud
WHEN NOT MATCHED THEN INSERT (punktid_id, projekti_punktid, projekti_esituste_arv, aktiivsuspunktid,
teemade_1_2_vahepunktid, teemade_1_2_vahetestide_arv, teemade_3_4_vahepunktid,
teemade_3_4_vahetestide_arv, teemade_5_7_vahepunktid, teemade_5_7_vahetestide_arv,
lopptesti_punktid, lopptestide_arv, on_projekt_rohkem_kui_kaks_nadalat_hilinenud)
VALUES (S.punktid_id, S.projekti_punktid, S.projekti_esituste_arv, S.aktiivsuspunktid,
S.teemade_1_2_vahepunktid, S.teemade_1_2_vahetestide_arv, S.teemade_3_4_vahepunktid,
S.teemade_3_4_vahetestide_arv, S.teemade_5_7_vahepunktid, S.teemade_5_7_vahetestide_arv,
S.lopptesti_punktid, S.lopptestide_arv, S.on_projekt_rohkem_kui_kaks_nadalat_hilinenud);
END;
```

Joonis 67. *Keeruline lause, kus esineb rohkem kui üks lauset lõpetav märk - semikoolon.*

```sql
SELECT *
FROM Ruum
WHERE hind>1_000_000;

SELECT *
FROM Ruum
WHERE hind>0xF4240;
```

Joonis 68. *SQL laused, kus esinevad alakriipsudega arv ja kuueteistkümnendsüsteemis esitatud arv.*

```sql
SELECT
    p.proname AS function_name,
    l.lanname AS function_language,
    CASE
        WHEN l.lanname = 'internal' THEN p.prosrc
        ELSE pg_get_functiondef(p.oid)
    END AS definition,
    pg_get_function_arguments(p.oid) AS function_arguments,
    t.typname AS return_type
FROM pg_proc p
    LEFT JOIN pg_namespace n ON p.pronamespace = n.oid
    LEFT JOIN pg_language l ON p.prolang = l.oid
    LEFT JOIN pg_type t ON t.oid = p.prorettype
WHERE n.nspname = 'public'
ORDER BY function_name;
```

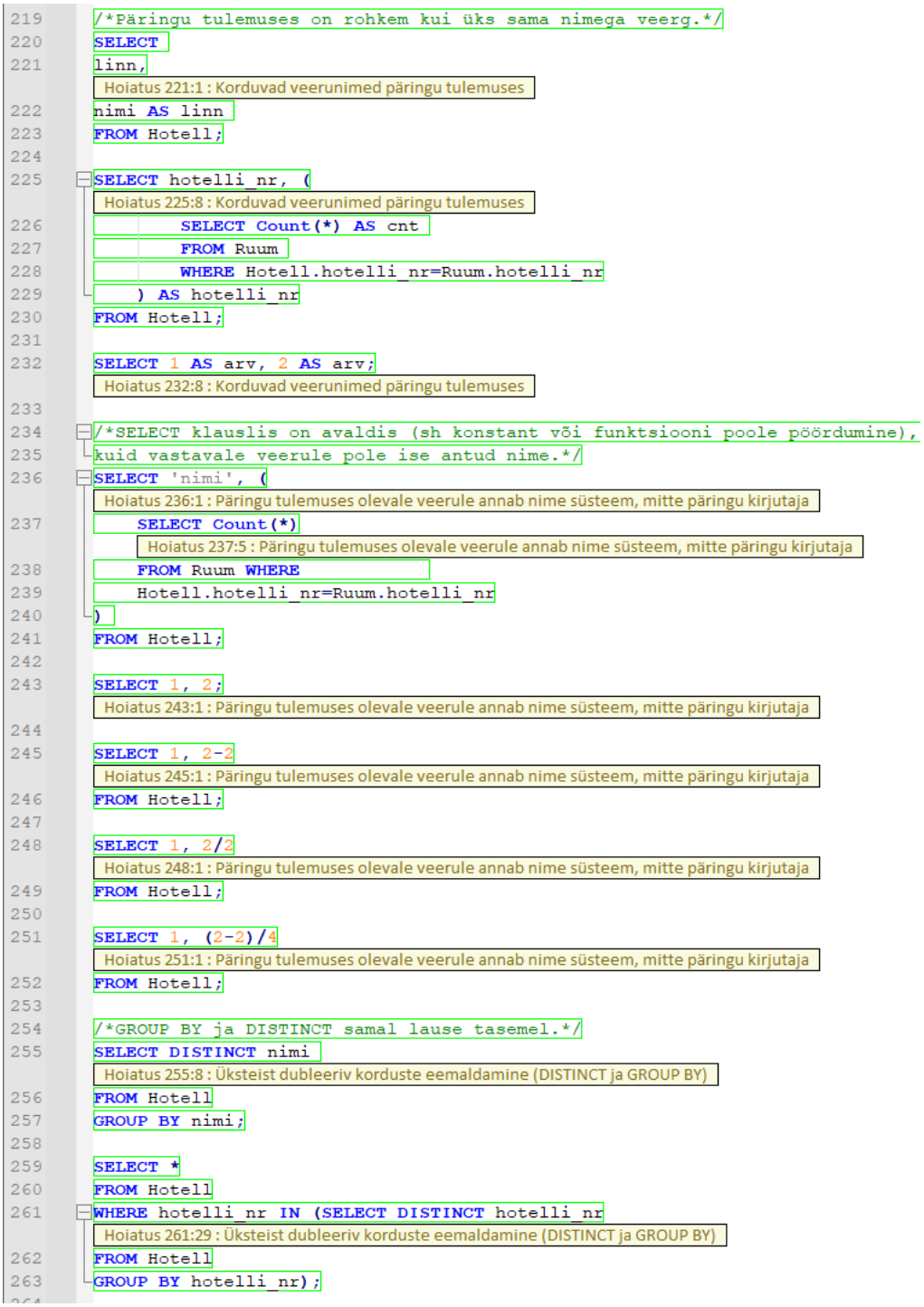
Joonis 69. *SQL lause, millega küsitakse infot funktsioonide kohta.*

```sql
SELECT
    attr.attname AS column_name,
    format_type(attr.atttypid, attr.atttypmod) AS data_type,
    pg_get_expr(d.adbin, d.adrelid) AS column_default,
    attr.attnotnull::TEXT AS is_nullable
FROM pg_attribute AS attr
    LEFT JOIN pg_attrdef d ON (attr.attrelid, attr.attnum) = (d.adrelid, d.adnum)
    JOIN pg_class AS cls ON cls.oid = attr.attrelid
    JOIN pg_namespace AS ns ON ns.oid = cls.relnamespace
WHERE ns.nspname = 'public'
    AND cls.relname = 'hotell'
    AND attr.attnum >= 1
ORDER BY attr.attnum
```

Joonis 70. *SQL lause, millega küsitakse infot tabeli veergude kohta.*

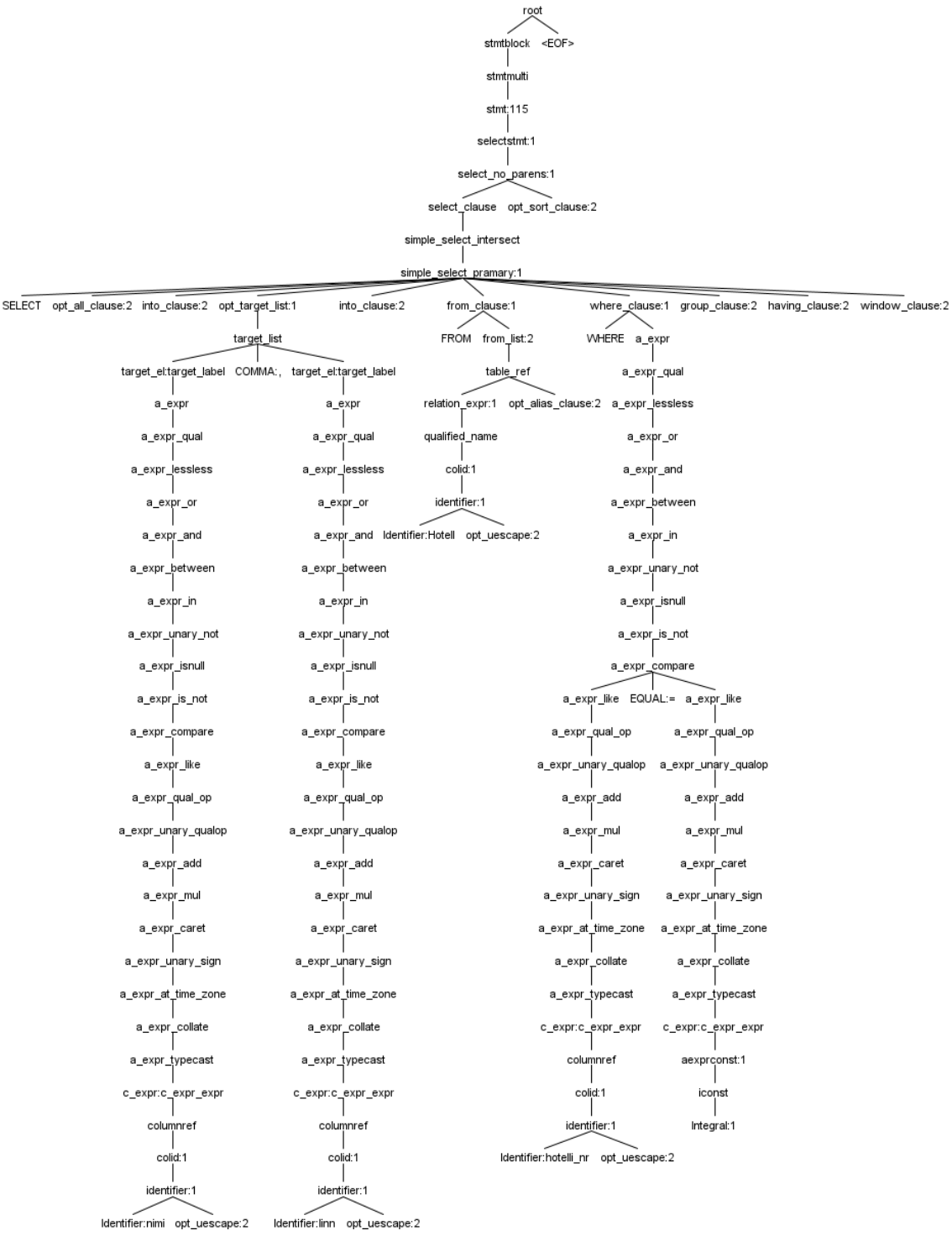
```
SET client_encoding = 'UTF8';

ALTER DATABASE hotell OWNER TO username;

START TRANSACTION;

CREATE TABLE public.hotell (
    hotelli_nr integer NOT NULL,
    nimi character varying(50) NOT NULL,
    linn character varying(50) NOT NULL
);


ALTER SEQUENCE public.hotell_hotelli_nr_seq
OWNED BY public.hotell.hotelli_nr;

INSERT INTO public.hotell(hotelli_nr, nimi, linn)
VALUES (1, 'Viru', 'Tallinn');

UPDATE public.hotell
SET linn = 'Tartu'
WHERE hotelli_nr = 1;

DELETE FROM public.hotell;

COMMENT ON TABLE public.hotell
IS 'Hotellide andmed';

COMMIT;

SHOW timezone;

SELECT *
FROM public.hotell
WHERE hotelli_nr>1_000_000;

SELECT *
FROM public.hotell
WHERE hotelli_nr < 0xF4240;
```

Joonis 71. *Fragment SQL lausetega failist, millega kontrolliti SQL lausete PostgreSQLi andmebaasis käivitamist.*

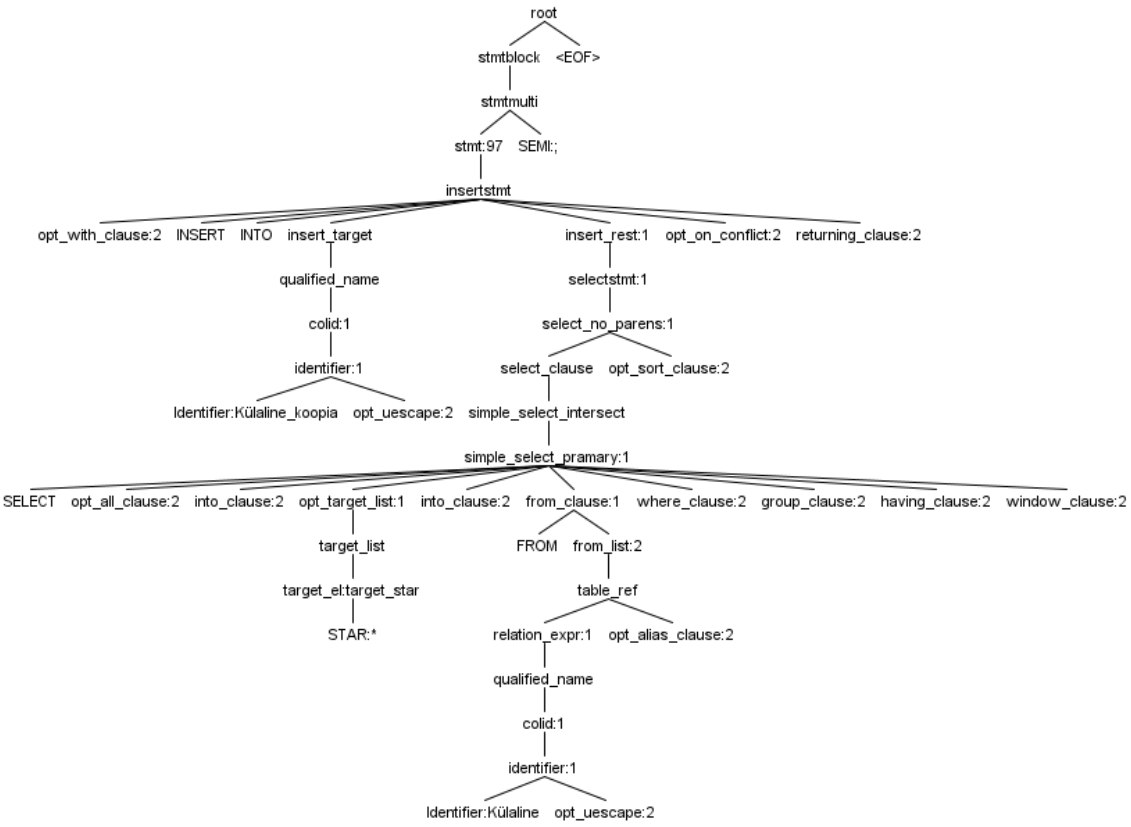


Joonis 72. *Fragment SQL lausetega failist, millega kontrolliti PostgreSQL SQL lausete vigade kontrollimist.*

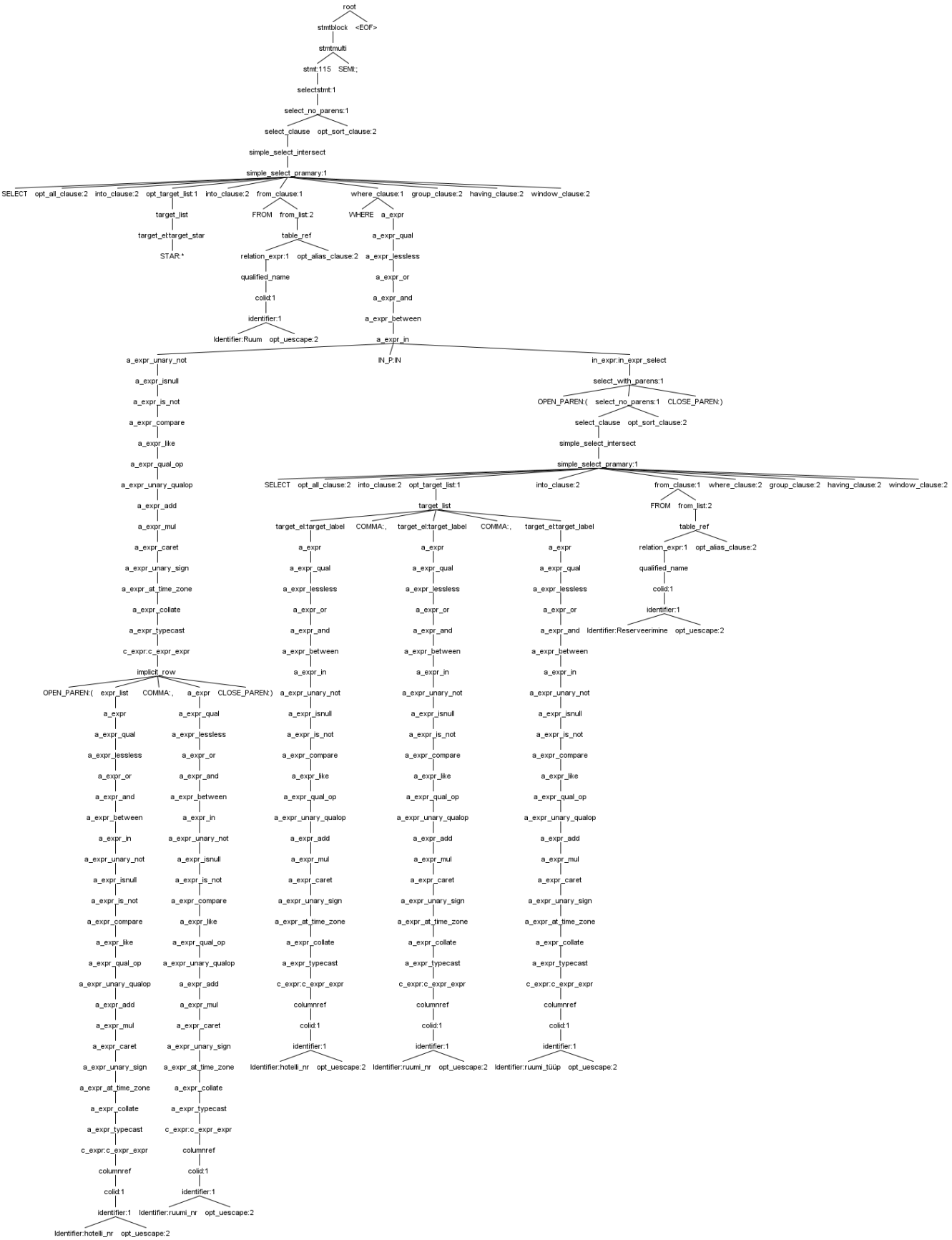
Lisa 4 – ANTLR



Joonis 73. *Näide parseri genereeritud puust SQL lausele* `SELECT nimi, linn FROM Hotell WHERE hotelli_nr = 1;`



Joonis 74. *Näide parseri genereeritud puust SQL lausele* `INSERT INTO Külaline_koopia SELECT * FROM Külaline;`



Joonis 75. *Näide parseri genereeritud puust SQL lausele* `SELECT * FROM Ruum WHERE (hotelli_nr, ruumi_nr) IN (SELECT hotelli_nr, ruumi_nr, ruumi_tüüp FROM Reserveerimine);`

```
3535    a_expr_like
3536       : lhs=a_expr_qual_op (NOT? (operands+=LIKE | operands+=ILIKE | operands+=SIMILAR TO) rhs=a_expr_qual_op opt_escape)?
3537       ;
3538
3539    a_expr_qual_op
3540       : lhs=a_expr_add (operands+=qual_op rhs+=a_expr_add)*
3541       ;
3542
3543    a_expr_add
3544       : lhs=a_expr_mul (operands+=(MINUS | PLUS) rhs+=a_expr_mul)*
3545       ;
3546
3547    a_expr_mul
3548       : lhs=a_expr_caret (operands+=(STAR | SLASH | PERCENT) rhs+=a_expr_caret)*
3549       ;
3550
3551    a_expr_caret
3552       : lhs=a_expr_unary_sign (operands+=CARET rhs=a_expr)?
3553       ;
```

Joonis 76. *Näide ANTLR grammatikafaili tehtud täiendustest.*

Lisa 5 – Tellija tagasiside

Käesolev bakalaureusetöö ja ka sellele eelnenud magistritöö tõukusid tellija vajadusest SQL lauseid õpetamise käigus kiiresti ja mugavalt käivitada. Tellija õpetab ülikoolis andmebaaside kursuseid. Ühes kursuses toimub SQL lausete koostamise harjutamine. Õppijad lahendavad praktikumi jooksul suurel hulgal harjutusülesandeid, saadavad need Teamsi kaudu õppejõule, kes annab reaalajas tagasisidet. Tagasiside andmiseks peab õppejõud muuhulgas neid lauseid käivitama. Lausete edastamisel tekivad lausetesse mittetrükitavad märgid, mis tuleks enne käivitamist eemaldada ja mille jaoks Notepad++ on väga mugav. Võimalus peale seda lause selles samas keskkonnas käivitada (ilma vajaduseta käivitada teisi rakendusi ning lauseid erinevate akende/rakenduste vahel kopeerida) muudab õppejõu töö palju kiiremaks ja sujuvamaks ning võimaldab anda ühenduses olevatele õppijatele tagasisidet kiiremini. Oletame, et harjutustunnis osaleb 20 üliõpilast, lahendamiseks on neli ülesannet ja ühe ülesande lahendatud saamiseks saadetakse ühe üliõpilase poolt lause ülevaatamiseks keskmiselt kolm korda. Sellisel juhul oleks vaja 90 minuti jooksul käivitada õppejõul 240 lauset. Kui ühe lause Notepad++'ist kopeerimiseks ja teises programmis käivitamiseks kuluks viis sekundit, siis kulub selleks 240*5=1200 sekundit e 20 minutit. Kui nädalas oleks kaheksa praktikumi, siis teeks see 160 minutit e 2.7 tundi. Lause käivitamine Notepad++'is vähendab seda aega märkimisväärselt.

Teisalt teevad õppijad lausetes sageli korduma kippuvaid vigu. Õppimise ja õpetamise mõttes oleks efektiivne kui õppijad saavad selle kohta tagasisidet automaatselt, juba enne lause õppejõule edastamist. Käesoleva töö tulemusena loodud lahendus võimaldab seda eesmärki saavutada.

Teises aines on vaja koostada üliõpilastele keerukamaid SQL keele koodi näiteid, kus SQL failis on nii laused kui ka kommentaarid nende kohta. Loodud lahendus võimaldab nii õppejõul kui õppijal neid lauseid käivitada selles samas keskkonnas, kus fail avatakse.

Kokkuvõttes hindan saavutatud tulemust enda tööprotsesside sujuvamaks muutmise jaoks väga kasulikuks. Kuna valminud lahendus on avatud lähtekoodiga ja maailma avaldatud, siis saavad inimesed üle maailma järgi proovida, kas ka neile on see lahendus nii kasulik kui tellijale.