TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Priit Post

MS Accessi andmebaasides SQL programmeerimist lihtsustava pistikprogrammi loomine lähtekoodiredaktorile Notepad++

Magistritöö

Juhendaja: Erki Eessaar
PhD

Tallinn 2023

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: Priit Post

09.05.2023

Annotatsioon

Käesoleva magistritöö eesmärgiks oli kavandada ja arendada teksti- ja lähtekoodiredaktorile Notepad++ pistikprogramm, millel on kaks ülesannet. Esiteks peab selle abil olema võimalik redaktoris käivitada SQL lauseid (sh andmekirjelduskeele lauseid – CREATE, ALTER, DROP ja SELECT … INTO) otse töölaua andmebaasisüsteemi MS Access andmebaasis. Teiseks peab see andma kasutajale tagasisidet käivitatavates SQL lausetes esinevate probleemide kohta. Pistikprogramm peab olema kirjutatud viisil, mis muudaks võimalikult lihtsaks hilisema teiste andmebaasisüsteemide toe lisamise, uute probleemide esinemise kontrollimise lisamise ja uute probleemidest raporteerimise keelte lisamise.

Töö käigus uuriti olemasolevaid MS Accessi andmebaasi kasutamiseks mõeldud programme. Samuti uuriti Notepad++ olemasolevaid pistikprogramme, mida oleks võimalik edasi arendada ning laiendada andmebaasisüsteemile MS Access. Töö käigus kaasajastati pistikprogrammi NppDB. Sellele lisati juurde andmebaasisüsteemis MS Access SQL lausete käivitamise võimekus ning arendati ka võimekus leida SQL lausetes esinevaid mõningaid probleeme. Probleemsete SQL lausete osas analüüsiti olemasolevaid uuringuid, et leida probleemid, mille tuvastamisele SQL lausetest tuleb magistritöös keskenduda. Autor tuvastas kirjanduse põhjal suure hulga SQL lausete võimalikke probleeme ja tegi koostöös juhendajaga nende hulgast valiku.

Töö käigus loodud tarkvara testiti nii autori enda kui ka juhendaja poolt ja valideeriti teiste sarnaste programmide vastu. Saadud tagasiside põhjal hinnati rakenduse vastavust püstitatud nõuetele ja tehti parandusi.

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

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 79 leheküljel, 8 peatükki, 43 joonist, 3 tabelit.

Abstract

Creating a Plugin for Source Code Editor Notepad++ that Simplifies SQL Programming in MS Access Databases

The aim of this master's thesis was to design and develop a plugin for the text and source code editor Notepad++. The plugin has two tasks. Firstly, it must be possible to execute SQL statements (including data definition language statements – CREATE, ALTER, DROP, and SELECT … INTO) directly in a database of MS Access desktop database management system (DBMS). Secondly, it must provide feedback to the user about the problems in SQL statements. The problems could be generic to SQL (i.e., possible in many different DBMS) or specific to MS Access. The problems could be errors that prevent executing the statements or these could be code smells that make it more difficult to comprehend the statements. The plugin must be created in a manner that in the future makes it easy to add support to other DBMS's, add checking of new problems, and add new languages that are used to report about problems.

We explored the existing tools in terms of executing SQL statements in MS Access databases and checking SQL statements. We studied the existing Notepad++ plugins that could be extended to MS Access DBMS. We found NppDB from the Notepad++ plugins list. It was developed to make it possible to execute SQL statements in MS SQL Server and SQLite databases. During the work, the plugin was extended by adding the ability to work with MS Access DBMS and the ability to check some problems in SQL statements was also developed.

In terms of problematic SQL statements, the previous studies were examined. The problems that the plugin must be able to detect in SQL statements were determined in collaboration with the supervisor because there are many possible problems. Thus, it is impossible to develop detection of all these as a part of this work.

The functional requirements of the software were described as user stories. The extension of NppDB was created by using the C# language. It uses an OLEDB driver for interacting with the DBMS. It uses the parser generator ANTLR4. The extended plugin allows users to execute all MS Access SQL statements. One can execute multiple statements one after another with one command (i.e., execute a script).

The plugin was tested both by the author himself and by the supervisor and it was validated against other similar programs. Based on the received feedback, the compliance of the plugin with the established requirements was evaluated and improvements were made.

The program that was developed in this thesis is more capable compared with the existing programs that allow users to execute SQL statements in a MS Access database (including MS Access itself) or check SQL statements.

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

The thesis is in Estonian and contains 79 pages of text, 8 chapters, 43 figures, 3 tables.

Lühendite ja mõistete sõnastik

.NET	Microsofti arendatav raamistik programmeerimiseks
AB	Andmebaas
ABS	Andmebaasisüsteem
ANTLR4	Parseri generaator (*ANother Tool for Language Recognition*) [1]
API	Rakendusliides rakenduste omavaheliseks suhtlemiseks (*Application Programming Interface*) [2]
C#	Objektorienteeritud programmeerimiskeel
CR	Reavahetus (*carriage return*)
IDE	Tarkvara arendamise tarkvara (*Integrated Development Environment*)
Lexer	Vahend, mis tükeldab lause sõnastiku reeglite järgi sõnastiku ühikuteks
LF	Reavahetus (*line feed*)
NppDB	Notepad++ pistikprogramm, mis on mõeldud Notepad++ ühendamiseks erinevate andmebaasidega, selleks, et saaks redaktoris SQL lauseid käivitada [3]
ODBC	Andmebaasidraiveri kasutamisel põhinev meetod rakenduste ühendamiseks andmebaasiga (*Open Database Connectivity*) [4]
OLEDB	Standardne API, mis mh võimaldab ühendada rakendust andmebaasiga (*Object Linking and Embedding, Database*) [5]
Parser	Vahend, mis võtab lexeri töö tulemuse ja loob sellest lausete grammatikareeglite järgi konkreetse süntaksipuu
Pistikprogramm	Väiksem programm, mis lisab suuremale programmile uue funktsiooni või teenuse [6]
Scintilla	Vabavaraline lähtekoodi muutmise komponent [7]
SQL	*Structured Query Language*, relatsioonilise andmemudeli alusel väljatöötatud andmebaasikeel [8]
VBA	*Visual Basic for Applications*. "Microsofti sündmustepõhise programmeerimiskeele Visual Basic 6.0 teostus, mis on sisse ehitatud enamikesse töölaua Microsoft Office'i rakendustesse" [9]

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

SQL lausete andmebaasisüsteemis käivitamiseks on mitmeid viise. Töölaua andmebaasisüsteemide nagu Microsoft Access (MS Access) [10] ja LibreOffice Base [11] tarkvara osaks on graafiline kasutajaliides, mille abil on kasutajal võimalik SQL lauseid käivitada ning saada käivitatud lausetele vastused. Serveripõhiste andmebaasisüsteemide puhul tuleb andmebaasisüsteemiga tavaliselt kaasa käsureapõhise kasutajaliidesega terminaliprogramm (nt psql [12] PostgreSQL [13] korral ja SQLcl [14] Oracle Database [15] korral), mille kaudu saab andmebaasimootoriga suhelda ja andmebaasidega töötada. Andmebaasisüsteemi arendajad võivad ka pakkuda eraldi graafilise kasutajaliidesega haldusprogrammi (nt SQL Developer Oracle [16] korral) ja lisaks leidub palju kolmandate osapoolte programme, mis pakuvad andmebaaside haldamiseks ja programmeerimiseks, sh SQL lausete käivitamiseks, graafilist kasutajaliidest. [17] Erinevatel andmebaasisüsteemidel on programmide hulk, mille kaudu saab SQL lauseid käivitada ja seega ka pakutavad võimalused, erinevad.

1.1 Taust ja probleem

Mitmed Tallinna Tehnikaülikooli programmeerimise õppeainete õppejõud eelistavad loengus või seminaris õpilastele demonstreerimise eesmärgil kasutada koodi kirjutamisel ja selle käivitamisel tekstiredaktoreid nagu näiteks Notepad++ [18] või SciTe [19]. Tekstiredaktori kasutamise eeliseks on selle käivitumise kiirus. Kui näiteks tavapärase andmebaasisüsteemi haldamise tarkvara käivitamine võtab küllaltki palju aega, siis tekstiredaktori käivitamine toimub sellega võrreldes reeglina hetkega. Lisaks on redaktorites võimalik kasutada olemasolevaid pistikprogramme [20], mis on arendatud erinevate programmeerimiskeelte või andmebaasisüsteemide jaoks. Tekstiredaktori installeerimisfaili suurus ja jalajälg kettal on samuti palju väiksem. See teeb redaktorist kasuliku töövahendi nii tarkvaraarenduse õppijale kui ka tarkvaraarendajale.

Teksti- ja lähtekoodiredaktor Notepad++ on programmeerijate seas populaarne töövahend. [21] Näiteks avaldatud edetabelis asus see kuuendal kohal. Notepad++'il oli

2022. aasta sügise seisuga olemas võimekus käivitada pistikprogrammi [3] abil SQL lauseid andmebaasisüsteemides MS SQL Server [22] ja SQLite. [23] Küll aga puudus selline funktsionaalsus näiteks MS Access ja PostgreSQL andmebaasisüsteemide jaoks. Pistikprogrammi arendus oli peatunud, sest 2022. aasta sügise seisuga oli viimane muudatus tehtud 2014. aastal.

SQL oli üldises programmeerimiskeelte populaarsusindeksis 2023. aasta aprillis kaheksandal kohal. [24] IEEE Spectrum programmeerimiskeelte 2022. aasta populaarsusindeksis oli SQL kuuendal kohal ja tööandjate poolse keeleoskuse ootuste pingereas esikohal. [25] Tööandjad ootavad, et lisaks muudele keeltele tunneks arendajad kindlasti ka SQLi. MS Access oli andmebaasisüsteemide populaarsuse indeksi [26] andmetel 2023. aasta aprillis kümnendal kohal. See süsteem on olnud esikümnes või selle piirimail kogu populaarsuse indeksi koostamise aja jooksul. [27] Näiteks on selles vahendis graafiline liides päringute koostamiseks, mis realiseerib *Query by Example* graafilist andmebaasikeelt. [28] Kasutaja saab koostada andmekäitluse lause graafilises keeles ja tõlkida selle SQLi ning ka vastupidi. Samuti on selles vahendis head sisseehitatud võimalused aknapõhise andmebaasirakenduse loomiseks. Seega on tegemist hea vahendiga SQLi, SQL-andmebaasi disaini ja aknapõhiste andmebaasirakenduse üldpõhimõtete õppimise jaoks. Samuti sobib see hästi andmebaasirakenduste prototüüpide loomiseks ja ühele kasutajale mõeldud andmebaaside ja andmebaasirakenduste loomiseks.

Samas on selles vahendis kasutatav SQL redaktor 2023. aasta alguse seisuga väga halva kasutatavusega. Näiteks puudub seal koodi elementide värvimine, ei näe ridade järjekorranumbreid, ei saa kasutada koodi kommenteerimist, ei saa kontrollida sulgude tasakaalustatust ja puudub automaatne tabeli ja veergude nimede väljapakkumine SQL koodi kirjutamisel. Samuti puudub vahendisse sisseehitatud võimalus mitme SQL lause korraga üksteise järel käivitamiseks ehk SQL skripti käivitamiseks. Olemasolevates andmebaasides SQL lausete käivitamise programmides puudub üldse võimekus MS Accessi andmebaasis SQL lauseid käivitada (nt JetBrainsi tooted programmeerimiseks, PopSQL [29], HeidiSQL [30]) või pole võimalik käivitada andmekirjelduskeele lauseid (CREATE, ALTER, DROP ja SELECT … INTO) (nt DBeaver [31] ja DBSchema [32]). MS Access on pika ajalooga, kuid seni pole selle arendajad näinud vajadust redaktorit parandada. Õppetöö käigus on paljud üliõpilased avaldanud rahulolematust MS Accessi SQL redaktoriga. Lahenduseks on olnud kirjutada SQL koodi tekstiredaktoris ja

kopeerida siis MS Accessi käivitamiseks. Oleks mugavam, kui lauseid saaks redaktoris otse käivitada.

Notepad++'ile loodud pistikprogramm [3] näitab, et seda vahendit saab täiendada SQL lausete käivitamise võimekusega.

Huvi MS Accessi SQL redaktori parandamiseks kerkis asjaolust, et Tallinna Tehnikaülikoolis on see üks andmebaasisüsteem, mille najal õpitakse SQLi ja andmebaasirakenduse prototüübi loomist. Samas näitab MS Accessi jätkuv populaarsus, et vahendil oleks kindlasti laiem kasutajaskond. Kuna MS Access pole avatud lähtekoodiga, siis eraldi programmi loomine ongi ainuke võimalus, kuidas huviliste kogukond saaks seda MS Accessi puudust parandada.

SQL laused peavad käivitamise õnnestumiseks olema korrektselt koostatud. Samuti peaksid laused olema koostatud viisil, mis muudab nendest arusaamise ja nende parandamise võimalikult lihtsaks. Erinevates teadusartiklites [33] [34] [35] [36] [37] [38] [39] [40] on uuritud tüüpilisi vigu, mida SQL lausete koostajad kõige sagedamini teevad. Samuti on SQLi antimustrite raamatus [41] eraldi peatükk (*Query Antipatterns*) levinud eksimuste kohta SQL päringutes. Iga valesti kirjutatud lause käivitamine andmebaasi vastu võtab kasutajalt tarbetut aega ja koormab ka süsteemi, sest lause tuleb saata andmebaasisüsteemile võib-olla üle arvutivõrgu, andmebaasisüsteem peab seda lauset töötlema ja tagastama vastuse. Seetõttu oleks kasulik, et redaktor tooks enne koodi käivitamist mõned enamlevinud probleemid välja nii, et kasutajal oleks võimalik need lause kirjutamisel juba ka parandada. Selline vahend oleks eeskätt abiks SQLi õppijatele ja kasutajatele, kes SQLi igapäevaselt ei kasuta ja seega kipuvad SQL'i lausete süntaksi ja nendega väljendatavat tähendust ehk semantikat unustama. Vastavat funktsionaalsust Notepad++ ega ka populaarsed SQL lausete käivitamise vahendid ei paku. Sellise funktsionaalsuse jaoks on teaduskirjanduses välja pakutud eraldi programm (liivakast) [42] ja see programm on veebirakendusena realiseeritud [43]. Vahendis on realiseeritud vaid viie antimustri esimene kontroll [44], kuid SQL lausetes teaduskirjanduse alusel esinevaid probleeme on palju rohkem. Kellegi teise hallatava veebirakenduse korral on alati ka usalduse küsimus – kas kasutaja on nõus, et tema lauseid logitakse või kas kasutaja üldse teab seda. Selles mõttes oleks kasutaja enese arvutis olev programm parem.

1.2 Ülesandepüstitus

Töö eesmärgiks on arendada teksti- ja lähtekoodiredaktorile Notepad++ pistikprogramm, millel on kaks ülesannet. Esiteks peab selle abil olema võimalik redaktoris käivitada SQL lauseid töölaua andmebaasisüsteemi MS Access andmebaasis, sh andmekirjelduskeele lauseid. Teiseks peab see andma kasutajale tagasisidet käivitatavates SQL lausetes esinevate probleemide kohta. Programm peaks suutma ära tunda nii selliseid (universaalseid) probleeme, mis võivad esineda erinevate andmebaasisüsteemide SQL dialektis kirjutatud lausetes kui ka MS Accessile spetsiifilisi probleeme. Programm tuleb kirjutada viisil, et teiste andmebaasisüsteemide tuge ja uute probleemide kontrollimist oleks sinna võimalikult lihtne hiljem lisada. Samuti peab saama hiljem lihtsalt lisada uusi keeli, milles raporteerida SQL koodis esinevatest probleemidest. Loodav pistikprogramm peab olema avatud lähtekoodiga ja peab olema maailmale kasutamiseks GitHubis [45] avaldatud. Kasutajaliides ja tagasiside peaks olema inglise keeles, kuid teistesse keeltesse tõlkimine peaks olema võimalik ja lihtne.

1.3 Töö struktuur

Magistritöö koosneb kaheksast peatükist.

Esimese osa moodustab sissejuhatus, milles antakse ülevaade taustast, lahendatavast probleemist ja esitatakse ülesandepüstitus.

Teises peatükis tuuakse välja magistritöö metoodika ja kirjeldatakse arendusega seotud metoodilisi aspekte ja nimetades kasutatavaid tööriistu.

Kolmandas peatükis antakse ülevaade erinevatest töö kontekstis huvipakkuvatest programmidest. Kirjutatakse teksti- ja lähtekoodiredaktorist Notepad++ ning selle pistikprogrammist NppDB. Samuti antakse ülevaade olemasolevatest universaalsetest andmebaasi haldusprogrammidest, mis võimaldavad töötada andmebaasisüsteemiga MS Access.

Neljandas peatükis antakse ülevaade sellest, kuidas teadusartiklites klassifitseeritakse SQL lausetes esinevaid vigu. Samuti analüüsitakse neid vigu, mida käesoleva töö raames arendatud pistikprogramm kontrollib. Teadusartiklites toodud SQL lausetes tehtavad

vead on lisades 2 kuni 5. Lisaks antakse selles peatükis üldine ülevaade SQL lausete korrektsust kontrollivatest programmidest.

Viiendas peatükis esitatakse arendatava tarkvara funktsionaalsed ja mittefunktsionaalsed nõuded nii koodi käivitamise kui ka SQL lausete kontrollimise funktsionaalsusele. Lisaks kirjeldatakse selles peatükis arendusprotsessi, pistikprogrammi töötamise loogikat ning põhjendatud arenduse käigus tehtud erinevaid valikuid. Selles peatükis on toodud ka installeerimise juhis ning info arendatud tarkvara litsentsi kohta.

Kuuendas peatükis valideeritakse arendatud tarkvara. Selleks võrreldakse tarkvara olemasolevate programmidega ja antakse ülevaade juhendaja poolsest kasutamisest.

Seitsmendas peatükis esitatakse pistikprogrammi edasised arendusvõimalused ja tuuakse välja juhised pistikprogrammile teiste andmebaasisüsteemide toe lisamise kohta.

Kaheksandas peatükis on magistritöö kokkuvõte.

2 Metoodika

Käesolevas peatükis kirjutatakse täpsemalt töö tegemise viisist, kirjeldades töö tegemise protsessi ning selle käigus kasutatud töövahendeid.

2.1 Ülevaade töö protsessist

Tegemist on disaini tegevusuuringu põhimõttel läbiviidava tööga [46]. See tähendab, et arendatakse tehnilist tehist (antud juhul pistikprogrammi) ja paralleelselt üritatakse seda kasutusele võtta. Tehise arendamisel tuleb arvestada nii tehtud teadustöö, valdkonna parimate praktikate, tulevaste kasutajate nõuetega kui ka tehise praktilisest kasutusest saadud tagasisidega.

Töö käigus edastas autor regulaarselt juhendajale pistikprogrammi uusi versioone, kes seda oma tööprotsessis kasutas ja teavitas autorit tekkinud vigadest ja uutest soovidest. Sellisel viisil toimus tehise hindamine suure osa tööperioodi jooksul, mitte eraldi etapina tööperioodi lõpus.

Enne uue tarkvara loomisega alustamist otsiti programme, mis võimaldavad MS Accessi andmebaasis SQL lauseid käivitada ja SQL koodi kontrollida, et leida sealt käesoleva programmi jaoks eeskuju ning hiljem loodud programmi nendega võrrelda.

Probleemsete SQL lausete osas uuriti varasemaid uuringuid (nt [33] [34] [35] [36] [37] [38] [40] [39]), et määratleda probleemid, millele leidmisele SQL koodist tuleb magistritöös keskenduda.

Tarkvara funktsionaalsed nõuded esitati kasutuslugudena. [47]

Tulemuse hindamiseks kasutati nii autori kui juhendaja poolt vahendit erinevat tüüpi SQL lausete käivitamiseks erinevates andmebaasides. Koostati testandmebaas ja vigadega SQL lausete hulk, selle andmebaasi põhjal, et hinnata kuidas programm reageerib vigadele SQL lausetes. Kuna SQL on keeleliselt liiane, st seal saab sama ülesannet lahendada paljudel erinevatel viisidel [48], siis pidi see lausete hulk olema küllaltki suur.

2.2 Kasutatud tööriistad

SQL lausete analüüsimiseks ja eeltöötluse tegemiseks ning vigade kontrollimiseks kasutatakse parseri generaatorit. Käesolevas töös saab parseri generaator sisendiks MS Accessi põhise SQL süntaksi grammatikareeglid, mis on koostatud vastavalt MS Accessi dokumentatsioonile ja muudele asjakohastest allikatest saadud juhistele, ja genereerib automaatselt parseri lähtekoodi, mis on võimeline looma konkreetse süntaksipuu parsitud SQL lausetest. Rekursiivse meetodiga puu analüüsimise käigus kogutakse ärireeglitele vastavat informatsiooni (nt vigu ja hoiatusi) ja nende asukohti. Nimetatud asukohad aitavad redaktoris esialgses tekstis vigu ja hoiatusi visuaalselt märgistada.

Käesoleva töö raames uuriti erinevaid võimalusi, kuidas magistritöö eesmärki tehniliselt teostada ning milliseid tehnilisi lahendusi ja olemasolevat tarkvara saaks ära kasutada. Muu hulgas tutvuti *Language Server Protocol*'iga. [49] *Language Server Protocol* pakub funktsioone nagu automaatne lõpetamine (*code completion*), definitsioonile navigeerimine (*go to definition*), viidete leidmine (*find references*), automaatne ümbernimetamine (*rename*) vorminguvahendid (*format*) või kursori alust dokumentatsiooni (*documentation on hover*). [49] Nagu välja toodud loetelust nähtub, on tegemist redaktorite üldiste funktsioonidega, mis oleks mõistlik kasutusele võtta redaktori enda arendamise käigus. Pistikprogrammi arendamisel ei ole selliste, Notepad++ redaktori olemasolevatest funktsioonidest erinevate, funktsioonide kasutamine mõistlik. Lisaks ei aitaks selle kasutuselevõtt kaasa ühegi käesoleva töö funktsionaalse nõude lahendamisele. Nimetatud protokolli kasutamist saaks kaaluda uue redaktori loomise või olemasoleva redaktori edasiarendamise käigus.

Käesoleva töö raames valminud Notepad++ pistikprogramm NppDB on arendatud programmeerimiskeeles C# [50] ja kasutades .NET raamistikku. [51] Suhtlus andmebaasiga toimub läbi OLEDB draiveri. [5] Parseri generaatorina kasutatakse ANTLR4. [1] Arendamisel on tarvis ka Java keskkonda (*Java Runtime Environment*) [52] parseri genereerimiseks. Käesoleva töö raames arendatud tarkvara lähtekoodi hoiustatakse koodihoidlas GitHub. [45]

Tarkvara testimiseks kasutati andmebaasisüsteemi MS Accessi versioone 2019 (64-bit) ja 365 (32-bit) ning Notepad++ versioone 8.4.7, 8.4.9 ja 8.5 (64-bit).

Lõputöö dokumenti hoiti Google pilves, jooniste tegemiseks kasutati rakendusi Lucidchart [53] ja Bizagi [54] ning kohtumiseks juhendajaga kasutati MS Teamsi.

3 Huvipakkuv tarkvara

Selles peatükis kirjutatakse tarkvarast, mis on oluline käesoleva töö kontekstis. Tegemist on kas edasiarendatava tarkvaraga (Notepad++ ja selle pistikprogramm) või loodava tarkvaraga võrreldava tarkvaraga.

3.1 Notepad++ ja selle laiendamine

Notepad++ [18] on vabavaraline avatud lähtekoodiga teksti- ja lähtekoodiredaktor, mis on mõeldud kasutamiseks operatsioonisüsteemil MS Windows. Alates versioonist v3.1 on Notepad++ võimalik laiendada pistikprogrammidega. [55]

Notepad++ põhineb tekstiredaktori komponendil Scintilla. [7]

Redaktorile on arendatud suur hulk erinevaid Notepad++ omaniku poolt aktsepteeritud pistikprogramme erinevateks eesmärkideks. [56] Notepad++ dokumentatsioonis on esitatud juhised, kuidas pistikprogramme Notepad++'ile arendada.

Võimalik on arendada ka pistikprogramme, mida Notepad++ omanik ametlikuks pistikprogrammiks ei registreeri.

3.1.1 NppDB pistikprogramm

Notepad++ mitteametlikku pistikprogrammi NppDB [3] uuendati viimati 2014. aastal. Pistikprogramm on mõeldud redaktori ühendamiseks erinevate andmebaasisüsteemidega. Enne käesoleva töö valmimist toetas pistikprogramm andmebaasisüsteemide MS SQL Server ja SQLite ühendamist redaktoriga.

NppDB võimaldab registreerida, eemaldada, avada ja sulgeda ühendust, käivitada SQL lauseid ja saada ühendatud andmebaasist lausetele vastuseid.

Pistikprogramm NppDB ei võimalda kasutajal töötada Notepad++ redaktoris andmebaasisüsteemiga MS Access.

3.2 MS Accessi andmebaasi haldusprogrammid

Käesolevas jaotises antakse lühiülevaade andmebaasi haldusprogrammidest, millel on andmebaasisüsteemi MS Access tugi. See tähendab, et nende kaudu on võimalik hallata MS Accessi andmebaase, sh käivitada seal SQL lauseid.

3.2.1 Universaalsed andmebaasi haldusprogrammid

Need on programmid, mis võimaldavad luua ühendust erinevate andmebaasisüsteemide andmebaasidega, käivitada seal SQL lauseid ja vaadata tulemusi. Need programmid võivad ka näiteks pakkuda võimalust andmebaasi struktuuri visualiseerimiseks.

MS Accessi tugi on ainult osades sellistes programmides. Järgnevalt on esitatud lühiülevaade leitud universaalsetest andmebaasi haldusprogrammidest ning teksti- ja lähtekoodiredaktoritest, millel on vähemalt mingi suutlikkus MS Accessiga töötada.

- RazorSQL [57] on SQL redaktor, mis võimaldab töötada erinevate andmebaasisüsteemidega, sh MS Access. SQLRazor'i puhul on tegemist tasulise tarkvaraga.
- DBeaver [31] on arendajatele, andmebaasihalduritele ja teistele mõeldud SQL redaktor, mis võimaldab töötada erinevate andmebaasisüsteemidega, sh MS Access. Sellest on nii tasuta kui tasuline versioon.
- DbSchema [32] on visuaalne andmebaasi haldur, mis võimaldab töötada erinevate andmebaasisüsteemidega, sh MS Access. Sellest on nii tasuta kui tasuline versioon.
- Visual Studio [58] on universaalne arendustarkvara, mis võimaldab töötada erinevate andmebaasisüsteemidega, sh MS Access.
- DbVisualizer [59] on andmebaasi tööriist/redaktor, mis võimaldab töötada erinevate andmebaasisüsteemidega, sh MS Access. Sellest on nii tasuta kui tasuline versioon.
- FlySpeed SQL Query [60] on andmebaasi tööriist, mis võimaldab töötada erinevate andmebaasisüsteemidega, sh MS Access.

3.2.2 MS Accessi andmebaasi haldusprogrammid

Lisaks universaalsetele programmidele leiti ka kaks programmi, mis on mõeldud spetsiaalselt MS Accessi andmebaaside kasutamiseks.

MDB Admin [61] on redaktor, mis on mõeldud MS Access andmebaasidega töötamiseks. Kasutaja saab:

- ühenduda nii mdb kui accdb formaadis andmebaasifailidega,
- vaadata andmebaasiobjekte,
- kasutada SQL lausete kirjutamiseks redaktorit, kus on näha rea numbrid ja kasutatakse koodi värvimist (erinevat tüüpi koodi elemendid erineva värviga),
- käivitada nii andmekirjelduskeele kui andmekäitluskeele lauseid,
- luua vaateid ja protseduure,
- luua päringuid graafilise kasutajaliidese kaudu,
- eksportida andmebaasi loomiseks mõeldud SQL lauseid,
- genereerida HTML lehe andmebaasiobjektide kirjeldustega,
- vaadata käivitatud SQL lausete ajalugu ja võtta ette mõni varem käivitatud lause.

Programm ei võimalda:

- sulgude tasakaalustatuse kontrolli SQL redaktoris,
- võtmesõnade, tabeli nimede ja veergude nimede väljapakkumist lausete kirjutamisel,
- lausete käivitamist skriptina (mitu lauset korraga üksteise järel),
- kommentaari kasutamist lause sees,
- kiirklahvi kasutamine lause käivitamiseks (selle asemel tuleb vajutada väikese suurusega nupule),
- vaadata kasutajaliideses korraga mitme päringu tulemust,

- SQL lausete kontrollimist probleemide suhtes.

Iga mitte-SELECT lause käivitamisel väljastatakse hüpikaknas teade, et lause ei tagasta tulemust. Fondi suurus SQL lausete kirjutamise aknas on väike ja seda ei saa suurendada.

MDB Viewer Plus [62] on redaktor, mis võimaldab töötada MS Accessi andmebaasiga. Kasutaja saab:

- ühenduda nii mdb kui accdb formaadis andmebaasifailidega,
- vaadata andmebaasiobjekte,
- kasutada SQL lausete kirjutamiseks redaktorit, kus on näha rea numbrid ja kasutatakse vähesel määral koodi värvimist (erinevat tüüpi koodi elemendid erineva värviga),
- käivitada nii andmekirjelduskeele kui andmekäitluskeele lauseid,
- käivitada lauseid skriptina (mitu lauset korraga üksteise järel),
- luua vaateid,
- vaadata käivitatud SQL lausete ajalugu ja võtta ette mõni varem käivitatud lause.

Programm ei võimalda:

- sulgude tasakaalustatuse kontrolli SQL redaktoris,
- võtmesõnade, tabeli nimede ja veergude nimede väljapakkumist lausete kirjutamisel,
- vaadata kasutajaliideses korraga mitme päringu tulemust,
- kommentaari kasutamist peale lause lõppu,
- SQL lausete kontrollimist probleemide suhtes.

Fondi suurus SQL lausete kirjutamise aknas on väike ja seda ei saa suurendada.

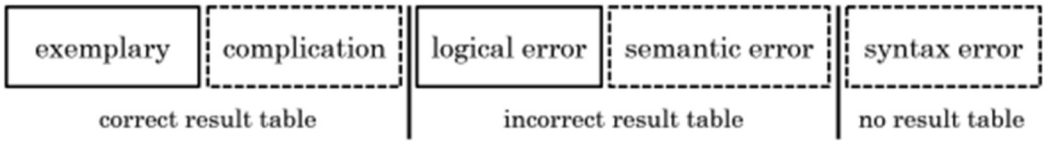
4 SQL lausete tüüpvead

Käesolevas peatükis analüüsitakse teadusartiklites [35] [34] [33] [39] [36] [36] [37] [38] [40] ja õppematerjalis [63] välja toodud SQL lausete kirjutamisel tehtavaid vigu, esitatakse kategooriad, mille alusel käesolevas töös vigu kategoriseeritakse ning tuuakse välja vead, mille esinemist käesoleva töö raames arendatud pistikprogramm kontrollib koos täpsema vea selgitusega.

4.1 Vigade analüüs

Erinevad teadusartiklid kategoriseerivad SQL lausetes tehtavaid vigu erinevalt. Osa artikleid (nt [33], [34]) jagab SQL vead üksnes kaheks: süntaktilised ja semantilised vead. Esimesel juhul ei vasta SQL lause kasutatava andmebaasisüsteemi reeglitele ja selle täitmine ebaõnnestub. Teisel juhul küll kirjutatud SQL lause täitmine õnnestub, kuid lause täitmisega ei saavutata soovitud tulemust.

Samas on teadusartikleid, mis kategoriseerivad SQL vigu veelgi põhjalikumalt. Artiklitest kõige põhjalikum SQL vigade käsitlus oli Taipalus jt artiklis: [35]



Joonis 1. Taipalus jt artiklis esitatud SQL vigade kategooriad.

Nagu jooniselt 1 näha, siis jagatakse artiklis [35] semantilised vead omakorda kaheks: semantilised ja loogilised vead. Lisaks süntaktilistele, semantilistele ja loogikavigadele tuuakse välja ka vead, mis on seotud kirjutatud lausete keerukusega. Keerukusega seotud vead teevad kirjutatud lause täitmise aeganõudvaks või lausest arusaamise raskeks, kuid tulemus on sellele vaatamata korrektne. Loogiliste ja semantiliste vigade puhul on lause täitmise tulemus ebaõige. Vigade avastamisel, mis asuvad joonisel 1 punktiirjoonega kastides, ei ole õige päringu tulemuse teadmine oluline – see tähendab, et vead on avastatavad teadmata korrektset lause täitmise tulemust. Seevastu loogiliste vigade avastamiseks peab olema teada ka oodatav korrektne SQL lause täitmise tulemus. [35]

Selleks, et eristada semantilist viga loogilisest, saab tuua näite, kus andmebaasist päritakse andmeid, kus WHERE klauslis seotakse AND operaatoriga kaks tingimust, mis ei saa olla samaaegselt täidetud (näide joonisel 2 ridadel 8 ja 9 toodud lause). Seega on käivitatava lause tulemuseks alati tühi ridade hulk. Sellisel juhul on tegemist semantilise veaga. Kui aga tegemist ei oleks olnud teineteist välistavate tingimustega (joonisel 2 ridadel 11 ja 12 olev lause), siis loogilise vea olemasolu hindamiseks tuleb teada, millist tulemust sooviti. Kui saadud tulemust ei soovitud, siis järelikult oli tegemist loogilise veaga. Näiteks on joonisel 2 ridadel 11 ja 12 olev lause semantiliselt korrektne. Hindamaks, kas see on ka loogiliselt korrektne, peame teadma päringu tegija kavatsust. Kui ta soovis saada andmebaasist andmeid kõikide 1998. ja 2000. aastal sündinud isikute kohta, siis on lause loogiliselt korrektne. Seevastu, kui ta soovis saada andmeid nt 1999. ja 2000. aastal sündinud isikute kohta, siis on lause loogiliselt ebakorrektne.

```
8   SELECT * FROM Isik
9   WHERE sünniaasta = 2000 AND sünniaasta = 1998;
10
11  SELECT * FROM Isik
12  WHERE sünniaasta = 2000 OR sünniaasta = 1998;
```

Joonis 2. Semantilise ja loogilise vea mõistmist abistavad laused.

Käesolevas töös kasutatakse SQL vigade kategoriseerimisel joonisel 1 toodud liigitust.

Lisades 2 kuni 5 on toodud tabelid teadusartiklites [35] [34] [33] [39] [36] [36] [37] [38] [40] ja õppematerjalis [63] välja toodud SQL vigadest. Lisaks vea kirjeldusele on seal autori hinnang ka selle kohta, kas konkreetne SQL lauses esinev viga on spetsiifiline konkreetsele andmebaasisüsteemile (näiteks TOP predikaati saab kasutada MS Accessis ja MS SQL Serveris, kuid mitte PostgreSQLis) või mitte (näiteks SQL üldiselt lubab ilma nimeta tabelis mitut sama nimega veergu). Samuti esitatakse autori hinnang, kas viga on sõltuv parasjagu kasutatavast andmebaasist või mitte. Osad välja toodud vead on seotud teadusartiklis käsitletud kontekstiga, mis tähendab, et konkreetse vea olemus võibolla näiteks seotud uuringu aluseks oleva ülesande spetsiifikaga. Autor möönab, et mõnikord on vea klassikuuluvust raske hinnata, mõni viga kuulub rohkem kui ühte klassi ning et osad välja toodud vead on teiste väljatoodud vigade täpsustused või erijuhud. Tuleb võtta arvesse ka asjaolu, et vead võivad olla konkreetse artikli-spetsiifilised.

4.1.1 Süntaktilised vead

Süntaktilise veaga lause käivitamisel annab andmebaasisüsteem veateate ning kasutaja antud käsku ei täideta. Süntaktilised vead tekivad, kui kasutaja ei ole järginud lause kirjutamisel konkreetse andmebaasisüsteemiga seotud süntaktilisi nõudeid. Teiste sõnadega ei ole lause kooskõlas andmebaasisüsteemi loojate poolt paika pandud keelereeglitega.

Lisas 2 esitatakse levinud SQL süntaksi vead, mille esinemisel lauses annab andmebaasisüsteem süntaksi vea ja kasutaja soovitud eesmärk lause käivitamisel jääb saavutamata.

4.1.2 Semantilised vead

Käesoleva töö raames käsitletakse semantiliste vigadena vigu, mille korral lause on süntaktiliselt korrektne, kuid selle käivitamisel ei tagastata kasutajale õiget tulemust, kuna SQL lause kirjutamisel on tehtud viga. Täpsemalt, on tegemist veaga, mille üles leidmiseks ei pea teadma, millist päringu tulemust sooviti lause käivitamisel saada. Vea olemasolu saab tuvastada analüüsides lauset ja teadmata, milliseid andmeid sooviti andmebaasist saada. Semantilise vea näide on, kui päritakse andmebaasist andmeid, kus inimene on sündinud 1998. aastal ja 2000. aastal või kui päritakse andmebaasist andmeid isikute kohta, kes on samaaegselt nii elus kui ka surnud. Sellised tingimused ei saa reaalselt kunagi korraga täidetud olla, kuna inimese sünni aeg ja staatus elus/surnud peavad olema üheselt määratletud.

Loetelu erinevates allikates kirjeldatud semantilistest vigadest on esitatud lisas 3.

4.1.3 Loogilised vead

Käesoleva töö raames käsitletakse loogilise veana vigu, mille korral lause on süntaktiliselt korrektne, kuid selle käivitamisel ei tagastata kasutajale õiget tulemust, kuna SQL lause kirjutamisel on tehtud viga. Täpsemalt, on tegemist veaga, mille üles leidmiseks peab teadma lause kirjutaja kavatsust ja soove. Näiteks, kui kasutaja soovib saada andmebaasist andmeid alaealiste kohta, kuid SQL lausesse kirjutab, et päringu tulemuses peavad olema 18 aastased ja vanemad, siis on päringu tulemus vale, kuigi keegi teine, kes andmeid vaatab, ei pruugi peale vaadates aru saada valedest andmetest, kuna ta ei tea, milliseid andmeid sooviti andmebaasist pärida.

Loetelu erinevates allikates kirjeldatud loogilistest vigadest on esitatud lisas 4.

4.1.4 Keerukusega seotud vead

Ainult keerukusega seotud veaga lause on korrektne nii süntaktiliselt, semantiliselt kui ka loogiliselt. Samas on sellise veaga lause keerulisem kui vaja. Teiste sõnadega nende vigade esinemine ei mõjuta väljastatavat tulemust – tulemus on vaatamata keerukusele korrektne, selline, mida kasutaja soovis saavutada. Siiski vähendavad keerukusega seotud vead andmebaasioperatsioonide töökiirust ning SQL lause jälgitavust ja arusaadavust. Keerukusega seotud vea tuvastamiseks on sageli tarvis teada ka oodatavat päringu tulemust ja andmebaasi struktuuri, sest saamaks näiteks aru, kas mingit tabelit tuleb päringus ühendada, peab teadma, milline oli ülesandepüstitus, millised on tabelite võtmed ja kuidas on tabelid omavahel seotud.

Loetelu erinevates allikates kirjeldatud keerukuse vigadest on esitatud lisas 5. Keerukuse vigade puhul ei tooda välja hinnangut selle kohta, kas see sõltub andmebaasisüsteemist või mitte, sest üldiselt sõltub kõigi selliste probleemide mõju lause täitmise kiirusele andmebaasisüsteemist (mõni süsteem saab sellise lause täitmisega hakkama paremini kui teine).

4.2 Käesoleva töö raames käsitletavad vead

Käesolevas jaotises tuuakse välja SQL lausetes esinevad vead, mille esinemist käesoleva töö raames valminud pistikprogramm kontrollib. Osad nendest vigadest takistavad lause täitmist. Osad nendest vigadest on tegelikult probleemid ehk koodi halvad lõhnad, mis halvendavad lausest arusaamist ja lause tulevast täiendamist. Vigade valik tehti koostöös juhendajaga, kes tõi välja, milliseid probleeme ta sageli üliõpilaste kirjutatud lausetes kohtab. Käesolevas töös kasutatakse sõnu “viga” ja “probleem” sünonüümidena.

1. **LIKE operaator ilma metamärkideta**. [33] [34] LIKE operaatorit kasutavas tingimuses ilma metamärkideta („%“ ja „_“) mustri kasutamine on sarnane, kuid mitte samaväärne [64] võrdsuse kontrolli operaatori (nimega “=”) kasutamisega. LIKE operaator koos mustriga, mis sisaldab metamärke, võimaldab otsida ridu, kus veerus olev väärtus vastab etteantud mustrile. Kui metamärke ei kasutata, siis otsitakse veerust täpset vastet. Sellisel juhul tuleks kasutada võrdsuse kontrolli operaatorit. Kuna LIKE operaatori kasutamine ilma metamärkideta üldiselt

tulemust ei mõjuta, siis on tegemist ennekõike keerukusega seotud veaga, kuid võib olla ka loogiline viga, kui kasutaja unustas mustrisse metamärgid lisada. Joonisel 3 on näide SQL lausest, kus vastav viga esineb.

```
 98    SELECT *
 99    FROM Hotell
100    WHERE nimi LIKE 'Viru';
```

Joonis 3. Näide SQL lausest, kus esineb probleem nr 1.

2. **Otsingutingimuses: =NULL**. [63] NULL pole väärtus, vaid marker ehk tähis, mis tähistab väärtuse puudumist. Väärtuse puudumise kontrollimiseks on SQLis ettenähtud operaator IS NULL. Kui päringu tingimus on veerg=NULL, siis pole tulemuses ühtegi rida, sest iga rea korral on tingimuse kontrolli tulemus UNKNOWN ehk teadmata. Kuna süntaktiliselt =NULL viga ei ole, kuid seda sisaldav lause annab vale tulemuse, siis on tegemist semantilise veaga. Joonisel 4 on näide SQL lausest, kus vastav viga esineb.

```
102    SELECT *
103    FROM Reserveerimine
104    WHERE lopu_aeg=NULL;
```

Joonis 4. Näide SQL lausest, kus esineb probleem nr 2.

3. **Otsingutingimuses: <>NULL**. [63] Väärtuse olemasolu kontrollimiseks on ettenähtud operaator IS NOT NULL, mitte tingimus <>NULL. Kuna süntaktiliselt <>NULL viga ei ole, kuid seda sisaldav lause annab vale tulemuse, siis on tegemist semantilise veaga. Joonisel 5 on näide SQL lausest, kus vastav viga esineb.

```
106    SELECT *
107    FROM Reserveerimine
108    WHERE lopu_aeg<>NULL;
```

Joonis 5. Näide SQL lausest, kus esineb probleem nr 3.

4. **Otsingutingimus: >NULL, >=NULL,<NULL, <=NULL**. [63] NULL ei ole väärtus ja seda ei ole korrektne niimoodi päris väärtustega võrrelda. Võrdluse tulemus on alati UNKNOWN ehk teadmata. Tegemist on semantilise veaga. Joonisel 6 on näide SQL lausest, kus vastav viga esineb.

```
110  SELECT *
111  FROM Reserveerimine
112  WHERE lopu_aeg>NULL;
```

Joonis 6. Näide SQL lausest, kus esineb probleem nr 4.

5. **Otsingutingimus pole loogikaavaldis (nt hinne=1 OR 2)**. [63] Sõltuvalt andmebaasisüsteemist võib tulemuseks olla süntaksiviga, kuid lause võib ka käivituda. Näiteks Joonisel 7 toodud lause käivitub MS Accessis, kuid ei käivitu PostgreSQLis. Tegemist on semantilise veaga ja lause mittekäivitumisel ka süntaksiveaga. Joonisel 7 on näide SQL lausest, kus vastav viga esineb.

```
118  SELECT *
119  FROM Hotell
120  WHERE linn='Tallinn' OR 'Tartu';
```

Joonis 7. Näide SQL lausest, kus esineb probleem nr 5.

6. **Otsingutingimuses: <veerg> LIKE <veerg>**. [63] [35] Kui soovida otsida ridu, kus kahes veerus on samasugune väärtus, siis tuleks kasutada võrdsuse kontrolli operaatorit. Joonisel 8 on tegemist süntaktiliselt korrektse lausega. Tegemist on keerukusega seotud veaga, mis võib mõjutada lause täitmise kiirust. Sõltuvalt kasutatavast andmebaasisüsteemist ja veergude andmetüüpidest võib tegemist olla ka semantilise veaga, kui andmebaasisüsteem ei paku selliseks võrdluseks operaatorit ja ei vii ise läbi tüübiteisendust. Näiteks PostgreSQLis ei saa operaatori puudumise tõttu sellisel viisil täisarvulistes veergudes olevaid väärtuseid võrrelda, kuid MS Accessis saab.

```
137  SELECT *
138  FROM Hotell
139  WHERE linn LIKE nimi;
```

Joonis 8. Näide SQL lausest, kus esineb probleem nr 6.

7. **Päringu tulemuses on rohkem kui üks sama nimega veerg**. [35] Päringu tulemuses ei saa hiljem sellistele veergudele nime järgi viidata. Päringu kirjutaja võis paluda ekslikult väljastada samu andmeid mitu korda. Tegemist on süntaktiliselt korrektse lausega, kuid semantiliselt ebakorrektsega. Joonisel 9 on näide SQL lausest, kus vastav viga esineb.

```
144     SELECT Hotell.linn, Hotell.nimi AS linn
145     FROM Hotell;
```

Joonis 9. Näide SQL lausest, kus esineb probleem nr 7.

8. **SELECT klauslis on avaldis (sh konstant või funktsiooni poole pöördumine), kuid vastavale veerule pole ise antud nime**. [63] Sellisel juhul annab veerule nime andmebaasisüsteem. MS Accessi puhul on veeru nimi sellisel juhul näiteks „Expr1000“ või sarnane, mis ei ütle kasutajale midagi sisulist ja seetõttu tuleb sellist nime vältida. Siiski on tegemist süntaktiliselt korrektse lausega. Tegemist on keerukusega seotud veaga (tulemusest on keeruline aru saada) ning ka semantilise veaga. Joonisel 10 on näide SQL lausest, kus vastav viga esineb.

```
156     SELECT hotelli_nr, 'ilus', linn
157     FROM Hotell;
```

Joonis 10. Näide SQL lausest, kus esineb probleem nr 8.

9. **GROUP BY ja DISTINCT samal lause tasemel**. DISTINCT predikaati kasutatakse, et eemaldada tulemusest korduvad read. [34] [35] GROUP BY lauset kasutatakse samasuguste väärtustega ridade grupeerimiseks ehk rühmitamiseks ja see võimaldab saavutada DISTINCT’i kasutamisega samasuguse tulemuse. Nende koos kasutamine ei anna kasutajale lisandväärtust. Tegemist on keerukusega seotud veaga. Joonisel 11 on näide SQL lausest, kus vastav viga esineb.

```
180     SELECT DISTINCT külalise_nr
181     FROM Reserveerimine
182     GROUP BY külalise_nr;
```

Joonis 11. Näide SQL lausest, kus esineb probleem nr 9.

10. **ORDER BY <arv>**. [63] Sellise ORDER BY klausli kasutamine sorteerib tulemuse päringu tulemuses oleva veeru järjekorranumbri järgi. Süntaktiliselt on tegemist korrektse lausega. Tegemist on keerukuse veaga, kuna veergude järjekorra muutmine lähtetabelis (SELECT * korral) või SELECT klauslis tingib vajaduse muuta sorteerimiseeskirja. Joonisel 12 on näide SQL lausest, kus vastav viga esineb.

```
184     SELECT *
185     FROM Hotell
186     ORDER BY 3;
```

Joonis 12. Näide SQL lausest, kus esineb probleem nr 10.

11. **Kokkuvõttefunktsioone kasutava avaldise SUM()/COUNT() kasutamine kokkuvõttefunktsiooni AVG asemel**. [63] Aritmeetilise keskmise leidmiseks on SQLis olemas kokkuvõttefunktsioon AVG. Sellise avaldise kasutamine on süntaktiliselt lubatud. Samas, kui näiteks veerus *palk* on lubatud väärtuse puudumine, siis ei anna need kaks lauset ühesugust tulemust:

```
SELECT Sum(palk)/Count(*) AS keskmine
FROM Osalus_projektis;

SELECT Avg(palk) AS keskmine
FROM Osalus_projektis;
```

Seega on tegemist kas loogilise veaga või keerukuse veaga. Joonisel 13 on näide SQL lausest, kus vastav viga esineb.

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

Joonis 13. Näide SQL lausest, kus esineb probleem nr 11.

12. **SUM(1) kasutamine COUNT(*) asemel**. [63] Kokkuvõttefunktsiooni SUM kasutamisel viisil SUM(1) liidab summale ühe iga rea kohta. Kui tabelis ei ole ühtegi rida, siis `SELECT Sum(1) AS arv FROM Tabel;` tulemuseks on nimeta tabel, kus on üks rida, kus on NULL. Samas `SELECT Count(*) AS arv FROM Tabel;` tulemuseks on nimeta tabel, kus on üks rida, kus on väärtus 0. Süntaktiliselt on lubatav kasutada avaldist SUM(1). Seega on tegemist kas loogilise veaga või keerukuse veaga. Joonisel 14 on näide SQL lausest, kus vastav viga esineb.

```
191     SELECT Sum(1) AS arv
192     FROM Reserveerimine;
```

Joonis 14. Näide SQL lausest, kus esineb probleem nr 12.

13. **INSERT lause INSERT klauslis puuduvad veerunimed**. [63] Veerunimede puudumine on süntaktiliselt lubatav. Veerunimede puudumisel tuleb anda ette väärtused kõikidele veergudele, sõltumata vaikeväärtuste olemasolust nendel veergudel. Kui lisada tabelisse uus veerg või muuta tabelis veergude järjekorda, siis olemasolev lause enam ei tööta või annab vale tulemuse (andmed paigutatakse valedesse väljadesse) ning lause tuleb ümber kirjutada. Tegemist on keerukuse veaga. Joonisel 15 on näide SQL lausest, kus vastav viga esineb.

```
194    INSERT INTO Hotell
195    VALUES (1, 'Top', 'Tallinn');
```

Joonis 15. Näide SQL lausest, kus esineb probleem nr 13.

14. **SELECT * alampäringus, mille tulemuse põhipäringuga võrdlemiseks: IN/NOT IN/=ANY/<>ALL/=/<>/>/>=/</<= operaatorit**. [63] Nimetatud operaatorite puhul peab vasakoperand olema skalaar ja paremoperand loend skalaaridest. See tähendab, et paremoperand ei saa olla loend väärtuste kogumitest. Kui SELECT * lause tehakse ühe veeruga tabeli vastu, millele hiljem lisandub uus veerg, siis kui algul lause töötas, siis pärast veeru lisamist see enam ei tööta. Pärast veeru lisamist on tegemist süntaktilise veaga. Seda saab ka pidada semantiliseks veaks ja keerukuse veaks. Joonisel 16 on näide SQL lausest, kus vastav viga esineb.

```
12    SELECT *
13    FROM Reserveerimine
14    WHERE hotelli_nr IN (SELECT *
15    FROM Hotell
16    WHERE nimi='Viru');
```

Joonis 16. Näide SQL lausest, kus esineb probleem nr 14.

15. **FROM klauslis rohkem kui üks tabel ja SELECT klauslis * (ilma täpsustava tabelita)**. [63] Süntaktiliselt on lause korrektne. Tulemuses on andmed kõikidest nende tabelite veergudest, sh on tulemuses dubleeritud andmed nendest kandidaatvõtme ja välisvõtme veergudest, üle mille toimub tabelite ühendamine. Päringu lugeja ei saa aru, millised on täpselt päringu tulemuses olevad veerud. Tegemist on semantilise veaga ja ka keerukuse veaga. Joonisel 17 on näide SQL lausest, kus vastav viga esineb.

```
91  SELECT *
92  FROM Hotell INNER JOIN Reserveerimine ON Hotell.hotelli nr = Reserveerimine.hotelli nr;
```

Joonis 17. Näide SQL lausest, kus esineb probleem nr 15.

16. **OUTER JOIN + COUNT(*)**. [63] Välisühendamise ehk OUTER JOIN puhul tagastatakse COUNT(*) funktsiooniga tulemuste arv põhitabeli ridade järgi sõltumata sellest, kas ühendamise tingimus on täidetud või mitte. Seega, kui ülesandeks on leida iga põhitabeli rea kohta, kui mitu seotud rida on sõltuvas tabelis, siis annab see põhitabeli ridade korral, millel pole ühtegi seotud rida sõltuvas tabelis, tulemuseks 1, mitte 0. Tegemist on semantilise veaga. Joonisel 18 on näide SQL lausest, kus vastav viga esineb.

```
215  SELECT Hotell.hotelli_nr, nimetus, Count(*) AS arv
216  FROM Hotell LEFT JOIN Reserveerimine ON Hotell.hotelli_nr=Reserveerimine.hotelli_nr
217  GROUP BY Hotell.hotelli nr, nimetus;
```

Joonis 18. Näide SQL lausest, kus esineb probleem nr 16.

17. **UNION [ALL] lauses SELECT *.** [63] Väljatoodud operatsioonide puhul peavad päringutulemuste veergude arv ja veergude tüübid ühtima. Lause lugeja ei saa aru, millised veerud on päringu tulemuses. Kui lauses kasutatud tabelisse lisatakse uusi veerge või muudetakse veergude järjekorda, siis SELECT * kasutamise puhul UNION lause enam ei tööta või annab see vale tulemuse. Seega tuleb lause ümber kirjutada. Tegemist on keerukuse veaga ja ka semantilise veaga. Pärast veeru lisandumist ühe SELECT lause tulemusse on tegemist süntaktilise veaga. Joonisel 19 on näide SQL lausest, kus vastav viga esineb.

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

Joonis 19. Näide SQL lausest, kus esineb probleem nr 17.

18. **Alampäringus ORDER BY (välja arvatud siis, kui seal on TOP n)**. [33] [35] Kuna peale TOP ridade leidmise ei ole MS Accessi päringutes operatsiooni, mis arvestaks ridade järjekorraga, siis on ORDER BY klausli kasutamine alampäringus kasutu. Tegemist on keerukusega seotud veaga. Mõnes andmebaasisüsteemis on see ka süntaksi viga (nt Oracle). Joonisel 20 on näide SQL lausest, kus vastav viga esineb.

```
88    SELECT *
89    FROM Hotell
90    WHERE hotelli_nr IN (SELECT hotelli_nr
91    FROM Reserveerimine
92    ORDER BY hotelli_nr);
```

Joonis 20. Näide SQL lausest, kus esineb probleem nr 18.

19. **INSERT .. SELECT *.** [63] Lause lugeja ei saa aru, milliseid andmeid tabelisse lisatakse. Kui SELECT * lause aluseks oleva tabeli veergude hulk või järjekord muutub, siis lause enam ei tööta või töötab valesti. Siis tuleb lause ümber kirjutada. Tegemist on keerukuse veaga ja ka semantilise veaga. Pärast veeru lisamist tabelisse, mille põhjal päring tehakse, on tegemist süntaktilise veaga. Joonisel 21 on näide SQL lausest, kus vastav viga esineb.

```
94    INSERT INTO Külaline_koopia
95    SELECT *
96    FROM Külaline;
```

Joonis 21. Näide SQL lausest, kus esineb probleem nr 19.

20. **Tekstilised väärtused jutumärkides (aga äkki identifikaator?)**. [35] SQL-is peavad tekstilised väärtused olema esitatud apostroofide vahel (nt 'tekst'). MS Access aktsepteerib ka tekstilisi väärtusi jutumärkide vahel (nt "tekst"). SQLis võib identifikaatoreid e nimesid panna jutumärkidesse ja siis on need tõstutundlikud piiritletud identifikaatorid. SQLis üldiselt on tegemist süntaktilise veaga, kuid MS Accessi puhul semantilise veaga. Joonisel 22 on näide SQL lausest, kus vastav viga esineb.

```
263   SELECT *
264   FROM Hotell
265   WHERE linn="Viru";
```

Joonis 22. Näide SQL lausest, kus esineb probleem nr 20.

21. **Otsingutingimuses nii AND (BETWEEN x AND y ei lähe arvesse) kui OR, kuid pole sulge**. [35] [37] [38] [63] Rea otsingutingimusele vastavuse hindamisel viiakse AND operatsioonid läbi enne OR operatsioone. Otsingutingimuse alamtingimuste kontrolli järjekorda saab mõjutada sulgude kasutamisega.

Sulgude puudumine ei pruugi anda vale tulemust, kuid kindlasti on sulgude mittekasutamise tulemusena vead tõenäolisemad. Samuti parandab sulgude kasutamine tingimuse arusaadavust. Sulgude kasutamine kinnitab soovitud tingimuste kontrolli järjekorda. Süntaktiliselt on tegemist korrektse lausega. Tegemist on keerukuse veaga (lausest on keerulisem aru saada). Tegemist võib olla loogilise veaga. Joonisel 23 on näide SQL lausest, kus vastav viga esineb.

```
126    SELECT *
127    FROM Hotell
128    WHERE linn='Tallinn' OR 'Tartu' AND 'Pärnu';
```

Joonis 23. Näide SQL lausest, kus esineb probleem nr 21.

22. **HAVING klauslis tingimus, mis ei sisalda kokkuvõttefunktsiooni tulemusega võrdlemist**. [34] [35] HAVING klausel on mõeldud tulemuste piiramiseks kokkuvõttefunktsioonide tulemuste alusel. Kui soovitakse piirata grupeerimise e rühmitamise sisendiks olevaid ridu, siis tuleks piirang kirjutada WHERE klauslisse. Tegemist on semantilise veaga ja ka keerukuse veaga, sest efektiivsem on kõigepealt eemaldada mittesoovitavad read ja siis tulemust rühmitada võrreldes rühmitamisega ja siis tagantjärgi ebavajalike rühmade eemaldamisega. Joonisel 24 on näide SQL lausest, kus vastav viga esineb.

```
61    SELECT hotelli_nr, Count(*) AS arv
62    FROM Reserveerimine
63    GROUP BY hotelli_nr
64    HAVING hotelli_nr=1;
```

Joonis 24. Näide SQL lausest, kus esineb probleem nr 22.

23. **{=|<>}<tekst kus on % või _>**. [35] Tõenäoliselt on soovitud kasutada LIKE operaatorit ja kontrollida vastavust mustrile. Tegemist on süntaktiliselt korrektse lausega, mis võib, kuid ei pruugi, olla loogiliselt ebakorrektne. Joonisel 25 on näide SQL lausest, kus vastav viga esineb.

```
76    SELECT *
77    FROM Hotell
78    WHERE 'T%'=linn;
```

Joonis 25. Näide SQL lausest, kus esineb probleem nr 23.

24. **SELECT klauslis kokkuvõttefunktsioon + veerud, kuid puudub GROUP BY**. [33] [37] [40] Kui SELECT klauslis on valitud veerud ja pöördutakse kokkuvõttefunktsiooni poole, siis tuleb GROUP BY klauslis esitada ridade grupeerimise e rühmitamise eeskiri. Seal klauslis tuleb viidata samadele veergudele, millele viidatakse SELECT klauslis. Kui seda ei tehta, siis lause täitmine ei õnnestu. Seega on tegemist süntaktilise veaga. Joonisel 26 on näide SQL lausest, kus vastav viga esineb.

```
200  SELECT Hotell.hotelli_nr, nimetus, Count(Reserveerimine.hotelli_nr) AS arv
201  FROM Reserveerimine RIGHT JOIN Hotell ON Hotell.hotelli nr=Reserveerimine.hotelli nr;
```

Joonis 26. Näide SQL lausest, kus esineb probleem nr 24.

25. **Kokkuvõttefunktsioon WHERE klauslis või GROUP BY klauslis**. [35] [40] [36] Tegemist on süntaktilise veaga, kuna andmebaasisüsteemid sellist lauset käivitada ei lase. Joonistel 27 ja 28 on näited SQL lausetest, kus vastavad vead esinevad.

```
211  SELECT hotelli_nr, Count(*) AS arv
212  FROM Reserveerimine
213  GROUP BY Count(*);
```

Joonis 27. Näide SQL lausest, kus esineb probleem nr 25 GROUP BY klauslis.

```
207  SELECT Hotell.hotelli_nr, nimetus, Count(Reserveerimine.hotelli_nr) AS arv
208  FROM Reserveerimine RIGHT JOIN Hotell ON Hotell.hotelli_nr=Reserveerimine.hotelli_nr
209  WHERE Count(Reserveerimine.hotelli_nr)<2;
```

Joonis 28. Näide SQL lausest, kus esineb probleem nr 25 WHERE klauslis.

26. **=ALL kasutamine =ANY asemel**. [63] Väga võimalik, et kasutaja eesmärk on olnud leida kõik read ühest tabelist, millel on vähemalt üks seotud rida teises tabelis. Siis tuleks kasutada tingimuses = ANY. See tähendab, et põhitabeli reas olev väärtus peab võrduma vähemalt ühe väärtusega alampäringu tulemuses. Tegemist võib olla loogilise veaga. Joonisel 29 on näide SQL lausest, kus vastav viga esineb. Antud juhul otsitakse hotelle, mille number võrdub kõigi alampäringu tulemusena leitud hotelli numbritega. See päring tagastab hotelli, kui tegemist on ainsa hotelliga, kus tehtud reserveerimisi on registreeritud.

```
194   SELECT *
195   FROM Hotell
196   WHERE hotelli_nr=ALL (SELECT hotelli_nr
197   FROM Reserveerimine);
```

Joonis 29. Näide SQL lausest, kus esineb probleem nr 26.

27. **<>ANY kasutamine <>ALL asemel**. [63] Väga võimalik, et kasutaja eesmärk on olnud leida kõik read ühest tabelist, millel pole ühtegi seotud rida teises tabelis. Siis tuleks kasutada tingimuses <> ALL. See tähendab, et põhitabeli reas olev väärtus ei tohi võrduda mitte ühegi väärtusega alampäringu tulemuses. Tegemist võib olla loogilise veaga. Joonisel 30 on näide SQL lausest, kus vastav viga esineb.

```
199   SELECT *
200   FROM Hotell
201   WHERE hotelli_nr<>ANY (SELECT hotelli_nr
202   FROM Reserveerimine);
```

Joonis 30. Näide SQL lausest, kus esineb probleem nr 27.

28. **SELECT alampäringu tulemuses on mitu veergu juhul kui alampäringut kasutatakse IN, NOT IN, =ANY, =SOME ja <>ALL operaatoritega tingimuses**. [63] Nende operaatorite kasutamise puhul peab vasakoperand olema skalaar ja paremoperand loend skalaaridest. See tähendab, et paremoperand ei saa olla loend väärtuste kogumitest. Tegemist on süntaktilise veaga. See on andmebaasisüsteemi (MS Access) spetsiifiline, sest näiteks MS Accessis ei saa järgnevat lauset kasutada, kuid PostgreSQLis saab:

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

Joonisel 31 on näide SQL lausest, kus vastav viga esineb.

```
194   SELECT *
195   FROM Reserveerimine
196   WHERE hotelli_nr NOT IN (SELECT hotelli_nr, linn
197   FROM Hotell
198   WHERE nimi='Viru');
```

Joonis 31. Näide SQL lausest, kus esineb probleem nr 28.

4.3 Programmid, mis on võimelised selliseid vigu tuvastama

Lisaks käesoleva töö raames arendatud vigade kontrollile on samalaadne võimekus olemas ka JetBrains'i arenduskeskkonna integreeritud programmeerimiskeskkonnas (IDE) ja SQLFluff käsureapõhises töövahendis. [65]

JetBrains'i tarkvara ei võimalda töötada MS Access andmebaasisüsteemiga, kuid võimaldab kontrollida SQL lausetes olevaid vigu läbi staatilise analüüsi. Samas on IDE sellise funktsionaalsuse kasutamiseks mitte-arendajale liiga keeruline, sest IDE paigaldamine on vaid SQL vigade kontrollimiseks liiga suur ettevõtmine ja lisaks on IDE ka aeglane ning kohmakas. Autor testis JetBrains IDE't just seetõttu, et seda vahendit kasutati ka käesoleva töö raames valminud tarkvara arendamisel.

SQLFluff on linttöövahend (*linter*), mis töötab programmeerimiskeele Python keskkonnas. Kuna linttöövahend töötab vaid analüüsivahendina, siis see ei sisalda andmebaasi klientrakendust. Sellise vahendi kasutusmugavus ei ole kuigi hea, kuna nii käsud sisestatakse kui ka tulemused kuvatakse käsuterminalis, kus vead on toodud loeteluna ja viidetega analüüsitud skriptile. Kasutaja jaoks on ebamugav neid kahte omavahel visuaalselt kokku viia. Seega on oluliselt mugavam, kui vead kuvatakse kasutajale välja redaktoris, vastava SQL lause lähedal, kus viga esines.

Lisaks JetBrains'i ja FluffSQL tarkvarale uuriti erinevaid andmebaasi klientrakendusi, mis võiksid osata ka analüüsida SQL lausete korrektsust. Uuriti järgmisi programme: RazorSQL, DbSchema, DBeaver, MDB Admin, MDB Viewer Plus ja DbVisualizer Pro. Ükski neist programmidest siiski sellist vigade kontrollimise funktsionaalsust ei sisaldanud. DBeaver hoiatas UPDATE ja DELETE lausete eest, mis ei sisaldanud WHERE klauslit. Selline lause võib olla loogiline viga. Nii andmebaasi klientrakendusi kui ka SQL linttöövahendeid on palju teisigi, kuid sellist, kus need kaks koos toimiksid (kus redaktoris kuvatakse SQL lausete vigu välja), ei suutnud autor rohkem leida (peale JetBrains'i arenduskeskkonna tarkvara).

QuerySandbox [43] on eksperimentaalne veebikeskkond SQL antimustrite tuvastamiseks. See loodi SQLi õppimise uurimiseks. Allika [44] kohaselt kontrollib QuerySandbox SQL lausetes viie antimustri esinemist. Need on SELECT ja GROUP BY klausli mittekooskõla, väärtuse puudumise/esinemise vale kontrollimine (=NULL/<>NULL), LIKE predikaadil põhinev mustripõhine otsing, SELECT *

kasutamine (veergudele ei viidata ilmutatud kujul) ja juhuslike ridade otsimine. Veebikeskkond ei ole seotud konkreetse andmebaasisüsteemiga. Kontrollid põhinevad lause analüüsimisel – kontrolliprogramm ei vaja infot andmebaasi kohta, milles see lause käivitatakse.

Täpsem info vigade kohta, mida need programmid suudavad kontrollida on esitatud jaotises 6.2.

5 Notepad++ pistikprogramm

Selles peatükis esitatakse käesoleva töö raames arendatava pistikprogrammi eesmärgid, nõuded ning põhjendatakse erinevate komponentide valikut. Samuti kirjeldatakse pistikprogrammi tööpõhimõtet ja arenduse käiku, antakse ülevaade kasutajaliideses tehtud muudatustest ja pistikprogrammi testimisest. Lisaks esitatakse loetelu eelduslikest puudustest ning antakse installeerimise juhised.

5.1 Eesmärgid

Käesoleva töö raames valmiv pistikprogramm peab võimaldama kasutajal käivitada piiranguteta MS Accessi jaoks aktsepteeritavaid SQL lauseid. Programm peab eristama SQL lauseid kommentaaridest. Lisaks peab pistikprogramm võimaldama kasutajal kontrollida SQL lauseid probleemide esinemise suhtes jaotises 4.2 toodud ulatuses.

5.2 Funktsionaalsed nõuded

Käesolevas jaotises tuuakse välja nõuded arendatavale pistikprogrammile, mis tuleb realiseerida. Nõuded on jagatud funktsionaalseteks ja mittefunktsionaalseteks.

5.2.1 MS Accessi andmebaasis koodi käivitamise funktsionaalsus

Käesolevas jaotises esitatakse MS Accessi koodi käivitamise funktsionaalsusele esitatavad funktsionaalsed nõuded kasutuslugudena. Tabelis 1 on toodud nimekiri kasutuslugudest, mis käesoleva töö raames pistikprogrammina realiseeriti.

Tabel 1. Pistikprogrammi SQL lausete käivitamisega seotud kasutuslood.

Identifikaator	**Kasutuslugu**
1	Kasutajana soovin luua Notepad++ redaktori ja MS Access andmebaasi vahel ühenduse, et kasutada Notepad++ redaktorit andmebaasisüsteemi MS Access andmebaasides SQL abil töötamisel.

Identifikaator	Kasutuslugu
2	Kasutajana soovin Notepad++ redaktoris töötada mitme MS Access andmebaasiga üheaegselt, et mitte kulutada aega andmebaasiühenduse sulgemise ja uuesti loomise peale.
3	Kasutajana soovin sisestada Notepad++ redaktorisse SQL lauseid, et neid MS Accessi andmebaasis käivitada.
4	Kasutajana soovin töötada ühe andmebaasiga samal ajal erinevates akendes, et erinevate eesmärkidega lauseid lahus hoida.
5	Kasutajana soovin Notepad++ redaktoris käivitada kõiki SQL lauseid, mida MS Access võimaldab käivitada, et redaktoris ei oleks MS Accessi kasutamisel piiranguid.
6	Kasutajana soovin käivitada mitut SQL lauset korraga, et ei peaks kulutama aega lausete ükshaaval käivitamisele.
7	Kasutajana soovin jätta lausetesse kommentaare ilma, et need segaksid lausete käivitamist, et saaksin säilitada märkmeid ja selgitusi kirjutatud lause kohta ning saaksin säilitada mittetöötavaid lauseid.
8	Kasutajana soovin käivitada nii valitud lauseid kui ka lauset, kus parasjagu asub kursor selleks, et saaksin efektiivsemalt tööd teha.
9	Kasutajana soovin saada tagasisidet SQL lausete käivitamise tulemuste kohta, et veenduda lausete edukas käivitamises.
10	Kasutajana soovin mitme SELECT lause ehk päringu käivitamisel saada vastused iga päringu kohta eraldi aknas, et eristada erinevate päringute tulemusi üksteisest.
11	Kasutajana soovin näha andmebaasi ühenduse aknas nii tabelite kui ka vaadete struktuuri, et mul oleks hea ülevaade veergudest, võtmetest ja indeksitest.

Identifikaator	Kasutuslugu
12	Kasutajana soovin, et andmebaasi tabelite kuvamisel ühenduse aknas ei oleks näha süsteemikataloogi tabeleid, et säilitada hea ülevaade olulistest andmebaasiobjektidest.
13	Kasutajana soovin, et andmebaasiga ühenduse sulgemisel sulguks ka aknad, kus esitatakse käivitatud lausete kohta käivaid teateid ja SELECT lausete tulemusi, et ma ei peaks seda käsitsi tegema.
14	Kasutajana soovin mitme SELECT lause käivitamisel tulemuste kuvamist erinevatel sakkidel, et mul oleks parem ülevaade käivitatud lausete tulemustest.
15	Kasutajana soovin, et pistikprogrammi abil saaks avada olemasolevat andmebaasi või luua uue andmebaasi, et saaksin töötada nii olemasoleva kui ka uue andmebaasiga. Uue andmebaasi loomisel olemasoleva nimega, kirjutatakse see andmebaas tühja andmebaasiga üle.
16	Kasutajana soovin, et SELECT lause käivitamisel tehakse leitud tulemuste sakk automaatselt aktiivseks, et ma ei peaks vastavat sakki käsitsi valima.
17	Kasutajana soovin, et SELECT lause tulemuse sakil oleks kuvatud lause käivitamise aeg, lause ise ja käivitatud lausega vastuseks saadud ridade arv, et mitmete lausete käivitamise järel saaksin aru, millises aknas millised käivitatud lausete tulemused kuvati.
18	Kasutajana soovin sulgeda sakke ja saki sulgemisel peab aktiivseks jääma eelviimane sakk (või kui rohkem sakke avatud ei ole, siis logi sakk), et sakkide käitumine oleks sarnane veebibrauseri sakkide käitumisele.

5.2.2 SQL lausete kontrollimise funktsionaalsus

Käesolevas jaotises esitatakse SQL lausete kontrollimise funktsionaalsusele esitatavad funktsionaalsed nõuded kasutuslugudena. Tabelis 2 on toodud nimekiri kasutuslugudest, mis käesoleva töö raames pistikprogrammina realiseeriti.

Tabel 2. Pistikprogrammi SQL lausete kontrollimisega seotud kasutuslood.

Identifikaator	**Kasutuslugu**
1	Kasutajana soovin, et pistikprogrammi funktsionaalsused (MS Access koodi käivitamine, vigade kontroll) peavad töötama teineteisest lahus või koos, et saaksin ise valida kumba funktsionaalsust kasutada ning funktsionaalsused ei segaks üksteist.
2	Kasutajana soovin, et Notepad++ redaktor kontrolliks SQL lauseid selliste probleemide esinemise suhtes, mille kontroll ei eelda konkreetse andmebaasi struktuuri tundmist, et veenduda käivitatavate lausete korrektsuses.
3	Kasutajana soovin saada tagasisidet sisestatud lausete vigade kohta vigase lause osa juures, et vea tegemise koha üles leidmine ja vea parandamine toimuks kiiresti.
4	Kasutajana soovin, et käivitatavas lauses probleemide tuvastamine ei takistaks lause käivitamist, et saaksin ise olla otsustaja, millist lauset käivitatakse.
5	Kasutajana soovin, et saaksin valitud lauseid või kõiki lauseid kontrollida või käivitada, et saaksin ise otsustada, mis ulatuses soovin lausete kontrollimist ja käivitamist.
6	Kasutajana soovin, et saaksin pistikprogrammi vigade kontrolli tulemused redaktorist eemaldada, et kontrolli tulemused ei häiriks ülevaate saamist lausetest.

5.3 Mittefunktsionaalsed nõuded

Käesolevas jaotises esitatakse arendatava pistikprogrammi mittefunktsionaalsed nõuded, mis ka käesoleva töö raames realiseeriti. Nõuded on toodud tabelis 3.

Tabel 3. Mittefunktsionaalsed nõuded.

Identifikaator	**Nõue**
1	Pistikprogramm peab olema laiendatav teistele andmebaasisüsteemidele.
2	Pistikprogramm tuleb luua 64 bitisele Notepad++ versioonile.
3	Pistikprogramm kasutab MS Access baasiga suhtlemiseks MS Access andmebaasimootorit (ACE) läbi OLE DB draiveri.
4	Tuvastatud vead lausetes kuvatakse redaktoris viga sisaldavale reale järgneval real.
5	Peab saama kasutada .accdb ja .mdb formaadis faile.
6	Info probleemide kohta peaks olema kuvatud vastavalt redaktori keele valikule, kas eesti või inglise keeles
7	Uute keelte toe lisamine peaks olema võimalikult lihtne.

5.4 Andmebaasiga suhtlemiseks mõeldud draiveri valik

Andmebaasisüsteemiga ühendumiseks ja suhtlemiseks on vajalik draiver, mis ühendab klientrakendust andmebaasisüsteemiga. MS Accessi kasutamiseks on kaks võimalikku draiverit:

- ODBC draiver [66] ja
- OLEDB draiver. [67]

Omaduste poolest on OLEDB oluliselt suuremate võimalustega kui ODBC ja OLEDB võimaldab suhelda erinevate andmeallikatega ühetaolisel viisil. OLEDB on Microsofti-spetsiifiline spetsiaalselt Windowsile mõeldud draiver. Kuna OLEDB on universaalne, saab seda programmeerimise liidest kasutada kõikide teiste andmebaasisüsteemide jaoks, millel on olemas OLEDB spetsiifiline draiver. ODBC draiveri realisatsioon erineb erinevate andmebaasisüsteemide puhul.

5.5 Tagarakenduse realisatsioon

Selles jaotises kirjutatakse täpsemalt sellest, kuidas töötab tagarakendus.

5.5.1 Pistikprogrammile esitatud nõuete täitmisel tekkivad valikud

Täiesti uue pistikprogrammi arendamine olukorras, kus on olemas olemasolevad pistikprogrammid, mida on võimalik laiendada, on ebamõistlik. Põhiline osa pistikprogrammist koos kasutajaliidesega peaksid olema kasutatavad ka uue andmebaasisüsteemi lisamisel. Seetõttu tuli leida sobiv avatud lähtekoodiga pistikprogramm, mida edasi arendada.

Töö koostamise käigus uuriti Notepad++ olemasolevaid pistikprogramme, mida oleks võimalik edasi arendada ning laiendada andmebaasisüsteemile MS Access. Notepad++ projekti koodihoidlas on nimekiri ametlikest pistikprogrammidest, mille hulgas on pistikprogramm [68], mis võimaldab käivitada andmebaasisüsteemi MS SQL Serveri lauseid. Nimetatud pistikprogramm on arendatud programmeerimiskeeles Pascal. [69] Lähtekoodi uurides selgus, et tegemist ei ole modulaarselt arendatud programmiga, mis tähendab, et seda ei ole lihtne laiendada teistele andmebaasisüsteemidele.

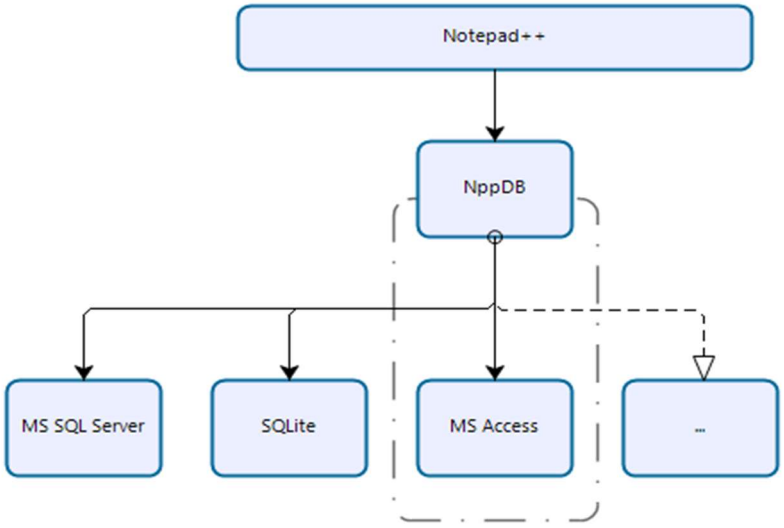
Mitteametlikest Notepad++ pistikprogrammidest õnnestus leida NppDB, mis oli arendatud käivitamaks SQL lauseid andmebaasisüsteemide MS SQL Server ja SQLite andmebaasides. Tegemist oli modulaarselt arendatud programmiga, mis on loodud kasutades programmeerimiskeelt C# .NET raamistikul. Samas oli programm aegunud ning arendatud 32 bitisele Notepad++ versioonile. 2023. aasta alguse seisuga oli viimane uuendus sellesse programmi tehtud 2014. aastal.

NppDB kasutajaliides meenutab erinevate andmebaasi haldusprogrammide nagu näiteks Oracle SQL Developer [70] kasutajaliidest, mis on loogiline ja kasutajasõbralik. See

lihtsustab ka arendatava pistikprogrammi kasutamist. Andmebaasiühenduste nimekiri ja andmebaasiobjektid kuvatakse kaustade struktuuris, mis võimaldab saada visuaalselt hea ülevaate andmebaasiobjektidest. Lisaks on NppDB arendatud programmeerimiskeeles C# kasutades .NET raamistikku, mis on käesoleva töö autori jaoks eelistatuim arenduskeel.

5.5.2 Notepad++ pistikprogrammi NppDB laiendamise üldine lahendus

Notepad++ pistikprogramm NppDB on modulaarne, mis tähendab, et andmebaasisüsteemide MS SQL Server ja SQLite ühenduste haldamise funktsionaalsus on arendatud moodulitena. Modulaarsus tagab, et pistikprogrammi on võimalik samalaadse moodulina lisada ka ühenduse loomise võimekus andmebaasisüsteemiga MS Access ning tulevikus lisada tugi veel teistegi andmebaasisüsteemide jaoks. Joonisel 32 on näidatud Notepad++ redaktori ja pistikprogrammi NppDB üldine skeem.



Joonis 32. Notepad++ ja NppDB üldine skeem.

Notepad++ kasutab Scintilla tekstiredaktori komponenti. Scintilla pakub erinevaid lisavõimalusi, näiteks süntaksi kujundamist (*styling*), vea indikaatoreid (*error indicators*), koodi lõpetamist (*code completion*), rea annotatsioone (*annotations*) ja vihje mulle (*call tips*). Kogu Scintilla funktsionaalsust saab kasutada läbi Windows API funktsiooni *SendMessage*. Nimetatud funktsiooni kasutatakse Windowsi akendele

rakenduse-spetsiifiliste või Windowsi standardsõnumite (käskude) saatmiseks. Sellel viisil suhtleb NppDB ka redaktoriga.

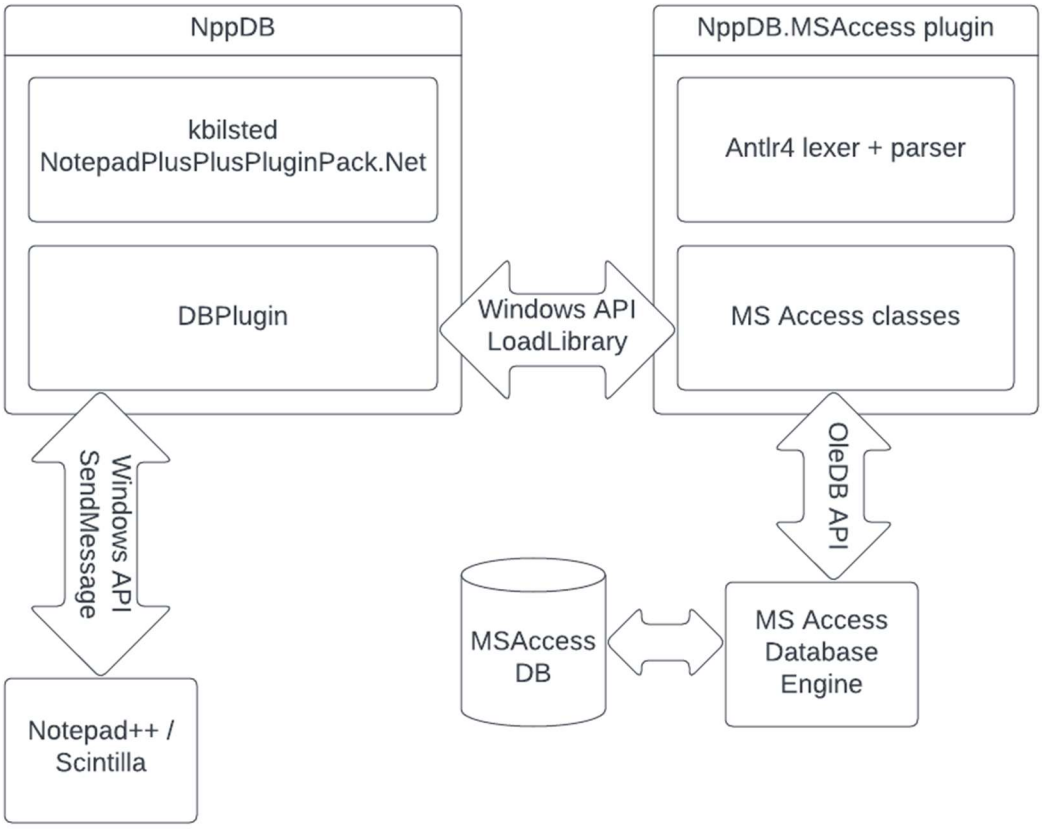
NppDB kasutab Notepad++ pistikprogrammi malli (*NotepadPlusPlusPlugin.Net*), mis on arendatud programmeerimiskeeles C#. Mallis on defineeritud ära kõik Scintilla spetsiifilised sõnumid (käsud) ja muu oluline baaskood pistikprogrammi arendamiseks.

Kuna NppDB on modulaarne, siis laetakse MS Accessi moodul pistikprogrammi selle käitusajal (*runtime*) Windows API funktsiooni *LoadLibrary* abil.

MS Accessi moodul koosneb andmebaasiga suhtlemiseks loodud klassidest (nt olulisemad *MSAccessConnect* ja *MSAccessExecutor*) ja parseri generaatori ANTLR4 [1] genereeritud klassidest (*MSAccessParser* ja *MSAccessLexer*). ANTLR4 tööpõhimõtet käsitletakse põhjalikumalt jaotistes 5.5.5 ja 5.5.6.

Suhtluseks andmebaasisüsteemiga kasutatakse OLEDB API poolt pakutavaid klasse ja nende meetodeid. OLEDB API suhtleb MS Accessi andmebaasisüsteemiga läbi selleks ettenähtud OLEDB-spetsiifilise andmebaasi draiveri.

Joonisel 33 on eeltoodu kujutatud visuaalselt.



Joonis 33. Pistikprogrammi tehniline struktuur.

5.5.3 Ettevalmistavad tööd

Pistikprogrammi NppDB arendati viimati 2014. aastal Notepad++ versioonile 6.5. Käesolevas töös esitletava arenduse alustamise hetkel oli väljas Notepad++ versioon 8.4.9. Seega esimese tööna tuli uuendada pistikprogrammi NppDB, et see töötaks tõrgeteta Notepad++ viimase versiooniga. Olulisem osa oli C#/.NET pistikprogrammi malli uuendamine ajakohase 64-bitise versiooniga. [71] Mall defineerib suhtluse pistikprogrammi ja Notepad++ vahel.

Seejärel loodi uus C#/.NET projekt (komponent) andmebaasisüsteemi MS Access andmebaasiga ühendumiseks ja käskude käivitamiseks. Järgmisena lisati projekti parseri generaator SQL lausete analüüsimiseks. Tuli paigaldada ka 64-bitine draiver MS Access andmebaasisüsteemiga suhtlemiseks.

5.5.4 MS Accessi andmebaasis koodi käivitamise funktsionaalsus

Pistikprogrammi abil peab saama töötada MS Accessi andmebaasifailidega, mille laiendid on .accdb ja .mdb. Tähtis on ka see, et kasutaja peab saama luua nii uut andmebaasi kui ka töötada olemasolevaga. Selle jaoks võeti pärast draiveri installeerimist kasutusele MS OLEDB API, mis võimaldab arendada rakendusi, mis pääsevad juurde erinevatele andmeallikatele, olenemata sellest, kas need on andmebaasisüsteemid või mitte.

Selleks, et redaktorist oleks võimalik käivitada korraga mitmeid SQL lauseid ning arvata käivitatavast lausest välja koodi kommentaarid (joonisel 34 on näide keskmiselt keerukamatest kommentaaridest), on vaja teha mitmeid muudatusi.

```
1  SELECT '''tekst'''; -- /*' & veergl
2  /* /*
3  mitmerealine kommentaar käsu keskel semikooloniga;
4  */
5  FROM tabell; -- üherealine kommentaar käsu keskel rea lõpus; SELECT 1;
6  SELECT * FROM /* --kommentaar käsu ja rea keskel */ tabel2;
```

Joonis 34. Süntaktiliselt korrektsed SQL lausete näidised.

Juhul kui saata laused redaktorist otse andmebaasimootorile täitmiseks, siis joonisel 34 toodud lausete käivitamisega tekiks järgmised probleemid.

- MS Accessi andmebaasimootor ei võimalda käivitada mõlemat lauset korraga.
- MS Accessi andmebaasimootor ei erista ega arva SQL lausest välja ühelgi real olevat kommentaari.

Ülaltoodud probleemide lahendamiseks oleks esmane idee tükeldada lauseid semikoolonite kohalt. See aga tekitaks järgmised probleemid.

- Semikoolonid võivad asuda tekstiliteraalide sees, mis tähendab, et selliseid semikooloneid ei tohiks lugeda lause lõpetajaks ning literaal tuleb lugeda lause osaks.
- Semikoolonid võivad asuda kommentaaride sees, mis tähendab, et selliseid semikooloneid ei tohiks lugeda lause lõpetajaks ja kommentaar tuleb SQL lausest välja arvata.

Seega on tarvis analüüsida, missugust lause süntaksi tuleb ignoreerida – kõiki semikooloneid ja sidekriipse ei saa SQL lausest niisama välja arvata. Väljatoodud probleemid oleksid ilmselt keskmise keerukusega algoritmiga lahendatavad. Samas on käesoleva töö üheks eesmärgiks tuvastada ka SQL lausete koostamisel tehtavaid vigu. Seega on vajalik lauseid süntaksielementide kaupa tükeldada, kontrollida, analüüsida ja seejärel need laused kokku panna ning suunata edasi draiveri vahendusel andmebaasisüsteemile täitmiseks. Seetõttu on mõistlik lahendada need kaks probleemi koos. Kõige paremaks lahenduseks nimetatud kahe probleemi korral on parseri generaatori kasutamine.

Parseri generaator (täpsemalt kirjeldatud punktides 5.5.5 ja 5.5.6) on tööriist, mis võtab sisendiks nt konkreetse programmeerimiskeele või päringukeele grammatika ja genereerib automaatselt lähtekoodi, mis saab grammatikat kasutades trükimärkide vooge parsida ehk sõeluda.

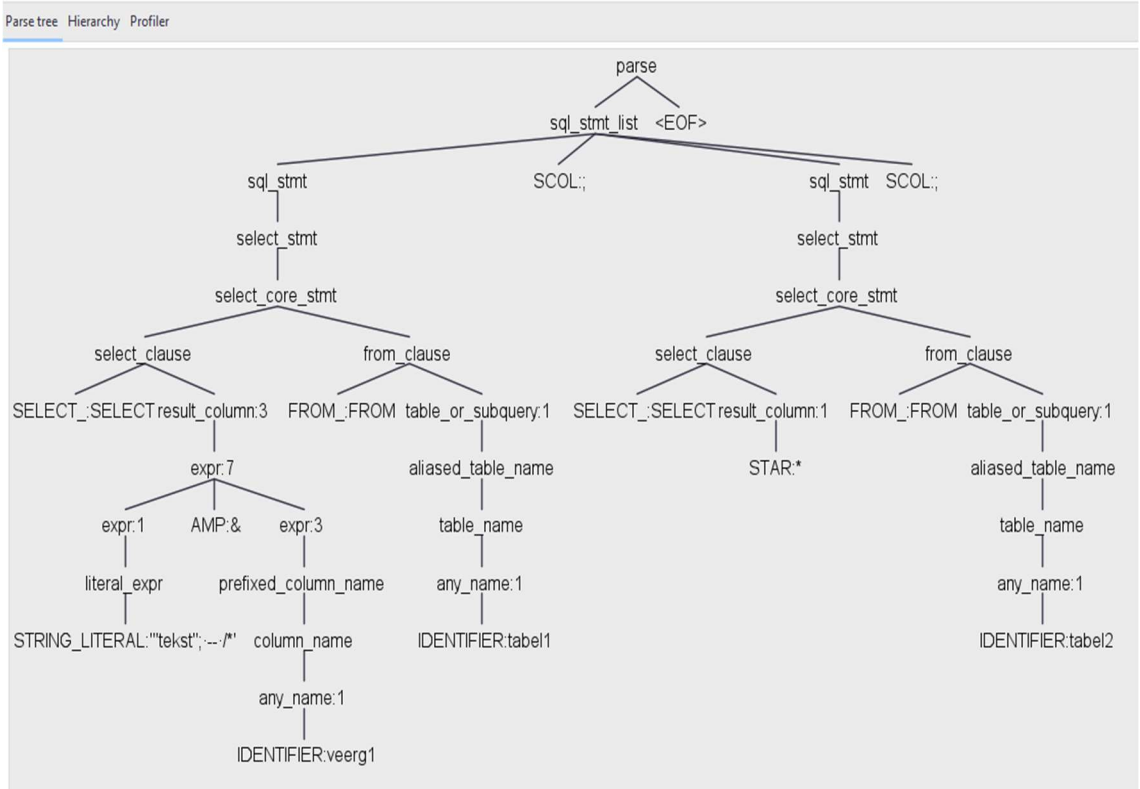
SQL lause käivitamine peab olema kasutaja jaoks mugav. See tähendab, et kasutaja peaks saama konkreetset lauset käivitada ka ilma, et ta valiks terve käivitatava lause. Lause käivitamiseks peaks piisama sellest, et tekstikursor paikneb lausel. Selline lahendus on tavapärane ka teistele SQL lausete käivitamist võimaldavatele tööriistadele. Sellise funktsionaalsuse realiseerimiseks on oluline, et kogu redaktori aknas olev tekst oleks korralikult läbi parsitud ehk sõelutud, et tuvastada, kus algab ja lõppeb lause, millel tekstikursor parajasti asetseb.

5.5.5 Parseri generaatori töö põhimõte

Parseri generaator võtab sisendiks näiteks konkreetse programmeerimiskeele või päringukeele grammatika ja genereerib automaatselt lähtekoodi, mis saab grammatikat kasutades trükimärkide jada parsida. Loodud kood on parser, mis võtab trükimärkide jada ja proovib seda jada grammatikaga sobitada. Parser loob konkreetse süntaksipuu, mis näitab, kuidas trükimärkide jada jaotatakse vastavalt grammatikareeglite puule süntaksi ühikuteks. Süntaksipuu juur on grammatika esimene mitteterminal. Süntaksipuu iga sõlm laieneb üheks või mitmeks mitteterminaliks (grammatika parsimise reegliks) ja/või terminaliks (sõnastiku reegliks). [72]

Süntaksipuu iga sõlmega seondub kindel informatsioon: sh sõlme tüüp, sõlme tekstiline väärtus ja selle asukoht sisendtekstis. Rekursiivse meetodiga puu analüüsimise käigus

kogutakse ärireeglitele vastavat informatsiooni (nt vigu ja hoiatusi) ja nende asukohti. Nimetatud asukohad aitavad redaktoris esialgses tekstis vigu ja hoiatusi visuaalselt märgistada.



Joonis 35. Parseri generaatori moodustatud süntaksi puu näide.

Joonisel 35 on näha, kuidas parseri generaator moodustas SQL lausetest käivitamisel puustruktuuri. Võtmesõnad, tabeli nimed, veeru nimed, tekstiliteraalid, operaatorid jms on süntaksipuu terminalideks ja nende reeglipärased kooslused süntaksipuu mitteterminalideks.

5.5.6 Parseri generaatori valik, põhjendus ja kasutuselevõtt

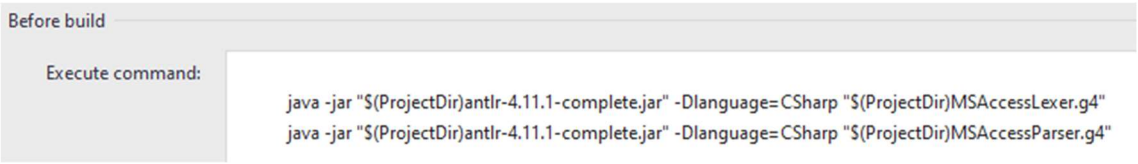
Parseri generaatoreid on arendatud mitmeid. Käesolevas töös said parseri generaatori valimisel määravaks järgmised kriteeriumid.

- Genereeritud parseri koodi keel. Kuna NppDB ja käesoleva töö raames arendatud tarkvara on valminud programmeerimiskeeles C#, siis pidi ka valitud parseri generaator genereerima SQL parseri koodi C# programmeerimiskeeles.

- Toimivate grammatikareeglite saadavus mõnele SQL andmebaasisüsteemile, nt SQLite grammatikafaili olemasolu.
- Põhjaliku dokumentatsiooni olemasolu.
- Suure kasutajaskonna olemasolu, et leida lisainformatsiooni arenduse käigus kerkivate probleemide lahendamiseks.
- Soovitavalt JetBrains Riderile arendatud pistikprogramm. See muudaks arenduse efektiivsemaks: genereeriks puu joonise ja märgistaks vigu grammatikafailides, mis soodustaks kiiremat vigade leidmist jms.

Nendele tingimustele vastas ANTLR4. [1]

Genereeritav parser lisati käesoleva töö raames arendatud pistikprogrammi MS Accessi mooduli osaks tulenevalt sellest, et grammatikareeglid on andmebaasisüsteemi MS Access põhised. Grammatikareeglid on kirjeldatud kahes erinevas failis – ühes on parseri reeglid (*MSAccessParser.g4*) ja teises sõnastiku reeglid (*MSAccessLexer.g4*). Need failid lähevad sisendiks käsurea käskudele (joonis 36), mis genereerivad vastavad programmeerimiskeele C# klasside failid (*MSAccessParser.cs* ja *MSAccessLexer.cs*).



Joonis 36. Kompileerimisahela seadistus parseri genereerimiseks.

Joonisel 37 on näidatud, kuidas väga efektiivselt moodustatakse parsitud süntaksipuu kasutades C# genereeritud klasse (*MSAccessParser* ja *MSAccessLexer*). Joonisel 37 toodud meetod „parse()“ moodustab süntaksipuu juure ehk esimese reegli.

```
var input = CharStreams.fromString(sql);
var lexer = new MSAccessLexer(input);
var tokens = new CommonTokenStream(lexer);
var parser = new MSAccessParser(tokens);
var tree = parser.parse();
```

Joonis 37. Lihtsustatud näide koodist, mis kasutab genereeritud klasse süntaksipuu loomiseks.

Kuna MS Accessi grammatikareeglite faili saadaval ei olnud, siis tuli see koostada käesoleva töö käigus. Selle põhjaks võeti andmebaasisüsteemi SQLite vastav reeglifail [73] ning kohandati andmebaasisüsteemile MS Access. MS Accessis kasutatav SQL on vähemate võimalustega, mille tõttu tuli esialgset grammatikafaili väga suures osas ümber teha vastavalt MS Accessi dokumentatsioonile ja muudele asjakohastele allikatele.

5.5.7 SQL lausete kontrollimise funktsionaalsus

Redaktorisse sisestatud SQL lausete parsimisel moodustatakse parsitavast sisust abstraktne süntaksipuu. Seejärel käib algoritm moodustatud puu ühe korra läbi, et kontrollida sisendtekstis esinenud kindlaid reegleid, mille ülesehitust analüüsitakse SQL lausete kirjutamisel tehtavate sagedaste vigade leidmiseks.

Vigade kontroll on realiseeritud C# failis MSAccessAnalyzer.cs. Joonisel 38 on toodud nimetatud failist näidis vea kontrollimise kohta. Antud näidis kontrollib viga, kus DISTINCT võtmesõna kasutatakse koos GROUP BY klausliga. Kokkuvõttes tähendab see kahekordset instruktsiooni päringu tulemusest korduste eemaldamiseks.

```
private static void _AnalyzeRuleContext(RuleContext context, ParsedCommand command)  1 usage
{
    switch (context.RuleIndex)
    {
        case MSAccessParser.RULE_select_into_stmt:
        {
            if (context is MSAccessParser.Select_into_stmtContext ctx)
            {
                if (ctx.selectClause?.distinct != null && ctx.groupByClause != null)
                    command.Warnings.Add(CreateWarning(ctx, ParserMessageType.DISTINCT_KEYWORD_WITH_GROUP_BY_CLAUSE));
```

Joonis 38. Vea kontrolli näidis.

Joonisel 39 on näide reeglite definitsioonist, mida kasutatakse joonisel 38 toodud koodis kontrollimaks SELECT … INTO lause SELECT klauslis DISTINCT võtmesõna olemasolu ja selles lauses ka GROUP BY klausli olemasolu. Selle tingimuse täidetusel luuakse parsitud käsu kohta hoiatusobjekt, mida kasutatakse hiljem kasutajale redaktoris kuvamisel. Analoogne kontroll on tehtud ka SELECT klausli kohta.

```
select_clause:
    SELECT_ distinct=(DISTINCT_ | DISTINCTROW_ | ALL_)?
    (TOP_ limit=NUMERIC_LITERAL)?
    resultColumns+=result_column (COMMA resultColumns+=result_column)*
;

select_into_stmt:
    selectClause=select_clause INTO_ tableName=table_name
    (
        fromClause=from_clause
        joinClause+=join_clause*
        whereClause=where_clause?
        groupByClause=group_by_clause?
        orderByClause=order_by_clause?
    )?
;
```

Joonis 39. Reeglite definitsioonide näidis.

Vigade kuvamisel kasutatakse pistikprogrammi tõlkefaili. Iga inimkeele jaoks, milles infot vigade kohta kuvatakse, peab olema oma tõlkefail. Pistikprogramm kasutab automaatselt tõlkefaili vastavalt Notepad++ keeleseadistusele. See tähendab, et kui kasutaja kasutab eesti keelse menüüga Notepad++, siis võetakse veahoiatus automaatselt vastavast keelefailist, näiteks „estonian.ini“. Tõlkefaili nimi sõltub Notepad++ enda keelefaili nimest, mis asub kaustas „localization“ ja kus asub eelnevale näitele vastavalt „estonian.xml“. Käesoleva töö tulemusena loodi eestikeelne ja ingliskeelne tõlkefail.

Arendatud kontrollid koos selgitustega on toodud jaotises 5.2.

5.5.8 Arenduse käigus ette tulnud takistused

Käesoleva töö käigus valminud pistikprogrammi arendamisel tekkisid ka erinevad takistused, mis tuli ületada.

Arenduse käigus ilmnes, et kui arvutisse on paigaldatud 32-bitine MS Office rakendus Access ja soovitakse kasutada käesoleva töö raames arendatud pistikprogrammi koos 64-bitise draiveriga, siis ei ole võimalik sama aasta 32-bitist ja 64-bitist draiveri versiooni arvutisse paigaldada. Sel juhul tekib mõlema rakenduse kasutamisel konfiguratsiooniviga ja rakendust kasutada ei saa. Seetõttu tuleb kasutada erineva aastakäigu draivereid,

kusjuures pistikprogrammiga töötamiseks tuleb kasutada vanemat draiverit, kuna ajakohane MS Access vajab töötamiseks uuemat 32-bitist draiverit.

MS Office Accessis on VBA laiendus, kus on kasutusel erinevad funktsioonid (nt „nz“), mis on OLEDB draiveri jaoks defineerimata. Nendel juhtudel on tarvis funktsioonid pistikprogrammi koodis realiseerida (nt funktsioon „nz“ realiseeriti kasutades funktsioone „IIf“ ja „IsNull“).

Erinevates kohtades tuleb rõhku panna teksti kodeeringule. Näiteks kui redaktorisse sisestatud tekst kodeeritakse kodeeringuga, mis kasutab kahe baidi suuruseid sümboleid, siis dekodeerimisel vale kodeeringu kasutamisel võib draiver neid käsitleda kahe sümbolina, mis kumbki on suurusega üks bait. Redaktorist loetud tekst tuleb edastada draiverile nõutud kodeeringus.

Tabelite loomisel ja muutmisel ei saanud defineerida vaikeväärtuseid kasutades temporaalseid funktsioone koos sulgudega (nt Now()). Selleks, et draiver käsku aktsepteeriks, tuli sulud koodis eemaldada.

5.6 Kasutajaliides

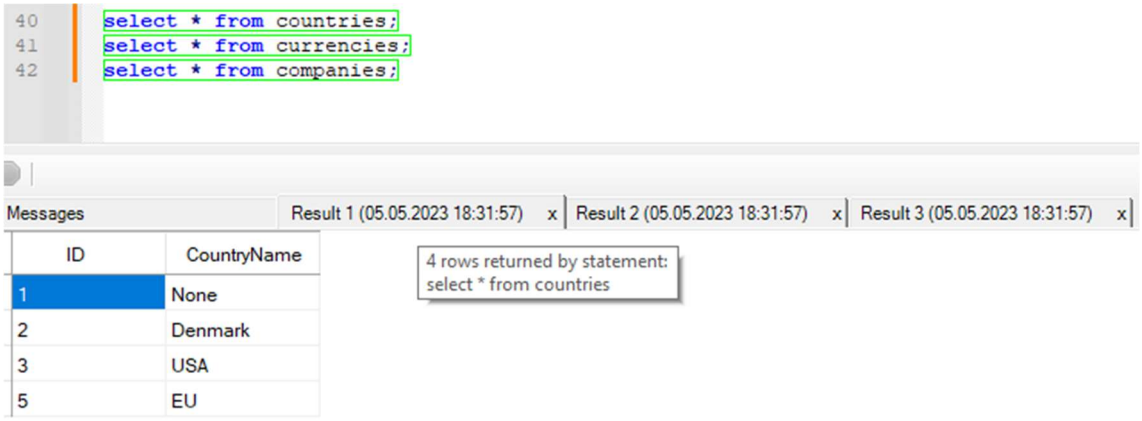
Käesoleva töö käigus valminud arendus kasutab Notepad++ ja selle pistikprogrammi NppDB kasutajaliidest. Pistikprogrammi kasutajaliides on arendatud kasutades programmeerimiskeelt C# ja .NET raamistikku. Samas täiendati kasutajaliidest järgmiselt:

- Andmebaasi ühenduse loomisel kuvatakse järgmisi valikuid: „Create an empty database“ teeb uue tühja MS Access faili ja „Open an existing database“ avab olemasoleva andmebaasi. Kui teha valik „Create an empty database“, kuid avanevas aknas valida mõni olemasolev andmebaasi fail, siis pistikprogramm kirjutab valitud andmebaasi üle tühja andmebaasiga. Kui sisestada aknas uue andmebaasi nimi, siis pistikprogramm loob uue tühja andmebaasi. Enne käesoleva töö raames tehtud muudatusi ei teinud valik „New“ uut andmebaasi (vt lisa 6 kuvatõmmis 1).

- Ühenduse loomisel andmebaasiga luuakse automaatne seos Notepad++ aktiivse failiakna ja ühenduse vahel. Enne käesoleva töö raames tehtud arendusi ei olnud

võimalik olemasolevale Notepad++'is avatud failile andmebaasi ühendada, vaid see tekitas alati uue faili.

- Iga käivitatud SELECT lause tulemus kuvatakse eraldi sakina. Aktiivseks muutub sakk, kuhu kuvatakse viimati käivitatud SELECT lause tulemus. Enne käesoleva töö raames tehtud muudatusi oli võimalik kasutada vaid üht logi sakki ja üht tulemuse sakki. Kuna varem sai käivitada vaid üht lauset korraga, siis piisas ka ühest tulemuste sakist (vt joonis 40).



Joonis 40. Tulemuste sakid koos saki nimede ja kohtspikriga.

- Käivitatud SELECT lausete puhul kuvatakse lause ja tagastatud tulemuste ridade arv saki kohtspikrina (*tooltip*). Enne andmeid ei kuvatud (vt joonis 40).

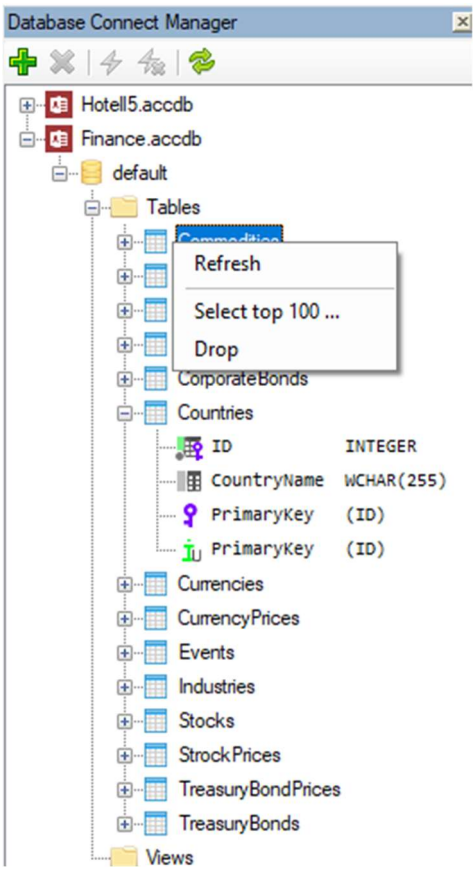
- Käivitatud SELECT lausete puhul kuvatakse käivitamise aeg saki nime järel. Enne käivitamise aega ei kuvatud (vt joonis 40).

- Käivitatud lausete tulemus kuvatakse logi sakis nimega „Messages". Õnnestumise korral kuvatakse saki lehel lause käivitamise aeg, õnnestumise teade ja lause ise. Ebaõnnestumise korral kuvatakse õnnestumise teate asemel ebaõnnestumise teade koos eeldatava põhjusega (vt lisa 6 kuvatõmmis 2).

- Kuna varem kirjutas pistikprogramm iga lause käivitamise järel logi lehe sisu üle, siis käesoleva töö raames muudeti loogikat nii, et iga uus logi kirje lisatakse logi sakis varasemate kirjete alla. Seetõttu loodi logi sakile kontekstmenüü kaudu logi puhastamise võimalus (vt lisa 6 kuvatõmmis 2).

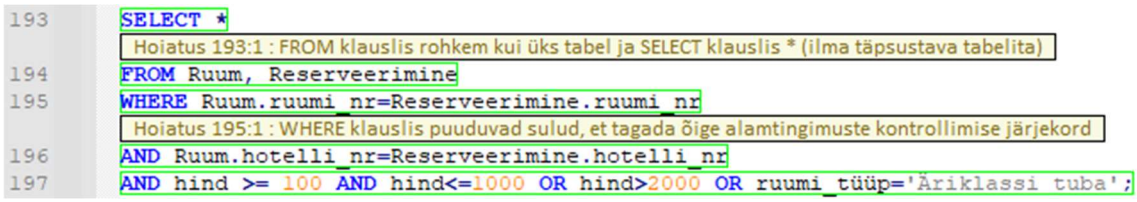
- Andmebaasi tabelite kuvamisel peideti süsteemiobjektid.

- Lisati võimalus tabeli kontekstmenüü kaudu täita andmebaasis järgmised laused (vt joonis 41):
    - `DROP TABLE /tabeli_nimi/` ja
    - `SELECT TOP 100 * FROM /tabeli_nimi/`.



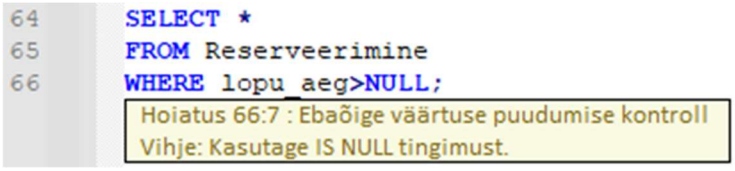
Joonis 41. Andmebaasiühenduse haldur.

- Iga tulemuse saki paremasse serva lisati sulgemise nupp „x“ sümboli näol, mille kaudu saab konkreetset sakki sulgeda. Lisaks lisati logi sakile kontekstmenüü kaudu võimalus sulgeda kõik tulemuse sakid. Sakkide ükshaaval sulgemisel muudetakse aktiivseks sellest eelmine sakk (vt joonis 40).
- Ühenduse sulgemisel suletakse automaatselt ka ühenduse aken.
- Täiendati andmebaasihalduri ühenduse kontekstmenüüd (vt lisa 6 kuvatõmmis 3).
- Arendati SQL lausete kontrollimisel tuvastatud vigade kuvamise lahendus. Vead kuvatakse vigase lause osa juures (vt joonis 42).



Joonis 42. SQL lause vea kuvamise näide.

- Veateadete tekste sisaldavas failis saab soovi korral lisada vea tekstile üldise vihje lause parandamise kohta. Kui panna jutumärkide sisse „\n“ ning sinna järgi tekst, siis kuvatakse vihje järgmisele reale. Joonisel 43 on näide veateatest koos vihjega lause parandamise kohta.



Joonis 43. SQL lause vea ja vihje kuvamise näide.

5.7 Testimine

Käesoleva töö raames valminud rakendust testiti pidevalt töö autori poolt arendamise käigus. Lisaks testis pistikprogrammi ka juhendaja.

Töö käigus ilmnesid erinevad probleemid, mis puudutavad draiveri ja MS Access töölauarakenduse erisusi, mille tõttu tuli käesolevas programmis teha lisaarendusi selleks, et MS Access’i töölauarakenduses lubatud, kuid draiveris mitte aktsepteeritud lauseid, saaks siiski käivitada. Täpsemalt antakse neist probleemidest ülevaade jaotises 5.5.8.

Käesoleva töö juhendaja testis pistikprogrammi erinevate SQL lausetega, mida tuli kontrollida seoses läbiviidava õppeaine õppetöö käigus kontrollimist vajanud SQL lausetega. Juhendaja saatis testimise käigus esile kerkinud mittekäivitunud SQL laused käesoleva töö autorile, kes analüüsis neid ning tegi vajalikud parandused pistikprogrammis. Testimisel tuvastatud puudused parandati.

5.8 Teadaolevad puudused

Käesolevas jaotises tuuakse välja nii teadaolevad kui ka oletuslikud puudused loodud pistikprogrammis.

5.8.1 MS Accessi andmebaasis koodi käivitamise funktsionaalsus

Notepad++'is SQL lausete käivitamisel analüüsib pistikprogramm süntaksi vastavust parseri grammatikareeglitele. Grammatikareeglid on välja töötatud vastava faili järgi, mis oli mõeldud andmebaasisüsteemile SQLite. Kuna MS Accessi dokumentatsioon on kesine ega sisalda keerulisemate lausete kohta grammatikareegleid, siis võib pistikprogrammi parseri grammatikareeglites esineda vigu. Võimalike vigade tõttu ei pruugi pistikprogramm mõningaid väga keerulisi lauseid käivitada. Kuna pistikprogrammi on testitud võrdlemisi palju, siis on tõenäosus selliste vigade avastamiseks väike. Kui vigu avastatakse, siis tuleb parseri grammatikafaili analüüsida ja vajadusel seal muudatusi teha.

Arendamise käigus lähtuti Notepad++ vaikimisi dokumendivormingust, milles kasutatakse UTF-8 kodeeringut, Windowsi stiilis reavahetusi (CR LF) ja tabulaatori pikkust neli tühikut. Teiste kodeeringute, reavahetuste ja tabulaatori pikkuste puhul võib esineda anomaaliaid, kuna neid ei ole läbi testitud.

Pärast andmebaasi tabelite struktuuris muudatuste tegemist (nt tabeli kustutamisel) ei uuenda arendatud programm automaatselt andmebaasiühenduse all tabelite loetelu kuva. Kuigi kasutajaliideses on olemas nupp, et käsitsi tabelite vaadet uuendada, on see siiski kasutajamugavuse seisukohast puudus.

5.8.2 SQL lausete kontrollimise funktsionaalsus

Arendatud tarkvaras on töö kirjutamise hetkel realiseeritud üksnes osad võimalikud SQL lausete kontrollid. Selleks, et pistikprogrammi saaks laialdasemalt kasutada SQL lausetes esinevate probleemide kontrollimiseks, võiks selliseid kontrolle olla rohkem.

Käesoleva töö raames ei arendatud andmebaasist sõltuvat vigade kontrolli. Näiteks võiks kontrollida veerunimede korrektsust või siis realiseerida funktsionaalsus, kus Notepad++'is lause kirjutamisel pakutakse veergude nimed andmebaasi põhjal automaatselt välja.

5.9 Installeerimine

Käesoleva töö raames arendatud tarkvara kasutamiseks peab arvutis olema järgmine tarkvara:

- Notepad++ 64-bitine versioon (testitud versioonidega 8.4.7 või 8.4.9).
- MS Access Database Engine 2010 Redistributable (draiver). Kui arvutisse on paigaldatud ka MS Office töölauarakendus Access, siis tuleb kaustas, kuhu draiver laetakse, administraatori õigustes avada Windowsi Command Prompt ning käivitada alla laetud installatsioonifail „quiet“ atribuudiga, näiteks: „accessdatabaseengine_X64.exe /quiet“. Vastasel juhul ei saa installeerida, kuna tekib konflikt.
- Notepad++ pistikprogramm NppDB. Käesoleva töö raames arendatud failid peavad paiknema järgmiselt.
    - Fail „NppDB.Comm.dll“ tuleb tõsta samasse kausta, kus asub käivitusfail „notepad++.exe“.
    - Ülejäänud kaks .dll faili ja vigade tõlkefail „estonian.ini“ tuleb tõsta kausta „…/plugins/NppDB“.

5.10 Litsentseerimine ja avalikkusega jagamine

Vastavalt autoriõiguste seaduse [74] § 4 lõige 3 punktile 3 on ka arvutiprogrammid teosteks, millele tekib autoriõigus. Arvutiprogrammina on selgelt käsitletav ka käesoleva töö raames arendatud tarkvara. Vastavalt sama seaduse § 46 lõikele 1 lubatakse teiste isikute poolt teose kasutamist autori antud litsentsi alusel. Seega on tarvis käesoleva töö käigus arendatavale tarkvarale määrata asjakohane litsents.

Litsentsid jagunevad suures plaanis kaheks – tasuta litsentsid ja tasulised litsentsid. Viimase puhul peavad kasutajad maksma tarkvara kasutamise eest tasu, kui esimese puhul on tarkvara kasutamine enamasti tasuta. Litsentse saab liigitada ka selle järgi kas ja kui suures ulatuses saab kasutaja uurida ning muuta tarkvara lähtekoodi ja muudatuse tulemust kasutada.

Käesolev tarkvara on tasuta ja avatud lähtekoodiga. Pistikprogramm NppDB on välja antud MIT litsentsi [75] all. Kuna käesoleva töö raames arendatav tarkvara on pistikprogrammi NppDB edasiarendus ning tarkvara on tarvis hiljem ka teiste poolt edasi arendada, siis on mõistlik määrata litsentsiks samuti MIT. MIT litsents annab igale isikule õiguse teha litsentsialuse materjaliga sisuliselt kõike, mida ta soovib.

Koodihoidlaid on väga palju ning autoril puudub konkreetne eelistus ühe või teise kasutamise suhtes. Käesoleva töö tulemus on avalikult kättesaadav GitHub'ist [76] aadressilt: https://github.com/pripost/NppDB.

6 Tulemuste valideerimine

Käesolevas peatükis antakse ülevaade magistritöö tulemuste valideerimisest. Tööd valideeriti nii käesoleva töö autori ja juhendaja poolt kui ka teiste sarnase funktsionaalsusega rakendustega võrdlemise kaudu.

Kuna programm valmis lõplikult suhteliselt hilja, siis ei jõudnud seda testida andmebaaside õppeaines SQLi harjutavad üliõpilased.

6.1 Koodi käivitamise funktsionaalsuse võrdlemine olemasolevate vahenditega

Käesolevas jaotises on toodud välja programmid, mis on suutelised töötama andmebaasisüsteemiga MS Access. Loetelu koostamiseks otsiti Internetist otsingumootori jaoks populaarsemaid MS Accessi tuge omavaid andmebaasi haldusprogramme, installeeriti need arvutisse ning testiti erinevate SQL lausetega nende suutlikkust ka reaalselt töötada käesoleva töö objektiks oleva andmebaasisüsteemiga. Tulemused on toodud järgnevas loetelus.

- RazorSQL (10.3.3, tasuline versioon) ei võimalda käivitada mitmeid lauseid, mis on süntaktiliselt korrektsed. Näiteks ei võimaldanud RazorSQL käivitada andmekirjelduskeele lauseid CREATE, ALTER, DROP ega ka kommentaaridega SELECT lauseid. Samas võimaldas programm käivitada SELECT, INSERT, UPDATE, DELETE ja SELECT … INTO lauseid.

- DBeaver Community Edition (23.0.3, tasuta versioon) ei võimalda käivitada andmekirjelduskeele lauseid CREATE, ALTER, DROP ja SELECT … INTO. DBeaver suudab täita SELECT, INSERT, UPDATE ja DELETE lauseid ning ka eristada SQL lauseid kommentaaridest. Vahend võimaldab MS Accessi andmebaasi käivitada ka selliseid SQL lauseid, mille süntaks ei vasta MS Accessi reeglitele. Need laused kasutavad SQL konstruktsioone, mida MS Accessi SQL dialekt ei toeta. Sellises võimekuses võib näha nii head kui halba. Hea on see, et saab kasutada suuremat hulka SQL võimalusi, kui MS Access ise toetab ning saab kasutada suuremat hulka teiste andmebaasisüsteemide jaoks kirjutatud koodi. Halb on see, et vahendis kirjutatud kood ei pruugi otse MS Accessis käivituda.

- DbSchema Community Edition (9.3.0, tasuta versioon) ei võimalda käivitada andmekirjelduskeele lauseid CREATE, ALTER, DROP ja SELECT … INTO. DbSchema suudab täita SELECT, INSERT, UPDATE ja DELETE lauseid ning eristada ka lauset kommentaaridest. Vahend võimaldab MS Accessi andmebaasi käivitada ka selliseid SQL lauseid, mille süntaks ei vasta MS Accessi reeglitele. Need laused kasutavad SQL konstruktsioone, mida MS Accessi SQL dialekt ei toeta.

- Visual Studio Community 2022 (17.0.0, tasuta versioon) ei võimalda käivitada andmekirjelduskeele lauseid CREATE, ALTER, DROP. Visual Studio suudab täita SELECT, INSERT, UPDATE, DELETE ja SELECT … INTO lauseid ning suudab eristada ka SQL lauset kommentaaridest.

- MDB Admin (2.5.7, tasuta versioon) võimaldab lisaks andmekäitluskeele lausetele käivitada andmekirjelduskeele lauseid CREATE, ALTER, DROP ja SELECT … INTO. Samas ei võimalda see käivitada mitut lauset korraga ega võimalda ka käivitada lauseid, milles on kommentaarid lause keskel või lõpus.

- MDB Viewer Plus (2.63, tasuta versioon) võimaldab lisaks andmekäitluskeele lausetele käivitada andmekirjelduskeele lauseid CREATE, ALTER, DROP ja SELECT … INTO ning suudab eristada käivitatavaid lauseid kommentaaridest. Samas kui käivitada mitu lauset korraga, siis see kuvab logi, et käivitas laused, kuid SELECT lause korral kuvab see siiski vaid ühe lause tulemused. Töötamisel tarkvaraga olid probleemiks nt tekstiväljade mitte väljakuvamine ja kodeeringuga seotud probleemid täpitähtede kuvamisel tabeli nimedes.

- DbVisualizer Pro (23.1, tasuline versioon) võimaldab käivitada CREATE, ALTER, DROP, SELECT, INSERT, UPDATE ja DELETE lauseid ning suudab eristada käivitatavaid lauseid kommentaaridest. Vahend võimaldab MS Accessi andmebaasi käivitada ka selliseid SQL lauseid, mille süntaks ei vasta MS Accessi reeglitele. Need laused kasutavad SQL konstruktsioone, mida MS Accessi SQL dialekt ei toeta. Samas ei võimalda see käivitada SELECT … INTO lauset ega kasutada süntaksi (nt erinevate andmetüüpide ja kantsulgude kasutus CREATE lausetes), mida MS Access andmebaasisüsteem võimaldab, kuna see kasutab oma spetsiifilist draiverit.

- FlySpeed SQL Query (4.5.1, tasuta versioon) võimaldab käivitada CREATE, ALTER, DROP, SELECT, INSERT, UPDATE, DELETE ja SELECT … INTO lauseid ning suudab eristada käivitatavaid lauseid kommentaaridest. Samas võimaldab see käivitada vaid esimest lauset, sõltumata sellest, mitu lauset valiti käivitamiseks.

Notepad++ võimaldab suurendada ja vähendada fonti, näitab ridade järjekorranumbreid, pakub koodi värvimist, sulgude tasakaalustatuse kontrolli ning pakub SQL koodi kirjutamisel välja võtmesõnu. Loodud pistikprogrammis saab SQL lauseid käivitada ühekaupa või skriptina, need laused võivad sisaldada kommentaare ja lausete käivitamiseks piisab kiirklahvile vajutamisest.

Kokkuvõtvalt saab öelda, et käesolevast pistikprogrammist paremat tööriista MS Accessi andmebaasiga töötamiseks ei õnnestunud autoril leida.

6.2 SQL lausete kontrollimise funktsionaalsuse võrdlus olemasolevate vahenditega

SQL vigade kontrollimise funktsionaalsuse valideerimiseks otsiti sarnase funktsionaalsusega tarkvara. Leiti järgmised vahendid: JetBrains IDE, SQLFluff ja QuerySandbox. [43]

Pistikprogrammi valideerimisel kasutati käesoleva töö juhendaja koostatud SQL lauseid, mis on toodud jaotises 4.2.

JetBrains IDE suutis seal esitatud lausete käivitamisel tuvastada järgmiseid vigu (sulgudes on esitatud JetBrains IDE antud tagasiside):

- Oodatakse loogikaavaldist (*Boolean expression is expected*).
- HAVING klausel ilma kokkuvõttefunktsioonita (*Using aggregate-free condition(s) in HAVING clause might be inefficient. Consider moving them to WHERE*).
- =NULL asemel tuleks kasutada IS NULL (*Suspicious comparison with NULL, probably IS NULL operator should be used*).

- <>NULL asemel tuleks kasutada IS NOT NULL (*Suspicious comparison with NULL, probably IS NOT NULL operator should be used*).

- WHERE või GROUP BY klauslis ei tohiks kasutada kokkuvõttefunktsiooni (*Aggregate calls are not allowed here (in WHERE / GROUP BY clause)*).

Kõiki neid välja toodud vigu suutis tuvastada ka käesoleva töö raames arendatud pistikprogramm.

SQLFluff suutis nende lausete käivitamisel tuvastada järgmiseid vigu.

- Kasutatakse SELECT * ehk ei ole tehtud valikut väljastatavate veergude osas (*Query produces an unknown number of result columns.*). Selle vea tuvastamisel on probleemiks see, et programm tõi välja kõik SELECT * kasutamise juhtumid välise SELECT lause puhul, kuid ei märkinud ära selle esinemist sisemises SELECT lauses, kus ka on tegelikult tegemist problemaatilise konstruktsiooniga.

- Mittevõrdsuse kontrollimisel operaatori „<>“ kasutamine operaatori „!=“ asemel (*Use '!=' instead of '<>' for "not equal to" comparisons.*). Selle vea tuvastamisel oli probleemiks see, et ta fikseeris vea lauses, kus oli kasutatud <>ANY või <>ALL, kuid see on täiesti korrektne. Samas ei fikseerinud ta viga mujal lausetes, kus oli kasutatud operaatorit „<>“. Samas on “<>” SQL standardis ette nähtud mittevõrdsuse kontrolli operaatori nimi. “!=” on mittevõrdsuse kontrolli operaatori nimi, mida saab kasutada mitmetes andmebaasisüsteemides, kuid mitte MS Accessis. Seega oleks “!=” kasutamine MS Accessis süntaksiviga ja MS Accessis SQL lauseid käivitav programm võiks seda kontrollida.

- Puuduvad veerunimed, kui andmeid on küsitud mitmest tabelist või valitud veerule ei ole antud prefiksiks tabeli nime või aliast (*Unqualified reference '*' found in select with more than one referenced table/view* ja *Unqualified reference 'nimetus' found in select with more than one referenced table/view.*). See kontroll töötas korrektselt. Teist probleemi kontrolli käesoleva töö raames arendatud programm ei kontrolli. SELECT nimetus FROM *Tabel1*, *Tabel2* … on probleem vaid siis, kui veerg *nimetus* on nii tabelis *Tabel1* kui ka tabelis *Tabel2*.

- SELECT klauslis avaldis (sh konstant või funktsiooni poole pöördumine), kuid tulemuseks olevale veerule pole ise antud aliast (*Column expression without alias.*

Use explicit `AS` clause.). See kontroll töötas sarnaselt käesoleva töö raames arendatud kontrolliga.

- GROUP BY ja DISTINCT samal lause tasemel (*Ambiguous use of 'DISTINCT' in a 'SELECT' statement with 'GROUP BY'.*). Programm tuvastas vea ühel juhul, kuid käesoleva töö raames arendatud pistikprogramm tuvastas vea kolmel korral.

- Tabelite ühendamisel RIGHT JOIN kasutamine LEFT JOIN asemel (*Use 'LEFT JOIN' instead of 'RIGHT JOIN'.*). Sellise vea esinemist ei toodud teadusartiklites välja ja ükski muu materjal ei nimeta seda veaks. See võib olla loogiline viga kui ühendamise tulemus ei ole selline nagu oodatud.

- UNION kasutamine UNION DISTINCT asemel (*'UNION [DISTINCT|ALL]' is preferred over just 'UNION'.*). Sellise vea esinemist ei toodud teadusartiklites välja ja ükski muu materjal ei nimeta seda veaks, seega ei saa seda veana käsitleda, pigem eelistuse küsimus. Selliselt kirjutatud lause võiks olla lugejale paremini arusaadav, sest tuletab meelde, et UNION [DISTINCT] operatsiooni tulemusest eemaldatakse kordused. MS Access UNION DISTINCT kasutamist ei toeta. Seega MS Accessi puhul oleks selle kasutamine süntaksiviga ja MS Accessis SQL lauseid käivitav programm võiks seda kontrollida.

- Puuduv alias (*Implicit/explicit aliasing of columns.*). Välimises ja sisemises lauses kasutatakse ühte ja sama nimetust ilma tabeli prefiksita. Testitud lause oli siiski korrektne, mistõttu seda veana käsitleda ei saa.

- Kahekordsete jutumärkide kasutamine (*Unnecessary quoted identifier "Viru".*). Testitud programm käsitles jutumärkides olevat teksti veeru nimena (identifikaatorina). Samas MS Accessis käsitletakse seda väärtust tekstilise literaalina. Teisalt on ka käesolevas töös loodud jutumärkide kasutamise kontroll, sest MS Accessi käitumine, lubades tekstilist väärtust jutumärkides, on ebastandardne ning jutumärkide selles kontekstis kasutama harjumisest oleks targem hoiduda. SQL standardi kohaselt tuleb tekstilised literaalid panna apostroofide vahele ning jutumärkidesse saab panna andmebaasiobjektide nimesid.

Allika [44] kohaselt kontrollib QuerySandbox SQL lausetes viite antimustrit, millest kahe esinemise kontroll on realiseeritud ka käesoleva töö tulemusena loodud vahendis.

- Kasutatakse SELECT * (*Avoid usage of wildcard selector*). QuerySandbox reageerib ka päringutele, mis on tehtud ühe tabeli põhjal, kuid loodud pistikprogramm reageerib SELECT * põhipäringus vaid siis, kui FROM klauslis on rohkem kui üks tabel. Põhjenduseks on, et kuigi andmebaasirakendustes pole SELECT * kasutamine otstarbekas, on see laialt levinud kirjapilt andmetega „mängimisel“ ja tulemuste avastamisel. Mõlemad programmid reageerivad SELECT * kasutamisele alampäringus, kusjuures erinevalt pistikprogrammist teeb QuerySandbox seda ka EXISTS alampäringute korral. SELECT * kasutamine EXISTS alampäringus, võib kuid, ei pruugi mõjutada lause täitmise kiirust – sõltub andmebaasisüsteemist. QuerySandbox reageeris ka SELECT * kasutamisele UNION ja INSERT lausetes, mida teeb ka käesolev pistikprogramm.

- =NULL asemel tuleks kasutada IS NULL (*Incorrect NULL usage*).

- <>NULL asemel tuleks kasutada IS NOT NULL (*Incorrect NULL usage*).

6.3 Juhendaja poolne testimine

Lisaks eeltoodule testis pistikprogrammi funktsionaalsust programmist huvitatud isikuna ka käesoleva töö juhendaja. Juhendajal oli programmiga töötamise hetkel üle 100 andmebaaside õppeainet õppiva üliõpilase. Semestri jooksul pidid üliõpilased lahendama kümneid erinevaid ülesandeid, kusjuures igas nädala praktikumis olid erinevad ülesanded. Üliõpilased saatsid praktikumides juhendajale hindamiseks sadu erinevaid SQL lauseid, mida kontrolliti käesoleva töö objektiks oleva pistikprogrammiga. Juhendaja saatis aja jooksul töö autorile 59 lauset, mida tal ei õnnestunud käivitada. Käesoleva töö raames töötati need laused läbi ning tehti vajalikud arendused, mille järel need laused käivitusid. Lisas 8 on toodud näited mitte töötanud lausetest ning nende järel programmi lähtekoodis tehtud muudatustest.

Kuna üliõpilased esitasid SQL lauseid MS Teamsi vahendusel ja kopeerimise tulemusena tekkisid lausetesse mitteprinditavad sümbolid, siis oli juhendajal väga mugav neid sümboleid Notepad++ vahendis lausetest tuvastada, need sealt kustutada ja siis kohe

sealsamas lause käivitada. Juhendaja kommenteeris, et see muutis tööprotsessi ning tagasiside andmise oluliselt kiiremaks ja sujuvamaks kui oleks olnud selle vahendi mittekasutamisel.

Juhendaja valideeris ka pistikprogrammi SQL lausete kontrollimise funktsionaalsust. Ta testis programmi arvukate SQL lausetega ning saatis tagasi 65 lauset (osade vigade kontrolliks olid mitmed erinevad laused), kuhu oli teadlikult tekitatud probleemne koht, kuid mida pistikprogramm ei suutnud algselt tuvastada. Pistikprogrammi täiustamise järgselt leiti ka neis 65-s lauses pistikprogrammi poolt probleemsed kohad üles.

Seega hõlmas juhendaja poolne testimine mitut andmebaasi ja suurt lausete hulka.

7 Arendusvaade

Käesoleva töö raames arendatud tarkvara on toimiv ja võimaldab kasutajal käivitada Notepad++ tekstiredaktorisse kirjutatud SQL lauseid MS Accessi andmebaasis. Lisaks on arendatud välja loogika ja funktsionaalsus, mis puudutab SQL vigade (probleemide) kohta tagasiside saamist. Töö maht oli piiratud, mille tõttu valmis käesoleva töö raames planeeritud funktsionaalsus, kuid pistikprogrammi võiks edasi arendada, et kasutajatel oleks sellest veelgi enam kasu. Järgnevalt on välja toodud loetelu arendusvõimalustest.

- Pistikprogramm NppDB pandi tööle 64-bitise Notepad++ versiooniga. Samas käesoleva töö raames ei uuendatud andmebaasisüsteemide MS SQL Server ja SQLite mooduleid, mistõttu neid ei ole võimalik 64-bitise Notepad++ versiooniga töö kirjutamise hetkel kasutada.

- Realiseerida SQL lausete käivitamine ka teiste andmebaasisüsteemide andmebaasides nagu PostgreSQL ja MySQL.

- Realiseerida redaktorisse kirjutatud SQL lausete kontrollimine nende probleemide suhtes, mille esinemist praegu ei kontrollita ja teha seda ka teiste andmebaasisüsteemide korral kui MS Access. Käesoleva töö peatükis 4 on välja toodud erinevat liiki SQL lausete kirjutamisel tehtavaid vigu, mille hulgast paljusid on võimalik ka kontrollideks realiseerida.

- Arendada välja andmebaasist sõltuvate SQL lausete vigade kontroll.

- Pakkuda lisaks infole vea esinemise kohta välja ka parandatud lause.

- Pakkuda lisaks infole vea esinemise kohta välja ka detailne info vea, selle tagajärgede ning parandamise võimaluste kohta.

- Pärast andmebaasi tabelite struktuuris muudatuste tegemist (nt tabeli kustutamisel) ei uuenda loodud programm automaatselt tabelite loetelu kuva, kuigi võiks seda teha. Samas käsk täidetakse. Värskendamine toimub üksnes kasutaja sekkumise tulemusena. Tulevikus võiks kuva värskendamine toimuda automaatselt.

- Kui registreerida ühendus andmebaasiga, mille puhul parooli ei kasutata, siis võiks pistikprogramm andmebaasiühenduse järgmise loomise korral parooli mitte küsida.

- Arendada pistikprogrammile koodi lõpetamise funktsionaalsus (*code completion*).

- Kuvada andmebaasiühenduse puu vaates liitprimaarvõtmed, -välisvõtmed ja -indeksid grupeerituna, hõlmatud veergude nimed loendina sulgudes.

- Kuvada andmebaasi puu vaates salvestatud protseduurid.

- Peale MS SQL Serveri ja SQLite osa uuendamist võiks võtta ette pistikprogrammi registreerimise ametliku pistikprogrammina.

7.1 Täiendused, mida tuleks teha uue andmebaasisüsteemi toe lisamiseks

Kui tulevikus tahetakse pistikprogrammile lisada uue andmebaasisüsteemi nagu näiteks PostgreSQL tugi, siis tuleb teha järgnevat.

- Tuleb paigaldada arvutisse, kas tasuline OLEDB draiver [77] või tasuta ODBC draiver.

- Vastavalt valitud draiverile tuleb kasutada loodavas .NET projektis kas OLEDB API-t või ODBC API-t. OLEDB draiveri puhul saab loodavas projektis kasutada eeskujuna MS Accessi vastava projekti koodi. ODBC draiveri puhul tuleb loodavas projektis koodi rohkem muuta, kuna OLEDB ja ODBC klassid ja nende meetodid võivad olla erinevad.

- Tuleb kasutades *System.Windows.Forms.TreeNode* ülemklassi arendada andmebaasisüsteemi objektide spetsiifilised alamklassid, mille abil esitada ühenduste haldamise aknas andmebaasi puustruktuuri. Lisaks tuleb loodavates klassides realiseerida liidesed (näiteks *IRefreshable* ja *IMenuProvider*) ja nende meetodid (vastavalt *Refresh* ja *GetMenu* meetodid).

- ANTLR4 abil genereerida lexeri ja parseri jaoks klassid. Parseri puhul on minimaalselt oluline, et see oskaks eraldada lausetest kommentaarid ja mitme lause korral neid üksteisest eristada.

- Ühenduse loomiseks tuleb realiseerida kasutajaliides, et küsida kasutajalt serveri aadressi, porti, andmebaasi nime, kasutajanime ja parooli.

Selleks, et laiendatava pistikprogrammi puhul realiseerida ka SQL vigade kontroll uue andmebaasisüsteemi puhul, saab süntaksipuu analüüsimise lähtekoodi osaliselt kopeerida käesoleva programmi lähtekoodist. Taaskasutamine ei ole üks-ühele võimalik, kuna iga andmebaasisüsteemi SQL lausete parseri genereerimiseks kasutatav grammatikafail võib olla erinev ja selle tulemusena ka genereeritav parseri klassi lähtekood on erinev. Süntaksipuu analüüsimisel sõltutakse nimetatud parseri klassist.

8 Kokkuvõte

Käesoleva töö eesmärgiks oli arendada teksti- ja lähtekoodiredaktorile Notepad++ pistikprogramm, millel on kaks ülesannet. Esiteks peab pistikprogrammi abil olema võimalik redaktoris käivitada SQL lauseid (sh andmekirjelduskeele lauseid – CREATE, ALTER, DROP ja SELECT … INTO lauseid) töölaua andmebaasisüsteemi MS Access andmebaasis. Teiseks peab see andma kasutajale tagasisidet käivitatavate SQL lausetes esinevate probleemide kohta. Need probleemid võivad olla nii koodi käivitumist takistavad vead kui ka koodi arusaadavust vähendavad halvad lõhnad. Need probleemid võivad olla universaalsed (esineda paljude erinevate SQL andmebaasisüsteemide kasutamise korral) või spetsiifilised MS Accessile. Pistikprogramm tuli kirjutada viisil, mis muudaks võimalikult lihtsaks hilisema teiste andmebaasisüsteemide toe lisamise, uute probleemide esinemise kontrollimise lisamise ja probleemidest raporteerimise keele lisamise.

Esmalt tuli välja selgitada, kuidas püstitatud eesmärki saavutada. Selleks otsiti programme, mis võimaldavad MS Accessi andmebaasis SQL lauseid käivitada ja SQL koodi kontrollida, et leida sealt käesoleva programmi jaoks eeskuju ning hiljem loodud programmi nendega võrrelda. Töö koostamise käigus uuris autor Notepad++ olemasolevaid pistikprogramme, mida oleks võimalik edasi arendada ning laiendada andmebaasisüsteemile MS Access. Autor leidis Notepad++ pistikprogrammi NppDB, mis oli arendatud, et võimaldada käivitada SQL lauseid andmebaasisüsteemide MS SQL Server ja SQLite andmebaasides. Tegemist oli modulaarsena arendatud tarkvaraga ning seda sai käesoleva töö raames laiendada eespool nimetatud funktsionaalsustega.

Probleemsete SQL lausete osas uuriti varem läbiviidud uuringuid, et koguda kokku info võimalikest vigadest, mida SQL lausete koostamisel tehakse. Pistikprogrammi poolt kontrollitavad probleemid leiti koostöös juhendajaga, kuna käesoleva töö maht ei võimalda kaugeltki realiseerida kõikide teadusartiklites väljatoodud probleemide kontrolli. Seejärel kirjeldati tarkvara funktsionaalsed nõuded kasutuslugudena.

Töö tulemusena valmis pistikprogrammi NppDB edasiarendus, kuhu lisati SQL lausete MS Access andmebaasis käivitamise funktsionaalsus ning arendati juurde ka võimekus SQL lausetest probleeme otsida. Pistikprogrammi laiendamisel kasutati programmeerimiskeelt C#, andmebaasisüsteemiga suhtlemiseks OLEDB draiverit ja

parseri generaatorina ANTLR4. Täiendatud pistikprogrammi abil saab käivitada kõiki MS Accessi SQL lauseid, saab käivitada mitut lauset korraga (st käivitada skripti) ning on võimalik kontrollida töö käigus välja valitud probleemide puudumist SQL lausetes.

Töö käigus loodud tarkvara testiti nii autori enda kui ka juhendaja poolt ja valideeriti teiste sarnaste programmide vastu. Saadud tagasiside põhjal hinnati programmi vastavust püstitatud nõuetele ja tehti parandusi. Võrreldes teiste programmidega, mis võimaldavad käivitada MS Access andmebaasis lauseid (sh MS Access ise) või mis võimaldavad kontrollida SQL lauseid probleemide esinemise suhtes, on käesoleva magistritöö raames loodud programm võimekam.

Kindlasti ei ole käesoleva magistritöö raames ja mahus arendatud tarkvara lõplik ning valmis. Pistikprogrammi on võimalik laiendada teistes andmebaasisüsteemides lausete käivitamise võimekusega (nt PostgreSQL, MySQL) ja lisada juurde lausetest täiendavate probleemide otsimist.

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

Kasutatud kirjandus

[1] A. /. T. Parr, „ANTLR,“ [Võrgumaterjal]. Available: https://www.antlr.org/. [Kasutatud 19 04 2023].

[2] salesforce, „What is an API?,“ [Võrgumaterjal]. Available: https://www.mulesoft.com/resources/api/what-is-an-api. [Kasutatud 28 04 2023].

[3] S. Jung, „nppDB,“ [Võrgumaterjal]. Available: https://github.com/gutkyu/NppDB. [Kasutatud 25 10 2022].

[4] Progress, „What is an ODBC Driver?,“ [Võrgumaterjal]. Available: https://www.progress.com/faqs/datadirect-odbc-faqs/what-is-an-odbc-driver. [Kasutatud 28 04 2023].

[5] SYBASE, „What is OLE DB?,“ [Võrgumaterjal]. Available: https://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc37776.1252/html/connpb/conndb_oledb_whatis.htm. [Kasutatud 28 04 2023].

[6] E. Entsüklopeedia, „Mittetulundusühing Entsüklopeedia,“ [Võrgumaterjal]. Available: http://entsyklopeedia.ee/artikkel/pistikprogramm. [Kasutatud 27 04 2023].

[7] „Scintilla,“ [Võrgumaterjal]. Available: https://www.scintilla.org/. [Kasutatud 15 04 2023].

[8] W3Schools, „SQL Tutorial,“ [Võrgumaterjal]. Available: https://www.w3schools.com/sql/. [Kasutatud 29 04 2023].

[9] Wikipedia, „Visual Basic for Applications,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Visual_Basic_for_Applications. [Kasutatud 07 05 2023].

[10] Microsoft, „Microsoft Access,“ [Võrgumaterjal]. Available: https://www.microsoft.com/en-us/microsoft-365/access. [Kasutatud 29 11 2022].

[11] LibreOffice, „LibreOffice Base,“ [Võrgumaterjal]. Available: https://www.libreoffice.org/discover/base/. [Kasutatud 30 11 2022].

[12] The PostgreSQL Global Development Group, „PostgreSQL/psql,“ [Võrgumaterjal]. Available: https://www.postgresql.org/docs/current/app-psql.html. [Kasutatud 30 11 2022].

[13] The PostgreSQL Global Development Group, „PostgreSQL,“ [Võrgumaterjal]. Available: https://www.postgresql.org/. [Kasutatud 30 11 2022].

[14] Oracle, „Oracle SQLcl,“ [Võrgumaterjal]. Available: https://www.oracle.com/database/sqldeveloper/technologies/sqlcl/. [Kasutatud 30 11 2022].

[15] Oracle, „Oracle Database,“ [Võrgumaterjal]. Available: https://www.oracle.com/database/. [Kasutatud 30 11 2022].

[16] Oracle, „SQL Developer,“ [Võrgumaterjal]. Available: https://www.oracle.com/database/sqldeveloper/. [Kasutatud 30 11 2022].

[17] InterviewBit, „Top 10 SQL IDEs To Know,“ 20 06 2022. [Võrgumaterjal]. Available: https://www.interviewbit.com/blog/sql-ides/. [Kasutatud 28 11 2022].

[18] D. Ho, „What is Notepad++,“ [Võrgumaterjal]. Available: https://notepad-plus-plus.org/. [Kasutatud 26 11 2022].

[19] N. Hodgson, „SciTe,“ [Võrgumaterjal]. Available: https://www.scintilla.org/SciTE.html. [Kasutatud 30 11 2022].

[20] M. Entsüklopeedia, „Eesti Entsüklopeedia,“ [Võrgumaterjal]. Available: http://entsyklopeedia.ee/artikkel/pistikprogramm. [Kasutatud 26 11 2022].

[21] „Best text editors of 2022,“ techRadar, 2022. [Võrgumaterjal]. Available: https://www.techradar.com/best/best-text-editors. [Kasutatud 24 10 2022].

[22] Microsoft, „SQL Server,“ [Võrgumaterjal]. Available: https://www.microsoft.com/en-us/sql-server/. [Kasutatud 30 11 2022].

[23] SQLite Consortium, „SQLite,“ [Võrgumaterjal]. Available: https://www.sqlite.org/index.html. [Kasutatud 30 11 2022].

[24] TIOBE, „TIOBE Index for October 2022,“ 2022. [Võrgumaterjal]. Available: https://www.tiobe.com/tiobe-index/. [Kasutatud 25 10 2022].

[25] S. Cass, „Top Programming Languages 2022,“ IEEE Spectrum, 23 08 2022. [Võrgumaterjal]. Available: https://spectrum.ieee.org/top-programming-languages-2022. [Kasutatud 27 04 2023].

[26] „DB-Engines Ranking,“ solidIT consulting & software development gmbh, [Võrgumaterjal]. Available: https://db-engines.com/en/ranking. [Kasutatud 24 10 2022].

[27] solid IT, „DB-Engines Ranking - Trend of Microsoft Access Popularity,“ [Võrgumaterjal]. Available: https://db-engines.com/en/ranking_trend/system/Microsoft+Access. [Kasutatud 30 11 2022].

[28] M.M.Zloof, „Query by example: a database language,“ *IBM Systems Journal,* kd. 16, nr 4, 1977.

[29] PopSQL Inc., „Collaborative SQL editor for your team,“ [Võrgumaterjal]. Available: https://popsql.com/. [Kasutatud 29 11 2022].

[30] A. Becker, „HeidiSQL,“ [Võrgumaterjal]. Available: https://www.heidisql.com/. [Kasutatud 29 11 2022].

[31] D. Community, „DBeaver,“ [Võrgumaterjal]. Available: https://dbeaver.io/. [Kasutatud 29 11 2022].

[32] Wise Coders GmbH, „DbSchema,“ [Võrgumaterjal]. Available: https://dbschema.com/. [Kasutatud 29 11 2022].

[33] A. Ahadi, J. Prior, V. Behbood ja R. Lister, „Students’ Semantic Mistakes in Writing Seven Different Types of SQL Queries,“ *ACM Conference on Innovation and Technology in Computer Science Education,* p. 6, 2016.

[34] S. Brass ja C. Goldberg, „Semantic Errors in SQL Queries: A Quite Complete List,“ Martin-Luther-Universität Halle-Wittenberg, Halle, 2005.

[35] T. Taipalus, M. Siponen ja T. Vartiainen, „Errors and Complications in SQL Query Formulation,“ *ACM Transactions on Computing Education,* kd. 18, nr 3, p. 29, 08 2018.

[36] D. Miedema ja E. A. George Fletcher, „Expert Perspectives on Student Errors in SQL,“ *ACM Transactions on Computing Education,* kd. 23, nr 1, pp. 1-28, 2022.

[37] D. Miedema, G. Fletcher ja E. Aivaloglou, „Identifying SQL misconceptions of Novices: Findings from a Think-Aloud Study,“ *acm Inroads,* kd. 13, nr 1, pp. 52-65, 2022.

[38] D. Miedema, G. Fletcher ja E. Aivaloglou, „So many brackets! An analysis of how SQL learners (mis)manage complexity during query formulation,“ %1 *ICPC ’22, May 16–17, 2022*, Virtual Event, 2022.

[39] S. Shao, Z. Qiu, X. Yu, W. Yang, G. Jin, T. Xie ja X. Wu, „Database-Access Performance Antipatterns in Database-Backed Web Applications,“ %1 *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Adelaide, 2020.

[40] T. Taipalus ja V. Seppänen, „SQL Education: A Systematic Mapping Study and Future Research Agenda,“ *ACM Transactions on Computing Education,* kd. 20, nr 3, p. 33, 2020.

[41] B. Karwin, SQL Antipatterns, Pragmatic Bookshelf, 2010.

[42] L. Mathon ja D. Miedema, „Increasing Awareness of SQL Anti-Patterns for Novices: A Study Design,“ *Proceedings of the 2022 ACM Conference on International Computing Education,* Kd-d. %1/%2Research-Volume 2, 2022.

[43] L. Mathon ja D. Miedema, „QuerySandbox,“ [Võrgumaterjal]. Available: https://querysandbox.com/. [Kasutatud 26 11 2022].

[44] L. Mathon ja D. Miedema, „Increasing awareness of SQL anti-patterns novices QuerySandbox,“ [Võrgumaterjal]. Available: https://querysandbox.com/icer22/poster.pdf. [Kasutatud 30 11 2022].

[45] GitHub Inc, „GitHub,“ [Võrgumaterjal]. Available: https://github.com/. [Kasutatud 30 11 2022].

[46] M. K. Sein, O. Henfridsson, S. Purao, M. Rossi ja R. Lindgren, „Action Design Research,“ *MIS Quarterly,* kd. 35, nr 1, pp. 37-56, 2011.

[47] Mountain Goat Software, „User Stories,“ [Võrgumaterjal]. Available: https://www.mountaingoatsoftware.com/agile/user-stories. [Kasutatud 29 11 2022].

[48] M. Männil, Mõnede SQL-andmebaasisüsteemide võimekusest SQLi keelelise liiasuse silumisel, Tallinn: Tallinna Tehnikaülikool, 2014.

[49] Microsoft, „What is the Language Server Protocol?,“ [Võrgumaterjal]. Available: https://microsoft.github.io/language-server-protocol/. [Kasutatud 28 11 2022].

[50] Microsoft, „C# documentation,“ [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/. [Kasutatud 17 04 2023].

[51] Microsoft, „Build it with .NET,“ [Võrgumaterjal]. Available: https://dotnet.microsoft.com/en-us/. [Kasutatud 17 04 2023].

[52] Oracle, „Download Java for Windows,“ [Võrgumaterjal]. Available: https://www.java.com/download/ie_manual.jsp. [Kasutatud 01 05 2023].

[53] L. Software, „The new way to collaborate,“ [Võrgumaterjal]. Available: https://www.lucid.co. [Kasutatud 07 05 2023].

[54] Bizagi, „Modern Apps,“ [Võrgumaterjal]. Available: https://bizagi.com/en. [Kasutatud 07 05 2023].

[55] D. Ho, „Notepad++ Resources,“ [Võrgumaterjal]. Available: https://notepad-plus-plus.org/resources/. [Kasutatud 15 04 2023].

[56] D. Ho, „notepad-plus-plus/nppPluginList,“ [Võrgumaterjal]. Available: https://github.com/notepad-plus-plus/nppPluginList. [Kasutatud 15 04 2023].

[57] R. Software, „RazorSQL,“ [Võrgumaterjal]. Available: https://razorsql.com/index.html. [Kasutatud 29 04 2023].

[58] Microsoft, „Visual Studio,“ [Võrgumaterjal]. Available: https://visualstudio.microsoft.com/. [Kasutatud 01 05 2023].

[59] D. S. AB, „The database client with the highest user satisfaction.,“ [Võrgumaterjal]. Available: https://www.dbvis.com/. [Kasutatud 05 05 2023].

[60] A. D. Software, „FlySpeed SQL Query,“ [Võrgumaterjal]. Available: https://www.activedbsoft.com/download-querytool.html. [Kasutatud 05 05 2023].

[61] M. Degasperi, „MDB Admin,“ [Võrgumaterjal]. Available: https://sourceforge.net/projects/mdbadmin/. [Kasutatud 05 05 2023].

[62] A. Nolan, „Portable database utility: MDB Viewer Plus,“ [Võrgumaterjal]. Available: http://www.alexnolan.net/software/mdb_viewer_plus.htm. [Kasutatud 05 05 2023].

[63] E. Eessaar, „SQL tüüpvead ja kommentaarid enne SQL kontrolltööd,“ Tallinn, 2022.

[64] S. Exchange, „Equals(=) vs. LIKE,“ [Võrgumaterjal]. Available: https://stackoverflow.com/questions/543580/equals-vs-like. [Kasutatud 07 05 2023].

[65] A. Cruickshank, „The SQL Linter for Humans,“ [Võrgumaterjal]. Available: https://docs.sqlfluff.com/en/stable/. [Kasutatud 05 05 2023].

[66] Microsoft, „Microsoft Open Database Connectivity (ODBC),“ [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/sql/odbc/microsoft-open-database-connectivity-odbc?view=sql-server-ver16. [Kasutatud 17 12 2023].

[67] Microsoft, „Microsoft Access Database Engine 2010 Redistributable,“ [Võrgumaterjal]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=13255. [Kasutatud 17 04 2023].

[68] V. Korobenkov, „npp.connections,“ [Võrgumaterjal]. Available: https://github.com/vladk1973/npp.connections. [Kasutatud 17 04 2023].

[69] F. P. team, „free pascal,“ [Võrgumaterjal]. Available: https://www.freepascal.org/. [Kasutatud 17 04 2023].

[70] Oracle, „SQL Developer,“ [Võrgumaterjal]. Available: https://www.oracle.com/database/sqldeveloper/. [Kasutatud 17 04 2023].

[71] K. B. Graversen, „NotepadPlusPlusPluginPack.Net,“ [Võrgumaterjal]. Available: https://github.com/kbilsted/NotepadPlusPlusPluginPack.Net. [Kasutatud 17 04 2023].

[72] M. EECS, „Reading 18: Parser Generators,“ [Võrgumaterjal]. Available: https://web.mit.edu/6.005/www/fa15/classes/18-parser-generators/#:~:text=A%20parser%20generator%20takes%20a,the%20sequence%20against%20the%20grammar.. [Kasutatud 18 04 2023].

[73] K. Domino, „antlr/grammars-v4,“ [Võrgumaterjal]. Available: https://github.com/antlr/grammars-v4/tree/master/sql/sqlite. [Kasutatud 19 04 2023].

[74] E. Vabariik, „Autoriõiguse seadus,“ [Võrgumaterjal]. Available: https://www.riigiteataja.ee/akt/129062022016. [Kasutatud 18 04 2023].

[75] O. S. Initiative, „The MIT License,“ [Võrgumaterjal]. Available: https://opensource.org/license/mit/. [Kasutatud 18 04 2023].

[76] I. GitHub, „GitHub,“ [Võrgumaterjal]. Available: https://github.com/. [Kasutatud 18 04 2023].

[77] Intellisoft, „PGNP OLEDB Providers for PostgreSQL*, Greenplum and Redshift,“ [Võrgumaterjal]. Available: https://www.pgoledb.com. [Kasutatud 28 04 2023].

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

Mina, Priit Post

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „MS Accessi andmebaasides SQL programmeerimist lihtsustava pistikprogrammi loomine lähtekoodiredaktorile Notepad++“, 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.

09.05.2023

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 – Süntaksiga seotud vead

Vea kirjeldus	**Viide**	**ABS-ist sõltuv**	**AB-st sõltuv**
Andmebaasisüsteemi mitte toetatud lausete kasutamine või õigete lausete kasutamine vales kohas	[35]	JAH	EI
Klauslite loogika või järjekorra segamini ajamine	[35] [37] [35]	EI	EI
Valesti struktureeritud alampäringu lause (sh kui tingimused pole paigutatud õigele tasemele)	[33] [37]	EI	EI
Viitamine veeru nimele, mis esineb alampäringu tulemuses korduvalt	[35] [37] [36]	EI	EI
Defineerimata aliase kasutamine	[37]	EI	EI
Tahetakse kirjutada lause lühemalt, kuigi see pole süntaktiliselt lubatud	[34] [40]	EI	EI
Defineerimata andmebaasiobjektide kasutamine	[37]	EI	JAH
Viidatakse veergudele, mida lähtetabelis või alampäringu tulemuses ei ole	[35] [34] [33] [39] [37] [40]	EI	JAH
Tabeli- ja veerunimede segamini ajamine	[35] [37]	EI	JAH
Puudub tabel, mille veerge soovitakse kasutada	[37] [40]	EI	JAH
GROUP BY mittekasutamine	[33] [37] [40]	EI	EI
Välja jäetud veerg GROUP BY klauslist	[35]	JAH	EI
Vead, mis on seotud GROUP BY klauslis kokkuvõttefunktsiooni kasutamisega	[35] [40]	EI	EI

Vea kirjeldus	**Viide**	**ABS-ist sõltuv**	**AB-st sõltuv**
Andmetüübi mittevastavus	[35]	EI	EI
Kokkuvõttefunktsiooni kasutamine väljaspool SELECT või HAVING klauslit (nimetatud probleem on artikli spetsiifiline, saab olla veel FROM ja WHERE klauslis olevates alampäringutes)	[35]	EI	EI
WHERE ja HAVING klauslis tingimused, mis pole tõeväärtusavaldised	[35] [33]	EI	EI
Projektsioon vales klauslis (nt WHERE eesnimi, perekonnanimi)	[35]	EI	EI
WHERE klausli mitu korda kasutamine samal lause tasemel (saab olla alampäringus – alampäringu ja põhipäringu WHERE klauslid on erinevatel lause tasemetel ja see on lubatud)	[35]	EI	EI
WHERE klauslis viidatakse kokkuvõttefunktsioonile, kuid see peab olema HAVING klauslis	[36]	EI	EI
Korduvad/mitmetähenduslikud funktsiooni nimed	[35]	JAH	JAH
Funktsiooni ja funktsiooni parameetri segamini ajamine	[35]	EI	EI
Veeru kasutamine funktsiooni parameetrina juhul kui veeru ja parameetri andmetüübid ei kattu	[35] [40]	EI	JAH
Defineerimata/ebaõige funktsiooni kasutamine	[35] [40]	JAH	JAH

Vea kirjeldus	**Viide**	**ABS-ist sõltuv**	**AB-st sõltuv**
SELECT * alampäringus, mille põhipäringuga võrdlemiseks: IN/NOT IN/=ANY/<>ALL	[63]	EI	EI
Sobimatu operaator (nt IS vs LIKE)	[35]	EI	EI
Ebaõige tingimuste süntaks (nt LIKE('A','B')	[35]	EI	EI
Operaatori IN kasutamine ilma väärtuste hulgata	[35]	EI	EI
Operaatori IS kasutamine seal, kus pole ette nähtud	[35]	EI	EI
Andmebaasisüsteemi mittetoetatud operaatorite kasutamine (nt &&, == või IS NOT)	[35] [37]	JAH	EI
Koma ja AND operaatori segamini ajamine	[36] [37]	EI	EI
Üleliigne või puuduv semikoolon	[35]	EI	EI
Komade ära jätmine	[35] [37]	EI	EI
Jutumärkide ära jätmine	[35] [37]	EI	JAH
Jutumärkide kasutamine, kui neid pole vaja või apostroofide ja jutumärkide segamini ajamine	[35]	EI	JAH
Loogeliste, kant- või ümarsulgudega seotud vead	[35] [37] [38]	EI	EI
Vale skeemi nimi	[35]	EI	JAH
Unustatakse BY võtmesõna fraasist ORDER BY või võtmesõna ON ühendamisoperatsiooni läbiviimise lausest või INTO võtmesõna INSERT lausest	[37] [36]	EI	EI

Vea kirjeldus	**Viide**	**ABS-ist sõltuv**	**AB-st sõltuv**
INSERT .. SELECT *	[63]	EI	EI
Unustatakse kasutada võtmesõna FROM, kui see on vajalik	[37]	EI	EI
Funktsiooni parameetrina kasutatakse DISTINCT, kui see pole kohane	[35] [34] [40]	JAH	EI
Trükiviga	[35] [37]	EI	Sõltub
Puuduv eessõna ON	[33]	EI	EI
Üleliigne (nt IN/EXISTS ees) või puuduv operaator	[35]	EI	EI
Tabelite ühendamisel teise tabeli määramata jätmine	[35] [36]	EI	EI
FROM klausli ära jätmine	[35]	JAH	EI
Sama klausli korduv kasutamine samal lause tasemel	[35]	EI	EI

Lisa 3 – Semantikaga seotud vead

Vea kirjeldus	**Viide**	**ABS-ist sõltuv**	**AB-st sõltuv**
Jutumärkide ära jätmine	[35] [37]	EI	JAH
Trükiviga, mis asub mujal kui võtmesõnas või operaatori/funktsiooni nimes	[35] [37]	EI	Sõltub
Puuduv, tautoloogiline, vastuoluline või vale avaldis	[35] [40]	EI	EI
WHERE ja HAVING klauslis valed tingimused (nt hinne=1 või 2)	[35] [33] [63]	EI	EI
Ebakorrektne võrdlusoperaatori kasutamine või ebakorrektse väärtusega võrdlemine	[35]	EI	Sõltub
Võrreldamatute väärtuste omavaheline võrdlemine (nt tänav = linn) (artikli spetsiifiline viga)	[37]	EI	EI
Otsingutingimused: =NULL või <>NULL	[63]	EI	EI
LIKE predikaadi muster ilma metamärkideta	[35] [34]	EI	EI
OUTER JOIN + COUNT(*)	[63]	EI	EI
DISTINCT kasutamine, kui see pole vajalik (nt SELECT lauses või koondandmete leidmise funktsioonis)	[35] [34]	EI	EI
Päringus ei tegeleta olukorraga, kus (alam)päringu tulemuseks võib olla NULL (WHERE või HAVING klauslis oleva NOT	[34] [35]	EI	JAH

Vea kirjeldus	**Viide**	**ABS-ist sõltuv**	**AB-st sõltuv**
IN/<>ALL (alampäring) tingimuse korral tagastab alampäring NULLi)			
HAVING klausli kasutamine ilma kokkuvõttefunktsioonita ja ilma GROUP BY klauslita	[35] [34]	EI	EI
Ebaefektiivne HAVING klausel (kui tingimust on võimalik kontrollida WHERE lauses, tuleks seda teha)	[35] [34]	EI	EI
Ühesugused aliased	[34] [40]	EI	EI
Sama nimega veerg on päringu tulemuses korduvalt	[35]	EI	JAH
SELECT * alampäringus, mille põhipäringuga võrdlemiseks: IN/NOT IN/=ANY/<>ALL	[63]	EI	EI
FROM klauslis rohkem kui üks tabel ja SELECT klauslis * (ilma täpsustava tabelita)	[63]	EI	EI
Puuduv ühendamise tingimus või selle ebaõige kasutamine	[35] [34] [33] [37] [40]	EI	EI
Korduvad read päringu vastuses või puuduv DISTINCT	[35] [34] [39]	EI	JAH
WITH klauslis defineeritakse ühine tabeli avaldis, kuid seda põhipäringus ei kasutata	[37]	JAH	EI

Lisa 4 – Loogikaga seotud vead

Vea kirjeldus	**Viide**	**ABS-ist sõltuv**	**AB-st sõltuv**
Puuduvad või ülemäärased veerud päringu tulemuses	[35] [34] [33] [39] [37] [40]	EI	EI
Üleliigne avaldis	[35] [40]	EI	EI
WHERE klauslis puuduv asjakohane tingimus	[33]	EI	EI
Päringus WHERE klausli otsingutingimuses väärtus, mida veerus ei ole (nt soo klassifikaatori väärtused M ja F, kuid päringusse kirjutatakse W)	[34]	EI	JAH
Aetakse segamini SUM ja COUNT	[37]	EI	EI
Ebakorrektse funktsiooni kasutamine, sh ebakorrektne veerg funktsiooni parameetrina	[35]	JAH	JAH
SELECT lauses puuduv AS, kui oli vaja tulemuse tabelis veerg ümber nimetada (artikli spetsiifiline viga)	[35]	EI	EI
=ALL kasutamine =ANY asemel või <>ANY kasutamine <>ALL asemel	[63]	EI	EI
Kokkuvõttefunktsiooni argumendi positsioonist puudub DISTINCT, kui see on vajalik	[35] [40]	EI	EI
Veerus konstantne väärtus (ei pruugi alati viga olla, viidatud artiklites käsitleti seda veana)	[35] [39]	EI	EI
ORDER BY klauslis puuduv või vale veerg	[35]	EI	EI

Vea kirjeldus	**Viide**	**ABS-ist sõltuv**	**AB-st sõltuv**
DISTINCT ei ole kasutatav esimese elemendi saamiseks järjendist	[36] [37]	EI	EI
Metamärkide ebaõige kasutamine: kasutatakse märkide % ja _ asemel mõnda muud märki või aetakse need segamini	[35]	JAH	EI
Ühendatakse vale tabel või ühendamiseks kasutatakse vale veergu	[35] [40]	EI	EI
Ühendamise tingimuses ebaõige operaator	[35]	EI	EI
Aetakse segamini a>0 ja IS NOT NULL või tühi string ja NULL	[35]	EI	EI
Üleliigne DISTINCT kasutamine juhul, kui see eemaldab vajalikud duplikaadid	[35] [34]	EI	EI
Mustri kasutamine ilma LIKE predikaadita	[35] [34]	EI	EI
Üleliigsete tabelite ühendamine (sh kui ühendatakse tabelid, mille primaarvõtmeid vajatakse – nendel juhtudel on need juba välisvõtmetena olemas)	[35] [34] [37] [40]	EI	JAH
Ebavajalik GROUP BY, kui pole kasutatud HAVING klauslit	[34] [33] [36] [40] [35]	EI	EI
Ebavajalik kokkuvõttefunktsioon	[35] [34] [37] [40] [33]	EI	EI
Puuduv kokkuvõttefunktsioon, nt SUM()/COUNT() kasutamine AVG asemel	[63] [33]	EI	EI

Vea kirjeldus	**Viide**	**ABS-ist sõltuv**	**AB-st sõltuv**
Puuduv või üleliigne ORDER BY klausel, sh üleliigne alampäringus (välja arvatud siis, kui seal on TOP n, FETCH FIRST n ROWS)	[35] [33]	EI	EI
HAVING klausli mittekasutamine	[33] [40]	EI	EI
Kasutatakse INSERT lauset SELECT asemel	[37]	EI	EI
Operaatori segamini ajamine (AND asemel OR ja vastupidi, = asemel <>)	[35]	EI	EI
Alampäringu tingimus, mis väljastab rohkem kui ühe tulemuse	[34]	EI	JAH
Alampäringu mitte kasutamine (nt JOIN asemel, kui tabelid on suured)	[33] [40]	EI	JAH
Kasutamata alias	[35]	EI	EI
Avaldiste (sh alampäringute) ebaõige pesastamine, see tähendab sulgude mitte kasutamisest tingitud vead	[35]	EI	EI
IN alampäring koos ainult ühe võimaliku väärtusega tulemuses	[34]	EI	Sõltub
Kui andmetüübid on teada, on erinevate andmetüüpidega väärtuste võrdlemine kahtlust äratav	[34]	EI	JAH
Vahel teisendatakse sõned arvudeks, mis võib anda tulemuseks RUNTIME ERROR	[34]	EI	EI

Vea kirjeldus	**Viide**	**ABS-ist sõltuv**	**AB-st sõltuv**
Otsingutingimuses nii AND (BETWEEN x AND y ei lähe arvesse) kui OR, kuid pole sulge	[63]	EI	EI
IN/EXISTS on asendatav võrdlemisega	[35] [34]	EI	EI
GROUP BY kasutamine ühe reaga gruppides	[35] [34]	EI	JAH
Ebaefektiivne UNION, mis tuleks asendada UNION ALL-ga, kui kahe päringu tulemused on alati mittekattuvad ja kumbki päring ei tagasta duplikaate	[35] [34]	EI	JAH
SUM(1) kasutamine COUNT(*) asemel	[63]	EI	EI
Alampäringus olevat tingimust on võimalik viia kõrgemale (peapäringusse)	[35] [34]	EI	EI
Otsingutingimus on tarbetult keeruline nt: (SAL<500 AND SAL<200)	[34]	EI	EI

Lisa 5 – Keerukusega seotud vead

Vea kirjeldus	**Viide**	**AB-st sõltuv**
Üleliigne tabel	[40]	EI
Loogeliste, kant- või ümarsulgude liigne kasutamine, mis lisab keerukust	[35] [37] [38]	EI
Ebavajalik WHERE klausel (kui klausel ei sea täiendavaid tingimusi)	[39]	EI
FROM klauslis viidatud tabelite mitte kasutamine	[34]	EI
Veeru aliase kasutamine juhtudel, kui seda poleks vaja teha	[37]	EI
UNION kasutamine OR operaatoriga otsingutingimuse aseme	[35] [34]	EI
INSERT klauslis puuduvad veergude nimed	[63]	EI
Liiga keeruline SELECT lause EXISTS alampäringus	[35]	EI
WHERE (või HAVING) klauslis ebavajalik võrdlusoperatsioon	[34]	EI
OUTER JOIN-i on võimalik asendada INNER JOIN-iga	[35] [34]	JAH
Korduste eemaldamiseks kasutatakse GROUP BY klauslit DISTINCT-i asemel või kasutatakse GROUP BY ja DISTINCT-i koos	[35] [34]	EI
LIKE ilma metamärkideta	[35] [34]	EI

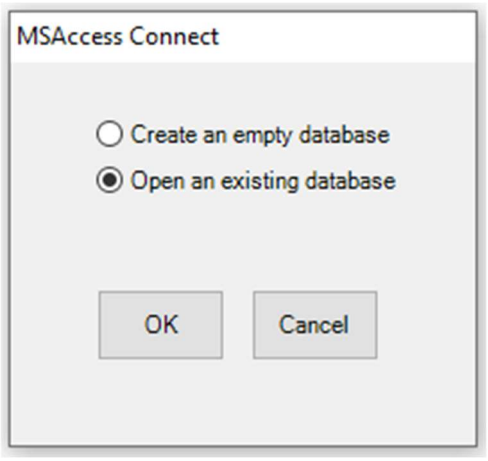
Vea kirjeldus	**Viide**	**AB-st sõltuv**
Otsingutingimuses <veerg> LIKE <veerg>	[63]	EI
Ühendamise mittekasutamine ja alampäringu liigne kasutamine (nt kui tabelid on väikesed)	[35] [34] [33] [39] [36] [38]	EI
IN/EXISTS on asendatav võrdlemisega	[35] [34]	EI
Üleliigsete tabelite ühendamine (sh kui ühendatakse tabelid, mille kandidaatvõtme väärtuseid vajatakse – nendel juhtudel on need juba välisvõtmetena olemas)	[35] [34] [37] [40]	EI
SELECT klauslis avaldis (sh konstant või funktsiooni poole pöördumine), kuid pole ise antud aliast.	[63]	EI
Ebavajalik kokkuvõttefunktsioon	[35] [34] [33]	EI
Puuduv kokkuvõttefunktsioon, nt SUM()/COUNT() kasutamine, kuid mitte AVG	[63]	EI
SUM(1) kasutamine COUNT(*) asemel	[63]	EI
Alampäringus ebavajalik GROUP BY, kui pole kasutatud HAVING klauslit	[35] [34]	EI
GROUP BY kasutamine ühe reaga gruppides, kui on teada, et kõik grupid sisaldavad vaid üht rida ehk reaalset grupeerimist ei toimu ja GROUP BY kasutamine on mõttetu	[35] [34]	JAH
Korduste eemaldamiseks kasutatakse GROUP BY klauslit, kuid kasulikum oleks kasutada DISTINCT-i	[35] [34]	EI

Vea kirjeldus	**Viide**	**AB-st sõltuv**
Veergude järjekorra segamini ajamine SELECT lauses	[40]	EI
GROUP BY kasutamine juhul kui eksisteerib vaid üks grupp, välja arvatud juhul, kui GROUP BY klauslis olevat veergu kasutatakse SELECT klauslis	[35] [34] [37]	JAH
ORDER BY <arv>	[63]	EI
UNION kasutamine OR operaatoriga otsingutingimuse aseme	[35] [34]	EI
Alampäringu mitte kasutamine (nt JOIN asemel, kui tabelid on suured)	[33] [40]	JAH
Üleliigne DISTINCT kasutamine juhul, kui duplikaadid on niigi eemaldatud	[35] [34]	EI
Ebaefektiivne UNION, mis tuleks asendada UNION ALL-ga, kui kahe päringu tulemused on alati mitte kattuvad ja kumbki päring ei tagasta duplikaate	[35] [34]	JAH
WHERE klauslis tingimus on ebaoluline	[33]	EI
ORDER BY klauslis on ebavajalikke veerge (sorteeritakse unikaalsete väärtustega veeru ja seejärel veel mingite veergude järgi)	[35]	EI
Alampäringus on ilma vajaduseta ORDER BY klausel	[35] [34]	EI

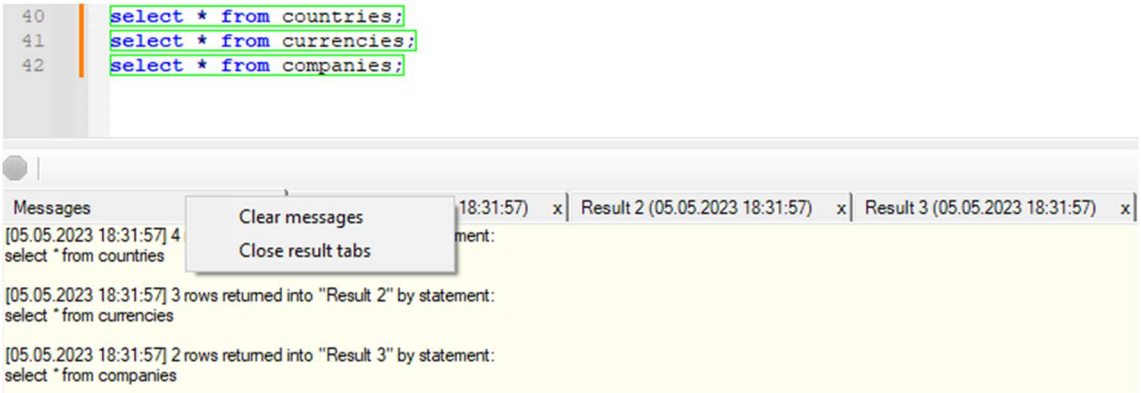
Vea kirjeldus	**Viide**	**AB-st sõltuv**
Ebaefektiivne HAVING klausel (kui tingimust on võimalik kontrollida WHERE lauses, siis tuleks seda teha seal)	[35] [34]	EI
Alampäringus olevat tingimust on võimalik viia kõrgemale (peapäringusse)	[35] [34]	EI
UNION [ALL]/INTERSECT [ALL]/EXCEPT [ALL]/MINUS [ALL]/ lauses SELECT *	[63]	EI
Otsingutingimus on tarbetult keeruline nt: (SAL<500 AND SAL<200)	[34]	EI
Tarbetult keeruline SELECT klausel EXISTS alampäringus. Enamasti tuleks kasutada märki * või 1 või viidet ühele veerule	[34]	EI
Vajalikke indekseid ei looda (artikli spetsiifiline viga)	[39]	JAH
Skripti vähene või puudulik treppimine	[38]	EI

Lisa 6 – Kuvatõmmised rakendusest

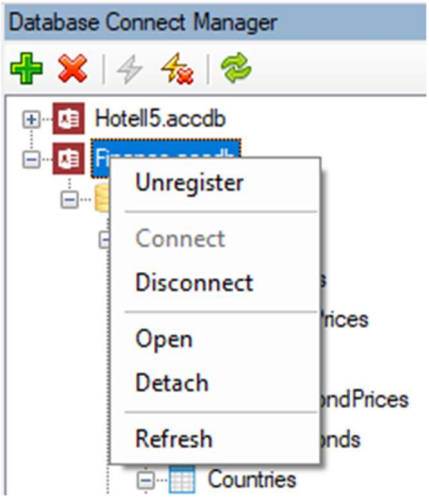
1. Muudetud ühenduse loomise aken:



2. Tulemuste logi sakk ja selle kontekstmenüü:



3. Andmebaasihalduri ühenduse kontekstmenüü:



Lisa 7 – Näited mitte käivitunud lausetest ja tehtud parandustest

1. Näited lausetest, mida algselt ei õnnestunud käivitada

RIGHT JOIN ühendamise operatsiooni kasutamine kuna SQLite sellist ühendamise viisi ei toeta (grammatikareeglite koostamisel sai võetud aluseks SQLite grammatika, täpsemalt jaotises .5.6):

```
367    SELECT Hotell.hotelli_nr, nimetus, Count(Reserveerimine.hotelli_nr) AS arv
368    FROM Reserveerimine RIGHT JOIN Hotell ON Hotell.hotelli_nr=Reserveerimine.hotelli_nr;
```

Lause DROP määrangu CASCADE tõttu ei saanud käivitada järgnevat lauset:

```
56     DROP TABLE B CASCADE;
```

2. Punktis 1 toodud probleemide lahendamine lähtekoodis

Algne SQL grammatika tabelite ühendamise süntaksireeglite kohta:

```
449  join_operator:
450      COMMA
451      | NATURAL_? (LEFT_ OUTER_? | INNER_ | CROSS_)? JOIN_
452  ;
```

Muudatuste järgne grammatika tabelite ühendamise süntaks:

```
474      join_clause:
475          ((LEFT_ | RIGHT_) OUTER_? | INNER_) JOIN_ table_or_subquery (ON_ expr)?
476      ;
```

Lause DROP määrangu CASCADE puudumine SQLite grammatikafailis:

```
248  drop_stmt:
249      DROP_ object = (INDEX_ | TABLE_ | TRIGGER_ | VIEW_) (
250          IF_ EXISTS_
251      )? (schema_name DOT)? any_name
252  ;
```

Lause DROP määrangu CASCADE puudumine MS Access'i dokumentatsioonist:

Syntax

DROP {TABLE *table* | INDEX *index* ON *table* | PROCEDURE *procedure* | VIEW *view*}

Järgnev kuvatõmmis on tehtud parandatud DROP lausest grammatikafailis:

```
422     drop_stmt:
423         DROP_ (object=(VIEW_ | PROCEDURE_) any_name
424         | object=TABLE_ table_name CASCADE_?
425         | object=INDEX_ index_name ON_ table_name)
426     ;
```