

TARTU ÜLIKOOL  
Arvutiteaduse instituut  
Informaatika õppekava

**Herbert Konnula**  
**PostgreSQL logifailide analüsaatori edasiarendus**  
**ning SQL-lausetes tehtud vigade analüüs**  
**Bakalaureusetöö (9 EAP)**

Juhendajad:  
Piret Luik, PhD  
Marina Lepp, PhD

Tartu 2025

# PostgreSQL logifailide analüsaatori edasiarendus ning SQL-lausetes tehtud vigade analüüs

## Lühikokkuvõte:

Andmebaasihalduri PostgreSQL logifailid sisaldavad palju informatsiooni, mille põhjal õppejõud saavad analüüsida protsessi, kuidas õppija jõudis korrektse tulemuseni. Kuna logifailide käsitsi analüüsimine on ajamahukas, töötati aastal 2024 kursuse „Andmebaasid“ tarbeks välja veebirakendus logifailide automaatseks analüüsimiseks. Bakalaureusetöö eesmärk oli arendada eelnevalt mainitud rakendust edasi, et muuta üliõpilaste esitatud logifailide analüüsimine mugavamaks ja efektiivsemaks. Lisaks analüüsiti 2025. aasta kevadel kursusele „Andmebaasid“ esitatud logifailide põhjal struktuurpäringukeele SQL õppimisel tehtavaid vigu. Analüüsi tulemustest selgus, et 41% SQL-lausetest sisaldasid vähemalt üht viga ning enim tehti kitsenduste rikkumise või SQL-keele süntaksiga seotud vigu.

**Võtmesõnad:** PostgreSQL, andmebaasid, logifailid, analüüs, veebirakendus

**CERCS:** P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine; S281 Arvuti õpiprogrammide kasutamise meetoodika ja pedagoogika

# **Development of the PostgreSQL Log File Analyzer and Analysis of Errors Made in SQL Statements**

## **Abstract:**

The log files of PostgreSQL contain a lot of information that instructors can use to analyze how students reached a given goal. Since manual log file analysis is time-consuming, a web application was built in 2024 to automate the process. The goal of this thesis was to further develop the application to make log file analysis more convenient and effective. The application was used with log files from the Databases course (spring 2025) to identify the types of errors students made while learning SQL. The results showed that 41% of the SQL statements contained at least one error, with the most common mistake related to constraint violations or issues with SQL syntax.

**Keywords:** PostgreSQL, databases, log files, analysis, web application

**CERCS:** P170 Computer science, numerical analysis, systems, control; S281 Computer-assisted education

## Sisukord

Sissejuhatus.....	5
1. Relatsiooniline andmebaas ja struktuurpääringukeel .....	7
2. Levinumad vead SQL pääringukeele õppimisel.....	8
3. PostgreSQL logifailide analüüsimise võimalused .....	11
3.1 PostgreSQL logifailid .....	11
3.2 PostgreSQL logifailide analüüsimise võimalused .....	13
3.3 Kursusel „Andmebaasid“ kasutatav logianalüsaator .....	14
4. PostgreSQL logianalüsaatori edasiarendus.....	18
4.1 Vigade kategoriseerimise täiendamine .....	18
4.1.1 Kategooriatesse „süntaksivead“ ja „muud vead“ kuuluvate vigade analüüs .....	19
4.1.2 Uute kategooriate lisamine põhiprogrammi .....	22
4.1.3 Uute kategooriate lisamine kasutajaliidesesse .....	23
4.2 SQL-lausetate filtreerimine vastavalt veakategooriale .....	24
4.3 Andmebaaside kopeerimise tuvastamine .....	26
4.4 Parandused põhiprogrammis ja kasutajaliideses.....	28
4.5 Rakenduse integreerimine ülikooli võrku .....	28
5. Kursusele „Andmebaasid“ esitatud logifailide analüüs.....	30
5.1 Analüüsitud logifailide valik.....	30
5.2 Metoodika .....	31
5.3 Analüüsi tulemused.....	31
Kokkuvõte.....	38
Viidatud kirjandus.....	39
Lisad.....	41
Lisa A – Juturobotile ChatGPT (GPT-4-turbo) esitatud pääringud ja nende vastused .....	41
Lisa B – Logifailide analüüsimise tulemused .....	46
Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks.....	47

## Sissejuhatus

Andmebaaside haldamist on arvutiteaduse erialade üliõpilastele õpetatud aastakümneid – Tartu Ülikooli Informaatika õppekavas oli kursus „Andmebaasid“ juba 1994. aastal [1] ning alates aastast 2001 kuulub see ka ACM-i<sup>1</sup> arvutiteaduse õppekava soovitude alla [2]. Lisaks õpetamisele on vajalik üliõpilaste arengu hindamine. Selleks on mitu võimalust: kas hinnata vaid lõpptulemust (kas üliõpilane jõudis korrektse lahenduseni) või jälgida kogu õppeprotsessi (ehk milliseid vigu õige lahenduseni jõudmise käigus tehti) [3].

Lõpptulemuse hindamiseks esitab üliõpilane loodud SQL-lause ning selle korrektsust kontrollitakse kas käsitsi või automatiseeritult. Õppeprotsessi analüüsimiseks on aga lisaks lõplikule (esitatud) SQL-lausele vaja teada ka seda, kuidas üliõpilane korrektse lahenduseni jõudis. Keeruline on kontrollida, milliseid materjale kasutati ülesande lahendamisel, kuid kuna kursusel „Andmebaasid“ on kasutusel andmebaasihaldur (ingl *Database Management System*) PostgreSQL, millel on olemas logimise funktsionaalsus, saab õige seadistuse korral logifailidest hiljem välja võtta kõik käsud, mis andmebaasile on tehtud [4]. Logifailidest on võimalik lisaks SQL-lausele välja lugeda käsu käivitamise aeg, sellega kaasnenud veateated ning palju muud, kuid nende käsitsi analüüsimine on väga aeganõudev.

2024. aasta kevadel arendas Külli Kangro välja SQL logifailide analüsaatori kursuse „Andmebaasid“ tarbeks [5]. Rakendus lihtsustas oluliselt logifailide analüüsimist, kuid vigade kategoriseerimine polnud piisavalt täpne – enamik vigadest liigitati süntaksivigade alla. Lisaks oli logidest vigaste lausete otsimine raskendatud, kuna puudus võimalus vigu kategooria järgi filtreerida, ning rakendus kulutas analüüsi teostamiseks õppejõudude arvates liiga palju aega.

Bakalaureusetöö eesmärk oli arendada edasi „Andmebaaside“ kursuse jaoks loodud logifailide analüsaatorit ning uurida valminud analüsaatori abil, milliseid vigu tegid 2025. aasta kevadel kursusel „Andmebaasid“ osalevad üliõpilased. Selle bakalaureusetöö fookus oli vigade kategoriseerimise parendamisel ning õppejõudude soovitud funktsionaalsuste lisamisel. Rakenduse probleemide ja võimalike edasiarenduste avastamiseks tehti pidevalt koostööd kursuse õppejõududega – töö autor lisati Discordi kanalisse, kus toimus logifailide ja analüsaatori teemaline arutelu ning anti teada, kui analüsaator ei toimunud nii, nagu oodati.

---

<sup>1</sup> ACM – Association for Computing Machinery

Töö on jagatud viieks peatükiks. Esimeses peatükis antakse lühiülevaade SQL-keelest ja PostgreSQL-ist, teises peatükis kirjeldatakse varasemate uuringute põhjal struktuurpäringukeele õppimisel tekkivaid vigu ning kolmandas peatükis keskendutakse PostgreSQL-i logifailide analüüsimisele ja tutvustatakse olemasolevaid logianalüsaatoreid. Neljandas peatükis kirjeldatakse selle bakalaureusetöö käigus loodud edasiarendusi ning viiendas peatükis antakse ülevaade sel kevadel kursusel „Andmebaasid“ osalenud üliõpilaste tehtud vigadest.

Bakalaureusetöö lõpus on kaks lisa. Lisas A on välja toodud tehisintellektiga logifailide analüüsimiseks (vt peatükk 3.2) kasutatud päringud ning juturobotilt ChatGPT (versioon GPT-4-turbo) [6] saadud vastused. Lisas B on peatükis 5 kirjeldatud logifailide analüüsimise tulemused tabelina.

# 1. Relatsiooniline andmebaas ja struktuurpäringukeel

Relatsiooniline andmebaas<sup>2</sup> (edaspidi andmebaas) põhineb relatsioonilisel andmemudelil. See tähendab, et andmebaasi struktuur on kirjeldatav matemaatiliste relatsioonidega [8]. Andmebaas koosneb ühest või mitmest tabelist, milles iga tulp on üks atribuut (nt *id*, *nimi*, *vanus*) ning iga rida sisaldab andmeid, kus igale atribuudile on määratud konkreetne väärtus (nt *57*, *Juhan*, *29*) [9]. Oluline on, et iga tulba andmetüüp ja piirangud (ingl *constraints*) määratakse tabeli loomisel ning andmebaasihaldur kontrollib, et kõik sisestatavad andmed on korrektsed – näiteks arvutüüpi tulpa ei saa sisestada teksti või mitut arvu [10].

Mitme tabeliga andmebaasis on tabelid omavahel seostatavad võtmete kaudu. Iga tabeli üks atribuut on määratud primaarvõtmeks (ingl *primary key*), mis on unikaalne, ehk kahel erineval real ei saa olla vastava atribuudi väärtus võrdne – näiteks tabelis *Riigid* võib primaarvõtmeks olla *riik\_id*. Tabeli *Riigid* ühendamiseks tabeliga *Linnad* tuleb linnade tabelisse lisada atribuut (nt *riik*), mille väärtus on võrdne linnale vastava riigi *riik\_id* väärtusega ning mis tabeli *Linnad* vaates on välisvõti (ingl *foreign key*) [11].

Struktuurpäringukeel ehk SQL (ingl *Structured Query Language*) on interaktiivne programmeerimiskeel, mis on loodud relatsiooniliste andmebaasidega manipuleerimiseks. Keele arendas aastal 1974 välja IBM ning see on tänaseni väga laialdaselt kasutusel [12]. SQL on deklaratiivne keel, mis tähendab, et selle põhifookus on programmi töö tulemusel, mitte tulemuse saavutamisel [13].

Struktuurpäringukeel koosneb hulgast võtmesõnadest (mille täpne hulk sõltub kasutatavast andmebaasihaldurist), mis on võetud inglise keelest (nt *SELECT*, *FROM*, *WHERE*), et muuta keel inimesele võimalikult loetavaks ja lihtsasti mõistetavaks. SQL-i käsustik jaguneb kolmeks: andmekirjeldus- (ingl *data definition*), andmekäitlus- (ingl *data manipulation*) ning andmepärimiskäsud (ingl *data extraction statements*) ehk päringud [10]. Esimesse kategooriasse kuulub näiteks käsk *CREATE*, teise *ALTER* ning kolmandasse *SELECT*. Lisaks käskudele sisaldab struktuurpäringukeel tingimusi (ingl *clause*, näiteks *WHERE*, *FROM*, *ORDER BY*) ja operaatoreid (nt *>*, *<*, *LIKE*) [14]. Lauset, mis sisaldab käske, tingimusi ja operaatoreid, nimetatakse SQL-lauseks (ingl *statement*) [15].

---

<sup>2</sup> Andmebaas – infokogum, mis on organiseeritud nii, et vajalikud andmed oleks arvutiprogrammile kergesti leitavad [7].

## 2. Levinumad vead SQL päringukeele õppimisel

Vigade tegemine on õppimisprotsessi loomulik osa – nii on see ka struktuurpäringukeele (SQL) õppimisel. SQL-lausetes tekivad vead peamiselt väärarusaamadest, mis on seotud varasemate teadmiste/kogemustega (muude programmeerimiskeelte või inglise keelega), üldistamisega või struktuurpäringukeele omadustega [16]. Vigade esinemise sagedus on tihti seotud nii kasutatava andmebaasi kui ka ülesande keerukusega, kuid sõltub ka sellest, kui selgelt on SQL-lause koostajale andmebaasi struktuuri kirjeldatud [17]. Selleks, et SQL-lausetes esinevaid vigu täpsemalt analüüsida, on otstarbekas vead klassifitseerida.

Varasemalt on SQL-lausetes vead liigitatud enamasti kahte klassi [18] – süntaksivead, mille puhul andmebaasihaldur annab veateate, ning semantika vead, mida andmebaasihaldur ei tuvasta – kuid Taipalus jt artiklis on laused jaotatud järgnevasse klassidesse [19]:

1. laused, mille tulemus on korrektne;
2. laused, mille korral on tulemustabel ebakorrektne;
3. laused, mille korral andmebaasihaldur annab veateate ning tulemust ei väljastata.

Taipalus jt [19] klassifikatsioonis kuuluvad esimesse kategooriasse nii korrektsed laused kui ka sellised laused, mis sisaldavad komplikatsioone, kuid tulemust ei mõjuta. Komplikatsioonid on enamasti seotud päringu ebaloogilise ülesehitusega või üleliigse osaga SQL-lausel (nt kaks tabelit on omavahel *JOIN*-käsuga ühendatud, kuid ühe tabeli sisu päringus tegelikult ei kasutata).

Teise kategooria moodustavad SQL-laused, mis on küll süntaktiliselt korrektsed, kuid ei väljasta õiget tulemustabelit [19]. See kategooria jaguneb omakorda loogika- ja semantikavigadeks. Loogikavead on konkreetselt seotud etteantud andmebaasi ja nõutud tulemusega, seega on nende tuvastamiseks vaja piisavat taustainfot. Kui ülesandeks on leida tabelist „Inimesed” isikud, kelle vanus on 20 või rohkem, siis päring

```
SELECT * FROM Inimesed WHERE vanus > 20
```

sisaldab loogikaviga – inimesi, kelle vanus on täpselt 20, ei väljastata. Samas, kui ülesanne oleks olnud leida üle 20-aastased isikud, oleks see päring olnud korrektne.

Erinevalt loogikavigadest saab semantilisi vigu tuvastada vaid SQL-lauset analüüsides, sõltumata andmebaasi struktuurist või ülesande sisust. Semantikavigade tuvastamisel

lähtutakse SQL-keele üldreeglid, mis määrab, et ühes reas saab ühes tulbas olla vaid üks väärtus, eelneva näite põhjal – ühel inimesel ei saa olla kaks erinevat vanust. Näiteks päring

```
SELECT * FROM Inimesed WHERE vanus = 20 OR vanus = 30
```

ei sisalda semantikavigu (kuigi võib sisaldada loogikaviga, sõltuvalt ülesande püstitusest), kuid päring

```
SELECT * FROM Inimesed WHERE vanus = 20 AND vanus = 30
```

on semantiliselt ebakorrekne, kuna vanusel ei saa olla mitu väärtust korraga. Sarnaselt loogikavigadele ei tuvasta andmebaasihaldur üldiselt ka semantikavigu.

Taipaluse jt [19] liigituses moodustavad kolmanda kategooria SQL-laused, mis sisaldavad vähemalt üht süntaksiviga. Selliste lausete esitamisel annab andmebaasihaldur veateate ning tulemustabelit ei teki, mistõttu on lause koostajal süntaksivigu kõige kergem märgata ning parandada [20]. Süntaksiviga võib olla seotud nii SQL-keele põhimõtete vastu eksimisega, näiteks vale võtmesõna kasutamisega (näiteks *SELECT* asemel *UNION*), kui ka konkreetse andmebaasi struktuuri vale tõlgendamisega. Andmebaasi struktuuriga seotud süntaksivigu sisaldav lause võib andmebaasilt küsida näiteks tabelit või veergu, mida andmebaasis ei eksisteeri.

Miedema jt [16] artiklis on kasutatud veelgi spetsiifilisemat jaotust, kus vead on jagatud kaheksasse kategooriasse: süntaksiprobleemid, ebakorrekne või puuduv tabel/atribuut, ebakorrekne või puuduv võtmesõna, vale tagastatav tulemus, struktuuri viga, probleemid *alias*'tega, koondamine ja komplikatsioonid [16]. Sarnaselt Taipaluse jt [19] jaotusele on ka siin võetud kõik süntaksivead üheks kategooriaks kokku, kuid kuna suure osa süntaksivigade põhjuseks on trükivead, on Ahadi jt [21] otsustanud süntaktilised vead eraldi alamkategoriateks jagada.

Esmalt on Ahadi jt [21] artiklis süntaksivead (PostgreSQL-is veakood 42601) jagatud neljaks vastavalt tekkepõhjusele – vale süntaks, lõpetamata kooditükk, viga alamlauses ning muud süntaksivead – kusjuures esimene kategooria moodustab 97% ning ülejäänud kategooriatesse jaguneb vaid 3%. *SELECT*-päringutes tekkinud süntaksivead on kategoriseeritud vastavalt

lause keerukusele ning tabelist selgub, et enim süntaksivigu tekib just keerukamate SQL-lausetel – näiteks *GROUP BY* või *JOIN* võtmesõnade kasutamisel [21].

Taipalus ja Grahn [20] on samuti välja toonud, et süntaksivead on struktuurpäringukeeles kõige levinumad ning sellised vead võivad tekkida erinevatel põhjustel. Samas on nad artiklis välja toonud, et süntaksivigadega kaasnevad veateated on üldjuhul, sõltuvalt andmebaasihaldurist, segadust tekitavad ning kasutud. Sellest tulenevalt pakkusid nad välja soovitusi, kuidas SQL-i veateateid andmebaasi kasutajale kasulikumaks muuta – nad töötasid välja raamistiku, mille järgi, mille järgi peaks veateade sisaldama:

- 1) vea asukohta (näitama rida, kus viga tekkis);
- 2) vea tekkimise põhjust (mis tekitas vea või miks viga tekkis);
- 3) soovitusi, kuidas viga parandada.

Lisaks soovitavad nad kasutada lihtsat sõnastust, eemaldada veateatest ebavajalikud elemendid ning tõsta teates esikohale kõige olulisem osa.

### 3. PostgreSQL logifailide analüüsimise võimalused

PostgreSQL on avatud lähtekoodiga relatsioonandmebaasihaldur, mida on arendatud üle 30 aasta. PostgreSQL 17 (2025. aasta märtsi seisuga uusim versioon) võimaldab lisaks SQL:2023 standardis kirjeldatud funktsionaalsustele (PostgreSQL täidab 177-st nõutud funktsionaalsusest vähemalt 170) luua programmeerijal ise muutujatüüpe ja funktsioone [22], kusjuures funktsioonide loomiseks saab kasutada peale standardse SQL-i ka muid programmeerimiskeeli, näiteks PL/pgSQL ja PL/Python [23].

PostgreSQL-i logifailide analüüsimiseks on mitmeid võimalusi. Esiteks on võimalus analüüsida faile käsitsi, mis on väga ajakulukas [24] ning ei võimalda näha suuremat pilti (näiteks vigaste lausete protsenti). Teiseks on olemas erinevaid (nii tasulisi kui ka tasuta) programme, mis võimaldavad logifailidest kokkuvõtteid teha, kuid selleks, et mõista, kuidas logianalüsaatorid toimivad, tuleb esmalt tutvuda PostgreSQL-i logifailide struktuuriga.

#### 3.1 PostgreSQL logifailid

PostgreSQL võimaldab kasutaja tegevuse logimist viiel erineval meetodil: „*stderr*“, „*csvlog*“, „*jsonlog*“, „*syslog*“ ja „*eventlog*“. Logimise meetod ning logifaili asukoht määratakse PostgreSQL-i konfiguratsioonifailis *postgresql.conf*. Võimalik on korraga kasutada mitut logimismeetodit ning eelmainitud konfiguratsioonifailis saab ka määrata, millal ja mida logidesse salvestatakse [4]. „Andmebaaside“ kursusel kasutatakse „*jsonlog*“ meetodit, mis salvestab kasutaja tegevuse JSON-formaadis.

JSON-formaadis dokumendi struktuur koosneb objektidest, millest iga objekt on piiratud loogeliste sulgudega (“{“ ja “}” vahel) [25]. Objekti sees on võti-väärtus-paarid, kus võtmeks on sõne (ingl *string*) ning väärtuseks kas sõne, arv, tõeväärtus (ingl *boolean*), järjend (ingl *array*), objekt (ingl *object*) või *null* ehk defineerimata väärtus. Võti ja väärtus on omavahel eraldatud kooloniga (“:”) ning võti-väärtus-paarid on üksteisest eraldatud komaga (“,”).

PostgreSQL-i „*jsonlog*“ logifail koosneb objektidest, mis sisaldavad kuni 29 võti-väärtus-paari. Logifaili põhjal lausete ja neis sisalduvate vigade analüüsimiseks on tarvis SQL-lause sisu (väli „*statement*“), veateate sisu (väli „*message*“) ning veakoodi (väli „*state code*“), millele mõnel juhul (peamiselt pisemate trükivigade korral) lisandub ka nõuanne (väli „*hint*“). Kõik logifaili objektid sisaldavad ka välja „*timestamp*“, mille põhjal saab määrata käsu

käivitamise aja. „*Timestamp*“ väärtus koosneb kuupäevast, kellaajast (millisekundi täpsusega) ning ajatsoonist, mis on määratud eelpool mainitud konfiguratsioonifailis. Joonisel 1 on kujutatud „*jsonlog*“ meetodil loodud logifaili objekt.

```
{
  "timestamp": "2025-02-23 12:10:45.866 EET",
  "user": "postgres",
  "dbname": "postgres",
  "pid": 11168,
  "remote_host": "127.0.0.1",
  "remote_port": 56646,
  "session_id": "67baf106.2ba0",
  "line_num": 11,
  "ps": "",
  "session_start": "2025-02-23 11:57:26 EET",
  "vxid": "12/12",
  "txid": 0,
  "error_severity": "ERROR",
  "state_code": "42601",
  "message": "syntax error at or near \"from\"",
  "statement": "select klubid.id, klubid.nimi, \\r\\nfrom klubid",
  "cursor_position": 34,
  "application_name": "DBeaver 24.3.4 - SQLEditor <Script-4.sql>",
  "backend_type": "client backend",
  "query_id": 0
}
```

Joonis 1. PostgreSQL logifaili objekt.

Seda, kas logifaili kirjutatakse kõik SQL-laused, hoiatusega laused või vaid veateate tekitanud laused, saab määrata PostgreSQL-i konfiguratsioonifailis parameetriga „*log\_min\_messages*“, millel on 12 erinevat väärtust [4]:

1. *PANIC* – logitakse laused, mis põhjustasid kõigi andmebaasisessioonide katkestamise;
2. *FATAL* – logitakse laused, mis põhjustasid konkreetse sessiooni katkestamise;
3. *LOG* – logitakse administraatorile kasulik informatsioon;
4. *ERROR* – logitakse laused, mis põhjustasid päringu ebaõnnestumise;
5. *WARNING* – logitakse laused, mis põhjustasid hoiatuse tekkimise (tegevus ei ebaõnnestunud, kuid sisaldab veaohlikku süntaksi);
6. *NOTICE* – logitakse kasutajale kasulik informatsioon;
7. *INFO* – logitakse ka kaudselt küsitud informatsioon;
8. *DEBUG* (tasemed 1 kuni 5) – logitakse vastavalt valitud tasemele informatsiooni, mis on mõeldud arendajatele.

Iga logimistase hõlmab ka kõiki eelnevaid tasemeid – näiteks, kui „*log\_min\_messages*“ väärtuseks on määratud *ERROR*, logitakse kõik laused, mille veatase (*error\_severity*) on *PANIC*, *FATAL*, *LOG* või *ERROR*. Selle parameetri vaikeväärtus on *WARNING*. Kursusel „Andmebaasid“ määratakse parameetri „*log\_statement*“ väärtuseks „*all*“, et kirjutada logifaili ka korrektsed SQL-laused. JSON-tüüpi logifailide genereerimiseks peab parameetri „*logging\_collector*“ väärtuseks olema määratud „*on*“. Lisaks eelnevale on logide analüüsimisel oluline õige ajatsoon (kuna see mõjutab „*timestamp*“ välja väärtust) ja keel (tekitab probleeme, kui kasutatav logianalüsaator töötleb lisaks veakoodidele ka veatedete sisu).

### 3.2 PostgreSQL logifailide analüüsimise võimalused

PostgreSQL on suhteliselt populaarne andmebaasihalduse tarkvara ning on sellele loodud mitmeid logifailide analüsaatoreid – populaarseim neist on avatud lähtekoodiga pgBadger [26]. See võimaldab kiirelt analüüsida suurte andmebaaside logisid ning annab kasutajale väga palju erinevaid andmeid – võimalik on näha nii esitatud SQL-lausete koguarvu, esitajate IP-aadresse/kasutajanimed, lausete jaotust tüüpide järgi kui ka seda, kui palju mingi käsu jooksutamine serveris aega on võtnud. Selline informatsioon on kasulik suurte andmebaaside puhul, kuid õppetöös kasutatavate väikeste andmebaaside puhul ei ole lausete efektiivsus oluline. Samas ei võimalda pgBadger vigu sisaldavaid lauseid eriti täpselt klassifitseerida – see on võimalik vaid PostgreSQL-i sisseehitatud veatasemete alusel.

Lisaks pgBadgerile on PostgreSQL logifailide analüüsiks võimalik kasutada pganalyze Log Insights [27] programmi, mis on tasuline ning eeldab programmi installimist serverisse [28]. Erinevalt pgBadgerist suudab pganalyze Log Insights analüsaator ka veakoodide alusel kasutaja tehtud vigu tuvastada [29].

Tulenevalt viimasel kümnendil toimunud tehisintellekti arengust, ei saa mainimata jätta tekstiroboteid, mis suudavad lisaks küsimustele vastamisele ka faile töödelda (näiteks OpenAI loodud ChatGPT [6]). Kuna tekstiroboti vastus sõltub väga palju konkreetsest päringust (ingl *prompt*) [30], on lisas A välja toodud nii päringute tekst kui ka roboti vastused. Testimiseks kasutati suvaliselt valitud JSON-formaadis logifaili 2025. aasta kevadel toimunud „Andmebaasid“ kursusel ning enne faili jagamist tehisintellektiga kontrolliti, et see ei sisaldaks tundlikku informatsiooni.

Juturobotile ChatGPT (versioon GPT-4-turbo) [6] tehti kokku kolm päringut. Esimese päringu tulemusel leidis tehisintellekt failist kokku 74 SQL-päringut, millest kõik sisaldasid süntaksivigu. Ajavahemikku tuvastada ei õnnestunud, kuid genereeriti üht kategooriat sisaldav tulpdiaagramm.

Teises päringus paluti analüüsida faili uuesti ning jagada leitud vead vähemalt kaheksasse kategooriasse. Sel korral ei leidnud juturobot failist ühtki SQL-päringut, seega ka mitte ühtegi viga, kuid eelmise päringu tulemuste põhjal koostati uuesti üht kategooriat sisaldav tulpdiaagramm. Kolmandal katsel ei nõustunud programm enam andmeanalüüsi tegema, kuna testimiseks kasutati ChatGPT tasuta versiooni. Katse ei saavutanud küll soovitud tulemusi, kuid on tõenäoline, et juturoboti tasuline versioon oleks analüüsiga paremini toime tulnud.

### 3.3 Kursusel „Andmebaasid“ kasutatav logianalüsaator

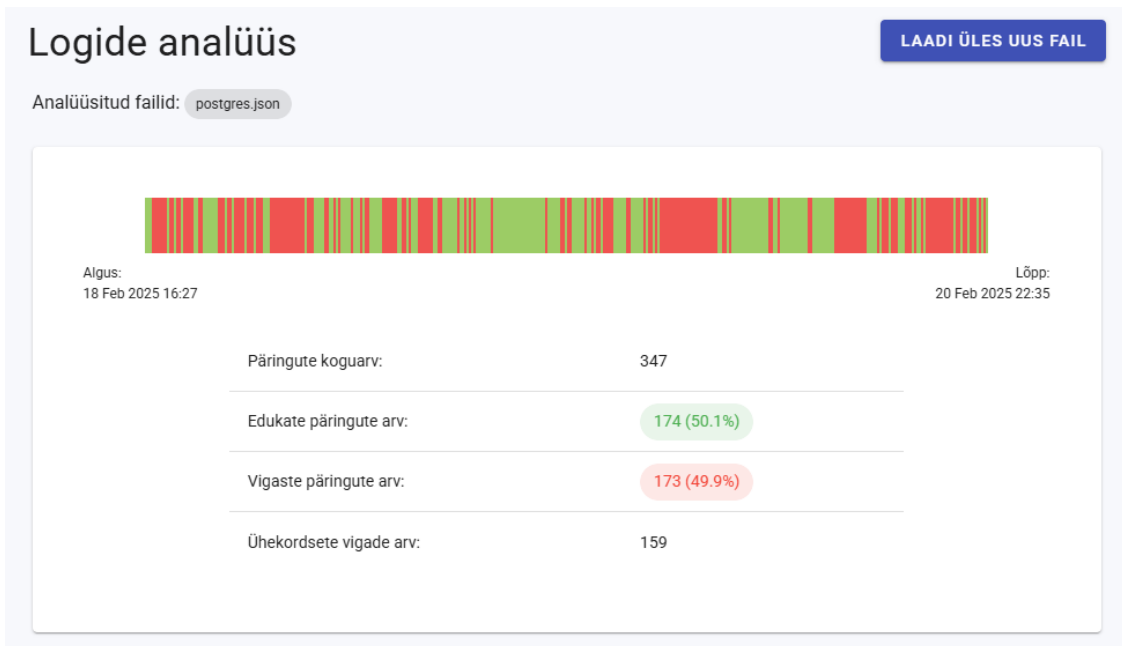
Varasemalt oli kursusel „Andmebaasid“ kasutusel 2024. aastal Külli Kangro bakalaureuse-tööna [5] valminud logifailide analüsaator<sup>3</sup>. Rakendus koosneb veebipõhisest kasutajaliidestest (*front-end*) ja põhiprogrammist (*back-end*). Kasutajaliides kasutab Vue.js raamistikku ning põhiprogramm Java Spring Boot'i [5].

Kangro loodud rakendus võimaldab analüüsida nii üksikuid *log*- või *json*-laiendiga faile kui ka *zip*-failina esitatud logifailide kausta. Pärast faili üleslaadimist saab kasutaja valida ajavahemiku, milles logisid analüüsitakse – kõik logid, viimane nädal või käsitsi määratud vahemik. Kui analüüsitav fail ning ajavahemik on määratud, vajutab kasutaja „Analüüsi“ nuppu ning rakendus alustab failide analüüsimist.

Kui logifail(id) on analüüsitud, suunatakse kasutaja edasi vaatesse „Logide analüüs“. Selles vaates kuvatakse vastavalt esitatud failile kas ühe või mitme logifaili kokkuvõte. Kui esitati *zip*-fail, saab kasutaja vasakus ääres asuvast menüüst valida, milliseid logifaile soovib analüüsida, kusjuures failid on grupeeritud faili esitaja nime järgi. Kuvatõmmis vaatest „Logide analüüs“ on toodud joonisel 4.

---

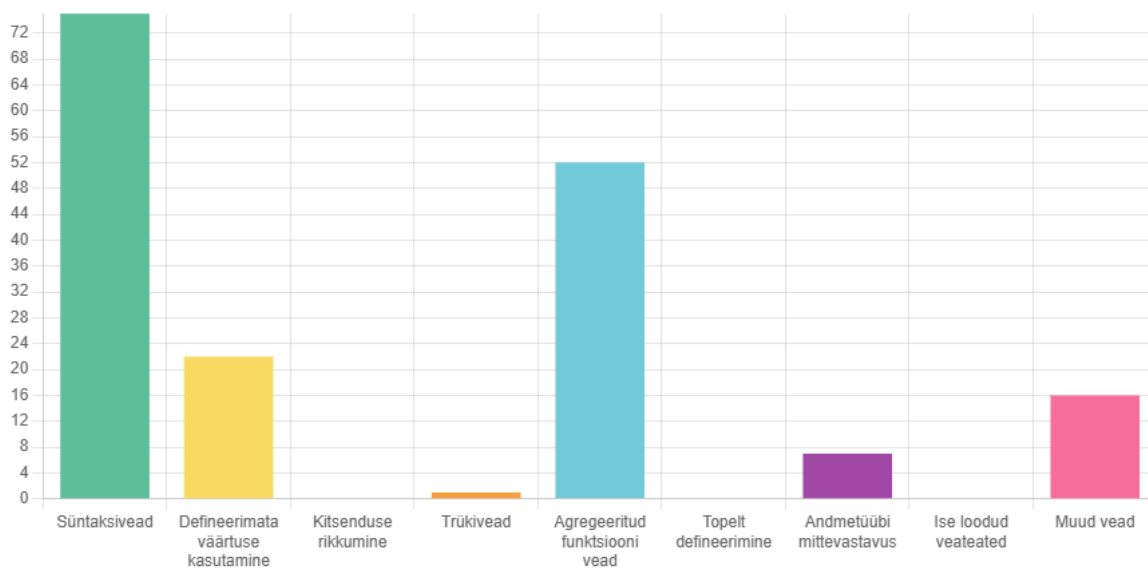
<sup>3</sup> Kangro loodud rakendus on kättesaadav aadressil <https://postgresql-log-analyser.onrender.com/> ning rakenduse lähtekood aadressil <https://github.com/kangrok/postgresql-log-analyser>.



Joonis 4. Vaade „Logide analüüs“.

Analüüsi vaates kuvatakse esmalt päringute koguarv, edukate ja vigaste päringute arv, ühekordsete vigade arv – analüsaatoris kasutatakse terminit „päring“ kõigi SQL-lausetega kohta. Seejärel kuvatakse veatüüpide jaotus tulpdiagrammina, kus vead on liigitatud üheksasse kategooriasse: süntaksivead, defineerimata väärtuse kasutamine, kitsenduse rikkumine, trükivead, agregeeritud funktsiooni vead, topelt defineerimine, andmetüübi mittevastavus, ise loodud veateated ning muud vead. Diagrammi all kuvatakse korduvad vead koos korduste arvuga. Tulpdiagramm on esitatud joonisel 5.

## Veatüüpide jaotus



Joonis 5. Veatüüpide jaotuse tulpdiaagramm.

Individuaalse analüüsi vaates kuvatakse lisaks eelnevale käskude sooritamise ajaskaala (vt joonist 4) ning kõik logifailis sisalduvad SQL-laused ajalises järjestuses. Iga lause juures kuvatakse vea tüüp (veatu lause puhul „korrektne“) ning lause sisu. Moodul „Kõik päringud“ on kujutatud joonisel 6.

## Kõik päringud

	PIKENDA PÄRINGUD	LÜHENDA PÄRINGUD
Aeg	Vea tüüp	Päring
20/02/2025, 20:34:54	Korrektne	<code>select isikud.eesnimi    ' '    isikud.perenimi as Must, (parti...</code>
20/02/2025, 20:34:56	Korrektne	<code>select isikud.eesnimi    ' '    isikud.perenimi as Must, (parti...</code>
20/02/2025, 20:35:00	Korrektne	<code>select isikud.eesnimi    ' '    isikud.perenimi as Must, (parti...</code>
20/02/2025, 20:35:02	Korrektne	<code>select isikud.eesnimi    ' '    isikud.perenimi as Must, (parti...</code>
20/02/2025, 20:35:05	Korrektne	<code>select isikud.eesnimi    ' '    isikud.perenimi as Must, (parti...</code>
20/02/2025, 20:35:05	Korrektne	<code>select isikud.eesnimi    ' '    isikud.perenimi as Must, (parti...</code>
20/02/2025, 20:35:08	Korrektne	<code>select isikud.eesnimi    ' '    isikud.perenimi as Must, (part...</code>
20/02/2025, 20:35:09	Korrektne	<code>select isikud.eesnimi    ' '    isikud.perenimi as Must, (part...</code>
20/02/2025, 20:35:33	Korrektne	<code>select isikud.eesnimi    ' '    isikud.perenimi as Must, (parti...</code>
20/02/2025, 20:35:35	Korrektne	<code>select isikud.eesnimi    ' '    isikud.perenimi as Must, (parti...</code>

Päringute arv leheküljel:  1-10 of 200 < > >>

Joonis 6. Moodul „Kõik päringud“.

Moodulis „Kõik päringud“ kuvatakse iga SQL-lause juures ka kuupäev ja kellaaeg, millal lause andmebaasile saadeti. Tulbas „Päring“ kuvatakse vaikimisi vaid SQL-lause algus, kuid paremal olevale noolekesele vajutades saab näha kogu lauset ning ebakorrekse lause korral ka sellega kaasnenud veateadet. Mooduli ülemises ääres on nupp „Pikenda päringud“, mis kuvab kõik SQL-laused täispikkuses (koos veateadetega), ning nupp „Lühenda päringud“, mis kõik laused uuesti lühendab (taastab algse vaate).

## 4. PostgreSQL logianalüsaatori edasiarendus

Bakalaureusetöö raames arendati edasi Külli Kangro loodud logianalüsaatorit, võttes arvesse nii Kangro bakalaureusetöös välja toodud ettepanekuid [5] kui ka kursuse „Andmebaasid“ õppejõudude soove. Osa uuendustest lepiti bakalaureusetöö juhendajaga kokku 2024. aasta sügisel, kuid mõned probleemid ja ettepanekud selgusid 2025. kevadel kursuse „Andmebaasid“ toimumise ajal. Vastavalt bakalaureusetöö autori varasemale kogemusele ning edasiarenduste elluviimiseks vajaminevale töömahule seati edasiarenduse tulemusele järgnevad eesmärgid:

- kategooriatesse „süntaksivead“ ja „muud vead“ kuuluvad laused jaotatakse eraldi alamkategooriatesse vastavalt lause ja/või veateate sisule;
- trükivigade tuvastamine muudetakse täpsemaks, teostatakse sõnade sarnasuse tuvastus;
- parandatakse veatuvastust – lisatakse tuvastatavaid tingimusi olemasolevatele vealiikidele ning vajadusel luuakse uusi kategooriaid;
- tuuakse eraldi välja SQL-laused, mis on seotud tabeliga „Inimesed“, et tuvastada andmebaaside kopeerimist;
- luuakse filtreerimise võimalus „Kõik päringud“ moodulisse;
- parandatakse logide filtreerimist aja järgi, lisatakse ajatsoonide tuvastamine;
- *zip*-faili sisestamisel järjestatakse selle sees olevad failid alfabeetiliselt;
- rakendus kolitakse ülikooli serverisse ning luuakse vastav nimeserveri kirje.

Selles peatükis on selgitatud edasiarenduste sisu ning sobivate lahendusteni jõudmise protsessi. Rakenduse<sup>4</sup> toimimise analüüsimiseks kasutati 2024. aastal kursusel „Andmebaasid“ kogutud logifaile, mille kasutamiseks oli saadud luba Tartu Ülikooli eetikakeskuselt<sup>5</sup>. Rakendus on avalikult kättesaadav aadressil <https://sql-analyzer.cs.ut.ee/>.

### 4.1 Vigade kategoriseerimise täiendamine

Esimese probleemina tõi kursuse „Andmebaasid“ vastutav õppejõud 2024. aasta sügisel välja, et rakenduses kuvatav veatüüpide jaotus ei ole piisavalt informatiivne, kuna suur hulk vigu liigitatakse kas süntaksivigade või muude vigade alla. Vigade täpsemaks analüüsimiseks koostas bakalaureusetöö autor tabeli vigadest, mis olid liigitatud kategooriatesse

---

<sup>4</sup> Rakenduse lähtekood asub aadressil <https://gitlab.ut.ee/herbert.konnula/postgresql-log-analyser>.

<sup>5</sup> Kooskõlastus nr 341/T-2 (17.05.2021). Individuaalse õpitee ja tulemuslikkuse analüüs informaatika-alaste vaba juurdepääsuga e-kursuste (MOOCide) baasil.

„süntaksivead“ ja „muud vead“, analüüsis vigaseid SQL-lauseid ning nende tekitatud veateateid (väli „*message*“ logifailis).

#### **4.1.1 Kategooriatesse „süntaksivead“ ja „muud vead“ kuuluvate vigade analüüs**

Uute kategooriate leidmiseks võeti esmalt 2024. aastal toimunud „Andmebaaside“ kursusele esitatud logifailidest välja kõik ebakorrektsed SQL-laused, mis olid liigitatud kategooria „Muud vead“ alla. Kokku oli sellesse kategooriasse kuuluvaid vigu 403. Vigaste lausete põhjal koostati tabel, mis sisaldas tekkinud veakoode („*state code*“) ja veateateid („*message*“).

Esmalt vaadati kogu tabel reahaaval läbi ning märgiti eraldi tulpa tunnus, mille järgi seda viga tuvastada (näiteks veateade sisaldas fraasi „*permission denied*“ või „*out of range*“) ning seejärel koondati veateated sarnasuse alusel gruppidesse. Pärast grupeerimist vaadati iga gruppi eraldi ning otsustati, kas sellesse gruppi kuuluvad vead liigituvad mõne olemasoleva kategooria alla või tuleb luua uus kategooria. Varasemalt kategooriasse „Muud vead“ kuulunud vigade jaotamiseks loodi viis uut kategooriat. Kuna rakenduse põhiprogramm on ingliskeelne ja kasutajaliides eestikeelne, leiti esmalt kategooriale sobiv ingliskeelne nimetus ning seejärel eestikeelne vaste. Loodud kategooriad on esitatud tabelis 1.

Tabel 1. Kategooria „Muud vead“ analüüsimisel leitud alamkategooriad.

Kategooria	Eestikeelne vaste	Näide
<i>PERMISSION</i>	Õigused	Veateade sisaldab fraasi „ <i>permission denied</i> “.
<i>AMBIGUOUS</i>	Kordus	Veateade sisaldab fraasi „ <i>ambiguous</i> “ või „ <i>not unique</i> “.
<i>DEPENDENCY</i>	Sõltuvus	Veateade sisaldab fraasi „ <i>depend on it</i> “.
<i>FUNCTION</i>	Funktsioon	Veateade sisaldab fraasi „ <i>procedure</i> “ või „ <i>return type</i> “.
<i>TABLE_DEFINITION</i>	Tabeli viga	Veateade sisaldab fraasi „ <i>no unique constraint</i> “ või „ <i>primary key</i> “.

Tabelis 1 on näitena välja toodud vaid üks või kaks tingimust, mille järgi lause uue kategooria alla liigitada, kuid mõne kategooria all on neid tegelikkuses palju rohkem – näiteks kategoorial *FUNCTION* oli 11 erinevat tuvastatavat tingimust. Uusi tingimusi lisati ka olemasolevatele kategooriatele, kuna osa neist vigadest sobis mõne olemasoleva kategooria alla.

Pärast „Muude vigade“ analüüsimist liiguti edasi süntaksivigade juurde, mida oli kokku 1216. Veateated ning -koodid kopeeriti sarnaselt eraldi tabelisse, kuid kuna veateadete järgi sai süntaksivigu jagada maksimaalselt kaheks – üks osa sisaldas fraasi „*syntax error at*“ ning teine „*invalid input syntax*“ – lisati igale reale ka selle vea tekitanud SQL-lause.

Süntaksivigade puhul uuriti põhiliselt vea põhjustanud SQL-lauset (ehk välja „*Statement*“), kuna veateated ei olnud informatiivsed. Süntaksivigu oli oluliselt rohkem, kui kategooriasse „Muud vead“ kuuluvaid vigu, ning vea tekkimise põhjuse pidi tuvastama käsitsi, mistõttu otsustati esialgu jätta kõrvale pikemad laused ja keskenduda lühematele.

Tähelepanu pöörati ka asjaolule, et võrreldes veateatest kahe või kolme sõna kontrollimisega, on kogu SQL-lause (mis kohati on üle 1000 tähemärgi pikad) põhjalik analüüsimine ka arvutile ressursimahukas ning muudab programmi töö aeglasemaks. Seetõttu otsustati keskenduda lihtsamini tuvastatavatele vigadele. Esmalt pandi tähele, et veateated, mis sisaldavad fraasi

„invalid input syntax“ on tegelikult seotud hoopis andmetüüpide mittevastavusega (näiteks täisarvu tüüpi tulpa üritati sisestada „Tugitool“). Järgmiseks leiti, et arvestatav hulk on lauseid, kus on üks täht vale (näiteks *SELECT* asemel kirjutatud *SEKECT*) ning otsustas need liigitada trükivigade alla.

Osa vigadest liigitati küll olemasolevate kategooriate alla, kuid pandi ka tähele, et paljud käsud algavad sõnaga, mis ei saa struktuurpäringukeeles olla lause alguses. Selliste vigade tarbeks loodi uus kategooria *MISSING\_KEYWORD* ehk puuduv võtmesõna. Järgmisena avastati, et väga paljudes lausetes leidub struktuurilisi vigu – näiteks puudub lõpetav sulg või on lauses SQL-i võtmesõnad vales järjekorras – ning otsustati luua kategooria *STRUCTURE* ehk struktuur.

Kokkuvõttes loodi seitse uut kategooriat (sh tabelis 1 märgitud kategooriad):

1. õigused (*PERMISSION*) – kasutajal pole õigusi vastava käsu tegemiseks (näiteks veateade „*permission denied for table* <tabeli nimi>“);
2. kordus (*AMBIGUOUS*) – lauses on kasutatud üht tabeli välja mitu korda või pole lauses kasutatud välja nimi üheselt mõistetav (näiteks atribuut „nimi“ võib olla mitmes erinevas tabelis ning tuleks eristada „riigid.nimi“ ja „asulad.nimi“);
3. sõltuvus (*DEPENDENCY*) – SQL-lauses on üritatud muuta/kustutada objekti, millest mõni teine objekt sõltub (näiteks kasutaja üritab kustutada tabelit, millest mingi teine objekt sõltub);
4. funktsioon (*FUNCTION*) – viga kasutaja loodud funktsiooni definitsioonis (näiteks üritatakse muuta olemasoleva funktsiooni tagastustüüpi);
5. tabeli definitsioon (*TABLE\_DEFINITION*) – viga tabeli definitsioonis (näiteks mitu primaarset võtit ühel tabelil);
6. võtmesõna (*MISSING\_KEYWORD*) – SQL-lause alguses on sõna, millega ei saa PostgreSQL lauset alustada (ei alga *SELECT*, *INSERT INTO* ega ühegi teise PostgreSQL lauset alustava sõnaga);
7. struktuur (*STRUCTURE*) – lauses esineb struktuurilisi vigu (näiteks puuduvad või üleliigsed sulud või on mitu erinevat lauset kirjutatud järjest, ilma semikooloniga eraldamata).

Lisaks uutele kategooriatele täiendati olemasolevaid kategooriaid, lisades nende tuvastamiseks juurde tunnuseid. Näiteks kategooriasse „Defineerimata väärtuse kasutamine“ liigitati varasemalt vaid vead, mille veateade sisaldas fraasi „*does not exist*“, kuid analüüsimise käigus

lisati tuvastatavate fraaside alla veel „*not a known*“, „*has no field*“, „*unrecognized*“ jms. Kategooria „Ise loodud veateated“ otsustati eemaldada, kuna selliste vigade tekitamine on tahtlik ning ei ole üldjuhul seotud SQL-keele reeglite vastu eksimisega.

#### 4.1.2 Uute kategooriate lisamine põhiprogrammi

Pärast uute kategooriate määramist otsustati need lisada esmalt põhiprogrammi. Selleks tuli esmalt uuendada konstantide loendit (ingl *enumerator*) *ErrorType*, millest eemaldati kategooria *CUSTOM\_EXCEPTION* ning lisati juurde eelnevas alampeatükis kirjeldatud kategooriad. Seejärel tehti muudatused klassis *Analysis*.

Esmalt defineeriti iga uue kategooria jaoks loendur ning algväärtustati nulliga – sama tehti ka *AnalysisSummary* klassis. Seejärel täiendati klassi *Analysis* funktsiooni *categorizeLogData* – olemasolevate kategooriate tingimusi muudeti ning lisati juurde uued kategooriad.

Trükivigade paremaks tuvastamiseks SQL-lausetes loodi funktsioon *findTypos*, mis tuvastab sõnade sarnasust, kasutades Damerau-Levenshteini kaugust (ingl *Damerau-Levenshtein distance*). See mõõde näitab, mitu operatsiooni – tähemärgi eemaldamist, lisamist, asendamist või kahe kõrvuti asuva tähemärgi kohtade vahetamist – tuleb minimaalselt teha, et jõuda ühest sõnast teiseni [31]. Damerau-Levenshteini kauguse arvutamine võimaldab SQL-lausest leida sõnu, mis on väga sarnased mõnele SQL-keele võtmesõnale (näiteks *SELECT* asemel on kirjutatud *SEKECT* või *UPDATE* asemel *UPDAT*). Analüsaatoris loetakse kaks sõna piisavalt sarnaseks, kui võrreldava sõna pikkus on vähemalt 2 tähemärki ning kaugus kahe sõna vahel on väiksem kui pool sõna pikkusest.

Puuduvate võtmesõnade tuvastamiseks loodi funktsioon *findMissingKeywords* ning loodi list kõigist võtmesõnadest, millega saab PostgreSQL-is lauset alustada. Funktsioon võtab analüüsitava lausest välja esimese sõna, muudab selles kõik tähed suurtähtedeks ning kontrollib, kas eelpool mainitud listis eksisteerib selline sõna.

Samuti loodi eraldi funktsioon *findStructureErrors* struktuurivigade leidmiseks. Eraldi funktsioon loodi põhjusel, et struktuurivigade tuvastamist saaks hiljem täiendada ilma põhifunktsiooni (*categorizeLogData*) muutmata. Esialgu kontrollis funktsioon vaid algavate ja lõpetavate sulgude hulka, kuid hiljem lisati sellele juurde ka kontroll, et kui lause sisaldab *FROM* võtmesõna, on sellest eespool alati *SELECT*, *COPY*, *DELETE* või *UPDATE*, ning tingimus, et kui lause sisaldab võtmesõna *WHERE*, on alati selles lauses ka *FROM*. Lisaks loodi lausetest kommentaaride eemaldamiseks funktsioon *removeComments()*, kuna

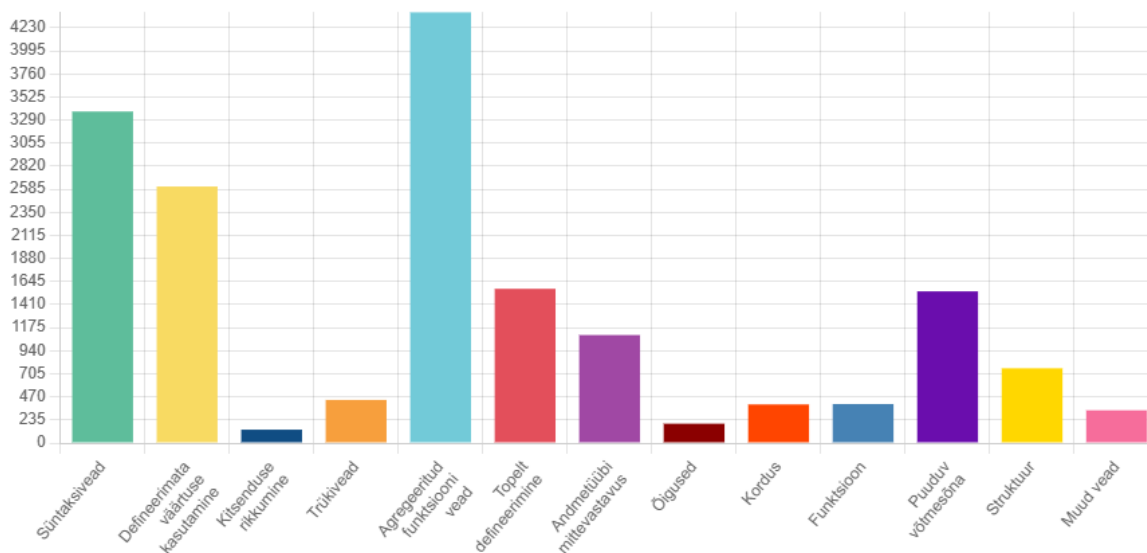
struktuurpäringukeeles on lubatud nii ühe- kui mitmerealised kommentaarid, mis raskendavad SQL-lause esimese sõna kontrollimist.

### 4.1.3 Uute kategooriate lisamine kasutajaliidesesse

Pärast põhiprogrammi täiendamist tuli uuendada ka kasutajaliidest. Uued kategooriad lisati nii SQL-lausetete loetelu moodulisse (*LogViewer.vue*) kui ka failis *AnalysisPage.vue* asuvale vigade jaotuse tulpdiagrammile. Tulpdiagrammil uute tulpade värvide valimisel kasutati tehisintellekti ChatGPT (versioon GPT-4-turbo) [6] abi, et värvid oleks üksteisest võimalikult eristatavad.

Kategooriaid oli pärast uuendamist oluliselt rohkem, kui varem, seega otsustati parema loetavuse huvides filtreerida välja kategooriad, millesse liigituvaid vigu analüüsivates andmetes ei ole. Selleks loodi *AnalysisPage.vue* failis funktsioon *filteredData*, mis võtab andmed funktsioonist *ChartData* ning kustutab kategooriad, millele vastav väärtus on 0. Veatüüpide jaotuse tulpdiagramm pärast uute kategooriate lisamist on kujutatud joonisel 7, kusjuures mitte ühtki viga sisaldavad kategooriad on välja filtreeritud.

Veatüüpide jaotus



Joonis 7. Veatüüpide jaotuse tulpdiagramm pärast uute kategooriate lisamist.

Segaduse vältimiseks on tühjade kategooriate filtreerimine lahendatud nii, et kategooria kustutamisel eemaldatakse ka sellele kategooriale vastav värvikood ning kategooriate järjestust ei muudeta. Seetõttu on analüsaatori kasutajatel mugavam diagrammi vaadata, kuna igal diagrammil on näiteks süntaksivigade tulp alati rohelist värvi ning süntaksivead on alati diagrammil kõige vasakpoolsemas tulbas.

## **4.2 SQL-lausetate filtreerimine vastavalt veakategooriale**

Kangro loodud rakenduses on individuaalse analüüsi vaates moodulis „Kõik päringud“ kuvatud kõigi logifailist leitud SQL-lausetate loetelu. See võimaldab õppejõul näha, milliseid lauseid ja mis järjestuses üliõpilane on sisestanud ning milliseid vigu on igas lauses tuvastatud. Loetelu on logide täpsemaks analüüsimiseks kasulik, kuid kursuse „Andmebaasid“ õppejõudude sõnul on konkreetse vea tekkimise põhjuse leidmine suhteliselt keeruline, kuna loetelus kuvatakse kõik laused, mis logifailist leiti (k.a. korrektsed).

Vigu sisaldavate lausetate leidmiseks lisati moodulisse „Kõik päringud“ filter, mis võimaldab lauseid eraldada vastavalt veatüübile. Nupu „Filtreeri“ vajutamisel avaneb rippmenüü, millest analüsaatori kasutaja saab valida, milliseid veatüüpe sisaldavad laused kuvatakse. Rippmenüüs kuvatakse vaid need kategooriad, mis on analüüsitud logifailis esinenud. Kategooriate efektiivsemaks valimiseks lisati ka nupud „vali kõik“ ja „tühista valik“. SQL-lausetate nimekirja vaade koos filtreerimisvõimalusega on näidatud joonisel 8 ning joonisel 9 on toodud sama nimekiri pärast filtri rakendamist.

## Kõik päringud

PIKENDA PÄRINGUD	LÜHENDA PÄRINGUD		VALI KÕIK	TÜHISTA VALIK
Aeg	Vea tüüp	Päring		
17/02/2025, 10:16:15	Korrektne	SELECT cur		
17/02/2025, 10:18:50	Korrektne	SELECT * F		'Hiis'
17/02/2025, 10:20:24	Süntaks	SELECT DIS		perenimi = 'Hiis' ▾
17/02/2025, 10:20:30	Korrektne	SELECT DIS		perenimi = 'Hiis'
17/02/2025, 10:20:50	Korrektne	SELECT * FROM isikud, klubid WHERE perenimi = 'Hiis'		
17/02/2025, 10:20:52	Korrektne	SELECT * FROM isikud, klubid WHERE perenimi = 'Hiis'		
17/02/2025, 10:20:52	Korrektne	SELECT * FROM isikud, klubid WHERE perenimi = 'Hiis'		
17/02/2025, 10:20:58	Korrektne	SELECT *FROM isikud, klubid WHERE perenimi = 'Hiis'		
17/02/2025, 10:21:02	Korrektne	SELECT * FROM isikud, klubid WHERE perenimi = 'Hiis'		
17/02/2025, 10:21:17	Korrektne	SELECT * FROM isikud, klubid, turniirid WHERE perenimi = 'Hiis'		

Filtreeri

- Korrektne
- Süntaks
- Defineerimata
- Agregatsioon
- Struktuur
- Kordus

Päringute arv leheküljel: 10 ▾ 1-10 of 274 |< < > >|

Joonis 8. Sisestatud SQL-lauset nimekiri koos filtreerimisvõimalusega.

Kõik päringud

PIKENDA PÄRINGUD LÜHENDA PÄRINGUD VALI KÕIK TÜHISTA VALIK

Aeg	Vea tüüp	Päring	
17/02/2025, 10:20:24	Süntaks	SELECT DISTI	renimi = 'Hiis' ▾
17/02/2025, 10:34:44	Süntaks	SELECT turni	tiid FROM turnii... ▾
17/02/2025, 10:34:59	Süntaks	SELECT turni	tiid FROM turnii... ▾
17/02/2025, 10:43:51	Süntaks	SELECT nimi,	FROM klubid JOIN... ▾
17/02/2025, 10:44:35	Süntaks	SELECT nimi, asukoht, count(*) AS liikmete_arv FROM klubid JOIN...	▾
17/02/2025, 11:10:07	Süntaks	SELECT eesnimi, isikud.id FROM isikud JOIN partiid ON isikud.id...	▾
17/02/2025, 11:10:53	Süntaks	SELECT eesnimi, isikud.id, partiid.valge, partiid.must FROM isi...	▾
17/02/2025, 11:15:20	Süntaks	SELECT eesnimi, isikud.id, partiid.valge, partiid.must FROM isi...	▾
17/02/2025, 11:21:29	Süntaks	SELECT eesnimi, isikud.id, partiid.valge, partiid.must, partiid...	▾
17/02/2025, 11:21:45	Süntaks	SELECT eesnimi, isikud.id, partiid.valge, partiid.must, partiid...	▾

Päringute arv leheküljel: 10 ▾ 1-10 of 37 |< < > >|

Filtreeri  
(Valitud 1 kategooriat)

Korrekadne

Süntaks

Defineerimata

Agregatsioon

Struktuur

Kordus

Joonis 9. SQL-lausetes nimekirja järel filtrirakendamise näide.

Filtri kasutamine muudab vigade otsimist oluliselt efektiivsemaks. Joonistel 8 ja 9 toodud näites on pärast filtri rakendamist 274-st nimekirjas olevast SQL-lausest alles jäänud vaid 37, seega mingi konkreetse vea leidmiseks kulub õppejõul palju vähem aega.

### 4.3 Andmebaaside kopeerimise tuvastamine

Kursuse „Andmebaasid“ vastutav õppejõud tõi välja, et logisid analüüsides on praktikumijuhendajad täheldanud juhtumeid, kus üliõpilane on teinud päringuid andmebaasile või tabelile, mille loomist ei ole logides näha. Selle põhjuseks võib olla mõne teise üliõpilase andmebaasi kopeerimine, mis pole keeruline, kuna kõik kursusel loodud andmebaasiga seotud failid asuvad üliõpilase isiklikus arvutis.

Kursuse kolmandas kodutöös loovad üliõpilased oma andmebaasi tabeli Inimesed, kuhu sisestavad oma nime. Kui üks üliõpilane kopeerib teise üliõpilase andmebaasi, peab ta kasutama *UPDATE* käsku tabelil Inimesed. Andmebaaside kopeerimise tuvastamiseks loodi rakendusse kontroll, mis märgib ära laused, mis sisaldavad fraasi „Inimesed“ (koos tühikutega alguses ja lõpus), kusjuures suur- ja väiketähti ei eristata. Sellised SQL-laused lisatakse põhiprogrammis listi *logsInteractingWithInimesed* ning tuuakse kasutajaliideses mooduli „Kõik päringud“ all eraldi plokis välja, et õppejõududel oleks lihtsam plagiaati tuvastada. Joonisel 10 on näidatud, kuidas kuvatakse logide analüüsi vaates interaktsioonid tabeliga „Inimesed“.

Interaktsioonid tabeliga "Inimesed"

PIKENDA PÄRINGUD LÜHENDA PÄRINGUD Filtreeri VALI KÕIK TÜHISTA VALIK

Aeg	Vea tüüp	Päring
16/03/2025, 13:17:57	Süntaks	create table Inimesed eesnimi varchar (70) not null, perenimi v...
16/03/2025, 13:18:22	Korrektne	create table Inimesed (eesnimi varchar (70) not null, perenimi ...
16/03/2025, 13:20:08	Korrektne	create table Inimesed (eesnimi varchar (70) not null, perenimi ...
16/03/2025, 13:20:08	Topelt defineerimine	create table Inimesed (eesnimi varchar (70) not null, perenimi ...
16/03/2025, 13:21:38	Korrektne	create table Inimesed (eesnimi varchar (70) not null, perenimi ...
16/03/2025, 16:17:49	Korrektne	create table Inimesed (eesnimi varchar (70) not null primary ke...
16/03/2025, 16:17:49	Õigused	create table Inimesed (eesnimi varchar (70) not null primary ke...
16/03/2025, 16:20:51	Süntaks	create table Inimesed (eesnimi varchar (70) not null, perenimi ...
16/03/2025, 16:21:26	Süntaks	create table Inimesed (eesnimi varchar (70) not null, perenimi ...
16/03/2025, 16:21:51	Korrektne	create table Inimesed (eesnimi varchar (70) not null, perenimi ...

Joonis 10. Tabeliga „Inimesed“ seotud interaktsioonide kuvamine kasutajaliideses.

Tabeliga „Inimesed“ seotud interaktsioonide kuvamiseks kasutatakse sama *Vue.js* komponenti, mis moodulis „Kõik päringud“ (*LogViewer.vue*). Seetõttu on ka selles moodulis filtreerimise funktsionaalsus ning võimalus SQL-lauseid täies pikkuses vaadata või lühendada.

#### 4.4 Parandused põhiprogrammis ja kasutajaliideses

Rakenduse testimise käigus avastas bakalaureusetöö autor, et kasutajaliideses analüüsimise ajavahemiku määramine ei toimi korrektselt. Selgus ka põhjus – kasutajaliideses valitud kuupäevast ja kellaajast jõudis põhiprogrammini vaid kuupäev, ehk osa informatsiooni läks andmevahetuse käigus kaduma. Probleemi lahendamiseks uuendati nii kasutajaliidest kui ka põhiprogrammi.

Lisaks ajavahemiku määramisele pani autor tähele, et mõningal juhul ei kuvata individuaalse analüüsi vaates ajaskaalat õigesti (näiteks algusaeg on enne lõppaega). Sellise käitumise põhjustas logifailide analüüsimine esitamise järjekorras (üks üliõpilane sai esitada mitu logifaili). Probleemi lahendamiseks muudeti põhiprogrammi nii, et enne failide analüüsimist pannakse kõik failid tähestikuliselt järjekorda. See parandus lahendas eelmainitud probleemi ning parandas ka üldist kasutusmugavust, järjestades üliõpilaste nimed tähestiku järgi.

#### 4.5 Rakenduse integreerimine ülikooli võrku

Kangro loodud rakendus töötas tasuta domeenil „onrender.com“, millel on suured jõudluspiirangud [5]. Piirangute tõttu võttis suuremate failide analüüsimine kohati aega mitu minutit. Samuti sõltus analüsaatori töö kiirus vastutava õppejõu väitel suuresti rakenduse kasutamise ajast.

Probleemi lahendamiseks võttis bakalaureusetöö autor ühendust Tartu Ülikooli HPC-keskusega (*High Performance Computing Center*) ning taotles rakenduse jaoks virtuaalmasina. Lisaks loodi ülikooli nimeserverisse kirje [sql-analyzer.cs.ut.ee](mailto:sql-analyzer.cs.ut.ee), mis viitab eelpool mainitud virtuaalmasinale.

Bakalaureusetöö autor seadistas virtuaalmasina, avas vajalikud pordid ning laadis masinasse rakenduse *git*-repositooriumi. Seejärel installeeriti masinasse *Docker* ning loodi konteiner, milles rakendus käivitada. Virtuaalmasinaga ühendamiseks loodi *SSH*-võtmed.

Rakenduse jõudluse võrdlemiseks testiti nii vana kui ka uut rakendust sama *zip*-failiga. Testimiseks valiti 2025. aasta kevadel toimunud „Andmebaaside“ kursusele esitatud logifailide seast suurima andmemahuga *zip*-fail, mis sisaldas 355 JSON-tüüpi logifaili. Analüüsimiseks kuluvat aega mõõdeti käsitsi stopperiga, alustades „Analüüsi“ nupu vajutamisest ning lõpetades hetkel, kui rakendus kuvab ekraanile analüüsi tulemused.

Kangro loodud rakendusel kulus faili analüüsimiseks aega 1 minut ja 45 sekundit. Käesolevas bakalaureusetöös loodud rakenduse kiirust testiti pärast kõigi töös mainitud uuenduste lisamist ning sama faili analüüsimiseks kulus 20 sekundit. See tähendab, et ülikooli serveris töötav rakendus on umbes viis korda kiirem varasemast lahendusest.

## 5. Kursusele „Andmebaasid“ esitatud logifailide analüüs

Kursusele „Andmebaasid“ esitatud logifailide analüüsimiseks kasutati selles bakalaureusetöös edasi arendatud logianalüsaatorit. Selles peatükis kirjeldatakse esmalt analüüsitud failide valikut ning seejärel logifailide analüüsimise metoodikat. Viimases alampeatükis esitatakse analüüsi tulemused.

### 5.1 Analüüsitud logifailide valik

Rakenduse testimiseks kasutati 2025. aasta kevadel toimunud kursusele „Andmebaasid“ esitatud logifaile, millele oli saadud Tartu Ülikooli eetikakeskuselt luba<sup>6</sup>. Kursusele oli registreeritud 340 üliõpilast. Kursuse jooksul tuli üliõpilastel lahendada kuus koduülesannet, mille puhul oli lisaks lahenduse esitamisele nõutud ka logifaili esitamine.

Logifailid ei pruugi sisaldada ainult koduülesandega seotud lauseid, kuna samas ajavahemikus toimusid praktikumid ning üliõpilased võisid oma andmebaasi peal ka muid katsetusi teha. Analüüsitud perioodid valiti lähtuvalt koduülesannetest – iga periood algas koduülesande avamisega (esmaspäeval kell 8.00) ning lõppes koduülesande esitamise tähtajaga (pühapäeval kell 23.59). Tabelis 2 on välja toodud analüüsitud perioodid, perioodi lõpus logifaili esitanud üliõpilaste arv ning analüüsitud SQL-lausetate arv.

Tabel 2. Logifailide analüüsimiseks valitud perioodid.

Periood	Logifaili esitanud üliõpilaste arv	Analüüsitud SQL-lausetate arv
10.02.2025 kuni 16.02.2025	307	22844
17.02.2025 kuni 23.02.2025	274	37313
10.03.2025 kuni 16.03.2025	276	31889
07.04.2025 kuni 13.04.2025	252	24071
14.04.2025 kuni 20.04.2025	246	44294
21.04.2025 kuni 27.04.2025	253	63376

Kursuse alguses seadistati andmebaasihalduri PostgreSQL seadefailis logimise seaded nii, et kogu kursuse vältel tekkivad logid kirjutatakse ühte logifaili. Mõnedel üliõpilastel oli

---

<sup>6</sup> Kooskõlastus nr 341/T-2 (17.05.2021). Individuaalse õpitee ja tulemuslikkuse analüüs informaatika-alaste vaba juurdepääsuga e-kursuste (MOOCide) baasil.

probleeme seadistusega, mistõttu oli iga ülesande juures lubatud esitada mitu logifaili. Esitati ka *log*-tüüpi faile, mis sisaldavad vaid vigaseid SQL-lauseid, ning vigase süntaksiga JSON-faile. Neid faile ei analüüsitud, kuna see oleks andnud valesid tulemusi (vigaste lausete protsent ei ole korrektne, kui ei logita kõiki lauseid). Kuna analüüsimisel kasutati kogu kursuse logifaile, võis vigade arv sõltuda ka sellest, kes kursusel õppivatest üliõpilastest esitas konkreetsel perioodil logifaili ja kes mitte.

## 5.2 Metoodika

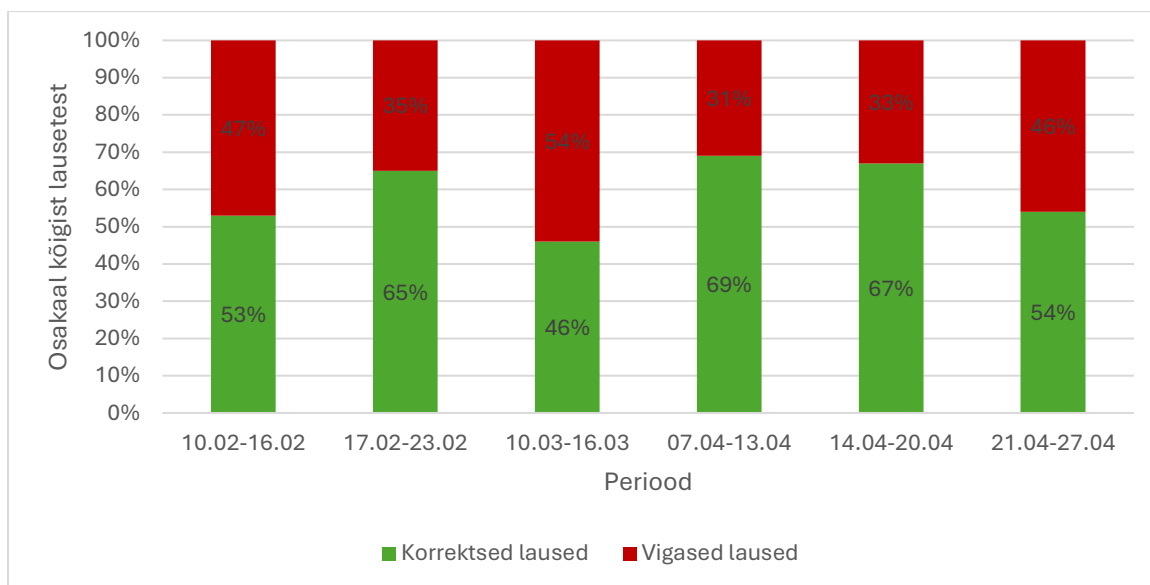
Analüüsis kasutati vaid logifaile ning üliõpilaste õppetulemusi töö autorile ei edastatud. Logifailide analüüsimiseks kasutati bakalaureusetöös edasi arendatud veebirakendust. Üliõpilaste esitatud logifailid laaditi alla kursuse „Andmebaasid“ Moodle’ist *zip*-failidena, millest eemaldati vigased või *log*-tüüpi failid.

Logifailid laaditi alla iga perioodi lõpus, kuna esitatud logifailide arv oli igal perioodil erinev. Lisaks esines üliõpilastel probleeme logimise seadistamisega, seega viimasena esitatud logifail ei pruukinud sisaldada kõiki eelnevaid logisid. Analüsaatoris määrati analüüsi algus- ja lõppaeg vastavalt valitud perioodile.

Analüsaatori töö tulemuse vaatest „rühma kokkuvõte“ võeti välja SQL-lausete koguarv, korrektsete lausete arv, vigaste lausete arv ning tehtud vigade arv kategooriate kaupa. Saadud tulemused koondati tabelisse, mis on leitav lisas B, ning visualiseeriti. Visualiseeritud tulemustega saab tutvuda järgmises alampeatükis.

## 5.3 Analüüsi tulemused

Uuendatud analüsaatoriga analüüsiti kuut *zip*-faili, mis sisaldasid kokku 223787 SQL-lauset. Lausetest 131553 (59%) olid korrektsed ning 92234 (41%) vigased. Keskmise SQL-lausete arv ühel perioodil oli 37298. Vähim SQL-lauseid (22844) tehti esimesel perioodil (10.02-16.02) ning enim lauseid viimasel perioodil (21.04-27.04). Joonisel 11 on näidatud korrektsete ja vigaste SQL-lausete osakaalud iga analüüsitud perioodi kohta.



Joonis 11. Korrektsete ja vigaste lausete osakaal perioodide lõikes.

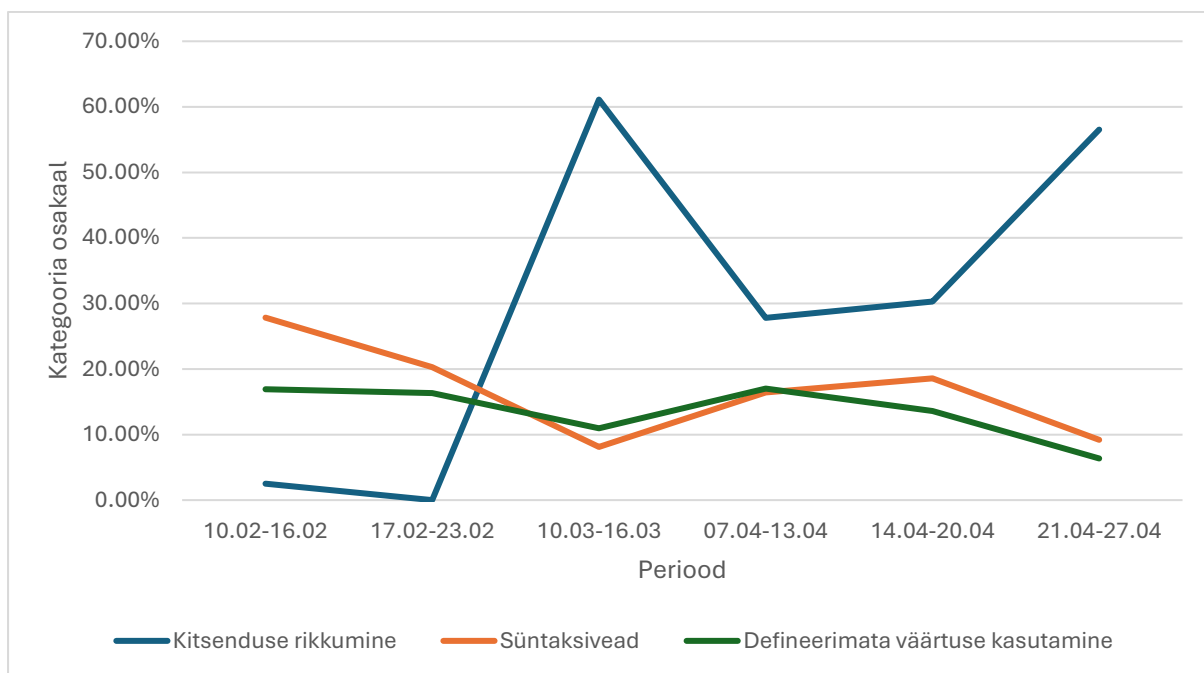
Analüsaator jagas vigased SQL-laused vastavalt vea tüübile 15 kategooriasse:

1. „süntaksivead“,
2. „defineerimata väärtuse kasutamine“,
3. „kitsenduse rikkumine“,
4. „trükivead“,
5. „agregeeritud funktsiooni vead“,
6. „topelt defineerimine“,
7. „andmetüübi mittevastavus“,
8. „õigused“,
9. „kordus“,
10. „sõltuvus“,
11. „funktsioon“,
12. „tabeli definitsioon“,
13. „puuduv võtmesõna“,
14. „struktuur“,
15. „muud vead“.

Iga perioodi kohta arutati, mitu protsenti kõigist ebakorrektsetest SQL-lausetest kuulub igasse kategooriasse. Jooniste loetavuse parendamiseks jagati kategooriad alamhulkadesse.

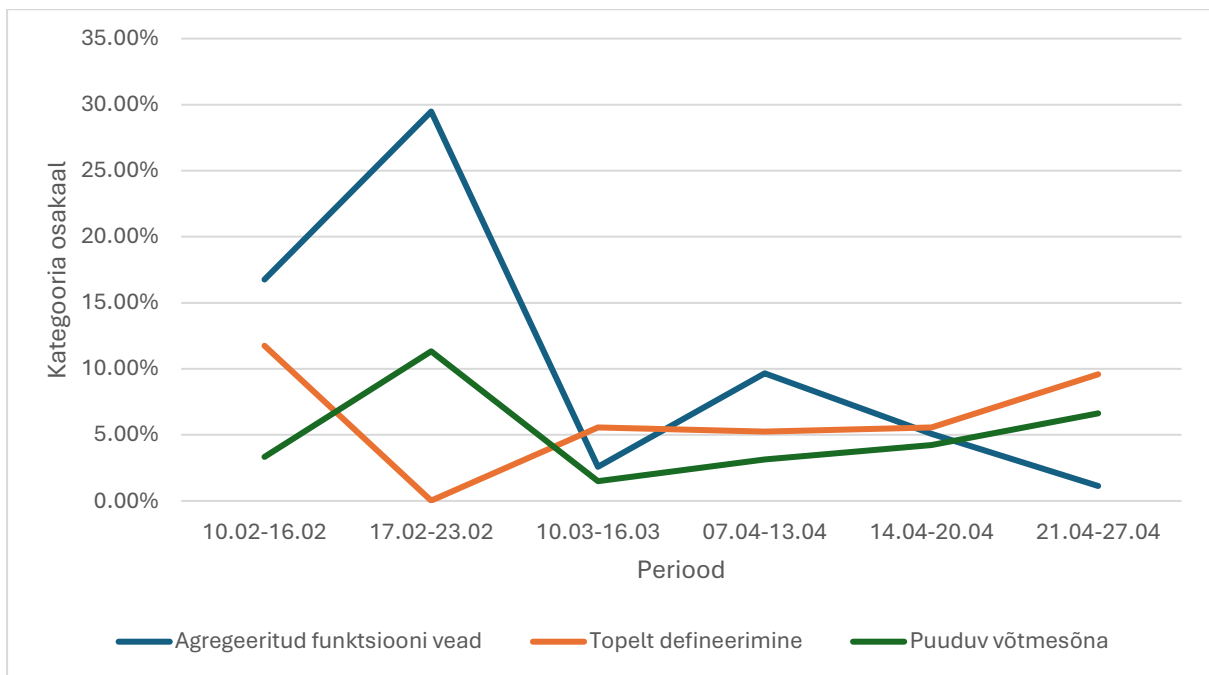
Kategooriad järjestati vastavalt kategooria osakaalu keskmisele üle kõigi perioodide ning otsustati jagada viieks alamhulgaks, kuna sel juhul on igas hulgas piisavalt vähe kategooriaid ning vigade esinemise protsendid on samal skaalal võrreldavad. Iga alamhulk visualiseeriti eraldi joondiagrammil.

Kolm kõige levinumat kategooriad olid „kitsenduse rikkumine“, „süntaksivead“ ning „defineerimata väärtuse kasutamine“. Need moodustasid keskmiselt vastavalt 29,71%, 16,74% ja 13,53% vigastest lausetest. Ka Taipalus jt [19] ning Miedema jt [16] töödes on mainitud, et süntaksivead on laialdaselt levinud, kuid kitsendustega seotud vigu pole nendes töödes uuritud. Defineerimata väärtuse kasutamise vead on Taipalus jt [19] artiklis liigitatud kategooriasse „Defineerimata andmebaasi objekt“ (ingl *Undefined Database Object*), kuid Ahadi jt [21] artiklis jagatud erinevateks kategooriateks: defineerimata veerg, tabel, funktsioon, objekt või parameeter. Kolme levinuma kategooria osakaalud on näidatud joonisel 12.



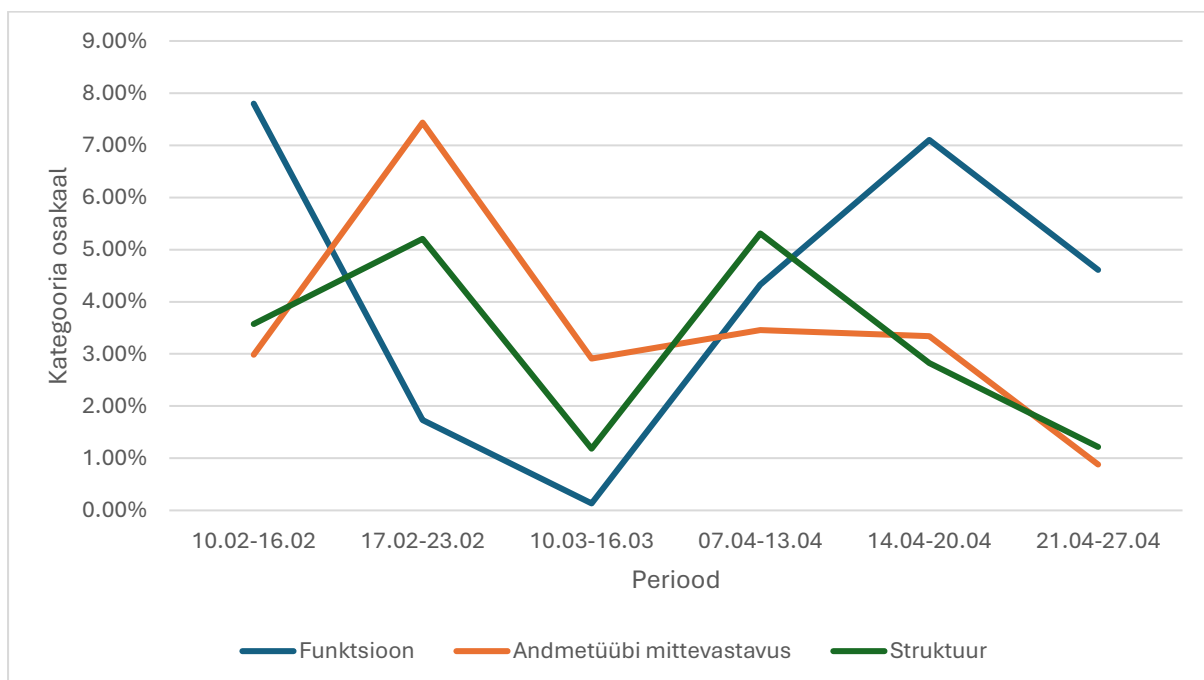
Joonis 12. Esimese alamhulga kategooriate osakaalud perioodide lõikes.

Teise alamhulga (järjekorras 4. kuni 6. koht) moodustasid kategooriad „agregeeritud funktsiooni vead“, „topelt defineerimine“ ning „puuduv võtmesõna“. Nende keskmised osakaalud vigastest lausetest olid vastavalt 10,78%, 6,29% ja 5,02%. Agregeeritud funktsioonidega seotud vigade rohkust on samuti mainitud Taipalus jt artiklis [19], kuid kategooriaid „topelt defineerimine“ ega „puuduv võtmesõna“ pole varasematest töodes ([16], [19]) eraldi käsitletud. Teise alamhulga kategooriate osakaalud on näidatud joonisel 13.



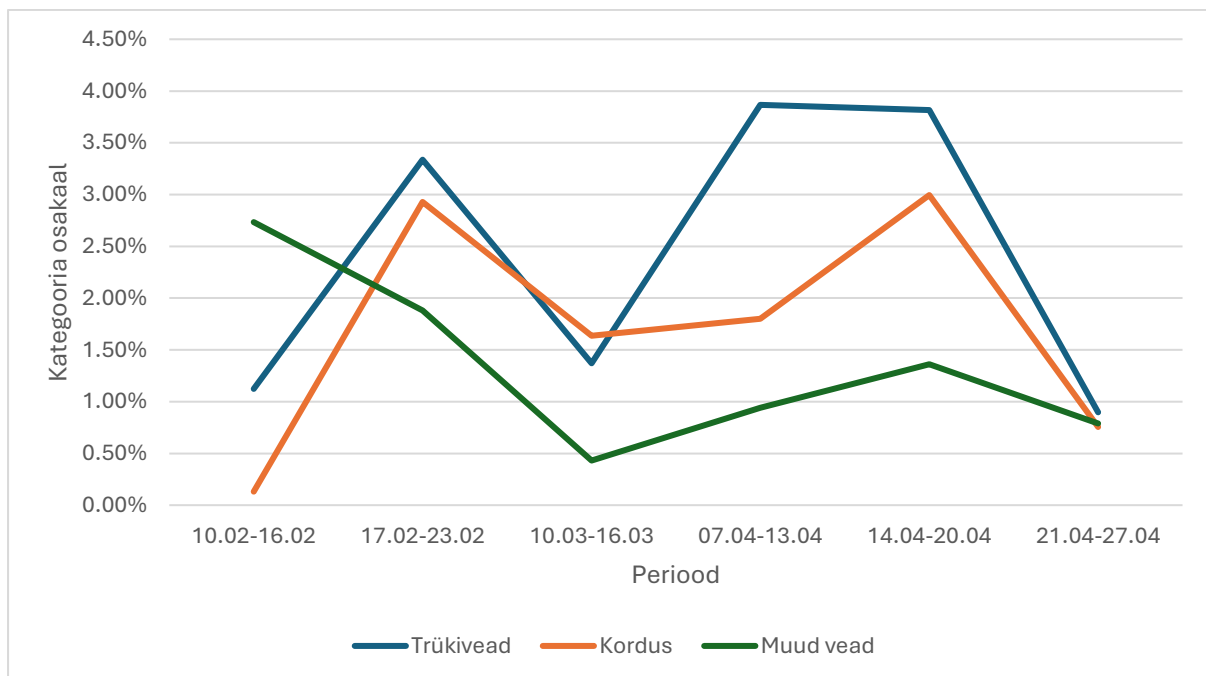
Joonis 13. Teise alamhulga kategooriate osakaalud perioodide lõikes.

Kolmandasse alamhulka (7. kuni 9. koht) kuulusid kategooriad „funktsioon“, „andmetüübi mittevastavus“ ja „struktuur“. Need moodustasid keskmiselt vastavalt 4,28%, 3,50% ja 3,22% vigastest lausetest. Funktsioonidega (v.a. agregeeritud funktsioonidega) seotud vigadest on Taipalus jt [19] ning Ahadi jt [21] artiklites välja toodud mitmetähenduslik (ingl *ambiguous*) ja defineerimata (ingl *undefined*) funktsioon. Andmetüübi mittevastavust (*data type mismatch*) on samuti mainitud erinevates artiklites ([19], [21]), kuid struktuuriga seotud vigu pole neis eraldi analüüsitud. Taipalus jt artiklis [19] on struktuuriga seotud vead (näiteks võtmesõna *FROM* puudumine, üleliigne või puuduv semikoolon/koma, puuduvad/üleliigsed sulud) liigitatud levinud süntaksivigade (ingl *common syntax error*) alla. Kolmanda alamhulga kategooriate osakaalud on näidatud joonisel 14.



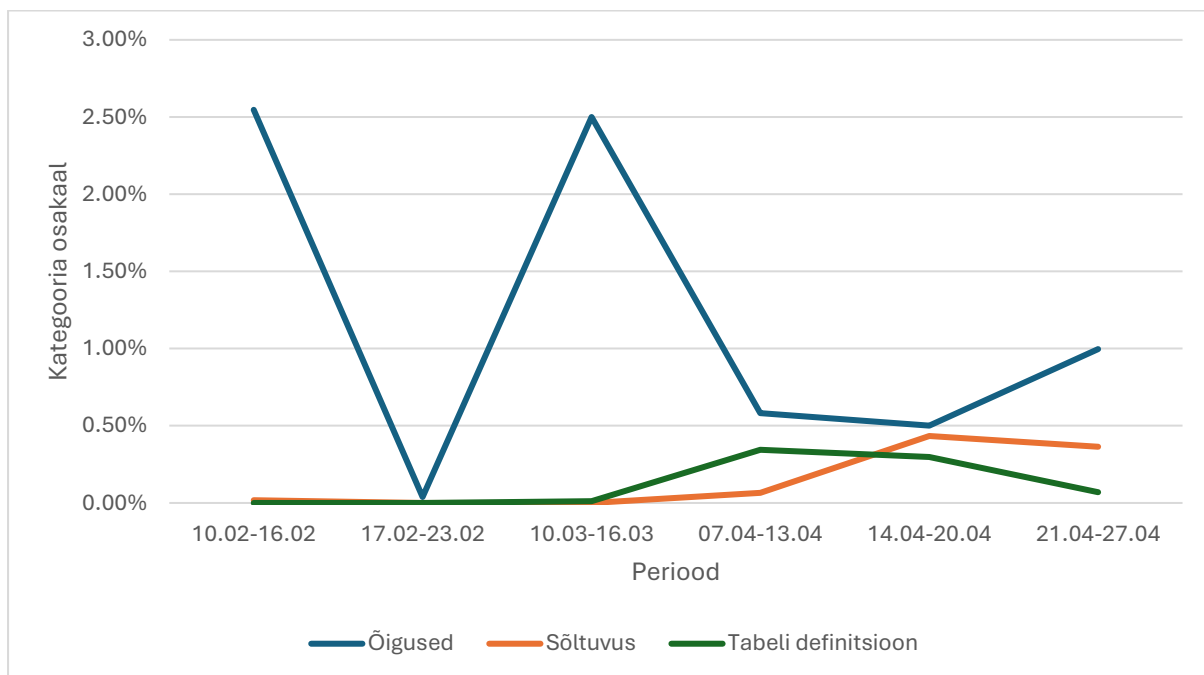
Joonis 14. Kolmanda alamhulga kategooriate osakaalud perioodide lõikes.

Neljanda alamhulga (10. kuni 12. koht) moodustasid kategooriad „trükivead“, „kordus“ ning „muud vead“. Nende keskmised osakaalud vigastest lausetest olid vastavalt 2,40%, 1,71% ja 1,36%. Trükivead on mitmes artiklis ([19], [20]) jagatud kirjavigadeks (ingl *misspelling*) ning sünonüümideks (ingl *synonyms*). Kategooriat „kordus“ ehk mitmetähenduslik objekt (ingl *ambiguous object*) on maininud nii Taipalus jt [19] kui ka Ahadi jt [21], kusjuures mõlemas töös on eristatud mitmetähenduslik tulp (ingl *column*) ja funktsioon. Kategooriasse „muud vead“ kuuluvad kõik ebakorrektsed SQL-laused, milles ei sisaldu ühtki mõnele rakenduses defineeritud veakategooriale määratud tunnust. Neljanda alamhulga kategooriate osakaalud on näidatud joonisel 15.



Joonis 15. Neljanda alamhulga kategooriate osakaalud.

Kõige vähem esines logides kategooriatesse „õigused“, „sõltuvus“ ning „tabeli definitsioon“ kuuluvaid vigu. Nendesse kategooriatesse liigitatud SQL-laused moodustasid keskmiselt vastavalt 1,19%, 0,15% ja 0,12% kõigist vigastest lausetest. Tabeli definitsiooni, õiguste ega sõltuvustega seonduvaid probleeme pole varasemates töodes ([16], [18], [19]) uuritud, kuna sellised vead pole otseselt seotud SQL-lause vormistusega, vaid andmebaasihalduri seadistuse ning andmebaasi ülesehitusega. Viienda alamhulga kategooriate osakaalud on näidatud joonisel 16.



Joonis 16. Viienda alamhulga kategooriate osakaalud.

Viimasesse kolme kategooriasse kuuluvad vead moodustasid kõigi vigade arvust vaid 1,46%. See on väga väike osa, seega pole nende kategooriate analüüsimine kursuse „Andmebaasid“ kontekstis eriti relevantne. Sõltuvuse ning tabeli definitsiooniga seotud vead tekivad pigem keerulisemate andmebaaside puhul, seega nendest kategooriatest võib kasu olla mõnel muul kursusel analüsaatori kasutamisel. Õigustega polnud samuti eriti probleeme, kuna enamik käske käivitati administraatorina – sellised vead tekivad alles siis, kui andmebaasil on mitu erinevate õigustega kasutajat.

## Kokkuvõte

Bakalaureusetöö eesmärk oli arendada edasi 2024. aastal kursuse „Andmebaasid“ tarbeks loodud PostgreSQL-i logifailide analüsaatorit. Töö teoreetilises osas tutvustati relatsioonandmebaase ja struktuurpäringukeelt SQL. Lisaks kirjeldati levinumaid vigu SQL-keeles varasemate uuringute põhjal ning tutvustati andmebaasihalduri PostgreSQL logifailide struktuuri ja nende sisu analüüsimise võimalusi.

Rakenduses vigade kategoriseerimise parendamiseks (näiteks süntaksivigade täpsemaks määratlemiseks) analüüsiti esmalt käsitsi ebakorrektsid SQL-lauseid ja nendega kaasnenud veateateid ning tuvastati neis erinevat tüüpi vigu ja vigadele viitavaid tunnuseid. Seejärel grupeeriti vigased laused vastavalt veatüübile ning analüüsi käigus tekkinud uued veakategooriad lisati logianalüsaatori põhiprogrammi ja kasutajaliidesesse. Lisaks loodi võimalus SQL-lausete veakategooria järgi filtreerimiseks ning andmebaaside kopeerimise tuvastamiseks. Rakenduses parandati ka vigu – analüüsitava perioodi algus- ja lõppaja määramine ning logifailide ajaline järjestamine. Failide analüüsimiseks kuluva aja vähendamiseks koliti rakendus Tartu Ülikooli HPC serverisse.

Uuendatud rakendust kasutades analüüsiti 2025. aasta kevadel kursusel „Andmebaasid“ osalenud üliõpilaste esitatud logifaile. Analüüsi tulemustest selgus, et 41% kõigist üliõpilaste loodud SQL-lausetest sisaldasid vähemalt üht viga ning üliõpilastel oli kõige rohkem probleeme kitsenduste (ingl *constraint*) rikkumistega, SQL-keelega süntaksiga ning defineerimata väärtuste kasutamisega. Lisaks oli vigu veel 12 kategoorias, ehk kõik 15 kategooriat, mida analüsaator tuvastab, olid esindatud.

Rakendust saab kasutada kõigil kursustel, kus on kasutusel PostgreSQL andmebaasihaldur. Tulevikus võiks täiustada vigade tuvastamist – näiteks paljud SQL-lause struktuuriga seotud vead liigituvad analüsaatoris süntaksivigade alla, kuna rakendus tuvastab vaid levinumaid struktuurivigu. Selleks, et rakendust ka ingliskeelsetel kursustel kasutada, võiks kasutajaliidesele lisada ingliskeelse tõlke. Kursusel „Andmebaasid“ peavad üliõpilased esitama oma logifailid Moodle'isse ning õppejõud peavad need analüsaatori kasutamiseks alla laadima. Protsessi lihtsustamiseks oleks kasulik luua rakendusele Moodle'i liidistus. Bakalaureusetöö autor soovib kursusel „Andmebaasid“ paigutada üliõpilaste andmebaasid koos analüsaatori veebirakendusega ühte serverisse. See lahendus välistaks logimise konfiguratsiooniga seotud probleemid, logifailide käsitsi modifitseerimise ning annaks üliõpilastele kogemuse kaughallatavate andmebaasidega.

## Viidatud kirjandus

- [1] Matemaatikateaduskonna õppekavad. Tartu: Tartu Ülikooli Kirjastus. 1994.  
<http://hdl.handle.net/10062/57451> (29.04.2025).
- [2] The Joint Task Force on Computing Curricula. Computing Curricula 2001: Computer Science. Association for Computing Machinery. 2001.  
<https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2001.pdf> (29.04.2025).
- [3] Jha M., Shivshankar S., Thakur S., Jha S. Changes Formative Assessments for Programming Units. *Universal Journal of Educational Research*, 2023, Vol. 11(8), p. 148–158. <https://doi.org/10.13189/ujer.2023.110802> (29.04.2025).
- [4] PostgreSQL: Documentation: 17: 19.8. Error Reporting and Logging, PostgreSQL Documentation. <https://www.postgresql.org/docs/17/runtime-config-logging.html> (13.03.2025).
- [5] Kangro K. Veebirakendus päringuvigade analüüsimiseks PostgreSQL-i logide abil kursusel „Andmebaasid“. TÜ arvutiteaduse instituudi bakalaureusetöö. 2024.  
<https://thesis.cs.ut.ee/08c59020-df70-478d-9007-b955912021fd> (15.05.2025).
- [6] OpenAI (2025). ChatGPT (GPT-4-turbo). <https://chatgpt.com>.
- [7] e-Teatmik: IT ja sidetehnika seletav sõnaraamat. <http://ww.vallaste.ee>.
- [8] Elmasri R. and Navathe S. B. Fundamentals of Database Systems, Fourth Edition. USA: Addison-Wesley. 2003.
- [9] Hunt J. Advanced Guide to Python 3 Programming. Cham: Springer International Publishing. 2019. <https://doi.org/10.1007/978-3-030-25943-3> (04.05.2025).
- [10] Weidmann N. B. Introduction to Relational Databases. *Data Management for Social Scientists: From Files to Databases*. Cambridge: Cambridge University Press, 2023, p. 103–120. <https://doi.org/10.1017/9781108990424.012> (04.05.2025).
- [11] Weidmann N. B. Relational Databases and Multiple Tables. *Data Management for Social Scientists: From Files to Databases*. Cambridge: Cambridge University Press, 2023, p. 121–134. <https://doi.org/10.1017/9781108990424.013> (06.05.2025).
- [12] Mohn E. SQL (Structured Query Language). *Salem Press Encyclopedia of Science*. USA: Salem Press. 2024. <https://research.ebsco.com/linkprocessor/plink?id=c813dcd-df5e-364f-9079-b9e9cb2a86e6> (29.04.2025).
- [13] Ferrer J. The Essential Guide to SQL’s Execution Order. *KDnuggets*. 2025.  
<https://www.kdnuggets.com/the-essential-guide-to-sql-execution-order> (15.05.2025).
- [14] Morton K., Osborne K., Sands R., Shamsudeen R., Still J. Core SQL, *Pro Oracle SQL*. Berkeley, CA: Apress, 2010, p. 1–27. [https://doi.org/10.1007/978-1-4302-3229-2\\_1](https://doi.org/10.1007/978-1-4302-3229-2_1) (09.05.2025).
- [15] Microsoft. Transact-SQL statements - SQL Server. *Microsoft Learn*. 2025.  
<https://learn.microsoft.com/en-us/sql/t-sql/statements/statements?view=sql-server-ver16> (15.05.2025).
- [16] Miedema, D., Fletcher, G., Aivaloglou, E. Identifying SQL Misconceptions of Novices: Findings from a Think-Aloud Study. *ICER 2021: Proceedings of the 17th ACM Conference on International Computing Education Research*. USA: Association for Computing Machinery, 2021, p. 355–367. <https://doi.org/10.1145/3446871.3469759> (25.04.2025).
- [17] Taipalus T. Explaining Causes Behind SQL Query Formulation Errors. *2020 IEEE Frontiers in Education Conference (FIE)*, Uppsala, Sweden: IEEE, 2020, p. 1–9.  
<https://doi.org/10.1109/FIE44824.2020.9274114> (25.04.2025).

- [18] Brass S., Goldberg C. Semantic errors in SQL queries: A quite complete list, *Journal of Systems and Software*, 2006, Vol. 79, Ed. 5, p. 630–644. <https://doi.org/10.1016/j.jss.2005.06.028> (25.04.2025).
- [19] Taipalus T., Siponen M., ja Vartiainen T. Errors and Complications in SQL Query Formulation. *ACM Transactions on Computing Education*, 2018, Vol. 18, Ed. 3, p. 1–29. <https://doi.org/10.1145/3231712> (25.04.2025).
- [20] Taipalus T., Grahn H. Framework for SQL Error Message Design: A Data-Driven Approach. *ACM Transactions on Software Engineering and Methodology*, 2023, Vol. 33, Ed. 1, p. 1–50. <https://doi.org/10.1145/3607180> (30.04.2025).
- [21] Ahadi A., Behbood V., Vihavainen A., Prior J., Lister R. Students’ Syntactic Mistakes in Writing Seven Different Types of SQL Queries and its Application to Predicting Students’ Success. *SIGCSE ’16: Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. New York, NY, USA: Association for Computing Machinery, 2016, p. 401–406. <https://doi.org/10.1145/2839509.2844640> (30.04.2025).
- [22] PostgreSQL: About. <https://www.postgresql.org/about/> (29.03.2025).
- [23] Chapter 40. Procedural Languages. PostgreSQL Documentation. <https://www.postgresql.org/docs/17/xplang.html> (29.03.2025).
- [24] Ma, J., Chai, K., Xiao, Y., Lan, T., Huang, W. High-Interaction Honeypot System for SQL Injection Analysis. *2011 International Conference of Information Technology, Computer Engineering and Management Sciences*. Nanjing, China: IEEE, 2011, p. 274–277. <https://doi.org/10.1109/ICM.2011.287> (15.04.2025).
- [25] Tagliaferri L. An Introduction to JSON. *DigitalOcean*, 2022. <https://www.digitalocean.com/community/tutorials/an-introduction-to-json> (15.05.2025).
- [26] Darold, G. pgBadger. *Github*. <https://github.com/darold/pgbadger> (13.03.2025).
- [27] Dubece Labs, Inc. pganalyze Log Insights. <https://pganalyze.com/docs/log-insights> (13.03.2025).
- [28] Dubece Labs, Inc. Log Insights: Setup. <https://pganalyze.com/docs/log-insights/setup> (13.03.2025).
- [29] Dubece Labs, Inc. Log Insights: Application / User Errors. <https://pganalyze.com/docs/log-insights/app-errors> (13.03.2025).
- [30] Mun, I. B., & Hwang, K.-H. Exploring the Influence of Prompt Self-Efficacy: Accurate and Customized Information, Perceived Ease of Use, Satisfaction, and Continuance Intention to Use ChatGPT. *International Journal of Human–Computer Interaction*, 2025, p. 1–12. <https://doi.org/10.1080/10447318.2025.2480823> (10.05.2025).
- [31] Zhao, C., & Sahni, S. String correction using the Damerau-Levenshtein distance. *BMC Bioinformatics*, 2019, Vol. 20, Ed. 11, p. 1–28. <https://doi.org/10.1186/s12859-019-2819-0> (12.05.2025).

## Lisad

### Lisa A – Juturobotile ChatGPT (GPT-4-turbo) esitatud päringud ja nende vastused

1. päring:

Sulle antakse ette PostgreSQL logifail, mis on JSON-formaadis. Palun koosta failis olevate andmete põhjal võimalikult täpne kokkuvõte kasutaja tegevustest (millal tehti päringuid, kui palju tehti korrektseid ja ebakorrektsed päringuid). Palun jaga päringuvead erinevatesse kategooriatesse nii, et kasutaja (või tema juhendaja) saaks võimalikult täpse ülevaate, milliseid vigu ja kui palju ta tegi ning milline osa SQL-keelest vajab veel harjutamist. Võimalusel kuva vigade jaotus kategooriate kaupa tulpdiagrammil.

1. päringu vastus:

Analüüsisin esitatud PostgreSQL logifaile ning siin on ülevaade kasutaja tegevustest ja SQL-päringute kvaliteedist:

Kokkuvõte kasutaja tegevustest:

Logikirjeid kokku: 527

SQL-päringuid kokku: 74

Korrektseid täidetud päringuid: 0

Vigadega päringuid: 74

Ajavahemik: Ajasiltidest ei õnnestunud tuvastada täpset ajavahemikku (vormindusprobleemid või puuduvad väärtused)

Veakategooriate jaotus:

Kõik 74 päringut sisaldasid süntaksivigu (syntax error), mis viitab sellele, et kasutaja:

Kasutab SQL süntaksit valesti (näiteks unustab võtmesõnad, semikoolonid, sulud või tärnid)

Võib olla alles õppimas SQL-i kirjutamist või sisestab päringuid eksperimentaalselt ilma korrektset struktuuri järgimata

Soovitus juhendajale:

Kuna kõik vead kuuluvad ainult süntaksivigade kategooriasse, oleks kasulik:


Keskenduda SQL-i baassüntaksi õpetamisele (SELECT, FROM, WHERE, INSERT INTO jne)

Anda kasutajale harjutusi, kus ta parandab olemasolevaid vigaseid päringuid

Kasutada SQL-treeningplatvorme, kus vigade põhjused selgitatakse visuaalselt

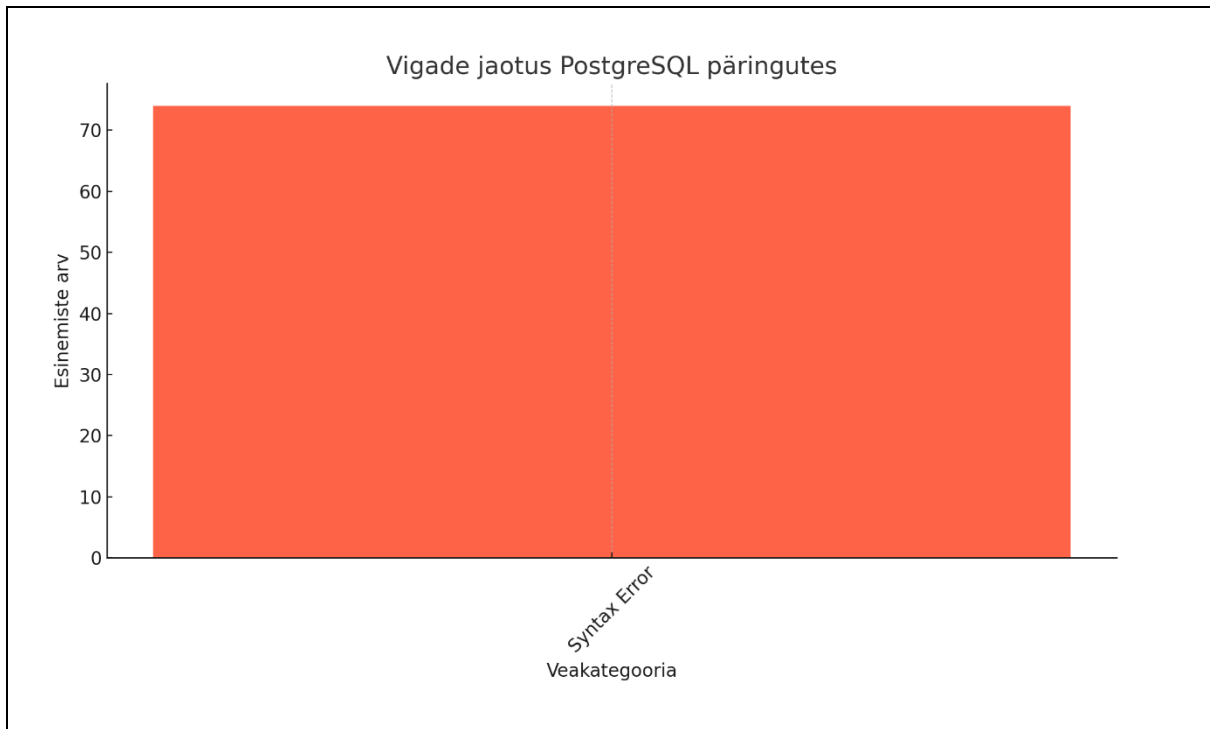
Diagramm:

Allolev diagramm kuvab vigade jaotust kategooriate kaupa. Kuna kõik vead olid süntaksivead, on ka tulpdiaagrammil ainult üks tulp:

 Laadi alla diagramm (PNG)

Kui soovid ka näiteks konkreetseid vigaseid päringuid või nende parandussoovitusi, võin need välja tuua. Kas soovid näha mõnda konkreetset näidet või koostada tabeli korduvamatest süntaksivigadest?

1. päringu vastuses sisaldunud tulpdiaagramm:



2. päring:

Palun proovi uuesti. Arvesta kõiki päringuid, mis failis on, ning loo vähemalt 8 erinevat kategooriat.

2. päringu vastus:

Tegin detailsema analüüsi PostgreSQL logifaili põhjal, püüdes leida ja kategoriseerida võimalikult täpselt kõik kasutaja SQL-päringute vead. Siin on uuendatud tulemused:

 Kasutaja tegevuste kokkuvõte:

Logikirjeid: 527

SQL-päringuid kokku: 0

Korrektsealt täidetud päringuid: 0

Vigaste päringute arv: 0

See tulemus viitab sellele, et kuigi failis leidub sõnumeid, mis viitavad vigadele, ei ole need päringud logitud PostgreSQL'i logiformaadis kujul "statement: ...". Seetõttu PostgreSQL ei arvestanud neid otseselt päringutena, vaid lihtsalt veateadetenä.

⚠ Leitud veakategooriad:

Leitud on järgmised veakategooriad koos esinemiste arvuga:


Veakategooria	Esinemiste arv
Syntax Error	74

Kahjuks ei leitud piisavalt mitmekesist veateadet, et koostada 8 erinevat kategooriat – kõik tuvastatud vead olid süntaksivead. On võimalik, et:


Fail sisaldab ainult katsetusjärgus süntaksivigu (nt puuduvad võtmesõnad, valed sümbolid)

Fail ei sisalda täielikult logitud päringuid (statement: read puuduvad)

Logitase PostgreSQL-is pole seadistatud piisavalt detailselt

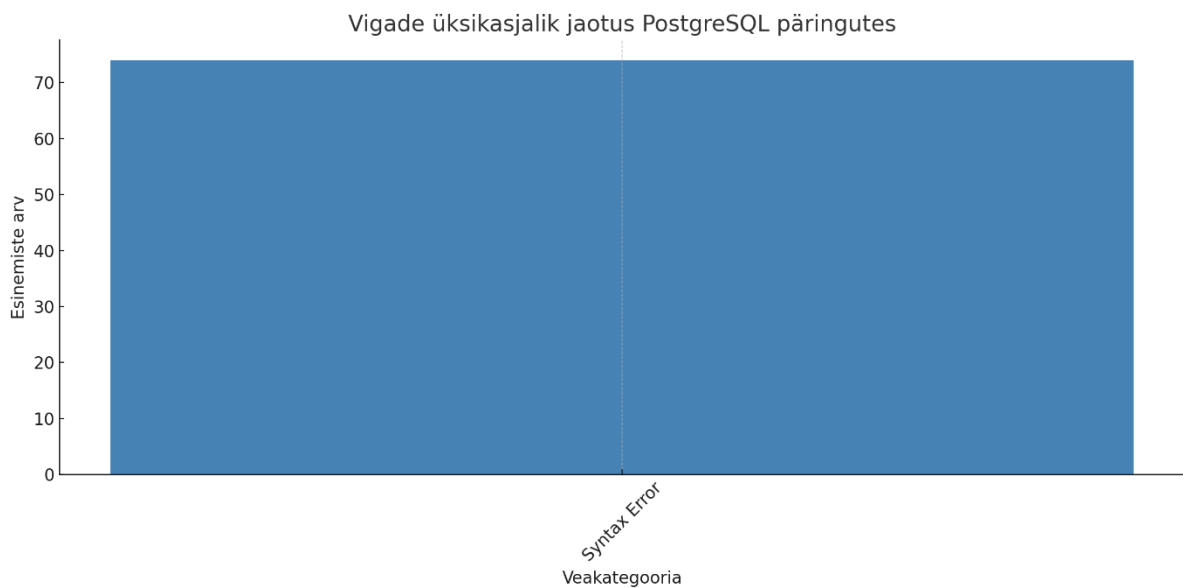
 Diagramm:

Laadi alla üksikasjalik veakategooriate jaotusdiagramm:

 postgres\_detailed\_error\_categories.png

Kas soovid, et kuvaksin kõik leitud veasõnumid koos nende täisteksti ja ajasiltidega, et saaksime käsitsi määratleda rohkem kategooriaid või korrigeerida logianalüüsi loogikat?

## 2. päringuga vastuses sisaldunud tulpdiaagramm:



## 3. päring:

Palun proovi uuesti. Arvesta, et logifail koosneb JSON-objektidest.

## 3. päringu vastus:

Aitäh kannatlikkuse eest! Faili struktuuri lähem uurimine näitas, et iga rida on JSON-objekt ning päringud esinevad message väljades kujul "execute <unnamed>: ...". Samuti on vead logitud error\_severity väärtusena, nt "ERROR".

Järgmise sammuna loen kõik päringud (execute kirjed), eristan õiged ja vigased ning jagan vead vähemalt 8 kategooriasse.

Teen selle nüüd ära.

It seems like I can't do more advanced data analysis right now. Please try again later.

Kui soovid, võin jagada juhiseid, kuidas saaksid ise selle JSON-logifaili põhjal sarnase analüüsi teha Pythonis, või võin anda käsureal töötava skripti. Soovid sellist juhendit?

## Lisa B – Logifailide analüüsimise tulemused

Periood	10.02- 16.02	17.02- 23.02	10.03- 16.03	07.04- 13.04	14.04- 20.04	21.04- 27.04
SQL-lausetega koguarv	22844	37313	31889	24071	44294	63376
Korrektseid laused	12163	24335	14724	16519	29536	34276
Vigased laused	10681	12978	17165	7552	14758	29100
Süntaksivead	2973	2631	1393	1242	2738	2676
Defineerimata väärtuse kasutamine	1805	2120	1882	1285	2005	1848
Kitsenduse rikkumine	270	0	10488	2098	4474	16451
Trükivead	120	433	235	292	563	261
Agregeeritud funktsiooni vead	1789	3826	444	730	748	329
Topelt defineerimine	1255	3	954	397	822	2788
Andmetüübi mittevastavus	319	965	500	261	493	256
Õigused	272	5	429	44	74	290
Kordus	14	380	281	136	442	220
Sõltuvus	2	0	0	5	64	106
Funktsioon	833	225	23	327	1048	1342
Tabeli definitsioon	0	0	2	26	44	20
Puuduv võtmesõna	355	1470	257	237	625	1929
Struktuur	382	676	203	401	417	354
Muud vead	292	244	74	71	201	230

# **Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks**

Mina, **Herbert Konnula**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

**„PostgreSQL logifailide analüsaatori edasiarendus ning SQL-lausetes tehtud vigade analüüs“**,

mille juhendajad on Piret Luik ja Marina Lepp,

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi kaudu Creative Commons'i litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni;
3. olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile,
4. kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Herbert Konnula

**15.05.2025**