TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karmo Jürgenson 213623IABB

Tabelite elimineerimise teisenduse rakendamine erinevate SQL-andmebaasisüsteemide poolt ja selle muutused ajas

Bakalaureusetöö

Juhendaja: Erki Eessaar
PhD

Tallinn 2024

Autorideklaratsioon

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

Autor: Karmo Jürgenson

20.05.2024

Annotatsioon

Lõputöö eesmärgiks oli uurida mõnede kõige populaarsemate SQL-andmebaasisüsteemide poolt tabelite elimineerimise teisenduse tegemist ja andmebaasisüsteemides toimuvate operatsioonide keskmiseid töökiiruseid ning proovitakse leida seoseid töökiiruse ja andmemahtude vahel. Lisaks uuriti kuidas on muutunud kolme andmebaasisüsteemi võimekus teostada tabelite elimineerimise teisendust võrreldes kümme aastat tagasi toimunud uuringuga.

Töö sisaldab tööprotsesside ja töös kasutatud tööriistade kirjeldust, teemaga seonduvat teoreetilist tausta, eksperimendi kirjeldust, eksperimendi tulemusi ja tulemuste analüüsi.

Töö eesmärgi saavutamiseks testiti kuute andmebaasisüsteemi. Nendeks olid PostgreSQL16, Orace23c Free, MySQL 8.3, MariaDB 11.4, MS SQL Server 2022 Developer ja DB2 11.5. Tabelite elimineerimise teisenduse muutuseid ajas võrreldi tulemustega, mis saadi Oracle12c, PostgreSQL9.3 ja Microsoft SQL Server 2012ga. MySQLi puhul on kirjandusest teada, et see ei oska seda teisendust teha, kuid see süsteem kaasati uuringusse, et võrrelda lausete töökiiruse tulemusi MariaDB-ga.

Kõige rohkem rakendas tabelite elimineerimise teisendust Oracle 23c, mis teostas teisendust 27-st valitud lausest 21-l korral. Teisena järgnes DB2 11.5, mis suutis tabelite elimineerimise teisendust teha 18-l korral. Ülejäänud andmebaasisüsteemid jäid kaugele, suutes teostada tabelite elimineerimise teisendust kõik alla kümne lause korral.

Töökiiruste võrdluses näitas kõige kiiremaid keskmiseid töökiiruseid MS SQL Server 2022, järgnesid DB2 11.5, PostgreSQL16 ja MariaDB 11.4. Andmemahtude ja lausete täitmisaja vahel on tugev kasvav lineaarne seos.

Eksperimentide jaoks kirjutatud kood on leitav GitHubi salvest https://github.com/Karjur/JoinElimination

Töö tulemusena leiab autor, et tabelite elimineerimise teisendus tegemine võib küll kiirendada andmekäitlusoperatsioonide läbiviimist, aga see vahe ei ole alati märgatav.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 49 leheküljel, seitset peatükki, 16 joonist, 28 tabelit.

Abstract

Application of the Table Elimination Transformation by Different SQL Database Management Systems and its Changes Over Time

The aim of the bachelor's thesis was to study the capability of performing table elimination transformation and the average query execution times, while also attempting to find correlations between operation speed and data volumes, of some of the most popular SQL database management systems (DBMSs). In addition, it was investigated how the ability of three DBMSs to perform table elimination has changed compared with a study that was made ten years ago. Table elimination transformation means that a DBMS simplifies an executed SQL statement to omit tables that are not needed to get the desired result.

The thesis contains a description of the work processes and tools used, theoretical background related to the topic, description of the experiment, experiment results, and analysis of the results.

To achieve the goal of the thesis, six DBMSs were tested in the experiment. These were PostgreSQL16, Orace23c Free, MySQL 8.3, MariaDB 11.4, MS SQL Server 2022 Developer, and DB2 11.5. Changes in the ability to perform table elimination transformation were determined by comparing the results with the results of a previous study that used Oracle12c, PostgreSQL9.3, and Microsoft SQL Server 2012. In case of MySQL it was known from the literature that it does not perform the transformation. However, the system was included to compare its query speed with MariaDB. One of the experiments involved making queries based on a view of a database that was created by using anchor modelling.

Oracle 23c applied table elimination transformation the most, performing the transformation in 21 out of 27 selected statements. DB2 11.5 followed with the ability to perform table elimination transformation in 18 cases. The other DBMSs were far behind, being able to perform the table elimination transformation in less than ten statements out of 27.

In case of PostgreSQL and Oracle there were no changes of making the transformation compared with the study that was made ten years ago. However, in case of MS SQL Server there were five statements (out of 13) in case of which MS SQL Server 2012 (the earlier version) made the transformation, but MS SQL Server 2022 did not.

In the comparison of operation speeds based on 19 statements, MS SQL Server 2022 showed the fastest average speeds, followed by DB2 11.5, PostgreSQL16, and MariaDB 11.4. All the DBMSs showed very strong linear relationship between data size and execution time, i.e., increasing the data size increased the execution time.

As a result of the work, the author finds that while performing table elimination transformation can speed up the execution of data handling operations, the difference is not always significant. It is not the silver bullet to speed up database operations. Instead, it is useful in case of querying views that have been defined based on different base tables and is a kind of safety measure to reduce performance penalty of overly complicated SQL statements.

The code that was written for the thesis is in GitHub: https://github.com/Karjur/JoinElimination

Lühendite ja mõistete sõnastik

CSV fail	*Comma-Separated Values* fail
DML	*Data Manipulation Language*
SQL	*Structured Query Language*
SQO	*Semantic Query Optimization*

Sisukord

Jooniste loetelu

Tabelite nimekiri

1 Sissejuhatus

Käesolevas peatükis selgitatakse töö tausta, lahendatavat probleemi, uurimisküsimusi ning töö dokumendi struktuuri.

1.1 Üldine taust

Tänapäeval on loodud kasutamiseks väga palju erinevaid andmebaasisüsteeme ning nende vahel valida võib olla keeruline. Andmebaasikeel SQL on vaatamata oma pikale ajaloole endiselt väga populaarne ning SQLi kasutada võimaldavad ning selle aluseks oleval andmemudelil põhinevad andmebaasisüsteemid on turul domineerivad. 2024. aasta mai seisuga oli kümnest kõige populaarsemast andmebaasisüsteemist seitse SQL-andmebaasisüsteemid [1]. SQL keele standard ei kirjuta ette seda, kuidas peaksid andmebaasisüsteemid sisemiselt SQL lauseid töötlema. Selle võimalikult efektiivsel viisil tegemine võib olla andmebaasisüsteemi üks konkurentsieelistest. Üks võimalik viis kuidas andmebaasisüsteem saab lauset efektiivsemalt täita e oma tööd optimeerida on selle enne täitmist lihtsustamine e asendamine loogiliselt samaväärsete, kuid lihtsamate lausega. Üks lihtsustamise viis on tabelite elimineerimise teisenduse tegemine, mida tänapäeval üritavad paljud SQL-andmebaasisüsteemid rakendada. IEEE Spectrum 2023. aasta programmeerimiskeelte populaarsuse indeksis on SQL tööandjate poolt kõige oodatum keel. Arendajad peaksid lisaks teistele keeltele kindlasti oskama SQLi, et lahti muukida andmetes peituv potentsiaal [2]. Keeles, mida palju kasutatakse, tehakse paratamatult ka palju vigu, sealhulgas ka selliseid, kus kirjutatakse lauseid, mis on keerulisemad kui ülesande lahendamiseks vaja. Andmebaasisüsteemi poolt tabelite elimineerimise teisenduse tegemine aitab selliseid eksimusi siluda. Tabelite elimineerimise teisenduse korral lihtsustab andmebaasisüsteem lauset nii, et ei loe tabeleid (kas otse või läbi indeksite), millele küll lauses viidatakse, kuid mida pole tulemuse saamiseks tegelikult vaja kasutada.

Antud bakalaureusetöö uurib mõnede kõige populaarsemate SQL-andmebaasisüsteemide poolt tabelite elimineerimise teisenduse tegemist.

1.2 Lahendatav probleem

Käesoleva töö eesmärgiks on võrrelda mõningaid populaarseid SQL-andmebaasisüsteeme selles osas, millised neist teostavad tabelite elimineerimise teisendust kõige paremini ning kuidas on tabelite elimineerimise teisenduse tugi ajas muutunud. Veel uuritakse andmebaasisüsteemides toimuvate operatsioonide keskmiseid töökiiruseid ning proovitakse leida seoseid töökiiruse ja andmemahtude vahel. Eesmärgi saavutamiseks tehakse töös erinevaid eksperimente. Tegemist on kvantitatiivse uurimistööga [3], kus vastused uurimisküsimustele leitakse läbi mõõtmiste.

1.3 Uurimisküsimused

Töö otsib vastust järgnevatele uurimisküsimustele:

Küsimused teisenduse tegemise oskuse kohta

1. Kui hästi oskavad valitud andmebaasisüsteemid tabelite elimineerimise teisendust läbi viia? Täpsemalt:
    a. Milline on nende andmebaasisüsteemide pingerida seda oskust silmas pidades?
    b. Kas andmebaasisüsteemid oskavad tabeli elimineerimise teisendust teha ka sellise andmebaasi korral, mis on loodud ankurmodelleerimist kasutades?
    c. Kas andmebaasisüsteemid oskavad tabeli elimineerimise teisendust ka muude lause tüüpide kui SELECT korral?
2. Milline on Oracle, MS SQL Serveri ja PostgreSQL puhul olnud selle võimekuse muutus võrreldes eelmise uuringuga kümme aastat tagasi?
3. Kas see kui tabelid on tühjad *vs.* see, et tabelites on andmeid, mõjutab seda, kas andmebaasisüsteem otsustab tabelite elimineerimise teisendust teha?

Küsimused mõju kohta töökiirusele

4. Milline on operatsioonide töökiiruse erinevus MySQL ja MariaDB andmebaasides kui tehakse operatsioone, mille korral MariaDB teeb teisendust, kuid MySQL ei tee?

5. Kui võrrelda MySQL ja MariaDB andmebaasides tehtud operatsioone, siis kas tabelite elimineerimise tegemine *vs.* mittetegemine mõjutab seda, kas andmemahtude ja töökiiruse vahel on kasvav lineaarne seos või mitte?

6. Kui konkreetse andmebaasisüsteemi puhul lahendatavas ülesandes on kaks lihtsustamata lauset, mis annavad loogiliselt samaväärse tulemuse, kuid ühe korral tehakse andmebaasisüsteemi poolt teisendus ja teise korral mitte, siis:

    a. Kui palju mõjutab teisenduse tegemine lause täitmise kiirust?

    b. Millal muutub teisenduse tegemise mõju töökiirusele märgatavaks?

7. Milline on nende andmebaasisüsteemide töökiiruste pingerida erinevate andmehulkade puhul kõikide operatsioonide keskmist kiirust arvestades? Kas kõikide operatsioonide töökiiruse keskmist arvestades on igas konkreetses süsteemis andmemahtude ja töökiiruse vahel kasvav lineaarne seos või mitte?

1.4 Töö edasine struktuur

Töö koosneb seitsmest peatükist. Teises peatükis kirjeldatakse töö metoodikat. Selgitatakse, millised on kasutatud tööriistad, milline on tööprotsess ja eksperimenteerimisel kasutatud riistvara. Kolmandas peatükis selgitatakse lühidalt mõisteid, mis aitavad käesolevast tööst paremini aru saada. Neljandas peatükis pannakse paika, milliseid eksperimente tehakse uurimisküsimustele vastuste saamiseks. . Viiendas peatükis tuuakse välja eksperimendis saadud tulemused. Kuuendas peatükis analüüsitakse viiendas peatükis näidatud tulemusi ning vastatakse uurimisküsimustele. Lõpuks esitatakse töö kokkuvõte.

2 Metoodika

Antud peatükis antakse ülevaade töö protsessist ning tulemuste saavutamiseks kasutatavatest tööriistadest

2.1 Tööprotsessi kirjeldus

Töö puhul on tegemist uurimistööga, mis laiendab kümme aastat tagasi (2014. aastal) samal teemal tehtud bakalaureusetööd [4]. Eksperimentides kasutatakse tulemuste võrreldavuse huvides samasuguse struktuuriga andmebaasi, mis oli kasutusel eelnevas bakalaureusetöös. Lisaks kolmele eelnevas töös kasutatud SQL-andmebaasisüsteemile (PostgreSQL, Oracle, MS SQL Server) võetakse võrdlusesse veel kolm tükki (IBM DB2 Community Edition, MariaDB Community Edition ja MySQL Community Edition). Kõik need süsteemid kuulusid 2024. aasta mai seisuga kõige populaarsemate andmebaasisüsteemide hulka [1]. Ankurmodelleerimisega seotud uurimisküsimusele vastamiseks luuakse fragment ankurmudeli andmebaasist, mis loodi magistritöös [5].

Loodud andmebaasides katsetatakse erinevaid SQL lauseid, kusjuures korratakse nii eelnevas töös kasutatuid kui lisatakse uusi, mille korral võiks eeldada, et andmebaasisüsteem rakendab tabelite elimineerimise teisendust. Uuritakse nende lausete täitmisplaane, et näha, kas süsteem on lause täitmisplaani optimeerimiseks kasutanud tabelite elimineerimise teisenduse meetodit.

Genereeritakse testandmeid ja tehakse lausete töökiiruse katsetusi erineva tabelite ridade arvu korral, et näha, millise andmehulga puhul muutub tabelite elimineerimise teisendusest tulenev töökiiruse paranemine märgatavaks. Töökiiruse uuringute korral tehakse iga uuritava lause korral kolm mõõtmist ja leitakse nende töökiiruste geomeetriline keskmine, sest nende mõõtmiste tulemused sõltuvad üksteisest, kuna sama töökiiruse lause korduval käivitamisel kasutatakse eelmisest käivitamisest loodud tööplaani ja muutmällu loetud andmeid. Käesolevas töös esitatakse need keskmised, mitte üksikute mõõtmiste tulemused.

2.2 Tööriistade kirjeldus

Töös uuritavad andmebaasisüsteemid ja nende versioonid on järgmised.

1. Oracle 23c Free [6]
2. PostgreSQL16 [7]
3. MS SQL Server 2022 Developer [8]
4. MySQL 8.3 Community Edition [9]
5. MariaDB 11.4 [10]
6. IBM DB2 11.5 Community Edition [11]

Kõik nimetatud andmebaasisüsteemid peale MariaDB kuulusid 2024. aasta mai seisuga andmebaasisüsteemide populaarsuse indeksis esikümnesse. MariaDB oli 13-s. [1] Kõigil andmebaasisüsteemidel peale MySQLi on kirjanduse põhjal tabelite elimineerimise teisenduse tugi olemas. MySQL võeti valikusse, et võrrelda päringute töökiirust MariaDB-ga olukorras, kus esimene ei kasuta ja teine kasutab tabelite elimineerimise teisendust.

Kõik andmebaasisüsteemid, peale Oracle, installeeriti otse. Oracle valitud versiooni puhul ei olnud loodud Windowsile otse installeerimist ja pidi kasutama kas Dockerit või virtuaalmasinat. Autor valis virtuaalmasina võimaluse.

Andmebaasisüsteemiga suhtlemiseks ja andmebaasi kasutamiseks on igal andmebaasisüsteemil enamasti olemas spetsiifilised programmid. Oracle puhul valiti SQL CLI. PostgreSQL-iga suhtlemiseks kasutati pgAdmin 4. MS SQL Serveri puhul valiti SQL Server Management Studio 19. IBM DB2-ga suhtlemiseks kasutatakse Data Studio 4.1.4 Client tarkvara. MySQL ja MariaDB puhul kasutati Beekeeper studio [12] tarkvara. Kui teiste andmebaasisüsteemidega suhtlemiseks kasutati andmebaasisüsteemi-spetsiifilist haldurit, siis Beekeeper studio abil saab kasutada mitmeid erinevaid andmebaasisüsteeme.

Pearsoni korrelatsioonikordaja väärtuste arvutamiseks kasutati veebist leitavat Statistics Kingdomi korrelatsioonikordaja koefitsiendi kalkulaatorit [13].

2.3 Eksperimenteerimisel kasutatud riistvara

Andmebaasisüsteemidel eksperimenteerimiseks kasutati autori sülearvutit Dell Inspiron 7610 11th Gen Intel® Core(TM) i7-11800H @ 2.30GHz. Erandiks oli Oracle23c, mille puhul tehti eksperimenti läbi virtuaalmasina. Oracle ja teiste andmebaasisüsteemide töökiiruste võrdlemisel peab seda arvestama, sest süsteemid millel töökiiruseid mõõdeti, ei ole samad.

3 Teoreetiline taust

Selles peatükis selgitatakse lühidalt mõisteid, mis aitavad käesolevast tööst paremini aru saada.

3.1 Andmekäitluskeele lausete täitmine andmebaasisüsteemi poolt

SQL andmekäitluskeel (DML) käitub kui sild andmebaasisüsteemi kasutaja ja andmebaasisüsteemi vahel. SQL andmekäitluskeel sisaldab INSERT, SELECT, DELETE, UPDATE ja MERGE lauseid. [14] Nende abil saab andmebaasist andmeid otsida ja andmeid muuta. Laused on juhised andmebaasisüsteemidele käskude täitmiseks, aga inimestele lihtsamini mõistetavas keeles. Lause tõlgitakse madalamatesse keeltesse, et andmebaasisüsteem oskaks seda täita. [15] Kasutaja saab nende lausete abil süsteemile öelda, mida ta soovib tulemuseks saada. Selle kohta öeldakse, et kasutaja deklareerib oma soovi ja et andmekäitluskeel on deklaratiivne keel. Kasutaja soovi alusel koostab andmebaasisüsteem protseduuri e täitmisplaani soovitud tulemuseni jõudmiseks.

Esitatud lausest kasutaja soovitud tulemusteni jõudmiseks läbib andmebaasisüsteem mitu olulist sammu [16].

Kompileerimine – SQL lause kompileerimise käigus tõlgitakse lause nii, et andmebaasisüsteem suudab seda paremini töödelda. Otsitakse süntaksivigu ja analüüsitakse lause struktuuri. Selle tulemusel saadakse valmis lause loogiline täitmisplaan, mis sisaldab relatsioonialgebra operatsioone.

Optimeerimine – Selles sammus leitakse optimaalseim algoritm loogilise taseme plaanis määratud operatsioonide läbiviimiseks. Määratakse kindlaks kasutatavad füüsilise taseme operatsioonid ja nende läbiviimise järjekord. Tulemuseks on füüsiline täitmisplaan. Optimeerimise eesmärk on saavutada optimeerimise eesmärki täitev füüsiline täitmisplaan. Optimeerimise eesmärgiks võib näiteks olla kogu lause võimalikult kiire täitmine.

Täitmine – täidetakse koostatud täitmisplaan ja tulemused tagastatakse kasutajale.

3.2 Tabelite elimineerimise teisendus

Tabelite elimineerimise teisendus on SQL-andmebaasisüsteemi poolt täidetava SQL lause andmebaasisüsteemi poolne lihtsustamine, eemaldades lausest viited tabelitele, mida lause täitmiseks pole vaja lugeda [17]. See optimeerimisprotsess on suunatud lausete täitmise tõhususe suurendamisele, sest lihtsamat lauset on loodetavasti kiirem täita [18].

Andmebaasisüsteemide poolt andmekäitluskeele lausete töötlemise käigus tehtav tabelite elimineerimise teisendus on üks SQL lausete täitmise optimeerimise meetod, mis aitab parandada lausete täitmise jõudlust ja tõhusust. Teisenduse mõju on eriti ilmne keerukamate lausete korral, mis hõlmavad mitut tabelit, kuid mitte kõiki neist ei ole vaja lõpptulemuste saavutamiseks lugeda. [18] Lausete jõudluse paranemine aitab parandada lauseid käivitava tarkvarasüsteemi üldist jõudlust.

Tabelite elimineerimise teisendus on üks andmekäitluslausete semantilise optimeerimise (*semantic query optimization*, SQO) meetoditest. Semantilise optimeerimise eesmärgiks on proovida vähendada lausete keerukust ja tagada, et nende täitmisel kasutatakse minimaalset hulka vajalikke andmeid. Sellise optimeerimise käigus asendatakse lause andmebaasisüsteemi poolt loogiliselt samaväärse kuid lihtsama lausega, võttes arvesse seda, mida optimeeriv süsteem “teab” andmebaasi kohta. Tänu andmebaasis deklareeritud kitsendustele teab süsteem seda, millistele reeglitele peavad andmebaasis olevad andmed igal juhul vastama. Eesmärgiks on saavutada efektiivsem andmete kasutamine. [19]

3.3 Ankurmodelleerimine

Ankurmodelleerimine on andmebaasi kavandamise metoodika. Ankurmodelleerimise kasutamise idee lähtub eeldusest, et andmete struktuur on sageli ajas muutlik ja selle eesmärgiks on luua paindlik ning kergesti kohandatav SQL-andmebaas. Kui andmebaasi struktuuri on vaja muuta, siis tabelitesse uute veergude lisamise asemel luuakse uusi tabeleid. Eesmärk on säilitada vana struktuur ning lisada juurde väiksemaid muudatusi. See tähendab, et ankurmodelleerimise alusel loodud andmebaas hõlmab endas alamhulgana eelnevaid andmebaasi versioone. See võimaldab säilitada muudatuste korral ajaloolist infot ning hoida rakendusi töökorras. [20] Ankurmodelleerimise metoodika

kirjeldab ankurmodelleerimise keele, teisenduse selles keeles loodud mudelist SQL-andmebaasi objektide (tabelid, vaated, rutiinid, trigerid) loomiseks ning veebipõhise töövahendi, mis võimaldab nii ankurmudeleid luua kui nendest SQL andmebaasiobjektide loomise koodi genereerida.

Ankurmudelil on neli põhielementi: ankrud, sõlmed, atribuudid ja seosed. Mudelis on ankruteks olemitüübid e reaalse maailma füüsiliste või abstraktsete asjade üldistused. Sõlmede abil modelleeritakse omadusi, mida võivad jagada erinevad ankrud. Sisuliselt on sõlmed klassifikaatorid, mis aitavad liigitada ankrutele vastavaid andmeid. Atribuudid vastavad ankrute huvipakkuvatele omadustele. Seoste eesmärk on näidata seoseid kahe või enama ankru vahel. Kui kujutleda ankurmudelit graafina, siis ankrud, sõlmed ja atribuudid on graafi sõlmed ja nendevahelised seosed on graafi servad. [21]

Ankurmodelleerimise metoodikat kasutades tekib olukord, kus andmebaasis luuakse palju erinevaid tabeleid, milles olevaid andmeid on vaja kasutajatele suunatud vaadetes kokku ühendada. Tabelite elimineerimise teisendus võib olla sellistes olukordades väärtuslik, sest see aitab vaadete põhjal tehtud päringuid optimeerida, eemaldades lausete täitmisel ebavajalike tabelite kasutamise.[21] Ankurmodelleerimist kasutatakse eeskätt andmeaitade ja -vakkade loomiseks, kus on tüüpiliselt suured andmehulgad ja tehakse koondandmete päringuid.

3.4 Olemasolevad uuringud tabelite elimineerimise teisenduse kohta

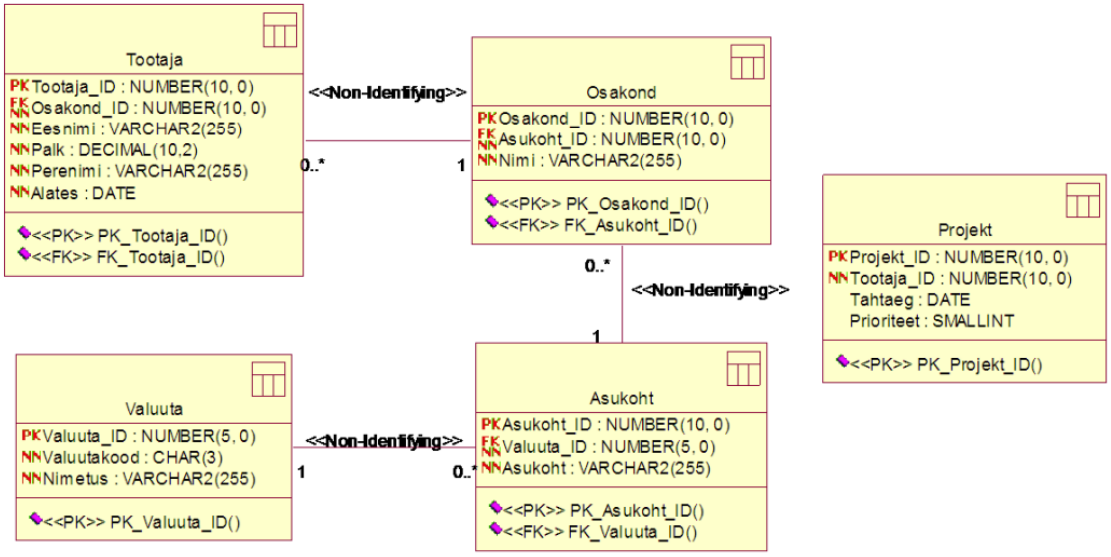
Tabelite elimineerimise kohta ei ole väga palju uuringuid tehtud. 2004. aastal tehtud uuring käsitles OUTER JOIN operaatorit kasutava ühendamise puhul tabelite elimineerimise teisenduse ja jõudluse seoseid [18] ja 1999. aastal tehtud uuring käsitles INNER JOIN operaatorit kasutava ühendamise puhul tabelite elimineerimise teisenduse ja jõudluse seoseid [19]. Tabelite elimineerimise teisenduse kasulikkus tuuakse välja just suuremates andmeaitades kasutusel olevate universaalsete vaadete puhul [18, 19]. Sellistel juhtudel ei ole kasutajal sageli võimalik päringut esitada vastavalt vajatud tabelitele, vaid peab kasutama ühte suurt vaadet, kuhu on toodud kokku andmed paljudest erinevatest tabelitest. Mõlemas uuringus leitakse, et tabelite elimineerimise teisendusel on suur mõju jõudlusele ja töökiirusele just universaalsete vaadete kasutamise puhul.

4 Eksperimentide kirjeldus

Selles peatükis kirjutatakse, kuidas täpselt eksperimente läbi viidi, et huvilistel oleks võimalus neid eksperimente korrata või laiendada. Eksperimentide jaoks kirjutatud kood on leitav GitHubi salvest https://github.com/Karjur/JoinElimination

4.1 Andmebaasi struktuur

Töö üheks eesmärgiks on uurida kuidas on SQL-andmebaasisüsteemides tabelite elimineerimise teisenduse tugi ajas muutunud. Sellel põhjusel valiti katsetamiseks sama andmebaasi struktuur, mida kasutati 2014. aastal tehtud bakalaureusetöös. Andmebaasi loomisel on oluline, et oleksid jõustatud välisvõtme kitsendused, mis võimaldavad andmebaasisüsteemidel kontrollida viidete terviklikkuse reegli kehtivust. [4] Teiste sõnadega tagavad välisvõtme kitsendused, et andmebaasis ei registreerita katkiseid viiteid. Andmebaasi struktuur on nähtav jooniselt 1.



Joonis 1. Tabelite elimineerimise teisenduse andmebaasi füüsilise disaini diagramm Oracle andmebaasisüsteemi jaoks.

Töö autor sai sisendina tabelite loomise laused Oracle ja PostgreSQL jaoks ning teisendas neid, et saada laused ka teiste andmebaasisüsteemide jaoks. Muudatused seisnesid peamiselt selles, et erinevate andmebaasisüsteemide poolt pakutavad andmetüübid on erinevad.

Lisaks loodi andmebaasis vaated *Tootaja_osakond* ja *Tootaja_osakond_klassikaline*, mille loomise kood on esitatud lisas 5.

4.2 Testandmete hulk ja genereerimine

Tabelite elimineerimise teisenduse võimekuse testimise võrdlemiseks peavad olema kõigis andmebaasides samasugused andmed. Testimine viiakse läbi kolmes erineva andmete hulgaga, et uurida seoseid ridade arvu ja tööaja vahel. Ridade hulka muudeti ainult tabelites *Projekt* ja *Tootaja* (vt tabel 1). Mõõtmisi erinevate ridade hulkadega tehakse selleks, et teha kindlaks, kas ridade hulga ja täitmise aja vahel on lineaarne seos või mitte. Selleks arvutatakse tabelis *Tootaja* oleva ridade arvu ja lause täitmise aegade põhjal välja Pearsoni korrelatsioonikordaja väärtused. Korrelatsioonikordaja võimalike väärtuste vahemik on -1 ja 1. Positiivne väärtus tähendab kasvavat seost e ühe tunnuse suur väärtus tingib ka teise suure väärtuse ja ühe tunnuse väärtuse kasvades kasvab ka teine. Negatiivne väärtus tähendab vastupidi kahanevat seost. [22] Mida lähemal on väärtus 1-le või -1-le, seda tugevam on seos tunnuste väärtuste vahel. Väärtuseid 0.7-0.89 loetakse tavaliselt tugevaks seoseks ning nendest suuremaid väga tugevaks seoseks. [23]

Tabel 1. Ridade arv tabelites.

Tabeli nimetus	**Esimene test**	**Teine test**	**Kolmas test**
Tootaja	100 000 rida	200 000 rida	400 000 rida
Projekt	100 000 rida	200 000 rida	400 000 rida
Osakond	10 000 rida	10 000 rida	10 000 rida
Asukoht	100 rida	200 000 rida	400 000 rida
Valuuta	50 rida	50 rida	50 rida

Testandmete genereerimiseks kaalus autor kahte valikut. Esimene variant oli PostgreSQL-i pakutav funktsioon *generate_series()* [24] koos juhuslike väärtuste genereerimisega. Teiseks valikuks oli veebipõhine generaator, nt Mockaroo [25]. Valituks osutus PostgreSQLil põhinev lahendus. Valiku põhjuseks oli selle kasutamise lihtsus ning autori varasem kogemus. Lisaks on PostgreSQL-ist kerge eksportida andmeid CSV formaadis, et neid teistesse andmebaasisüsteemidesse importida. Andmete genereerimise laused on nähtavad tabelist 2.

Tabel 2. Andmete genereerimine PostgreSQL-is esimese ridade arvu korral.

Tabeli nimetus	**Andmete genereerimise laused**

Valuuta	`INSERT INTO Valuuta (valuutakood, nimetus)`
	`SELECT`
	`  'a' || s.i,`
	`  'Valuuta ' || s.i`
	`FROM`
	`  generate_series(1, 50) AS s(i);`

Asukoht	`INSERT INTO Asukoht (asukoht_ID, valuutakood, asukoht)`
	`SELECT`
	`  s.i,`
	`  'a' || ((s.i – 1) % 50) + 1,`
	`  'Asukoht ' || s.i`
	`FROM`
	`  generate_series(1, 100) AS s(i);`

Osakond	`INSERT INTO Osakond (osakond_ID, asukoht_ID, nimi)`
	`SELECT`
	`  s.i,`
	`  ((s.i – 1) % 100) + 1,`
	`  'Osakond ' || s.i`
	`FROM`
	`  generate_series(1, 10000) AS s(i);`

Tootaja	`INSERT INTO Tootaja (osakond_ID, eesnimi, palk, perenimi, alates)`
	`SELECT`
	`  (s.i % 10000) + 1,`
	`  'Eesnimi ' || s.i,`
	`  (s.i % 2000) + 1000,`
	`  'Perenimi ' || s.i,`
	`  CURRENT_DATE – (s.i % 365) * INTERVAL '1 day'`
	`FROM`
	`  generate_series(1, 100000) AS s(i);`

Projekt	`INSERT INTO Projekt (tootaja_ID, tahtaeg, prioriteet)`
	`SELECT`
	`  T.tootaja_ID,`
	`  CURRENT_DATE + (s.i % 100) * INTERVAL '1 day',`
	`  (s.i % 5) + 1`
	`FROM`
	`  generate_series(1, 100000) AS s(i)`
	`JOIN Tootaja T ON T.tootaja_ID = s.i;`

4.3 Andmebaasides käivitatavad SQL laused

Käivitatavad laused, mis olid juba töös [4], esitatakse tabelis 3. Tabelis 4 esitatakse käesolevas töös täiendavalt katsetavad laused ja tabelis 5 põhjendatakse nende valikut. Kõik laused on kirja pandud PostgreSQLi SQL dialekti e mägimurrakut kasutades.

Enamik nendest lausetest on muutmata kujul käivitatavad ka teistes vaadeldavates süsteemides.

Eranditeks olid järgnevad laused.

- MySQL-is pidi muutma lauseid S20, S21 ja S22. Lauset S19 ei ole võimalik teostada.
- MariaDB pidi muutma lauseid S21 ja S22.
- MS SQL Serveris pidi muutma lauseid S21, S22 ja S23.
- DB2-s pidi muutma lauseid S21 ja S23.
- Oracle puhul pidi muutma lauset S22.

Enamik muudatusi on seotud andmete uuendamise või kustutamisega läbi ühendamise ning päringutulemuste materialiseerimisega. Muudetud laused on esitatud lisas 6.

Tabel 3. Laused eelmisest uuringust.

Tunnus	**Lause**	**Milline võiks olla lihtsustatud lause?**
S1	`SELECT tootaja FROM Tootaja_osakond;`	`SELECT perenimi FROM Tootaja;`
S2	`SELECT perenimi FROM Tootaja_osakond_klassikaline;`	`SELECT perenimi FROM Tootaja;`
S3	`SELECT Osakond.asukoht_id FROM Osakond INNER JOIN Asukoht ON Osakond.asukoht_id = Asukoht.asukoht_id;`	`SELECT nimi, osakond_ID FROM Osakond;`
S3*	`SELECT Osakond.asukoht_id FROM Osakond LEFT OUTER JOIN Asukoht ON Osakond.asukoht_ID = Asukoht.asukoht_ID;`	`SELECT nimi, osakond_ID FROM Osakond;`
S4	`SELECT T.tootaja_ID, T.perenimi FROM Tootaja T, Osakond O WHERE T.osakond_ID =O.osakond_ID;`	`SELECT tootaja_ID, perenimi FROM Tootaja;`

Tunnus	Lause	Milline võiks olla lihtsustatud lause?
S5	SELECT O.nimi, O.osakond_ID FROM Osakond O WHERE NOT EXISTS (SELECT 1 FROM Asukoht WHERE Asukoht.asukoht_ID=O.asukoht_ID);	SELECT nimi, osakond_ID FROM Osakond WHERE asukoht_ID IS NULL;
S6	SELECT O.nimi, O.osakond_ID FROM Osakond O WHERE O.asukoht_id IN (SELECT asukoht_ID FROM Asukoht WHERE Asukoht_id IS NOT NULL);	SELECT nimi, osakond_id FROM Osakond;
S7	SELECT Osakond.nimi, Osakond.osakond_ID FROM Osakond LEFT OUTER JOIN Tootaja ON Tootaja.osakond_ID = Osakond.osakond_ID AND Tootaja.tootaja_ID= (SELECT max (tootaja_ID) FROM Tootaja WHERE Tootaja.osakond_ID = Osakond.osakond_ID);	SELECT nimi, osakond_id FROM Osakond;
S8	SELECT tootaja_ID FROM Tootaja UNION SELECT tootaja_ID FROM Tootaja;	SELECT tootaja_ID FROM Tootaja;
S9	SELECT tootaja_ID FROM Tootaja INTERSECT SELECT tootaja_ID FROM Tootaja;	SELECT tootaja_ID FROM Tootaja;
S10	SELECT T1.tootaja_ID FROM Tootaja T1 JOIN Tootaja T2 ON T1.tootaja_ID=T2.tootaja_ID;	SELECT tootaja_ID FROM Tootaja;
S11	SELECT T1.tootaja_ID FROM Tootaja T1, Tootaja T2 WHERE T1.tootaja_ID=T2.tootaja_ID;	SELECT tootaja_ID FROM Tootaja;
S12	SELECT P.tootaja_ID FROM Projekt P INNER JOIN Tootaja T ON T.Tootaja_ID=P.Tootaja_ID;	SELECT P.tootaja_ID FROM Projekt P INNER JOIN Tootaja T ON T.Tootaja_ID=P.Tootaja_ID;

Tabel 4. Käesoleva uuringu uued laused.

Tunnus	**Lause**	**Milline võiks olla lihtsustatud lause?**
S13	SELECT O.nimi FROM Osakond O INNER JOIN Asukoht A ON O.asukoht_ID = A.asukoht_ID INNER JOIN Valuuta V ON A.valuutakood = V.valuutakood;	SELECT nimi FROM Osakond;
S14	SELECT O.nimi FROM Osakond O LEFT OUTER JOIN Asukoht A ON O.asukoht_ID = A.asukoht_ID LEFT OUTER JOIN Valuuta V ON A.valuutakood = V.valuutakood;	SELECT nimi FROM Osakond;
S15	SELECT T.tootaja_ID FROM Tootaja T INNER JOIN Osakond O ON T.osakond_id = O.osakond_id INNER JOIN Asukoht A ON O.asukoht_id = A.asukoht_id INNER JOIN Valuuta V ON A.valuutakood = V.valuutakood;	SELECT tootaja_id FROM Tootaja;
S16	SELECT T.tootaja_ID FROM Tootaja T LEFT OUTER JOIN Osakond O ON T.osakond_id = O.osakond_id LEFT OUTER JOIN Asukoht A ON O.asukoht_id = A.asukoht_id LEFT OUTER JOIN Valuuta V ON A.valuutakood = V.valuutakood;	SELECT tootaja_id FROM Tootaja;
S17	SELECT O.nimi FROM Osakond O WHERE O.osakond_ID IN (SELECT osakond_ID FROM Osakond);	SELECT nimi FROM osakond;
S18	SELECT O.nimi FROM Osakond O WHERE EXISTS (SELECT 1 FROM Osakond WHERE Osakond.osakond_ID = O.osakond_ID);	SELECT nimi FROM osakond;
S19	DELETE FROM Tootaja WHERE tootaja_id IN (SELECT tootaja_ID FROM Tootaja);	DELETE FROM Tootaja;
S20	UPDATE Tootaja SET eesnimi = 'UusNimi' WHERE tootaja_id IN (SELECT tootaja_ID FROM Tootaja);	UPDATE Tootaja SET eesnimi = 'Uusnimi';

Tunnus	Lause	Milline võiks olla lihtsustatud lause?
S21	DELETE FROM Tootaja USING Tootaja t LEFT JOIN Osakond o ON t.osakond_ID = o.osakond_ID LEFT JOIN Asukoht a ON o.asukoht_ID = a.asukoht_ID LEFT JOIN Valuuta v ON a.valuutakood = v.valuutakood WHERE Tootaja.tootaja_ID = t.tootaja_ID;	DELETE FROM Tootaja;
S22	UPDATE Tootaja SET eesnimi = 'Uusnimi' FROM Tootaja t LEFT JOIN Osakond o ON t.osakond_ID = o.osakond_ID LEFT JOIN Asukoht a ON o.asukoht_ID = a.asukoht_ID LEFT JOIN Valuuta v ON a.valuutakood = v.valuutakood WHERE Tootaja.tootaja_ID = t.tootaja_ID;	UPDATE Tootaja SET eesnimi = 'Uusnimi';
S23	CREATE TABLE Tootaja2 AS SELECT * FROM Tootaja WHERE 1=0; INSERT INTO Tootaja2 (tootaja_ID, osakond_ID, eesnimi, palk, perenimi, alates) SELECT T.tootaja_ID, T.osakond_ID, T.eesnimi, T.palk, T.perenimi, T.alates FROM Tootaja T LEFT JOIN Osakond O ON T.osakond_ID = O.osakond_ID LEFT JOIN Asukoht A ON O.asukoht_ID = A.asukoht_ID LEFT JOIN Valuuta V ON A.valuutakood = V.valuutakood;	CREATE TABLE Tootaja2 AS SELECT * FROM Tootaja WHERE 1=0; INSERT INTO Tootaja2 (tootaja_ID, osakond_ID, eesnimi, palk, perenimi, alates) SELECT T.tootaja_ID, T.osakond_ID, T.eesnimi, T.palk, T.perenimi, T.alates FROM Tootaja T
S24	SELECT T.tootaja_ID FROM Tootaja T LEFT OUTER JOIN Osakond O ON T.osakond_id = O.osakond_id INNER JOIN Asukoht A ON O.asukoht_id = A.asukoht_id LEFT OUTER JOIN Valuuta V ON A.valuutakood = V.valuutakood;	SELECT tootaja_id FROM Tootaja;

Tunnus	Lause	Milline võiks olla lihtsustatud lause?
S25	SELECT T.tootaja_ID FROM Tootaja T INNER JOIN Osakond O ON T.osakond_id = O.osakond_id LEFT OUTER JOIN Asukoht A ON O.asukoht_id = A.asukoht_id LEFT OUTER JOIN Valuuta V ON A.valuutakood = V.valuutakood;	SELECT tootaja_id FROM Tootaja;
S26	SELECT tootaja_id FROM Tootaja UNION SELECT tootaja_id FROM Tootaja;	SELECT tootaja_id FROM Tootaja

Tabel 5. Eksperimendi uute lausete selgitused.

Tunnus	Selgitus, miks selline uus lause analüüsimiseks valiti
S13	Kolme tabeli ühendamine INNER JOIN operaatoriga. Kasutatakse ainult ühes tabelis olevaid veerge, seega on kaks tabelit üleliigsed.
S14	Kolme tabeli ühendamine OUTER JOIN operaatoriga. Kasutatakse ainult ühes tabelis olevaid veerge, seega on kaks tabelit üleliigsed.
S15	Nelja tabeli ühendamine INNER JOIN operaatoriga. Kasutatakse ainult ühes tabelis olevaid veerge, seega on kolm tabelit üleliigsed.
S16	Nelja tabeli ühendamine OUTER JOIN operaatoriga. Kasutatakse ainult ühes tabelis olevaid veerge, seega on kolm tabelit üleliigsed.
S17	Mittekorreleeruv alampäring, kus sama tabelit uuesti loetakse, on üleliigne, sest päringu tingimus on kõigi ridade puhul täidetud.
S18	Korreleeruv alampäring, kus sama tabelit uuesti loetakse, on üleliigne, sest päringu tingimus on kõigi ridade puhul täidetud.
S19	Soovitakse teada, kas tabelite elimineerimise teisendust osatakse teha DELETE lause korral, kui lauses on mittekorreleeruv alampäring, mis loeb sama tabelit.
S20	Soovitakse teada, kas tabelite elimineerimise teisendust osatakse teha UPDATE lause korral, kui lauses on mittekorreleeruv alampäring, mis loeb sama tabelit.
S21	Soovitakse teada, kas tabelite elimineerimise teisendust osatakse teha DELETE lause korral, kui kasutatakse kustutamist läbi ühendamise.
S22	Soovitakse teada, kas tabelite elimineerimise teisendust osatakse teha UPDATE lause korral, kui kasutatakse uuendamist läbi ühendamise.
S23	Soovitakse teada, kas tabelite elimineerimise teisendust osatakse teha INSERT lause korral.
S24	Kolme tabeli ühendamine läbisegi INNER ja OUTER JOIN operaatoritega. Kasutatakse ainult ühes tabelis olevaid veerge, seega on kaks tabelit üleliigsed. Selle tabeliga ühendatakse tabel kasutades OUTER JOIN operaatorit.
S25	Kolme tabeli ühendamine läbisegi INNER ja OUTER JOIN operaatoritega. Kasutatakse ainult ühes tabelis olevaid veerge, seega on kaks tabelit üleliigsed. Selle tabeliga ühendatakse tabel kasutades INNER JOIN operaatorit.
S26	Soovitakse teada, kas tabelite elimineerimise teisendust osatakse teha kui kasutatakse UNION operaatorit tabeli ühendi leidmiseks iseendaga.

4.3.1 Ridade hulgast tingitud töökiiruse muutused

Ridade hulgast tingitud töökiiruse muutuste mõõtmiseks võrreldakse lauseid kus on sees tabel *Tootaja*, kuna selles tabelis muutub ridade arv ja seetõttu võimaldab see näha

töökiiruse vahet erinevate ridade hulkade korral. Nende lausete identifikaatorid e tunnused on esitatud tabelis 6.

Tabel 6. Laused, millega hinnata ridade hulgast tingitud töökiiruse muutuseid.

S1	S2	S4	S7	S8
S9	S10	S11	S12	S15
S16	S19	S20	S21	S22
S23	S24	S25	S26	

4.4 Ankurmodelleerimise abil loodud andmebaasi struktuur

Lisaks tabelite elimineerimise teisenduse testimisest olemasoleva andmebaasi struktuuri peal, valiti katsetamiseks ka 2023. aastal tehtud magistritöös ankurmodelleerimise abil tehtud andmebaasi alamosa (vt joonis 2) [5]. Selles andmebaasis registreeritakse andmeid filmide kohta. Katsetatavad päringud on vaate *lMV_movie põhjal*. “Vaade tagastab filmide loetelu, kus iga filmi kohta on selle atribuudi väärtused. Kui filmi mõnel atribuudil väärtus puudub, siis tänu välisühendamisele vaate aluseks olevas päringus on vaates selle atribuudi väärtuse asemel NULL. Kui ankur sisaldab ajaloostatud atribuute, siis nendest tagastatakse kõige suurema changetAt väärtusega atribuudi rida.” [5] See vaade kasutab lisaks ankurmudeli andmebaasi baastabelitele ka vaadet *lMV_rat_movie_rating*. “Kuna rating on ajaloostatud atribuut, siis tegemist on abistava vaatega, mida vaade Lmv_movie kasutab leidmaks maksimaalse changedAt väärtusega rating rida.” [5]

Ankurmodelleerimise andmebaasi loomiseks võeti aluseks ankurmodelleerimise vahendi poolt PostgreSQL jaoks genereeritud kood. Tabel 7 toob välja erinevate andmebaasisüsteemide korral ankurmudeli andmebaasi loomise koodis tehtud muudatused.

Tabel 7. Peamised erinevused andmebaasisüsteemide SQL keele süntaksis andmebaasi loomisel.

Andmebaasisüsteem	**Muutused**
MySQL 8.3	SERIAL => AUTO_INCREMENT, BOOLEAN => TINYINT, *Primary key*, *foreign key*, *unique constraint* määramise süntaks
MariaDB 11.4	SERIAL => AUTO_INCREMENT, BOOLEAN => TINYINT, *Primary key*, *foreign key*, *unique constraint* määramise süntaks
MS SQL Server 2022	SERIAL => IDENTITY, BOOLEAN => BIT, *Primary key*, *foreign key*, *unique constraint* määramise süntaks, Eemaldada ORDER BY vaate loomisel, sest selle võimekus puudub
Oracle23c	INTEGER => NUMBER, SERIAL => GENERATED BY DEFAULT AS IDENTITY NOT NULL, *Primary key*, *foreign key*, *unique constraint* määramise süntaks
DB2 11.5	SERIAL => GENERATED BY DEFAULT AS IDENTITY NOT NULL, *Primary key*, *foreign key*, *unique constraint* määramise süntaks

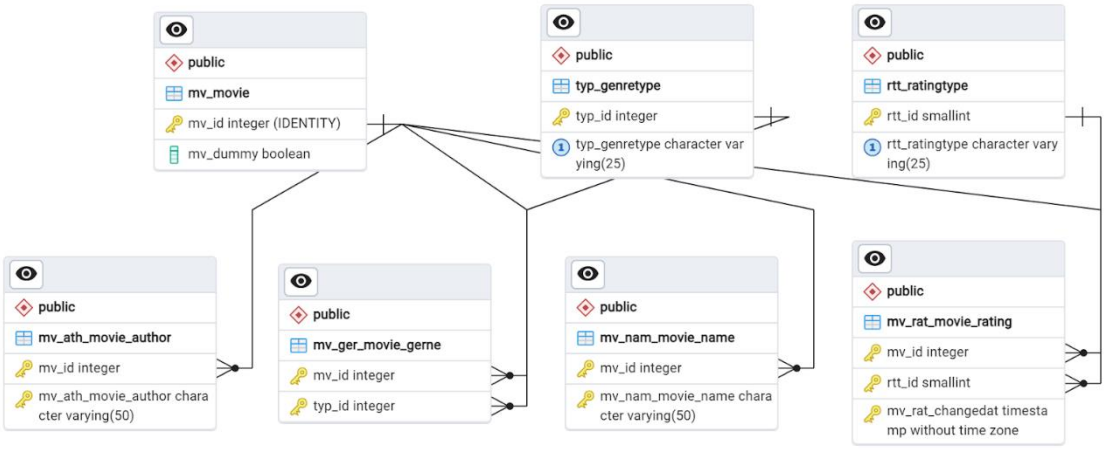
Testitakse, kas vaate *lMV_movie* põhjal päringut tehes rakendatakse tabelite elimineerimist ning mitut tabelit kasutatakse andmebaasisüsteemi poolt lause täitmisel. Ankurmodelleerimise andmebaasis käivitati vaate põhjal päringud nelja ülesande lahendamiseks:

- A1: Leia selliste filmide id ja nimi, mille žanr on identifikaatoriga 2.
    - Täitmiseks oleks vaja tabeleid *mv_nam_movie_name* ja *mv_ger_movie_genre*.
- A2: Leia iga žanri tüübi kohta, mida on kasutatud mõne filmi iseloomustamiseks, selles olevate filmide arv. Väljastage žanri tüübi id ja filmide arv.
    - Täitmiseks oleks vaja tabeleid *typ_genretype* ja *mv_ger_movie_genre*.

- A3: Leia selliste filmide id ja nimi, mille hetkel kehtiv reitingu tüüp on koodiga 2.
    - Täitmiseks oleks vaja tabeleid *mv_nam_movie_name* ja *mv_rat_movie_rating*.
- A4: Leia iga hetkel filmide iseloomustamiseks kasutatud reitingu tüübi kohta, selles olevate filmide arv. Väljastage reitingu tüübi id ja filmide arv.
    - Täitmiseks oleks vaja tabeleid *rtt_ratingtype* ja *mv_rat_movie_rating*.

Selliste ülesannete valiku põhjendus.

- Ülesanded A1 ja A3 hõlmavad staatilist atribuuti (filmi nimi).
- Ülesanded A1 ja A2 hõlmavad sõlmitud atribuuti (filmi žanr).
- Ülesanded A3 ja A4 hõlmavad sõlmitud ajaloolist atribuuti (filmi reiting).
- Ülesanded A2 ja A4 leiavad koondandmeid. Ankurmodelleerimist kasutatakse senini eeskätt andmeaitade ja -vakkade loomiseks, kus tüüpilised päringud leiavad koondandmeid.
- Ülesanded A1 ja A3 otsivad infot konkreetsete olemite kohta. Heade omaduste tõttu (skeemimuudatuste lihtsus, puuduvate andmetega toimetuleku lihtsus) võiks ankurmodelleerimise kasutamist kaaluda ka operatiivandmete andmebaasides, kus tüüpilised laused otsivad üksikute olemite andmeid.



Joonis 2. Ankurmodelleeerimist kasutades loodud andmebaasi baastabelite struktuur.

4.5 Ankurmodelleerimise andmebaasis käivitatavad SQL laused

Tabelis 8 esitatakse laused, millega katsetatakse tabelite elimineerimist ankurmodelleerimise abil loodud andmebaasi põhjal. Kõik laused on kirja pandud PostgreSQLi SQL dialekti kasutades. Kõik nendest lausetest on muutmata kujul käivitatavad ka teistes vaadeldavates süsteemides.

Tabel 8. Laused, millega katsetatakse tabelite elimineerimist ankurmodelleerimise tulemusel loodud andmebaasis.

Tunnus	Päring	Milline võiks olla lihtsustatud lause?	Minimaalne tabelite arv
A1	SELECT mv_id AS movie_id, mv_nam_movie_name AS movie_name FROM lMV_Movie WHERE TYP_ID =2;	SELECT MV_nam_movie_name.MV_ID AS movie_id, MV_NAM_Movie_Name AS movie_name FROM MV_nam_movie_name JOIN MV_GER_Movie_Gerne ON MV_nam_Movie_name.MV_ID = MV_GER_Movie_Gerne.MV_ID WHERE MV_GER_Movie_Gerne.TYP_ID = 2;	2
A2	SELECT TYP_GenreType AS genre, Count(*) AS nr_of_movies FROM lMV_Movie GROUP BY TYP_GenreType;	SELECT kGER.TYP_GenreType AS genre, Count(*) AS nr_of_movies FROM MV_GER_Movie_Gerne AS GER JOIN TYP_GenreType AS kGER ON GER.TYP_ID = kGER.TYP_ID GROUP BY kGER.TYP_GenreType;	2
A3	SELECT mv_id AS movie_id, mv_nam_movie_name AS movie_name FROM lMV_Movie WHERE RTT_ID =2;	SELECT mv_id AS movie_id, MV_NAM_Movie_Name AS movie_name FROM MV_NAM_Movie_Name WHERE mv_id IN (SELECT MV_ID FROM MV_RAT_Movie_Rat	2
A4	SELECT RTT_RatingType AS rating, Count(*) AS nr_of_movies FROM lMV_Movie GROUP BY RTT_RatingType;	SELECT RTT_RatingType AS rating, COUNT(*) AS nr_of_movies FROM MV_RAT_Movie_Rating JOIN RTT_RatingType ON MV_RAT_Movie_Rating.RTT_ID = RTT_RatingType.RTT_ID GROUP BY RTT_RatingType;	2

4.6 Töökiiruse mõõtmine ja täitmisplaanide vaatamine

Tabel 9 esitab käsud, mida erinevates andmebaasisüsteemides kasutati füüsiliste täitmisplaanide (edaspidi täitmisplaanide) vaatamiseks. Täitmisplaane on vaja vaadata, et näha, kas andmebaasisüsteem on teinud tabelite elimineerimise teisendust või mitte.

Peale ridade hulkade muutmist tuli ka värskendada andmebaasi statistikat, mille alusel andmebaasisüsteem täitmisplaane koostab, et andmebaasisüsteemi optimeerimismooduli käsutuses oleks võimalikult täpne info andmebaasis olevate andmete kohta. Käsud statistika värskendamiseks on samuti välja toodud tabelis 9.

Tabel 9. Kasutatud täitmisplaanide vaatamise ja andmebaasi statistika värskendamise käsud.

Andmebaasi-süsteem	**Täitmisplaani vaatamise käsk**	**Andmebaasi statistika värskendamise käsud**
PostgreSQL16	`EXPLAIN ANALYZE {lause};`	`VACUUM ANALYZE public.{tabeli nimi};`
Oracle23c	`SET AUTOTRACE TRACEONLY; {lause}`	`EXEC DBMS_STATS.GATHER_SCHEMA_STATS({skeemi nimi});`
MS SQL Server 2022	-	`UPDATE STATISTICS {tabeli nimi};`
MySQL 8.3	`EXPLAIN ANALYZE {lause};`	`ANALYZE TABLE {tabeli nimi};`
MariaDB 11.4	`ANALYZE {lause};`	`ANALYZE TABLE {tabeli nimi};`
DB2 11.5	-	-

4.6.1 PostgreSQL 16

Andmebaasi statistika värskendamiseks kasutati käsku tabelist 9.

Töökiiruse mõõtmiseks kasutati PgAdmin4 sisseehitatud *query complete* funktsiooni.

Täitmisplaani vaatamiseks (vt joonis 3) pandi lause ette täitmisplaani vaatamise käsk tabelist 9 ning käivitati lause. Täitmisplaanist näeme, et ei toimu tabeli elimineerimist, sest kasutatakse nii *Tootaja* kui ka *Osakond* tabeleid.

	QUERY PLAN text
1	Hash Join (cost=288.00..9076.44 rows=400000 width=15) (actual time=1.886..159.133 rows=400000 loops=1)
2	Hash Cond: (tootaja.osakond_id = osakond.osakond_id)
3	-> Seq Scan on tootaja (cost=0.00..7738.00 rows=400000 width=17) (actual time=0.009..37.993 rows=400000 loo…
4	-> Hash (cost=163.00..163.00 rows=10000 width=2) (actual time=1.816..1.817 rows=10000 loops=1)
5	Buckets: 16384 Batches: 1 Memory Usage: 461kB
6	-> Seq Scan on osakond (cost=0.00..163.00 rows=10000 width=2) (actual time=0.006..0.797 rows=10000 loop…

Joonis 3. Lause S1 tätimisplaan PostgreSQL16, PgAdmin4-s.

4.6.2 Oracle23c

Andmebaasi statistika värskendamiseks kasutati käsku tabelist 9. Selle käsuga uuendatakse kõigi skeemi kuuluvate objektide statistika.

Töökiiruse mõõtmiseks kasutati käsku: SET TIMING ON; Pärast seda tuli käivitada lause.

Täitmisplaani vaatamiseks (vt joonis 4) kasutati täitmisplaani vaatamise käsku tabelist 9. Peale seda tuli käivitada lause. Täitmisplaanist näeme, et tabeli elimineerimise teisendus toimub, sest loetakse ainult tabelit *Tootaja*.

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
SQL_ID  c7jwmydtbd935, child number 0
-------------------------------------
SELECT tootaja FROM Tootaja_osakond

Plan hash value: 1476519426

-----------------------------------------------
| Id  | Operation         | Name    | E-Rows |
-----------------------------------------------
|   0 | SELECT STATEMENT  |         |        |
|   1 |  TABLE ACCESS FULL| TOOTAJA |      1 |
-----------------------------------------------
```

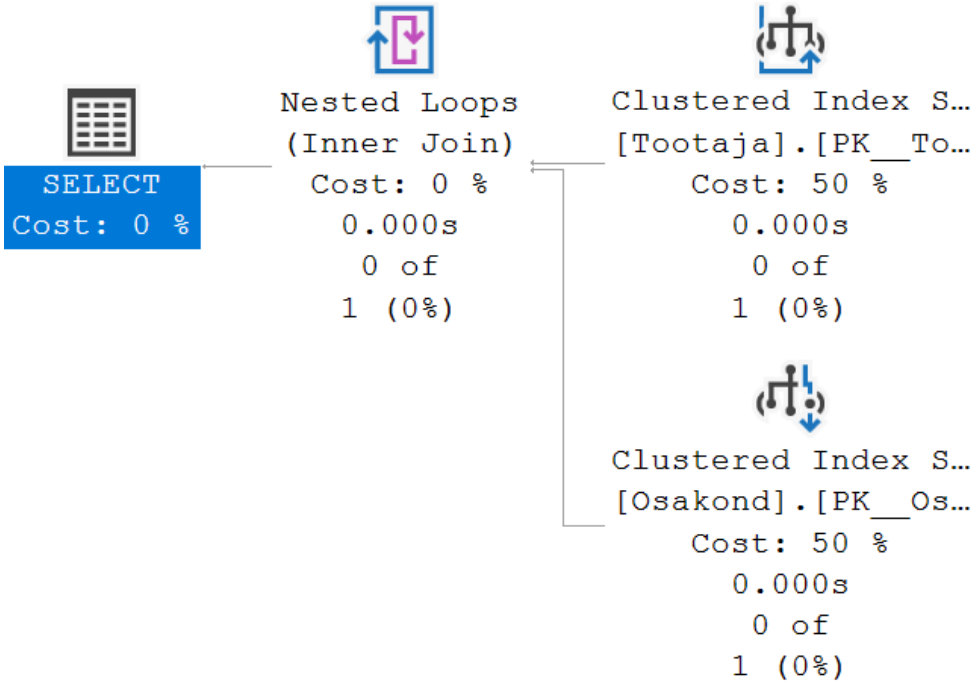
Joonis 4. Lause S1 täitmisplaan Oracle23c-s.

4.6.3 MS SQL Server 2022

Andmebaasi statistika värskendamiseks kasutati käsku tabelist 9.

Töökiiruse mõõtmiseks kasutati SQL Server Management Studio 19 sisseehitatud töökiiruse mõõtmise funktsiooni.

Täitmisplaani vaatamiseks (vt joonis 5) kasutati enne lause käivitamist kiirkorraldust CTRL+M, peale mida käivitati lause. Täitmisplaanist näeme, et ei toimu tabeli elimineerimist, sest kasutatakse nii *Tootaja* kui ka *Osakond* tabeleid (läbi indeksite).



Joonis 5. Lause S1 täitmisplaan MS SQL Server 2022, SQL Server Management Studio 19s.

4.6.4 MySQL 8.3

Andmebaasi statistika värskendamiseks kasutati käsku tabelist 9.

Töökiiruse mõõtmiseks kasutati Beekeeper Studio-sse sisseehitatud töökiiruse mõõtmise funktsiooni.

Täitmisplaani vaatamiseks (vt joonis 6) pandi lause ette täitmisplaani vaatamise käsk tabelist 9 ning käivitati lause. Täitmisplaanist näeme, et ei toimu tabeli elimineerimist, sest kasutatakse nii *Tootaja* kui ka *Osakond* tabeleid (läbi indeksite).

```
-> Nested loop inner join  (cost=137698 rows=390571) (actual time=12.8..1122 rows=400000 loops=1)
    -> Covering index scan on osakond using idx_osak_asukoht  (cost=998 rows=9921) (actual time=0.79..5.07 rows=10000 loops=1)
    -> Index lookup on tootaja using idx_tootaja_osak (osakond_ID=osakond.osakond_ID)  (cost=9.84 rows=39.4) (actual time=0.106..0.11 rows=40 loops=10000)
```

Joonis 6. Lause S1 täitmisplaan MySQL 8.3s.

4.6.5 MariaDB 11.4

Andmebaasi statistika värskendamiseks kasutati käsku tabelist 9.

Töökiiruse mõõtmiseks kasutati Beekeeper Studio-sse sisseehitatud töökiiruse mõõtmise funktsiooni.

Täitmisplaani vaatamiseks (vt joonis 7) pandi lause ette täitmisplaani vaatamise käsk tabelist 9 ning käivitati lause. Täitmisplaanist näeme, et ei toimu tabeli elimineerimist, sest kasutatakse nii *Tootaja* kui ka *Osakond* tabeleid (läbi indeksite).

	id	select_type	table	type	possible_keys	key	key_len	ref	rows	r_rows	filtered	r_filtered	Extra
1	1	SIMPLE	osakond	index	PRIMARY	idx_osak_asukoht	2	(NULL)	10028	10000.00	100	100	Using index
2	1	SIMPLE	tootaja	ref	idx_tootaja_osak	idx_tootaja_osak	2	mariabaka.osakond.osakond_ID	4	10.00	100	100	(EMPTY)

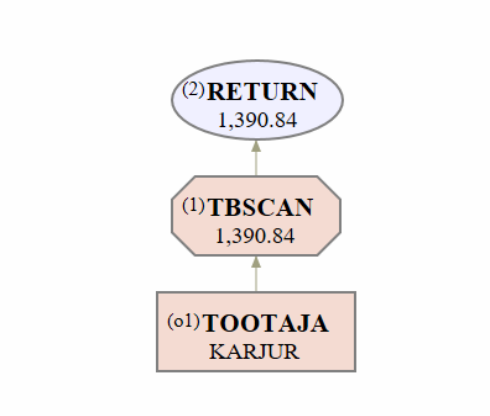
Joonis 7. Lause S1 täitmisplaan MariaDB 11.4, Beekeeper Studios.

4.6.6 DB2 11.5

Andmebaasi statistika värskendamiseks tuli Data Studios tabelite loendis teha paremklõps tabelil, seejärel valida Manage ja peale seda Run Statistics.

Töökiiruse mõõtmiseks kasutati Data Studio-sse sisseehitatud töökiiruse mõõtmise funktsiooni.

Täitmisplaani vaatamiseks (vt joonis 8) valiti enne lause käivitamist Start Tuning, peale mida sai luua andmebaasi vastavad täitmisplaanide salvestamise tabelid. Peale tabelite loomist käivitati lause ja siis valiti Open Visual Explain. Täitmisplaanist näeme, et toimub tabeli elimineerimise teisendus, sest loetakse ainult tabelit *Tootaja*.



Joonis 8. Lause S1 täitmisplaan DB2 11.5, Data Studios.

5 Eksperimendi tulemused

Selles peatükis käsitletakse eksperimentide tulemusi. Tulemuste analüüs esitatakse peatükis 6.

5.1 Tabelite elimineerimise teisenduse rakendamine

Jaotises 4.3 esitatud lausete täitmisplaanide analüüsimise tulemused on väljatoodud tabelis 10. Tabeli lõpus on summeeritud tulemused.

- X näitab, et tabelite elimineerimise teisendus toimus ja eemaldati kõikide tabelite kasutus, mida lause täitmiseks pole vaja. Selle kohta öeldakse, et toimus täielik teisendus.
- O näitab, et tabelite elimineerimise teisendust ei toimunud.
- V näitab, et toimus osaline tabelite elimineerimise teisendus. Osaline teisendus tähendab, et eemaldatakse osade, aga mitte kõikide tabelite kasutus, mida lause täitmiseks pole vaja.
- - tähendab, et lauset ei saanud käivitada.

Tabel 10. Tabelite elimineerimise teisenduse katsetamise tulemused.

Lause	**PostgreSQL 16**	**Oracle23c**	**MS SQL Server 2022**	**MySQL 8.3**	**MariaDB 11.4**	**DB2 11.5**
S1	O	X	O	O	O	X
S2	O	X	O	O	O	X
S3	O	X	O	O	O	X
S3*	X	X	X	O	X	X
S4	O	X	O	O	O	X
S5	O	X	O	O	O	O
S6	O	X	O	O	O	X
S7	X	O	X	O	X	O
S8	O	O	O	O	O	O

Lause	PostgreSQL 16	Oracle23c	MS SQL Server 2022	MySQL 8.3	MariaDB 11.4	DB2 11.5
S9	O	O	O	O	O	X
S10	O	X	O	O	O	X
S11	O	X	O	O	O	X
S12	O	O	O	O	O	O
S13	O	X	O	O	O	X
S14	X	X	X	O	V	X
S15	O	X	O	O	O	X
S16	X	X	X	O	V	X
S17	O	X	O	O	O	X
S18	O	X	O	O	O	X
S19	O	X	O	-	O	O
S20	X	X	O	O	O	O
S21	X	O	O	O	V	O
S22	O	X	X	O	V	O
S23	X	X	X	O	V	X
S24	V	X	O	O	V	X
S25	V	X	O	O	V	X
S26	O	O	O	O	O	O
X	7	21	6	0	2	18
O	18	6	21	26	18	9
V	2	0	0	0	7	0
-	0	0	0	1	0	0

Tabelis 11 võrreldakse 2014. aasta eksperimendi (kui kasutusel olid kolme ka selles töös kasutatava andmebaasisüsteemi vanemad versioonid) ja käesoleva eksperimendi tulemusi. Kui tabelite elimineerimise võimekus on andmebaasisüsteemi puhul ajas muutunud, siis on vastavates väljades väärtused rasvase fondiga. Tabeli lõpus on summeeritud tulemused.

Tabel 11. Uuringu vanade lausete tulemuse muutused ajas.

Lause	**PostgreSQL (9.3)**	**PostgreSQL (16)**	**Oracle12c**	**Oracle23c**	**MS SQL Server 2012**	**MS SQL Server 2022**
S1	O	O	X	X	**X**	**O**
S2	O	O	X	X	**X**	**O**
S3	O	O	X	X	**X**	**O**
S3*	X	X	X	X	X	X
S4	O	O	X	X	**X**	**O**
S5	O	O	X	X	O	O
S6	O	O	X	X	**X**	**O**
S7	X	X	O	O	X	X
S8	O	O	O	O	O	O
S9	O	O	O	O	O	O
S10	O	O	X	X	O	O
S11	O	O	X	X	O	O
S12	O	O	O	O	O	O
X	2	2	9	9	7	2
O	11	11	4	4	6	11

5.2 Lausete täitmiseks kuluv aeg

Tabelis 12 on välja toodud lause S7 töökiirused MariaDB ja MySQL-i kohta ning tabelis 13 on mõõtmistulemuste alusel leitud Pearsoni korrelatsioonikordaja väärtused. S7 on ainuke lause, kus MariaDB teeb tabelite elimineerimise teisendust täielikult ning ridade hulga muutus mõjutab lauset. Tabelis 13 on välja toodud Pearsoni korrelatsioonikordaja väärtused lause S7 korral.

Tabel 12. Lause S7 täitmise ajad MariaDB *vs.* MySQL erinevate andmemahtude korral.

Tootaja ridade arv	**MariaDB 11.4**	**MySQL 8.3**
100 000	34,333ms	250,333ms
200 000	23,333ms	340ms
400 000	53,667ms	353,333ms

Tabel 13. Pearsoni korrelatsioonikordaja väärtused lause S7 korral.

Lause	**MariaDB 11.4**	**MySQL 8.3**
S7	0,765	0,8285

Tabelis 14 on välja toodud laused S3 ja S3*, sest tegemist on kahe lihtsustamata lausega, mis annavad loogiliselt samaväärse tulemuse (eeldusel, et välisvõtme kitsendused on jõustatud). Lause S3 puhul antud andmebaasisüsteemid ei teosta tabelite elimineerimist, S3* puhul teostavad (v.a MySQL). Lisaks on väljatoodud MySQLi töökiirused, et võrrelda omavahel MariaDB ja MySQLi tulemusi.

Tabel 14. Lausete S3 ja S3* täitmise aja võrdlus.

Lause	**PostgreSQL16**	**MS SQL Server 2022**	**MariaDB 11.4**	**MySQL 8.3**
S3	176,6ms	30,333ms	42ms	59ms
S3*	124ms	23,667ms	50ms	57,6ms

Tabelis 15 on välja toodud laused S13 ja S14, sest tegemist on kahe lihtsustamata lausega, mis annavad loogiliselt samaväärse tulemuse (eeldusel, et välisvõtme kitsendused on jõustatud). Lause S13 korral ei toimu tabelite elimineerimist, aga S14 puhul toimub (v.a MySQL). Lisaks on väljatoodud MySQLi töökiirused, et võrrelda omavahel MariaDB ja MySQLi tulemusi.

Tabel 15. Lausete S13 ja S14 täitmise aja võrdlus.

Lause	**PostgreSQL16**	**MS SQL Server 2022**	**MariaDB 11.4**	**MySQL 8.3**
S13	133,6ms	33,4ms	44,6ms	111,2ms
S14	109,8ms	24,667ms	45,4ms	69,8ms

Tabelis 16 on näidatud laused S15 ja S16, sest tegemist on kahe lihtsustamata lausega, mis annavad loogiliselt samaväärse tulemuse (eeldusel, et välisvõtme kitsendused on jõustatud). S15 puhul ei toimu tabelite elimineerimise teisendust, S16 puhul toimub (v.a MySQL). Kuigi Oracle ja DB2 said ka lause S15 korral tabelite elimineerimisega hakkama, siis töökiiruse mõõtmise jaoks eemaldati lause S15 jaoks antud andmebaasisüsteemide andmebaasidest välisvõtme kitsendused, et tabelite elimineerimise teisendust ei toimuks. Tabelis 17 on mõõtmistulemuste alusel väljaarvutatud Pearsoni korrelatsioonikordajate väärtused.

Tabel 16. Lausete S15 ja S16 täitmise aegade võrdlus.

Tootaja ridade arv	**Tunnus**	**PostgreSQL 16**	**MS SQL Server 2022**	**MySQL 8.3**	**MariaDB 11.4**	**Oracle 23c**	**DB2 11.5**
100 000	S15	118,667ms	143ms	511ms	102,333 ms	1123,333ms	7ms
100 000	S16	126ms	139,667 ms	535.333 ms	119,667 ms	716,333 ms	2,333ms
200 000	S15	265,333ms	163,667 ms	276ms	68ms	2606,667ms	9,667ms
200 000	S16	151ms	160,667 ms	328ms	78ms	1998,667ms	1ms
400 000	S15	223,333ms	357,667 ms	402ms	286,333 ms	3637,667ms	14,667ms
400 000	S16	103,667ms	333,667 ms	441.333 ms	376ms	3818,333ms	2,33ms

Tabel 17. Pearsoni korrelatsioonikordaja väärtused lausete S15 ja S16 korral.

Lause	**PostgreSQL16**	**MS SQL Server 2022**	**MySQL 8.3**	**MariaDB 11.4**	**Oracle23c**	**DB2 11.5**
S15	0,5441	0,9699	-0,2876	0,8869	0,9572	0,9999
S16	-0,6297	0,9726	-0,2761	0,8947	0,9959	0,1871

Tabelis 18 on näidatud lausete S21, S22 ja S23 täitmise ajad MySQL-is, kus ei toimu teisendust ja MariaDB-s, kus toimub osaline teisendus. Tabelis 19 on mõõtmistulemuste alusel väljaarvutatud Pearsoni korrelatsioonikordajate väärtused.

Tabel 18. Lausete S21, S22 ja S23 täitmise ajad erinevate andmemahtude korral.

Tootaja ridade arv	**Lause**	**MySQL 8.3**	**MariaDB 11.4**
100 000	S21	1286,667ms	1330ms
100 000	S22	1362,667ms	1796,667ms
100 000	S23	1226,333ms	1222,667ms
200 000	S21	3369,333ms	2057,667ms
200 000	S22	5807ms	1249,333ms
200 000	S23	2575,667ms	865ms
400 000	S21	11122,667ms	4446,667ms
400 000	S22	8154,667ms	10613ms
400 000	S23	6069,667ms	4353,333ms

Tabel 19. Pearsoni korrelatsioonikordaja väärtused lausete S21, S22 ja S23 korral.

Lause	**MySQL 8.3**	**MariaDB 11.4**
S21	0,9914	0,9941
S22	0,9336	0,9266
S23	0,9982	0,9103

Tabelis 20 on näidatud laused S24 ja S25, sest tegemist on kahe lihtsustamata lausega, mis annavad loogiliselt samaväärse tulemuse (eeldusel, et välisvõtme kitsendused on jõustatud). MySQL-i puhul ei toimu tabelite elimineerimise teisendust, MariaDB puhul toimub. Tabelis 21 on mõõtmistulemuste alusel väljaarvutatud Pearsoni korrelatsioonikordajate väärtused.

Tabel 20. Lausete S24 ja S25 täitmise ajad erinevate andmemahtude korral.

Tootaja ridade arv	**Lause**	**MySQL 8.3**	**MariaDB 11.4**
100 000	S24	510,667ms	116,667ms
100 000	S25	623,667ms	112,333ms
200 000	S24	145,667ms	61,667ms
200 000	S25	327,667ms	66ms
400 000	S24	362,667ms	267ms
400 000	S25	358,333ms	299,667ms

Tabel 21. Pearsoni korrelatsioonikordaja väärtused lausete S24 ja S25 korral.

Lause	**MySQL 8.3**	**MariaDB 11.4**
S24	-0,2229	0,8281
S25	-0,6909	0,8669

Tabelis 22 on näidatud kõikide tabelis 6 toodud lausete keskmised täitmise ajad erinevate andmemahtude korral ning tabelis 23 on selle alusel väljaarvutatud Pearsoni korrelatsioonikordaja väärtused. Keskmine on leitud aritmeetilise keskmisena.

Tabel 22. Keskmised täitmise ajad erinevate andmemahtude korral.

Tootaja ridade arv	**PostgreSQL16**	**Oracle23c**	**MS SQL Server 2022**	**MySQL 8.3**	**MariaDB 11.4**	**DB2 11.5**
100000	330ms	1736,4ms	175,1ms	752,6ms	538ms	249ms
200000	755,4ms	3255,4ms	420,9ms	1267,4ms	505,9ms	302,3ms
400000	1151,9ms	6672,6ms	540,982ms	2607,6ms	2607,3ms	1100,1ms

Tabel 23. Pearsoni korrelatsioonikordaja väärtused täitmisaegade keskmise põhjal.

PostgreSQL16	**Oracle23c**	**MS SQL Server 2022**	**MySQL 8.3**	**MariaDB 11.4**	**DB2 11.5**
0,9779	0,9996	0,9264	0,9981	0,9405	0,9617

5.3 Ankurmudeli tabelite elimineerimise teisenduse tulemused

Jaotises 4.5 näidatud päringute täitmisplaanide uurimise tulemused on väljatoodud tabelisse 24. X näitab, et tabelite elimineerimise teisendus toimus ja eemaldati kõigi tabelite kasutus, mida lause täitmiseks pole vaja. Selle kohta öeldakse, et toimus täielik teisendus. O näitab, et teisendust ei toimunud. V näitab, et toimus osaline teisendus. Lisaks näidatakse lause täitmiseks kasutatud tabelite arvu. Tabel loetakse kasutatuks ka siis, kui lause täitmiseks loetakse ainult sellele loodud indeksit, mitte tabeli andmeid sisaldavaid plokke.

Tabel 24. Ankurmudeli andmebaasis tabelite elimineerimise tulemused.

Lause	**PostgreSQL 16**	**Oracle23c**	**MS SQL Server 2022**	**MySQL 8.3**	**MariaDB 11.4**	**DB2 11.5**
A1	V ja 3	V ja 3	V ja 3	O ja 7	V ja 4	V ja 4
A2	V ja 3	V ja 4	V ja 4	O ja 7	V ja 4	V ja 4
A3	V ja 3	X ja 2	V ja 3	O ja 7	V ja 4	V ja 3
A4	V ja 3	V ja 3	V ja 3	O ja 7	V ja 4	V ja 3

6 Analüüs ja tulemused

Selles peatükis analüüsitakse ja võetakse kokku peatükis 5 välja toodud tulemused.

6.1 Vastused teisenduse tegemise uurimisküsimustele

Selles jaotises vastatakse töö alguses esitatud uurimisküsimustele.

6.1.1 Küsimus 1

Sellel küsimusel oli mitmeid alamküsimusi, millele vastatakse ükshaaval.

Küsimus: Kui hästi oskavad valitud andmebaasisüsteemid tabelite elimineerimise teisendust läbi viia? Milline on nende andmebaasisüsteemide pingerida seda oskust silmas pidades?

Vastus: Tabelis 25 esitatud pingerida on koostatud tabeli 10 põhjal, kus on näha tabelite elimineerimise teisenduse toimumist või mittetoimumist. Mida suurema hulga lausete korral oskab andmebaasisüsteem tabelite elimineerimise teisendust teha, seda eespool see pingereas on. Tulemus tulpdiagrammina esitatakse lisas 7 joonisel 23.

Tabel 25. Andmebaasisüsteemide poolse tabelite elimineerimise teisenduse tegemise võimekuse pingerida tehtud katsete alusel.

Koht	**Andmebaasisüsteem**
1.	Oracle23c Free
2.	DB2 11.5
3.	PostgreSQL16
4.	MS SQL Server 2022 Developer
5.	MariaDB 11.4
6.	MySQL 8.3

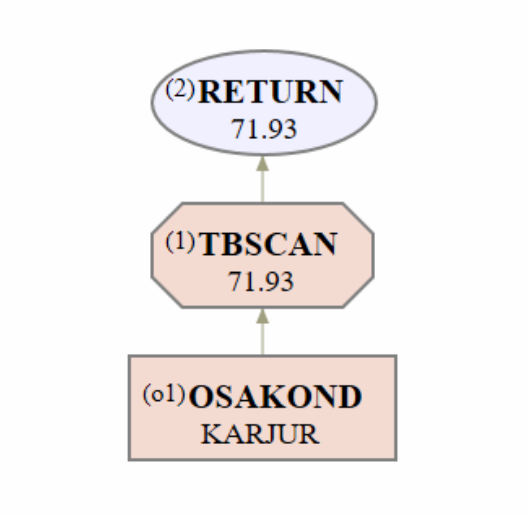
MySQL on viimane, sest see ei oska tabelite elimineerimise teisendust teha.

MS SQL Server on pingereas eespool kui MariaDB, sest täielik tabelite elimineerimise teisendus on autori arvates suurema kaaluga kui MariaDB tehtud osaline. MariaDB elimineerib osalise elimineerimise tagajärjel ainult ühe tabeli kasutuse, kuigi võiks

näiteks lauses S14 elimineerida kahe tabeli kasutuse. Joonisel 9 on näha MariaDB osalist elimineerimist, tabelid O ehk *Osakond* ja A ehk *Asukoht* on täitmisplaanis sees. Jooniselt 10 näeme, et DB2 on elimineerinud ka tabeli *Asukoht*.

	id	select_type	table	type	possible_keys	key	key_len	ref	rows	r_rows	filtered	r_filtered	Extra
1	1	SIMPLE	O	ALL	(NULL)	(NULL)	(NULL)	(NULL)	10028	10000.00	100	100	(EMPTY)
2	1	SIMPLE	A	eq_ref	PRIMARY	PRIMARY	2	mariabaka.O.asukoht_ID	1	1.00	100	100	(EMPTY)

Joonis 9. Lause S14 täitmisplaan MariaDB andmebaasis.



Joonis 10. Lause S14 täitmisplaan DB2 andmebaasis.

Küsimus: Kas andmebaasisüsteemid oskavad tabeli elimineerimise teisendust teha ka sellise andmebaasi korral, mis on loodud ankurmodelleerimist kasutades?

Vastus: Jah, kuid enamasti osaliselt. Ankurmodelleerimise puhul esines rohkem osalist tabelite elimineerimise teisendust. Tabelist 24 näeme, et ainus andmebaasisüsteem, mis suutis teostada mõne lause puhul täielikult tabelite elimineerimise teisendust, oli Oracle. Oracle suutis A3 lause puhul optimeerida nii, et kasutataks minimaalset arvu tabeleid. Kõikide ülejäänud lausete puhul suutsid kõik andmebaasisüsteemid teostada osalist tabelite elimineerimise teisendust. Ainsaks erandiks oli ka siin MySQL, mis ei teostanud ühtegi elimineerimist ja kasutas kõiki tabeleid, mis vaatesse olid ühendatud.

Küsimus: Kas andmebaasisüsteemid oskavad seda teisendust ka muude lause tüüpide kui SELECT korral?

Vastus: Jah, mingil määral oskavad. Kokku oli eksperimendis viis mitte-SELECT lauset. Nendest oli kaks DELETE lauset, kaks UPDATE lauset ja üks INSERT lause. Tabelist

26 on näha, et kõige paremini oskab mitte-SELECT lausete korral teha teisendust Oracle, jäädes hätta ainult DELETE lausega S21. PostgreSQL ei suuda teisendada DELETE lauset S19 ja UPDATE lauset S22. MS SQL Server ei teinud teisendust kahe DELETE lause – S19, S21 ja ühe UPDATE lause - S20 korral. Järgneb DB2, mis suutis tabelite elimineerimise teisendust ainult INSERT lause korral S23. MariaDB suutis kolme lause korral teostada osalist tabelite elimineerimise teisendust, kaotades ära ühe tabeli lugemise. Erandiks oli jälle MySQL, mis ei teostanud mitte ühegi lause puhul tabelite elimineerimise teisendust.

Tabel 26. Tabelite elimineerimise teisenduse katsetamise tulemused andmete muutmise lausete korral.

Tunnus	**PostgreSQL 16**	**Oracle23c**	**MS SQL Server 2022**	**MySQL 8.3**	**MariaDB 11.4**	**DB2 11.5**
S19	O	X	O	-	O	O
S20	X	X	O	O	O	O
S21	X	O	O	O	V	O
S22	O	X	X	O	V	O
S23	X	X	X	O	V	X
X	3	4	2	0	0	1
O	2	1	3	4	2	4
V	0	0	0	0	3	0
-	0	0	0	1	0	0

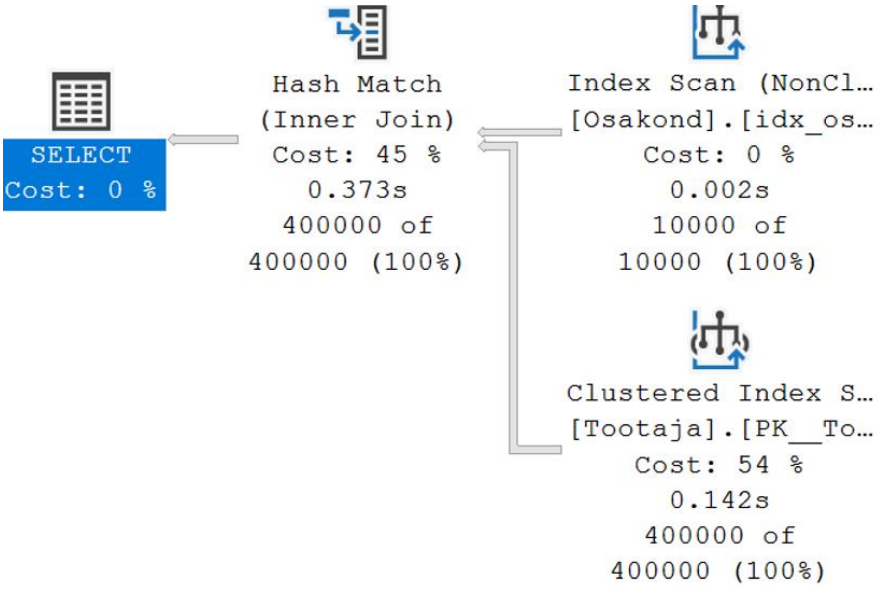
6.1.2 Küsimus 2

Küsimus: Milline on Oracle, MS SQL Serveri ja PostgreSQL puhul olnud selle võimekuse muutus võrreldes eelmise uuringuga kümme aastat tagasi?

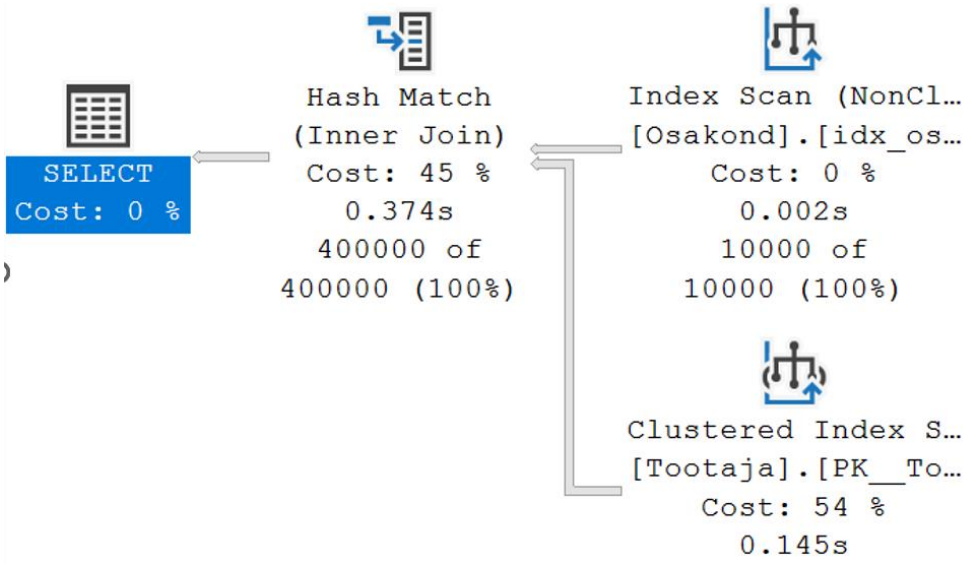
Vastus: Võimekus on kahe süsteemi puhul katsetatud lausete korral jäänud samaks ja ühe puhul langenud. Tabelist 11 näeme, et kõige tugevam tabelite elimineerimise teisenduse teostaja oli kümme aastat tagasi Oracle12c, mis suutis teisendada üheksa lauset 13st. Järgnes MS SQL Server 2012, mis suutis teisendada seitse lauset 13st. Kõige nõrgemat tulemust näitas PostgreSQL (9.3), teisendades ainult kahte lauset. Võrreldes neid tulemusi samade andmebaasisüsteemide uuemates versioonides saadud tulemustega, näeme, et Oracle ja PostgreSQL korral pole samade lausete korral teisenduse tegemises

midagi muutunud. Ainus andmebaasisüsteem, mis näitas muutust oli MS SQL Server. Kui 2012. aasta versioon suutis teostada teisendust seitsme lause puhul, siis 2022. aasta versiooni tulemus on palju nõrgem – langedes samale pulgale PostgreSQL-iga, ehk teostas tabelite elimineerimise teisendust ainult kahe lause korral.

Näiteks S1 ja S2 puhul sai MS SQL Server 2012 versioon aru, et vaatest küsitakse ainult tabeli *Tootaja* andmeid ning elimineeris plaanidest tabeli *Osakond* kasutamise. Joonistelt 11 ja 12 näeme, et lausete S1 ja S2 korral ei teosta MS SQL Server 2022 teisendust, sest täitmisplaanis on näha tabelite *Tootaja* ja *Osakond* ühendamist.



Joonis 11. Lause S1 täitmisplaan MS SQL Server 2022 andmebaasis.



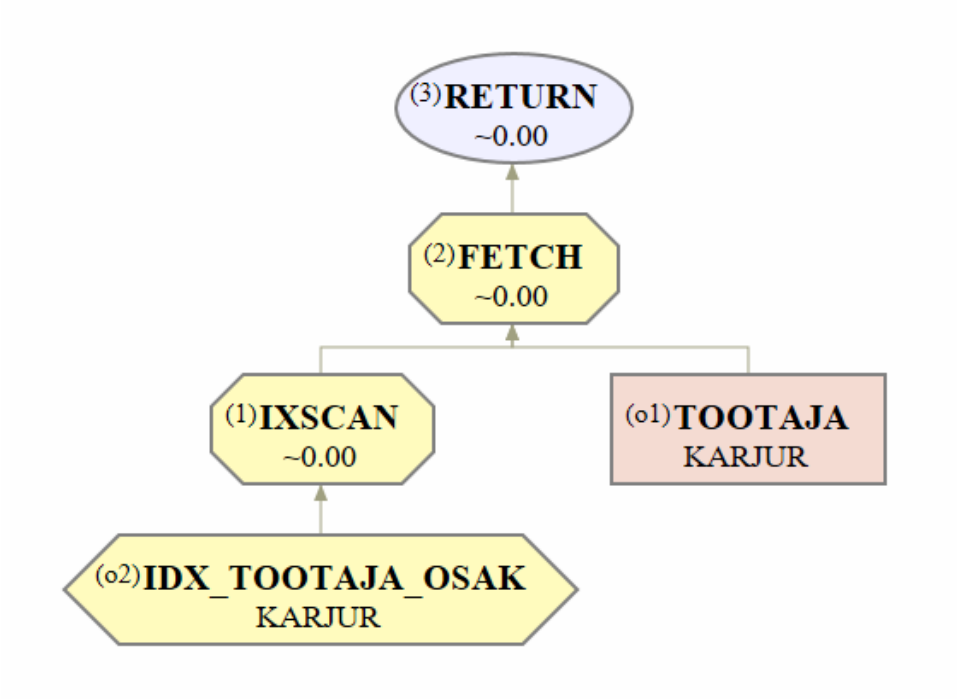
Joonis 12. Lause S2 täitmisplaan MS SQL Server 2022 andmebaasis.

6.1.3 Küsimus 3

Küsimus: Kas see kui tabelid on tühjad *vs.* see, et tabelites on andmeid, mõjutab seda, kas andmebaasisüsteem otsustab tabelite elimineerimise teisendust teha?

Vastus: Tabelite elimineerimise teisenduse rakendamisel ei olnud vahet kas tabelites on andmed sees või on tabelid tühjad. Ainsad muutused toimusid täitmisplaani sees. Näiteks DB2 puhul otsiti tühjadest tabelitest läbi indeksite, aga andmetega tabelite puhul oli täitmisplaanis otsing otse tabelist.

Joonisel 8 on näha DB2 täitmisplaani lause S1 korral kui tabelis on andmed. DB2 otsib sellisel juhul andmeid otse tabelist. Joonisel 13 on näha, et tühjade tabelite korral on täitmisplaanis otsing läbi indeksi. Antud juhul toimus tabelite elimineerimise teisendus ja andmebaasisüsteem luges ainult tabeli *Tootaja* andmeid (kas otse või läbi indeksi).



Joonis 13. Lause S1 täitmisplaan tühjade tabelite korral DB2 andmebaasis.

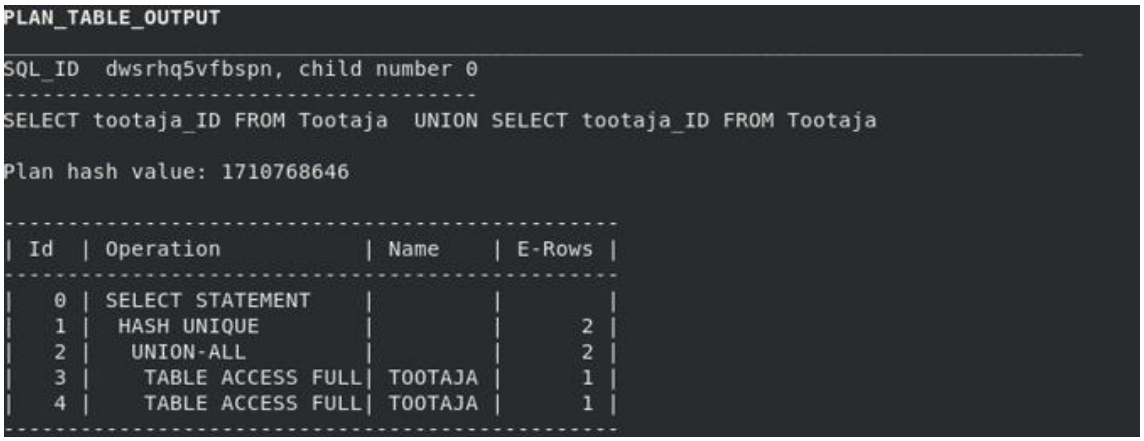
Oracles oli aga vastupidi. Jooniselt 14 näeme, kuidas lause S8 korral kasutatakse siis, kui tabelites on andmeid, lause täitmiseks ainult tabelile loodud indeksit. Jooniselt 15 näeme, kuidas tühjade tabelite korral asendub indeksite lugemine tabeliplokkide lugemisega. Antud juhul ei toimunud tabelite elimineerimise teisendust ja andmebaasisüsteem luges tabeli *Tootaja* andmeid kahekordselt (kas otse või läbi indeksi).

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------
SQL_ID  0rmdqp1ydd0hh, child number 0
-------------------------------------
SELECT tootaja_ID FROM Tootaja  UNION SELECT tootaja_ID FROM Tootaja

Plan hash value: 1642322779

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name       | E-Rows |  OMem |  1Mem | Used-Mem |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |            |        |       |       |          |
|   1 |  HASH UNIQUE          |            |    213K|  5838K|  2749K|   10M (0)|
|   2 |   UNION-ALL           |            |    213K|       |       |          |
|   3 |    INDEX FAST FULL SCAN| PK_TOOTAJA |    106K|       |       |          |
|   4 |    INDEX FAST FULL SCAN| PK_TOOTAJA |    106K|       |       |          |
-------------------------------------------------------------------------------------------
```

Joonis 14. Lause S8 täitmisplaan andmetega Oracle andmebaasis.



Joonis 15. Lause S8 täitmisplaan tühjade tabelite korral Oracle andmebaasis.

Kokkuvõttes, andmete olemasolu või puudumine tabelites ei mõjutanud testitud andmebaasisüsteemide korral elimineerimise teisenduse tegemise valikut.

6.2 Vastused töökiiruse uurimisküsimustele

Selles jaotises vastatakse uurimisküsimustele, mis on seotud andmekäitluse operatsioonide töökiirusega.

6.2.1 Küsimus 4

Küsimus: Milline on operatsioonide töökiiruse erinevus MySQL ja MariaDB andmebaasides kui tehakse operatsioone, mille korral MariaDB teeb tabelite elimineerimise teisendust, kuid MySQL ei tee?

Vastus: Tabelist 10 näeme, et on kaks sellist lauset, kus MariaDB teeb täiel määral tabelite elimineerimise teisendust, aga MySQL ei tee. Nendeks lauseteks on S3* ja S7. Lisaks on laused S14, S16, S21, S22, S23, S24 ja S25 kus MariaDB teeb osaliselt tabelite elimineerimist.

Kuna lausega S3* otsitakse andmeid ainult tabelitest *Osakond* ja *Asukoht* ning nende tabelite ridade arvud ei muutu erinevate andmemahtudega tehtavate katsetuste käigus, siis saab neid lauseid võrrelda ainult ühe mahu korral. Tabelist 14 näeme, et kui MariaDB keskmine töökiirus on lause S3* puhul 50ms, siis MySQLi keskmine oli 57,6ms.

Lause S7 puhul on tabeli *Tootaja* ridade arv oluline. Tabelite elimineerimisel suudab MariaDB kaotada enda täitmisplaanist *Tootaja* lugemise, aga MySQL-i puhul jääb tabeli

Tootaja lugemine täitmisplaani sisse. Tabelist 12 näeme, et kui MariaDB puhul jäävad töökiiruste keskmised olenemata ridade arvust väikestest, siis MySQL-is on esimese ridade hulga puhul töökiirus juba suurem ning kasvab ridade arvu kasvades ka edasi.

Tabelist 15 näeme, et S14 puhul on MariaDB 24,4ms kiirem kui MySQL.

Tabelist 16 näeme, et MariaDB on kõikide andmemahtude puhul kiirem kui MySQL.

Tabelist 18 näeme, et S21 ja S22 puhul on MySQL esimese andmemahu juures kiirem kui MariaDB. S23 juures on töökiirused võrdsed. Teise andmemahu juures on, aga MariaDB kõikide lausete puhul kiirem kui MySQL. Kolmanda andmemahu juures on MariaDB lausete S21 ja S23 puhul kiirem. Erandiks on S22, kus MySQL teeb töö ära kiiremini.

Kokkuvõtteks saab öelda, et kõikide SELECT lausete korral, kus MariaDB teeb tabelite elimineerimise teisendust, on MariaDB töökiirus parem kui MySQLi. Mitte-SELECT lausete korral on MariaDB töökiirus parem peale esimest andmemahu suurendamist, jäädes alla ainult MySQLile alla ainult ühe lausega kõige suurema ridade arvu korral.

6.2.2 Küsimus 5

Küsimus: Kui võrrelda MySQL ja MariaDB andmebaasides tehtud operatsioone, siis kas tabelite elimineerimise tegemine *vs*. mittetegemine mõjutab seda, kas andmemahtude ja töökiiruse vahel on kasvav lineaarne seos või mitte?

Vastus: Tabelist 17 näeme, et MySQL-is on lause S15 puhul Pearsoni korrelatsioonikordaja väärtus -0,2876 ja lause S16 puhul -0,2761. Sellest saab järeldada, et lausete S15 ja S16 puhul on MySQL-is nõrk lineaarne kahanev seos andmemahtude ja lausete täitmisaja vahel (andmemahu kasvades täitmisaeg väheneb). MariaDB puhul näeme, et lause S15 puhul on Pearsoni korrelatsioonikordaja väärtus 0,8869 ja lause S16 puhul 0,8947. Need väärtused viitavad juba andmemahtude ja lausete täitmisaja tugevale kasvavale seosele (andmemahu kasvades täitmisaeg kasvab).

Lausete S21 ja S22 kohta näeme tabelist 19, et nii MySQL-i kui ka MariaDB puhul on Pearsoni kordajate väärtused peaaegu võrdsed ja viitavad, et andmemahtude ja täitmisaja vahel on väga tugev kasvav seos. Lause S23 korral on samuti mõlema

andmebaasisüsteemi puhul näha väga tugevat kasvavat seost, kuid siin on MySQL-i kordaja väärtus kõrgem. MySQL-il on see 0,9982 ja MariaDB-l 0,9103.

Tabelist 21 näeme, et MySQLi puhul on lausete S24 ja S25 Pearsoni korrelatsioonikordaja väärtused negatiivsed, sest andmehulga kasvades täitmise aeg hoopis väheneb (e töökiirus kasvas). S24 korral on seos nõrk ja S25 korral mõõdukas. MariaDB puhul on S24 ja S25 korrelatsioonikordaja väärtused vastavalt 0,8281 ja 0,8669, mis näitavad tugevat kasvavat seost andmemahtude ja täitmise aja vahel.

Kokkuvõtteks saab öelda, et mitte-SELECT lausete puhul näitavad mõlemad andmebaasisüsteemid, olenemata tabeli elimineerimise teisenduse tegemisest, andmemahtude ja täitmisaja vahel väga tugevat kasvavat seost.

MySQL-i puhul, mis ei tee tabeli elimineerimise teisendust, on SELECT lausete korral seos andmemahtude ja täitmisaja vahel palju nõrgem kui MariaDB-l. Kummaline oli see, et MySQL andis suuremate andmemahtude puhul kiiremaid vastuseid kui väiksemate mahtude puhul.

6.2.3 Küsimus 6

Küsimus: Kui konkreetse andmebaasisüsteemi puhul lahendatavas ülesandes on kaks lihtsustamata lauset, mis annavad loogiliselt samaväärse tulemuse, kuid ühe korral tehakse andmebaasisüsteemi poolt teisendus ja teise korral mitte, siis kui palju mõjutab teisenduse tegemine lause täitmise kiirust?

Vastus: Laused S3 ja S3* annavad mõlemad sama tulemuse. Tabelist 15 näeme, et PostgreSQL ja MS SQL Server näitavad kiiremaid keskmiseid töökiiruseid teostades tabelite elimineerimise teisendust võrreldes juhuga kui seda ei teostata. MariaDB puhul ei ole lausete töökiiruste vahe märgatav, kuid tabelite elimineerimise teisenduse korral on töökiirus veidi väiksem kui teisendust mitte tehes.

Tabelist 15 näeme lauseid S13 ja S14. MariaDB puhul ei ole töökiirustes vahet kui andmebaasisüsteem teostab tabelite elimineerimise teisendust või ei teosta. MS SQL Serveri puhul on tabeli elimineerimise teisenduse korral lause täitmine natukene kiirem. Küll, aga on ajakulu mõlema lause korral suhteliselt väike. Tabelite elimineerimise puhul on ajakulu 24,7ms *vs.* 33,4ms kui teisendust ei tehta. PostgreSQL puhul on näha, et tabelite elimineerimise teisenduse toimimisel täidab andmebaasisüsteem lause kiiremini.

Siin on vahe natukene suurem, tabeli elimineerimise teisenduse tegemine annab 23,7ms võitu, kuid vahe on siiski üsna väike.

Tabelis 16 näeme lausete S15 ja S16 töökiiruseid. DB2 puhul on näha, et tabelite elimineerimisel tehakse töö ära mitu korda kiiremalt. Siiski peab märkima, et DB2 saab mõlemal juhul tööga väga kiiresti hakkama ja märgatavat vahet ei ole.

Oracle puhul on näha vahet esimese kahe ridade hulga korral kui tabelite elimineerimise teisenduse tegemine annab eelise. Kõige suurema ridade hulga korral tööajad võrdsustuvad ja kiiremini tehakse töö ära juhul kui ei teostata tabelite elimineerimise teisendust.

MS SQL Serveri puhul on tööajad mõlema lause puhul sarnased ja ei ole töökiiruse mõttes vahet kas tabelite elimineerimise teisendus toimub või mitte.

MariaDB on lausete S15 ja S16 puhul erandiks, sest töö tehakse kiiremini ära kui tabelite elimineerimist ei toimu.

PostgreSQL puhul on esimese ridade hulga korral näha, et tabelite elimineerimise korral võtab lause täitmine kauem aega võrreldes olukorraga kui ei tehta elimineerimist. Järgneva kahe suurema ridade hulga korral on aga juba kindel eelis tabelite elimineerimise tegemisel, sest lausete täitmise ajad on tabelite elimineerimise teisenduse korral peaaegu kaks korda väiksemad.

Neid tulemusi vaadeldes tuleb muidugi arvestada ka seda, et tabelite elimineerimise teisenduse tegemine või mittetegemine pole ainus asi, mis lausete täitmise aega mõjutab.

Küsimus: Kui konkreetse andmebaasisüsteemi puhul lahendatavas ülesandes on kaks lihtsustamata lauset, mis annavad loogiliselt samaväärse tulemuse, kuid ühe korral tehakse andmebaasisüsteemi poolt teisendus ja teise korral mitte, siis millal muutub teisenduse tegemise mõju töökiirusele märgatavaks?

Vastus: Tabeli 16 korral on näha, et PostgreSQL näitel on kõige suurem vahe näha suuremate mahtude korral. Oracle andmebaasisüsteemi korral on, aga vahe just väiksemate mahtude korral, suuremate mahtude korral juba töökiirused võrdsed. Kõikide ülejäänud lausete ja andmebaasisüsteemide puhul ei ole antud juhul näha, et tabeli elimineerimise teisenduse tegemine vähendaks märgatavalt täitmise aega.

Kokkuvõtteks tuleb öelda, et tabelite elimineerimise teisenduse tegemine ei ole võluvits, mis andmekäitlusoperatsioonide läbiviimist märgatavalt kiirendaks. Pigem on tegemist turvavõrguga, mis tasandab mingil määral SQL lausete kirjutajate poolt ettevaatamatusest liiga keeruliseks muudetu lause täitmisest tulenevaid töökiiruse probleeme.

6.2.4 Küsimus 7

Küsimus: Milline on nende andmebaasisüsteemide töökiiruste pingerida erinevate andmehulkade puhul kõikide operatsioonide keskmist kiirust arvestades?

Vastus: Pingerida on tabelis 27. Pingerida on koostatud tabeli 22 põhjal. Oracle puhul peab arvestama, et eksperiment viidi läbi virtuaalmasinal.

Tabel 27. Pingerida keskmiste töökiiruste põhjal.

Koht	**100 000 rida tabelis** ***Tootaja***	**200 000 rida tabelis** ***Tootaja***	**400 000 rida tabelis** ***Tootaja***
1.	MS SQL Server 2022	DB2 11.5	MS SQL Server 2022
2.	DB2 11.5	MS SQL Server 2022	DB2 11.5
3.	PostgreSQL16	MariaDB 11.4	PostgreSQL16
4.	MariaDB 11.4	PostgreSQL16	MariaDB 11.4
5.	MySQL 8.3	MySQL 8.3	MySQL 8.3
6.	Oracle23c	Oracle23c	Oracle23c

Küsimus: Kas kõikide operatsioonide töökiiruse keskmist arvestades on igas konkreetses süsteemis andmemahtude ja töökiiruse vahel kasvav lineaarne seos või mitte?

Vastus: Tabelist 24 näeme, et kõikide andmebaasisüsteemide puhul on tegemist väga tugeva kasvava lineaarse seosega. Kõige tugevam lineaarne seos on Oracle ja MySQLi puhul. Kõige väiksem seos on MS SQL Serveril, kuid Pearsoni kordaja väärtus 0,9264 näitab ikkagi väga tugevat seost.

6.3 Võrdlus olemasolevate uuringutega

Käesoleva tööga kõige sarnasem uuring tehti aastal 2017, kus uuriti peaaegu samu andmebaasisüsteeme, kuid varasemaid versioone. Ainus andmebaasisüsteem, mis puudus oli MariaDB. Antud uuringus testiti nelja tüüpi andmete otsimise lauseid, mis olid

vastavalt lausetes kasutatud tabelite ühendamise viisile INNER JOIN, INNER JOIN nullable, OUTER JOIN ja OUTER JOIN DISTINCT.[26]

2017. aasta uuringus suutis DB2 LUW 10.5 teostada tabeli elimineerimise teisendust kõigi nelja lausetüübi puhul. Oracle 12.2.0.1 ja MS SQL Server 2014 suutsid teostada tabeli elimineerimise teisendust kolme lausetüübi puhul. Oracle jäi hätta OUTER JOIN DISTINCT lausetüübiga ja MS SQL Server ei teinud teisendust INNER JOIN nullable lausetüübi korral. PostgreSQL 9.6 suutis tabeli elimineerimise teisendust teha ainult OUTER JOIN lausetüübi korral. MySQL 8.0.2 ei teinud mitte ühegi lausetüübi korral teisendust.

Võrreldes kahte uuringut on näha palju ühist. Sarnaselt 2017. aasta uuringuga näitavad bakalaureusetöö eksperimendid, et MySQL ei teinud endiselt mitte ühegi lause puhul tabelite elimineerimist. PostgreSQL16 puhul on samuti näha, et sarnaselt PostgreSQL 9.6-ga tehakse teisendust OUTER JOIN operaatorit kasutava lausete puhul. Võrreldes kahe uuringu pingeridasid, siis on näha kahte muutust. Kui 2017. aastal oli esimene DB2 ja teist/kolmandat kohta jagas Oracle, siis antud eksperimendis oli vastupidi. Kui 2017. aastal jagas teist/kolmandat kohta MS SQL Server ja neljas oli PostgreSQL, siis antud eksperimendis saavutas kolmanda koha PostgreSQL ja neljanda koha MS SQL Server.

Käesoleva töö eksperimendis ei testitud ühte lause tüüpi, mida katsetati 2017. aasta uuringus. Joonisel 16 esitatud kaks lauset on loogiliselt samaväärsed.

```
SELECT DISTINCT nimi FROM Osakond LEFT JOIN Tootaja ON
Osakond.osakond_id=Tootaja.osakond_id;
SELECT DISTINCT nimi FROM Osakond;
```

Joonis 16. OUTER JOIN DISTINCT tüüpi lause ja selle lihtsustamise tulemus.

Proovides järgi, millised käesoleva töö andmebaasisüsteemid sellist tabeli elimineerimise teisendust tegid, oli tulemus selline nagu näitab tabel 28. X näitab, et tabelite elimineerimise teisendus toimus ja eemaldati kõigi tabelite kasutus, mida lause täitmiseks pole vaja. Selle kohta öeldakse, et toimus täielik teisendus. O näitab, et ei toimunud. Tabelist näeme, et andmebaasisüsteemide uuemate versioonide puhul ei ole muutusi näha.

Tabel 28. OUTER JOIN DISTINCT tüüpi lause tabeli elimineerimise teisenduse tulemused.

PostgreSQL 16	**Oracle23c**	**MS SQL Server 2022**	**MySQL 8.3**	**MariaDB 11.4**	**DB2 11.5**
O	O	X	O	O	X

6.4 Töö tegemise reflektsioon

Selles jaotises kirjutatakse, mis läks töö tegemisel hästi, mis halvasti, kellel oleks tööst kasu ning mida sellest kõigest õppida.

6.4.1 Asjad, mis läksid töö tegemisel hästi

Töö tegemisel läks hästi uute andmebaasisüsteemide kasutusele võtmine ja nende kasutamine. Autoril oli küll varasem kogemus SQL-iga olemas, kuid suures pildis eksperimendis kasutatavate andmebaasisüsteemidega kogemus puudus. Oracle23c Free installeerimisel tekkinud probleemid ja mured, et Windowsile otse installeerimise puudub ja peab kasutama virtuaalmasinat ning käske edastama käsurealt, said kiiresti lahendatud ja edasised probleemid puudusid.

6.4.2 Asjad, mis läksid töö tegemisel halvasti

Töö tegemisel läks nii mõndagi halvasti. Kõige suuremaks veaks saab lugeda liigset kiirustamist eksperimentide tegemisel. Esiteks soovis autor kohe hakata programmeerima ilma piisavalt läbi mõtlemata, milliseid lauseid võiks katsetada. Seetõttu jäi algsesse eksperimenti kaasamata vähemalt üks lausetüüp. Samuti tekkis probleem töökiiruste mõõtmistel. Varasem kogemuse puudumine andmebaasisüsteemides töökiiruste mõõtmistel tõi esile puuduse teadmistes. Peale andmete importimist ehk andmemahtude muutumisel oleks pidanud tabelite statistika värskendama, et andmebaasisüsteemil oleks värsked andmed tabelites olevate ridade hulga kohta. See viga ilmnes peale kahe esimese andmebaasisüsteemi töökiiruste mõõtmist ehk nende puhul pidi tegema topelttööd, mis tähendas suuremat ajakulu kui algselt eeldati.

6.4.3 Tööst huvitatud osapooled

Tööst võiksid olla huvitatud kõik andmetega tegelejad nagu näiteks erinevad ettevõtted ja andmebaasidega töötavad analüütikud, arendajad ning administraatorid. Huvitatud võiksid olla nii juba eksisteerivate andmebaaside kasutajad kui ka need, kes alles valivad

millist andmebaasisüsteemi kasutada või kuidas oma andmebaasi ja andmekäitluse lauseid kujundada. See töö annab ülevaate ja praktilisi teadmisi ning ideid, mis võib aidata teha vastavalt olukorrale kaalutletumaid otsuseid. Ühtlasi annab töö õpetust kõigile SQL lausete kirjutajatele selles mõttes, et andmebaasisüsteem võib kuid ei pruugi osata andmebaasikeele lauset lihtsustada ning liigse keerukuseta lause kirjutamine on seega ka SQL lausete kirjutajate oluline vastutus.

6.4.4 Asjad, mida tööd uuesti tehes tuleks teisiti teha

Tehes uuesti sama põhimõttega tööd, siis kindlasti tuleks planeerida eksperimentide tegemist hoolikamalt. Töökiiruse uurimise käigus selgus, et oleks võinud kohe mõelda sellele, mis lausete puhul töökiirused on olulised. Kui lauseid on kokku 27 tükki, siis *Tootaja* ridade arvu muutumine mõjutab ainult 19 lauset. Eksperimentide käigus mõõdeti aga peale iga andmemahu muutust ikkagi uuesti ka neid lauseid kus tegelikult ridade arv ei muutunud. See lisas tööle ebavajalikku ajakulu, mille oleks saanud pühendada analüüsi tegemisele.

6.5 Edasised uurimissuunad

Edasisteks uurimissuundadeks saaks olla andmebaasisüsteemide valimi laiendamine. Veel oleks võimalik edasi uurida lausete täitmise kiirust ankurmodelleerimise abil loodud andmebaasides. Optimeerimise ja töökiiruse poole pealt oleks võimalik uurida ka erinevate NoSQL andmebaasisüsteemide võimekusi ning neid omavahel võrrelda.

7 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli uurida mõnede kõige populaarsemate SQL-andmebaasisüsteemide tabelite elimineerimise teisenduse tegemise võimekust ning erinevate andmebaasisüsteemides toimuvate operatsioonide keskmiseid töökiiruseid. Lisaks võrreldakse tulemusi 2014. aastal tehtud bakalaureusetööga [4], et näha kas võimekus on aja jooksul muutunud.

Eksperimendi käigus testiti kuuel andmebaasisüsteemil 27 lauset, et koostada pingerida tabelite elimineerimise teisenduse tegemise võimekuse järgi ja 19 lauset, et koostada pingerida keskmiste töökiiruste järgi. Töökiiruseid testiti kolme erineva andmemahuga, et lisaks uurida seoseid andmemahtude kasvu ja töökiiruse vähenemise vahel. Andmebaasisüsteemideks olid PostgreSQL16, Orace23c Free, MySQL 8.3, MariaDB 11.4, MS SQL Server 2022 Developer ja DB2 11.5.

Kõige edukamalt suutis tabelite elimineerimise teisendust teha Oracle23c. 27st lausest 21 puhul teostas Oracle tabelite elimineerimise teisendust. Järgnes DB2 11.5, mis suutis teostada teisendust 18 lause puhul. Ülejäänud andmebaasisüsteemid suutsid teostada tabelite elimineerimise teisendust vähem kui kümne lause korral. MySQLi puhul oli juba eelnevalt kirjandusest teada, et see tabelite elimineerimise teisendust ei tee. Lisaks uuriti tabelite elimineerimise teisenduse tegemist ankurmodelleerimist kasutades loodud andmebaasi vaate põhjal ja leiti, et teisendus enamasti toimub, kuid see on osaline (st kasutatavate tabelite hulka ei viida miinimumini).

Keskmiste töökiiruste pingereas kerkis esile MS SQL Server 2022, mis oli esimese ja kolmanda andmemahu juures kõige kiirem andmebaasisüsteem. Esimeses ja kolmandas mahus jäi teiseks DB2 11.5 ja kolmandaks PostgreSQL 16. Teise andmemahu juures oli kõige kiirem DB2 11.5, teiseks jäi MS SQL Server 2022 ja kolmandaks MariaDB 11.4. Kõik andmebaasisüsteemid näitasid väga tugevad lineaarset kasvavat seost andmemahtude ja täitmisaja vahel.

Töö tulemusena saab öelda, et andmebaasisüsteemi poolt tabelite elimineerimise teisenduse tegemine võib küll kiirendada andmekäitlusoperatsioonide läbiviimist, aga see vahe ei ole alati märgatav. Tabeli elimineerimise teisendus on kasulik erinevate baastabelite andmeid koondavatest vaadetest päringute tegemisel ja see aitab ka

vähendada kasutajapoolseid ettevaatamatusest tehtuid vigu, kus liiga keeruliseks muudetud lause muudab töökiiruse aeglaseks.

Eksperimentide jaoks kirjutatud kood on leitav GitHubi salvest https://github.com/Karjur/JoinElimination

Kasutatud kirjandus

[1] “DB-Engines Ranking.” - DB-Engines.com. Accessed: Feb. 29, 2024. [Online.] Available: https://db-engines.com/en/ranking

[2] S.Cass, “The Top Programming Languages 2023.”, IEEE.org, Accessed May, 19. 2024. [Online.] Available: https://spectrum.ieee.org/the-top-programming-languages-2023

[3] “Kvantitatiivsed uurimismeetodid”, et.wikipedia.org. Accessed May, 19. 2024. [Online.] Available: https://et.wikipedia.org/wiki/Kvantitatiivsed_uurimismeetodid

[4] K.Pukk, “Tabelite elimineerimise teisenduse rakendamine erinevate SQL-andmebaasisüsteemide poolt”, B.Sc thesis, School of Information, Tallinn University of Technology, Tallinn, Estonia, 2014.

[5] K.Karuauk, “Ankurmudeli põhjal PostgreSQL jaoks SQL koodi koostava generaatori Edasiarendamine”, M.S thesis, School of Information, Tallinn University of Technology, Tallinn, Estonia, 2023.

[6] Oracle Database 23c Free - Oracle Database. Accessed: Mar. 13, 2024. [Online.] Available: https://docs.oracle.com/en/database/oracle/oracle-database/23/index.html

[7] PostgreSQL 16.2 Documentation - PostgreSQL Documentation. Accessed: Mar. 14, 2024. [Online.] Available: https://www.postgresql.org/docs/16/

[8] SQL Server technical documentation - Microsoft Learn. Accessed: Mar. 14, 2024. [Online.] Available: https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16

[9] MySQL 8.3 Release Notes - MySQL Documentation. Accessed: Mar. 14, 2024. [Online.] Available: https://dev.mysql.com/doc/relnotes/mysql/8.3/en/

[10] MariaDB 11.4.0 Release Notes - MariaDB Knowledgebase. Accessed: Mar. 14, 2024. [Online.] Available: https://mariadb.com/kb/en/mariadb-11-4-0-release-notes/

[11] Db2 11.5 - IBM Db2 documentation. Accessed: Mar. 14, 2024. [Online.] Available: https://www.ibm.com/docs/en/db2/11.5

[12] What is Beekeeper Studio? - Beekeeper Studio. Accessed: Mar. 14, 2024. [Online.] Available: https://docs.beekeeperstudio.io/

[13] Correlation Coefficient Calculator. Statistics Kingdom. Accessed May. 19, 2024 [Online.] Available: https://www.statskingdom.com/correlation-calculator.html

[14] R. Tiffany, “Data manipulation language,” in SQL Server CE Database Development with the .NET Compact Framework. Apress, Berkeley, CA, 2003, doi: https://doi.org/10.1007/978-1-4302-0785-6_62003

[15] M.L.Rupley Jr., “Introduction to Query Processing and Optimization”, 2008 [Online.] Available: https://clas.iusb.edu/computer-science-informatics/research/reports/TR-20080105-1.pdf

[16] H. Dombrovskaya, B. Novikov, A. Bailliekova, “Theory: Yes, We Need It!” in PostgreSQL Query Optimization: The Ultimate Guide to Building Efficient Queries, Apress, Jan, 2024, doi: https://doi.org/10.1007/979-8-8688-0069-6_2

[17] “What is Table Elimination?” – MariaDB Knowledgebase. Accessed: Feb. 29, 2024. [Online.] Available: https://mariadb.com/kb/en/what-is-table-elimination/

[18] Ghazal, A., Crolotte, A. and Bhashyam, R., “Outer join elimination in the RDBMS”, In

International Conference on Database and Expert Systems Applications, (pp. 730-740). Berlin, Heidelberg: Springer Berlin Heidelberg, Aug, 2004.

[19] Cheng, Q., Gryz, J., Koo, F., Leung, T.C., Liu, L., Qian, X. and Schiefer, B., "Implementation of two semantic query optimization techniques in DB2 universal database", In VLDB (Vol.99, pp. 687-698). Sep, 1999.

[20] L. Rönnbäck, O. Regardt, M. Bergholtz, P. Johannesson ja P. Wohed, "Anchor Modeling - An Agile Modeling Technique Using the Sixth Normal Form for Structurally and Temporally Evolving Data", Conceptual Modeling - ER 2009

[21] L. Rönnbäck, O. Regardt, M. Bergholtz, P. Johannesson ja P. Wohed, „Anchor modeling —Agile information modeling in evolving data environments," Data & Knowledge Engineering, 2010.

[22] K.Rootalu, "Korrelatsioonikordajad." samm.ut.ee. Accessed: May. 19, 2024. [Online.] Available: https://samm.ut.ee/korrelatsioonikordajad/

[23] P.Schober, C.Boer, and L.A.Schwarte, "Correlation Coefficients: Appropriate Use Interpretation", Anesthesia & Analgesia 126(5):p 1763-1768, May 2018. [Online.] Available: https://journals.lww.com/anesthesiaanalgesia/fulltext/2018/05000/correlation_coefficients__appropriate_use_and.50.aspx

[24] 9.25. Set Returning Functions - PostgreSQL Documentation. Accessed: May. 4, 2024. [Online.] Available: https://www.postgresql.org/docs/current/functions-srf.html

[25] Random Data Generator and API Mocking Tool - Mockaroo. Accessed: May. 4, 2024. [Online.] Available: https://www.mockaroo.com/

[26] L.Eder, "JOIN Elimination: An Essential Optimiser Feature for Advanced SQL Usage". Accessed: May. 19, 2024. [Online.] Available: https://blog.jooq.org/join-elimination-an-essential-optimiser-feature-for-advanced-sql-usage/

Lisa 1 - Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Karmo Jürgenson

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Tabelite elimineerimise teisenduse rakendamine erinevate SQL-andmebaasisüsteemide poolt ja selle muutused ajas" , 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.

20.05.2024

Lisa 2 - Esimese andmemahu töökiiruste mõõtmiste koondtulemused

Tabel 29. Esimese andmemahu töökiiruste mõõtmiste koondtulemused.

Lause	**DB2 11.5**	**Oracle23c**	**MS SQL Server 2022**	**MariaDB 11.4**	**PostgreSQL 16**	**MySQL 8.3**
S1	1ms	1444ms	168,333ms	148,333ms	232,667ms	600ms
S2	6ms	976,333ms	140ms	140ms	193ms	503,667ms
S3	4ms	338ms	30,333ms	42ms	176,6ms	59ms
S3*	2ms	106,667ms	23,667ms	50ms	124ms	57,6ms
S4	1ms	1519,667 ms	165ms	364,333ms	136,667ms	599,667ms
S5	9,2ms	4,8ms	18,667ms	123,9ms	194,4ms	49,8ms
S6	1,2ms	205,8ms	38,1ms	61,8ms	156,9ms	70,6ms
S7	26,667ms	434,667ms	48ms	34,333ms	106,667ms	250,333ms
S8	12,667ms	1201ms	140ms	205ms	127,667ms	523ms
S9	11ms	1096ms	110,667ms	211,667ms	127ms	614,667ms
S10	5,667ms	1128ms	109,333ms	336,667ms	117ms	571,333ms
S11	1ms	1144,333 ms	93,333ms	310,333ms	129,333ms	607,667ms
S12	16ms	941,667ms	234ms	295ms	109ms	622,333ms
S13	3,2ms	85,1ms	33,4ms	44,6ms	133,6ms	111,2ms
S14	1,6ms	90,9ms	24,667ms	45,4ms	109,8ms	69,8ms
S15	1,667ms	689ms	143ms	102,333ms	118,667ms	511ms
S15 ilma FK	7ms	1123,333 ms	-	-	-	-
S16	2,333ms	716,333ms	139,667ms	119,667ms	126ms	535,333ms
S17	1,4ms	120,1ms	23,333ms	63,1ms	117,333ms	96,4ms
S18	3,1ms	77,7ms	30,6ms	63,4ms	104,9ms	241,8ms
S19	1200ms	8837,333 ms	652,667ms	1605ms	216ms	-

Lause	**DB2 11.5**	**Oracle23c**	**MS SQL Server 2022**	**MariaDB 11.4**	**PostgreSQL 16**	**MySQL 8.3**
S20	526,6ms	1178ms	269,333ms	1570,667 ms	2288,667ms	1958,667 ms
S21	1544,667 ms	7945,667 ms	291,333ms	1330ms	461,667ms	1286,667 ms
S22	1053,333 ms	1127ms	167ms	1796,667 ms	1171ms	1362,667 ms
S23	312,667 ms	265ms	138,333ms	1222,667 ms	169,333ms	1226,333 ms
S24	2ms	788,667ms	106,667ms	116,667ms	110,333ms	510,667ms
S25	1ms	740ms	105ms	112,333ms	127,333ms	623,667ms
S26	15,66ms	819,333ms	105,333ms	209,667ms	201,667ms	638,667ms

Lisa 3 - Teise andmemahu töökiiruste mõõtmiste koondtulemused

Tabel 30. Teise andmemahu töökiiruste mõõtmiste koondtulemused.

Lause	**DB2 11.5**	**Oracle23c**	**MS SQL Server 2022**	**MariaDB 11.4**	**PostgreSQL 16**	**MySQL 8.3**
S1	3ms	3555,33ms	260,667 ms	348,667ms	151ms	374,333ms
S2	1ms	2585,333ms	197ms	229ms	133,667ms	455,333ms
S4	1,333ms	3381ms	284,333 ms	236ms	208ms	454,667ms
S7	30,333ms	400,333ms	29,333ms	23,333ms	123ms	340ms
S8	26,333ms	2754,333ms	948,667 ms	148,667ms	354,333ms	305,667ms
S9	1,667ms	1792,667ms	156ms	158,333ms	452,333ms	554ms
S10	1ms	1860ms	1922ms	213ms	171,333ms	451ms
S11	1,333ms	1829,333ms	183ms	218,667ms	165,333ms	448,667ms
S12	13,667ms	2008,333ms	65,333ms	205,333ms	319ms	785ms
S15	1,33ms	1921,333ms	163,667 ms	68ms	265,333ms	276ms
S15 ilma FK	9,667ms	2606,667ms	-	-	-	-
S16	1ms	1998,667ms	160,667 ms	78ms	151ms	328ms
S19	1284,667 ms	14192,333 ms	807,667 ms	1231,333 ms	953,333ms	-
S20	541ms	1854ms	396,667 ms	1283,667 ms	5146,s	4856,333 ms
S21	1393,667 ms	13274ms	828ms	2057,667 ms	843,333ms	3369,333 ms
S22	1887,333 ms	2371,667ms	387ms	1249,333 ms	3395,667ms	5807ms

Lause	**DB2 11.5**	**Oracle23c**	**MS SQL Server 2022**	**MariaDB 11.4**	**PostgreSQL16**	**MySQL 8.3**
S23	529,667 ms	188,667ms	729,667 ms	865ms	777,667ms	2575,667ms
S24	1,667ms	1837,333ms	210ms	61,667ms	237,667ms	145,667s
S25	1,667ms	1851,667ms	147ms	66ms	134ms	327,667ms
S26	22,333ms	2196,667ms	119,667ms	871,333ms	372ms	959,333ms

Lisa 4 - Kolmanda andmemahu töökiiruste mõõtmiste koondtulemused

Tabel 31. Kolmanda andmemahu töökiiruste mõõtmiste koondtulemused.

Lause	**DB2 11.5**	**Oracle23c**	**MS SQL Server 2022**	**MariaDB 11.4**	**PostgreSQL 16**	**MySQL 8.3**
S1	4,667ms	3950ms	388ms	1586,333 ms	141,333ms	1306,333ms
S2	4ms	3815ms	338,667ms	1681ms	123ms	1154,667ms
S4	2,667ms	4577ms	465,667ms	1771,333 ms	139,667ms	1164,333ms
S7	106ms	572,667ms	23,33ms	53,667ms	98,333ms	353,333ms
S8	85,667ms	3631,667ms	340ms	979,667ms	285ms	819,333ms
S9	2,333ms	3766ms	395ms	922,333ms	369,667ms	1004,667ms
S10	1,66ms	3745,333ms	382,333ms	1376,333 ms	147,667ms	1027,333ms
S11	1,66ms	3555,333ms	384,333ms	1380,667 ms	157ms	891ms
S12	27,667m	3897ms	202ms	1316,333 ms	334ms	1790ms
S15	2,333ms	3808ms	357,667ms	286,333ms	223,333ms	402ms
S15 ilma FK	14,667ms	3637,667ms	-	-	-	-
S16	2,333ms	3818,333ms	333,667ms	376ms	103,667ms	441,333ms
S19	8790,667 ms	31479ms	1454,333 ms	4117,667 ms	957,333ms	-
S20	1210,333 ms	6591,667ms	2019,333 ms	12814ms	9849,667ms	9788,667ms
S21	7592,667 ms	31248,333 ms	1531,667 ms	4446,667 ms	2071,667ms	11122,667 ms
S22	2204,667 ms	6573ms	393,667ms	10613ms	5215,333ms	8154,667ms

Lause	**DB2 11.5**	**Oracle23c**	**MS SQL Server 2022**	**MariaDB 11.4**	**PostgreSQL 16**	**MySQL 8.3**
S23	779,667 ms	592,667ms	286ms	4353,333 ms	706,333ms	6069,667ms
S24	1,667ms	3420,333ms	330,667ms	267ms	174,333ms	362,333ms
S25	1,333ms	3849,333ms	276ms	299,667ms	129ms	725ms
S26	81ms	3619,333ms	376,333ms	897ms	660,333ms	725ms

Lisa 5 – Vaadete *Tootaja_osakond* ja *Tootaja_osakond_klassikaline* loomise laused

```
CREATE OR REPLACE VIEW Tootaja_osakond AS
SELECT Tootaja.perenimi AS tootaja, Osakond.nimi AS osakond
FROM Tootaja INNER JOIN Osakond ON Tootaja.osakond_id=Osakond.osakond_id;


CREATE OR REPLACE VIEW Tootaja_osakond_klassikaline AS
SELECT Tootaja.perenimi, Osakond.nimi AS osakond
FROM Tootaja, Osakond
WHERE Tootaja.osakond_ID=Osakond.osakond_ID;
```

Joonis 17. Tootaja_osakond ja Tootaja_osakond_klassikaline loomise laused.

Lisa 6 – Tabelite elimineerimise teisenduse eksperimendis muudetud laused

```
S20

UPDATE Tootaja
SET eesnimi = 'UusNimi'
WHERE tootaja_id IN (
    SELECT tootaja_ID
    FROM (SELECT tootaja_ID FROM Tootaja) AS subquery
);

S21

DELETE t
FROM Tootaja AS t
LEFT JOIN Osakond AS o ON t.osakond_ID = o.osakond_ID
LEFT JOIN Asukoht AS a ON o.asukoht_ID = a.asukoht_ID
LEFT JOIN Valuuta AS v ON a.valuutakood = v.valuutakood
WHERE t.tootaja_ID = t.tootaja_ID;

S22

UPDATE Tootaja
LEFT JOIN Osakond o ON Tootaja.osakond_ID = o.osakond_ID
LEFT JOIN Asukoht a ON o.asukoht_ID = a.asukoht_ID
LEFT JOIN Valuuta v ON a.valuutakood = v.valuutakood
SET Tootaja.eesnimi = 'Uusnimi'
WHERE Tootaja.tootaja_ID = Tootaja.tootaja_ID;
```

Joonis 18. Muudetud laused S20, S21, S22 MySQLi jaoks.

```
S21

DELETE t
FROM Tootaja t
LEFT JOIN Osakond o ON t.osakond_ID = o.osakond_ID
LEFT JOIN Asukoht a ON o.asukoht_ID = a.asukoht_ID
LEFT JOIN Valuuta v ON a.valuutakood = v.valuutakood;
WHERE t.tootaja_ID = t.tootaja_ID;

S22

UPDATE Tootaja
LEFT JOIN Osakond o ON Tootaja.osakond_ID = o.osakond_ID
LEFT JOIN Asukoht a ON o.asukoht_ID = a.asukoht_ID
LEFT JOIN Valuuta v ON a.valuutakood = v.valuutakood
SET Tootaja.eesnimi = 'Uusnimi'
WHERE Tootaja.tootaja_ID = Tootaja.tootaja_ID;
```

Joonis 19. Muudetud laused S21, S22 MariaDB jaoks.

```
S21

DELETE Tootaja
FROM Tootaja t
LEFT JOIN Osakond o ON t.osakond_ID = o.osakond_ID
LEFT JOIN Asukoht a ON o.asukoht_ID = a.asukoht_ID
LEFT JOIN Valuuta v ON a.valuutakood = v.valuutakood
WHERE t.tootaja_ID = t.tootaja_ID;

S22

UPDATE Tootaja
SET eesnimi = 'Uusnimi'
FROM Tootaja t
LEFT JOIN Osakond o ON t.osakond_ID = o.osakond_ID
LEFT JOIN Asukoht a ON o.asukoht_ID = a.asukoht_ID
LEFT JOIN Valuuta v ON a.valuutakood = v.valuutakood
WHERE t.tootaja_ID = t.tootaja_ID;

S23

SELECT T.tootaja_ID, T.osakond_ID, T.eesnimi, T.palk, T.perenimi, T.alates
INTO Tootaja2
FROM Tootaja T
LEFT JOIN Osakond O ON T.osakond_ID = O.osakond_ID
LEFT JOIN Asukoht A ON O.asukoht_ID = A.asukoht_ID
LEFT JOIN Valuuta V ON A.valuutakood = V.valuutakood;
```

Joonis 20. Muudetud laused S21, S22, S23 MS SQL Serveri jaoks.

```
S21

DELETE FROM Tootaja
WHERE EXISTS (
    SELECT 1
    FROM Tootaja t
    LEFT JOIN Osakond o ON t.osakond_ID = o.osakond_ID
    LEFT JOIN Asukoht a ON o.asukoht_ID = a.asukoht_ID
    LEFT JOIN Valuuta v ON a.valuutakood = v.valuutakood
    WHERE Tootaja.tootaja_ID = t.tootaja_ID
);

S23

CREATE TABLE Tootaja2 LIKE Tootaja;

INSERT INTO Tootaja2 (tootaja_ID, osakond_ID, eesnimi, palk, perenimi,
alates)
SELECT T.tootaja_ID, T.osakond_ID, T.eesnimi, T.palk, T.perenimi, T.alates
FROM Tootaja T
LEFT JOIN Osakond O ON T.osakond_ID = O.osakond_ID
LEFT JOIN Asukoht A ON O.asukoht_ID = A.asukoht_ID
LEFT JOIN Valuuta V ON A.valuutakood = V.valuutakood;
```

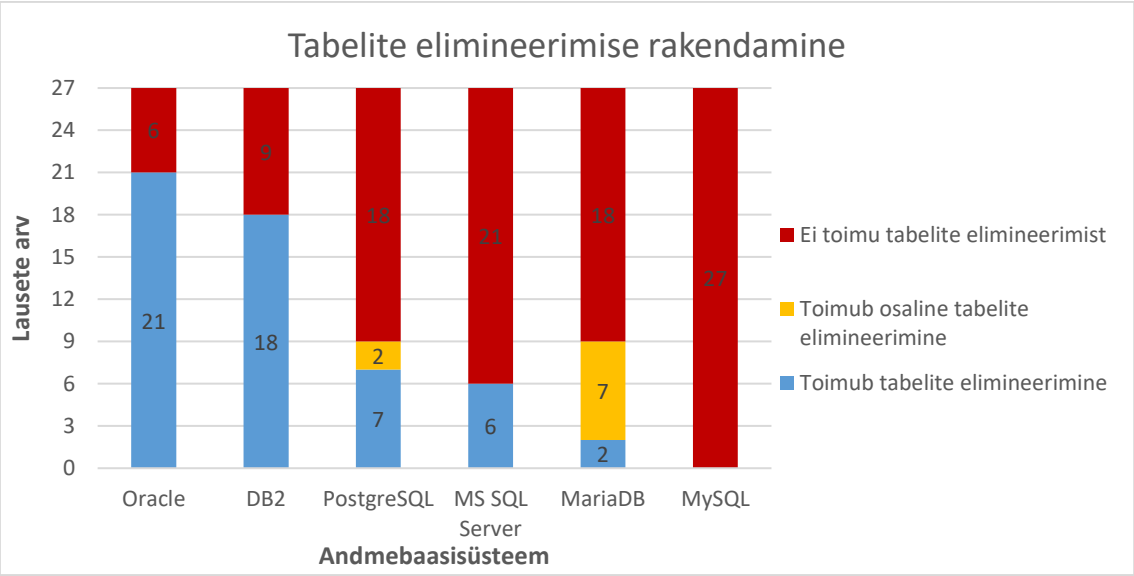
Joonis 21. Muudetud laused S21, S23 DB2 jaoks.

```
S22

UPDATE Tootaja t
SET eesnimi = 'Uusnimi'
WHERE EXISTS (
    SELECT 1
    FROM Tootaja t2
    LEFT JOIN Osakond o ON t2.osakond_ID = o.osakond_ID
    LEFT JOIN Asukoht a ON o.asukoht_ID = a.asukoht_ID
    LEFT JOIN Valuuta v ON a.valuutakood = v.valuutakood
    WHERE t.tootaja_ID = t2.tootaja_ID
);
```

Joonis 22. Muudetud lause S22 Oracle jaoks.

Lisa 7 – Tabelite elimineerimise rakendamine



Joonis 23. Tabelite elimineerimise rakendamine.