TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Emilia Niilo 213653IABB

Andmebaasis JSON dokumentide kasutamise mõju andmekäitluse operatsioonide jõudlusele PostgreSQL ja MongoDB andmebaasisüsteemide näitel

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: Emilia Niilo

20.05.2024

Annotatsioon

PostgreSQL ja MongoDB on 2024. aasta kevade seisuga kaks populaarset andmebaasisüsteemi. PostgreSQL on SQL-andmebaasisüsteem ning MongoDB kuulub NoSQL-andmebaasisüsteemide hulka. JSON dokumendid võimaldavad esitada andmeid hierarhiana. Bakalaureusetöö eesmärgiks on analüüsida andmete JSON dokumentidena esitamise mõju andmekäitlusoperatsioonide jõudlusele PostgreSQL (versiooni 16.2) ja MongoDB (versiooni 7.0.2) andmebaasisüsteemide näitel.

Töös luuakse erinevate disainidega andmebaase, kusjuures neljas neist on kasutusel JSON dokumendid. Andmebaasidesse genereeritakse testandmed kolme erineva andmehulga korral ja mõõtmistulemuste saamiseks lahendatakse kakskümmend andmete otsimise või muutmise ülesannet. Mõõtmistulemuste alusel leitakse, kas andmehulga ja operatsiooni täitmise aja vahel leidub kasvav lineaarne seos. Lisaks uuritakse indeksite mõju töökiirusele ja luuakse GIN indeksid PostgreSQL disainide jaoks, kus kasutatakse JSON dokumente.

Töö tulemusena on leitud mõõtmistulemused kolme erineva andmehulga ja viie disaini kohta. PostgreSQL ja MongoDB disainide korral tehtud eksperimentide tulemusi võrreldakse omavahel ja analüüsitakse saadud tulemusi. Tulemuste põhjal järeldab autor, et kõikide disainidega esineb enamus eksperimentide lahendamisel andmehulga ja täitmise aja vahel lineaarne kasvav seos. Erinevad andmebaasisüsteemid, JSON-dokumentide kasutamine ja erinevad disainid annavad erisuguseid töökiiruse tulemusi, sõltuvalt erinevatest andmekäitlusoperatsioonidest, koostatud eksperimentidest ja andmete esitamisest hierarhias. Samuti parandab indeksite kasutamine oluliselt PostgreSQL päringute töökiiruseid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 42 leheküljel, 7 peatükki, 21 joonist, 22 tabelit.

Abstract

The Influence of Using JSON Documents to the Performance of Data Manipulation Operations in the Example of PostgreSQL and MongoDB Database Management Systems

PostgreSQL and MongoDB are two popular database management systems (DBMSs) as of spring 2024. PostgreSQL is an SQL DBMS, while MongoDB is a NoSQL DBMS. JSON documents are a lightweight data-interchange format that allows data to be represented hierarchically. The goal of this thesis is to analyze the impact of using JSON documents to represent data in the database on the performance of data manipulation operations in PostgreSQL (version 16.2) and MongoDB (version 7.0.2).

Databases with different designs are created, utilizing JSON documents in four of them. Test data is generated for three different dataset sizes and twenty data manipulation tasks are solved to obtain measurement results. These results will be used to determine if there is a correlation between the increase in data volume and query execution time. Additionally, the impact of indexes will be investigated, with GIN indexes being created for PostgreSQL designs that use JSON documents.

The study results in measurements for three different data volumes and five designs. The results for PostgreSQL and MongoDB designs are compared and analyzed. Based on the results, the author concludes that for all the designs, there is a linear relationship between the increase in data volume and query execution time in most experiments. Different database management systems, the use of JSON documents, and various designs yield different performance results depending on the specific data handling operations, experiments conducted, and hierarchical data presentation. Furthermore, the use of indexes considerably improves the speed of PostgreSQL queries.

The thesis is in Estonian and contains 42 pages of text, 7 chapters, 21 figures, 22 tables.

Lühendite ja mõistete sõnastik

ACID	*Atomicity, consistency, isolation, durability.* Atomaarsus, terviklikkus, isoleeritus, püsivus – andmebaasitehingute läbiviimiselt oodatavad omadused.
CSV	*Comma-separated values*. Tekstifaili tüüp, mis sisaldab vorminguta andmeid ning kus andmebaasikirjed on üksteisest eraldatud komadega.
GIN indeks	*Generalized Inverted Index.* PostgreSQLi indeksitüüp, mida kasutatakse massiividel ja muudes keerulistes andmestruktuurides olevate väärtuste otsingute jõudluse parandamiseks.
ISO	*International Organization for Standardization.* Rahvusvaheline organisatsioon, mis ühendab standardiasutusi.
ISO 8601	*The standard, that specifies accepted way to represent dates and time*. Kuupäeva ja kellaaja vormingu standard.
JSON	*JavaScript Object Notation.* Lihtne andmevahetusvorming, mis põhineb JavaScripti alamhulgal.
JSONB	*Binary JSON.* Kahendkoodis talletatud JSON objekt, toetab JSON-andmete indekseerimist.
NoSQL	*Not only SQL*. Andmebaasisüsteemide klass, kuhu kuuluvad andmebaasisüsteemid ei kasuta SQL keelt ja selle aluseks olevat andmemudelit. Kuigi selliseid andmebaasisüsteeme loodi ka varem, siis tüüpiliselt loetakse siia alates 2010st aastast turule tulnud andmebaasisüsteeme.
SQL	*Structured Query Language*. Struktureeritud andmebaasikeel, mida saab kasutada SQL-andmebaasisüsteemides andmete lugemiseks, muutmiseks, loomiseks, kustutamiseks ja muudeks toiminguteks.
UTF-8	*Unicode Transformation Format, 8-bit.* Unicode'i 8-bitine teisendusvorming süsteem Unicode'i märkide kodeerimiseks.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Selles peatükis antakse lugejale esmane arusaam käesoleva töö eesmärkidest ja ülesehitusest.

1.1 Üldine taust

Andmekäitluse operatsioonide jõudlus on oluline tegur igasuguste andmeid kasutavate süsteemide efektiivses toimimises. Selleks, et paremini mõista, kuidas andmebaasisüsteemide pakutavad erinevad lähenemised andmete esitamiseks mõjutavad andmekäitluse jõudlust, on plaanis uurida kahe populaarse andmebaasisüsteemi, PostgreSQLi [1] ja MongoDB [2], erinevate salvestamisstrateegiate mõju. 2024. aasta veebruari seisuga on need süsteemid andmebaasisüsteemide populaarsuse indeksis [3] vastavalt neljandal ja viiendal kohal.

JSON dokumendid võimaldavad esitada andmeid hierarhiana ning teha seda viisil, et need on loetavad ja arusaadavad nii inimkasutajale kui ka arvutiprogrammile. MongoDB on dokumendipõhine NoSQL andmebaasisüsteem. MongoDB andmebaas on hulk JSON dokumentide kollektsioone. PostgreSQL on SQL-andmebaasisüsteem. PostgreSQLi tabelite väljades saab soovi korral salvestada JSON dokumente, sest PostgreSQL toetab JSON ja JSONB andmetüüpe ja võimaldab tabelites luua sellist tüüpi veerge. Mõlemas andmebaasisüsteemis saab andmebaasis jõustada kirjutamise skeemi, mis tähendab, et andmeid kontrollitakse reeglitele vastavuse osas kohe salvestamisel ning kui andmed reeglitele ei vasta, siis salvestamine ei õnnestu. See võib veidi aeglustada andmete salvestamist. Samas, kui andmebaasisüsteem saab olla kindel andmete teatud reeglitele vastamises, siis saaks andmebaasisüsteem mõnda päringut kiiremini täita.

1.2 Lahendatav probleem

Bakalaureusetöö eesmärgiks on analüüsida ja võrrelda PostgreSQL ja MongoDB andmebaasisüsteemide abil loodud andmebaasides andmekäitluse operatsioonide

jõudlust JSON dokumentide kasutamise kontekstis. Tuleb selgitada välja, kuidas erinevad disainivalikud, JSON dokumentide ja indeksite kasutamine, mõjutavad andmekäitluse (andmete lugemise, sisestamise, muutmise ja kustutamise) operatsioonide jõudlust e töökiirust.

Oodatavaks tulemuseks on eksperimentidel põhinevad hinnangud erinevate disainide mõjule andmekäitluse operatsioonide jõudlusele ning saadud hinnangute võrdlus varasemate uuringute tulemustega.

1.3 Uurimisküsimused

Autor soovib töö käigus leida vastused järgmistele uurimisküsimustele.

1. Millised on erinevate andmebaasi disainide korral erinevat tüüpi andmebaasioperatsioonide töökiirused?
2. Kas JSON dokumentide kasutamise või mittekasutamise korral saab tuua välja andmebaasisüsteemi (PostgreSQL või MongoDB), mille puhul on need kiirused üldiselt kõige paremad/halvemad?
3. Kas andmemahtude ja operatsioonide täitmise aja vahel on andmebaasides lineaarne kasvav sõltuvus või mitte?
4. Kas indeksid aitaksid PostgreSQLis üksikute olemite andmete otsimisel töökiirust parandada?
5. Kas saadud tulemused on kooskõlas varasemate samade andmebaasisüsteemide jõudluse uuringutega?

1.4 Töö edasine struktuur

Töös tuuakse välja metoodika, teoreetiline taust, eksperimentide kirjeldused, tulemused ja analüüs ning kokkuvõte. Metoodika peatükk teeb ülevaate tööprotsessist ning kasutatud tarkvarast ja riistvarast. Teoreetiline taust annab põgusalt ülevaate dokumendipõhistest andmebaasisüsteemidest, JSON dokumentidest SQL andmebaasides ning eelnevatest sarnastest uuringutest.

Eksperimentide kirjelduse all toob autor täpsemalt välja, mille põhjal on loodud PostgreSQL ja MongoDB andmebaaside struktuur, kuidas loodi andmebaasidesse testandmed ning millised kakskümmend ülesannet on koostatud eksperimentide jaoks. Tulemuste peatükis on välja toodud eksperimentide tulemused. Seejärel analüüsitakse saadud tulemusi ja võrreldakse neid eelnevate uuringute tulemustega. Lõpuks esitatakse kokkuvõte.

2 Metoodika

Käesolev peatükk annab ülevaate töö tegemise protsessist ning töös kasutatud tark- ja riistvarast.

2.1 Tööprotsessi kirjeldus

Reiko Roopärgi 2022. aasta bakalaureusetöös [4] uuriti kirjutamise skeemi ja JSON formaadis andmete kasutamise mõju PostgreSQL ja MongoDB andmebaasirakenduste loomisele. Selles töös ei käsitletud andmekäitluse operatsioonide töökiiruse küsimusi. Käesolev töö viib seda uuringut edasi. Selleks luuakse töö käigus samasuguse struktuuriga andmebaasid nagu Roopärgi töös. Eksperimentide abil kogutakse andmeid erinevate disainide alusel loodud PostgreSQL ja MongoDB andmebaasides toimuvate andmekäitluse operatsioonide jõudluse kohta. Saadud andmeid analüüsitakse, võrreldakse ja tõlgendatakse, et hinnata erinevate disainide mõju andmekäitluse operatsioonide jõudlusele. Kokku on vaatluse all viis andmebaasi disaini. Iga disaini juures tuuakse välja lühike identifikaator, mida kasutatakse edaspidi sellele disainile viitamiseks.

- T – Traditsiooniline" PostgreSQL andmebaas, kus JSONB tüüpi veerge ei kasutata.
- H – PostgreSQL andmebaas, kus andmed on JSONB tüüpi veergudes ja seal olevad dokumendid on hierarhilised (*nested documents*). Dokumente kontrollitakse salvestamisel reeglite suhtes.
- MH – PostgreSQL andmebaas, kus andmed on JSONB tüüpi veergudes ja seal olevad dokumendid ei ole hierarhilised. Dokumente kontrollitakse salvestamisel reeglite suhtes.
- HM – MongoDB andmebaas, kus dokumendid on hierarhilised (*embedded documents*). Dokumente kontrollitakse skeemi suhtes.

- MM – MongoDB andmebaas, kus dokumendid on seotud viidete kaudu. Dokumente kontrollitakse skeemi suhtes.

Eksperimentide käigus luuakse erineva disainiga andmebaase, genereeritakse sinna testandmeid ja lahendatakse nende andmebaaside põhjal erinevaid andmete otsimise, lisamise, muutmise ja kustutamise ülesandeid. Seejärel hinnatakse saadud tulemusi ning võrreldakse neid varasemate uuringute jooksul saadud tulemustega, et tuua välja sarnasused ja erinevused.

2.2 Eksperimenteerimisel kasutatud riistvara

Autor kasutas eksperimentide läbiviimiseks MacBook Pro 13-tollist sülearvutit 2019. aastast. Sülearvuti protsessoriks oli 1,4 GHz neljatuumaline Intel Core i5 ja see oli varustatud 8 GB (2133 MHz) LPDDR3 sisemäluga. Operatsioonisüsteemina kasutati macOS-i versiooni 14.1.

2.3 Eksperimenteerimisel kasutatud tarkvara

Eksperimentide läbiviimiseks kasutati PostgreSQL-i versiooni 16.2 ja MongoDB versiooni 7.0.2. Arvutisse olid need andmebaasisüsteemid installeeritud otse, ilma Dockeri konteinerite virtualiseerimisplatvormi kasutamata, ning arvuti, millel eksperimente tehti, oli 64-bitine, x86_64-apple-darwin20.6.0 arhitektuuriga (*x86_64-apple-darwin20.6.0, compiled by Apple clang version 12.0.5 (clang-1205.0.22.9), 64-bit*.).

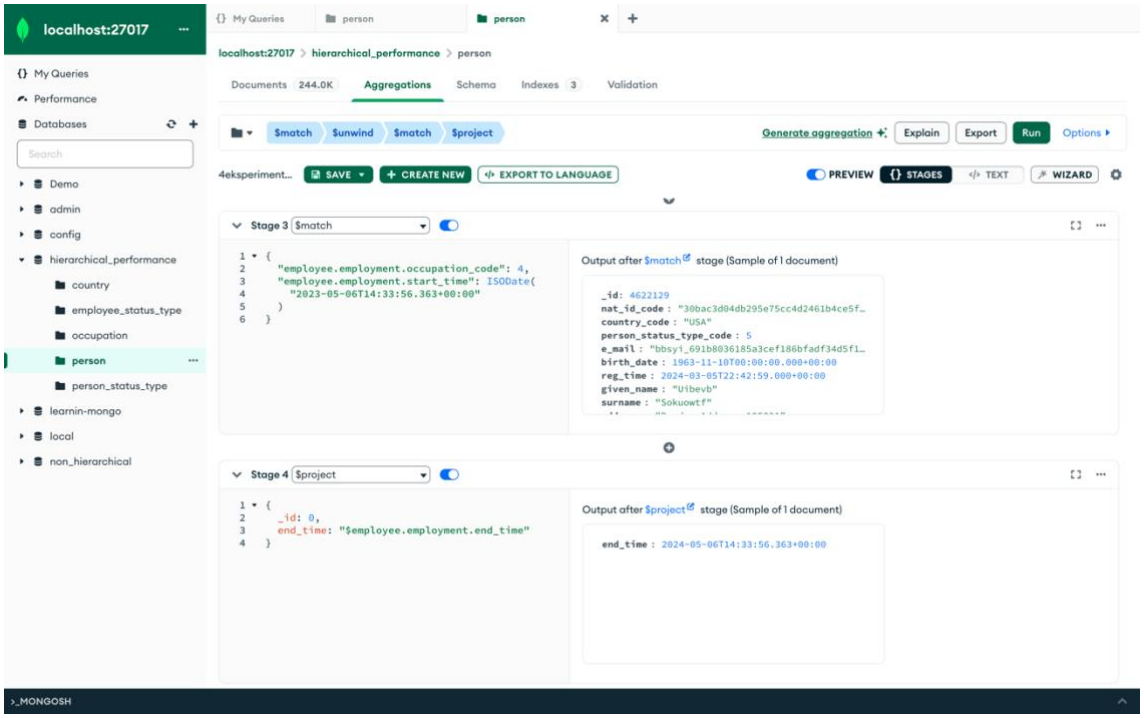
PostgreSQL andmebaasi haldamiseks kasutati pgAdmin 4 tarkvara [5], mis võimaldas lihtsat ja tõhusat eksperimentide läbiviimist.

MongoDB andmebaasi haldamiseks kasutati MongoDB Compass tarkvara [6], mis võimaldas andmebaasi visualiseerimist graafilisel viisil. Lisaks kasutati MongoDB Shell [7] käsurea tööriista, mis võimaldas interaktiivset suhtlust MongoDB andmebaasisüsteemiga otse käsurealt.

Mõningate käskluste jaoks, näiteks MongoDB käivitamiseks ja lõpetamiseks, kasutati macOS-i sisseehitatud käsurea rakendust Mac Terminal. Tänu sellele oli autoril võimalik teha mitmesuguseid toiminguid otse käsurealt, ilma graafilist kasutajaliidest kasutamata.

Aja kokkuhoidmise eesmärgil on abivahendina teoreetilise tausta grammatika ja testandmete loomise ning eksperimentide ülesannete sõnastuse kontrollimiseks, parandamiseks ning erinevate disainide puhul ka ülesannete lahenduste koostamiseks kasutatud tehisintellekte ChatGPT 3.5 [8] ja Microsoft Copilot [9]. Näiteks pärast pikemate ülesannete lahenduste koostamist traditsioonilise skeemi jaoks, nagu ülesanne 5_1, sisestati kood tehisintellekti vahendisse, et see genereeriks hierarhilise ja mitte-hierarhilise skeemi jaoks sama ülesande PostgreSQL ja MongoDB lahenduse koodi. Seejärel tegi autor vajadusel tehisintellekti genereeritud koodis parandusi, et koodi oleks võimalik ülesannete lahendamisel kasutada. Kokku genereeriti sellisel viisil erinevate disainide peale 25 lauset. Genereeritud koodis pidi peamiselt muutma õigeks andmebaaside põhjal lausetes kasutatud tabelite ja nende veergude nimed ning kui otsiti kindlat olemit, siis vastavalt testandmetele andma ette õiged andmed – näiteks õige meiliaadress testandmete hulgast, mille järgi olemit otsida.

MongoDB korral kontrolliti lisaks MongoDB Compass *Aggregations* alt, kas lause igas faasis leitakse kriteeriumitele vastavad andmed (vt Joonis 1).



Joonis 1. MongoDB HM skeemi (ülesanne 2_2) päringu faaside vahetulemused.

Pearsoni korrelatsioonikordaja väärtuse ning geomeetrilise keskmise arvutamiseks kasutatakse *Google Sheetsi* ja seal vastavalt olemasolevaid *CORREL()* [10] ja *GEOMEAN()* funktsioone [11].

3 Teoreetiline taust

Selle peatüki jaotised annavad ülevaate teoreetilisest taustast, võttes lühidalt kokku dokumendipõhiste andmebaasisüsteemide olemuse ja funktsionaalsuse. See aitab paremini aru saada töö teoreetilisest taustast, võimaldades saada paremini aru dokumendipõhiste andmebaasisüsteemide olulisusest ja toimimisest.

3.1 Dokumendipõhised andmebaasisüsteemid

Pärast 2010-ndat aastat on dokumendipõhised andmebaasisüsteemid muutunud üha populaarsemaks, pakkudes suuremat paindlikkust ja erinevate uuringute kohaselt ka paremat jõudlust võrreldes traditsiooniliste SQL-andmebaasisüsteemidega. Need süsteemid võimaldavad andmete salvestamist hierarhiliste dokumentidena, mille struktuur võib olla igal dokumendil erinev. Dokumendis talletatakse iga olemi andmeid võti-väärtus paaridena ning dokumente salvestatakse enamasti JSON formaadis tekstina või kahendformaadis BSON objektina [4, p. 18].

Selline lähenemine võimaldab arendajatel realiseerida andmete registreerimist kiiremini ja paindlikumalt, kuna andmete puhul pole vaja järgida kindlat skeemi, nagu SQL-andmebaasides, mis võimaldab arendajatel paremini kohaneda muutuvate nõudmistega. Dokumendipõhised andmebaasisüsteemid pakuvad arendajatele suuremat vabadust andmete struktuuri kujundamisel. Paljude dokumendipõhiste andmebaasisüsteemide nõrkus on, et need ei toeta mitut dokumenti hõlmavaid ACID omadustega tehinguid [12]. Kuigi dokumendipõhised andmebaasid pakuvad paindlikkust andmete struktuuri osas, võivad need piirata keerukate päringute tegemist võrreldes traditsiooniliste SQL-andmebaasidega. Näiteks võib andmete filtreerimine ja ühendamine olla keerulisem, kui andmeid salvestatakse mitmes erinevas dokumendis.

Lisaks pakuvad dokumendipõhised andmebaasisüsteemid paremat horisontaalset skaalatavust. See tähendab, et süsteemi saab hõlpsasti laiendada, lisades rohkem servereid, mis jagavad andmeid ja koormust. See võimaldab süsteemil paremini toime tulla suure hulga kasutajate ja suure andmekogumiga, säilitades samal ajal jõudluse ja vastupidavuse. Kuna dokumendid sisaldavad tavaliselt kogu vajalikku infot omavahel

seotud erinevat tüüpi olemite kohta, võib nende lugemine olla efektiivsem kui relatsiooniliste andmete hankimine mitmest tabelist. See võib olla eriti kasulik juhul, kui andmeid erinevat tüüpi olemite kohta loetakse sageli korraga, kuna dokumendi lugemine võib nõuda vähem erinevate andmete ühendamist ja lihtsamat loogikat kui relatsiooniliste andmete päringud [13]. Probleem on aga selles, et dokumendi struktuur, mis on sobiv mingi ühe ülesande kiireks täitmiseks võib olla jälle vähem sobiv mõne teise ülesande jaoks, mistõttu tuleb siis hakata filtreerima ja ühendama erinevates dokumentides olevaid andmeid koos sellega kaasnevate probleemidega.

3.2 JSON dokumendid SQL-andmebaasides

JSON ehk *JavaScript Object Notation* on minimaalne, tekstipõhine andmevahetusformaat, mis põhineb JavaScripti alamhulgal. See ei ole dokumendivorming ega märgistuskeel, vaid pigem programmeerimiskeele mudeli andmevahetuse formaat. JSON on kergekaaluline ja lihtsasti analüüsitav, ning sellel on laialdane tugi erinevates programmeerimiskeeltes ja süsteemides ning iga salvestatud väärtus peab olema JSONi reeglite kohaselt kehtiv. JSONis kasutatavad väärtused võivad olla tekstitüüpi, arvutüüpi ja tõeväärtustüüpi. Need võivad ka olla objektid, massiivid ning väärtus võib ka puududa. [14]

2016. aastal väljastas ISO ehk *International Organization for Standardization* SQL-standardi uue versiooni, mis asendas eelnevat 2011. aasta versiooni. See muudatus tõi sisse 44 uut valikulist erisust, mille hulgas oli ka JSON andmetele tugi [15]. Puudub küll standardiseeritud JSONi päringukeel, kuid SQL-standard määratleb siiski võimalused JSON-andmetes navigeerimiseks ning pakub JSON-dokumendis päringute tegemiseks süsteemi-defineeritud funktsioone [16].

JSON-andmetüüpide kasutamine PostgreSQL-is võimaldab salvestada andmeid JSON formaadis. PostgreSQL-is on võimalik andmeid salvestada nii JSON kui ka JSONB tüüpi veergudesse. Praktiline erinevus seisneb tõhususes – andmetüübi JSON korral salvestatakse sisendteksti täpse koopia, samas kui JSONB korral salvestatakse andmed lahutatud kandformaadis, mis võib muuta andmete sisestamise pisut aeglasemaks. Lisaks toetab JSONB tüüp ka indekseerimist, mis võib oluliselt suurendada päringute jõudlust.

JSON-tüüpidele kehtivad teatavad kodeerimisreeglid, näiteks nõutakse, et kodeering oleks UTF8. [17]

PostgreSQL hakkas JSON-i kasutamise võimalusi pakkuma alates versioonist 9.2 (töö kirjutamise ajal on viimane versioon PostgreSQL 16), kuid peamised täiustused, sealhulgas uued funktsioonid ja operaatorid, lisati versiooniga 9.4 [18, Sec. 4]. JSONB tüüpi veerus salvestatakse andmeid aeglasemalt võrreldes salvestamisega JSON tüüpi veerus, kuna nende teisendamisega kaasnevad kulud, kuid neid töödeldakse oluliselt kiiremini. Lisaks toetab JSONB-andmetüüp rohkem operaatoreid kui JSON-tüüp. PostgreSQL pakub tabeli ridade JSON-dokumentidena esitamiseks näiteks funktsiooni *row_to_json()* [18] ja GIN indeksit indekseerimiseks. GIN indeksid kuuluvad funktsioonipõhiste indeksite klassi, mis võimaldavad tõhusalt otsida JSONB-tüüpi veergudes olevatest dokumentidest võtmeid või võtme/väärtuse paare. [19]

3.3 Kirjutamise skeem *vs.* lugemise skeem

Andmebaasitehnoloogias on kaks põhilist lähenemist andmebaasi skeemi struktuuri loomisel ja kasutamisel: kirjutamise skeem (*schema-on-write*) ja lugemise skeem (*schema-on-read*). Kirjutamise skeemi puhul tuleb enne andmete sisestamist andmebaasi määratleda andmete struktuur ehk skeem ning sisestatavad andmed peavad sellele skeemile vastama. Lugemise skeemi korral pole see vajalik ehk lugemise skeem võimaldab salvestada igal kujul andmeid, samuti kui andmed oleks erineva struktuuriga. [4, pp. 22-23]

Kuna kirjutamise skeemi kasutamisel toimub andmete kontroll andmete andmebaasi kirjutamisel, siis kõik sisestatavad andmed peavad vastama etteantud struktuurile ja piirangutele. Näiteks SQL-andmebaasis tuleb ette määrata baastabelid ja nende veerud ning kõik andmed peavad sobima nende tabelite ja veergude kirjeldatud struktuuriga. MongoDB võimaldab samuti defineerida kirjutamise skeemi dokumentidele, näiteks saab skeemis kontrollida välja olemasolu, väärtuse kuulumist andmetüüpi või väärtuse vastamist mingitele täiendavatele reeglitele [20].

Lugemise skeemi korral andmebaasi tasemel struktuuripiiranguid ei jõustata, kuid kasutaja peab struktuuri teadma, et oleks võimalik loetud andmeid kasutada, otsida ning

uusi sisestatavaid andmeid kontrollida. Usutakse, et lugemise skeemiga andmetöötlus on paindlikum suurte andmete, struktureerimata andmete või sagedaste skeemimuutuste korral. [21] Dokumendipõhised NoSQL süsteemid tulid turule lugemise skeemi lähenemisega, kuid aina rohkematesse süsteemidesse (sh MongoDB) lisandub ka kirjutamise skeemi defineerimise võimalus.

3.4 Olemasolevad PostgreSQL ja MongoDB operatsioonide töökiiruse uuringud

Sarnaseid uuringuid on tehtud juba varasemalt ning neid tulemusi võrreldakse selles töös saadud tulemustega. Uute uuringute tegemine on õigustatud dokumendipõhise andmete salvestamise jätkuva populaarsusega ning andmebaasisüsteemide uute versioonide turule tulekuga.

Näiteks 2024. aasta märtsis ilmunud artiklis võrreldi JSON-formaadis esitatud andmetega tehtavate operatsioonide jõudlust MongoDB ja PostgreSQL korral. Uuringus leiti, et väiksema andmemahuga stsenaariumites näitas PostgreSQL kiiduväärset tõhusust, kuid kui andmekogum suurenes, siis täheldati PostgreSQL jõudluse märgatavat langust. Teisalt, MongoDB näitas järjepidevat head jõudlust, sõltumata andmekogumi suurusest. Olenemata sellest, kas tegemist oli mõne olemi andmetega või miljoni olemi andmetega, oli MongoDB võime säilitada stabiilne jõudlus märkimisväärne. [22]

Samal teemal on ka 2015. aastal Tallinna Tehnikaülikoolis tehtud magistritöö, mille järeldustes kirjutati, et JSON-andmetüüpide kasutuselevõtt PostgreSQL-is ei tähenda, et see võib asendada NoSQL-süsteeme [23].

Samal aastal ilmunud artiklis „Performance investigation of selected SQL and NoSQL databases“ [24] on märgitud, et geofunktsioonidega päringute puhul toimivad NoSQL-andmebaasisüsteemid väga stabiilselt. Mõõdetud vastamisajad varieeruvad suureneva andmehulga puhul ainult umbes mõne sekundiga. Samas väikeste, keerulise geomeetriaga andmekogumite puhul toimis SQL-andmebaas paremini.

Lisaks leiti Geoinformaticas 2020. aastal avaldatud uuringus, et viiesõlmelises klastris reageerimisaja alusel leitud tulemused näitavad, et PostgreSQL on peaaegu kõigil juhtudel parem kui MongoDB. Teisalt väheneb keskmine reageerimisaeg MongoDB

puhul tohutult seoses indeksite kasutamisega, samas kui PostgreSQLi puhul on selle positiivne mõju oluliselt väiksem. [25]

4 Eksperimentide kirjeldus

Selles peatükis selgitatakse eksperimentide sisu ning esitatakse vajalikud testandmed ja ülesanded, mis on vajalikud eksperimentide läbiviimiseks.

4.1 Andmebaasi struktuur

Vastavalt Roopärgi [4, p. 16] tööle luuakse viis erineva struktuuriga andmebaasi – kolm PostgreSQLi ning kaks MongoDB jaoks. PostgreSQLi puhul realiseeritakse need andmebaasid kolme eraldi skeemina ühes PostgreSQL andmebaasis. MongoDB korral luuakse kaks eraldi andmebaasi. Andmebaasi skeemi loomiseks mõeldud kood on saadud Roopärgi lõputöö GitHubi pesast [26] vastavale disainile pühendatud kataloogist. Järgnevalt nimetatakse need disainid. Iga disaini juures viidatakse lisale, kus on selle disaini visuaalne kirjeldus. Lisades on toodud disaine esitavad skeemid Roopärgi tööst kuvatõmmistena, kuna praeguse töö autor kasutas samasuguse ülesehitusega andmebaase.

T – PostgreSQL traditisoonilise skeemi disain (vt Lisa 2).

H - PostgreSQL hierarhilise skeemi disain (vt Lisa 3).

MH - PostgreSQL mitte-hierarhilise skeemi disain (vt Lisa 4).

HM – hierarhilise MongoDB andmebaasi disain (vt Lisa 5).

MM – mitte-hierarhilise MongoDB andmebaasi disain (vt Lisa 6).

MongoDB andmebaasisüsteemis mitte-hierarhilise andmebaasi disaini juures muudeti isikute, töötajate ja tööhõive kollektsioonides valideerimisel skeemis *person_id* ja *mentor_id* tüüpi, asendades tüübi *objectId* tüübiga *integer*. Sama tehti ka hierarhilise andmebaasi isikute kollektsiooni puhul ning samuti lubati seal märkida töötajale ja aadressile *null* e väärtuse puudumine. Ilma nende muudatusteta polnud võimalik testandme faile MongoDB vastavatesse kollektsioonidesse üles laadida, kuna failides olnud andmed ei olnud valideerimisreeglitega kooskõlas.

4.2 Testandmete hulk ja genereerimine

Testandmete genereerimiseks kasutati PostgreSQL lauseid, et luua esialgsed andmed traditsioonilise skeemi (T) jaoks. Nendes lausetes kasutati *generate_series*() ja *random*() funktsioone. Hiljem kopeeriti need andmed ka hierarhilise (H) ja mitte-hierarhilise (MH) JSON kasutusega skeemidesse. Andmete genereerimiseks koostatud SQL-laused on lisatud failidena *GitHub*-i lehel *DBMS-perforamnce-testing* [27]. Selline lähenemine tagas, et kõikide disainide ja andmebaasisüsteemide korral kasutati testimiseks ühesuguseid andmeid.

Katsetusi tehti kolme andmehulgaga, et saaks arvutada Pearsoni korrelatsioonikordaja väärtust [28]. Need andmehulgad (T1, T2, T3) on esitatud tabelis 1, kus iga PostgreSQLi traditsioonilise disaini tabeli kohta näidatakse selles olev ridade arv. Ridade arvud, mis erinevate andmehulkade korral on erinevad, on tähistatud rasvase fondiga.

Tabel 1. Andmemahud (T1, T2, T3) eksperimentide läbiviimiseks.

Tabel	**T1**	**T2**	**T3**
Country	20	20	20
Employee	**80 000**	**160 000**	**320 000**
Employee_status_type	10	10	10
Employment	**400 000**	**800 000**	**1 600 000**
Occupation	50	50	50
Person	**100 000**	**200 000**	**400 000**
Person_status_type	8	8	8

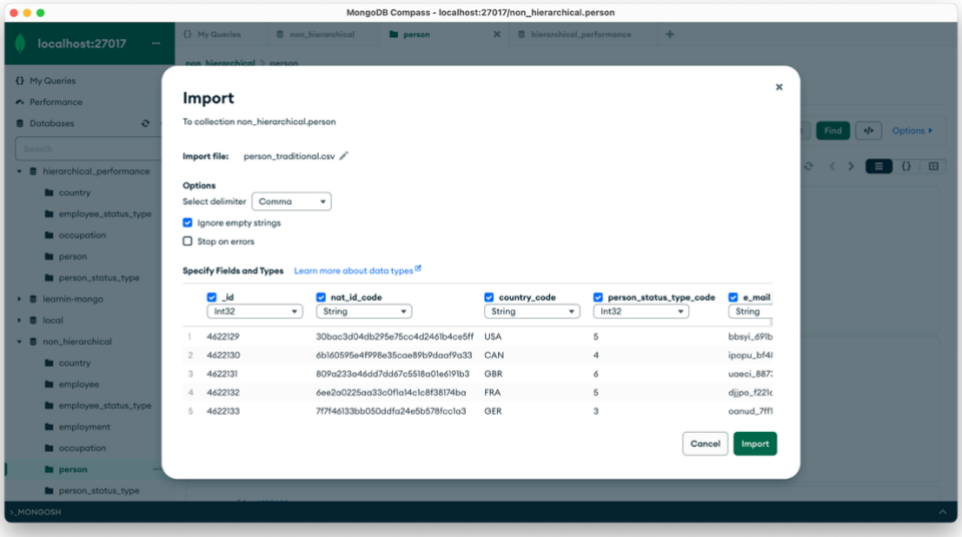
Esialgsete testandmete tõstmiseks PostgreSQL traditsioonilisest skeemist teistesse PostgreSQL andmebaasi skeemidesse, kasutati peamiselt INSERT INTO .. SELECT lauseid. Joonisel 2 esitatakse näide ühest sellisest lausest.

```
INSERT INTO non_hierarchical.occupation (occupation_code, data)
SELECT occupation_code, jsonb_build_object( 'name', name, 'description',
description )
FROM traditional.occupation;
```

Joonis 2. Testandmete kandmine PostgreSQLis skeemist T skeemi MH.

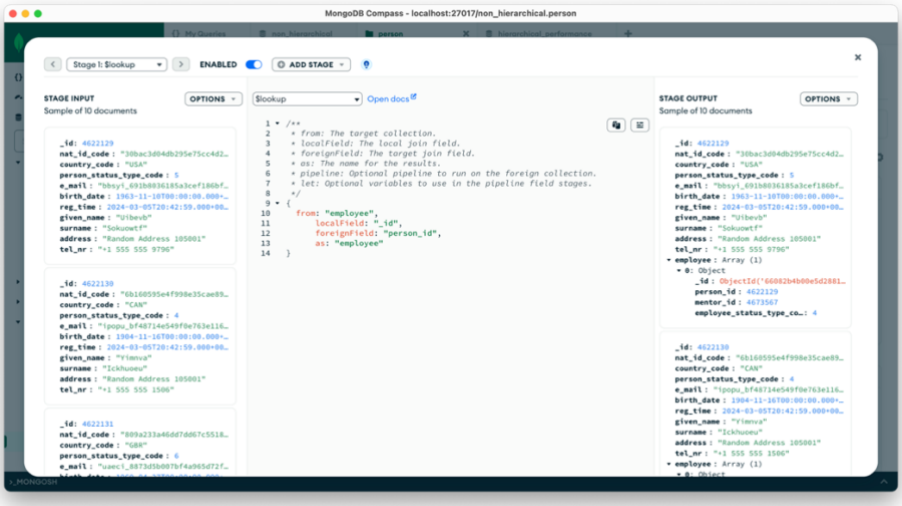
Testandmete lisamiseks MongoDB andmebaasidesse eksporditi esialgu andmed PostgreSQL pgAdmin4 tarkvara abil CSV-failidesse. Selleks, et testandmeid oleks võimalik salvestada õiges formaadis MongoDB hierarhilisse isikute kollektsiooni jaoks, kasutati lisaks SQL-lauseid, mille leiab *GitHubi* pesa failist nimega *MongoDB_hierarchical_person_data.sql*. Loodud lausega määrati kindlaks, et andmed oleksid õiges formaadis (näiteks JSON kasutab kuupäevade kodeerimisel ISO 8601 stringiformaati [29]) ja valideerimisreeglitele vastavalt õigetes veergudes (töötaja ja töötamistega seotud andmed ühes veerus).

Testandmete MongoDB andmebaasi saamiseks kasutati MongoDB Compass tarkvara. Täpsemalt kasutati selleks failide importimiseks sisseehitatud *Add data import JSON or CSV file* võimalust. Enne importimist tuli üle vaadata, kas kõigi objektide tüübid olid õigesti märgitud, näiteks hierarhilise (HM) ja mitte-hierarhiliste (MM) isiku kollektsioonide puhul tuli isiku ID märkida *integer* tüüpi (vt Joonis 3) ja kontrollida, et kuupäevad oleksid *date* tüüpi.



Joonis 3. MongoDB kollektsiooni CSV-faili import MM disaini korral.

Lisaks tekkis küsimus, kui disainide korral on isikute tabelis ID tüübiks määratud *integer*, kas siis kollektsioonide vahelised seosed jäävad siiski alles. Vaadates MongoDB Compass vahendis isiku kollektsioonis olevaid andmeid *Aggregations* alt (vt Joonis 4), siis on näha pildil paremal, et isikutega on seotud ka töötaja väli.



Joonis 4. Isikute seosed töötajatega MongoDB andmebaasis MM disaini korral.

4.2.1 GIN indeksite loomine

Selleks et uurida indeksite mõju üksikute olemite otsimise töökiirusele PostgreSQL andmebaasisüsteemis, loodi PostgreSQL hierarhilise ning mitte-hierarhilise disainiga andmebaaside jaoks GIN indeksid.

PostgreSQL dokumentatsiooni järgi [30] koostati autori arvamusel vajalikud indeksid ning selleks, et päringud neid ka kasutaksid, muudeti olemasolevaid lauseid ülesannete lahendamiseks. Seda seetõttu, et *jsonb_path_ops*, mida kasutati indeksi koostamisel, toetab ainult päringuid, milles kasutatakse operaatoreid @>, @? ja @@ .

GIN indeksitega ning muudetud ülesannete lausetega on võimalik tutvuda lisas 10.

4.3 Andmebaasides lahendatavad ülesanded

Eksperimentide jaoks on välja valitud andmete otsimise, lisamise, uuendamise ja kustutamise operatsioonid, mida täidetakse PostgreSQL ühe andmebaasi kolme erineva skeemi ning MongoDB kahe erineva andmebaasi peal.

Ülesanded on koostatud nii, et peaaegu kõigi ülesande tüüpide korral on üks ülesanne selline, kus otsitakse andmeid, mis hierarhilise esituse korral on kõige ülemisel tasemel ja teine ülesanne selline, kus hierarhilise esituse korral ei ole otsitavad andmed kõige ülemisel tasemel.

4.3.1 Funktsiooni kasutus ja sorteerimine

Valiku põhjendus: See eksperiment võimaldab uurida, kuidas otsingutingimuses väärtuse funktsiooni abil teisendamine mõjutab päringute täitmise kiirust. Samuti annab see ülevaate, kuidas sorteerimine mõjutab päringute täitmise kiirust.

Ülesanne S1_1: Leia isikud, kes on sündinud aastatel 1940 kuni 1980 (otspunktid kaasa arvatud). Väljasta isiku unikaalne ID, meiliaadress ja sünnikuupäev. Sorteerida tulemus sünniaasta ja selle sees meiliaadressi järgi.

Ülesanne S1_2: Leia töötamised, mis on alanud aastatel 2020 kuni 2022 (otspunktid kaasa arvatud). Väljasta töötajaks oleva isiku unikaalne ID, ametikoha nimetus ja

töötamise alguse ning lõpu aasta. Sorteerida tulemus alguse aasta ja selle sees töötaja ID järgi.

4.3.2 Üksiku olemi otsimine

Valiku põhjendus: Eksperimendi eesmärgiks on uurida üksikute olemite nende unikaalse identifikaatori järgi otsimise jõudlust erinevate disainide puhul.

Ülesanne S2_1: Otsi üksikut isikut tema meiliaadressi järgi. Väljasta isiku ID ja perenimi.

Ülesanne S2_2: Otsida konkreetse isiku (teada on meiliaadress) konkreetse alguse ajaga konkreetsel ametikohal (teada on ametikoha kood) töötamise lõpuaega.

4.3.3 Koondandmete leidmine

Valiku põhjendus: Selle ülesandega on võimalik vastata koondandmete päringutega seotud uurimisküsimustele.

Ülesanne S3_1: Leia iga riigi kohta sellega seotud isikute arv ning nende isikute keskmine vanus päevades registreerimise hetkel. Väljasta riigi kood, riigi nimetus, isikute arv ning keskmine vanus päevades. Väljasta ka riigid, kus ei ole ühtegi seotud isikut – nende korral on arv null ja keskmine teadmata (NULL).

Ülesanne S3_2: Leia iga ametikoha kohta sellega seotud töötamiste arv ja keskmine töötamise pikkus päevades. Arvesta vaid töötamistega, millel on teada nii algus kui lõpp. Väljasta ametikoha kood, ametikoha nimetus, töötamiste arv ja keskmine töötamise pikkus päevades. Väljasta ka ametikohad, kus pole ühtegi töötamist – nende korral on arv null ja keskmine teadmata (NULL).

Ülesanne S3_3: Leia iga isiku kohta temaga seotud töötamiste arv. Väljasta isiku ID, meiliaadress ja arv. Kui isikul töötamisi ei ole, siis on arv 0.

4.3.4 Detailandmete väljastamine

Valiku põhjendus: Päringud küsivad andmeid erinevatelt hierarhia tasemetelt ning hierarhilise andmete esituse korral lähtuvad vajadusest alustada otsingut erinevatelt tasemetelt.

Ülesanne S4_1: Leia iga isiku kohta, kes on töötaja, tema meiliaadress, perenimi, töötaja seisundi liigi nimetus ja töötamised.

Ülesanne S4_2: Leia iga töötamise kohta selle alguse aeg, lõpu aeg, ametikoha nimetus ning töötaja meiliaadress ja perenimi.

4.3.5 Hierarhiliste andmete otsimine

Valiku põhjendus: Andmeid võib esitada hierarhiliselt või mitte-hierarhiliselt, kuid nende andmete hulgas on ka olemuselt hierarhilised andmed – nimelt on igal töötajal null või üks mentorit, kes on samuti töötajad.

Ülesanne S5_1: Leia kõik sellised töötajad, kelle mentoriks oleva töötaja seisundi kood erineb selle töötaja enese seisundi koodist. Väljastada töötaja meiliaadress ja seisundi nimetus ning tema mentori meiliaadress ja seisundi nimetus.

4.3.6 Uute olemite registreerimine

Ülesanne S6_1: Lisa andmebaasi uus isik, kes on ühtlasi ka töötaja. Kasuta atribuutide väärtustena vabalt valitud väärtuseid.

Ülesanne S6_2: Lisa konkreetsele töötajale uus töötamine. Kasuta atribuutide väärtustena vabalt valitud väärtuseid.

4.3.7 Olemasolevate andmete uuendamine

Valiku põhjendus: Esimese stsenaariumi korral soovitakse uuendada ühe konkreetse olemi andmeid, kusjuures teada on selle olemi unikaalne identifikaator. Teise stsenaariumi korral soovitakse uuendada mitme olemi andmeid, kontrollides selleks seotud andmeid.

Ülesanne S7_1: Uuenda andmebaasis olevat konkreetset isikut (ette on antud meiliaadress), registreerides tema telefoninumbri.

Ülesanne S7_2: Määra konkreetse isiku (ette on antud meiliaadress), konkreetsel ametikohal konkreetse alguse ajaga töötamisel uus töötamise lõpu aeg.

Ülesanne S8_1: Kustuta andmebaasist selliste isikute aadressid, kellel on seotud tööalane suhe kus ametikood on vahemikus 10 kuni 30.

Ülesanne S8_2: Kustuta andmebaasist selliste töötamiste lõpu aeg, millega seotud töötajad on isiku seisundis 1 või 2 ning töötajaks olev isik on seotud riigiga Eesti.

4.3.8 Olemite kustutamine

Valiku põhjendus: Esimese stsenaariumi korral soovitakse kustutada ühte konkreetset olemit, kusjuures teada on selle olemi unikaalne identifikaator. Teise stsenaariumi korral soovitakse kustutada mitu olemit, kontrollides selleks seotud andmeid.

Ülesanne S9_1: Kustuta andmebaasist etteantud meiliaadressiga isik.

Ülesanne S9_2: Kustuta andmebaasist konkreetse isiku (ette on antud meiliaadress), konkreetsel ametikohal konkreetse alguse ajaga töötamine.

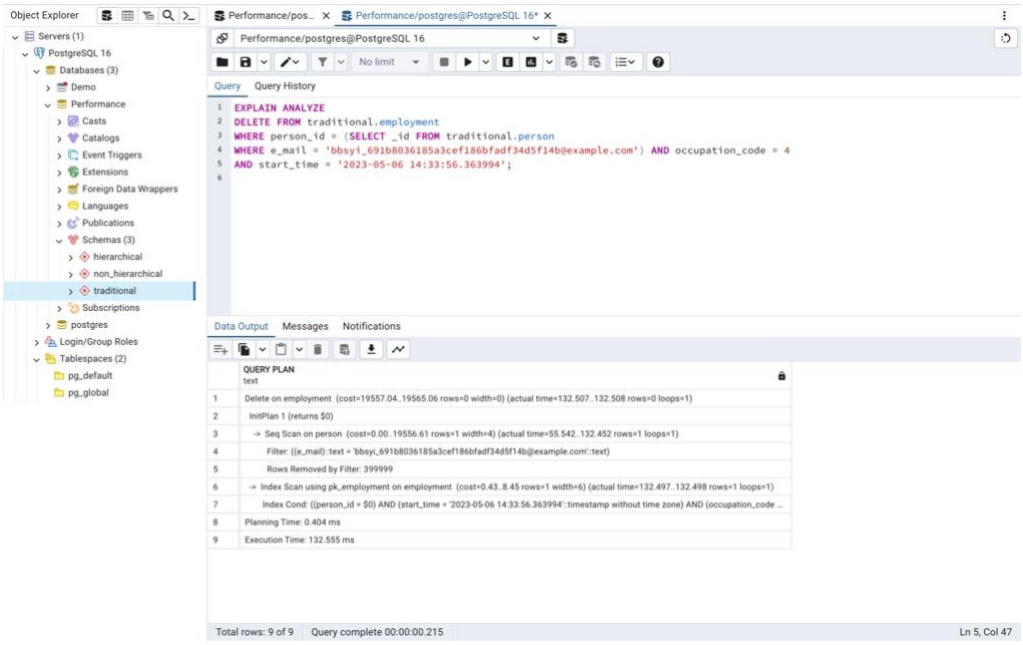
Ülesanne S10_1: Kustuta andmebaasist isikud, kellel on seotud tööalane suhe, kus ametikood on vahemikus 10 kuni 30.

Ülesanne S10_2: Kustuta andmebaasist töötamised, millega seotud töötajad on isiku seisundis 1 või 2 ning töötajaks olev isik on seotud riigiga Eesti.

4.4 Tulemuse leidmiseks kasutatud vahendid

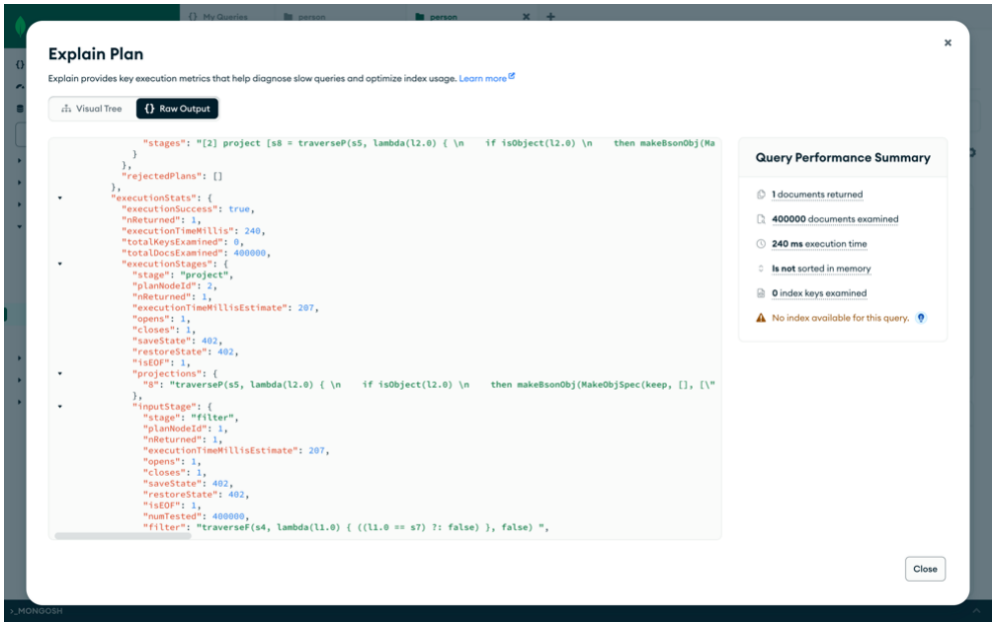
PostgreSQLis oli võimalik näha eraldi lause täitmise plaani koostamise (planeerimise) aega ja plaani täitmise aega. Tulemustes esitatakse ka nende kahe summa. MongoDB korral leiti ainult täitmise aeg kokku, mis hõlmab endas mistahes sisemist töötlust, mida MongoDB nende lausete täitmiseks teeb.

Mõõtmiste jaoks kasutati PostgreSQL puhul *EXPLAIN ANALYZE* [31] lauset (vt Joonis 5). Enne seda veel käivitati *ANALYZE* [32] lause, et uuendada andmebaasi statistikat, mida andmebaasisüsteem vajab lausete täitmisplaani koostamiseks. Autor kasutas lausega *ANALYZE* valikut *VERBOSE*, mis väljastab edusammude teateid ja millist tabelit parasjagu töödeldakse. *ANALYZE* lause käivitati, kui kõigis skeemides sisestatud, muudetud või kustutatud suurem hulk andmeid, samuti kui näiteks sisestati uus isik, kellega seotud andmetes tehti muudatusi järgmiste eksperimentide läbiviimisel.



Joonis 5. Näide PostgreSQL (T skeem) kustutamise mõõtmistulemused *EXPLAIN ANALYZE* lausega.

MongoDB Compass rakenduses oli eraldi *Explain* nupp, millega sai vaadata otsimise päringute täimise plaani, kus olid ka välja toodud väljastatud dokumentide arv ja täitmise aeg. Valides *Raw Output* sai vaadata päringu täitmise plaani täpsemalt, kus oli samuti välja toodud päringu täitmise aeg millisekundites (vt Joonis 6).



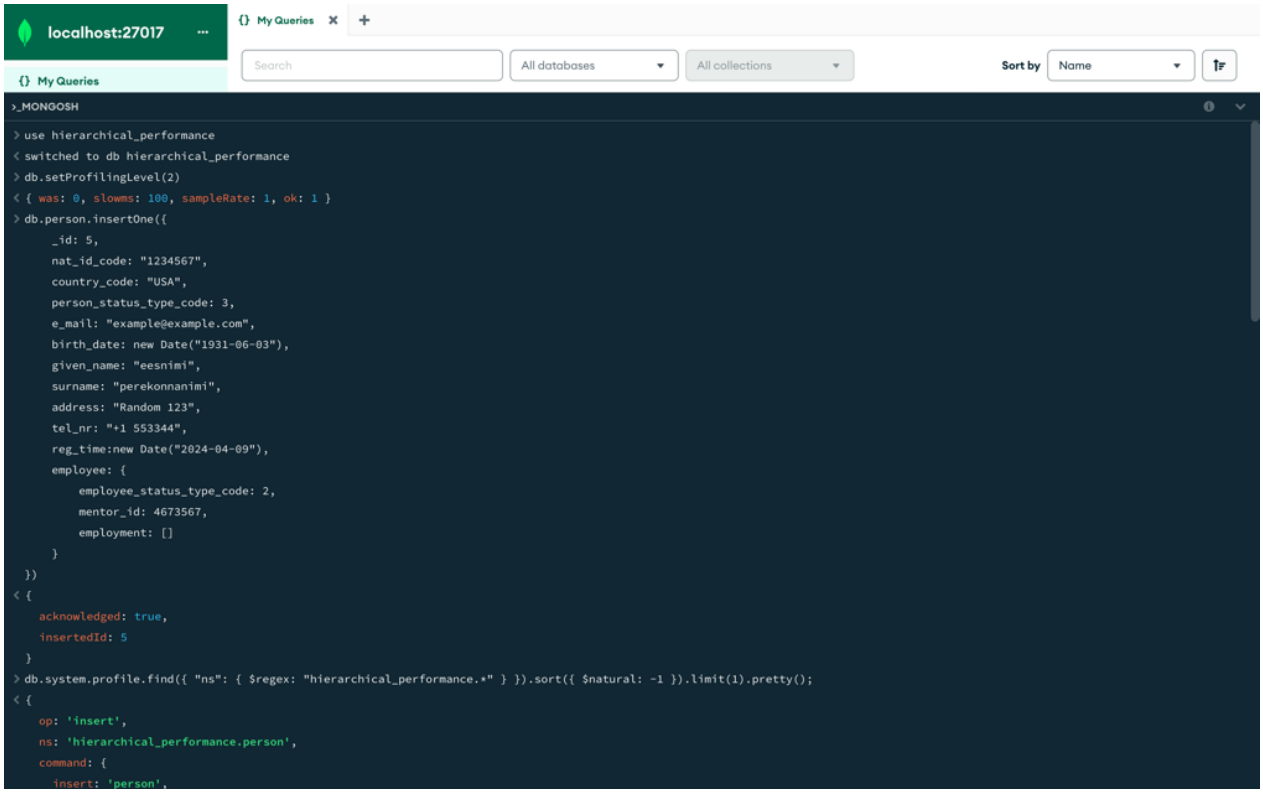
Joonis 6. Päringu EXPLAIN PLAN MongoDB Compass vaade (*Raw Output*).

MongoDB Shell käsurea tööriista puhul oli samuti võimalus päringu lõpus kasutada *explain* lauset (vt Joonis 7), et uurida päringu täitmise statistikat ja vaadata ka täitmise aega.

```
db.person.find({e_mail:
'acscu_ed69216fed359e36bd52421c86d40902@example.com'},
{_id: 1, surname: 1}).explain("executionStats");
```

Joonis 7. Explain lause kasutamine koodis Mongo Shell käsurea tööriistas.

Sisestamise, uuendamise ja kustutamise lausete täitmise statistika nägemiseks tuli kasutada MongoDB Shell käsurea tööriista. Mõõtmistulemuste nägemiseks vajalikud Shelli sisestatavad laused, on välja toodud joonisel 8, kus näitena on kasutatud andmete sisestamist MongoDB hierarhilise disainiga andmebaasi.



Joonis 8. Andmete sisestamise (HM skeem ülesanne 6_1) mõõtmistulemuste vaatamine Mongo Shell'is.

5 Eksperimendi tulemused

Käesolevas peatükis esitatakse eksperimendi alusel leitud mõõtmistulemused kõige suurema andmemahu – T3 korral (vt Tabel 1). Eksperimendi tulemused andmemahu T1 korral on lisas 7. Eksperimendi tulemused andmemahu T2 korral on lisas 8.

Mõõtmiste tulemused esitatakse eraldi tabelitena. Mõõtmiseks kasutatud lauseid on võimalik vaadata varem mainitud *GitHubi* pesas „*Experiments*“ kaustas [27, Ex 1]. Iga lause puhul on sinise värviga tõstetud esile kõige väiksem (parem) täitmise kiirus ja punase värviga kõige suurem (halvem) täitmise kiirus. Pearsoni korrelatsioonikordaja korral tähistatakse punasega need väärtused, mis näitavad nõrka seost. Selles peatükis on esitatud kõigi andmebaasi disainide ja ülesannete puhul arvutatud Pearsoni korrelatsioonikordaja kokkuvõtlikus tabelis. Tabelid iga disaini korral eraldi koos mõõtmistulemustega on välja toodud lisas 9.

Iga eksperimendi jaoks tehti kolm mõõtmist ja saadud tulemustest arvutati geomeetriline keskmine, mis esitatakse millisekundites. Geomeetrilist keskmist võiks kasutada siis kui tulemused sõltuvad üksteisest, nagu siis kui lause korduval täitmisel taaskasutatakse koostatud täitmisplaani ja muutmällu loetud andmeid [33]. Arvutamiseks kasutatakse *Google Sheets* geomeetrilise keskmise *GEOMEAN()* valemit. Mõõdetakse ka kiirust erinevate andmehulkade korral, et selgitada välja tabelites oleva andmemahu ja operatsioonide töökiiruse seost. Selleks täpsema hinnangu andmiseks arvutatakse välja Pearsoni korrelatsioonikordaja, kasutades selleks *Google Sheets CORREL()* valemit. Töökiiruseid mõõdetakse samuti kaks ja neli korda suuremate andmemahtudega.

5.1 Funktsiooni kasutus ja sorteerimine

Tabel 2. Funktsiooni kasutus ja sorteerimine (andmemaht T3) mõõtmistulemused.

Eksperiment	Aeg (millis.)	T	H	M	HM	MM
1_1	täitmine	561,737	2692,530	1408,745		
	planeerimine	0,511	0,262	0,367		
	kokku	**562,248**	**2692,792**	**1409,112**	**959,664**	**903,293**
1_2	täitmine	1977,966	10287,693	3669,828		
	planeerimine	0,671	1,440	1,683		
	kokku	**1978,637**	**10289,133**	**3671,511**	**6683,766**	**19904,966**

Tabelis 2 on näha, et kõige tõhusamaks osutus funktsiooni kasutuse ja sorteerimise mõlema ülesande korral PostgreSQL traditsioonilise skeemi disain. Esimesel mõõtmisel (T1 andmemahuga) oli ülesandega 1_1 kiireim MognoDB hierarhiline disain, kuigi erinevus PostgeSQL traditsioonilise skeemi töökiirusega oli peaaegu märkamatu (vt Lisa 7). Kõige aeglasem oli esimese eksperimendi puhul PostgreSQL hierarhilise skeemi disain. Samas planeerimise aeg oli selle disaini korral kõige kiirem. Teise eksperimendi mõõtmistulemustes, kus hierarhilise esituse korral ei olnud otsitavad andmed kõige ülemisel tasemel, oli kõige aeglasem MongoDB mitte-hierarhiline andmebaasi disain.

5.2 Üksiku olemi otsimine

Juhul kui andmed olid hierarhilise esituse korral kõige ülemisel tasemel (ülesanne 2_1), siis oli halvim mõõtmistulemus üksiku olemi otsimisel PostgreSQL hierarhilisel disainil ning parim MongoDB mitte-hierarhilisel andmebaasi disainil. Kuigi lisades 7 ja 8 on näha, et selle ülesande juures oli iga andmemahu (T1, T2, T3) korral kiireima mõõtmistulemusega disain erinev – T1 andmemahuga oli selleks PostgreSQL traditsiooniline ja T2 korral mitte-hierarhiline disain. Teise ülesande mõõtmistulemusi vaadates oli parima kiirusega taas PostgreSQL traditsiooniline disain ning halvim MongoDB mitte-hierarhiline disain (vt Tabel 3).

Tabel 3. Üksiku olemi otsimine (andmemaht T3) mõõtmistulemused.

Eksperiment	Aeg (millis.)	T	H	M	HM	MM
2_1	täitmine	252,736	2810,935	1325,546		
	planeerimine	0,073	0,293	1,497		
	kokku	**252,810**	**2811,228**	**1327,043**	**227,525**	**176,686**
2_2	täitmine	473,551	1197,218	837,951		
	planeerimine	0,801	0,181	0,861		
	kokku	**474,353**	**1197,399**	**838,812**	**519,069**	**7557,985**

5.3 Koondandmete leidmine

Koondandemete leidmisel ülesannetes 3_1 ja 3_2, kus oli vaja ka keskmine leida, olid kõige paremate mõõtmistulemustega PostgreSQL traditsiooniline disain ning MongoDB hierarhiline disain. Siiski on tabelis 4 näha, et kui andmed asetsesid hierarhilisel esitlusel hierarhias madalamal tasemel (ülesanne 3_2), siis MongoDB hierarhilise disaini korral

oli koondandmete leidmine hoopis kõige aeglasem. Teistel ülesannetel (ülesanne 3_1 ja 3_2) oli kõige aeglasema mõõtmistulemusega jällegi MongoDB mitte-hierarhiline disain.

Koondandmete leidmise ülesandes, kus polnud vaja leida keskmist (ülesanne 3_3) oli parima töökiirusega PostgreSQL traditsiooniline disain. Kuigi eelnevate andmemahtudega mõõtmisel (vt Lisa 7 ja Lisa 8) oli ülesande 3_3 puhul kõige kiireim mõõtmistulemus hoopis MongoDB hierarhilise disaini korral.

Tabel 4. Koondandmete leidmine (andmemaht T3) mõõtmistulemused.

Eksperiment	Aeg (millis.)	T	H	M	HM	MM
3_1	täitmine	505,916	1373,182	1228,114		
	planeerimine	0,854	0,540	0,585		
	kokku	**506,771**	**1373,722**	**1228,699**	**560,162**	**35349,436**
3_2	täitmine	1419,287	7646,741	4260,640		
	planeerimine	0,433	0,274	0,462		
	kokku	**1419,720**	**7647,015**	**4261,102**	**180264,846**	**173473,610**
3_3	täitmine	1522,028	2586,734	2193,257		
	planeerimine	14,999	0,179	8,476		
	kokku	**1537,027**	**2586,912**	**2201,733**	**1946,427**	**9539,118**

5.4 Detailandmete väljastamine

Tabel 5. Detailandmete väljastamine (andmemaht T3) mõõtmistulemused.

Eksperiment	Aeg (millis.)	T	H	M	HM	MM
4_1	täitmine	4958,524	837,908	2591,043		
	planeerimine	3,501	4,743	14,090		
	kokku	**4962,026**	**842,651**	**2605,133**	**4239,762**	**91355,413**
4_2	täitmine	3726,433	3259,468	11966,630		
	planeerimine	1,996	0,487	2,391		
	kokku	**3728,429**	**3259,954**	**11969,022**	**203838,079**	**67128,247**

Tabelis 5 on välja toodud, et detailandmete väljastamisel olid kõige paremad mõõtmistulemused seekord PostgreSQL hierarhilise disaini korral. Kui andmed asetsesid hierarhias kõige ülemisel tasemel, siis oli kõige aeglasema mõõtmistulemusega MondoDB mitte-hierarhiline disain ning teisel juhul oli aeglasem hierarhiline disain.

Samas, kui vaadata lisast 7 mõõtmistulemusi andmemahuga T1, siis ülesande 4_2 korral on parim mõõtmistulemus PostgreSQL traditsioonilisel disainil ning lisast 8 vaadates oli andmemahuga T2 halvim mõõtmistulemus samal ülesandel MondoDB mitte-hierarhilisel disainil.

5.5 Hierarhiliste andmete otsimine

Tabel 6. Hierarhiliste andmete otsimine (andmemaht T3) mõõtmistulemused.

Eksperiment	Aeg (millis.)	T	H	M	HM	MM
5_1	täitmine	1438,163	1648,556	4390,137		
	planeerimine	9,183	1,327	15,113		
	kokku	**1447,346**	**1649,883**	**4405,250**	**85199,586**	**195003,357**

Hierarhiliste andmete otsimisel on tabelis 6 näha, et parim oli PostgreSQL traditsiooniline disain, kuigi planeerimise aeg oli hierarhilise disaini puhul kiirem. Samas andmemahuga T2 oli kiirem hoopis PostgreSQL hierarhiline disain. Märgatavalt halvim oli jällegi MongoDB mitte-hierarhiline disain.

5.6 Uute olemite registreerimine

Võrreldes eelnevate eksperimentidega oli uute olemite registreerimise ülesannetes T3 andmemahu korral hoopis kõige kiiremad MongoDB andmebaaside disainid ning kõige aeglasem PostgreSQL hierarhiline disain. Hierarhiasse ülemisele tasemele (ülesanne 6_1) andmete lisamisel oli kiireim MongoDB hierarhiline disain, samas ülesande 6_2 lahendamise korral oli kiireim mitte-hierarhiline disain.

Tabelist 7 tuleb välja, et hierarhias madalamal tasemel asetsevate andmete salvestamisel (ülesanne 6_2) on ka MongoDB hierarhiline disain üsna aeglane. Muuhulgas ülesande 6_1 oli esimese (T1) ja teise (T2) andmemahu korral kõige aeglasem PostgreSQL mitte-hierarhiline disain (vaata lisa 7 ja 8).

Tabel 7. Uute olemite registreerimine (andmemaht T3) mõõtmistulemused.

Eksperiment	Aeg (millis.)	T	H	M	HM	MM
6_1	täitmine	12,192	27,572	14,882		
	planeerimine	0,043	0,183	0,029		
	kokku	**12,234**	**27,754**	**14,911**	**0,007**	**0,257**
6_2	täitmine	8,529	1099,211	21,259		
	planeerimine	0,042	0,533	0,037		
	kokku	**8,571**	**1099,744**	**21,295**	**814,111**	**0,141**

5.7 Olemasolevate andmete uuendamine

Kõigi andmemahtude korral jäid olemasolevate andmete uuendamise mõõtmistulemused peaaegu samaks, ainuke erinevus oli T2 mõõtmisel ülesande 7_1 töökiiruses, kus kõige

halvem tulemus oli PostgreSQL mitte-hierarhilisel disainil. Hierarhias kõige ülemisel tasemel kindla olemi andmete uuendamisel (ülesanne 7_1) oli kiireim PostgreSQL traditsiooniline disain ning aeglaseim MongoDB mitte-hierarhiline disain. Kindla olemi puhul, mille andmed olid hierarhias madalamal tasemel, oli parim mõõtmistulemus MongoDB mitte-hierarhilise andmebaasi disainil, kus tulemus võrreldes teiste disainidega oli väga väike (vt Tabel 8). Kõige halvem aga oli MongoDB hirearhilise disaini mõõtmistulemus.

Tabel 8. Olemasolevate andmete uuendamine (andmemaht T3) mõõtmistulemused.

Eksperiment	Aeg (millis.)	T	H	M	HM	MM
7_1	täitmine	150,760	438,870	286,386		
	planeerimine	0,380	0,246	0,459		
	kokku	**151,140**	**439,116**	**286,845**	**429,538**	**780,712**
7_2	täitmine	146,397	348,981	207,896		
	planeerimine	0,499	0,229	0,281		
	kokku	**146,895**	**349,209**	**208,177**	**575,828**	**0,004**
8_1	täitmine	18548,988	35897,545	27668,574		
	planeerimine	2,241	0,807	8,516		
	kokku	**18551,229**	**35898,352**	**27677,089**	**20089,547**	**15767,631**
8_2	täitmine	3881,353	2403,087	4760,256		
	planeerimine	3,210	2,424	4,448		
	kokku	**3884,563**	**2405,510**	**4764,704**	**1008,257**	**1221,374**

Kui uuendamisele minevad andmed olid mitme olemi muutmisel hierarhias kõige ülemisel tasemel, oli kiireim jällegi MongoDB mitte-hierarhiline disain, kõige aeglasemaks kujunes PostgreSQL hierarhiline disain. Ülesande 8_2 lahendamisel oli kiireim MongoDB hierarhiline disain ja aeglaseim PostgreSQL mitte-hierarhiline disain.

5.8 Olemite kustutamine

Tabelist 9 võib välja lugeda, et olemite kustutamisel, kui sooviti kustutada ühte konkreetset olemit ja andmed olid hierarhilise struktuuri korral kõige ülemisel tasemel (ülesanne 9_1), siis kiireim disain oli PostgreSQL traditsiooniline ja kõige aeglasem hierarhiline (T1 andmemahu korral MongoDB hierarhiline). Ülesande 9_2 puhul olid kiireima töökiirusega MongoDB disainid ning mõlema puhul olid mõõtmistulemused kõigi andmemahtude korral (vt Lisa 7 ja Lisa 8) väga palju väiksemad teiste disainide mõõtmistulemustest. Kõige aeglasem oli aga sama ülesande lahendamisel PostgreSQL hierarhiline disain.

Mitme olemi kustutamisel, kus kustutamisele minevad andmed olid hierarhias kõige ülemisel tasemel, oli parim mõõtmistulemus hierarhilisel PostgreSQL disainil ja halvim tulemus mitte-hierarhilisel disainil. Teisel juhul (ülesanne 10_2) oli mitme olemi kustutamise halvim mõõtmistulemus PostgreSQL hierarhilisel disainil ja parim MongoDB hierarhilisel disainil. Kusjuures T2 andmemahuga (vt Lisa 8) oli parim tulemus hoopis MongoDB mitte-hierarhilisel disainil ja halvim mitte-hierarhilisel PostgreSQL disainil. Lisaks võib välja tuua, et kuigi kokku mõõtmistulemus oli ülesande 10_2 korral halvim PostgreSQL hierarhilisel disainil, siis planeerimisaeg oli samal disainil selle ülesandega kõige kiirem.

Tabel 9. Olemite kustutamine (andmemaht T3) mõõtmistulemused.

Eksperiment	Aeg (millis.)	T	H	M	HM	MM
9_1	täitmine	158,679	2457,802	246,967		
	planeerimine	0,195	1,743	0,082		
	kokku	**158,874**	**2459,545**	**247,049**	**332,625**	**298,027**
9_2	täitmine	121,717	911,275	208,327		
	planeerimine	0,704	1,847	0,225		
	kokku	**122,421**	**913,122**	**208,551**	**0,002**	**0,084**
10_1	täitmine	18256,709	7018,472	24136,416		
	planeerimine	0,680	2,820	5,706		
	kokku	**18257,389**	**7021,292**	**24142,122**	**11488,869**	**9867,025**
10_2	täitmine	953,908	1395,537	1114,556		
	planeerimine	9,548	2,666	9,244		
	kokku	**963,456**	**1398,203**	**1123,800**	**418,526**	**604,396**

5.9 Mõõtmiste kokkuvõte

Tabelis 10 tähistab X, et vastav disain andis vastava ülesande puhul kõige parema tulemuse (töökiirus oli väikseim) ja O, et kõige halvema tulemuse (töökiirus oli suurim), Tabeli alaosas on näha kui palju esines erinevate disainide korral kokku kõige paremaid ja halvemaid tulemusi. Kõige parem tulemus oli PostgreSQLis 11-l korral ja MongoDB-s 9-l korral. Kõige halvem tulemus oli mõlemas andmebaasisüsteemis 10-l korral.

Tabel 10. Kõige parema ja halvema mõõtmistulemuse kokkuvõte andmemahu T3 korral.

Eksperiment	T	H	M	HM	MM
1_1	X	O			
1_2	X				O
2_1		O			X
2_2	X				O
3_1				X	O
3_2	X			O	
3_3	X				O
4_1		X			O
4_2		X		O	
5_1	X				O
6_1		O		X	
6_2		O			X
7_1	X				O
7_2				O	X
8_1		O			X
8_2			O	X	
9_1	X	O			
9_2		O		X	
10_1		X	O		
10_2		O		X	
Kokku (X)	8	3	0	5	4
Kokku (O)	0	8	2	3	7

PostgreSQL-is näitas parimaid tulemusi traditsiooniline disain, mille mõõtmistulemused polnud kordagi kõige halvemad ning olid parimad kaheksal korral. Halvimaid tulemusi näitas PostgreSQL-is hierarhiline disain, kus kõige halvemaid mõõtmistulemusi oli parimatest viie võrra rohkem. MongoDB disaine võrreldes oli veidi paremate tulemustega hierarhiline disain, kuna hierarhilises disainis oli parimaid tulemusi ühe võrra rohkem ja kõige halvemaid mõõtmistulemusi nelja võrra vähem.

Võrreldes PostgreSQL disaine, mis kasutavad JSON dokumente ja MongoDB disaine (H, M, HM, MM), siis kõige rohkem parimaid mõõtmistulemusi oli MongoDB hierarhilisel disainil ja kõige suurem arv halvimaid mõõtmistulemusi PostgreSQL hierarhilisel disainil.

5.10 Pearsoni korrelatsioonikordaja

Korrelatsioonikordaja positiivne väärtus tähendab, et tunnuste vahel on kasvav seos, kus ühe tunnuse väärtuse kasvades kasvab ka teine. Negatiivne korrelatsioonikordaja väärtus tähendaks kahanevat seost, kus ühe tunnuse väärtuse kasvades teise väärtus kahaneb [34].

Iga disaini korral tehti mõõtmisi kolme erineva andmehulgaga, et uurida kas andmehulga ja töökiiruste muutuse vahel on seoseid. Tabelis 11 on näidatud, kuidas enamus ülesannete lahendamisel on andmemahu ja täitmisaja vahel tugev kasvav lineaarne seos nii PostgreSQL kui ka MongoDB andmebaasi disainides. See tähendab korrelatsioonikordaja on ligidal arvule üks ehk tugev lineaarne seos jääb [0,5; 1] või [-0,5; -1] vahemikku. [28, Ch. 1]

Tabel 11. Pearsoni korrelatsioonikordaja kõigi eksperimentide korral.

Ülesanne	Pearsoni korrelatsioonikordaja				
	T	H	M	HM	MM
1_1	0,998423	0,900802	0,929378	0,847394	0,960975
1_2	0,968515	0,980013	0,998221	0,931195	0,978635
2_1	0,977044	0,960429	0,942858	0,874519	0,902763
2_2	0,946013	0,728435	0,946510	0,999989	0,998783
3_1	0,985813	0,999547	0,989055	0,991995	0,999865
3_2	0,998951	0,921218	0,998171	0,998817	0,998518
3_3	0,999999	0,988921	0,691185	0,976438	0,999256
4_1	0,995485	0,997484	**−0,059363**	0,991988	0,994110
4_2	1,000000	0,999071	0,999985	0,889710	0,996167
5_1	0,758951	0,999242	0,858414	0,994056	0,999232
6_1	0,962054	0,986459	0,581613	**0,173137**	0,945630
6_2	**0,483798**	0,985936	**0,096677**	0,984014	0,954494
7_1	0,962855	0,995284	0,954749	0,975624	0,970246
7_2	0,938412	0,998363	0,999979	0,997740	0,944911
8_1	0,999693	0,994085	0,996848	0,994864	0,998830
8_2	0,987827	0,819573	0,997240	0,969577	0,991833
9_1	0,987924	0,980098	**0,438922**	0,958486	0,989874
9_2	0,999596	0,998856	0,627505	0,944911	0,944911
10_1	0,998741	0,761410	0,999742	0,994582	0,999947
10_2	0,999990	0,993624	0,946817	0,996160	0,980937

Kokku tuli vaid viis erandit, kus kahe ülesande korrelatsioonikordaja tuli keskmise tugevusega (märgitud tabelis rasvase kirjaga) ning kolme ülesande korral oli see väga nõrgalt esinev (märgitud tabelis punase rasvase kirjaga). Nendest omakorda ainult ühel ülesandel tuli Pearsoni korrelatsioonikordaja negatiivne (andmemahu kasvades lause täitmise aeg kahaneb) ja väga nõrgalt esinev. Keskmine korrelatsioonikordaja tuli

PostgreSQL traditsioonilise disaini uue olemi registreerimisel (ülesanne 6_2) ja mitte-hierarhilise disaini üksiku olemi kustutamisel (ülesanne 9_1). Väga nõrk kahanev seos oli PostgreSQL mitte-hierarhilise disaini detailandmete väljastamisel (ülesanne 4_1) ning väga nõrk kasvav seos oli sama disaini üksiku olemi registreerimisel (ülesanne 6_2). Lisaks esines ka MongoDB hierarhilise disaini puhul uue olemi registreerimisel (ülesanne 6_1) kõigest väga nõrk seos andmehulga suurenemise ja töökiiruse vahel.

5.11 Indeksite kasutamine PostgreSQL andmebaasis

Selles jaotises kirjutatakse indeksite kasutamisest. Andmemahuga T3 (vt Tabel 1) üksiku olemi otsimise ülesannetega 2_1 ja 2_2 uuriti, kas indeksite kasutamine muudab tulemusi PostgreSQL andmebaasi disainide mõõtmistulemustes.

5.11.1 Päringute ümberkirjutamine, et kasutataks indeksit

Lower() funktsioonil põhinev indeks, mis loodi PostgreSQL traditsioonilise disaini puhul lausega (vt Joonis 9), oli juba andmebaasi loomisel Roopärgi tööst võetud koodis olemas. Kusjuures, *Lower()* funktsioonil põhinevad indeksid olid olemas ka PostgreSQL hierarhilise ja mitte-hierarhilise andmebaasi disainide loomise koodis. Esialgsetes katsetustes ei kasutatud otsingutingimustes *Lower()* funktsiooni ja seetõttu ei kasutanud andmebaasisüsteem lause täitmiseks ka indeksit.

```
CREATE UNIQUE INDEX idx_ak_person_e_mail ON Person(LOWER(e_mail));
```

Joonis 9. *Lower()* funktsioonil põhinev indeks PostgreSQL skeemis T.

Ümberkirjutatud ülesandes 2_1 kasutati *Lower()* funktsiooni (vaata Joonis 10) PostgreSQL traditsioonilises skeemis (lisaks autori enda uudishimu tõttu hierarhilises ja mitte-hierarhilises skeemis).

```
SELECT _id, surname FROM traditional.person
WHERE LOWER(e_mail) = 'acscu_ed69216fed359e36bd52421c86d40902@example.com';
```

Joonis 10. *Lower()* funktsiooni kasutamine ülesandes 2_1 PostgreSQL skeemis T.

Tabelis 12 võib näha, et võrreldes tabelis 3 esitatud algsete tulemustega on töökiirused märgatavalt paremad ning kõige parem tulemus oli mitte-hierarhilise skeemil ja halvim hierarhilisel.

Tabel 12. Mõõtmistulemused *Lower()* funktsiooni kasutamise korral PostgeSQL andmebaasis (andmemaht T3) .

Eksperiment	Aeg (millis.)	T	H	M
2_1	täitmine	0,088	0,067	0,054
	planeerimine	0,145	0,388	0,146
	kokku	**0,233**	**0,455**	**0,200**

5.11.2 GIN indeksite kasutamine

Järgmisena uuriti, kuidas mõjutab mõõtmistulemusi GIN indeksite lisamine skeemidesse PostgreSQL disainide puhul, mis kasutavad JSON dokumente.

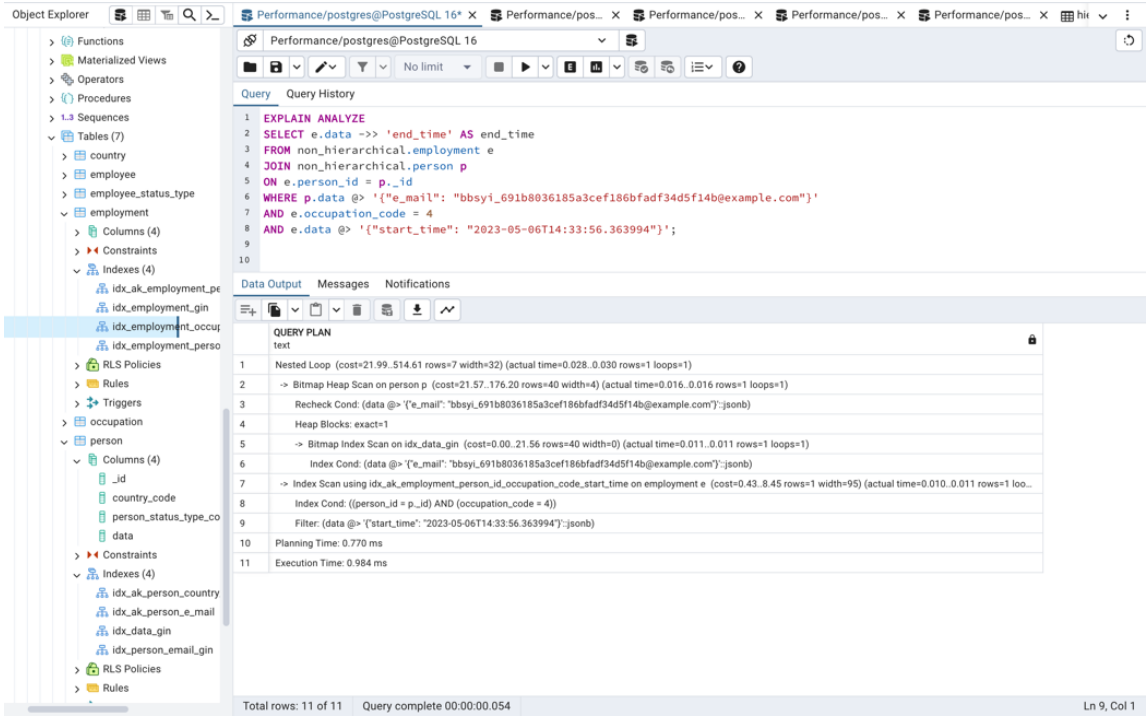
Tabelis 13 on esile toodud töökiirused, kui lisati hierarhilisse ja mitte-hierarhilisse disaini GIN indeksid (vaata Lisa 10) ning nende põhjal muudeti ülesannete lahendamiseks lauseid, et tulemuste leidmisel oleks kasutatud indekseid. Tänu loodud indeksitele on võimalik tõhusalt otsida suure hulga JSONB-väärtuste sees esinevate võtmete või võtme/väärtuse paare. Kuna indeksite loomisel kasutati GIN-i operaatorite klassi jsonb_path_ops, siis see ei toeta võtme eksistents operaatoreid. Seetõttu muudeti ülesandeid ning kasutati lausetes operaatoreid @>, @? ja @@, et päring kasutaks loodud indekseid (vt Lisa 10). GIN indeksite testimise päringutes ei kasutatud *Lower()* funktsiooni.

Tabel 13. PostgreSQL H ja M (andmemahuga T3) mõõtmistulemused kasutades GIN indekseid.

Eksperiment	Aeg (millis.)	H	M
2_1	täitmine	0,863	1,879
	planeerimine	1,505	0,986
	kokku	**2,368**	**2,865**
Eksperiment	**Aeg (millis.)**	**H**	**M**
2_2	täitmine	0,131	0,714
	planeerimine	0,978	2,342
	kokku	**1,109**	**3,056**

Kui võrrelda seda tulemustega tabelis 3, siis jällegi paranesid töökiirused väga palju ja paremad tulemused olid hierarhilisel skeemil. Joonisel 11 on kujutatud, milline näeb välja

päringu täitmisplaan, kui ülesannete lahendamisel mitte-hierarhilise disaini korral on PostgreSQL-is kasutusel GIN indeksid koos tabelite koostamisel lisatud indeksitega.



Joonis 11. Päringu täitmisplaan indeksite kasutamisel PostgreSQL disainis M.

6 Eksperimentide tulemuste analüüs

Selles peatükis analüüsitakse viiendas peatükis esitatud eksperimentide tulemusi. Antakse vastused uurimisküsimustele, võrreldakse töö tulemusi eelnevalt tehtud uuringutega, vaadatakse tagasi töö tegemisele ja tuuakse välja võimalikud edasised uurimissuunad.

6.1 Vastused uurimisküsimustele

Küsimus: Millised on erinevate andmebaasi disainide korral erinevat tüüpi andmebaasioperatsioonide töökiirused?

Vastus: Töö käigus võrreldi PostgreSQL traditsioonilise, hierarhilise, mitte-hierarhilise ning MongoDB hierarhilise ja mitte-hierarhilise disainide töökiiruseid erinevate andmebaasioperatsioonide ja kasvavate andmehulkade korral. Järjepidevalt näitas parimaid mõõtmistulemusi PostgreSQL traditsiooniline disain võrreldes teiste disainidega ja MongoDB-ga. Selle andmebaasisüsteemi traditsioonilise skeemi mõõtmistulemused polnud kordagi kõige halvemate hulgas. Enamuse otsinguga seotud ülesannete korral (jaotised 5.1 – 5.4) esimese andmemahuga (T1) andsid vaheldumisi parimaid mõõtmistulemusi PostgreSQL traditsiooniline ja MongoDB hierarhiline disain.

Andmemahtude suurenemisel jäi siiski parimaid tulemusi andma PostgreSQL traditsiooniline disain. Otsinguga seotud ülesannete korral näitas järjepidevalt halvemaid tulemusi MongoDB mitte-hierarhiline disain. Samuti oli PostgreSQL hierarhiline disain otsingutega seotud ülesannetes üsna aeglaste töökiirustega. Siiski kui vaadata just detailandmete väljastamist ja hierarhiliste andmete otsimist, siis olid parimad mõõtmistulemused PostgreSQL hierarhilisel disainil ja halvimad MongoDB disainidel.

Uute olemite registreerimisega olid tulemused peaaegu vastupidised ning parimaid tulemusi näitasid MongoDB mitte-hierarhiline ja hierarhiline disain, juhul kui andmed olid hierarhilise esituse korral kõige ülemisel tasemel. MongoDB hierarhiline esitus, kui andmed asusid hierarhiliselt madalamal tasemel, andis halvimaid tulemusi. Samuti andis PostgreSQL hierarhiline disain uute olemite sisestamisel kõige halvemaid mõõtmistulemusi.

Kindla olemi uuendamisel näitas parimaid tulemusi PostgreSQL traditsiooniline disain. Muuhulgas, kui andmed asusid hierarhias madalamal tasemel, siis oli parim mõõtmistulemus MongoDB mitte-hierarhilisel disainil ning halvim hierarhilisel disainil. Samas kui andmed asusid hierarhias kõrgemal tasemel, oli taas halvima tulemustega MongoDB mitte-hierarhiline disain. Mitme olemi muutmisel oli parimad töökiirused MongoDB mitte-hierarhilisel disainil ja halvimad PostgreSQL JSON dokumente kasutavatel disainidel.

Ühe kindla olemi kustutamisel hierarhias kõige ülemisel tasemel asetsevate andmete korral oli kiireima töökiirusega PostgreSQL traditsiooniline disain. Samasugusel andmete asetsemisel olid kindla olemi kustutamisel halvima tulemusega PostgreSQL ja MongoDB hierarhilised disainid. Samas andmete asetsemisel hierarhias allpool näitasid parimat töökiirust nii kindla olemi kui ka mitme olemi kustutamisel MongoDB disainid. Mitme olemi kustutamisel, kui andmed olid hierarhias kõige ülemisel tasemel, andis parima tulemuse PostgreSQL hierarhiline disain ning halvima mitte-hierarhiline disain.

Küsimus: Kas JSON dokumentide kasutamise või mittekasutamise korral saab tuua välja andmebaasisüsteemi (PostgreSQL või MongoDB), mille puhul on need kiirused üldiselt kõige paremad/halvemad?

Vastus: Mõõtmistulemuste järgi saab võrrelda ka andmebaasisüsteeme. Üdiselt oli andmete otsimisel märgatavalt parem PostgreSQL andmebaasisüsteem ja halvem MongoDB. Samas muude andmebaasioperatsioonide korral andsid mõlemad andmebaasisüsteemid häid tulemusi. JSON dokumentide kasutamisel, kui toimus detailandmete väljastamine, siis parimaid tulemusi näitas PostgreSQL hierarhiline disain. Teiste otsimisülesannete puhul andis PostgreSQL JSON dokumentide kasutamisel üsna keskmiseid tulemusi. Teiste andmebaasioperatsioonide korral näitas kokkuvõttes JSON dokumentide kasutamisel kõige halvemaid tulemusi PostgreSQL hierarhiline disain ning rohkem kõige paremaid tulemusi oli MongoDB-l.

Küsimus: Kas andmemahtude ja operatsioonide täitmise aja vahel on andmebaasides lineaarne kasvav sõltuvus või mitte?

Vastus: Töö käigus koguti kolme erineva andmehulga mõõtmistulemusi. Vaadates mõlemat andmebaasisüsteemi ja kõiki disaine, siis enamus ülesannete lahendamisel oli näha tugevat kasvavat lineaarset seost andmemahtude suurenemise ja operatsioonide töökiiruste vahel. Nende disainide puhul, kus ülesande lahendamise kiiruse ja andmehulkade suurenemise vahel oli keskmine või väga nõrk seos, tekkis see sellest, et keskmise andmehulgaga T2 (vt Lisa 7) töökiirus oli suurem kui suurima andmehulgaga T3 mõõtmistulemus.

Küsimus: Kas indeksid aitaksid PostgreSQLis üksikute olemite andmete otsimisel töökiirust parandada?

Vastus: Indeksite kasutamist töökiiruse parandamiseks vaadati PostgreSQL disainides suurima andmemahuga (T3). Eksperimentide läbimisel uuriti, kas kohe tabelite loomisel loodud indeksid, sealhulgas unikaalsuse indeksid, ning hiljem lisatud GIN indeksid mõjutasid töökiirust. Tulemusena leiti, et indeksite kasutamisel üksikute olemite andmete otsimisel paranes töökiirus mitmekordselt iga PostgreSQL disainiga. Indekseid kasutades olid ülesannete 2_1 ja 2_2 töökiirused vaid mõni millisekund või alla selle. Juhul kui indekseid ei kasutatud, siis olid tulemused mõnisada millisekundit või mõnede disainide korral kuni mõned sekundid.

Küsimus: Kas saadud tulemused on kooskõlas varasemate samade andmebaasisüsteemide jõudluse uuringutega?

Vastus: Üldiselt ühtisid PostgreSQL traditsioonilise disainiga saadud mõõtmistulemused varasemate uuringutega. MognoDB-ga saadud tulemused sarnanesid eelnevate uuringutega, kuid oli ka erinevusi – näiteks praeguse lõputöö mõõtmistulemustes polnud mõnikord andmemahu kasvamisel jõudluse kahanemine järjepidev. Põhjalikumalt käsitletakse eelnevate uuringute ja antud töö tulemuste erinevusi ning sarnasusi jaotises 6.2.

6.2 Võrdlus olemasolevate töökiiruse uuringutega

Autor võrdles, kas saadud tulemused on kooskõlas varasemate samade andmebaasisüsteemide jõudluse uuringutega. 2024. aasta märtsis ilmunud artiklis „JSON performance: PostgreSQL vs MongoDB Comparison“ [22, Tab. 16] leiti, et väiksema andmemahuga stsenaariumites oli PostgreSQL otsingutes palju tõhusam, kuid kui andmekogum suurenes, siis täheldati PostgreSQLis jõudluse märgatavat langust. Teisalt, MongoDB näitas otsingutes järjepidevat head jõudlust. Praeguses töös ühtisid PostgreSQL otsinguga seotud operatsioonide tulemused kirjeldatud uuringuga, kuid MongoDB ei näidanud samasuguseid tulemusi. Praeguses töös halvenesid MongoDB puhul koos andmemahtude suurenemisega ka andmete otsingu operatsioonide mõõtmistulemused.

Tallinna Tehnikaülikoolis 2015. aastal magistritöös [23, Ch. 5] kirjutas Maksimov, et andmete lugemisega seotud PostgreSQL JSON-formaadis andmete põhiste andmebaasioperatsioonide tulemused on üsna identsed MongoDB tulemustega. Praeguses töös olid pigem PostgreSQL traditsioonilise disaini mõõtmistulemused mõnikord sarnased MongoDB JSON dokumente kasutava (hierarhilise) disaini mõõtmistulemustega. PostgreSQLi JSON dokumente kasutatavad disainid ei andud MongoDB-ga kuigi sarnaseid tulemusi. Samas ühtisid Maksimovi ja praeguse töö uuringute tulemused selles osas, et andmete kirjutamise jõudlus on MongoDB-s kiirem kui PostgreSQLis JSON-andmetüüpe kasutavate disainide korra. Samuti on mitme olemi uuendamisel operatsioonide jõudlus MongoDB-s parem.

Artiklis „Performance investigation of selected SQL and NoSQL databases“ [24, Ch. 5] on märgitud, et suureneva andmehulga puhul varieeruvad geofunktsioonide päringute korral MongoDB-s mõõdetud vastamisajad ainult umbes mõne sekundi. Praeguses lõputöö uuringus varieerusid MongoDB vastamisajad mõne sekundiga suureneva andmehulga korral just hierarhilises disainis, juhul kui toimus üksikute olemite sisestamine hierarhia ülaosas või hierarhias allpool olevate üksikute olemite kustutamine. Väikeste keerulise geomeetriaga andmekogumite puhul toimis PostgreSQL andmebaas paremini, mis sarnanes ka selle töö tulemustega, kus PostgreSQL traditsioonilise disaini mõõtmistulemused olid üldjuhul kõige paremad.

Geoinformaticas 2020. aastal avaldatud uuringus [25, Ch. 6] selgus, et PostgreSQL on peaaegu kõigil juhtudel parem kui MongoDB. Kui vaadata praeguse töö PostgreSQL traditsioonilise disaini ja MongoDB mitte-hierarhilise disaini otsingutega seotud andmebaasioperatsioonide mõõtmistulemusi, siis võib teha *Geoinformatcas* artikliga sama järelduse. 2020. aasta uuringus toodi välja, et PostgreSQLi puhul on indeksite kasutamise positiivne mõju pigem väiksem, kuid praegune uuring näitas, et indeksite kasutamine parandus töökiiruseid üksiku olemi otsimisel märgatavalt.

6.3 Töö tegemise reflektsioon

Valminud töö on kasulik nendele, kes soovivad sarnasel teemal ehk andmebaasides toimuvate andmekäitluse operatsioonide jõudluse uuringutega jätkata ning minna teemas ja tulemuste leidmises veelgi sügavamale. Samuti oleks uuringust kasu tarkvaraarendajatele, kes otsivad rakenduse jaoks parimat andmebaasisüsteemi ning tahavad teada JSON-formaadis dokumentide andmebaasis talletamise mõju andmebaasioperatsioonide jõudlusele.

Töö tegemisel muutus lõpuks mugavaks mõlema andmebaasisüsteemi kasutamine, kuna autor tutvus nendega põhjalikult ning tegi ülesannete koostamise ja testimise käigus mõlema tarkvara kasutamise endale selgeks. Esialgu oli PostgreSQL-is lausete koostamine lihtsam, kuna varasemalt oli Tallinna Tehnikaülikooli kursuses „Andmebaasid I“ tutvutud juba ühe SQL-andmebaasisüsteemiga (MS Access). Hiljem aga muutus otsinguga seotud lausete testimine mugavamaks MongoDB Compass tarkvaras, kuna seal oli otsinguga seotud päringuid lihtsam salvestada ja uuesti käivitada. Autor koostab mõõtmistulemustest enda jaoks ülevaatlikud tabelid, mida oli kerge analüüsida kasutades *Google Sheetsis* olemasolevaid funktsioone.

Töö tegemisel avastas autor, et kõigi kolme andmemahuga testide läbiviimine võtab arvatust rohkem aega ning nende läbiviimisega oleks võinud varem alustada. Kuna MongoDB andmebaasisüsteem polnud autorile töö tegemise alustamisel niivõrd tuttav, siis alguses võttis selle kasutamine veidi rohkem aega. Lisaks soovis autor algselt analüüsida töös kirjutamise skeemi kasutamise ja mitte kasutamise mõju erinevust andmebaasisüsteemides toimuvate andmekäitluse operatsioonide jõudlusele, kuid seda ei jõutud selles töös käsitleda.

Tööd uuesti korrates tuleks kohe algusjärgus võimalikult kiiresti eksperimentide jaoks ülesanded koostada ja need läbi viia, et tulemuste analüüsimiseks jääks piisavalt aega. Samuti tuleks kasutada selliseid andmebaasisüsteeme, mis on autori jaoks rohkem tuttavad.

6.4 Edasised uurimissuunad

Praeguses töös uuriti just PostgreSQL ja MongoDB andmebaasisüsteeme, kuid järgnevate uuringute puhul oleks võimalik kaasata teisi populaarseid SQL-andmebaasisüsteeme nagu MySQL, Oracle ja MS SQL Server või dokumendipõhiseid NoSQL andmebaasisüsteeme nagu Couchbase ja CouchDB. Samuti võib uurida, kas tulemustes oleks erinevusi, kui teha mõõtmisi näiteks viie erineva andmemahuga või kui andmebaasides kirjutamise skeemi ei jõustata.

Lisaks, kuna antud töös uuriti, kuidas indeksite kasutamine mõjutab töökiiruseid PostgreSQL andmebaasisüsteemi erinevate disainide korral, siis sama võib ka uurida MongoDB puhul ning võrrelda indeksite kasutuse mõjusid laiemalt SQL ja NoSQL süsteemide korral. Võimalus oleks ka uurida graafipõhistes andmebaasides toimuvate operatsioonide töökiiruseid ja kuidas need muutuvad andmehulkade suurenemisel ning millised on nende eelised või piirangud võrreldes SQL-andmebaasidega.

7 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli uurida andmebaasis JSON dokumentide kasutamise mõju andmekäitluse operatsioonide jõudlusele kahe 2024. aasta kevade seisuga populaarse andmebaasisüsteemi PostgreSQL-i (versiooni 16.2) ja MongoDB (versiooni 7.0.2) näitel. Selleks kasutati eelneva bakalaureustöö [4, Ch. 1.2] raames valminud viit andmebaasi disaini, milleks olid PostgreSQL puhul traditsiooniline, hierarhiline ja mitte-hierarhiline ning MongoDB korral hierarhiline ja mitte-hierarhiline disain.

Eksperimentide läbiviimiseks loodi kakskümmend ülesannet, kusjuures need olid koostatud nii, et peaaegu kõigi ülesande tüüpide korral üks ülesanne otsib andmeid, mis hierarhilise esituse korral on kõige ülemisel tasemel ja teise ülesande korral ei ole otsitavad andmed kõige ülemisel tasemel. Lisaks loodi ka PostgreSQL JSON dokumente kasutavate disainide korral andmebaasis GIN indeksid ja uuriti PostgreSQL disainides indeksite kasutamise mõju töökiirustele suurima andmemahu korral.

Töö käigus tehti eksperimendid kolme erineva andmemahuga, et uurida, kas andmete hulga ja lausete täitmise aja vahel esineb lineaarne seos.

Eksperimentide tulemustena jõuti järeldustele:

- Järjepidevalt parimad tulemused olid PostgreSQL traditsioonilise disaini korral.
- MongoDB disainid andisid parimaid tulemusi nii kindla olemi kui ka mitme olemi kustutamisel (andmete asetsemisel hierarhias allpool) ning uute olemite registreerimisel ja mitme olemi muutmisel (just mitte-hierarhiline disain). Halvimad töökiirused olid MongoDB disainidel detailandmete väljastamisel ja hierarhiliste andmete otsimisel.
- JSON dokumentide kasutamine tuli kasuks hierarhiliste andmete otsimisel ja detailandmete väljastamisel PostgreSQL hierarhilisel disainil.
- Kõigi disainide korral oli enamuse ülesannete lahendamisel näha kasvavat lineaarset seost andmemahtude ja lausete täitmise aja vahel.

- Indeksite kasutamisel paranes üksikute olemite andmete otsimisel töökiirus mitmekordselt iga PostgreSQL disaini korral.

Lõputöö käigus kirjutatud kood on leitav *GitHubi* pesast [27].

Lõputöö edasiarenduseks võiks kasutada ka teisi SQL ja NoSQL andmebaasisüsteeme, teha mõõtmisi näiteks rohkemate erinevate andmemahtudega või uurida töökiiruseid andmebaasides kirjutamise skeemi kasutamise ning mittekasutamise korral.

Kasutatud materjalid

[1] "PostgreSQL: About," The PostgreSQL Global Development Group, [Online]. Available: https://www.postgresql.org/about/. [Accessed 17.05.2024].

[2] "MongoDB manual," MongoDB, Inc., [Online]. Available: https://www.mongodb.com/docs/manual/. [Accessed 17.05.2024].

[3] „DB-Engines Ranking,“ [Online]. Available: https://db-engines.com/en/ranking. [Accessed 17.05.2024].

[4] R. Roopärg, "Andmebaasis kirjutamise skeemi ja JSON dokumentide kasutamise mõju PostgreSQL ja MongoDB andmebaasirakenduste loomisele," B.S. thesis, Tallinn University of Technology, Tallinn, 2022. [Online]. Available: https://digikogu.taltech.ee/et/Item/94f642d0-d1e4-4860-b230-8ad7f0a015ce. [Accessed 17.05.2024].

[5] „PgAdmin,“ [Online]. Available: https://www.pgadmin.org/docs/pgadmin4/7.8/index.html. [Accessed 17.05.2024].

[6] „MongoDB Compass,“ [Online]. Available: https://www.mongodb.com/docs/compass/current/. [Accessed 17.05.2024].

[7] „Welcome to MongoDB Shell (mongosh),“ [Online]. Available: https://www.mongodb.com/docs/mongodb-shell/. [Accessed 17.05.2024].

[8] ChatGPT. (2024), OpenAI. [Online]. Available: https://chatgpt.com. [Accessed 17.05.2024].

[9] Copilot (2024). [Online]. Available: https://copilot.microsoft.com. [Accessed 17.05.2024].

[10] "Google Docs Editors Help: CORREL," [Online]. Available: https://support.google.com/docs/answer/3093990?hl=en&ref_topic=3105600&sjid=2463300684417871438-EU. [Accessed 19.05.2024].

[11] "Google Docs Editors Help: GEOMEAN," [Online]. Available: https://support.google.com/docs/answer/3094001?hl=en. [Accessed 19.05.2024].

[12] „What is a Document Database?," [Online]. Available: https://www.mongodb.com/resources/basics/databases/document-databases. [Accessed 19.05.2024].

[13] M. D. M. Papiernik, „An Introduction to Document-Oriented Databases," 21. 07. 2021. [Online]. Available: https://www.digitalocean.com/community/conceptual-articles/an-introduction-to-document-oriented-databases. [Accessed 17.05.2024].

[14] D. Crockford, „JSON: The Fat-Free Alternative to XML," 6. 12. 2006. [Online]. Available: https://www.json.org/fatfree.html. [Accessed 17.05.2024].

[15] M. Winand, „What's New in SQL:2016," 15. 06. 2017. [Online]. Available: https://modern-sql.com/blog/2017-06/whats-new-in-sql-2016. [Accessed 17.05.2024].

[16] J. Michels, K. Hare, K. Kulkarni, C. Zuzarte, Z. Hua Liu, B. Hammerschmidt ja F. Zemke, „The New and Improved SQL:2016 Standard," *ACM SIGMOD Record*, kd. 47, nr 2, pp. 51-60, 12. 2018.

[17] „PostgreSQL: Documentation 16: 8.14.4. jsonb Indexing," [Online]. Available: https://www.postgresql.org/docs/current/datatype-json.html. [Accessed 17.05.2024].

[18] „PostgreSQL: Documentation 16: 9.16. JSON Functions and Operators,“ [Online]. Available: https://www.postgresql.org/docs/current/functions-json.html. [Accessed 17.05.2024].

[19] E. Gorantla, „Postgres JSONB Usage and performance analysis,“ Medium, 07. 09. 2021. [Online]. Available: https://medium.com/geekculture/postgres-jsonb-usage-and-performance-analysis-cdbd1242a018. [Accessed 17.05.2024].

[20] „Schema Validation,“ [Online]. Available: https://www.mongodb.com/docs/manual/core/schema-validation/. [Accessed 19.05.2024].

[21] „Schema-on-Read vs Schema-on-Write,“ [Online]. Available: https://luminousmen.com/post/schema-on-read-vs-schema-on-write. [Accessed 19.05.2024].

[22] „JSON performance: PostgreSQL vs MongoDB Comparison,“ Document Database Community, 12. 03. 2024. [Online]. Available: https://documentdatabase.org/blog/json-performance-postgres-vs-mongodb/. [Accessed 19.05.2024].

[23] D. Maksimov, "Performance Comparison of MongoDB and PostgreSQL with JSON types," M.S thesis, Tallinn University of Technology, Tallinn, 2015. [Online]. Available: https://digikogu.taltech.ee/et/Item/8494a95f-ecf6-4566-bc25-09a26f142861. [Accessed 20.05.2024].

[24] S. Schmid, E. Galicz and W. Reinhardt, "Performance investigation of selected SQL and NoSQL databases," AGILE, Lisbon, 2015.

[25] A. Makris, K. Tserpes, G. Spiliopoulos, D. Zissis and D. Anagnostopoulos, "MongoDB Vs PostgreSQL: A comparative study on performance aspects," 05. 06. 2020. [Online]. Available: https://link.springer.com/article/10.1007/s10707-020-00407-w. [Accessed 20.05.2024].

[26] R. Roopärg, "postgresql-and-mongodb-db-application," [Online]. Available: https://github.com/reroop/postgresql-and-mongodb-db-application/tree/main/db/db_scripts. [Accessed 20.05.2024].

[27] "DBMS-performance-testing," [Online]. Available: https://github.com/emilianiilo/DBMS-performance-testing/tree/main. [Accessed 20.05.2024].

[28] S. Turney, "Pearson Correlation Coefficient (r) | Guide & Examples," 10. 02. 2024. [Online]. Available: https://www.scribbr.com/statistics/pearson-correlation-coefficient/. [Accessed 20.05.2024].

[29] „The “right” JSON Date Format,“ [Online]. Available: https://www.w3docs.com/snippets/javascript/the-right-json-date-format.html. [Accessed 20.05.2024].

[30] "PostgreSQL: Documentation 16: 8.14. JSON Types," [Online]. Available: https://www.postgresql.org/docs/current/datatype-json.html#JSON-INDEXING. [Accessed 20.05.2024].

[31] "PostgreSQL Documentation 16: 14.1. Using EXPLAIN," [Online]. Available: https://www.postgresql.org/docs/current/using-explain.html#USING-EXPLAIN-ANALYZE. [Accessed 20.05.2024].

[32] „PostgreSQL Documentation 16: ANALYZE,“ [Online]. Available: https://www.postgresql.org/docs/current/sql-analyze.html. [Accessed 20.05.2024].

[33] C. Gallant, "Geometric Mean vs. Arithmetic Mean: What’s the Difference?," 13. 06 2013. [Online]. Available: https://www.investopedia.com/ask/answers/06/geometricmean.asp. [Accessed 20.05.2024].

[34] K. Rootalu, "Korrelatsioonikordajad," [Online]. Available: https://samm.ut.ee/korrelatsioonikordajad/. [Accessed 20.05.2024].

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

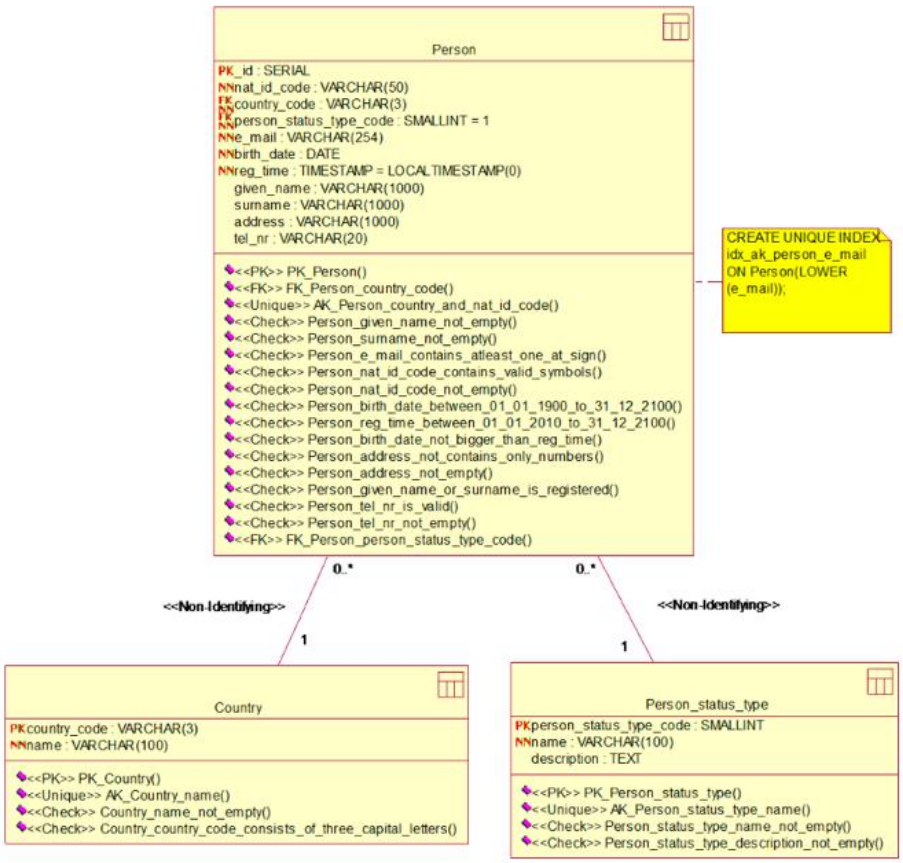
Mina, Emilia Niilo

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Andmebaasis JSON dokumentide kasutamise mõju andmekäitluse operatsioonide jõudlusele PostgreSQL ja MongoDB andmebaasisüsteemide näitel“, 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.

19.05.2024

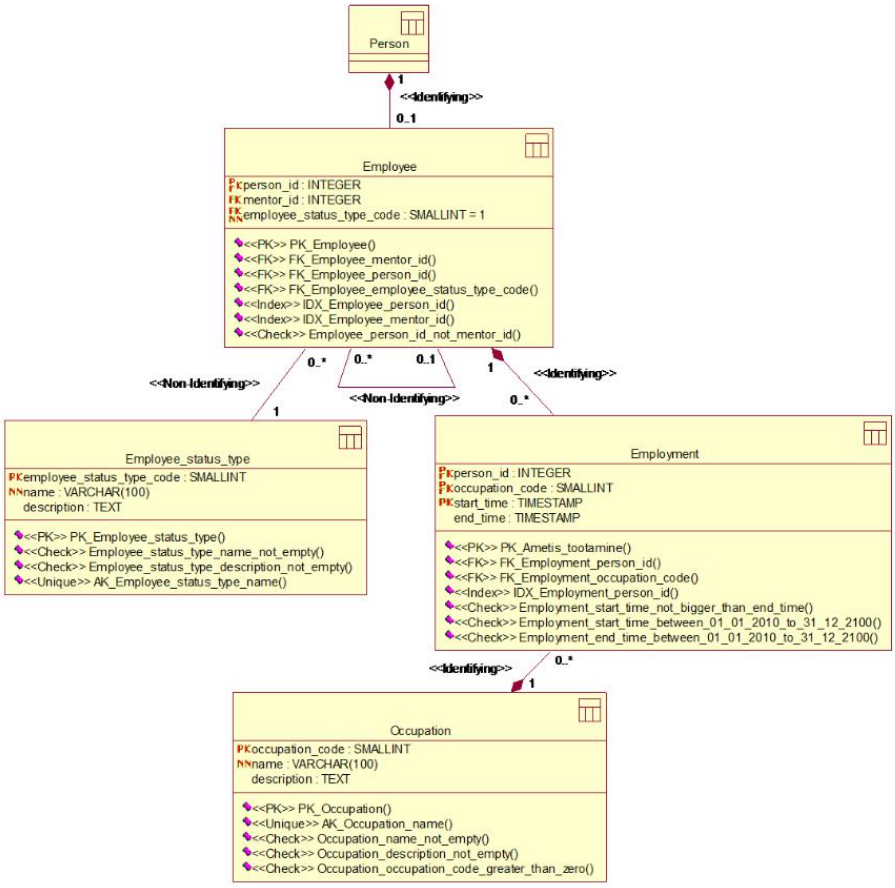
1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või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

LISA 2 - PostgreSQL traditisoonilise skeemi disain



Joonis 7. „Traditsioonilise“ PostgreSQL andmebaasi füüsilise disaini mudel (Isikuga seonduvad tabelid).

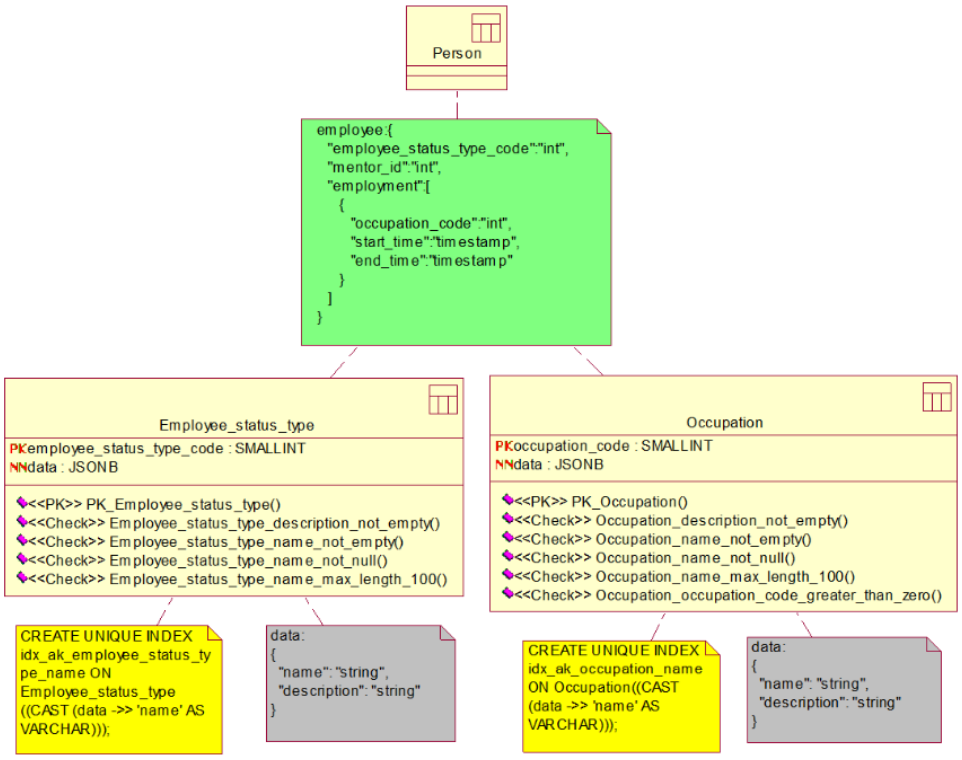
Joonis 12. Roopärgi PostgreSQL T disaini mudel (Isikuga seonduvad tabelid).



Joonis 8. „Traditsioonilise“ PostgreSQL andmebaasi füüsilise disaini mudel (Töötajaga seonduvad tabelid).

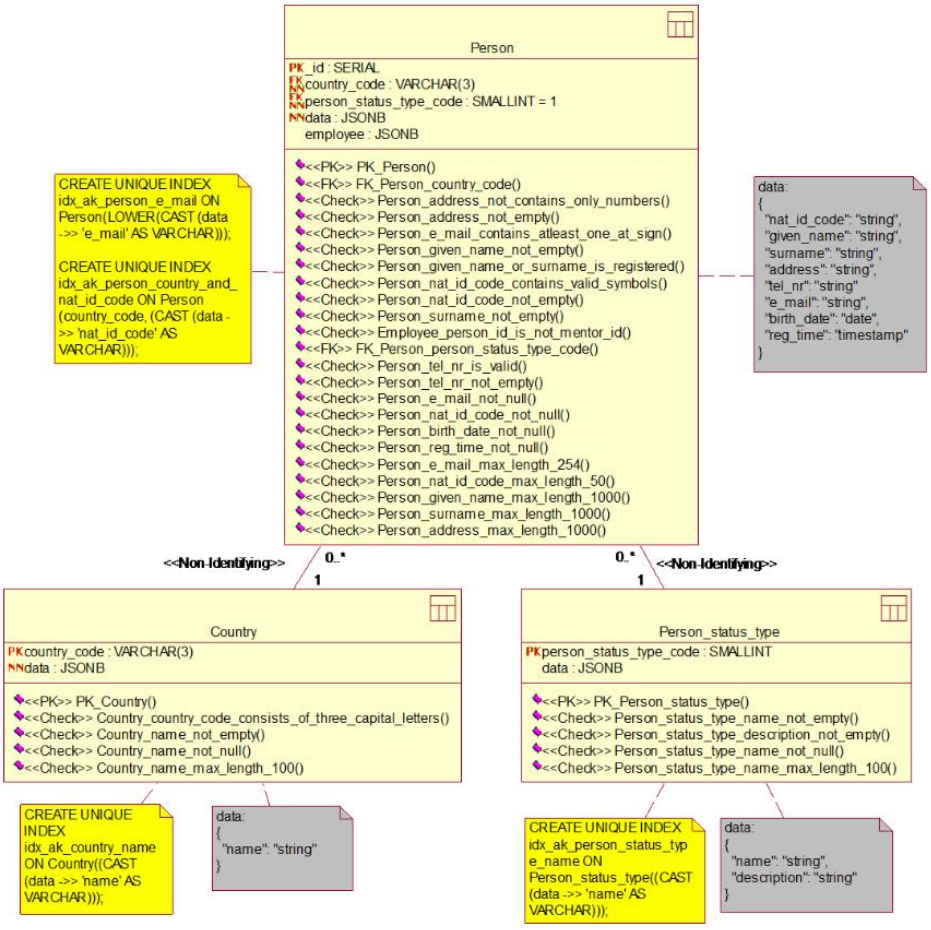
Joonis 13. Roopärgi PostgreSQL T disaini mudel (Töötajaga seonduvad tabelid).

LISA 3 - PostgreSQL hierarhilise skeemi disain



Joonis 10. Hierarhiliste JSON dokumentidega PostgreSQL andmebaasi füüsilise disaini mudel (Töötajaga seonduvad tabelid).

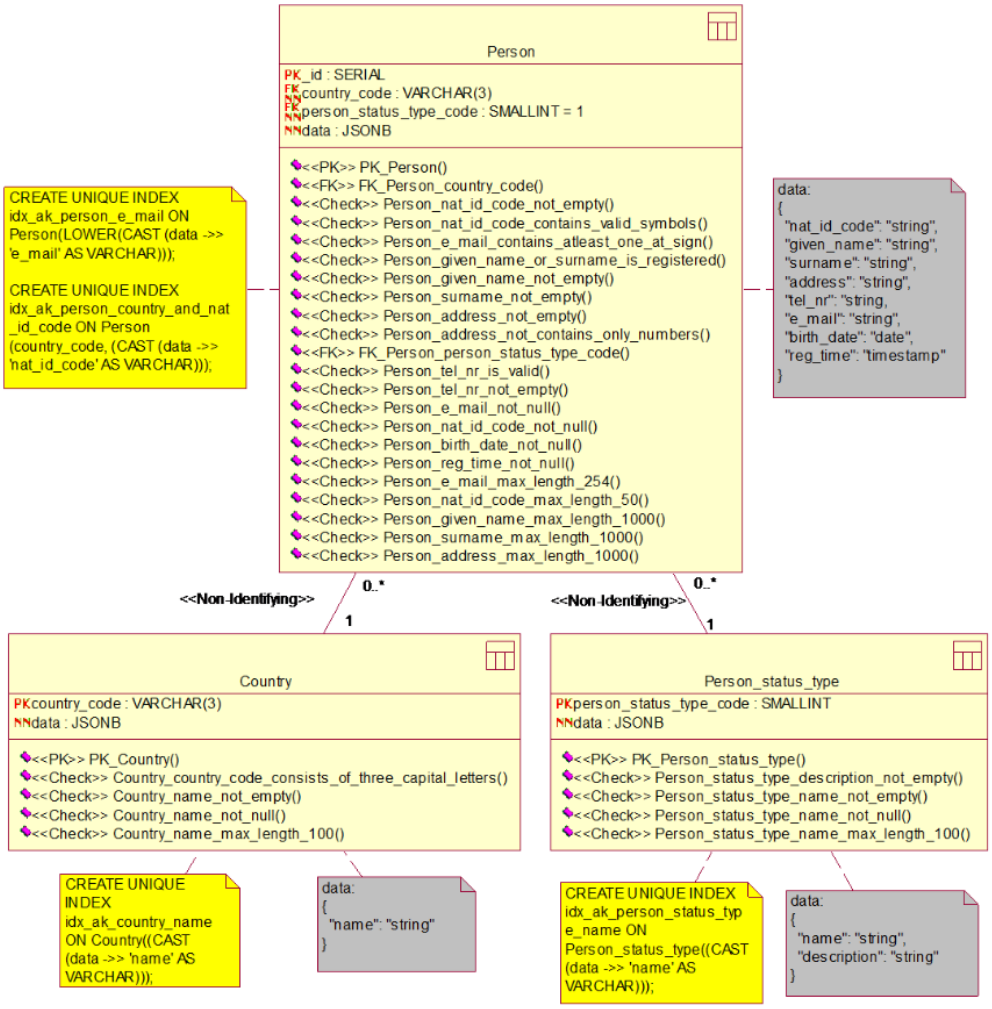
Joonis 14. Roopärgi PostgreSQL H disaini mudel (Töötajaga seonduvad tabelid).



Joonis 9. Hierarhiliste JSON dokumentidega PostgreSQL andmebaasi füüsilise disaini mudel (Isikuga seonduvad tabelid).

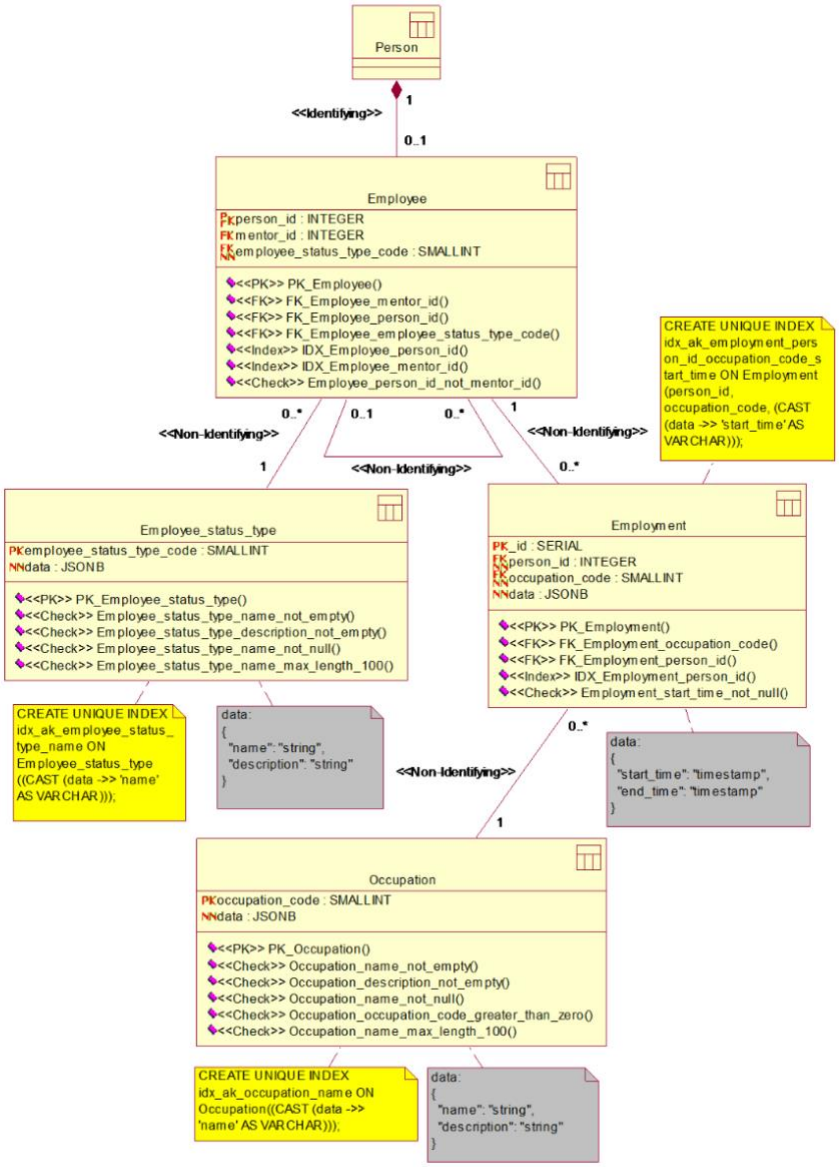
Joonis 15. Roopärgi PostgreSQL H disaini mudel (Isikuga seonduvad tabelid).

LISA 4 - PostgreSQL mitte-hierarhilise skeemi disain



Joonis 11. Mitte-hierarhiliste JSON dokumentidega PostgreSQL andmebaasi füüsilise disaini mudel (Isikuga seotud tabelid).

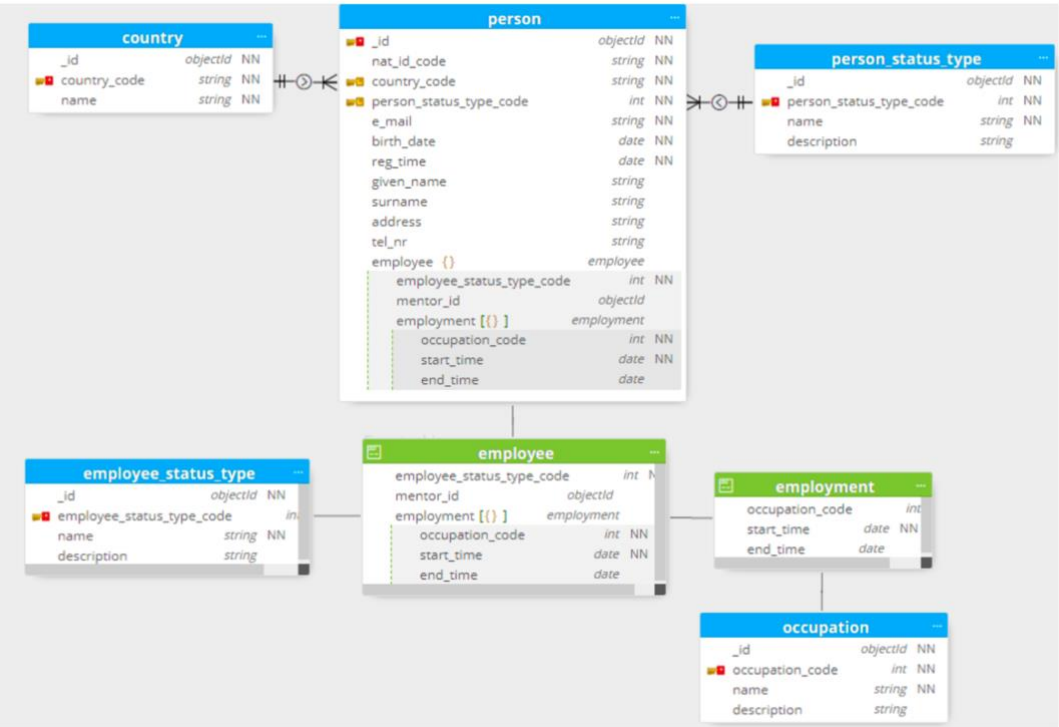
Joonis 16. Roopärgi PostgreSQL M disaini mudel (Isikuga seotud tabelid).



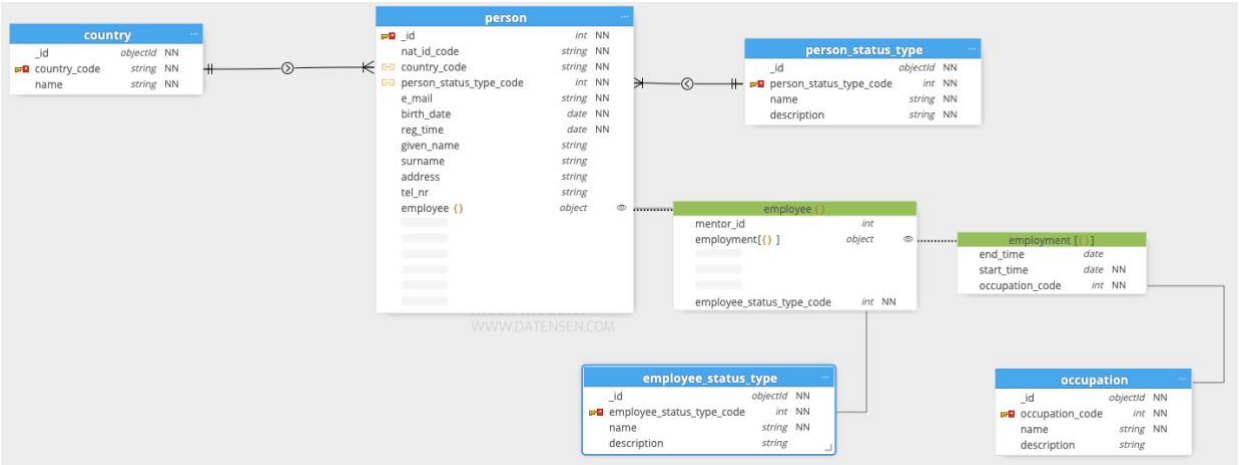
Joonis 12. Mitte-hierarhiliste JSON dokumentidega PostgreSQL andmebaasi füüsilise disaini mudel (Töötajaga seotud tabelid).

Joonis 17. Roopärgi PostgreSQL M disaini mudel (Töötajaga seotud tabelid).

LISA 5 - hierarhilise MongoDB andmebaasi disain

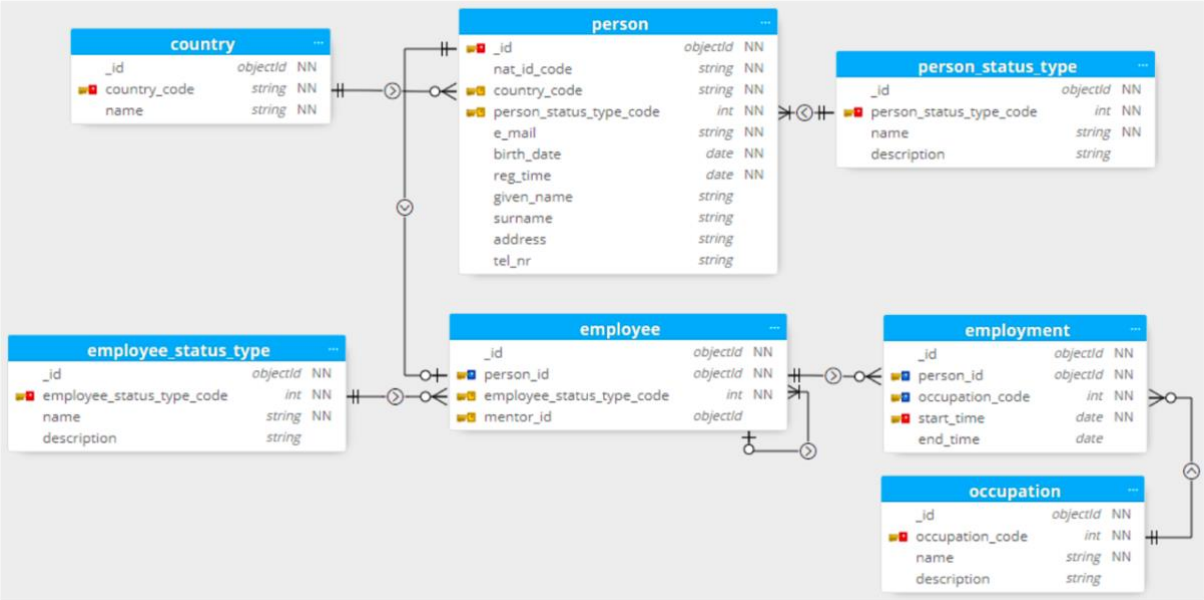


Joonis 18. Roopärgi hierarhiliste JSON dokumentidega MongoDB andmebaasi füüsilise disaini mudel.

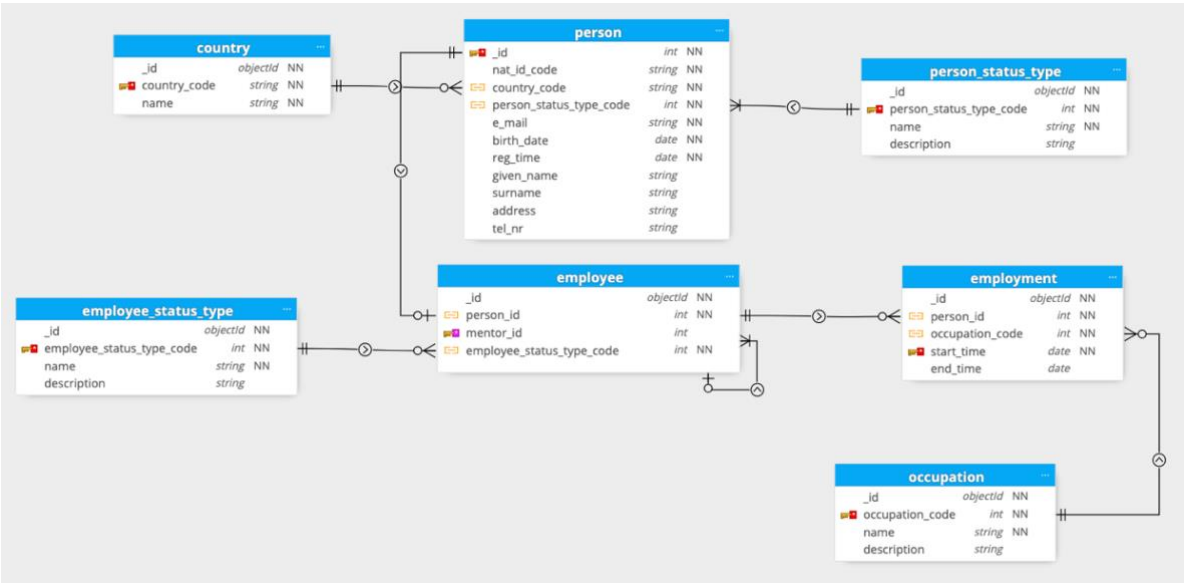


Joonis 19. Muudatustega hierarhiliste JSON dokumentidega MongoDB andmebaasi füüsilise disaini mudel.

LISA 6 - mitte-hierarhilise MongoDB andmebaasi disain



Joonis 20. Roopärgi mitte-hierarhiliste JSON dokumentidega MongoDB andmebaasi füüsilise disaini mudel.



Joonis 21. Muudatustega mitte-hierarhiliste JSON dokumentidega MongoDB andmebaasi füüsilise disaini mudel.

LISA 7 – Eksperimentide mõõtmistulemused andmemaht T1

Tabel 14. Eksperimentide 1_1 kuni 5_1 mõõtmistulemused andmemaht T1.

Eksperiment	Aeg (millis.)	T	H	M	HM	MM
1_1	täitmine	189,556	894,518	399,754		
	planeerimine	0,415	0,335	0,385		
	kokku	**189,556**	**894,853**	**400,139**	**182,833**	**186,641**
1_2	täitmine	591,298	2280,841	689,127		
	planeerimine	0,706	0,411	0,783		
	kokku	**592,004**	**2281,252**	**689,909**	**1419,470**	**5127,646**
Eksperiment	**Aeg (millis.)**	**T**	**H**	**M**	**HM**	**MM**
2_1	täitmine	24,952	118,276	60,923		
	planeerimine	0,398	0,150	0,273		
	kokku	**25,350**	**118,426**	**61,195**	**36,000**	**44,061**
2_2	täitmine	33,990	622,043	45,103		
	planeerimine	11,082	0,446	9,997		
	kokku	**45,072**	**622,488**	**55,100**	**42,664**	**1856,977**
Eksperiment	**Aeg (millis.)**	**T**	**H**	**M**	**HM**	**MM**
3_1	täitmine	126,007	385,142	269,614		
	planeerimine	0,253	2,158	1,726		
	kokku	**126,260**	**387,300**	**271,340**	**114,589**	**8855,775**
3_2	täitmine	426,390	1866,967	1054,017		
	planeerimine	2,124	0,991	1,077		
	kokku	**428,514**	**1867,958**	**1055,094**	**35448,657**	**35355,342**
3_3	täitmine	522,902	1171,128	626,795		
	planeerimine	2,687	0,468	12,628		
	kokku	**525,590**	**1171,595**	**639,423**	**525,275**	**2449,711**
Eksperiment	**Aeg (millis.)**	**T**	**H**	**M**	**HM**	**MM**
4_1	täitmine	902,397	197,027	2202,877		
	planeerimine	5,458	0,626	1,768		
	kokku	**907,855**	**197,653**	**2204,644**	**914,027**	**17280,473**
4_2	täitmine	543,585	864,441	1689,286		
	planeerimine	0,331	0,783	0,881		
	kokku	**543,916**	**865,224**	**1690,168**	**36329,754**	**12528,993**
Eksperiment	**Aeg (millis.)**	**T**	**H**	**M**	**HM**	**MM**
5_1	täitmine	365,520	507,765	910,620		
	planeerimine	1,530	0,879	3,147		
	kokku	**367,050**	**508,643**	**913,767**	**14977,031**	**37138,332**

Tabel 15. Eksperimentide 6_1 kuni 10_2 mõõtmistulemused andmemaht T1.

Eksperiment	Aeg (millis.)	T	H	M	HM	MM
6_1	täitmine	1,763	5,071	6,163		
	planeerimine	1,485	0,030	0,026		
	kokku	**3,247**	**5,101**	**6,189**	**0,005**	**0,007**
6_2	täitmine	4,870	122,509	18,050		
	planeerimine	0,040	1,585	0,095		
	kokku	**4,910**	**124,094**	**18,145**	**153,661**	**0,009**
Eksperiment	**Aeg (millis.)**	**T**	**H**	**M**	**HM**	**MM**
7_1	täitmine	13,371	78,781	56,995		
	planeerimine	1,737	1,825	2,260		
	kokku	**15,107**	**80,607**	**59,255**	**74,137**	**103,310**
7_2	täitmine	42,461	96,746	57,861		
	planeerimine	1,969	2,019	3,755		
	kokku	**44,430**	**98,765**	**61,616**	**123,233**	**0,000**
8_1	täitmine	5079,429	10639,101	7134,833		
	planeerimine	6,949	0,506	13,101		
	kokku	**5086,378**	**10639,608**	**7147,934**	**4472,249**	**3600,664**
8_2	täitmine	1206,962	920,520	1510,696		
	planeerimine	5,347	3,049	18,193		
	kokku	**1212,309**	**923,570**	**1528,889**	**88,257**	**180,476**
Eksperiment	**Aeg (millis.)**	**T**	**H**	**M**	**HM**	**MM**
9_1	täitmine	17,391	76,871	75,031		
	planeerimine	0,436	0,241	0,952		
	kokku	**17,826**	**77,112**	**75,983**	**115,803**	**76,656**
9_2	täitmine	26,990	834,382	71,811		
	planeerimine	0,470	1,647	1,988		
	kokku	**27,460**	**836,030**	**73,800**	**0,000**	**0,000**
10_1	täitmine	4284,390	2047,614	4945,590		
	planeerimine	5,172	2,834	11,628		
	kokku	**4289,562**	**2050,448**	**4957,218**	**2208,318**	**2207,519**
10_2	täitmine	187,745	706,690	270,772		
	planeerimine	3,410	0,698	14,554		
	kokku	**191,155**	**707,388**	**285,326**	**72,742**	**124,186**

LISA 8 – Eksperimentide mõõtmistulemused andmemaht T2

Tabel 16. Eksperimentide 1_1 kuni 5_1 mõõtmistulemused andmemaht T2.

Eksperiment	Aeg (millis.)	T	H	M	HM	MM
1_1	täitmine	294,440	635,641	1070,400		
	planeerimine	0,321	0,555	0,414		
	kokku	**294,760**	**636,196**	**1070,814**	**831,886**	**601,080**
1_2	täitmine	1358,767	3430,443	1841,018		
	planeerimine	0,471	0,888	0,895		
	kokku	**1359,238**	**3431,331**	**1841,912**	**4895,328**	**12733,338**
Eksperiment	**Aeg (millis.)**	**T**	**H**	**M**	**HM**	**MM**
2_1	täitmine	53,402	272,975	51,969		
	planeerimine	1,275	0,254	0,107		
	kokku	**54,677**	**273,230**	**52,075**	**186,035**	**140,255**
2_2	täitmine	46,442	220,175	57,857		
	planeerimine	0,303	1,003	1,681		
	kokku	**46,745**	**221,178**	**59,539**	**199,434**	**4007,786**
Eksperiment	**Aeg (millis.)**	**T**	**H**	**M**	**HM**	**MM**
3_1	täitmine	192,191	741,912	457,366		
	planeerimine	0,750	0,719	1,038		
	kokku	**192,941**	**742,631**	**458,404**	**312,836**	**18076,940**
3_2	täitmine	717,226	1407,872	1946,819		
	planeerimine	0,526	0,624	0,503		
	kokku	**717,752**	**1408,496**	**1947,322**	**77328,283**	**74562,179**
3_3	täitmine	860,002	1446,454	2372,001		
	planeerimine	4,147	0,443	9,486		
	kokku	**864,149**	**1446,897**	**2381,487**	**704,385**	**4565,319**
Eksperiment	**Aeg (millis.)**	**T**	**H**	**M**	**HM**	**MM**
4_1	täitmine	1903,079	452,810	5040,879		
	planeerimine	2,492	0,459	1,980		
	kokku	**1905,571**	**453,269**	**5042,859**	**1632,666**	**34563,770**
4_2	täitmine	1603,682	1569,571	5165,419		
	planeerimine	1,141	0,330	1,412		
	kokku	**1604,823**	**1569,901**	**5166,831**	**6514,593**	**26349,001**
Eksperiment	**Aeg (millis.)**	**T**	**H**	**M**	**HM**	**MM**
5_1	täitmine	1431,361	927,935	3744,099		
	planeerimine	10,227	0,767	14,016		
	kokku	**1441,588**	**928,702**	**3758,114**	**31328,439**	**84159,014**

Tabel 17. Eksperimentide 6_1 kuni 10_2 mõõtmistulemused andmemaht T2.

Eksperiment	Aeg (millis.)	T	H	M	HM	MM
6_1	täitmine	7,131	14,486	17,697		
	planeerimine	1,283	0,477	0,031		
	kokku	**8,414**	**14,962**	**17,728**	**0,011**	**0,007**
6_2	täitmine	10,518	530,392	28,918		
	planeerimine	0,024	0,870	0,046		
	kokku	**10,542**	**531,262**	**28,964**	**262,586**	**0,013**
Eksperiment	**Aeg (millis.)**	**T**	**H**	**M**	**HM**	**MM**
7_1	täitmine	92,711	192,733	195,002		
	planeerimine	0,247	0,140	0,187		
	kokku	**92,958**	**192,873**	**195,189**	**164,549**	**169,634**
7_2	täitmine	41,189	160,732	107,216		
	planeerimine	0,929	0,801	4,112		
	kokku	**42,118**	**161,533**	**111,328**	**243,623**	**0,000**
8_1	täitmine	9266,095	18317,567	12491,473		
	planeerimine	7,278	0,557	9,373		
	kokku	**9273,373**	**18318,124**	**12500,847**	**8222,462**	**7122,292**
8_2	täitmine	1696,618	743,426	2813,514		
	planeerimine	16,707	0,968	7,284		
	kokku	**1713,325**	**744,394**	**2820,798**	**175,668**	**404,175**
Eksperiment	**Aeg (millis.)**	**T**	**H**	**M**	**HM**	**MM**
9_1	täitmine	83,970	368,126	358,479		
	planeerimine	0,157	1,431	0,146		
	kokku	**84,127**	**369,558**	**358,625**	**126,629**	**121,127**
9_2	täitmine	61,264	281,983	239,742		
	planeerimine	0,263	2,261	0,193		
	kokku	**61,527**	**284,245**	**239,934**	**0,000**	**0,000**
10_1	täitmine	8295,863	2699,637	11718,188		
	planeerimine	13,473	2,353	23,861		
	kokku	**8309,336**	**2701,990**	**11742,050**	**4412,802**	**4831,294**
10_2	täitmine	431,056	464,655	780,926		
	planeerimine	14,468	3,098	24,097		
	kokku	**445,524**	**467,754**	**805,024**	**214,842**	**195,432**

LISA 9 – Pearsoni korrelatsioonikordaja viie erineva skeemi korral

Tabel 18. PostgreSQL traditsioonilise skeemi eksperimentide tulemused.

	T	Ridade arv			
	Ülesanne	100000	200000	400000	Pearson
Aeg (millis.)	1_1	189,556	294,760	562,248	0,998423
	1_2	592,004	1359,238	1978,637	0,968515
	2_1	25,350	54,677	252,810	0,977044
	2_2	45,072	46,745	474,353	0,946013
	3_1	126,260	192,941	506,771	0,985813
	3_2	428,514	717,752	1419,720	0,998951
	3_3	525,590	864,149	1537,027	0,999999
	4_1	907,855	1905,571	4962,026	0,995485
	4_2	543,916	1604,823	3728,429	1,000000
	5_1	367,050	1441,588	1447,346	0,758951
	6_1	3,247	8,414	12,234	0,962054
	6_2	4,910	10,542	8,571	**0,483798**
	7_1	15,107	92,958	151,140	0,962855
	7_2	44,430	42,118	146,895	0,938412
	8_1	5086,378	9273,373	18551,229	0,999693
	8_2	1212,309	1713,325	3884,563	0,987827
	9_1	17,826	84,127	158,874	0,987924
	9_2	27,460	61,527	122,421	0,999596
	10_1	4289,562	8309,336	18257,389	0,998741
	10_2	191,155	445,524	963,456	0,999990

Tabel 19. Postgre SQL hierarhilise skeemi eksperimentide tulemused.

	H	Ridade arv			
	Ülesanne	100000	200000	400000	Pearson
Aeg (millis.)	1_1	894,853	636,196	2692,792	0,900802
	1_2	2281,252	3431,331	10289,133	0,980013
	2_1	118,426	273,230	2811,228	0,960429
	2_2	622,488	221,178	1197,399	0,728435
	3_1	387,300	742,631	1373,722	0,999547
	3_2	1867,958	1408,496	7647,015	0,921218
	3_3	1171,595	1446,897	2586,912	0,988921
	4_1	197,653	453,269	842,651	0,997484
	4_2	865,224	1569,901	3259,954	0,999071
	5_1	508,643	928,702	1649,883	0,999242
	6_1	3,247	14,962	27,754	0,986459
	6_2	4,910	531,262	1099,744	0,985936
	7_1	15,107	192,873	439,116	0,995284
	7_2	44,430	161,533	349,209	0,998363
	8_1	5086,378	18318,124	35898,352	0,994085
	8_2	1212,309	744,394	2405,510	0,819573
	9_1	17,826	369,558	2459,545	0,980098
	9_2	27,460	284,245	913,122	0,998856
	10_1	4289,562	2701,990	7021,292	0,761410
	10_2	191,155	467,754	1398,203	0,993624

Tabel 20. PostgreSQL mitte-hierarhilise skeemi eksperimentide tulemused.

	M	Ridade arv			
	Ülesanne	100000	200000	400000	Pearson
Aeg (millis.)	1_1	400,139	1070,814	1409,112	0,929378
	1_2	689,909	1841,912	3671,511	0,998221
	2_1	61,195	52,075	1327,043	0,942858
	2_2	55,100	59,539	838,812	0,946510
	3_1	271,340	458,404	1228,699	0,989055
	3_2	1055,094	1947,322	4261,102	0,998171
	3_3	639,423	2381,487	2201,733	0,691185
	4_1	2204,644	5042,859	2605,133	**−0,059363**
	4_2	1690,168	5166,831	11969,022	0,999985
	5_1	913,767	3758,114	4405,250	0,858414
	6_1	6,189	17,728	14,911	0,581613
	6_2	18,145	28,964	21,295	**0,096677**
	7_1	59,255	195,189	286,845	0,954749
	7_2	61,616	111,328	208,177	0,999979
	8_1	7147,934	12500,847	27677,089	0,996848
	8_2	1528,889	2820,798	4764,704	0,997240
	9_1	75,983	358,625	247,049	**0,438922**
	9_2	73,800	239,934	208,551	0,627505
	10_1	4957,218	11742,050	24142,122	0,999742
	10_2	285,326	805,024	1123,800	0,946817

Tabel 21. MongoDB hierarhilise skeemi eksperimentide tulemused.

	HM	Ridade arv			
	Ülesanne	100000	200000	400000	Pearson
Aeg (millis.)	1_1	182,833	831,886	959,664	0,847394
	1_2	1419,470	4895,328	6683,766	0,931195
	2_1	36,000	186,035	227,525	0,874519
	2_2	42,664	199,434	519,069	0,999989
	3_1	114,589	312,836	560,162	0,991995
	3_2	35448,657	77328,283	180264,846	0,998817
	3_3	525,275	704,385	1946,427	0,976438
	4_1	914,027	1632,666	4239,762	0,991988
	4_2	36329,754	6514,593	203838,079	0,889710
	5_1	14977,031	31328,439	85199,586	0,994056
	6_1	0,005	0,011	0,007	0,173137
	6_2	153,661	262,586	814,111	0,984014
	7_1	74,137	164,549	814,111	0,975624
	7_2	123,233	243,623	429,538	0,997740
	8_1	4472,249	8222,462	20089,547	0,994864
	8_2	88,257	175,668	1008,257	0,969577
	9_1	115,803	126,629	332,625	0,958486
	9_2	0,000	0,000	0,002	0,944911
	10_1	2208,318	4412,802	11488,869	0,994582
	10_2	72,742	214,842	418,526	0,996160

Tabel 22. MongoDB mitte-hierarhilise skeemi eksperimentide tulemused.

	MM	Ridade arv			
	Ülesanne	100000	200000	400000	Pearson
Aeg (millis.)	1_1	186,641	601,080	903,293	0,960975
	1_2	5127,646	12733,338	19904,966	0,978635
	2_1	44,061	140,255	176,686	0,902763
	2_2	1856,977	4007,786	7557,985	0,998783
	3_1	8855,775	18076,940	35349,436	0,999865
	3_2	35355,342	74562,179	173473,610	0,998518
	3_3	2449,711	4565,319	9539,118	0,999256
	4_1	17280,473	34563,770	91355,413	0,994110
	4_2	12528,993	26349,001	67128,247	0,996167
	5_1	37138,332	84159,014	195003,357	0,999232
	6_1	0,007	0,007	0,257	0,945630
	6_2	0,009	0,013	0,141	0,954494
	7_1	103,310	169,634	780,712	0,970246
	7_2	0,000	0,000	0,004	0,944911
	8_1	3600,664	7122,292	15767,631	0,998830
	8_2	180,476	404,175	1221,374	0,991833
	9_1	76,656	121,127	298,027	0,989874
	9_2	0,000	0,000	0,084	0,944911
	10_1	2207,519	4831,294	9867,025	0,999947
	10_2	124,186	195,432	604,396	0,980937

LISA 10 – GIN indeksid ja muudetud ülesannete kood

```
CREATE INDEX idx_data_gin ON hierarchical.person USING
gin(data jsonb_path_ops);

CREATE INDEX idx_data_email_gin ON hierarchical.person
USING GIN ((data -> 'e_mail'));

CREATE INDEX idx_employee_gin ON hierarchical.person USING
gin(employee jsonb_path_ops);

CREATE INDEX idx_employee_occupation_start_end_gin ON
hierarchical.person USING GIN ((employee -> 'employment'));

CREATE INDEX idx_data_gin ON non_hierarchical.person USING
GIN(data jsonb_path_ops);

CREATE INDEX idx_person_email_gin ON
non_hierarchical.person USING GIN ((data -> 'e_mail'));

CREATE INDEX idx_employment_gin ON
non_hierarchical.employment USING GIN(data jsonb_path_ops);

CREATE INDEX idx_employment_occupation_start_end_gin ON
non_hierarchical.employment USING GIN ((data ->>
'start_time'), (data ->>

'end_time'));

EXPLAIN ANALYZE SELECT _id, (data ->> 'surname') AS surname
FROM hierarchical.person WHERE data @> '{"e_mail":
"acscu_ed69216fed359e36bd52421c86d40902@example.com"}';

EXPLAIN ANALYZE SELECT employee.value ->> 'end_time' AS
end_time FROM hierarchical.person p,
jsonb_array_elements(p.employee -> 'employment') employee
```

```
WHERE p.data @> '{"e_mail":
"bbsyi_691b8036185a3cef186bfadf34d5f14b@example.com"}' AND
employee.value @> '{"occupation_code": 4, "start_time":
"2023-05-06T14:33:56.363994"}';

EXPLAIN ANALYZE SELECT _id, (data ->> 'surname') AS surname
FROM non_hierarchical.person WHERE data @> '{"e_mail":
"acscu_ed69216fed359e36bd52421c86d40902@example.com"}';

EXPLAIN ANALYZE SELECT e.data ->> 'end_time' AS end_time
FROM non_hierarchical.employment e JOIN
non_hierarchical.person p ON e.person_id = p._id WHERE
p.data @> '{"e_mail":
"bbsyi_691b8036185a3cef186bfadf34d5f14b@example.com"}' AND
e.occupation_code = 4 AND e.data @> '{"start_time": "2023-
05-06T14:33:56.363994"}';
```