

Kian Moloney

NEO4J NYKYAIKAISENA TIETOKANTARATKAISUNA

Kandidaatintyö
Informaatioteknologian ja viestinnän tiedekunta
Tarkastajat: Mikko Nurminen
Toukokuu 2024

TIIVISTELMÄ

Kian Moloney: Neo4j nykyaikaisena tietokantaratkaisuna
Kandidaatintyö
Tampereen yliopisto
Tieto- ja sähkötekniikan kandidaatin tutkinto-ohjelma, tietotekniikka
Toukokuu 2024

Tiedon määrä maailmassa kasvaa jatkuvasti räjähdysmäisellä tahdilla. Datantuottajat, kuten sosiaalinen media ja IoT (*Internet of Things*), generoivat päivittäin suuria määriä vahvasti kytkeytynyttä dataa ja kaikelle tälle datalle tarvitaan tehokas ja johdonmukainen tallennuspaikka. Strukturoimattoman datan tallennuspaikkana suosittu vaihtoehto ovat pitkään olleet NoSQL (*Not Only SQL*) -tietokannat ja vahvasti kytkeytyneelle datalle erityisesti graafitietokannat kuten Neo4j.

Tämä tutkielma on toteutettu kirjallisuuskatsauksena, jonka tavoitteena on selvittää, minkälaisia ominaisuuksia graafitietokanta Neo4j:llä on liittyen suurien datamäärien käsittelyyn ja manipulointiin. Tutkielmassa käytetyt lähteet ovat tieteellisiä artikkeleita, konferenssijulkaisuja ja kirjoja. Tutkielmassa käsitellään ensin NoSQL- ja graafitietokantojen ominaisuuksia. Neo4j:n ominaisuuksiin ja piirteisiin pureudutaan tarkemmin tämän jälkeen. Lisäksi tutustutaan Neo4j:n tarjoaman Graph Data Science -kirjaston toimintaan sekä mahdollisuuksiin ja Neo4j:n suorituskykyyn suurilla datamäärillä verrattuna muihin tietokantoihin.

Kirjallisuuskatsauksen tuloksina havaittiin että Neo4j:n pohjalla oleva graafimalli on tehokas ja optimoitu vahvasti kytkeytyneelle kompleksille datalle. Lähteiden perusteella havaittiin myös että Neo4j suoriutui sekä relaatio- että graafitietokantoihin verrattuna mainiosti ja suurimmat erot näkyivät algoritmien suoritusajoissa. Lisäksi havaittiin että Neo4j:n oma kyselykieli Cypher on helpposti lähestyttävä ja ominaisuuksiltaan monipuolinen. Lähteiden perusteella havaittiin että vaikka Neo4j:n Graph Data Science -kirjasto on kehittyvässä vaiheessa, on se suorituskyvyltään lähellä samaa tasoa kuin vertailussa mukana ollut imperatiivinen lähestymistapa Networkx, ja pitää sisällään uniikkeja ominaisuuksia, kuten pysyvät näkymät. Havaittiin myös, että Neo4j:n tarjoaman koneoppimisputken konseptin avulla käyttäjä pystyy opettamaan omaan käyttökohteeseensa sopivan koneoppimismallin ja manipuloida dataa. Neo4j tarjoaa alustan kompleksisten verkostojen analysointiin ja koneoppimismallien integrointiin graafitietokantaan, mikä tekee siitä pätevän valinnan vahvasti kytkeytyneen datan käsittelyyn ja datan syällisen ymmärtämisen edistämiseen.

Avainsanat: Neo4j, graafitietokanta, NoSQL, Big Data, koneoppiminen

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

SISÄLLYSLUETTELO

1.	Johdanto	1
2.	Tietokannat	3
2.1	NoSQL-tietokannat	3
2.2	Graafitietokannat	5
3.	Neo4j	7
4.	Neo4j datan käsittelyn työkaluna	9
4.1	Graph Data Science -kirjasto	9
4.2	Big Data	11
5.	Pohdinta	16
6.	Yhteenveto	18
	Lähteet	19

1. JOHDANTO

Teknologisen kehityksen myötä tiedon määrä maailmassa kasvaa räjähdysmäisesti, ja tämän datan käsittelyyn ja varastointiin tarvitaan tehokkaita ratkaisuja. Datantuottajat, kuten sosiaalinen media ja IoT (*Internet of Things*), generoivat päivittäin massiivisia määriä dataa, jota on tärkeää käsitellä reaaliajassa. Kaikelle tälle datalle tarvitaan säilytyspaikka ja erilaisten tietokantaratkaisujen kehitys on vastannut tähän tarpeeseen monipuolisesti.

Perinteisempi tietokantamalli on nimeltään relaatiomalli. Relaatiomalli on 1970-luvulla kehitetty malli, jota suosituimmat tietokannat eli relaatiotietokannat noudattavat. Relaatiotietokantojen rinnalle on noussut joukko uusia vaihtoehtoja, jotka tarjoavat erilaisia lähestymistapoja datan tallentamiseen ja hallintaan. Vuonna 2009 alettiin kehittämään NoSQL-tietokantoja (*Not Only SQL*), sillä relaatiotietokannat eivät olleet tarpeeksi tehokas ratkaisu moderneille web-pohjaisille sovelluksille. (J.-K. Chen & Lee, 2019) NoSQL-tietokannat ovat kasvaneet suosiossa 15 viime vuoden aikana, ja nyt kymmenen suosituimman tietokannan hallintajärjestelmän joukossa on jopa kolme NoSQL-tietokantaa (DB-Engines, 2024).

Graafitietokannat ovat graafiteoriaan perustuvia tietyn tyyppisiä NoSQL-tietokantoja. Avoimeen lähdekoodiin perustuva Neo4j on kaikista graafitietokannoista suosituin (DB-Engines, 2024). Se julkaistiin ensimmäisen kerran 2007 ja nykyään palveluntarjoajat kuten eBay, Verizon ja AT&T varautuvat Neo4j:hin. Näiden lisäksi jopa 75 % Fortune 100:n listatuista yrityksistä käyttävät Neo4j:n palveluja. (Neo4j Inc., 2024e)

Tässä tutkielmassa perehdytään Neo4j:n ominaisuuksiin ongelmanratkaisun näkökulmasta. Tavoitteena on selvittää, miten Neo4j suoriutuu suurilla datamäärillä sekä millaisia palveluja ja työkaluja se tarjoaa datan käsittelyä varten, eritoten koneoppimisen ja Big Datan näkökulmasta. Neo4j:tä vertaillaan muihin relaatio- ja graafitietokantoihin sen suorituskyvyn perusteella.

Tutkielma on toteutettu kirjallisuuskatsauksena. Lähteinä on käytetty aiheeseen liittyvää tietotekstiä, kuten konferenssijulkaisuja, artikkeleita ja kirjoja. Lähteitä on haettu Tampereen yliopiston Andor-palvelusta, Google Scholar-palvelusta ja Association for Computing Machineryn tietokannasta. Lähteitä hakiessa käytettiin seuraavia hakusanoja "*Neo4j*", "*graph databases*", "*NoSQL databases*", "*graph data science*" ja "*Cypher*". Näitä hakusanoja yhdisteltiin AND- ja OR-operaattoreita käyttäen.

Ensimmäinen luku on johdanto aiheeseen, jossa käydään tutkielman motiiveja ja taustaa läpi. Toisessa luvussa esitellään tarkemmin NoSQL-tietokantoja ja graafitietokantoja. Kolmannessa luvussa tutustutaan Neo4j:n toimintaan ja palveluihin. Neljännessä luvussa tarkastellaan Neo4j:n Graph Data Science -kirjaston keskeisiä toimintoja ja suorituskykyä, sekä Neo4j:n yleistä suoriutumista suurilla vahvasti kytkeytyneillä datamäärillä vertaillaessa relaatio- ja graafitietokantoihin.

2. TIETOKANNAT

Tietokannat jaetaan korkealla tasolla relaatio- ja NoSQL-tietokantoihin. Relaatiotietokanta (engl. *relational database*, RDB) on kokoelma relaatioita, jotka noudattavat tietorakenteeltaan relaatiomallia. Relaatiomalli (engl. *Relational Database Model*, RDM) on 1970-luvulla kehitetty malli, jonka mukaan relaatio on kaksiulotteinen normalisoitu taulukko. Jokainen relaatio koostuu kahdesta osasta, relaatioskeemasta ja relaatioinstansseista. Relaatioskeemaan kuuluu muun muassa relaation nimi, relaation osapuolten nimet ja toimialueet (engl. *domains*). Relaatioinstanssi viittaa relaatioissa tallennettuun tietoon tietyllä ajanhetkellä. Relaatiotietokannat useimmiten käyttävät kyselykielenään SQL:ää (*Structured Query Language*). (J.-K. Chen & Lee, 2019)

Relaatiotietokannat perustavat toimintansa ACID-periaatteisiin. ACID tulee sanoista *Atomicity*, *Consistency*, *Isolation* ja *Durability*. Atomisuus (*Atomicity*) takaa että operaatiot suoritetaan loppuun, johdonmukaisuus (*Consistency*) takaa tietokannassa olevan datan stabilisuuden, eristäminen (*Isolation*) takaa useiden samanaikaisesti suoritettavien transaktioiden riippumattomuuden ja kestävyys (*Durability*) huolehtii siitä, että tallennetut transaktiot eivät muuta tilaansa edes vikaantumistilanteessa. (J.-K. Chen & Lee, 2019)

Relaatiotietokannan koon kasvaessa, tulee ongelmaksi kuitenkin sen huono skaalautuvuus ja tehokkuus suorittaessa liitosoperaatioita suurille taulukoille. Tästä syystä NoSQL-tietokannat ovat usein ratkaisu suurien datamäärien hallintaan. (Kunda & Phiri, 2017)

2.1 NoSQL-tietokannat

NoSQL-tietokantojen kehitys alkoi vuonna 2009. Alkuperäinen syy niiden kehitykselle oli tarve moderneille tietokannoille web-pohjaisille sovelluksille. Sitten kehitys on kiihtynyt kovaa tahtia ja ominaisuuksiensa vuoksi yhä useammat valitsevat perinteisemmän relaatiotietokannan sijasta NoSQL-tietokannan. (J.-K. Chen & Lee, 2019)

Relaatiotietokannoista poiketen NoSQL-tietokannat ovat skeemattomia. Skeemattomuus tarkoittaa sitä, että tietokantaan kirjoittaessa ei ole tarpeellista välittää ennalta määritellystä skeemasta, vaan sen sijaan tietokanta pystyy käsittelemään joustavammin erilaista dataa. Lisäksi NoSQL-tietokannat ovat hajautettuja, horisontaalisesti skaalautuvia, sekä pohjautuvat usein avoimeen lähdekoodiin. Hajautetun NoSQL-tietokannan data on usein tallennettu usealle eri palvelimelle, jolloin datan sijainnin määrittää sen metadata. (J.-K. Chen &

Lee, 2019) Horisontaalisesti skaalautuva NoSQL-tietokanta pystyy lisäämään ja vähentämään käyttämiään palvelimia vastatakseen datan kysynnän määrään eli se voi teoriassa skaalautua äärettömiin, toisin kuin vertikaalisesti skaalautuva relaatiotietokanta. ACID-periaatteiden sijaan ne noudattavat usein BASE-periaatteita (engl. *Basically Available, Soft-state, Eventually consistent*). (Kunda & Phiri, 2017)

Avataan BASE-periaatteita seuraavaksi. Käytännössä saatavilla (*Basically Available*) tarkoittaa sitä, että tietokanta pystyy tarjoamaan palveluja ja suorittamaan kyselyitä riippumatta tietokannan varsinaisesta tilasta eli osa tietokannasta voi kaatua ilman, että kysely epäonnistuu kokonaan. Tämän mahdollistaa se, että tietokanta pitää useita kopioita datasta eri palvelimilla, jolloin yksi kaatunut palvelin ei vaikuta tietokannan toimintaan häiritsevällä tavalla. Pehmeä tila (*Soft-state*) tarkoittaa sitä, että datan vahva johdonmukaisuus ei ole vaadittua. Vahva johdonmukaisuus tarkoittaa sitä, että jos jotain tiettyä kopiota datasta päivitetään, tulisi kaikilla seuraavilla samaan dataan kohdistuvilla kyselyillä olla uusin data hallussaan. (J.-K. Chen & Lee, 2019) Tämä mahdollistaa sen, että esimerkiksi lyhytaikaisille käyttäjäsessioille pehmeä tila voi olla tehokkaampi, sillä käyttäjäsessioon liittyvää dataa ei välttämättä ole jaettu kaikkien palvelimien kesken (Chandra, 2015). Aikanaan johdonmukainen (*Eventually consistent*) tarkoittaa sitä, että tietokannan osan tulee olla johdonmukainen uusimman päivityksen suhteen tietyn ajan jälkeen eli toisaalta epäjohdonmukaisuus on sallittua (J.-K. Chen & Lee, 2019). Tietokannan rakentaminen relaatiotietokantojen ACID-periaatteiden mukaan voi olla hankalaa, joten usein johdonmukaisuuden ja eristämisen periaatteista luovutaan. Tästä seuraa BASE-periaatteita mukaileva lähestymistapa. (Chandra, 2015)

NoSQL-tietokantoja on 14 eri tyyppiä (HostingData, 2024). Näistä kuitenkin neljä merkittävintä ovat avain-arvo-pareihin, sarakkeisiin, dokumentteihin ja graafeihin tiedon tallentavan perustavat tietokannat. Avain-arvo-parissa avain on tietokannassa oleva alkio, joka on attribuuttinsa mukaan yksilöity ja arvo puolestaan on attribuutin varsinaisen arvo. Avain-arvo-tietokantaa on helppo käyttää, mutta parien väliset suhteet eivät ole mahdollisia. Sarakkeisiin perustuvassa NoSQL-tietokannassa on relaatiotietokannan tapaan rivejä ja sarakkeita, mutta rivien sijaan sarakkeet ovat omissa tiedostoissaan eli tietokannassa avain viittaa sarakkeeseen. Sarakkeisiin perustuva tietokanta ei ole joustava verrattuna muihin NoSQL-tietokantoihin, mutta sen suorituskyky on hyvä. Dokumenttiorientoitunut NoSQL-tietokanta käyttää tiedon tallentamiseen dokumentteja, jotka ovat kokoelma avain-arvo-pareja. Joukko dokumentteja on nimeltään kokoelma. Kokoelmassa voi olla mikä tahansa määrä dokumentteja riippumatta datan tyypistä eli skeemattomuuden lisäksi on helppoa tallentaa tietyn tyypistä dataa kokoelmittain. Tietokannassa avain viittaa kokonaiseen dokumenttiin mahdollistaen monimutkaisia tietorakenteita, mutta ongelmana on jälleen suhteiden käsittely eri dokumenttien välillä. Graafitietokannat esittävät datan graafina ja perustuvat graafiteoriaan. Graafitietokannan skaalautuvuus ei ole kaikista parhain, mutta se tarjoaa vahvasti kytkeytyneelle datalle tehokkaan tallennuspai-

kan. (Chandra, 2015; Kunda & Phiri, 2017) Chandra (2015) tutkimuksessa esitetyt neljän edellämainitun NoSQL-tietokantatyypin ominaisuuksia on esitetty taulukossa 2.1.

Datamalli	Suorituskyky	Skaalautuvuus	Joustavuus	Kompleksisuus	Funktionaalisuus
Avain-arvo-pari	Korkea	Korkea	Korkea	Matala	Muuttuva
Sarake	Korkea	Korkea	Kohtuullinen	Matala	Minimaalinen
Dokumentti	Korkea	Muuttuva	Korkea	Matala	Muuttuva
Graafi	Muuttuva	Muuttuva	Korkea	Korkea	Graafiteoriaan perustuva

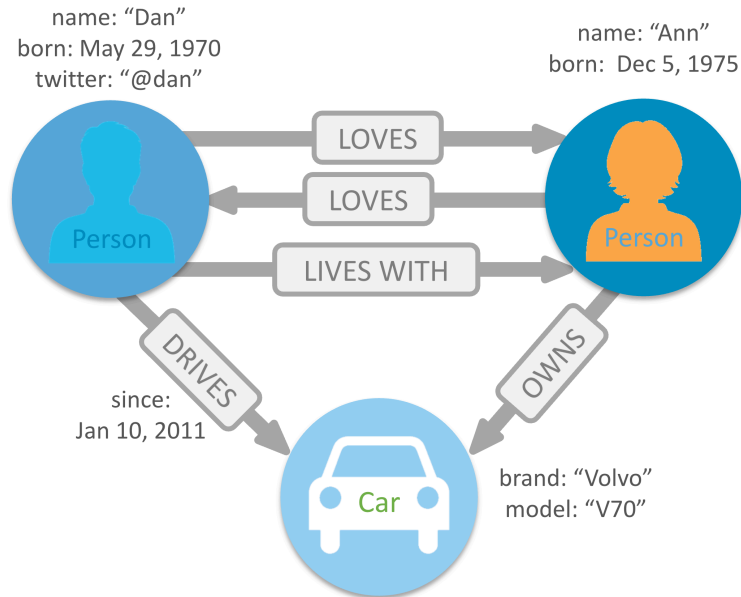
Taulukko 2.1. Eri datamalliin perustuvien NoSQL-tietokantojen ominaisuuksia (Chandra, 2015)

2.2 Graafitietokannat

Graafitietokanta on tietyn tyyppinen NoSQL-tietokanta ja pohjimmiltaan mikä tahansa tiedon varastointijärjestelmä, joka käyttää graafeja tietorakenteenaan. Suosituin graafimalli graafitietokantojen parissa on ominaisuusgraafimalli (engl. *property graph model*) (Pokorný, 2015). Kyseisessä mallissa on kytkeytyneitä solmuja ja kaaria, joita kutsutaan kyseisen mallin kohdalla suhteiksi (engl. *relationship*) (Neo4j Inc., 2024a). Solmut kuvaavat entiteettejä kuten esimerkiksi ihmisiä ja pankkitilejä (Francis ym., 2018). Ominaisuusgraafimallissa solmuja yhdistävät suunnatut suhteet, joilla on aina aloitus- ja lopetussolmu. Koska suhteiden tallennus on tehokasta, voi kahden eri solmun välillä olla mikä tahansa määrä suhteita tehokkuudesta tinkimättä. Solmuille ja suhteille voidaan määritellä ominaisuuksia, jotka koostuvat avain-arvo-pareista. Ominaisuuksien ansiosta solmut ja suhteet ovat käytännöllisempiä ja kuvauksellisempia, jolloin kompleksinkin datan mallinnus ja kysely helpottuu. (Francis ym., 2018; Pokorný, 2015) Kuvassa 2.1 on visualisoitu graafimallin mukaista dataa.

On tärkeää erottaa graafitietokanta käsitteenä graafitietokannan hallintajärjestelmästä (engl. *Graph Database Management System, GDBMS*). Esimerkiksi Neo4j on graafitietokannan hallintajärjestelmä, sillä se tarjoaa graafipohjaisen tietokantaratkaisun lisäksi palveluja ja työkaluja, kuten oman kyselykielen (engl. *query language*), sekä mahdollistaa käyttäjälle luonti-, luku-, päivitys-, sekä poisto-operaatiot (engl. *Create, Read, Update, Delete, CRUD*) (Pokorný, 2015). Usein tietokirjallisuudessa kuitenkin GDBMS:iin viitataan graafitietokantoina. Tästä eteenpäin tutkielmassa viitataan graafitietokannan hallintajärjestelmään graafitietokantana.

Graafitietokannat jaetaan prosessointi- ja tallennustapansa mukaan kahteen eri luokkaan: natiiviin ja ei-natiiviin. Ei-natiivi graafitietokanta käyttää prosessoinnissa lähtökohteisesti globaalia indeksöintiä (engl. *global indexing*) ylläpitääkseen solmujen välisiä yhteyksiä. Globaali indeksöinti tarkoittaa koko tietokannan kattavaa indeksöintiä, joka ei ole rajoitettu yksittäiseen tauluun tai sarakkeeseen. Nämä indeksit lisäävät jokaiseen solmun väliseen läpikulkuun yhden lisävaiheen, josta seuraa hitaampi kysely ja enemmän las-



Kuva 2.1. Graafimallinnettua dataa (Neo4j Inc., 2024d)

kentää. Indeksöintiin perustuva haku ei-natiivissa graafitietokannassa on kompleksisuudeltaan $O(\log(n))$ kun taas natiiviin graafitietokantaan haku tapahtuu kompleksisuudella $O(1)$. Indeksöintiin perustuva haku toimii pienille graafeille, mutta suurille graafeille ei-natiivi lähestymistapa vaatii liikaa laskentatehoa ollakseen käytännöllinen. Pienellä graafilla tarkoitetaan graafia jossa solmujen lukumäärä on muutama tuhat ja suurella graafilla graafia jossa solmujen lukumäärä on kymmenissä tuhansissa, molemmissa kaarien lukumäärä olisi noin kymmenkertainen solmujen lukumäärä. Ei-natiivi tallennustapa tarkoittaa sitä, että graafitietokanta sarjallistaa graafidatan esim. relaatiotietokantaan. Tämän lähestymistavan hyöty on siinä että taustalla oleva järjestelmä (esim. MySQL) on entuudestaan hyvin tunnettu ohjelmointipiireissä. (Robinson ym., 2015)

Natiivi graafitietokanta käyttää tiedon prosessointiin indeksittömän vierekkäisyyden periaatetta (engl. *index-free adjacency*). Indeksittömässä vierekkäisyydessä jokaisella solmulla on suorat viitteet sen molemmilla puolilla oleviin solmuihin. Tällöin jokainen solmu toimii vieressä olevien solmujen mikroindeksinä (engl. *micro-index*), jolloin globaalia indeksöintiä ei tarvita. Indeksivapaasta vierekkäisyydestä seuraa tehokkaampia kyselyitä, sillä kyselyajat ovat koko tietokannan koon sijasta verrannollisia ainoastaan haun aikana käsiteltäviin alkioihin. Indeksivapaa vierekkäisyys sopii erityisen hyvin kyselyihin joissa tarvitaan yksi indeksihaku aloitussolmun löytämiseksi jonka jälkeen voidaan graafissa edetä seuraamalla (engl. *dereferencing*) fyysisiä osoittimia pitkin. Vaikka kyselyt ovat tehokkaampia, on niiden hintana se että kyselyt joissa ei tarvitse tehdä hakua voivat olla hankalia ja intensiivisiä muistin käytön suhteen. Natiivi tallennustapa (engl. *native graph storage*) nimensä mukaisesti tallentaa tiedon graafin muodossa. Sen etuja on suorituskyky ja skaalautuvuus verrattuna ei-natiiviin tallennustapaan. (Pokorný, 2015; Robinson ym., 2015)

3. NEO4J

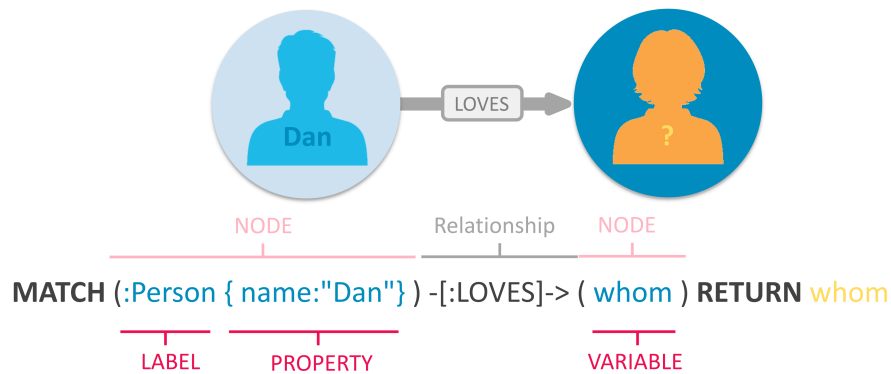
Neo4j on vuonna 2007 ensimmäisen kerran julkaistu natiivi graafitietokanta. Neo4j on suosituin kaikista graafitietokannoista (DB-Engines, 2024). Neo4j pohjautuu avoimeen lähdekoodiin ja se on implementoitu Javalla ja Scalalla. Vaikka Neo4j onkin NoSQL-tietokanta, se lupaa ACID-periaatteiden mukaisen eheyden. Tämän voi tulkita eduksi, sillä sen eheys ja luotettavuus on perinteisempien relaatiotietokantojen tasolla. (Fernandes & Bernardino, 2018)

Neo4j tarjoaa asiakkailleen graafitietokannan joko pilvipalveluna tai itse-ohjattavana. Itse-ohjatusta Neo4j:stä löytyy ilmainen Community-versio, jonka tarkoitus on olla opetusalus- ta tai pienten sovellusten tietokanta sekä myös maksullinen Enterprise-versio, joka on skaalautuvampi yritystason ratkaisu. Enterprise-version avulla on muun muassa mahdol- lista monitoroida ja varmuuskopioida dataa. Neo4j:llä on oma pilvipalvelualusta nimel- tään AuraDB, jonka avulla voi käyttää Neo4j-instanssia pilvessä. AuraDB:stä löytyy kol- me eri versiota: ilmaisversio joka, on verrattavissa itse-ohjattavaan Community-versioon, Professional-versio, jonka tavoite on olla ratkaisu skaalaltaan keskitason sovelluksille ja Enterprise-versio eli yrityksille kohdennettu suurimman skaalan versio. (Neo4j Inc., 2024c)

Neo4j:n avulla on helppo mallintaa verkostoja, sillä se on optimoitu kartoittamaan, ana- lysoimaan, tallentamaan ja hakemaan verkostopohjaista dataa. Neo4j:n yleisiä sovellus- kohteita ovat esimerkiksi koneoppiminen sekä tekoäly, IoT, reaaliaikaiset suositukset, pe- toksen tunnistaminen ja verkkoliikenteeseen sekä informaatioteknologiaan liittyvät sovel- lukset. IoT-järjestelmissä on usein tarve esittää reaaliaikaista dataa ja ne vaativat työkalu- ja, joilla yhteyksien esittäminen tapahtuu nopeasti. Neo4j:ssä tämä onnistuu operaatioilla mitkä relaatiotietokannassa muuten vaatisivat SQL:n paljon laskentatehoa vaativia JOIN- operaatioita. (Fosić & Šolić, 2019)

Neo4j on kehittänyt oman ominaisuusgraafimalleille suunnatun kyselykielen nimeltään Cypher. Cypher sai alkunsa Neo4j:ssä mutta on nyt sopeutettu myös muille graafitietokan- noille openCypher-projektin kautta, joka on avoimen lähdekoodin implementaatio Cyp- heristä. Graafitietokannat kuten esimerkiksi Redis Graph ja Memgraph käyttävät omaa implementaatiotaan Cypheristä. Cypher-kysely ottaa syötteenä ominaisuusgraafin ja pa- lauttaa taulukon. Cypher-kyselyt kootaan lineaarisesti, eli kyselyssä edetään kyselyn teks-

tin alusta loppuun. Tämän huomaa kyselyssä esimerkiksi siinä, että SQL-kielellä kyselyissä alussa olevan SELECT-lausekkeen sijaan Cypherissä on RETURN-lauseke, joka on kyselyn lopussa. Cypher on tarkoituksenmukaisesti syntaksiltaan samankaltainen kuin SQL, jotta käyttäjien olisi helpompaa vaihtaa kielten välillä. Cypher noudattaa syntaksiltaan samaa lausekerakennetta kuin SQL ja toteuttaa vakiintuneet semantiikat useille toiminnolle. Cypherin syntaksi on selkeä ja deklarativinen, joten se on helppolukuista ja helpottaa ymmärtämään, miten graafin solmut ja suhteet ovat yhteydessä toisiinsa. (Francis ym., 2018) Kuvassa 3.1 on esitettyä esimerkikysely Cypherillä. Kyselyn rakenne on yksinkertainen, MATCH-lausekkeella etsitään solmua, joka vastaa annettuun solmuun ja suhteeseen.



Kuva 3.1. Kuvaan 2.1 kohdistettu Cypher-kysely (Neo4j Inc., 2024d)

Useimmista tietokannoista poiketen Neo4j:tä voi käyttää palvelimen lisäksi sulautettuna. Tällöin Neo4j instanssi tallentaa kaiken tiedon suoraan levyille. Tämä sovelluskohde on yleinen työpöytäsovelluksille ja laitteille, mahdollistaen käyttäjälle Neo4j:n käytön matalalla kynnyksellä. DB-Engines (2024) suorittamassa vertailussa Neo4j:stä nousee esille lisäksi Neo4j:ssä tallennettuna olevan graafin kyky fragmentoitua, mikä minimoi kyselyn viiveen, korkean suorituskyvyn hajautettu klusteriarkkitehtuuri sekä helposti ymmärrettävä kyselykieli. Neo4j myös antaa käyttäjälle mahdollisuuden käyttää useampia ohjelmointikieliä sen käsittelyyn kuin esimerkiksi seuraavaksi suosituimmat Memgraph ja Nebula-Graph.

4. NEO4J DATAN KÄSITTELYN TYÖKALUNA

Tietokantaan saapuvan datan yhtenäisyys ei ole aina taattua ja saadakseeseen paremman ymmärryksen siitä tulee sitä erilaisilla työkaluilla käsitellä, manipuloida ja tutkia. Neo4j tarjoaa käyttäjille pääsyn Graph Data Science -kirjastoon, jonka avulla käyttäjät voivat manipuloida ja tutkia dataansa erilaisilla datatieteeseen liittyvillä operaatioilla ja työkaluilla kuten algoritmeilla ja koneoppimisputkilla. Suurten datamäärien normalisoituessa on myös tärkeää tutkia Neo4j:n suorituskykyä verrattuna muihin relaatio- ja NoSQL-tietokantoihin käsiteltäessä tätä dataa.

4.1 Graph Data Science -kirjasto

Neo4j Graph Data Science -kirjasto (*GDS*) julkaistiin ensimmäisen kerran vuonna 2020. Se on kirjasto joka pitää sisällään datatieteeseen (engl. *data science*) liittyviä työkaluja ja algoritmeja. Graafialgoritmit *GDS*:ssä ovat pääosin valvomattomia algoritmeja joiden avulla voidaan kerätä tietoa graafin rakenteesta. Rakenteesta voidaan muiden algoritmien avulla saada selville piirteitä kuten solmujen sentraalisuus eli tärkeys, solmu-yhteisöt ja samanlaisuus eli solmujen välistä vertailua samanlaisuutta määrittämällä. Polkuihin liittyvät algoritmit ovat solmujen välisiä polkuja varten. Lyhimmän polun löytäviä algoritmeja on lukuisia, mutta algoritmit kuten esimerkiksi A^* ja Dijkstra ovat implementoitu *GDS*:ssä. *GDS*:ää käyttäekseen ei välttämättä tarvitse käyttää Cypheriä, sillä *GDS*:n käyttö onnistuu myös Pythonilla. (Scifo, 2023)

Vargas-Solar ym. (2021) vertailivat tutkimuksessaan kirjastoa Networkx, jonka tarkoitus on analysoida graafitietokantoja käyttämällä Pythonia, ja Neo4j:n graafitietokannan hallintaan tehtyä *GDS*:ää. Taulukossa 4.1 on esitetty suoritusajat eri algoritmeille ensimmäisellä ajokerralla. Neo4j:n sarakkeessa suluissa oleva luku tuloksissa on ensimmäisen ajokerran jälkeen suoritettu saman operaation suoritusajan keskiarvo useammalla kokeilulla. Tämä on ilmoitettu siksi, että vaikka graafit on tallennettu Neo4j:n palvelimella, *GDS*:ää käyttäessä vaaditaan käyttäjältä muistinhallintaa ja graafin osien reitittämistä suoritusympäristöön. Tarkastelussa onkin Neo4j:n välimuistin vaikutus suoritusajaan verrattuna välimuistittomaan Pythonilla ajettavaan Networkx:ään.

Tuloksien perusteella Networkx vaikuttaa nopeammalta ratkaisulta suurimpaan osaan algoritmeista, mutta Jupyter-notebookissa Pythonilla ajettavalla Networkx:llä on kuitenkin

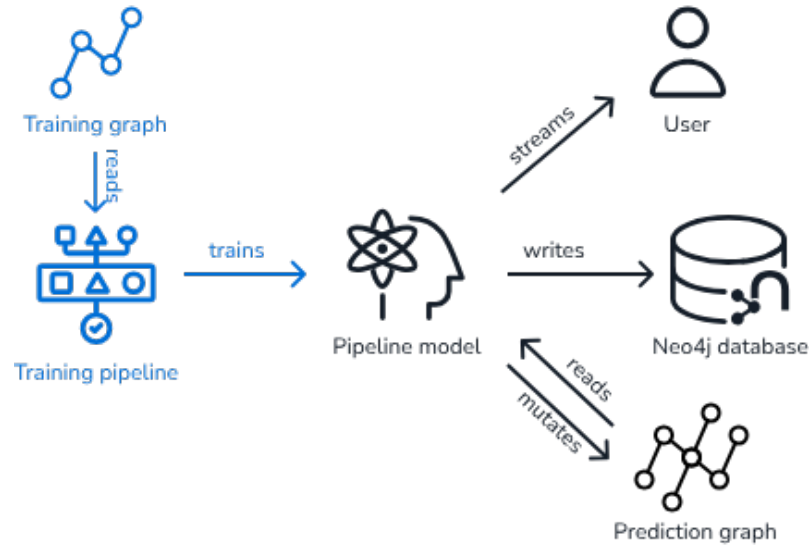
Taulukko 4.1. Suoritusvertailu erilaisille algoritmeille (Vargas-Solar ym., 2021)

Algoritmi	Neo4j (s)	Networkx (s)
Betweenness-sentraalisuus	0,32 (0,08)	1,6
PageRank (painoton)	0,36 (0,1)	0,09
PageRank (painollinen)	0,36 (0,1)	0,12
Label Propagation	0,17 (0,05)	1e-6
Breadth First Search, BFS	38	3e-7
Minimum spanning tree	0,52 (0,06)	7e-3

ongelmia muistinhallinnan kanssa. Ajon aikana kaikki data graafissa tallennetaan keskusmuistiin, jolloin erityisen suurilla graafeilla muistinhallinnasta saattaa tulla käyttäjälle ongelma. Myös Neo4j käsittelee graafeja keskusmuistissa, mutta Cypherin avulla käsiteltävät graafit ovat pysyviä näkymiä (engl. *persistent views*) alkuperäisestä graafista. Pysyvä näkymä voi esittää tiettyä osajoukkoa graafista jonkin rajoituksen mukaan, jolloin ei ole tarpeen tallentaa koko graafia keskusmuistiin. Neo4j:lla oli ongelmia BFS-algoritmin suorituksessa, joka johtui mahdollisesti siitä, että algoritmi käsiteltiin reaaliajassa. Kokonaisuutena Neo4j suoriutui hyvin siihen nähden, että käytetyt algoritmit olivat Alpha-vaiheessa eli kokeiluvaiheessa. (Vargas-Solar ym., 2021)

GDS-kirjastossa on myös koneoppimisputken (engl. *machine learning pipeline*) konsepti. Koneoppimisputkella tarkoitetaan kokonaisuutta jonka avulla voi automatisoida koko prosessin graafin piirteiden irroitukselta (engl. *feature extraction*) koneoppimismallin (engl. *pipeline model*) opettamiseen. Opetusputki (engl. *training pipeline*) on osa koneoppimisputkea ja sen tarkoitus GDS:ssä on yksinkertaistaa koneoppimismallin opetusvaiheita. Jo opetusvaiheen käyneen koneoppimisputken mallit eli opetusputkesta syntyneet koneoppimismallit, säilötään katalogissa, josta ne voi myöhemmin ladata tehdäkseen ennusteita. Kuvassa 4.1 on esitetty miten opetusputki ja siitä syntyvä koneoppimismalli voivat käyttäytyä. GDS tarjoaa kolme vaihtoehtoa opetusputken muodostaman koneoppimismallin valintaan. Solmun luokitus -putki (engl. *node classification pipeline*) on koneoppimisputki, jossa syntyvä malli määrittää jokaisen solmun yhteen luokkaan valvonnan alla. Solmuregressio-putki (engl. *node regression pipeline*) on koneoppimisputki, josta syntyvä malli yrittää estimoida numeerista arvoa solmuille. Yhteyksien ennustus -putki (engl. *link prediction pipeline*) on koneoppimisputki, josta syntyvän mallin avulla voidaan ennustaa tulevaisuuden yhteyksiä osittaisessa tai aikakehittyvässä verkossa. Kolme edellä mainittua koneoppimisputket ovat vielä Alpha- ja Beta -vaiheissa, eli kehityksen alla. (Scifo, 2023)

Yhteyksien ennustus -putkea varten sekä Networkx:n että Neo4j:n alustat yhdistettiin käyttäen hyväksi Apache Sparkia. Putki suunniteltiin löytämään piilotettuja yhteyksiä solmujen välillä perustuen niiden ominaisuuksiin. Datajoukko jaettiin oppimisjoukkoon ja tes-



Kuva 4.1. Koneoppimisputki (Neo4j Inc., 2024b)

tijoukkoon, ja solmuille luotiin ominaisuuksia graafissa reunojen luokittelua varten. Yhteyksien ennustamista varten kehitettiin kaksi mallia. Mallit suoriutuivat hyvin, saavuttaen parhaimmillaan noin 92 %:n tarkkuuden. Imperatiivisen (Networkx) ja deklarativisen (Neo4j) lähestymistavan vertailussa Vargas-Solar ym. (2021) totesivat lopputuloksesta että yhteyksien ennustus -putken yhteydessä tehdyt kokeilut osoittivat, että Neo4j:n tarjoamat pysyvät näkymät voivat olla elegantteja ja järkeviä tähän käyttötarkoitukseen. Apache Sparkin avulla lähestymistapojen yhdistäminen toi luonnollisemman tavan käsitellä muistinkäyttöä kahden alustan välillä.

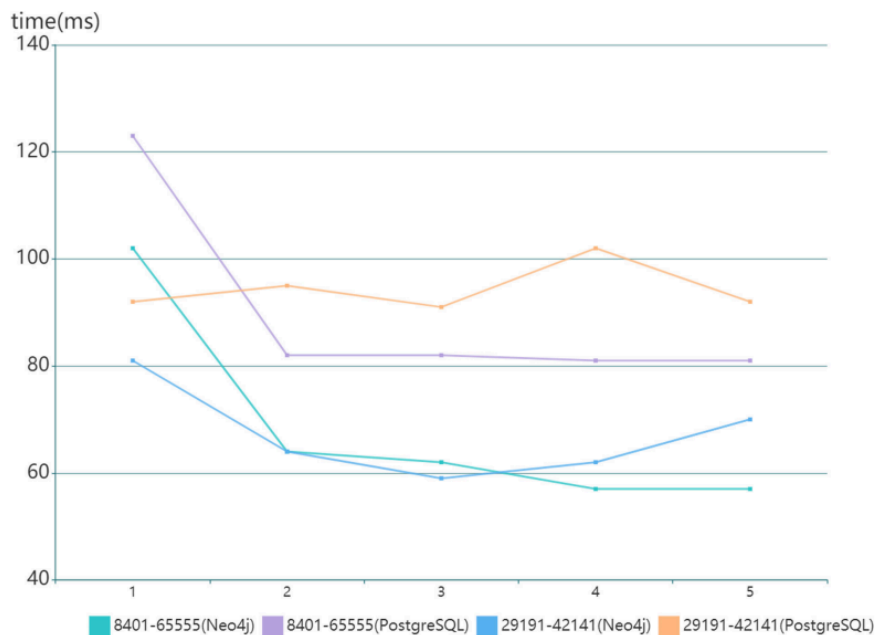
4.2 Big Data

Big Data on nykyajan suuria datamääriä kuvaava termi, jonka mukaan Big Dataa määrittää kolme V:tä: *Volume*, eli suuret datan määrät, *Variety* eli datan moninaisuus joka kuvaa sitä kuinka dataa tulee monista eri lähteistä eri muodoissa, ja *Velocity*, eli se että dataa generoituu nopealla tahdilla. Big Data-käsitteellä kuvaillaan jatkuvaa datan määrän kasvua, sekä työkaluja joita tarvitaan käsittelyyn. (Perçuku ym., 2017) Tämän datan käsittely, mainpulointi ja hallinta on tämän hetken tärkeimpiä teknisiä harkinnankohteita ja erilaiset tietokantaratkaisut ovat näiden harkintojen ydin. Perinteisemmän relaatiomallin mukaiseen tietokantaan kyselyä suorittaessa tulee vastaan kalliit liitosoperaatiot, sillä relaatioita rivien välillä on niin paljon. Datamäärä on kasvamaan päin jolloin relevantit relaatiot rivien välillä kasvavat eksponentiaalisesti. Tällöin relaatiotietokannan liitosoperaatiot ainoastaan kallistuvat. Graafitietokannat voivat käsitellä vahvasti kytkeytynyttä dataa ilman liitosoperaatioita, jolloin teoriassa ne suoriutuvat paremmin käsittelemään vahvasti kytkeytynyttä dataa. (J. Chen ym., 2020)

J. Chen ym. (2020) vertailivat tutkimuksessaan Neo4j:tä ja relaatiotietokanta PostgreSQL:ää

käsiteltäessä suuria data määriä jotka liittyvät Kiinassa sijaitsevan Shenzhen-alueen tieverkostoon. OpenStreetMap projektista saadussa datajoukossa on 36968 solmua ja 86230 kaarta, eli *suhdetta*. Solmut kuvaavat muun muassa risteyskiä, suojateitä ja teiden päätepisteitä ja kaaret puolestaan kuvaavat teitä solmujen välillä. Datajoukossa jokaisella tiellä on solmu sen molemmissa päissä, eli päättömiä kaaria ei ole. Ylimääräiset OpenStreetMapista tulleet tiedot, kuten tien nimi ja poikkeutuskulma, poistettiin datajoukosta. Kaarien painona algoritmeja varten käytettiin vastaavan tien pituutta ja tiedot kuten esimerkiksi aloitus- ja lopetussolmun avulla saatu tien suunta säilytettiin datajoukossa.

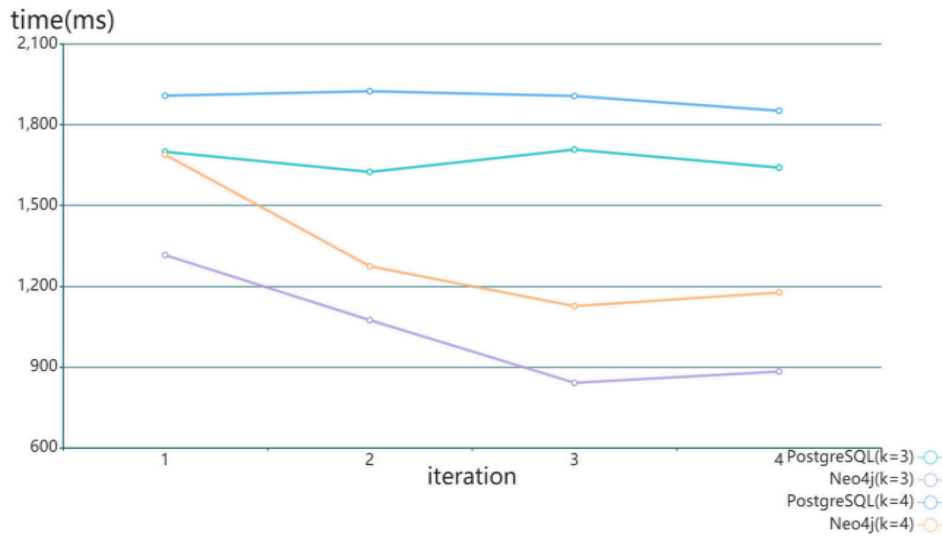
J. Chen ym. (2020) vertailivat tutkimuksessaan tieverkoston aligraafeihin (engl. *subgraph*) suoritettavien algoritmien suorituskykyä erikokoisilla graafeilla. Ensin tutkimuksessa vertailtiin lyhimmän polun löytävää algoritmia, tässä tapauksessa Dijkstran algoritmia. Tulokset on viidelle peräkkäiselle suorituskerralle esitetty kuvassa 4.2. Tässä kokeessa kuten muissakin esitettävissä tuloksissa y-akselilla on suoritus aika millisekunteina (*time (ms)*) ja x-akselilla iteraation, eli ajokerran järjestysnumero (*iteration*). Tuloksien perusteella voidaan tulkita, että Dijkstran algoritmi verkostopohjaisella datalla soveltuu Neo4j:lle esimerkiksi. Se suorituu PostgreSQL:ää huomattavasti paremmin sekä kooltaan suuremmalla että pienemmällä graafilla.



Kuva 4.2. Suoritusajat Dijkstran algoritmilla 2 eri graafin koolla, 5 iteraatiota (J. Chen ym., 2020)

Kuvassa 4.3 on esitetty K-lyhimmän polun algoritmin suoritusajat k :n arvoilla 3 ja 4. K-lyhimmän polun algoritmi etsii annetun k :n verran lyhimpiä polkuja, joten kyseessä on suhteellisen raskas algoritmi, jonka huomaa myös suoritusajoista verrattuna esimerkiksi Dijkstran algoritmin suoritusajoihin. Neo4j kuitenkin suoriutuu algoritmista huomattavasti paremmin kuin PostgreSQL, liki puolittaen PostgreSQL:n suoritusajan kolmannella

iteraatiolla k :n arvolla 3.



Kuva 4.3. K-lyhimmän polun algoritmin suoritusajat, 4 iteraatiota (J. Chen ym., 2020)

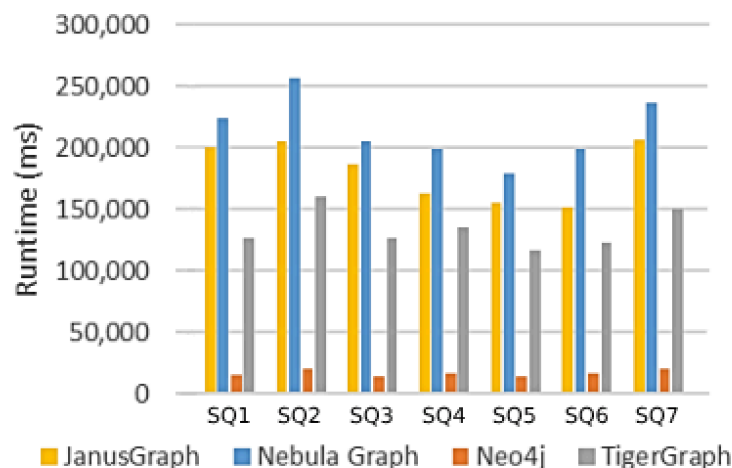
J. Chen ym. (2020) esittivät tutkimuksen yhteenvetona tulokset jotka osoittivat, että Neo4j suoriutui keskimäärin lähes 30 % paremmin kuin PostgreSQL. Neo4j:n etuna oli, että sen ei tarvinnut käydä koko graafia läpi halutun solmun löytämiseksi. Neo4j kuitenkin tarvitsi enemmän aikaa tieverkoston lataamiseen sekä pienille että suurille datamäärille. Tutkimuksen johtopäätöksenä todettiin, että tieverkosto on sopiva sovelluskohde Neo4j:lle, koska tieverkosto on luonnostaan graafirakenne ja vaatii joustavuutta muutoksille, jota Neo4j tarjoaa skeemattomuutensa avulla. Lisäksi tutkimuksessa todettiin, että graafitietokannat kuten Neo4j sopivat erityisesti aloille, joilla datan yhdistyneisyys tai topologia on tärkeää, kuten sosiaalisissa verkostoissa tai suositusjärjestelmissä.

Monteiro ym. (2023) vertailivat tutkimuksessaan neljää eri graafitietokantaa: JanusGraphia, NebulaGraphia, TigerGraphia ja Neo4j:tä. Nämä neljä valittiin, koska ne olivat neljän kärki graafitietokantojen joukossa (poislukien multi-model-tietokannat) DB-Enginesin vertailussa vuonna 2022. Suorituskykyä testattiin LDBC SNB (Linked Data Benchmark Social Network Benchmark) suorituskykytestillä. LDBC SNB on tieteellisissä piireissä tunnettu graafitietokantojen suorituskykytesti. Testin käyttämät datajoukot kuvaavat oikeita sosiaaliverkostoja, joten ne ovat ideaaleja graafitietokannoille, sillä sosiaaliverkot esitetään usein luonnostaan graafeina. Suorituskykytestien tulokset ovat siis relevantteja tietokantojen käytännön käyttötarkoituksiin. Suorituskykytestit koostuvat 29 eri kyselystä, joilla pelkkien kyselyjen lisäksi tehdään päivityksiä tietokantoihin. Kyselyistä 14 oli kompleksia ja kyselyitä, 7 yksityiskohtaisempaa ja kapeaa kyselyä nimellä *Short Query* ja 8 kyselyä, jotka suorittivat päivityksiä tietokantaan. Kyselyt poikkeavat toisistaan vaatavuustasolla.

Tutkimuksessa suoritettiin erikokoisille graafeille samat 29 kyselyä, jokainen kysely 5 kertaa, jotta voitiin tutkia välimuistin vaikutusta suoritukseen. Tuloksissa näkyvät luvut ovat 5

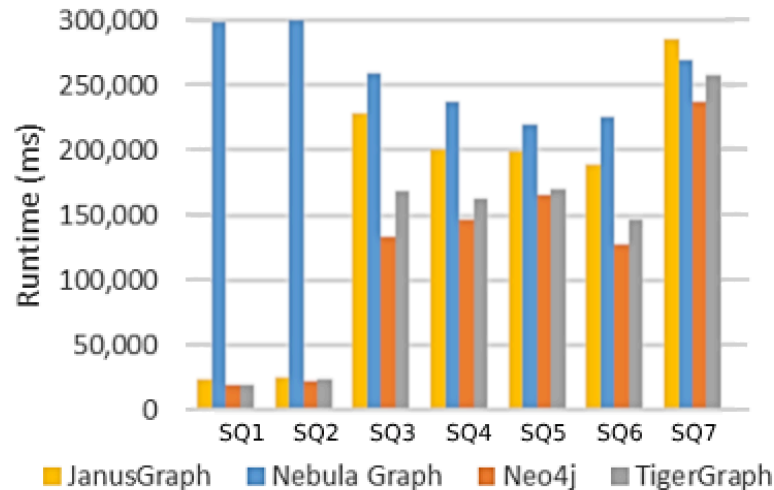
suorituskerran keskiarvoja. Graafien koot perustuivat mittakaavatekijään (engl. *scale factor*, SF), jonka mahdollisia arvoja oli 0,1, 0,3, 1, 3 ja 10. Datajoukon koko mittakaavatekijän ollessa arvoltaan 1 oli 1,22 GB, 3 oli 3,68 GB ja 10 oli 12,2 GB. Käytetään vertailussa SF:n arvoja 3 ja 10, jotta saadaan kuvaa graafitietojen suorituskyvystä suurehkoilla, Big Datan käsitettä mukailevilla datajoukoilla. (Monteiro ym., 2023)

Kuvissa 4.4 ja 4.5 on esitetty suorituskykytestin tulokset SF:n arvolla 3 ja 10. Tulokset on rajattu seitsemään raskaimpaan operaatioon, joissa näkyy suurimmat erot tietokantojen välillä. Nämä ovat testin *Short Queryt*. Tuloksissa y-akselilla on ajoaika millisekunteina (*Runtime (ms)*), ja x-akselilla kyselyt ajajärjestyksessä vasemmalta oikealle. Kuvassa oranssilla näkyvät Neo4j:n tulokset. SF:n arvon ollessa 3 on Neo4j huomattavasti muita nopeampi suoritusajoissaan. Tämä kertoo sen mahdollisuuksista haastavillakin kyselyillä kun datajoukko on rakenteeltaan sosiaaliverkosto. SF:n arvolla 10 datajoukon koko kasvaa huomattavasti noin 3 GB:stä noin 12 GB:een, jolloin graafien latausajatkin nousivat kaikilla tietokannoilla 100 % verrattuna 3 GB datajoukkoon. Jälleen Neo4j on kaikista tehokkain tietokanta vaativista kyselyistä huolimatta. Tämä kertoo Neo4j:n suorituskyvystä sekä skaalautuvuudesta kuvatulla datamäärän kasvulla.



Kuva 4.4. LDBC SNB- suorituskykytestin tulokset Short Queryjen osalta, SF = 3, muokattu lähteestä Monteiro ym. (2023)

Monteiro ym. (2023) vertailivat tutkimuksessaan samoilla SF:n arvoilla myös datan latausaikaa sekä keskusmuistin (RAM) että prosessorin (CPU) käyttöä suorituskykytestien aikana. Kyseisten mittausten keskiarvotulokset jokaisella SF:n arvolla kokeen aikana on esitetty taulukossa 4.2. Taulukkoon on myös lisätty suoritusajojen kumulatiivinen summa jokaisen tietokannan kohdalla. Neo4j oli selvästi nopein ladattaessa datajoukkoja tietokantaan. Se oli myös selvästi optimoiduin keskusmuistin käytön suhteen. Prosessorin käytössä oli suuria eroja, ja silti Neo4j oli selvästi vähiten CPU:ta käyttävä tietokanta. Kyselyajojen kumulatiivisessa summassa oli suurin ero. Neo4j oli ajallisesti jopa yli 4 ker-



Kuva 4.5. LDBC SNB- suorituskykytestin tulokset Short Queryjen osalta, SF = 10, muokattu lähteestä Monteiro ym. (2023)

taa nopeampi kuin seuraavaksi suosituin graafitietokanta NebulaGraph. Suoriuduttuaan muita paremmin kaikissa tutkimuksen kokeissa on ilmeistä, että Neo4j on tehokkuus- aspekteiltaan ylivoimainen verrattuna muihin vertailussa olleisiin graafitietokantoihin.

Taulukko 4.2. Kootut tulokset Monteiro ym. (2023) tutkimuksesta

	JanusGraph	NebulaGraph	Neo4j	TigerGraph
Latausaika (ms)	668917	683123	469297	536339
RAM käyttö (%)	49	62	42	51
CPU käyttö (%)	41	48	26	37
Kyselyaika (min)	77.58	101.58	24.3	60.97

Monteiro ym. (2023) totesivat tutkimuksen lopputuloksena että Neo4j suoriutui paremmin kuin muut grafitietokannat kokonaisuutena niin datan latausajan kuin kyselyn suoritusai-kojenkin osalta. Lisäksi se tarjoaa helpokäyttöisen käyttöliittymän sekä käyttäjien ja oh- jelmiojien keskuudessa tehokkaimpana ja intuitiivisimpana pidetyn Cypher-kyselykielen. Siitä huolimatta, että TigerGraph tarjoaa samanlaisia ominaisuuksia kuin Neo4j, se on huomattavasti hitaampi käsiteltäessä suuria datamääriä. Tuloksien perusteella Neo4j on tietokanta, joka tarjoaa muihin tutkittuihin tietokantoihin verrattuna tehokkaan alustan jos- sa käsitellä suuriakin datamääriä sekä korkean tason skaalautuvuuden että pienen la- tenssin.

5. POHDINTA

Neo4j:n rooli datan käsittelyn työkaluna korostuu erityisesti sen tarjoaman Graph Data Science -kirjaston myötä. Tuloksena voi olla syvällisempi ymmärrys datan rakenteesta, mikä puolestaan auttaa yrityksiä päätöksenteossa ja liiketoiminnan optimoinnissa. Tämä voi olla erityisen hyödyllistä monimutkaisten verkostojen kuten, sosiaali- ja tieverkostojen analysoinnissa. Kirjaston tarjoamat työkalut, kuten graafialgoritmit ja koneoppimisputket, tarjoavat yrityksille mahdollisuuden syventää analyysiä ja luoda monipuolisia malleja monimutkaisten verkostojen rakenteiden ymmärtämiseksi ja hyödyntämiseksi liiketoiminnassa. Vaikka Graph Data Science -kirjasto lupaa uusia mahdollisuuksia datatieteen ja koneoppimisen alueilla, on syytä muistaa, että se on vielä kehittyvässä vaiheessa. Tämä tarkoittaa sitä, että kirjaston tarjoamat algoritmit ja työkalut eivät välttämättä ole vielä täysin kypsiä kaikkiin käyttötapauksiin. Nämä algoritmit ja työkalut tuloksien perusteella kuitenkin vaikuttavat lupaavilta.

Se että Neo4j pystyy lupaamaan ACID-periaatteiden mukaisen eheyden, on sille suuri etu. Sen lisäksi että Cypher on suunniteltu syntaksiltaan samantyylliseksi kuin SQL, Neo4j:n eheys on perinteisempien relaatiotietokantojen tasolla, tehden siitä houkuttelevamman yrityksille ja pienentäen kynnystä vaihtaa Neo4j:hin. ACID-periaatteiden ja natiivin graafitietokannan tallennus- ja prosessointitapojen avulla Neo4j mahdollistaa käyttäjilleen eheän, vapaasti hallittavan ja tehokkaan paikan tallentaa dataa. Tämä yhdistelmän avulla Neo4j voi säilyttää asemansa johtavana graafitietokantana. Lisäksi Neo4j:n tarjoamat ilmaisversiot pienemmille sovelluksille ovat merkki siitä, että yritys pyrkii tekemään graafitietokannat saavutettavammiksi ja houkuttelevammiksi laajalle käyttäjäkunnalle. Tämä lähestymistapa avaa ovia pienille yrityksille ja kehittäjille hyödyntää Neo4j:n tehokkaita graafitietokantaratkaisuja ilman suurta taloudellista sitoutumista.

Suuria datamääriä käsitellessä Neo4j:n rooli voi olla merkittävä, etenkin kun kyseessä on vahvasti kytkeytyneen datan käsittely ilman liitosoperaatioita. Perinteiset relaatiotietokannat vaativat usein tarkan skeeman ennalta määrittelyn, mikä voi olla haastavaa tai jopa mahdotonta tilanteissa, joissa datan rakenne on kompleksinen ja muuttuva. Tässä graafitietokannat kuten Neo4j tarjoavat ratkaisun, sillä ne mahdollistavat datan tallentamisen ja käsittelyn ilman vahvaa ennalta määriteltyä skeemaa. Tämä tekee niistä erityisen soveltuvia tilanteisiin, joissa datan rakenne on dynaaminen ja verkostomainen. Neo4j:n oman AuraDB:n avulla käyttäjät voivat hyödyntää graafitietokannan tehokkuutta pilvipalveluna,

mikä tarjoaa joustavuutta ja skaalautuvuutta suurten datamäärien käsittelyyn. Tämä on erityisen tärkeää tilanteissa, joissa datan määrä kasvaa nopeasti tai kun tarvitaan resursien skaalaamista vastaamaan vaihtelevia käyttötarpeita.

Vertailut muihin relaatio- ja NoSQL-tietokantoihin tarjoavat perspektiiviä Neo4j:n suorituskyvystä erilaisissa käytännön käyttökohteissa. Vaikka Neo4j oli hitaampi joissain yksittäisissä operaatioissa, kuten Breadth First Search-algoritmissa, sen vahvuudet tulevat esiin erityisesti monimutkaisten graafialgoritmien ja koneoppimisputkien käytössä. Tämä voi olla ratkaisevaa yrityksille, jotka tarvitsevat nopeaa ja tehokasta analysointia verkostomaisen datan perusteella.

Kun vertaillaan Neo4j:tä muihin tietokantaratkaisuihin, on tärkeää ottaa huomioon kunkin ratkaisun vahvuudet ja heikkoudet eri käyttötilanteissa. Esimerkiksi relaatiotietokannat voivat tarjota tehokkaan suoritusalueen tietyn tyyppisten kyselyjen suorittamiseen, erityisesti kun kyseessä on rakenteellisesti selvästi määritellyt ja normalisoidut tietokantataulut. Toisaalta NoSQL-tietokannat kuten Neo4j voivat skaalautua paremmin suuriin datamääriin ja tarjota joustavuutta erilaisten tietorakenteiden hallinnassa.

Ilman Neo4j:n omaa Cypher-kieltä datan manipulointi ei olisi niin helppoa ja onkin mielenkiintoista nähdä millaisia kehityssuuntia eri implementaatioilla openCypher-projektista on tulevaisuudessa. Sillä kehityksessä on otettu huomioon vaihtamisen helppous SQL:n ja Cypherin välillä, openCypheristä todennäköisesti implementoidaan versioita yhä useampaan paikkaan ajan saatossa.

On mielenkiintoista nähdä, miten Neo4j:n ja muiden graafitietokantojen kehitys jatkuu tulevaisuudessa ja miten ne vastaavat entistä monimutkaisempiin tarpeisiin liittyen datankäsittelyyn. Tekoälyn ja koneoppimisen kehitys tuovat mukanaan uusia haasteita ja mahdollisuuksia datan käsittelyssä, ja graafitietokannat saattavat tarjota ainutlaatuisia ratkaisuja näihin haasteisiin. Yritysten ja organisaatioiden tarpeet datan käsittelyssä ja analysoinnissa ovat myös jatkuvassa muutoksessa, ja Neo4j:n kaltaiset innovatiiviset tietokantaratkaisut voivat auttaa vastaamaan näihin muuttuviin tarpeisiin tehokkaasti ja joustavasti.

Kokonaisuutena Neo4j:n vahvuudet graafitietokantana sekä sen tarjoamat mahdollisuudet datatieteen ja koneoppimisen alueilla vaikuttavat lupaavilta. Vaikka Neo4j:n suorituskyky hieman vaihteli eri käyttötapauksissa, sen kyky käsitellä ja analysoida verkkorakenteista dataa tekee siitä varteenotettavan vaihtoehdon monille yrityksille. Tulevaisuudessa voidaan odottaa lisää innovatiivisia sovelluksia ja käyttökohteita, jotka hyödyntävät Neo4j:n tarjoamia työkaluja ja ratkaisuja datan käsittelyssä ja analysoinnissa.

6. YHTEENVETO

Tämän tutkielman tavoitteena oli tutustua Neo4j:n tarjoamiin ominaisuuksiin ongelmanratkaisun näkökulmasta. Tarkastelun kohteena olivat erityisesti Neo4j:n oman Graph Data Science -kirjaston tarjoamat työkalut, kuten koneoppimisputket ja graafialgoritmit. Lisäksi tarkasteltiin Neo4j:n suorituskykyä verrattuna muihin tietokantoihin. Näiden tarkastelujen pohjalta yritettiin löytää syitä miksi Neo4j on suosituin graafitietokanta ja miten se soveltuu datankäsittelyn työkaluksi suurilla ja komplekseilla datajoukoilla.

Lähteiden perusteella voidaan todeta että Neo4j on potentiaalinen ja moderni graafitietokanta joka on monipuolisten ominaisuuksiensa avulla noussut suosituimmaksi graafitietokannaksi. Neo4j:n pohjalla oleva graafimalli on tehokas ja erityisen optimoitu juuri vahvasti kytkeytyneelle kompleksille datalle. Se suoriutui suurilla sosiaali- ja tieverkostoilla muihin tietokantoihin verrattuna mainiosti, ollen nopein lähes kaikissa esitetyissä tuloksissa. Suoritusajojen lisäksi se käytti selvästi vähiten keskusmuistia ja prosessoria verrattuna muihin graafitietokantoihin. Neo4j:n uniikeista ominaisuuksista nousi esille sen tehokas Cypher-kyselykieli, Cypherin avulla graafien käsittelyä nopeuttavat pysyvät näkymät ja se, että useimmista NoSQL-tietokannoista poiketen Neo4j tarjoaa ACID-periaatteiden mukaisen eheyden. Lähteiden perusteella myös havaittiin että Neo4j:n Graph Data Science -kirjasto on potentiaalinen kehittyvässä vaiheessa oleva työkalu, jonka avulla graafien käsittely ja datan syvällisen ymmärtämisen edistäminen luonnistuu sen tarjoamien koneoppimisputkien ja algoritmien avulla. Vaikka kirjaston tarjoamat algoritmit eivät olleet aina nopeimpia verratessa Networkx:ään, suoriutui se hyvin siihen nähden, että algoritmit olivat vertailuhetkellä vielä Alpha- ja Beta-vaiheissa. Koneoppimisputkien avulla Neo4j mahdollistaa käyttäjilleen saada syvällisemmän ymmärryksen graafien rakenteesta ja ominaisuuksista.

Tulevaisuuden tutkimus- ja kehityssuunnat voivat sisältää esimerkiksi Graph Data Science -kirjaston laajentamisen ja sen algoritmien parantamisen sekä Neo4j:n integroinnin muiden työkalujen ja teknologioiden kanssa. Näiden kehityssuuntien avulla Neo4j voi säilyttää asemansa johtavana graafitietokantana ja vastata entistä paremmin käyttäjien tarpeisiin monilla eri toimialoilla. Vahvasti kytkeytyneen datan määrän ainoastaan kasvaessa tarve tehokkaille graafitietokannoille kasvaa. Jatkuvan kehityksen avulla Neo4j voi vahvistaa asemaansa johtavana graafitietokantana ja vastata entistä paremmin käyttäjien monipuolisiin tarpeisiin eri toimialoilla.

LÄHTEET

- Chandra, D. G. (2015). BASE analysis of NoSQL database. *Future Generation Computer Systems*, 52, 13–21. <https://doi.org/10.1016/j.future.2015.05.003>
- Chen, J.-K., & Lee, W.-Z. (2019). An Introduction of NoSQL Databases based on their categories and application industries. *algorithms*, 12(5), 106. <https://doi.org/10.3390/a12050106>
- Chen, J., Song, Q., Zhao, C., & Li, Z. (2020). Graph database and relational database performance comparison on a transportation network. *Advances in Computing and Data Sciences: 4th International Conference, ICACDS 2020, Valletta, Malta, April 24–25, 2020, Revised Selected Papers 4*, 407–418. https://doi.org/10.1007/978-981-15-6634-9_37
- DB-Engines. (2024). *DB-Engines Ranking* [Haettu 14.02.2024]. <https://db-engines.com/en/ranking>
- Fernandes, D., & Bernardino, J. (2018). Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB. *Data*, 10.
- Fosić, I., & Šolić, K. (2019). Graph database approach for data storing, presentation and manipulation. *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1548–1552. <https://doi.org/10.23919/MIPRO.2019.8756793>
- Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., Plantikow, S., Rydberg, M., Selmer, P., & Taylor, A. (2018). Cypher: An Evolving Query Language for Property Graphs. *Proceedings of the 2018 International Conference on Management of Data*, 1433–1445. <https://doi.org/10.1145/3183713.3190657>
- HostingData. (2024). *NoSQL Databases* [Haettu 15.02.2024]. sql-database.org
- Kunda, D., & Phiri, H. (2017). A Comparative Study of NoSQL and Relational Database. *Zambia ICT Journal*, 1(1), 1–4. <https://doi.org/10.33260/zictjournal.v1i1.8>
- Monteiro, J., Sá, F., & Bernardino, J. (2023). Experimental Evaluation of Graph Databases: JanusGraph, Nebula Graph, Neo4j, and TigerGraph. *Applied Sciences*, 13(9), 5770. <https://doi.org/10.3390/app13095770>
- Neo4j Inc. (2024a). *Graph database concepts* [Haettu 08.02.2024]. <https://neo4j.com/docs/getting-started/appendix/graphdb-concepts/>
- Neo4j Inc. (2024b). *Machine learning - Neo4j* [Haettu 24.02.2024]. <https://neo4j.com/docs/graph-data-science/current/machine-learning/>
- Neo4j Inc. (2024c). *Neo4j Pricing* [Haettu 12.02.2024]. <https://neo4j.com/pricing/>

- Neo4j Inc. (2024d). *What is a Graph Database?* [Haettu 20.02.2024]. <https://neo4j.com/developer/graph-database/>
- Neo4j Inc. (2024e). *The World's Leading Organizations Rely on Neo4j* [Haettu 08.02.2024]. <https://neo4j.com/who-uses-neo4j/>
- Perçuku, A., Minkovska, D., & Stoyanova, L. (2017). Modeling and processing big data of power transmission grid substation using neo4j. *Procedia Computer Science*, 113, 9–16. <https://doi.org/10.1016/j.procs.2017.08.276>
- Pokorný, J. (2015). Graph databases: their power and limitations. *Computer Information Systems and Industrial Management: 14th IFIP TC 8 International Conference, CISIM 2015, Warsaw, Poland, September 24-26, 2015, Proceedings 14*, 58–69. https://doi.org/10.1007/978-3-319-24369-6_5
- Robinson, I., Webber, J., & Eifrem, E. (2015). *Graph databases: new opportunities for connected data*. "O'Reilly Media, Inc."
- Scifo, E. (2023). *Graph data science with Neo4j : learn how to use Neo4j 5 with Graph Data Science Library 2.0 and its Python driver for your project* (1st ed.). Packt Publishing, Limited.
- Vargas-Solar, G., Marrec, P., & Halfeld Ferrari Alves, M. (2021). Comparing graph data science libraries for querying and analysing datasets: towards data science queries on graphs. *International Conference on Service-Oriented Computing*, 205–216. https://doi.org/10.1007/978-3-031-14135-5_16