Tampereen yliopisto

Elisa Akkanen

DATAINTEGRAATIOMETODIT SQL- JA NOSQL-TIETOKANNOISSA

Ratkaisuja hybriditietolähteiden käsittelyyn

Informaatioteknologian ja viestinnän tiedekunta
Kandidaattitutkielma
Toukokuu 2021

TIIVISTELMÄ

Elisa Akkanen: Dataintegraatiometodit SQL- ja NoSQL-tietokannoissa - Ratkaisuja hybriditietolähteiden käsittelyyn
Kandidaattitutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Toukokuu 2021

Tässä tutkielmassa tarkastellaan hybriditietolähteiden välisen dataintegraation metodeja. Tarkastellut metodit kohdistuvat SQL (Structured Query Language) -kyselykieleen pohjautuvien relaatiotietokantojen sekä NoSQL (Not only SQL) -tietokantojen välille. SQL-tietokantojen tarkkaan määritellystä rakenteesta poiketen NoSQL-tietokannat ovat rakenteellisesti todella monimuotoisia. Tästä syystä SQL- ja NoSQL-järjestelmät eivät ole lähtökohtaisesti yhteensopivia, ja niiden yhtäaikainen käyttö on haastavaa. Tiedon tarve voi kuitenkin kohdistua yhtä aikaa sekä SQL- että NoSQL-järjestelmien sisältämiin tietoihin. Dataintegraatiolla pyritään yhdistämään tietoa erillisistä tietolähteistä ja ratkomaan yhteensopivuusongelmia erilaisten tietolähteiden välillä.

Tämä tutkielma on muodoltaan kirjallisuuskatsaus. Tutkielmassa tarkastellut dataintegraatiometodit on alun perin esitelty vertaisarvioiduissa tieteellisissä artikkeleissa tai konferenssijulkaisuissa. Tutkielmaan valikoitiin pääosin metodeja, jotka esiintyivät viimeisen viiden vuoden sisällä julkaistussa kirjallisuudessa. Työn ulkopuolelle jätettiin useita dataintegraatiometodeja, joissa esimerkiksi julkaisuartikkelin fokus tai julkaisuvuosi ei täsmännyt tämän tutkielman päämäärien kanssa.

Tutkielmassa avataan ensin SQL- ja NoSQL-tietokantojen välisen dataintegraation merkitystä, jonka jälkeen varsinaisena tutkimuskysymyksenä pyritään selvittämään viime vuosina julkaistussa tieteellisessä kirjallisuudessa esitettyjä ratkaisumalleja. Dataintegraatiota voidaan lähestyä monella tapaa. Eräs tämän tutkielman kiinnostuksenkohteista on tarkastella sitä, onko nykypäivän dataintegraatiometodeissa huomattavissa suurta monimuotoisuutta näiden lähestymistapojen suhteen. Tässä tutkielmassa tarkastellaan viittä hybriditietokantojen käsittelyyn kehitettyä metodia. Monissa tarkastelluissa metodeissa fokus on yhtäaikaisten kyselyjen mahdollistamisessa SQL-kyselykieltä käyttäen, mutta lähteisiin sisältyy myös tästä asetelmasta poikkeavia metodeja.

Lähteiden perusteella havaittiin, että nykypäivän kiinnostus SQL- ja NoSQL-tietokantojen välisessä dataintegraatiossa kohdistuu vahvasti hybriditietokantoihin ja mahdollisuuteen hyödyntää yhtäaikaisesti sekä relaatio- että NoSQL-tietokantajärjestelmiä. Saatavilla oleva data on monimuotoistunut sekä sisällöltään että rakenteeltaan. Datan säilöminen erilaisiin tietokantoihin juuri kyseisen datan vaatimien rakenteiden ja ominaisuuksien perusteella nähdään suurena etuna. Tällaisen tietojärjestelmien heterogeenisyyden aiheuttama haittapuoli on kuitenkin tietolähteiden välisen vuorovaikutuksen ja lähteiden yhtäaikaisen käytön vaikeus. Näihin haasteisiin on viime aikoina pyritty vastaamaan metodein ja arkkitehtuurein, jotka mahdollistavat yhtäaikaiset kyselyt hybriditietolähteisiin ja muodostavat niistä yhtenäiset kyselytulokset.

Avainsanat: dataintegraatio, SQL, NoSQL, tietokantajärjestelmät, hybriditietokannat

Sisällysluettelo

1 Johdanto 1

2 Lyhenteet ja termit 3

3 SQL- ja NoSQL-tietokannat 4

4 Dataintegraatio SQL- ja NoSQL-järjestelmien välillä 6

5 Integraatiometodit 7

5.1 UnityJDBC-virtualisointijärjestelmä 8

5.2 MSI-dataintegraatiomalli 8

5.3 Hybriditietokanta-arkkitehtuuri 9

5.4 NoSQL-tietokannan relaatiotietokannaksi muuntava metodi 11

5.5 HybridDB-dataintegraatiometodi 12

6 Metodien vertailu 14

6.1 Päämäärät 14

6.2 Arkkitehtuuri 15

6.3 Lähestymistapojen analyysi 19

7 Pohdinta 20

8 Yhteenveto 22

Lähdeluettelo 23

1 Johdanto

Relaatiotietokannat ovat tietokantajärjestelmien keskuudessa valta-asemassa. Viime vuosina NoSQL (Not only SQL) -tietokantojen käyttö ja tarjonta ovat kuitenkin lisääntyneet. Big Datan ja pilvipalveluympäristöjen kasvun ja kysynnän johdosta NoSQL on kasvattanut suosiotaan tehokkaampana vaihtoehtona suurten datamäärien talletukseen ja käsittelyyn. Samalla on ollut huomattavissa lisääntynyt kysyntä dataintegraatiotyökaluille. (Aftab ja muut, 2020) Myös NoSQL- ja SQL (Structured Query Language) -järjestelmien väliseen integraation keskittyvää tutkimusta on julkaistu paljon ja aihe on edellä mainituista syistä hyvin ajankohtainen (Bjeladinovic ja muut, 2020; Aftab ja muut, 2020; Li & Gu, 2019).

Tämän tutkielman pääasiallisena tavoitteena on tarkastella ja vertailla moderneja menetelmiä SQL- ja NoSQL-tietokantojen väliseen dataintegraatio-ongelmaan. Tutkimuskysymyksenä on ”millaisia SQL- ja NoSQL-tietokantojen välisen dataintegraation menetelmiä viime vuosien tieteellisessä kirjallisuudessa on esitetty”. Aihetta lähestytään tarkastelemalla viimeaikaisia kehityssuuntia ja esittelemällä ehdotettuja ratkaisumalleja. Tästä syystä viittä vuotta vanhemmat menetelmät ja metodit eivät lähtökohtaisesti ole tämän tutkielman kannalta oleellisia.

Tutkielman alkupuolella on oleellista selventää syitä SQL- ja NoSQL-tietokantojen välisen dataintegraation tarpeelle. Tarkoitus on saada vastauksia siihen, miksi molemmat järjestelmät ovat rinnakkaisessa käytössä ja miksi datan liikuttaminen niiden välillä ei ole triviaali ongelma.

Tutkielman lähteitä etsittiin Andorista, ScienceDirectistä (Elsevier) ja IEEE Electronic Librarysta. Tärkeimpiä yksittäisiä hakutermejä olivat “data integration”, “NoSQL”, “SQL” ja “relational database”. Haku keskittyi artikkeleihin ja konferenssijulkaisuihin, ja tuloksista rajattiin aluksi pois ennen vuotta 2017 julkaistut työt, sillä työssä haluttiin keskittyä dataintegraation nykyiseen kehityssuuntaan. Myöhemmin hakua jatkettiin myös ennen vuotta 2017 ilmestyneiden artikkelien joukosta sekä tarkastelemalla jo löydettyjen lähteiden sisältöä ja lähdeluetteloja. Monissa aihetta käsittelevissä artikkeleissa oli oma osionsa “related work”, joka havainnollisti aihepiiristä meneillään olevaa keskustelua ja tutkijoiden välistä vuorovaikutusta.

On huomioitava, että tämän tutkielman ulkopuolelle jätettiin useita artikkeleita, joissa esiteltiin jokin SQL- ja NoSQL-tietokantojen välisen dataintegraation metodi (Solanke & Rajeswari, 2017; Zhao ja muut, 2014). Tutkielman sisäisten rajoitteiden lisäksi artikkelien valintaan vaikuttivat niiden julkaisuvuosi, Julkaisufoorumin (JuFo, 2021) taso, artikkelin pituus ja fokus, sekä Scopuksesta tarkistettu artikkeleihin kohdistuvien ulkopuolisten viittausten määrä. Tutkielmassa esitellyissä metodeissa painopisteen havaittiin olevan ratkaisuissa, joissa NoSQL-rakenteissa pyrittiin hyödyntämään SQL-kieltä tai relaatiomallia (Lawrence, 2014; Bjeladinovic ja muut, 2020; Li & Gu, 2019; Vathy-Fogarassy & Hugyák, 2017; Aftab ja muut, 2020). Tutkielman ulkopuolelle jääneessä kirjallisuudessa esiintyy myös metodeja, joissa tietoa siirretään tai mukautetaan vastakkaiseen suuntaan, SQL-tietokannasta NoSQL-tietokantaan (Solanke & Rajeswari, 2017; Zhao ja muut, 2014). Tällaisten metodien ja niitä käsittelevän kirjallisuuden olemassaolo on huomioitu tutkielman johtopäätöksissä.

Tutkielmassa löydettiin useampia dataintegraatioratkaisuja, joissa mahdollistetaan SQL-kieliset kyselyt NoSQL-pohjaisiin tietokantajärjestelmiin. Tällaisia olivat Unity (Lawrence, 2014), MSI-malli (Multiple Sources Integration) (Li & Gu, 2019) sekä Bjeladinovicin ja muiden (2020) SQL/NoSQL-hybriditietokanta-arkkitehtuuri. Kaikki kolme ovat järjestelmiä, jotka kääntävät SQL-kyselyt NoSQL-kohdejärjestelmän vaatimaan muotoon ja yhdistävät yksittäisen kyselyn koskemaan useita hybriditietokantoja. Aftabin ja muiden (2020) ehdottama ratkaisu puolestaan muuntaa kohteena olevan NoSQL-tietokannan relaatiotietokannaksi. Viides ja viimeinen tutkielmaan sisällytetty metodi on JSON-objekteja (JavaScript Object Notation) kyselyn muodostukseen käyttävä, sekä SQL- että NoSQL-pohjaisiin tietokantoihin yhtäaikaisen kyselyn mahdollistava HybridDB-metodi (Vathy-Fogarassy & Hugyák, 2017). Lähes kaikille ratkaisuille yhteistä oli se, että ne tarjosivat jollakin tavalla kyvyn hyödyntää SQL-kyselykieltä myös tietokannoissa, jotka eivät sitä lähtökohtaisesti tue. Poikkeuksen tähän muodosti HybridDB-metodi (Vathy-Fogarassy & Hugyák, 2017).

Tutkielman luvussa kaksi määritellään työssä käytettävää termistöä. Luvussa kolme kuvataan SQL- ja NoSQL-järjestelmien eroja ja luvussa neljä selvitetään, mitä haasteita niiden väliseen dataintegraatioon sisältyy. Viidennessä luvussa esitellään lähdekirjallisuudessa ehdotettuja integraatiometodeja, joita vertaillaan keskenään kuudennessa luvussa. Seitsemäs luku on löydöksiin pohjautuvaa pohdintaa, ja kahdeksas luku sisältää tutkielman yhteenvedon.

2 Lyhenteet ja termit

ACID-periaate: Akronyymi ACID (Atomicity, Consistency, Isolation and Durability) vastaa suomen kielellä sanoja atomisuus, eheys, eristyneisyys ja pysyvyys. Periaate vaatii tietokannoissa tapahtuvien transaktioiden täyttävän mainitut neljä ominaisuutta, jotta tietokanta pysyy eheänä siinä tapahtuvista muutoksista huolimatta. (Techopedia, 2021)

ETL (Extract, Transform, Load): ETL-prosesseja käytetään siirtämään dataa lähteestä kohteeseen, kun siirto vaatii datan läpikäyvän rakenteellisia tai sisällöllisiä muunnoksia. Esimerkiksi yhteensopimattoman tyyppisten tietokantojen välillä liikkuvan datan siirtoon voidaan käyttää ETL-prosesseja. (Techopedia, 2021; Aftab ja muut, 2020)

Hybriditietolähteet: Tietolähteet, joiden rakenteellisuus tai tiedon organisointitavat eroavat toisistaan. Tällaiseksi luetaan esimerkiksi yhdistelmä SQL- ja NoSQL-tietokantoja. (Li & Gu, 2019; Vathy-Fogarassy & Hugyák, 2017)

JDBC (Java Database Connectivity): Java-ohjelmointikielelle kehitetty rajapinta, joka mahdollistaa yhteydenoton tietokantaan. (Techopedia, 2021)

NoSQL (Not only SQL): Tietokantajärjestelmä, joka ei pohjaudu relaatiomalliin. Tietorakenteet eri NoSQL-järjestelmien välillä eroavat suuresti. NoSQL-tietokannat eivät usein tue SQL-kyselykieltä (Aftab ja muut, 2020). Tässä tutkielmassa usein mainittu esimerkki NoSQL-järjestelmästä on dokumenttipohjainen **MongoDB**-tietojärjestelmä.

ODBC (Open Database Connectivity): Tietokanta- ja käyttöjärjestelmistä riippumaton rajapinta tietokannanhallintajärjestelmiin. (Techopedia, 2021)

Skeeman sovittaminen (Schema matching): Joskus termiä käytetään synonyyminä termille “schema mapping”, mutta toisinaan nämä kaksi myös erotetaan toisistaan. Tässä työssä termillä viitataan dataintegraatiomenetelmään, jossa pyritään löytämään merkityksiltään toisiaan vastaavat tiedot ja muuntamaan ne yhdenmukaiseen malliin. (Aftab ja muut, 2020; Kementsietsidis, 2009)

SQL (Structured Query Language): Relaatiotietokannoissa standardiksi muodostunut kyselykieli (Lawrence, 2014). Tässä tutkielmassa käytetään myös termiä “SQL-tietokanta” viittaamaan relaatiotietokantoihin, joiden natiivi kyselykieli SQL on.

Tietolähde: Varasto, joka sisältää tietoa jossain muodossa. Tietolähde voi olla rakenteellinen, kuten relaatiotietokanta tai Excel-taulukko, tai rakenteeton, kuten esimerkiksi tekstitiedosto. (Techopedia, 2021; Lawrence, 2014)

XML (Extensible Markup Language) -tietokanta: Tietokantatyyppi, joka sallii datan määrittelyn XML-kielisenä. Niin sanotut natiivit XML-tietokannat kuuluvat dokumenttipohjaisiin tietokantoihin, joka puolestaan on eräs NoSQL-tietokantatyyppi. (Techopedia, 2021)

3 SQL- ja NoSQL-tietokannat

SQL- ja NoSQL-tietokantojen erilaiset vahvuudet ja heikkoudet, sekä niiden eriävä asema organisaatioiden tietokantajärjestelminä nykypäivänä ovat avainasemassa selittämään, miksi dataintegraatio kyseisten järjestelmien välillä on tarpeellista. Toinen järjestelmä ei ole syrjäyttänyt toista, vaan niiden rinnakkaiselo tulevaisuudessakin on todennäköistä (Lawrence, 2014; Doncevic & Fertalj, 2020).

Tietokantajärjestelmät ovat vuosia pohjautuneet pääasiassa relaatiomalliin, johon suoritetaan kyselyjä SQL-kielellä. Tämä on vakiinnuttanut relaatiotietokantojen, ja samalla SQL-kyselykielen, aseman toimialan yleisenä standardina. Molemmat omaavat hyvin laajan ja kokeneen käyttäjäkunnan, mikä osaltaan vaikeuttaa uudenlaisten tietokantajärjestelmien ja mallien omaksumista. Relaatiotietokantajärjestelmissä on pieniä eroavaisuuksia, mutta ne ovat pitkälti keskenään yhteensopivia. Toisin sanoen yhdelle relaatiotietokantajärjestelmälle kirjoitetut applikaatiot voidaan verrattain helposti siirtää myös muihin relaatiomalliin pohjautuviin järjestelmiin. (Lawrence, 2014)

Relaatiotietokantojen välinen yhteensopivuus ei kuitenkaan aina ole itsestäänselvyys. Siirrettävyyttä voivat heikentää esimerkiksi maksulliset liitännäiset, jotka saattavat olla käytössä vain osassa järjestelmiä. (Kuchibhotla ja muut, 2009)

NoSQL-tietokantoihin kuuluvat objektitietokannat ja XML-tietokannat ovat osittain haastaneet relaatiotietokantojen valta-asemaa. Tämän seurauksena relaatiojärjestelmään on lisätty ominaisuuksia, joiden avulla se on kyennyt saavuttamaan suurimman osan sekä XML-tietokantojen että objektitietokantojen eduista. Samalla on säilytetty kuitenkin SQL:lle ja relaatiotietokannoille ominaiset vahvuudet, kuten siirrettävyys ja ACID

(Atomicity, Consistency, Isolation and Durability) -periaatteiden noudattaminen. (Lawrence, 2014)

NoSQL-järjestelmät on kehitetty tukemaan erilaisia applikaatioita ja järjestelmiä, joihin relaatiotietokantamalli ei täysin sovellu (Li & Gu, 2019). NoSQL täyttää sellaisten järjestelmien tarpeita, joilla on tarkat ja erityiset vaatimukset tiedon muodon, organisoinnin ja käsittelyn suhteen. Koska vaatimukset eroavat ja saattavat olla hyvin applikaatio-spesifisiä, myös NoSQL-järjestelmät eroavat merkittävästi toisistaan. NoSQL-järjestelmiä on olemassa yli viisikymmentä erilaista. Tämä heterogeenisyys on ongelma, mikäli on tarve tehdä siirtoja tai integraatiota eri NoSQL-järjestelmien välillä. (Atzeni ja muut, 2020)

Vaikka SQL-järjestelmillä on vakiintunut käyttäjäkunta, NoSQL voi olla tietynlaisiin applikaatioihin huomattavasti tehokkaampi ratkaisu. Esimerkiksi lääketieteessä potilaskohorttien löytämiseen suunnitelluista työkaluista suurin osa pohjautuu SQL:ään, mutta SQL:n rajoittunut kyky käsitellä suuria datamääriä voi muodostua ongelmaksi. Potilasdatan vaatima tila saattaa ylittää yksittäisessä taulussa SQL:n asettaman kolumnien maksimimäärän, jolloin data täytyy jakaa useampaan tauluun. Tämä vaikuttaa kyselyjen prosessointinopeuteen. Potilaskohorttidataan kohdistettujen kyselyjen evaluoinnissa käytetyt NoSQL-järjestelmät (MongoDB ja Cassandra) suoriutuivat käytettyä relaatiotietokantajärjestelmää (MySQL) paremmin. Havaittu ero kyselyjen prosessointinopeuksissa oli tilastollisesti merkittävä. (Zeng ja muut, 2017)

NoSQL-järjestelmien eduiksi luetaan muun muassa datan luku- ja kirjoitusnopeudet sekä kyky säilöä suuria datamääriä (Li & Gu, 2019). NoSQL-järjestelmät kykenevät parempaan suorituskykyyn suurten datamäärien käsittelyssä verrattuna SQL-järjestelmiin (Lawrence, 2014). Myös järjestelmien skaalautuvuus mainitaan NoSQL:n vahvuutena (Lawrence, 2014; Aftab ja muut, 2019). NoSQL-järjestelmiä luonnehtii niiden rakenteellinen joustavuus, joka erottaa kyseiset järjestelmät jäykän tietorakenteen omaavista relaatiotietokannoista (Lawrence, 2014; Atzeni ja muut, 2020). Pääosin NoSQL-tietokannat eivät kuitenkaan toteuta ACID-periaatteita transaktioiden atomisuudesta, eheydestä, eristyneisyydestä ja pysyvyydestä (Li & Gu, 2019). ACID-periaatteiden tarkoitus on tehdä tietokantojen transaktioista vakaita niin, että kantaan kohdistuvat muutokset eivät pääse korruptoimaan säilytettävää dataa (Techopedia, 2021). Relaatiotietokannat puolestaan toteuttavat lähtökohtaisesti ACID-periaatteet, jonka

vuoksi datan yhdenmukaisuus ja SQL-kieliset transaktiomekanismit luetaankin relaatiotietokantojen vahvuuksiksi (Doncevic & Fertalj, 2020).

4 Dataintegraatio SQL- ja NoSQL-järjestelmien välillä

Dataintegraation perimmäinen tarkoitus on yhdistää dataa useista eri lähteistä (Doan ja muut, 2012; Lenzerini, 2002). Datan erilaiset säilöntätavat tietotyyppien, datan organisoinnin, semantiikan tai ohjelman ja rajapinnan rakenteissa tekevät integroinnista haastavaa (Doan ja muut, 2012). Dataintegraatiolla pyritään tarjoamaan yhdenmukainen pääsy kokoelmaan itsenäisiä ja heterogeenisiä tietolähteitä (Doan ja muut, 2012; Lenzerini, 2002). Organisaatioille nopea ja saumaton pääsy useista eri tietolähteistä yhdistettyyn dataan on kilpailukyvyn kannalta elintärkeää (Kuchibhotla ja muut, 2009). Suurin osa tässä tutkielmassa tarkastelluista dataintegraatiojärjestelmistä tähtää yhtäaikaisten kyselyjen mahdollistamiseen useista eri lähteistä.

Dataintegraatiossa on tyypillisesti mukana tietolähteitä, jotka on kehitetty toisistaan erillisinä kokonaisuuksina. Usein tarve yhtenäistymiselle on ilmaantunut vasta lähteiden luomisen jälkeen, sillä tietokantaa luodessa ei ole mahdollista ennustaa kaikkia tulevaisuuden tarpeita kyseiselle järjestelmälle ja sen sisältämälle tiedolle. Tämän seurauksena eri tietolähteet käyttävät erilaisia järjestelmiä. Osa lähteistä voi olla rakenteeltaan vankkoja, kuten relaatiotietokannat, kun taas osa voi olla lähes tai täysin rakenteettomia esimerkiksi puhtaan tekstin muodossa. Myös käytetyt skeemat ja tiedon mallinnustapa voivat olla erilaisia, vaikka lähteet käsittelisivätkin samaa aihetta. (Doan ja muut, 2012)

NoSQL-järjestelmiä luonnehtiva suuri heterogeenisyys hankaloittaa niiden välistä dataintegraatiota (Atzeni ja muut, 2020). Sekä Lawrence (2014) että Li ja Gu (2019) mainitsevat järjestelmäriippumattoman kyselykielen ja API:n (Application Programming Interface) puutteen NoSQL-kantojen ongelmiksi. Usein NoSQL-järjestelmistä ei myöskään löydy tukea SQL-kyselykielen käytölle (Li & Gu, 2019; Lawrence, 2014). NoSQL-järjestelmät vaativat ohjelmointitasolla järjestelmälle spesifistä koodia ja käyttävät keskenään erilaisia kyselykieliä ja rajapintoja (Li & Gu, 2019; Lawrence, 2014). Li ja Gu (2019) mainitsevat näiden seikkojen johtavan siirrettävyyden puutteeseen järjestelmien välillä.

SQL- ja NoSQL-kantojen kaksi pääasiallista erottavaa tekijää ovat tietomallien rakenteet sekä niiden kyselykieli ja sen tukemat operaatiot. Nämä tekijät ovat syynä myös siihen, miksi integraatiota ja yhtäaikaista käyttöä näiden järjestelmien välille tarvitaan. (Bjeladinovic ja muut, 2020) Aftabin ja muiden (2020) mukaan saatavilla on monia työkaluja, jotka muuntavat dataa SQL-tietokannoista NoSQL-yhteensopivaan formaattiin, mutta päinvastaista muutosta tukevat työkalut ovat harvinaisempia.

Muunnos NoSQL-formaatista verrattain jäykkään relaatiomalliin on suurempi haaste, kuin muunnos relaatiomallista erilaisiin NoSQL-kantoihin. Tämä ero johtuu NoSQL-tietokantojen skeeman puutteesta. Relaatiotietokannoissa data noudattaa hyvin tarkkaa mallia, joten ennen relaatiotietokantaan siirtämistä NoSQL-tietokannoista tuleva data täytyy mallintaa oikean muotoiseksi. (Aftab ja muut, 2020; Atzeni ja muut, 2020)

5 Integraatiometodit

Tässä luvussa esitellään viime vuosien julkaisujen joukosta poimittuja SQL- ja NoSQL-kantojen väliseen dataintegraatioon esitettyjä metodeja. Metodeista kuvataan niiden toiminnallisuuksia ja arkkitehtuuria. Kahden ensimmäisen metodin (Unity ja MSI) arkkitehtuuria kuvaillaan tarkemmin vasta luvussa kuusi, sillä niiden arkkitehtuurit ovat tarpeeksi samankaltaisia, jotta niitä voidaan verrata suoraan toisiinsa.

Tutkielmaan valitut metodit valikoituivat seuraavien kriteerien pohjalta. Tärkein sisällyttämiskriteeri oli se, että metodi suorittaa SQL- ja NoSQL-tietokantojen välisen dataintegraation. Koska tutkielma on kirjallisuuskatsaus, metodit tuli olla esitelty vertaisarvioidussa tieteellisessä artikkelissa tai konferenssijulkaisussa. Koska fokus on viimevuotisissa ratkaisuissa, metodin esittelemän julkaisun tuli olla peräisin vuodelta 2017 tai myöhemmin. Poikkeukseksi muodostui UnityJDBC-järjestelmää käsittelevä julkaisu (Lawrence, 2014). Kyseessä on yhä päivittyvä järjestelmä, johon liittyen on kirjoitettu tuoreempiakin julkaisuja (Lawrence, 2017). Lawrencen julkaisuun viitattiin myös muissa tähän tutkielmaan valikoituneissa töissä (Bjeladinovic ja muut, 2020; Li & Gu, 2019; Vathy-Fogarassy & Hugyák, 2017). Näiden seikkojen myötä todettiin perustelluksi sisällyttää kyseinen järjestelmä tutkielmaan.

5.1 UnityJDBC-virtualisointijärjestelmä

Unity-arkkitehtuuri sallii käyttäjän sekä suorittaa SQL-kyselyjä että yhdistää dataa niin NoSQL- kuin SQL-järjestelmistäkin yksittäisen SQL-kyselyn kautta. Unity, julkaistu myöhemmin nimellä UnityJDBC, on virtualisaatiojärjestelmä, joka kääntää SQL-kyselyt kohdejärjestelmän vaatimalle API:lle. NoSQL-tietokannoista Unity tukee Cassandraa ja MongoDB:tä. (Lawrence, 2014; UnityJDBC, 2021)

Relaatiojärjestelmien välistä integraatiota on aiemmin työstetty skeeman sovittamisen (schema matching) menetelmin ja Lawrencen (2014) mukaan samoja konsepteja voidaan hyödyntää myös NoSQL-järjestelmien integraatiossa. SQL-kyselyjen hyödyllisyyteen NoSQL-järjestelmissä on kolme syytä. Ensinnäkin SQL:n syntaksi pohjautuu vahvasti luonnolliseen kieleen ja kuvaa siitä syystä hyvin kyselyjen toimintaa antamatta käyttäjälle kuitenkaan liikaa tietoa kyselyn teknisestä implementaatiosta ja suorittamisesta. (Lawrence, 2014) Toiseksi ja tärkeimmäksi syyksi Lawrence (2014) mainitsee SQL:n aseman standardoituna kielenä. Tämä mahdollistaa siirrettävyyden järjestelmien välillä. Kolmanneksi syyksi esitetään, että SQL-tuen avulla NoSQL-järjestelmät voivat olla saumattomasti vuorovaikutuksessa muiden SQL:ää, JDBC:tä (Java Database Connectivity) ja ODBC:tä (Open Database Connectivity) käyttävien järjestelmien kanssa. (Lawrence, 2014)

Unityn toimivuus on Lawrencen alkuperäisen julkaisun (2014) jälkeen laajentunut ja tehostunut (Lawrence, 2017; UnityJDBC, 2021). Unityn verkkosivujen (2021) mukaan järjestelmää päivitetään kuukausittain, ja vuosittain ohjelmistoon tulee myös suurempi päivitys. Unitya on muunnettu tukemaan monipuolisemmin eri taulujen tietoja yhdistäviä join-kyselyoperaatioita, ja metodin suorittamaa kyselyn optimointivaihetta on laajennettu myös tämän pohjalta (Lawrence, 2017; UnityJDBC, 2021). Testaus antoi viitteitä siihen, että nämä muutokset nopeuttivat kyselyjen prosessointia ja mahdollistivat aiempaa laajempien kyselyjen suorittamisen Unity-järjestelmällä (Lawrence, 2017).

5.2 MSI-dataintegraatiomalli

SQL-kyselyt mahdollistavia tekniikoita on kehitelty useisiin eri NoSQL-tietokantoihin. Nämä ovat kuitenkin NoSQL-järjestelmien heterogeenisyyden vuoksi usein keskittyneitä vain yksittäiseen kyselykieleen tai API:in, jonka vuoksi ne toimivat ainoastaan kohteeksi

valitussa NoSQL-tietokannassa. Kyselyn suorittaminen yhtäaikaisesti moniin hybrididatalähteisiin on jäänyt vähäiselle huomiolle. (Li & Gu, 2019) Tämän perusteella Li ja Gu (2019) esittävät, että tutkimus integraatio-ongelmaan on ollut vajavaista. Joihinkin NoSQL-kantoihin yhtäaikaisten kyselyjen rajapinnan rakentaminen on verrattain helppoa, koska ne sisältävät jo valmiiksi ominaisuuksia, joilla voidaan suodattaa ja järjestää rivitietoja useiden eri ehtojen mukaisesti. Toisissa taas vastaavia ominaisuuksia ei ole, jolloin ne täytyy implementoida ensin. (Li & Gu, 2019)

Multiple Sources Integration- eli MSI-malli esitetään ratkaisuna, joka toteuttaa yksittäisen kyselyn useisiin hybridisiin tietokanta-arkkitehtuureihin. MSI voi käsitellä dataa sekä relaatiotietokannoista että NoSQL-tietokannoista. Mahdollisina kohdejärjestelminä mainitaan NoSQL-tietokannat MongoDB ja Redi, sekä SQL-kanta MySQL, joita käytettiin myös evaluoimaan ehdotettua järjestelmää. (Li & Gu, 2019)

Li ja Gu (2019) vertasivat Lawrencen (2014) Unityä ja itse kehittämäänsä MSI-mallia toisiinsa. Unity valikoitui vertailukohdaksi muiden integraatiotyökalujen joukosta siksi, että se omasi samankaltaisia komponentteja, kuin MSI (Li & Gu, 2019). Molemmat käyttävät SQL-kyselykieltä ja mahdollistavat yhtäaikaisen pääsyn useisiin hybriditietolähteisiin (Li & Gu, 2019; Lawrence, 2014).

5.3 Hybriditietokanta-arkkitehtuuri

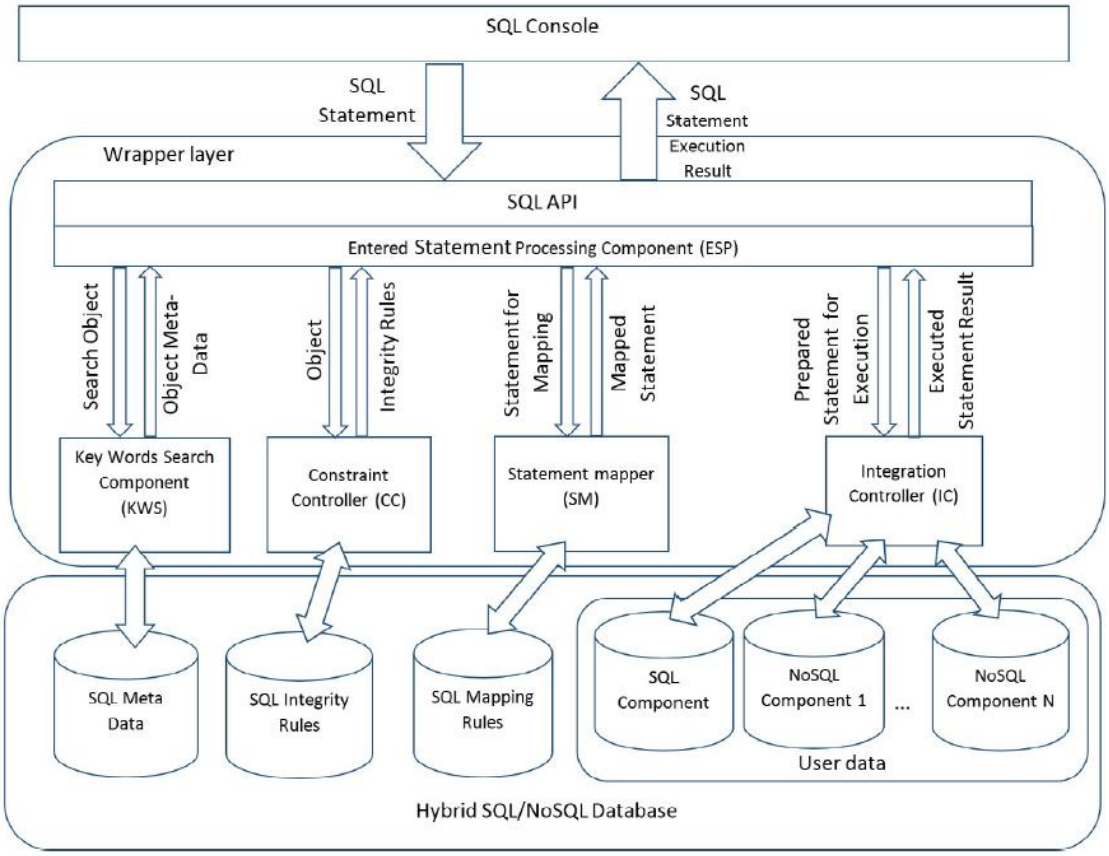
Useista heterogeenisistä tietokannoista koostuvan hybriditietolähteen yhtäaikaiseen, suoraviivaiseen käyttöön keskittyvät myös Bjeladinovic ja muut (2020). He esittelevät arkkitehtuurin, jonka tarkoitus on tarjota hybriditietolähteisiin käyttäjän näkökulmasta yhtä yksinkertainen käyttöliittymä kuin jos tämä suorittaisi kyselyjä vain yksittäiseen tietokantaan (Bjeladinovic ja muut, 2020). Esitelty menetelmä poikkeaa Bjeladinovicin ja muiden (2020) mukaan aiemmassa kirjallisuudessa esitetyistä integraatiomenetelmistä arkkitehtuurina, joka tarjoaa integraatiota ja yhtäaikaista käyttöä tietokannoille, jotka toimivat komponentteina SQL/NoSQL-hybriditietokannassa.

Lähdetietokantojen säilyttäminen heterogeenisinä mahdollistaa rakenteellisuudeltaan erilaisten tietojen säilömisen niille itselleen optimaalisissa tietokantatyypeissä. Tämä on integraatiojärjestelmälle positiivinen ominaispiirre. Erilaiset tietokantamallit ovat tarpeen voidakseen tukea rakenteellisuudeltaan erilaista tietoa. Vähemmän rakenteellista tietoa

syntyy nykyään muun muassa sosiaalisen median kautta, ja jo tiedon suuri määrä itsessään vaatii tietokannoilta joustavuutta. (Bjeladinovic ja muut, 2020)

Esitetty arkkitehtuuri mahdollistaa paitsi hybridikomponenttiensa datan yhdistämisen, myös kaikille lähdetietokannoille yhtenäisen, keskitetyn hallintajärjestelmän. Metodia testattiin prototyypillä, joka ei sisältänyt vielä kaikkia haluttuja toiminnallisuuksia. Metodi suoriutui testauksesta kuitenkin tehokkaasti. SQL-tietokantana testauksen yhteydessä käytettiin Oraclea ja NoSQL-kantana MongoDB:tä. Artikkelissa perustellaan kyseisten tietokantojen valintoja sillä, että ne ovat nykyajan käytetyimmät SQL- ja NoSQL-tietokantajärjestelmät. (Bjeladinovic ja muut, 2020)

Arkkitehtuurin rakennetta on kuvattu artikkelissa hyvin kattavasti ja sitä on havainnollistettu graafisesti kuvassa 1. SQL API -komponentti vastaanottaa SQL-lausekkeita ja toimii metodin alusta loppuun koko metodin toimeenpanoa kontrolloivana komponenttina. Samalle tasolle sen kanssa sisältyy useita integraation eri vaiheista vastaavia komponentteja. Entered Statement Processing -komponentti ottaa SQL API:n vastaanottaman kyselylausekkeen ensimmäisenä käsittelyyn ja analysoi sekä sen, että kyselyn lähdetietokannan. Key Word Search -komponentti hyödyntää metadatakirjastoa muokkaamaan kyselyä lähdetietokannan tyyppiin ja talletetun tiedon rakenteisiin sopivaksi. Constraint Controller -komponentti huolehtii mahdollisten rajoitteiden ja vaadittavien liitosoperaatioiden ehtojen tarkistamisesta. Statement Mapper -komponentti kääntää kyselyn juuri tietyn tietokannanhallintajärjestelmän ymmärtämälle kielelle ja syntaksille. Lopulta Integration Controller -komponentti ottaa yhteyden hybriditietokannan eri osiin ja suorittaa kyselyt niihin. Kun kyselyt on suoritettu, Integration Controller -komponentti lähettää palautetut tulokset takaisin Entered Statement Processing -komponentin analysoitavaksi. (Bjeladinovic ja muut, 2020)



Kuva 1. Hybriditietokanta-arkkitehtuurin rakenne (Bjeladinovic ja muut, 2020)

5.4 NoSQL-tietokannan relaatiotietokannaksi muuntava metodi

Vaikka NoSQL-tietokannat tukevat paremmin määrällisesti suuren ja rakenteellisesti monimuotoisen datan säilömistä, monet organisaatiot haluavat siirtää datansa SQL-pohjaisiin relaatiokantoihin voidakseen hyödyntää jo olemassa olevia työkaluja tiedon analysoinnissa (Aftab ja muut, 2020; Kuchibhotla ja muut, 2009). Aftab ja muut (2020) kehittivät tähän tarpeeseen soveltuvan metodin, joka muuntaa NoSQL-tietokannan relaatiotietokannaksi automaattisesti.

Muunnettaessa dataa NoSQL-tietokannasta relaatiomalliin on huomioitava tietovarastojen rakenteelliset erot. Relaatiotietokantaan talletettavan tiedon tulee noudattaa rakenteeltaan relaatiomalliin sopivaa skeemaa. NoSQL-kantoihin talletetulla tiedolla ei yleensä lähtökohtaisesti ole relaatiomallin kanssa yhteensopivaa skeemaa. Usein NoSQL-tietokantojen yhteydessä puhutaankin skeemattomasta datasta. (Doncevic ja Fertalj, 2020; Aftab ja muut, 2020)

Ennen tiedon siirtoa sopiva skeema täytyy ensin luoda, ja vasta sen jälkeen sen voi täyttää datalla. Manuaalisena prosessina skeeman luonti vaatii tekijältä asiantuntijuutta paitsi mallinnettavan tiedon erityisosa-alueelta, myös lähde- ja kohdetietokannoista. Tästä syystä lähdedataan soveltuvan skeeman tunnistaminen on ollut perinteisesti haastava, aikaa vievä ja virhealtis prosessi, joka on näin ollen usein myös kallista organisaatioille. (Aftab ja muut, 2020)

Aftabin ja muiden (2020) metodi automatisoi skeeman tunnistamisen prosessin. Metodi tunnistaa tarvittavaa skeemaa dynaamisesti samalla, kun se hakee dataa lähdetietokannasta. Kun tiedon haku lähdetietokannasta on kokonaisuudessaan suoritettu, metodi muuntaa datan kohdeformaattiin ja lataa datan kohdetietokantaan erissä. (Aftab ja muut, 2020)

Ehdotettua metodia evaluoitiin vertailemalla sen tehokkuutta TOS (Talend Open Studio) -työkaluun. TOS valittiin vertailukohdaksi koska se on uusi, ilmainen, avoimeen lähdekoodiin perustuva työkalu, joka on yksi laajimmin käyttöön otetuista ja menestyneimmistä dataintegraatiotyökaluista. Suurin ero vertailtujen työkalujen toiminnallisuuksissa oli skeeman tunnistuksen automatisoinnissa. Siinä missä TOS tarvitsee käyttäjän konfiguroimaan skeeman tunnistukseen liittyviä parametrejä saavuttaakseen onnistuneen lopputuloksen, Aftabin ja muiden (2020) ehdottama metodi on saman vaiheen suhteen täysin automatisoitu. Evaluoinnissa kävi ilmi, että pääpiirteittäin ehdotettu metodi suoriutui integraatioista merkittävästi TOS:ia nopeammin. (Aftab ja muut, 2020)

Metodien evaluoinnissa käytettiin lähdetietokantana NoSQL-pohjaista MongoDB:tä ja kohdetietokantoina SQL-mallisia MySQL- ja PostgreSQL-tietokantoja (Aftab ja muut, 2020). Aftabin ja muiden (2020) mukaan ehdotettu metodi on tarpeeksi geneerinen tukeakseen mitä tahansa relaatiomalliin perustuvaa kohdetietokantaa. Metodi on myös helppokäyttöinen, eikä vaadi käyttäjältään erityisosaamista dataintegraation osalta (Aftab ja muut, 2020).

5.5 HybridDB-dataintegraatiometodi

HybridDB-metodi on dataintegraatioväline, joka mahdollistaa itsenäiset kyselyt erilaisiin SQL- ja NoSQL-tietokantajärjestelmiin. Metodi pyrkii tarjoamaan yhtenäisen, loogisen näkymän dataan, joka sijaitsee fyysisesti useissa eri tietokannoissa. Liitos- ja joukko-

operaatioita eri tietolähteiden välillä ei tueta, eikä datan alkuperäistä talletuspaikkaa muuteta. Käyttäjäystävällisyyttä on painotettu kyseisessä ratkaisussa, eikä käyttäjältä vaadita lainkaan ohjelmointitaitoja. (Vathy-Fogarassy & Hugyák, 2017)

Kyselylausekkeet kootaan graafisessa käyttöliittymässä käyttämällä valitusta tietokannasta metadataan talletettuja avainsanatietoja. Näin käyttäjä voi valita haluamansa attribuutit tietokannoista tuntematta tietokantojen rakenteita. Syötetyt kyselyparametrit, kuten valitut attribuutit ja järjestämisehdot, kootaan JSON-olioihin. HybridDB ei hyödynnä kyselykielenä SQL:ää eikä näin ollen vaadi käyttäjältään kyseisen kielen syntaksin ymmärtämistä. Kyselyt muodostetaan JSON-olioiksi, jotka käännetään lähdetietokannan ymmärtämälle kyselykielelle. Saadut tulokset talletetaan uudestaan JSON-formaattiin. (Vathy-Fogarassy & Hugyák, 2017)

Julkaisuartikkelissaan HybridDB:n arkkitehtuuri ja jokaisen siihen kuuluvan komponentin vastuualueet on kuvattu kattavasti erikseen (Vathy-Fogarassy & Hugyák, 2017). Tämän tutkielman kontekstissa on oleellista ymmärtää toiminnan perusperiaatteet ja tunnistaa myöhempää vertailua varten muutama avainkomponentti.

Lähdetietokantojen rakenteellisia tietoja talletetaan geneeriseen skeemaan (general schema). Geneerinen skeema sisältää muun muassa lähdetietokantajärjestelmän attribuuttien nimet, datatyypit ja rajoitteet. Malliksi (model) kutsuttu komponentti kutsuu tietokantaoperaatioita. JSON-objektiin talletettujen kyselyparametrien muotoilu lähdetietokannalle sopivaan muotoon tapahtuu usean eri komponentin yhteistyönä. Tietokanta-adapteri (database adapter) -komponentti saa tietoja paitsi geneeristen skeemojen talletuskomponentilta, myös mallikomponentilta. Keräämänsä tiedon perusteella se tulkkaa käyttäjän syöttämän kyselyn lähdetietokantajärjestelmälle sopivaan syntaksiin. Eri tietokantajärjestelmien ajurit ovat tietokannanhallintajärjestelmän kautta yhteydessä tietokanta-adapterikomponenttiin. Ajurit suorittavat kaikki tietokantaoperaatiot lähdejärjestelmissään palauttaen sitten kyselytulokset tietokanta-adapterille. (Vathy-Fogarassy & Hugyák, 2017)

HybridDB -web applikaatio kehitettiin julkaisussa esitetyn metodologian havainnollistamiseksi ja testaamiseksi. Vaihtelevan kokoisia tietokantoja sekä MySQL:stä että MongoDB:stä käytettiin testaamaan metodia. Testauksessa todettiin, että HybridDB:n suoritusajat ovat pääasiassa riippuvaisia lähdetietokantojen omista suoritusajoista. Luotu web applikaatio kykeni artikkelin julkaisuajankohtana

käsittelemään MySQL- ja MongoDB-tietokantajärjestelmiä, mutta kirjoittajien mukaan metodiin on tulevaisuudessa helppo lisätä uusia ja erilaisia tietokantoja. Tämä vaatii aina kyseisen tietokantajärjestelmän rakenteellisen logiikan kartoittamista ja JSON-kielelle talletettavien kyselyjen sovittamista kyseiseen rakenteeseen. (Vathy-Fogarassy & Hugyák, 2017)

6 Metodien vertailu

Tässä luvussa vertaillaan keskenään edellä esiteltyjä dataintegraatiometodeja. Alaluvuissa tarkastellaan eri metodien päämääriä ja arkkitehtuuria. Lopuksi analysoidaan esiteltyjen integraatiometodien erilaisten lähestymistapojen etuja.

6.1 Päämäärät

Sekä Unity (Lawrence, 2014), MSI (Li & Gu, 2019) että Bjeladinovicin ja muiden (2020) arkkitehtuuri pyrkivät suorittamaan yhtäaikaisia kyselyjä hybriditietokantoihin SQL-kielen kautta. Aftabin ja muiden (2020) metodin päämäärä on muuntaa NoSQL-tietokanta relaatiotietokannaksi automaattisesti. Jokaisessa näistä menetelmistä muunnoksen suunta on NoSQL-tietokannoista kohti relaatiotietokantojen rakennetta. Tämä johtuu siitä, että tavoitteena kaikissa metodeissa oli voida hyödyntää SQL:ää kyselykielenä, eikä SQL ole lähtökohtaisesti yhteensopiva monien NoSQL-kantojen kanssa. (Lawrence, 2014; Li & Gu, 2019; Bjeladinovic ja muut, 2020; Aftab ja muut, 2020) Relaatiotietokannat omaavat jo valmiiksi keskenään yhtenäisen rakenteen ja SQL on nimenomaan niille ominainen kyselykieli (Lawrence, 2014).

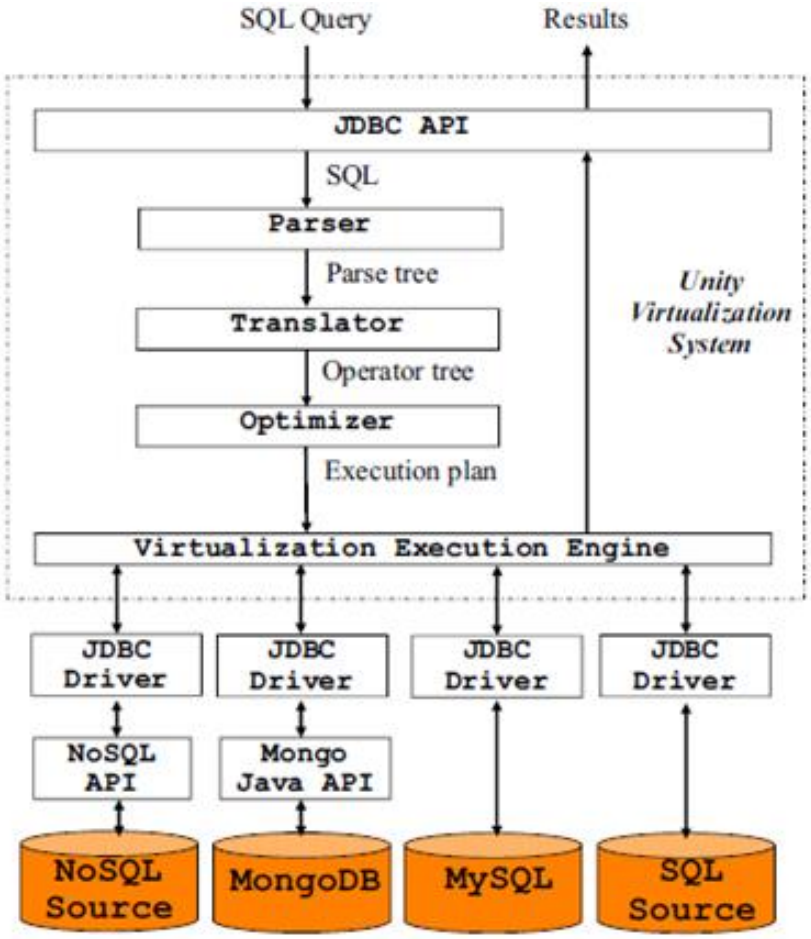
Poikkeukseksi muodostui HybridDB, jonka lähtökohtana muodostetuille kyselylausekkeille on tallentaa ne JSON-formaattiin. Lähdetietokannat voivat sisältää sekä SQL- että NoSQL-tietokantoja, mutta kummankaan natiivia kyselykieltä ei käytetä alusta asti muodostamaan halutun kyselyn logiikkaa. Myös HybridDB:n julkaisuartikkelissa on painotettu hybriditietokantoihin kohdistuvan yhtäaikaisen kyselyn tarvetta. Sen lisäksi käytön helppoutta on korostettu voimakkaasti. Käyttäjän ei tarvitse muodostaa SQL-kyselylausekkeita, vaan kyselyt muodostetaan valitsemalla halutut attribuutit graafisessa käyttöliittymässä. (Vathy-Fogarassy & Hugyák, 2017) Muihin tässä tutkielmassa esitettyihin metodeihin verrattuna HybridDB:n käyttö on mistään kyselykielestä riippumatonta.

Kaikille esitetyille metodeille yhteistä on tarve tulkata eri muodoissa tallennettua dataa niin, että niiden yhteen koostaminen samojen loogisten mallien mukaan on mahdollista (Lawrence, 2014; Li & Gu, 2019; Bjeladinovic ja muut, 2020; Vathy-Fogarassy & Hugyák, 2017; Aftab ja muut, 2020). Dataintegraatiometodi voi toteuttaa tämän siirtämällä datan uuteen, yhtenäisempään muotoon (Aftab ja muut, 2020), tai yhdistämällä eri muotoisia kyselytuloksia (Lawrence, 2014; Li & Gu, 2019; Bjeladinovic ja muut, 2020; Vathy-Fogarassy & Hugyák, 2017).

6.2 Arkkitehtuuri

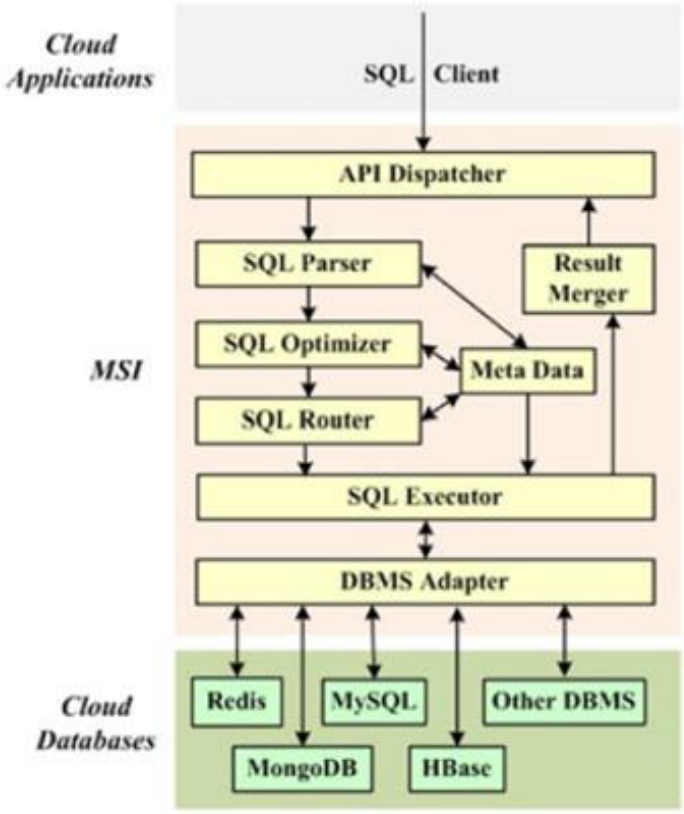
Bjeladinovic ja muut (2020) luonnehtivat integraatiometodien yleistä arkkitehtuuria kolmitasoiseksi. Tällainen rakenne on havaittavissa niin heidän esittämässään arkkitehtuurissa (Bjeladinovic ja muut, 2020), kuin Unity- (Lawrence, 2014) ja MSI -malleissakin (Li & Gu, 2019). Tässä tutkielmassa esitetyissä malleissa ensimmäinen taso keskittyy vuorovaikutukseen käyttäjän kanssa. Ensimmäisellä tasolla vastaanotetaan käyttäjän syöttämä kyselylauseke ja lopulta esitetään tälle prosessin kautta saatu tulos. Toinen taso keskittyy kyselylausekkeen tulkkaamiseen ja käsittelyyn, sisältäen suurimman osan varsinaisista integraatiometodeista. Kolmas osa koostuu lähdetietokannoista, joista kyselyn tuottaman tuloksen halutaan koostuvan. (Bjeladinovic ja muut, 2020; Lawrence, 2014; Li & Gu, 2019)

Unityn (Lawrence, 2014) ja MSI:n (Li & Gu, 2019) arkkitehtuureissa on paljon samaa. Kuvassa kaksi on esitetty Unityn ja kuvassa kolme MSI:n arkkitehtuurien kuvaukset. Eri työvaiheiden eteneminen tapahtuu suurimmaksi osaksi samassa järjestyksessä. SQL-kyselyjen jäsentämiseen (parse) sovellettiin järjestelmissä kahta toisistaan hieman eroavaa, jo olemassa olevaa työkalua. Työkalujen valintakriteerejä ei perusteltu. (Lawrence, 2014; Li & Gu, 2019) Unity jatkaa syötetyn kyselyn tulkintaa vielä kääntäjäkomponentillaan ennen kuin siirtyy kyselyn optimointiin (Lawrence, 2014). Molemmissa järjestelmissä suoritetaan kyselyn optimointivaihe, jossa alkuperäinen kysely pyritään muotoilemaan tehokkaammaksi ja jakamaan eri lähdetietokannoissa suoritettaviin palasiin. Tarkennettujen ja oikeisiin kohteisiinsa jaettujen kyselyjen lähettäminen lähdetietokannoille tapahtuu molemmissa järjestelmissä kyselyn käsittelyvaiheiden jälkeen. (Lawrence, 2014; Li & Gu, 2019) Unityssa Virtualization Execution Engine- (Lawrence, 2014), ja MSI:ssä SQL Executor- (Li & Gu, 2019) komponentit viittaavat tähän vaiheeseen.



Kuva 2. Unity-järjestelmän arkkitehtuuri (Lawrence, 2014)

Unityn toimintatapaa tulosten käsittelyssä ja yhdistämisessä ei avata tarkasti (Lawrence, 2014). MSI:n arkkitehtuurissa mainitaan erikseen sekä tulokset yhdistävät (Result Merger) että metadataa käsittelevät (Meta Data) komponentit (Li & Gu, 2019). MSI:n metadataa käsittelevä komponentti säilöö tietoja tietokantojen rakenteellisista konfiguraatioista ja SQL-kyselyyn kohdistuneista muutoksista, sekä sen tuloksista (Li & Gu, 2019). Unityssa täysin vastaavanlaisia toiminnallisuuksia ei kuvata, mutta sen arkkitehtuurissa mainitaan tiedon rakenteen metadataa koskeva, vapaaehtoinen skeeman generoiva (schema generator) komponentti (Lawrence, 2014). Komponentti tuottaa tarvittaessa skeemattomalle NoSQL-tietokannalle sopivan skeeman, joka kuvaa yleisluontoisesti tiedon rakennetta kannassa (Lawrence, 2014). Unity-metodissa (Lawrence, 2014) luotuun skeemaan ei kuitenkaan siirretä dataa, kuten Aftabin ja muiden (2020) metodissa.



Kuva 3. MSI-järjestelmän arkkitehtuuri (Li & Gu, 2019)

Sekä Unity (Lawrence, 2014) että MSI (Li & Gu, 2019) ovat keskenään rakenteellisesti samankaltaisia, vaikka käytetty termistö ja komponenttien nimet eroavat jokseenkin toisistaan. Myös Bjeladinovicin ja muiden (2020) metodin arkkitehtuuri muistuttaa vähemmän yksityiskohtaisella tasolla Unitya (Lawrence, 2014) ja MSI:tä (Li & Gu, 2019). Kaikissa kolmessa metodissa tiedon kulku ja samankaltaiset toiminnot tapahtuvat prosessin samoissa vaiheissa. Suurin osa kaikkien kolmen integraatiometodin toiminnallisuudesta keskittyy SQL-kielisen kyselylausekkeen tulkintaan ja muokkaamiseen. (Lawrence, 2014; Li & Gu, 2019; Bjeladinovic ja muut, 2020)

Arkkitehtuurien kuvailujen perusteella kaikki metodit, paitsi Aftabin ja muiden (2020) metodi, sisältävät monia hyvin samankaltaisia toiminnallisuuksia, mutta näiden toiminnallisuuksien tehtävänjako eri komponenttien välille voi erota toisistaan monin tavoin (Lawrence, 2014: Li & Gu, 2019; Bjeladinovic ja muut, 2020; Vathy-Fogarassy & Hugyák, 2017). Kaikissa näissä SQL-kyselyihin pohjautuvissa metodeissa SQL-kielisen kyselyn jäsentäminen ja tulkkaaminen lähdetietokantaan sopivaksi jakautuu useampien komponenttien vastuulle (Lawrence, 2014: Li & Gu, 2019; Bjeladinovic ja muut, 2020;

Vathy-Fogarassy & Hugyák, 2017). SQL-lausekkeen tulkkaamiseen, eli lauseen logiikan ymmärtämiseen, on sekä MSI:ssä (Li & Gu, 2019) että Unityssa (Lawrence, 2014) käytetty SQL Parser -komponenttia, joka molemmissa perustuu jo olemassa oleviin työkaluihin. Bjeladinovicin ja muiden arkkitehtuurissa tämä tulkkaaminen tapahtuu Entered Statement Processing -komponentissa (Bjeladinovic ja muut, 2020).

Kyselyn jäsentäminen ja kääntäminen lähdetietokantaan sopivaksi tapahtuu Bjeladinovicin ja muiden (2020) metodissa vaiheittain Key Word Search- ja Statement Mapper -komponenttien kautta, kun taas HybridDB-metodissa (Vathy-Fogarassy & Hugyák, 2017) Database Adapter -komponentti tekee sen suhteen suurimman työn. Monet metodeista sisältävät erillisen, metadatan käsittelyyn liittyvän komponentin, jota hyödynnetään yleensä kyselyn kääntämisessä. Tällaisia ovat esimerkiksi Bjeladinovicin ja muiden (2020) metodin Key Word Search -komponentti, HybridDB:n (Vathy-Fogarassy & Hugyák, 2017) General Schema Storage -komponentti ja MSI:n (Li & Gu, 2019) Meta Data -komponentti. Metadatan tarkempi sisältö kuitenkin eroaa arkkitehtuureissa hieman (Bjeladinovic ja muut, 2020; Vathy-Fogarassy & Hugyák, 2017; Li & Gu, 2019).

Bjeladinovicin ja muiden (2020) metodin arkkitehtuuri on kuvaukseltaan kattavin, ja siinä komponenttien nimeäminen ja tehtävänjako eroavat MSI:stä (Li & Gu, 2019) ja Unitysta (Lawrence, 2014), jotka puolestaan ovat keskenään hyvinkin samankaltaiset. HybridDB:n (Vathy-Fogarassy & Hugyák, 2017) arkkitehtuuri eroaa kolmesta edellä mainitusta siksi, että kyseessä on muista poikkeavasti web applikaatio, eikä kyseinen metodi alusta kyselyjä lähtökohtaisesti SQL-kielisiksi. Näille neljälle metodille yhteistä on aiemmin mainittu kolmitasoinen malli (Lawrence, 2014: Li & Gu, 2019; Bjeladinovic ja muut, 2020; Vathy-Fogarassy & Hugyák, 2017).

Aftabin ja muiden (2020) metodi ei keskity yhtäaikaisten kyselyjen suorittamiseen, vaan datan siirtämiseen yhdestä tietokantatyypistä toiseen. Tästä syystä metodin arkkitehtuuri ei ole suoraan verrattavissa aiemmin esiteltyihin rakenteisiin. Aftabin ja muiden (2020) metodissa keskeisimmät komponentit ovat skeeman analysoija (schema analyzer) sekä ETL-prosessit (ETL processes). Molemmat komponentit ovat vuorollaan yhteydessä sekä NoSQL-tietolähteeseen että SQL-kohdetietokantaan. Skeeman analysoija luo SQL-tietokannan vaatiman rakenteellisen mallin NoSQL-tietolähteen sisältämän datan perusteella. Tämän jälkeen metodin dataa muuntavat prosessit hakevat

tietolähteestä dataa osissa ja syöttävät ne kohdetietokantaan pala kerrallaan. (Aftab ja muut, 2020) Aftabin ja muiden (2020) metodissa toiminta on muihin esiteltyihin metodeihin verrattuna iteratiivisempaa ja dynaamisempaa, sillä skeeman rakentaminen tapahtuu pala kerrallaan analysoidun datan pohjalta (Aftab ja muut, 2020).

6.3 Lähestymistapojen analyysi

MSI:n (Li & Gu, 2019), Unityn (Lawrence, 2014), HybridDB:n (Vathy-Fogarassy & Hugyák, 2017) sekä Bjeladinovicin ja muiden (2020) tavoittelemassa yhtäaikaisten kyselyjen lähestymistavassa lähdetietokannat pysyvät alkuperäisissä muodoissaan. Ne säilyttävät heterogeenisyytensä, ja näin ollen omat erityispiirteensä ja vahvuutensa. Esimerkiksi erittäin suurten datamäärien muuntaminen NoSQL-tietokannasta SQL-tietokantaan ei usein ole mielekästä, sillä NoSQL-tietokannat ovat usein toimivampi ratkaisu tällaisen datan säilömiseen (Aftab ja muut, 2020; Zeng ja muut, 2017).

Aftabin ja muiden (2020) metodi taas pyrkii nimenomaan muuntamaan NoSQL-pohjaisen tietolähteen relaatiotietokannaksi. Tällainen lähestymistapa voi olla tarpeen, mikäli jokin toinen tietokantamalli soveltuu lähdedatan käsittelyyn ja säilömiseen alkuperäistä paremmin. Useat lähteet mainitsivat relaatiotietokantojen datankäsittelyoperaatiot ja yhteensopivuuden SQL-kielen kanssa nimenomaan relaatiomallin vahvuutena (Lawrence, 2014; Li & Gu, 2019; Aftab ja muut, 2020). Tiedon siirtäminen NoSQL-kannasta SQL-kantaan voi olla hyvä vaihtoehto myös tilanteessa, jossa tiedon säilöntämallilla ei ole suurta merkitystä, mutta kyseistä tietokantaa halutaan hyödyntää mahdollisimman saumattomasti muiden SQL-tyyppisten tietokantojen kanssa. Tämä perustuu siihen, että SQL-tietokannat ovat hyvin samankaltaisia, joka mahdollistaa siirrettävyyden niiden välillä (Lawrence, 2014).

Dataintegraation teorian mukaan (Lenzerini, 2002) kyselyjen suorittamista useampaan tietolähteeseen yhtäaikaisesti voidaan lähestyä globaalin näkymän (global-as-view) tai lokaalin näkymän (local-as-view) kautta. Globaaliin näkymään pohjautuvassa dataintegraatiojärjestelmässä lähdetietokantojen datasta yhdistetään kaikki lähteet kattava, laaja tietovarasto, johon kyselyt suoritetaan. Lokaalin näkymän järjestelmässä taas kyselyt suoritetaan jokaiseen lähdetietokantaan erikseen, jonka jälkeen tulokset yhdistetään vain halutut tiedot sisältäväksi näkymäksi käyttäjälle. Eräs oleellinen ero näiden lähestymistapojen välillä on se, että lokaalin näkymän järjestelmiin uusien

tietolähteiden lisääminen on merkittävästi helpompaa. (Lenzerini, 2002) Kaikki tässä tutkielmassa esitellyt metodit voidaan lukea lokaalin näkymän järjestelmiksi.

7 Pohdinta

Lähteiden perusteella on havaittavissa, että kiinnostus alalla kohdistuu siihen, miten dataintegraation avulla voitaisiin hyödyntää sekä SQL- että NoSQL-järjestelmämallien vahvuuksia. Vaikka erilaisten tietojärjestelmien integraatio onkin monimutkainen tehtävä, mahdollistaa se onnistuessaan kunkin datan säilömisen juuri kyseiselle datalle parhaiten sopivaan järjestelmään. Atzenin ja muiden (2020) mukaan ei olekaan realistista olettaa, että yksi tietorakennemalli sopisi kaikkiin mahdollisiin applikaatioihin.

Tähän tutkielmaan valituissa metodeissa pääosin mukautettiin NoSQL-tietojärjestelmiä sopimaan SQL-tietokantarakenteisiin. Yhteensopivuus voitiin luoda sekä muuntamalla datan säilytyspaikkaa itseään (Aftab ja muut, 2020) että tulkkaamalla siihen kohdistuvia kyselyjä (Lawrence, 2014; Li & Gu, 2019, Bjeladinovic ja muut, 2020). Vastakkaiseen suuntaan tapahtuvaa muutosta on myös tutkittu, vaikka sitä ei tässä tutkielmassa tarkasteltukaan. SQL-tietokantoja NoSQL-muotoon muuntavia metodeja (Zhao ja muut, 2014; Solanke & Rajeswari 2017) on esiintynyt tieteellisessä kirjallisuudessa. Oli muunnoksen suunta mikä hyvänsä, datan muuntamisessa yhdestä tietojärjestelmämallista toiseen ei kuitenkaan ole aina kyse toisen mallin etusijalle asettamisesta. Esimerkiksi Solanken ja Rajeswarin (2017) tietokantamuunnoksen toteuttavassa järjestelmässä on huomioitu halu hyödyntää hybriditietojärjestelmiä.

Nykyinen muutossuunta ja kehitykseen liittyvät trendit kasvattavat jatkuvasti tarvetta käsitellä yhä suurempia datamääriä ja entistä monimuotoisempaa dataa (Zeng ja muut, 2017). Tämän vuoksi alati kasvava kiinnostus NoSQL-järjestelmiin on odotettavissa. Relaatiokannoilla puolestaan on omat vahvuutensa transaktiomekanismien, datan eheyden, tietoturvan ja tietorakenteiden yhtenäisyyden suhteen (Lawrence, 2014; Doncevic & Fertalj, 2020; Aftab ja muut, 2020). Eräs useasti mainittu tekijä, joka pitää relaatiotietokannat ja SQL-kyselykielen yhä avainasemassa organisaatioiden tietokantajärjestelmissä on kuitenkin se, että ne ovat toimineet mallina kaikille tietokantajärjestelmille jo vuosia (Lawrence, 2014; Li & Gu, 2019). Näin ollen monet organisaatiot ovat hyvin vahvasti sitoutuneita relaatiokantoihin, koska ne ovat merkittävä osa niiden tietojärjestelmiä.

Organisaatiot ylittävän tiedonkulun mahdollistamiseksi myös relaatiokantoja käyttämättömien tahojen on huomioitava SQL-yhteensopivuus omissa järjestelmissään. Järjestelmiä, jotka hyötyisivät NoSQL-tietokantamalleista relaatiomallin sijaan, ei kannata lähteä muuttamaan pitämättä mielessä, että muutoksen seurauksena erilaisten tietokantojen yhteentoimimattomuus voi nousta ongelmaksi. SQL-pohjaisille järjestelmille on myös olemassa paitsi enemmän yhteensopivia työkaluja, myös ihmisresursseja, eli kokeneita työntekijöitä. Tämä on seurausta siitä, että kyseiset järjestelmät ovat vakiintuneisuutensa vuoksi olleet laajemmin ja pidempään käytössä, kuin NoSQL-järjestelmät (Lawrence, 2014).

Tässä tutkielmassa huomattiin, että useampi metodi pyrki hyödyntämään SQL-kyselykieltä myös NoSQL-rakenteissa. Tarkastelluista metodeista enemmistö pyrki suorittamaan kyselyjä sekä relaatio- että NoSQL-tietokantoihin SQL-kielen avulla (Lawrence, 2014; Li & Gu, 2019; Bjeladinovic ja muut, 2020). Tämä tukee näkökulmaa siitä, että lähtökohtainen yhteensopivuus SQL-kyselykieleen on eräs relaatiotietokantojen merkittävimmistä vahvuuksista. Halu säilyttää sekä relaatio- että NoSQL-mallien rakenteet eri tietojoukkojen säilömisessä taas viittaa näkemykseen, jossa tietojen säilytyspaikoille haluttu ominaisuus on mukautuvuus kunkin säilöttävän datan ominaisiin rakenteisiin.

Huomionarvoinen seikka oli myös se, että kaikki tutkielmassa esitellyt metodit olivat luettavissa lokaalin näkymän järjestelmiksi. Lokaalin näkymän järjestelmät sopivat paremmin järjestelmiksi, joihin voidaan tulevaisuudessa lisätä tietolähteitä (Lenzerini, 2002). Kiinnostus lokaalin näkymän järjestelmiä kohtaan viittaa haluun luoda tarpeen tullen joustavia ja mukautumiskykyisiä tietovarastoja.

Muutamassa artikkelissa esitettiin muuhun kirjallisuuteen liittyviä väitteitä, joiden todenmukaisuus on tämän tutkielman pojalta kyseenalaista. Aftabin ja muiden (2020) mukaan työkaluja, jotka muuntavat dataa SQL-mallista NoSQL-malleihin on olemassa enemmän, kuin vastakkaista muutossuuntaa tukevia työkaluja. Tämän tutkielman perusteella vaikuttaa kuitenkin siltä, että nykyisen muutossuunnan valossa tämä Aftabin ja muiden (2020) väite voi pian olla vanhentunut. Bjeladinovic ja muut (2020) taas luonnehtivat omaa metodiaan aiemmassa kirjallisuudessa esitetyistä integraatiomenetelmistä poikkavaksi. Syyksi tähän mainittiin metodin tarjoama integraation ja yhtäaikaisen käytön mahdollisuus SQL/NoSQL-hybriditietokantojen

komponenteille (Bjeladinovic ja muut, 2020). Tämän tutkielman perusteella tällainen toiminnallisuus ei kuitenkaan riitä luonnehtimaan esitettyä metodia uniikiksi tai muusta kirjallisuudesta poikkeavaksi.

8 Yhteenveto

Tiedon kasvava määrä ja monimuotoisuus kasvattavat vastavuoroisesti tarvetta SQL- ja NoSQL-tietokantojen väliselle dataintegraatiolle. Molemmilla tietokantamalleilla on erilaiset ominaisuudet ja vahvuudet, joita tarkasteltiin tutkielman luvussa kolme. Tämä monimuotoisuus nähdään nykypäivänä laajalti vahvuutena. Semanttisesti ja rakenteellisesti monimuotoinen data vaatii myös talletuspaikoiltaan monimuotoisuutta ja joustavuutta. Datan säilöminen nimenomaan sen rakenteelle parhaiten soveltuvaan tietokantaan tarkoittaa kuitenkin usein yhteensopivuusongelmia eri tietokantojen välillä.

Dataintegraatio pyrkii ratkomaan erilaisten tietokantojen yhteensopivuusongelmia, ja tämän saavuttamiseen on monia tapoja. Viime vuosina tieteellisessä kirjallisuudessa julkaistujen dataintegraatiometodien pohjalta on nähtävillä, millaiset ratkaisumallit ovat yleisiä. Tässä tutkielmassa tarkastellut metodit keskittyvät heterogeenisten tietolähteiden yhtäaikaiseen hyödyntämiseen sen sijaan, että lähtökohtana olisi pääosin ollut halu muokata tietolähteistä keskenään homogeenisiä. Vaikka lähdekirjallisuudessa esiintyi myös tästä lähtökohdasta poikkeavia metodeja, yleisesti ratkaisut lähestyivät dataintegraatiota tarkoituksenaan säilyttää hybriditietokannat heterogeenisinä.

Lähdeluettelo

Aftab, Z., Iqbal, W., Almustafa, K., Bukhari, F. & Abdullah, M. (2020) Automatic NoSQL to Relational Database Transformation with Dynamic Schema Mapping. *Scientific programming, 2020.* S. 1-13. doi: https://doi.org/10.1155/2020/8813350

Atzeni, P., Bugiotti, F., Cabibbo, L. & Torlone, R. (2020) Data modeling in the NoSQL world. *Computer standards and interfaces, 67*. doi: https://doi.org/10.1016/j.csi.2016.10.003

Bjeladinovic, S., Marjanovic, Z. & Babarogic, S. (2020) A proposal of architecture for integration and uniform use of hybrid SQL/NoSQL database components. *Journal of Systems and Software, 168(*110633*).* doi*:* https://doi.org/10.1016/j.jss.2020.110633

Doan, A., Halevy, A. & Ives, Z. (2012) Principles of Data Integration. 1. painos. *Elsevier Science & Technology*, 2012. Web. doi: https://doi.org/10.1016/C2011-0-06130-6

Doncevic, J. & Fertalj, K. (2020) Database Integration Systems. *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO).* Opatija, Croatia. 28.7.-2.10.2020. S. 1617-1622. doi: https://doi.org/10.23919/MIPRO48935.2020.9245245

JuFo. (2021) *Julkaisufoorumi*. https://www.tsv.fi/julkaisufoorumi/haku.php (Haettu 13.4.2021)

Kementsietsidis, A. (2009) Schema Matching. Teoksessa L. Liu & M.T. Özsu (toim), *Encyclopedia of Database Systems.* S. 2494–2497. Springer, Boston, MA. doi: https://doi.org/10.1007/978-0-387-39940-9_962

Kuchibhotla, H. N., Dunn, D. & Brown, D. (2009) Data integration issues in IT organizations and a need to map different data formats to store them in relational databases. *41st Southeastern Symposium on System Theory.* Tullahoma, Tennessee, USA. 15.-17.3.2009. S. 1-6. doi: https://doi.org/10.1109/SSST.2009.4806804

Lawrence, R. (2014) Integration and Virtualization of Relational SQL and NoSQL Systems Including MySQL and MongoDB. *2014 International Conference on Computational Science and computational Intelligence.* Las Vegas, Nevada, USA. 10.-13.3.2014. S. 285-290. doi: https://doi.org/10.1109/CSCI.2014.56

Lawrence, R. (2017) Faster Querying for Database Integration and Virtualization with Distributed Semi-Joins. *2017 International Conference on Computational Science and Computational Intelligence (CSCI).* Las Vegas, Nevada, USA. 14.-16.12.2017. S. 1406-1410. doi: https://doi.org/10.1109/CSCI.2017.246

Lenzerini, M. (2002) Data integration: A theoretical perspective. *21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2002*). Madison, Wisconsin, USA. 3.-5.6.2002. S. 233-246. doi: https://doi.org/10.1145/543613.543644

Li, C. & Gu, J. (2019) An integration approach of hybrid databases based on SQL in cloud computing environment. *Software, practice & experience, 49(*3*)*. S. 401-422. doi: https://doi.org/10.1002/spe.2666

Solanke, G. & Rajeswari, K. (2017) SQL to NoSQL transformation system using data adapter and analytics. *2017 IEEE International Conference on Technological Innovations in Communicatio, Control and Automation (TICCA).* Chennai, India. 6.4.2017. S. 59-63. doi: https://doi.org/10.1109/TICCA.2017.8344580

Techopedia sanakirja: *ACID, ETL, JDBC, ODBC, data source, XML database*. (2021) *Techopedia Dictionary.* Tietotekniikkaan keskittyvä uutissivusto. Techopedian teknologia-sanakirja. https://www.techopedia.com/dictionary (Haettu 13.4.2021)

UnityJDBC. (2021) *UnityJDBC Website.* UnityJDBC-virtualisointijärjestelmän kotisivut. https://unityjdbc.com (Haettu 20.02.2021)

Vathy-Fogarassy, Á. & Hugyák, T. (2017) Uniform data access platform for SQL and NoSQL database systems. *Information Systems (Oxford), 69*(93-105). doi: https://doi.org/10.1016/j.is.2017.04.002

Zeng, N., Zhang, G., Li, X. & Cui, L. (2017) Evaluation of relational and NoSQL approaches for patient cohort identification from heterogeneous data sources. *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM).* Kansas City, Missouri, USA. 13.-16.11.2017. S. 1135-1140. doi: https://doi.org/10.1109/BIBM.2017.8217817

Zhao, G., Lin, Q., Li, L. & Li, Z. (2014) Schema Conversion Model of SQL Database to NoSQL. *2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. Guangdong, China. 8.-10.11.2014. S. 355-362. doi: https://doi.org/10.1109/3PGCIC.2014.137