Tero Kadenius

SYNTAKSIVIRHEILMOITUSTEN KOETTU HYÖDYLLISYYS MYSQL- JA POSTGRESQL-TIETO-KANNANHALLINTAJÄRJESTELMISSÄ

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2020

TIIVISTELMÄ

Kadenius, Tero
Syntaksivirheilmoitusten koettu hyödyllisyys MySQL- Ja PostgreSQL-tietokannanhallintajärjestelmissä
Jyväskylä: Jyväskylän yliopisto, 2020, 46 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja: Taipalus, Toni

Kääntäjien ja tulkkien raportoimia diagnostisia virheilmoituksia on tutkittu yli 50 vuoden ajan. Aihe on merkittävä sillä virheilmoitukset ovat keskeinen palautteenantomuoto kääntäjältä käännettävän ohjelmakoodin kirjoittajalle. Tästä huolimatta virheilmoitukset ovat monilta osin riittämättömiä ja ongelmallisia eikä tällä osa-alueella ole tapahtunut merkittäviä läpimurtoja. Erityisen kriittisiä selkeät ja korjaamista helpottavat virheilmoitukset ovat vasta vasta-alkajille, mutta myös ammattilaiset hyötyisivät paremmista virheilmoituksista. Suurin osa tutkimuksesta on keskittynyt ohjelmointikielten virheilmoituksiin. Tietokantojen kääntäjien raportoimia virheilmoituksia on tutkittu merkittävästi vähemmän. Tässä tutkimuksessa selvitetään verkkopohjaisella kyselytutkimuksella kuinka kahden suosituimman avoimen lähdekoodin relaatietokannanhallintajärjestelmän, PostgreSQL:n ja MySQL:n SQL-kyselyihin raportoimat syntaksivirheilmoitukset auttavat opiskelijoita havaitsemaan ja korjaamaan virheitä. Tulokset osoittavat, että PostgreSQL:n tuottamat virheilmoitukset ovat hyödyllisempiä sekä koetussa virheiden havaitsemisessa, korjaamisessa, että korjausvarmuudessa.

Asiasanat: virheilmoitukset, syntaksivirheet, kääntäjävirheet, SQL, tietokannanhallintajärjestelmät

ABSTRACT

Kadenius, Tero
Perceived usefulness of syntax error messages in MySQL and PostgreSQL database management systems
Jyväskylä: University of Jyväskylä, 2020, 46 pp.
Information Systems, Master's Thesis
Supervisor: Taipalus, Toni

Diagnostic error messages reported by compilers and interpreters have been researched for over 50 years. The subject is fundamental because error messages are a key feedback channel from the compiler to the person writing the code. Despite this importance, error messages remain insufficient and problematic and there have been no breakthroughs on the field. Clear and helpful error messages are especially crucial for novices but also professionals would benefit from better error messages. Most of the research has focused on error messages generated by programming language error messages. There have considerably fewer studies on error messages generated by compilers in databases. In this study it is examined how the error messages reported by two of the most popular relational database management systems, PostgreSQL and MySQL help students detect and correct errors. The data is gathered with an online survey. The results indicate that error messages generated by PostgreSQL are more useful in perceived detection, correction and in the reliability of the correction.

Keywords: error messages, syntax errors, compiler errors, SQL, database management systems

KUVIOT

TAULUKOT

SISÄLLYS

1 JOHDANTO

Ohjelmointikielten kääntäjien tuottamien virheilmoitusten taso jättää paljon toivomisen varaa (Becker ym., 2019) eivätkä kääntäjien kehittäjät ole merkittävästi panostaneet virheilmoitusten parantamiseen (Traver, 2010). Virheilmoitukset ovat kuitenkin tärkeimpiä kontaktipisteitä ohjelmoijan ja järjestelmän välillä ja erityisen kriittisiä ne ovat vasta-alkajille, jotka eivät ole vielä oppineet käsittelemään ohjelmointiympäristön tuottamaa puutteellista tai vaikeaselkoista palautetta (Marceau ym., 2011). Virheilmoitusten ymmärtäminen on tärkeä kiinnostuksen kohde tietojenkäsittelyn opetuksessa mm. siksi, että tämän ymmärryksen avulla voidaan parantaa opetusta sekä suunnitella parempia työkaluja ohjelmoinnin tueksi (McCall & Kölling, 2019). Erityisesti kääntäjävirheet ovat haastavia vasta-alkajille, koska käännöksen pysähtyvä suoritus ei anna opiskelijalle minkäänlaisia tuloksia (McCall & Kölling, 2019). Vasta-alkajien lisäksi paremmat virheilmoitukset ovat myös ammattilaisten intresseissä (Becker ym., 2019).

Ohjelmointikielten virheilmoituksista on tuotettu mittava määrä tutkimusta (Becker, ym. 2019; Ahadi, ym. 2016). Sen sijaan SQL-lauseisiin liittyviä virheilmoitukset ovat saaneet vähemmän huomiota (Ahadi, ym. 2016), vaikka SQL:ää hyödyntävät relaatiotietokannanhallintajärjestelmät ovat edelleen markkinoiden suosituimpia tietokantaratkaisuja (DB-engines, 2020b). Erityisesti, kirjallisuuskatsauksen perusteella ei löytynyt viitteitä siitä, että eroja näiden tietokantaratkaisujen virheilmoitusten laadussa olisi tutkittu. Tässä tutkimuksessa osaltaan paikataan tätä puutetta.

Tärkeimpiä käsitteitä liittyen relaatiotietokantoihin, tietokannanhallintajärjestelmiin ja SQL:ään sekä katsaus virheilmoitustutkimukseen esitellään luvussa 2. Luvussa 3 käydään läpi tutkimuksen tausta, tutkimuskysymykset ja tutkimusmenetelmä. Tutkimuksessa vertailtiin kahden suosituimman avoimen lähdekoodin tietokannanhallintajärjestelmän (DB-engines, 2020b) eroja mitä tulee virheilmoituksen vaikutukseen opiskelijoiden koettuun virheiden havaitsemiseen, korjaamiseen ja korjausvarmuuteen. Tutkitut tietokannanhallintajärjestelmät olivat MySQL 8.0.12 ja PostgreSQL 12.1. Tutkimus tehtiin verkkovälitteisenä kyselynä. Koehenkilöille esitettiin 16 kysymystä, jossa oli esitetty syntaktisesti

puutteellinen SQL-lause sekä tietokannanhallintajärjestelmän ko. lauseelle raportoima virheilmoitus. Kyselyn virheiden pohjana oli Taipalus ym. (2018) tietokannanhallintajärjestelmäriippumaton virheluokittelu. Kysymyksiin valittiin 16 yleisintä syntaksivirhettä (Taipalus ym. 2018; Taipalus & Perälä, 2019). Vastaajien tehtäväksi jäi korjata SQL-lause ja arvioida miten hyödylliseksi he kokivat virheilmoituksen virheen havaitsemisen ja korjaamisen kannalta sekä kuinka varmoja he olivat korjauksen oikeellisuudesta. Tulosten analysointiin käytettiin epäparametrista Mann-Whitneyn U-testiä. Tutkimuksessa selvisi, että PostgreSQL suoriutui MySQL:ää paremmin kaikissa tutkimuskysymyksissä. Tutkimuksen tulokset raportoitiin kunkin tutkimuskysymyksen osalta sekä koostaen tietokannanhallintajärjestelmittäin, että eroteltuna kysymyslomakkeen tehtävien mukaan eli syntaksivirheittäin. Tarkemmin tulokset on esitelty luvussa 4. Tulosten merkitys tieteelle ja teollisuudelle, tutkimuksen rajoitteet sekä mahdolliset jatkotutkimusaiheet on käyty läpi luvussa 5 ja lopuksi yhteenveto on luvussa 6.

2 AIKAISEMPI TUTKIMUS

2.1 Relaatiotietokannanhallintajärjestelmät

Tässä luvussa esitellään mitä tarkoitetaan termillä relaatiotietokannanhallintajärjestelmä sekä siihen liittyvät oleelliset käsitteet.

2.1.1 Tietokanta

Käsitteelle tietokanta on koko joukko hieman toisistaan poikkeavia määritelmiä. Elmasri & Navathe (2015, s. 5) määrittelevät termin seuraavasti:

> Tietokanta on loogisesti johdonmukainen kokoelma tietoja, joille on ominaista merkitys. Satunnaisvalikoimaa dataa ei voida oikeaoppisesti kutsua tietokannaksi. Tietokanta on suunniteltu, rakennettu ja täytetty datalla tiettyä tarkoitusta varten. Sillä on tarkoitettu käyttäjäryhmä ja joitain sovelluksia, joista nämä käyttäjät ovat kiinnostuneita.

Darwen (2009, s. 14) puolestaan näin:

> Tietokanta on järjestetty, koneella luettava symbolikokoelma. Tietokanta on myös koneella-päivitettävä, joten sen on oltava myös muuttujien kokoelma. Tietokanta on tyypillisesti jonkin käyttäjäyhteisön käytettävissä. Käyttäjien tarpeet voivat vaihdella.

2.1.2 Tietomalli

Tietomalli määrittää mitä dataa tietokanta sisältää ja kuinka se on jäsennetty. Simsion & Witt (2005, s. 16-17) mukaan tietomallilla on kolme tasoa, joista kukin kuvaa eri abstraktiotasoa siirryttäessä liiketoimintamäärityksistä kohti toimivaa tietokantaa. Nämä tasot ovat käsitemalli (conceptual model), looginen malli (logical model) sekä fyysinen malli (physical model). Simsion & Witt (2005, s. 16-17) ja luonnehtivat tasoja seuraavasti: Käsitemalli määrittelee tiedon toteutusriippumattomalla tavalla. Käsitemallin painopiste on viestinnässä mallintajan ja liiketoimintasidosryhmien välillä. Looginen malli kuvaa tiedon rakenteina, jotka voidaan toteuttaa tietokannanhallintajärjestelmällä. Fyysinen malli määrittää tiedon säilytyksen, suorituskykyyn ja pääsynhallintaan liittyviä tietoja. (Simsion & Witt, 2004, s. 16-17). Elmasri & Navathe (2015, s. 33) kutsuu keskimmäistä tasoa, johon relaatiomallinkin luetaan kuuluvan, esitystapatietomalliksi (representational data model) tai toteutustietomalliksi (implementation data model). Tässä tekstissä ko. tietomalliin viitataan jatkossa termillä looginen tietomalli tai looginen taso.

Tietokannan loogista tasoa kuvaavaa mallia kutsutaan myös tietokantamalliksi (database model). Abiteboul, Hull & Vianu (1995, s. 28) mukaan tietokantamalli mahdollistaa tietorakenteiden määrittelyn, ko. rakenteisiin liittyvien

tietojoukkojen rajaamisen ja tiedon käsittelyn. Abiteboul ym. (1995, s. 28) mukaan huomattavimmat tietokannoissa käytetyt rakenteet ovat graafit verkko-, semanttisessa ja oliokeskeisessä mallissa, puut hierarkkisessa mallissa sekä relaatiot relaatiomallissa.

2.1.3 Relaatiomalli

Relaatiomalli on tietomalli, joka sijoittuu loogiselle tasolle. Tässä keskitytään relaatiomallia hyödyntäviin tietokantoihin. Tällaista tietokantaa kutsutaan lyhyesti relaatiotietokannaksi.

Relaatiomallin esitteli Codd (1970) uraauurtavassa artikkelissa "A relational model of data for large shared data banks". Artikkelissa termillä relaatiomalli viitataan tietomalliin, jossa tietorakenteina toimivat relaatiot. Samassa yhteydessä Codd esittää myös teoreettisen perustan kyselykielelle relaatiomuotoiseen tietoon. Codd kutsuu kieltä yleiskäyttöiseksi datakieleksi (universal data (sub)language). Abiteboul ym. (1995, s. 28) mukaan termillä relaatiomalli on sittemmin totuttu viittaamaan laajaan tietokantamallien joukon, joiden tietorakenteet ovat relaatioita ja jotka sisältävät kysely ja päivitystoiminnot sekä eheysrajoitteita (integrity constraints). Sekä Abiteboul ym. (1995, s. 29), että Elmasri & Navathe (2015, s. 149) mainitsevat relaatiomallin eduksi sen yksinkertaisuuden. Relaatiomallin käsitteet on helppo ymmärtää, mikä tekee relaatiotietokannoista käyttökelpoisen laajalle käyttäjäjoukolle. (Abiteboul ym., 1995, s. 29)

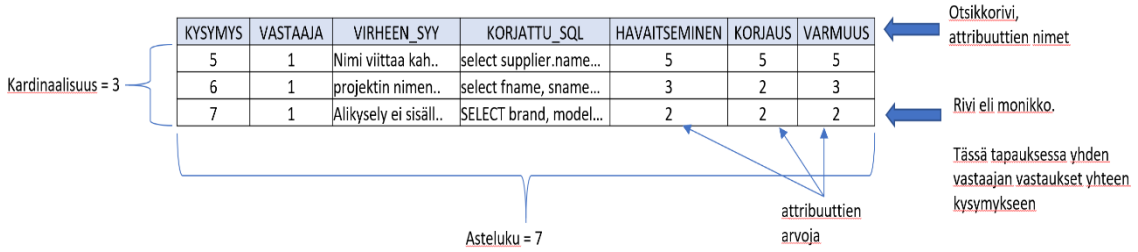
Akateemisessa tietokantakirjallisuudessa relaatiotietokannan perusrakenteisiin liittyviin käsitteisiin viitataan usein hieman eri nimillä kuin ohjelmistoteollisuudessa yleensä, jossa käytetään samoja termejä kuin SQL-kielessä. Seuraavaksi käydään läpi relaatioteorian mukaisten termien lisäksi SQL-pohjaisia termejä, koska empiriaosuudessa tutkittiin juuri SQL:n virheilmoituksia.

2.1.4 Relaatio

Relaatio on matematiikan termi, jolla kuvataan asioiden suhdetta. Joukko-opin termein, jos $R \subseteq X \times Y$ ($X, Y = \emptyset$), niin R on relaatio joukosta X joukkoon Y eli lyhemmin R on joukkojen X ja Y relaatio. Matematiikassa käsitellään usein tällaisia kaksipaikkaisia eli binäärirelaatiota. Relaatiotietokantojen kontekstissa emme ole kuitenkaan rajoittuneet pelkästään kahden välisiin suhteisiin.

Relaatio kuvataan usein tauluna (SQL:ssä table). Taulukkomuoto sopii hyvin relaatioiden esittämiseen. On huomattava, että esitystapana taulu ei ole täydellinen, mutta se on pragmaattinen ja luonteva tapa relaation sisällön ja siihen liittyvien käsitteiden esittämiseen. Taulu koostuu riveistä (SQL:ssä row), joita relaatioteorian (relational theory) mukaan kutsutaan monikoiksi. Relaation rivien alkioiden määrää eli monikkojen pituutta kutsutaan relaation asteluvuksi (degree). Asteluku kertoo, kuinka monipaikkaisesta relaatiosta on kyse. Se voi vaihdella välillä 1-n. Sarakkeet (SQL:ssä column) kuvaavat relaation attribuutteja. Otsikkorivi ilmoittaa, minkä nimisiä attribuutteja taulu sisältää. Kukin taulun solu vastaa pitää sisällään otsikkorivin mukaisen attribuutin arvon. Toisin sanoen

kukin rivi koostuu attribuuttien arvoista. Rivien määrää kutsutaan kardinaalisuudeksi. Se kasvaa, kun tauluun lisätään uusia rivejä ja pienenee kun niitä poistetaan. Relaation keskeisiä käsitteitä on esitelty kuviossa 1. Kyseinen relaatio kuvaa keksittyä esimerkkidataa tutkielman empiriaosan kyselylomakkeesta.



Kuvio 1 Relaation esitys tauluna ja keskeisiä käsitteitä

2.1.5 Relaatiotietokannanhallintajärjestelmä

Elmasri & Navathe (2015, s. 6) mukaan tietokannanhallintajärjestelmä (database management system - DBMS), on yleiskäyttöinen ohjelmisto, joka mahdollistaa tietokannan luonnin ja ylläpidon ja joka auttaa määrittelemään, luomaan, manipuloimaan ja jakamaan tietokantoja useiden käyttäjien ja sovellusten välillä. Tietokannanhallintajärjestelmään viitataan myöhemmin tekstissä lyhenteellä TKHJ. Elmasri & Navathe (2015, s. 6) mukaan edellä mainittu tietokannan määrittely pitää sisällään tietokantaan tallennettavien tietotyyppien, rakenteiden ja rajoitteiden määrittelyn. Lisäksi TKHJ:n vastuulle kuuluu suojata tietokantaa laitteisto- ja ohjelmistotoimintahäiriöiltä sekä estää luvaton pääsy tietokantaan (Elmasri & Navathe, 2015, s. 6). TKHJ mahdollistaa tietokannan ylläpidon tietokantaan kohdistuvien vaatimusten muuttuessa ajan myötä. (Elmasri & Navathe, 2015, s. 6). Relaatiotietokannanhallintajärjestelmä (Relational database management system – RDMS), myöhemmin RTKHJ on TKHJ, jonka hallitsema tietokanta noudattaa loogiselta malliltaan relaatiomallia.

RTKHJ:ssä data ja sen rakenne on eriytetty toisistaan. Tietokannan datan määrittelevää tietoa kutsutaan tietokantakaavaksi (database schema). Tämä metadata tallennetaan systeemitaulustoon (system catalog, catalog). Elmasri & Navathe (2015, s. 37) mukaan useimmat TKHJt tukevat ainakin jossain muodossa kolmitasoista kaavaa (3 schema architecture), jonka tasot ovat: sisäinen kaava (internal schema), käsitekaava (conceptual schema) ja ulkoinen kaava (external schema). Heidän mukaansa jako on seuraavanlainen:

- Sisäinen kaava kuvaa ne rakenteet, joita tietokanta käyttää fyysiseen tiedon tallennukseen.
- Käsitekaava piilottaa fyysiseen tallennukseen liittyvät yksityiskohdat ja keskittyy kuvaamaan tietoa korkeammalla abstraktiotasolla. Loogista tietomallia, (josta Elmasri & Navathe (2015, s. 33) käyttävät termiä representational data model), käytetään käsitekaavan kuvaamisessa.

- Ulkoinen kaava, joita voi olla useita, kuvaa ainoastaan jotain tiettyä tietokannan käyttäjäryhmää palvelevaa dataa piilottaen kaiken muun. Tässäkin pohjana on tyypillisesti loogisen tason tietomalli.

Elmasri & Navathe (2015, s. 39) mukaan tiukka 3-tasoisen kaavan noudattaminen edellyttäisi erillistä tallennusmäärityskieltä (storage definition language, SDL) sisäiselle kaavalle ja näkymämäärityskieltä (view definition language, VDL) ulkoiselle kaavalle, mutta käytännössä useimmat TKHJt käyttävät kaikkien kolmen tason määrittelyyn yhtä tiedon määrityskieltä (data definition language, DDL). Tällöin TKHJ sisältää DDL-kääntäjän (DDL-compiler), joka käsittelee DDL-lauseet, ja tallentaa kaavan määrityksen systeemitaulustoon (Elmasri & Navathe (2015, s. 39). Metatiedon käsittelyn lisäksi tarvitaan kieli itse tiedon noutamiseen, lisäämiseen, poistamiseen ja muokkaamiseen. Tällaista kieltä kutsutaan tiedon käsittelykieleksi (data manipulation language, DML)

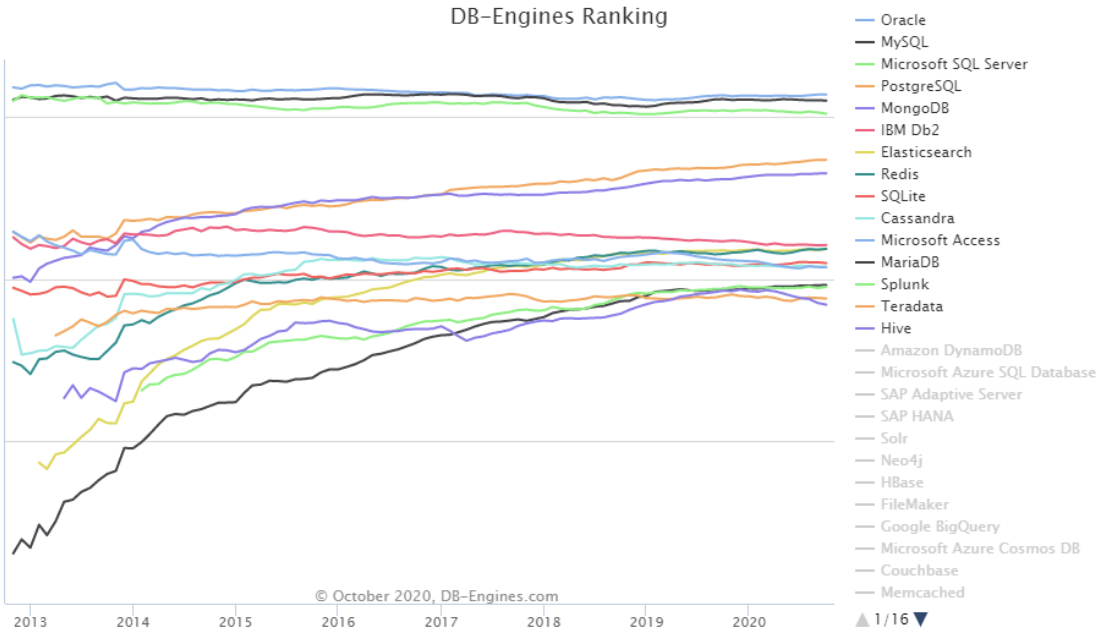
Ensimmäiset kaupalliset RTKHJt ilmaantuivat 1980-luvun alussa. Sittemmin markkinoille on ilmestynyt suuri joukko niin suljetun lähdekoodin kuin avoimen lähdekoodin RTKHJ-tuotteita. Taulukko 1 ja Kuvio 2 kuvaavat TKHJ-ratkaisujen suosiota. Tulokset ovat suuntaa antavia eivätkä mittaa eksakteja asennusmääriä. Suosiolistauksen laskennassa käytetty menetelmä (DB-engines, 2020a) on kuvattu liitteessä 2. Taulukko 1 listaa kirjoitushetken aikaiset (lokakuu 2020) suosituimmat TKHJ-ratkaisut.

Taulukko 1 10 suosituinta TKHJ:ta lokakuussa 2020 (DB-engines, 2020b)

SIJOITUS	TKHJ	TIETOKANTAMALLI
1	Oracle	Relationaalinen, tukee useaa mallia
2	MySQL	Relationaalinen, tukee useaa mallia
3	Microsoft SQL Server	Relationaalinen, tukee useaa mallia
4	PostgreSQL	Relationaalinen, tukee useaa mallia
5	MongoDB	Dokumentti, tukee useaa mallia
6	IBM Db2	Relationaalinen, tukee useaa mallia
7	Redis	Avain-arvo, tukee useaa mallia
8	Elasticsearch	Hakukone, tukee useaa mallia
9	SQLite	Relationaalinen
10	Cassandra	Sarakeperhe (wide column)

Taulukko 1:stä nähdään, että 4 suosituinta TKHJ:tä ovat tietokantamalliltaan relaatioihin pohjautuvia. Lisäksi 10 suosituimmasta TKHJ:sta seitsemän on RTKHJ-ratkaisuja. Tässä tutkielmassa tutkitut avoimen lähdekoodin tietokannat MySQL ja PostgreSQL ovat sijoilla 2 ja 4.

Kuvio 2 kuvaa TKHJ-ratkaisujen suosion kehitystä viimeisen vajaan 9 vuoden aikana. Siitä käy ilmi, että RTKHJ-järjestelmien suosio on pitänyt pintansa jo pitkään. Kärkipaikkaa pitävien Oraclen, MySQL:n ja Microsoft SQL Serverin suosio on säilynyt lähes muuttumattomana koko tarkastelujakson ajan. PostgreSQL:n trendi on tasainen ja nouseva ja jos tilanne jatkuisi samankaltaisena, se tulisi saavuttamaan kärkikolmikon lähivuosien aikana. Liite 2 sisältää kuvauksen siitä, miten sivusto mittaa tietokannanhallintajärjestelmien suosittuutta.



Kuvio 2 TKHJ-tuotteiden suosion kehitys vuodesta 2013 (DB-engines, 2020c).

2.2 SQL-kieli

SQL:ää voidaan pitää yhtenä tärkeimmistä syistä kaupallisten relaatiotietokantojen menestykselle (Elmasri & Navathe, 2015, s. 177). SQL kehitettiin alun alkaen ad hoc tiedon hakemiseen eli ns. kyselykieleksi (query language), mutta nykyisin SQL on kattava ja yhdenmukainen kieli tietokannan luontiin, tietojen turvaamiseen liittyvien määritysten tekemiseen sekä tietojen noutamiseen ja muokkaamiseen (Weinberg, Groff & Oppen, 2009, s. 10).

Kaikki edellisessä luvussa mainitut RTKHJt, mukaan lukien empiriaosuudessa tutkitut MySQL ja PostgreSql hyödyntävät SQL-kieltä edellä mainittuihin tehtäviin.

2.2.1 Historia ja standardointi

Relaatiomallin yhteydessä Codd esitti teoreettisen pohjan kielelle, jonka avulla relaatiodataa voitaisiin manipuloida. Hän kutsui kieltä nimellä DSL/Alpha. IBM toteutti näiden ajatusten perusteella yksinkertaistetun version nimeltä SQUARE, jonka pohjalta kehitettiin kieli nimeltä SEQUEL (Beaulieu, 2020, s. 8). SEQUEL, joka oli lyhenne sanoista *Structured English QUEry Language* tunnetaan nykyään nimellä SQL tai laajemmin *Structured Query language* (Elmasri & Navathe, 2015, s. 178).

SQL on käynyt läpi lukuisia muutoksia viimeisten 40-vuoden aikana (Beaulieu, 2020, s. 8). Kielestä on julkaistu lukuisia virallisia päivityksiä. SQL:n standardointityö tehdään American National Standards Institute (ANSI) ja International Standards Organization (ISO) välisenä yhteistyönä (Elmasri & Navathe,

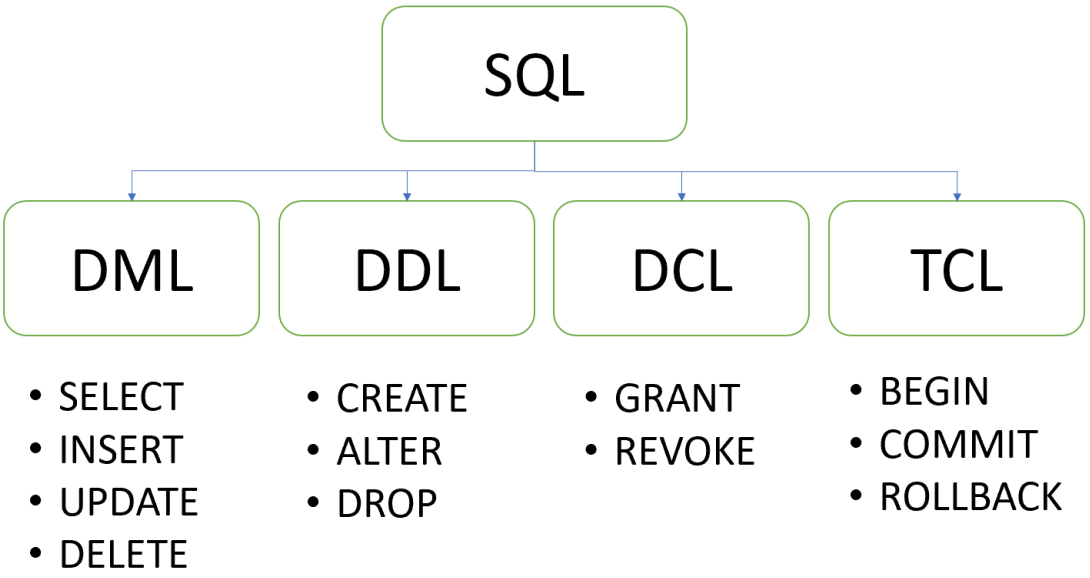
2015, s. 178). Ensimmäisen SQL-standardi julkistettiin vuonna 1986 (nimeltä SQL-86 tai SQL1), josta lähtien kielen kehitys on johtanut uusiin julkaisuihin vuosina 1989, 1992 (nimeltä SQL-92 tai SQL2), 1999 (nimeltä SQL:1999 tai SQL3), 2003, 2011 sekä 2016 (Beaulieu, 2020, s. 8; Elmasri & Navathe, 2015, s. 178). Kirjoitushetkellä voimassa oleva tuorein standardi on kuvattu lähteissä: (International Organization for Standardization, 2016a) ja (International Organization for Standardization, 2016b). Standardiyhteensopivuudesta on tullut tärkeä tekijä TKHJ-tuotteiden valinnassa ja RTKHJ-valmistajat väittävät tuotteensa olevan ISO/ANSI standardin mukaisia tai perustuvan siihen (Weinberg ym., 2009, s. 825). Käytännössä poikkeamia standardista kuitenkin löytyy (Mysql, 2020; PostgreSQL, 2020; Oracle, 2020).

Standardoinnin hyötyjä ovat mm. tietojen siirrettävyys standardia tukevasta RTKHJ:stä toiseen sekä se, että jos tietokantasovellus hyödyntää useampaa kuin yhtä tietokantaa, voidaan niiden kaikkien dataa käsitellä saman kielen avulla. (Elmasri & Navathe, 2015, s. 177). Siirrettävyyden osalta on huomioitava, että eri RTKHJ-ratkaisujen toteutusten välillä on eroavaisuuksia, jotka tulee ottaa huomioon, mutta tilanne yksinkertaistuu, jos molemmat RTKHJ-ratkaisut tukevat standardia ja sovellus käyttää vain standardin mukaisia SQL:n ominaisuuksia (Elmasri & Navathe, 2015, s. 177).

Vuonna 1999 julkaistusta SQL:1999 -standardista lähtien kieli on jaettu ydinmäärittelyyn (core specification) sekä laajennoksiin (extension). SQL-yhteensopivan RTKHJ:n oletetaan toteuttavan ydinmäärittelyn kokonaisuudessaan – laajennoksia voidaan tarjota tai myydä lisäominaisuuksiana spesifeihin käyttötarkoituksiin kuten tiedon louhinta (data mining), aika- (temporal), paikkatieto- (spatial) ja multimediadatan käsittely jne. (Elmasri & Navathe, 2015, s. 178)

2.2.2 SQL-lauseet ja niiden käyttö

SQL on kokonaisvaltainen tietokantakieli, joka sisältää toiminnallisuudet sekä tiedon määrittelyyn, noutamiseen, että päivittämiseen - toisin sanoen SQL on sekä DDL, että DML (Elmasri & Navathe, 2015, s. 178). Lisäksi sillä voidaan määritellä eheysrajoitteita, transaktioiden hallintaa sekä tietoturvaan ja valtuutuksen liittyviä asioita (Elmasri & Navathe, 2015, s. 178). Näiden alkuperäistä standardia merkittävästi laajempien ominaisuuksien vuoksi SQL jaetaan osassa kirjallisuutta neljään eri luokkaan, jotka ovat mainittujen DDL:n ja DML:n lisäksi kannanvalvontakieli (Data Control Language, DCL) sekä transaktionhallintakieli (Transaction Control Language, TCL) (Taipalus & Seppänen, 2020). Yhteenveto osasta SQL:n ominaisuuksista tämän nelijaon mukaisesti Kuviossa 3.



Kuvio 3 SQL:n lauseita nelijaottelun mukaisesti

Empiriaosassa tutkittiin kyselyiden (query), tarkasti ottaen SELECT-lauseiden, aiheuttamia virheilmoituksia. Tässä käytetään Taipalus, Siponen & Vartiainen (2018) kyselyn määritelmä: Kysely on TKHJ:lle lähetetty SQL-lause, jonka pyrkimyksenä on saada vastaus johonkin tietotarpeeseen (data demand). Näin olleen kysely on DML:n osajoukko.

Seuraavassa käydään läpi esimerkkejä SQL-kielen käytöstä. Esimerkit sijoittuvat pääasiassa DML ja DDL alikieliin. Empiriaosuudessa tutkittiin ainoastaan tiedon noutamiseen liittyviä virheilmoituksia.

Käydään SQL-lauseita läpi luomalla kuvitteellinen tietokanta empiriaosuuden datan käsittelyyn MySQL-tietokantaan, joka oli toinen tutkituista avoimen lähdekoodin RTKHJ:sta. (Kuvitteellinen siksi, että todellisuudessa datan käsittelyyn ei käytetty relaatiotietokantaa.) Esimerkkien tarkoituksena ei ole käydä läpi kaikki SQL:n ominaisuuksia vaan esitellä peruskäyttötapauksia.

Luodaan ensin uusi tietokanta/tietokantakaava, jonka pohjalle taulut tullaan luomaan. MySQL:ssä tietokantakaava (schema) ja tietokanta (database) ovat synonyymejä. Joissain RTKHJ:ssä käsitekaava on alisteinen tietokannalle.

Kaikki luontioperaatiot suoritetaan CREATE lauseen avulla. Annetaan uuden tietokannan nimeksi ”tutkimus”. Tämä tehdään tietokannan pääkäyttäjäoikeuksin.

```
CREATE SCHEMA tutkimus;
```

Luodaan uusi käyttäjä nimeltä tutkija, joka tulee suorittamaan tietojen lisäämisen ja muokkauksen.

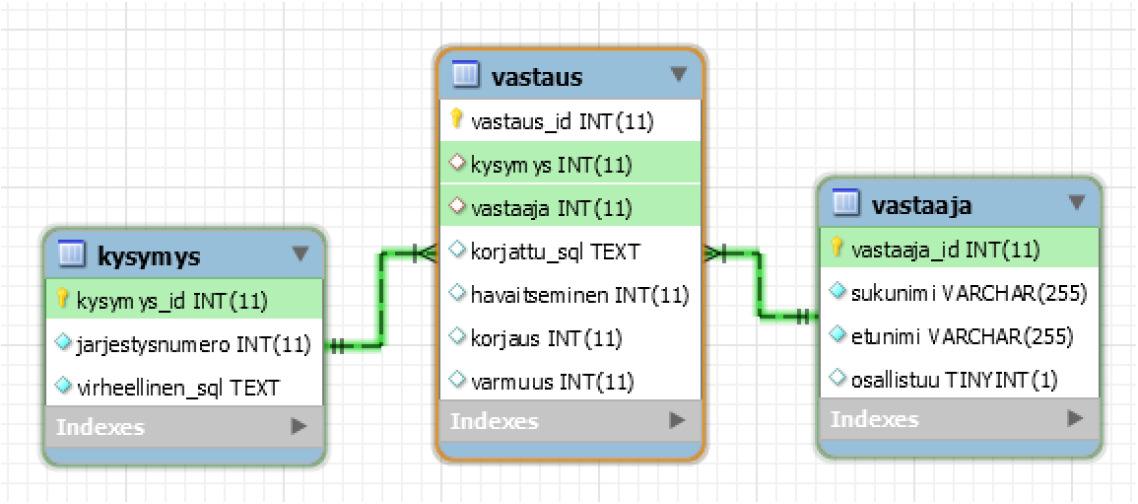
```
CREATE USER 'tutkija'@'localhost' IDENTIFIED BY 'password';
```

Tämä ei vielä yksin riitä, sillä tässä vaiheessa käyttäjältä puuttuu oikeudet tutkimus -tietokannan käsittelyyn. Myönnetään tutkijakäyttäjälle tarvittavat oikeudet

tutkimustietokantaan. Oikeudet myönnetään (DCL-alikielen) GRANT-lausetta käyttäen. Oikeuksia voitaisiin myöntää yksityiskohtaisesti, mutta esimerkin vuoksi myönnetään kaikki oikeudet (ALL PRIVILEGES)

```
GRANT ALL PRIVILEGES ON tutkimus.*
TO tutkija@'localhost';
```

Luodaan 3 taulua tutkimusdatan käsittelyä varten CREATE TABLE lauseen avulla. Luotavat taulut ovat *vastaaja*, *kysymys* ja *vastaus*. Taulut on esitetty kuviossa 4.



Kuvio 4 Esimerkkitaulut

CREATE TABLE lauseen perurakenne on:

```
CREATE TABLE taulun_nimi(
  sarake1 tietotyyppi,
  sarake2 tietotyyppi,
...
);
```

Tietotyyppi määrittelee minkälaista dataa sarake sisältää. SQL-standardi sisältää tietotyyppien määritykset (International Organization for Standardization, 2016a; International Organization for Standardization, 2016b), mutta harva RTKHJ noudattaa niitä tarkasti (Weinberg ym., 2009).

Huomioita seuraavista alla olevista CREATE TABLE lauseista:

- NOT NULL rajoite (constraint) varmistaa, että ko. sarakkeelle on annettava aina arvo riviä lisättäessä tai muokattaessa.
- PRIMARY KEY asettaa taulun pääavaimen ja FOREIGN KEY vierasavaimen. Pääavain on tieto, jonka avulla kukin taulun rivi voidaan tunnistaa yksikäsitteisesti. Vierasavain on viittaus toisen taulun pääavaimeen. Sitä käytetään taulujen yhdistämiseen.

- Vastuu vastaaja_id sarakkeen arvon asettamisesta on siirretty TKHJ:n (MySQL) vastuulle AUTO_INCREMENT määrityksellä. Myöhemmin nähdään, että ko. sarakkeelle ei aseteta eksplisiittistä arvoa riviä lisätessä.
- MySQL:ssä ei ole todellista BOOLEAN arvoa. BOOLEAN tallennetaan kokonaislukutyyppinä. Vertaa TINYTINT Kuvio 5:ssä.

```
CREATE TABLE vastaaja (
    vastaaja_id INT NOT NULL AUTO_INCREMENT,
    sukunimi VARCHAR(255) NOT NULL,
    etunimi VARCHAR(255) NOT NULL,
    osallistuu BOOLEAN,
    PRIMARY KEY (vastaaja_id)
);

CREATE TABLE kysymys (
    kysymys_id INT NOT NULL AUTO_INCREMENT,
    jarjestysnumero INT NOT NULL,
    virheellinen_sql TEXT NOT NULL,
    PRIMARY KEY (kysymys_id)
);

CREATE TABLE vastaus (
    vastaus_id INT NOT NULL AUTO_INCREMENT,
    kysymys INT,
    vastaaja INT,
    korjattu_sql TEXT,
    havaitseminen INT,
    korjaus INT,
    varmuus INT,
    PRIMARY KEY (vastaus_id),
    FOREIGN KEY (kysymys) REFERENCES kysymys(kysymys_id),
    FOREIGN KEY (vastaaja) REFERENCES vastaaja(vastaaja_id)
);
```

Tiedon lisääminen tapahtuu INSERT lauseen avulla.
INSERT lauseen perusrakenne on:

```
INSERT INTO taulun_nimi (sarake1, sarake2, ...)
VALUES (arvo1, arvo2, ...);
```

Lisätään esimerkin vuoksi vastaaja- ja kysymystauluihin.

```
INSERT INTO vastaaja (etunimi, sukunimi, osallistuu)
VALUES ('Elli', 'Esimerkki', TRUE);

INSERT INTO kysymys (jarjestysnumero, virheellinen_sql)
VALUES (1,
 'SELECT MIN(price_usd [...jne]');  ← Lyhennetty esimerkin vuoksi
```

SQL:ssä tiedon haku tapahtuu SELECT lauseen avulla. SELECT-lauseen tulos (result set) on joukko rivejä ja itsessään taulu, mistä seuraa se, että tulosjoukon pohjalta voidaan luoda uusi pysyvä taulu sekä tulosjoukkoa voidaan käyttää syötteenä uusille kyselyille samaan tapaan kuin pysyviä tauluja (Beaulieu, 2020). On huomattavaa, että SQL:n taulu ja SELECT-lauseen tulosjoukko sallii rivit, jotka sisältävät identtiset arvot. Tässä mielessä SQL:n tulosjoukko tai pysyvä

taulu eivät ole relaatioteorian mukainen monikkojen joukko (Elmasri & Navathe, 2015). SQL mahdollistaa kuitenkin joukoksi pakottamisen esimerkiksi DISTINCT avainsanalla.

SELECT-lauseen perusrakenne on:

```
SELECT <joukko_sarakenimiä>
FROM <joukko_tauluja>
WHERE <ehto>
```

<joukko_sarakenimiä> sisältää ne sarakenimet (toisin sanoen attribuuttien nimet), jotka joiden arvot noudetaan tietokannasta ja palautetaan tulosjoukossa.
<from> sisältää ne taulut, jotka ovat osa hakua
<ehto> Totuusarvoinen lause, jonka perusteella tunnistetaan ne rivit, jotka valitaan tulosjoukkoon

Haetaan niiden vastaajien nimet, jotka halusivat osallistua tutkimukseen.

```
SELECT etunimi, sukunimi
FROM vastaaja
WHERE osallistuu IS TRUE;
```

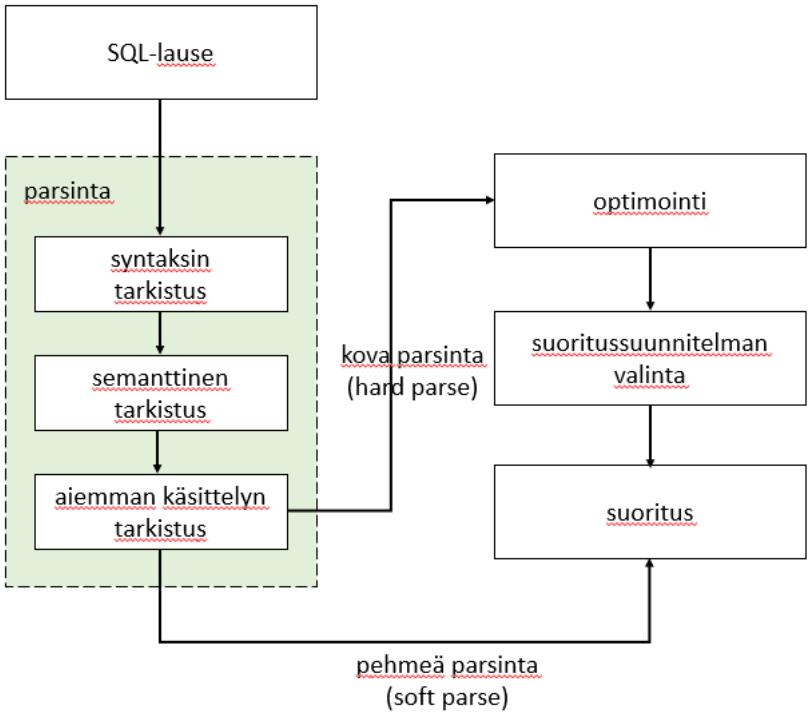
Haetaan kaikki kysymykset, joihin on vastattu korkeimmalla mahdollisella varmuudella vastauksen oikeellisuudesta. Tähän tarvitaan haku, jossa tietoa haetaan useammasta kuin yhdestä taulusta. Toisin sanoen tarvitaan liitos (join), jossa yhdistetään sarakkeita kahdesta tai useammasta eri taulusta liitosehdon mukaisesti. SQL tukee monenlaisia liitostyyppejä. Tässä käytetään sisäliitosta (inner join), joka valitsee ne rivit, jotka esiintyvät *molemmissa* tauluissa ehdon mukaisesti. Toinen liitostyyppi on ulkoliitos, josta on 3 versiota: vasen ulkoliitos (left (outer) join), oikea ulkoliitos (right (outer) join) sekä täysi ulkoliitos (full outer join). Vasen liitos palauttaa kaikki rivit vasemmasta taulusta ja ehdon täyttävät rivit oikeasta. Oikea liitos palauttaa kaikki rivit oikeasta taulusta ja ehdon täyttävät vasemmasta. Täysi liitos palauttaa kaikki rivit, kun ehto täyttyy joko oikeassa tai vasemmassa taulussa. Tämän lisäksi taulun voi liittää itseensä (self join) ja kyselyjen tuloksia voi yhdistää joukko-operaatioilla yhdiste (union), leikkaus (intersect) ja erotus (except tai minus).

```
SELECT kysymys.jarjestysnumero
FROM kysymys
INNER JOIN vastaus ON
kysymys.kysymys_id = vastaus.kysymys   ← Liitosehto
WHERE vastaus.varmuus = 5;
```

Tämän kyselyn tulosjoukko voi sisältää saman kysymyksen useita kertoja, jos useampi vastaaja on vastannut samaan kysymykseen korkeimmalla korjausvarmuudella. Duplikaatit voidaan poistaa lisäämällä hakuun avainsana DISTINCT ennen halutun sarakkeen nimeä.

2.2.3 Kuinka RTKHJ käsittelee SQL-lauseita

Kuvio 5 kuvaa vaiheet, joissa TKHJ käsittelee syötteenä vastaanottamaansa SQL-lausetta. Oracle (2017) kuvaa prosessin seuraavasti. Prosessi alkaa kolmiosaisesta parsintavaiheesta. Syntaksin tarkistus varmistaa, että lause on SQL:n sääntöjen mukaan oikein muodostettu (well-formed). Semanttinen tarkistus varmistaa, että lause on mielekäs, esimerkiksi että lauseen sisältämät taulut ja sarakkeet ovat olemassa. Aiemman käsittelyn tarkistus -vaiheessa (shared pool check) tarkistetaan, onko lause suoritettu aiemmin. Jos näin on, TKHJ voi käyttää aieimmin suoritettua laskentaa ja ohittaa seuraavat resurssi-intensiiviset vaiheet siirtyen suoraan lauseen suoritukseen. Tätä aiemman prosessoinnin hyväksikäyttöä kutsutaan nimellä pehmeä parsinta (soft parse). Jos TKHJ ei voi uudelleenkäyttää aiempaa prosessointia, se joutuu turvautumaan ns. kovaan parsintaan (hard parse) ja lause siirtyy optimointivaiheeseen. Optimointia ei suoriteta DDL-lauseille, ainoastaan DML-lauseille pois lukien tilanne, jossa DML-lause on osa DDL-lausetta. Suoritussuunnitelman valinta tuottaa ajettavan suoritussuunnitelman, jota muu tietokanta voi käyttää. Tämän jälkeen lause on valmiina suoritettavaksi. (Oracle, 2017)

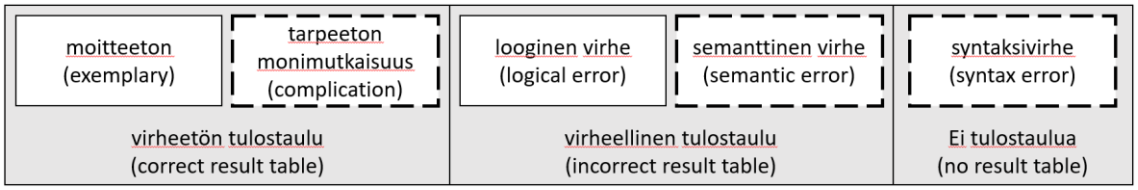


Kuvio 5 SQL-prosessointi (Oracle, 2017)

2.3 Syntaksivirheet

Brass & Goldberg (2005) mukaan syntaksivirhe on seurausta merkkijonosta, joka ei ole validi SQL-haku (query) ja tässä tilanteessa TKHJ palauttaa virheilmoituksen, koska ei voi suorittaa hakua. Taipalus ym. (2018) huomauttaa, että käytännössä tämä ei ole täysin yksiselitteistä, sillä jotkin TKHJ:t tulkitsevat saman SQL-lauseen syntaksivirheeksi ja toiset eivät. Taipalus ym. (2018) tulkitsee syntaksivirheeksi poikkeavuudet SQL-standardista. Tällöin TKHJ ei välttämättä raportoi kaikkia syntaksivirheitä (Taipalus ym., 2018).

Smelcer (1995) ja Brass & Goldberg (2005) luokittelevat SQL-kyselystä seuraavat käyttäjävirheet (user errors) kahteen kategoriaan: TKHJ:n kyselykielen tulkin havaitsemiin syntaksivirheisiin (syntactic errors) ja semanttisiin virheisiin (semantic errors), jotka ovat kieliopillisesti oikein, mutta eivät palauta sitä tietoa, jota kyselyn tekijä tavoitteli. Taipalus ym. (2018) laajentavat tätä jakoa moniulotteisemmaksi (kuvio 6) ottaen huomioon tulostaulun (result table) sekä tietotarpeen (data demand).



Kuvio 6 Virheluokat Taipalus ym. (2018) mukaan.

Seuraavassa käydään läpi, kuinka syntaksivirheet eroavat muista virheistä Taipalus ym. (2018) virheluokittelun mukaan. Taipaluksen ym. (2018) TKHJ-riippumattomassa virheluokittelussa (kuvio 6) erityyppiset virheet (virheluokat) on jaettu kolmeen kategoriaan sen perusteella minkälaisen tulostaulun kysely tuottaa. Lisäksi on tehty jako sen välille, onko virhe tunnistettavissa tietämättä kyselyn tekijän tietotarvetta. Katkoviivalliset laatikot kuvaavat tällaisia virheitä. Vasemmanpuolimmaisin laatikko (moitteeton) kuvaa tilannetta, jossa virhettä tai ongelmaa ei ole lainkaan. Tarpeeton monimutkaisuus on tilanne, jossa kyselystä itsestään käy ilmi, että samaan tietotarpeeseen voisi vastata yksinkertaisemmalla kyselyllä. Esimerkkejä tarpeettomasta monimutkaisuudesta ovat: LIKE-avainsanan käyttö ilman jokerimerkkiä, tarpeeton liitos tai ORDER BY:n käyttö alikyselyssä (Taipalus ym., 2018). Taipalus ym. (2018) viittaa loogisella virheellä semanttiseen virheeseen esim. siinä merkityksessä kuin Smelcer (1995) sen kuvaa, mutta sillä lisämääreellä, että emme voi tietää onko kyseessä virhe tietämättä käyttäjän tietotarvetta. Sen sijaan Taipalus ym. (2018) määritelmän mukainen semanttinen virhe on virhe, joka on tunnistettavissa virheeksi tietämättä kyselyn tekijän intentiota. Taipalus ym. (2018) mukaan esimerkki tällaisesta virheestä on SELECT-lause, jonka WHERE -osa on datasta riippumatta aina epätosi. Viimeisenä luokittelussa on syntaksivirhe, jossa kysely ei palauta tulostaulua, ja joka on tunnistettavissa virheeksi ilman tietoa kyselyn tekijän tietotarpeesta.

Taipalus ym. (2018) jakaa syntaksivirheet edelleen kuuteen eri luokkaan, joihin sisältyy yhteensä 38 erilaista syntaksivirhetyyppiä. Luokat ja virheet on esitetty Taulukossa 2. Tässä tutkimuksessa tarkastellut virheet on esitelty luvussa 3.

Taulukko 2 Syntaksivirheet ja niiden luokat (Taipalus ym., 2018)

ID	Virhe	(engl)
SYN-1	Monitulkintainen tietokantaobjekti	Ambiguous database object
1	puuttuva alias	omitting correlation names
2	monitulkintainen sarake	ambiguous column
3	monitulkintainen funktio	ambiguous function
SYN-2	Tietokantaobjekti puuttuu	Undefined database object
4	saraketta ei ole olemassa	undefined column
5	funktiota ei ole olemassa	undefined function
6	parametria ei ole olemassa	undefined parameter
7	objektia ei ole olemassa	undefined object
8	virheellinen tietokantakaavan nimi	invalid schema name
9	kirjoitusvirheet	misspellings
10	synonyymit	synonyms
11	puuttuvat lainausmerkit merkkijonossa	omitting quotes around character data
SYN-3	Yhteensopimaton tietotyyppi	Data type mismatch
12	sarakkeen toistamatta jättäminen	failure to specify column name twice
13	yhteensopimaton tietotyyppi	data type mismatch
SYN-4	Koostefunktion käyttö väärässä paikassa	Illegal aggregate function placement
14	koostefunktion käyttö muussa yhteydessä kuin SELECT tai HAVING	using aggregate function outside SELECT or HAVING
15	ryhmittelyvirhe: koostefunktioita ei voi käyttää sisäkkäin	grouping error: aggregate functions cannot be nested
SYN-5	Ei-sallittu tai puuttuva ryhmittely	Illegal or insufficient grouping
15	asiaankuulumaton ryhmittelysarake	grouping error: extraneous or omitted grouping column
17	virheellinen HAVING: HAVING ilman GROUP BY:ta	strange HAVING: HAVING without GROUP BY
SYN-6	Yleinen syntaksivirhe	Common syntax error
18	funktion ja funktion parametrin sekoittaminen	confusing function with function parameter
19	WHERE avainsanan käyttö kahdesti	using WHERE twice
20	FROM-avainsanan pois jättäminen	omitting the FROM clause
21	vertailu NULL-arvoon	comparison with NULL
22	puuttuva puolipiste	omitting the semicolon
23	aika-arvon ylitys	date time field overflow
24	lauseen toisto	duplicate clause
25	aliasta ei ole olemassa	using an undefined correlation name
26	liian monta saraketta alikyselyssä	too many columns in subquery
27	taulun ja sarakkeen nimen sekoittaminen	confusing table names with column names
28	rajoite SELECT-lauseessa	restriction in SELECT clause
29	projektio WHERE-lauseessa	projection in WHERE clause
30	avainsanat väärässä järjestyksessä	confusing the order of keywords
31	virhe avainsanan käytön logiikassa	confusing the logic of keywords
32	virhe avainsanan käytön syntaksissa	confusing the syntax of keywords
33	puuttuva pilkku	omitting commas
34	Aalto-, haka- tai puuttuvat sulkeet	curly, square or unmatched brackets
35	IS väärässä yhteydessä	IS where not applicable
36	epästandardit avainsanat tai käyttö väärässä yhteydessä	nonstandard keywords or standard keywords in wrong context
37	epästandardit operaattorit	nonstandard operators
38	ylimääräinen puolipiste	additional semicolon

2.4 Virheilmoitukset

Becker ym. 2019 mukaan ohjelmointivirheilmoitukset (programming error messages) ovat ohjelmointikielten kääntäjien (compilers) ja tulkkien (interpreters) tuottamia diagnostisia virheviestejä. Näihin luetaan esimerkiksi varoitukset, ilmoitukset syntaksivirheistä sekä ajonaikaiset virheet (Becker ym., 2019). Virheilmoitukset ovat pääasiallinen tapa viestiä ohjelmistoviasta kehittäjälle (Barik, 2017). Kölling (2016) ja McCall & Kölling (2019) suosivat termin diagnostinen viesti (diagnostic message) käyttöä kääntäjävirheen (compiler error) sijaan, koska ensimainittu on heidän mukaansa jälkimmäistä termiä yksiselitteisempi. Tässä käytetään pääasiassa termiä kääntäjävirheilmoitukset tai lyhyesti virheilmoitukset.

2.4.1 Virheilmoitusten merkitys

Becker ym. (2019) mukaan ohjelmoijan virheilmoitusten kautta saama palaute on äärimmäisen tärkeää. Ohjelmoijat myös tutkitusti lukevat virheilmoituksia ja käyttävät niiden käsittelyyn paljon aikaa (Barik ym., 2017). Barik ym. (2017) tutkimuksen mukaan ohjelmoijat käyttävät jopa 25% ohjelmointitehtävän suorittamiseen käyttämästään ajasta virheilmoitusten lukemiseen. Erityisen kriittistä virheilmoitusten ymmärtäminen ja tulkinta on vasta-alkajille (Lee & Ko, 2011). Kölling (1999) korosti virheilmoitusten merkitystä opiskelijoille jo yli 20 v. sitten. Köllingin mukaan hyvät virheilmoitukset ovat tärkeitä ohjelmistokehitystyökalun käytettävyydessä aloittelijan kannalta. Virheilmoituksen oikeanlainen muotoilu itsessään voi merkittävästi auttaa opiskelijaa ymmärtämään ja korjaamaan pienet virheet ilman ulkopuolista apua. (Kölling, 1999, s.145-146)

2.4.2 Mitä haasteita virheilmoituksiin liittyy?

Becker ym. (2019) mukaan vaikeaselkoiset virheilmoitukset aiheuttavat haasteita vasta-alkajille ja tätä tukevaa kirjallisuutta löytyy vuosikymmenien ajalta. Vaikka kääntäjävirheiden laadun merkitys on hyvin tiedossa, ei niistä ole juurikaan ole ollut paljon hyvää sanottavaa (Becker ym., 2019). Becker ym. (2019) mukaan virheilmoituksia on kutsuttu kirjallisuudessa riittämättömiksi, epäoptimaalisiksi, epäymmärrettäviksi, hyödyttömiksi, kryptisiksi, hämmentäviksi, pahamaineisen vaikeaselkoisiksi, pelottaviksi, käsittämättömiksi, vähemmän hyödyllisiksi kuin mitä ne voisivat olla sekä kehityksen esteiksi.

Hankalat virheilmoitukset eivät pelkästään vaikeuta virheiden teknistä korjaamista vaan aiheuttavat myös hämmennystä ja tuskastumista (Becker ym., 2019). Virheilmoitusten aiheuttamille negatiivisille tunteille ja niiden haittavaikutuksille löytyy muitakin mainintoja kirjallisuudesta. Lee & Ko (2011)

tutkimuksessa antropomorfisten virheilmoitusten käytöstä ohjelmoinnissa esitettiin, että aloittelevat ohjelmoijat tulkitsevat sävyltään kylmät virheilmoitukset henkilökohtaisiksi epäonnistumisiksi mikä madaltaa ohjelmoijan motivaatiota tehtävän suorittamiseen.

Vasta-alkajien lisäksi muillakin ohjelmoijilla on intressejä parempia virheilmoituksia kohtaan. Becker (2019) mainitsee parempien virheilmoitusten hyödyttävän ammattilaisia vasta-alkajien lisäksi. Lisäksi ammattilaisten siirtyessä uuteen ohjelmointikieleen, heistä tulee jossain mielessä uudelleen vasta-alkajia (Traver, 2010). McCall & Kölling (2019) mukaan kääntäjä voi hieman kontekstista riippuen tuottaa erilaisia virheilmoituksia samaan virheeseen sekä loogisesti eri virhe voi saada aikaan saman virheilmoituksen. Saman kääntäjän sisäinen epäjohdonmukaisuus on haastavaa ja vielä hankalammaksi asian tekevät kääntäjien väliset erot: eri kääntäjä tai saman kääntäjän eri versiot voivat tuottaa eri virheilmoituksia samaan virheilmoitukseen (McCall & Kölling, 2019).

McCall & Kölling (2019) mukaan sekä ajonaikaiset virheet, että kääntäjävirheet ovat tärkeitä ja molemmat tuottavat kokemattomille ohjelmoijille vaikeuksia, mutta erityisesti kääntäjävirheet ovat todellinen este (McCall & Kölling (2019) käyttävät termiä ”show stopper”), koska kääntymätön ohjelma ei anna opiskelijalle minkäänlaisia tuloksia.

Hyödyllisten virheilmoitusten luominen ei ole helppoa. Horning (1976, s.544) mukaan kääntäjän kehittäjälle merkittävä haaste on se, että on mahdotonta varmuudella tietää, missä virhe todellisuudessa sijaitsee. Turvallisinta on yrittää kuvata tilannetta mahdollisimman selvästi ennen yritystä tarjota ehdotuksia ongelman ratkaisemiselle (Horning, 1976, s.544). Becker ym. (2019) mukaan ehdotusten tarjoamiseen liittyy se haaste, että kääntäjä ei voi tietää ohjelmoijan intentiota. Virheilmoitus voi ehdottaa korjausta, joka ei vastaa ohjelmoijan aietta (Becker ym., 2019). Traver (2010) myös kritisoi kääntäjien kirjoittajia ja aihepiirin tutkimusta yrityksen puutteesta virheilmoitusten parantamisessa. Edistymistä tapahtuu esimerkiksi kääntäjien nopeudessa, resurssien käsittelyssä sekä uusien ohjelmointikielten ominaisuuksien toteuttamisessa, mutta kääntäjien käytettävyyden parantaminen eli ohjelmoijan työn helpottaminen näyttää aiheuttavan silmiinpistävän vähän huolta (Traver, 2010).

2.4.3 Kuinka virheilmoituksia on tutkittu?

Virheviesteihin liittyvää tutkimusta on löydettävissä yli 50 vuoden ajalta ja etenkin viime vuosien aikana aihepiiriä käsittelevien julkaisujen määrä on lisääntynyt (Becker ym., 2019). Becker ym. (2019) selvittivät mitä kirjallisuudessa on kerrottu ohjelmointivirheilmoituksista. Lopulliseen kokoelmaan päätyi 307 artikkelia. Becker ym. (2019) tunnisti aineistosta joukon ala-aihepiirejä tutkimuksen fokuksen mukaan ja luokitteli kunkin artikkelin näiden perusteella (taulukko 2). Luokittelussa yksi artikkeli voi kuulua useampaan luokkaan. Esim. alla kaikki parannusluokassa mainitut tutkimukset ovat luonteeltaan empiirisiä. Kirjallisuuskatsauksen perusteella kaikkein eniten tutkittu luokka on olemassa olevien virheilmoitusten parantamiseen tähtäävä tutkimus, jota käsitteli 35 % kaikista

tutkimuksista. Seuraavassa on esitetty joitain esimerkkejä ja oleellisia teemoja näistä löydöksistä.

Taulukko 3 Virheilmoituksiin liittyvä kirjallisuus aihepiireittäin (Becker ym., 2019)

Luokka	Kuvaus **Tutkimuksia jotka…**	Artikkeleja yhteensä
parannus **(enhancement)**	ehdottavat muutoksia tai parannuksia olemassa oleviin virheilmoituksiin	107
tekninen **(technical)**	kuvaavat teknisiä haasteita, liittyen parempien virheilmoitusten tarjoamiseen	91
empiirinen **(empirical)**	esittävät empiirisiä tuloksia	89
vaikeudet **(difficulties)**	kuvaavat vaikeuksia, joita ohjelmojilla on virheviestien kanssa	78
perustelu **(justification)**	tarjoavat vahvan perustelun/oikeutuksen ohjelmointivirheiden tutkimukselle	57
esisuositukset **(pre-guidelines)**	tarjoavat ehdotuksia suositusten tekemiseen, mutta eivät sisällä konkreettisia suosituksia	50
opetus **(pedagogy)**	käsittelevät opetuksen ja virheviestien välistä suhdetta	43
suositukset **(guidelines)**	sisältävät eksplisittisiä ja konkreettisia suosituksia ja ohjeita virheviestien suunnitteluun	35
vain virheet **(errors-only)**	käsittelevät virheitä, mutta eivät virheilmoituksia	25
anekdoottinen **(anecdotal)**	tarjoavat anekdoottisia tuloksia virheilmoituksiin liittyen	23
työkalu **(tool)**	kuvaavat työkaluja, jotka käyttävät virheilmoituksia, mutta eivät luo tai paranna niitä.	20
suorituskyky **(performance)**	osoittavat virheiden ja virheilmoitusten välisen yhteyden ohjelmoijan suorituskykyyn	15
ajonaikaiset virheet **(runtime errors)**	käsittelevät ajonaikaisia virheitä	8

Becker ym. (2019) määrittelevät virheilmoituksen parantamisen olevan tekstuaalisen virheviestin muuttamista sellaiseksi, että se auttaa viestin lukevaa ihmistä paremmin korjaamaan virheviestin aiheuttaneen virheen. Parannuksella toimintona he viittaavat työkalun toimintaan, joka parantaa toisen työkalun (kuten IDE tai kääntäjä) tarjoamia virheilmoituksia. Seuraavassa esimerkkejä virheilmoitusten parantamiseen tähtäävästä tutkimuksesta. Denny, Luxton-Reilly &

Carpenter (2014) kehittivät työkalun, joka paransi Javan virheilmoituksia. Tutkimuksessa selvitettiin, kuinka parannetut virheilmoitukset auttavat ohjelmoinnin opiskelijoita tunnistamaan ja korjaamaan virheitä. Hieman yllättäen tässä tutkimuksessa ei löydetty empiirisiä todisteita parannettujen virheilmoitusten hyödyllisyydestä (Denny ym., 2014). Becker (2016) kehitti Java-ohjelmointiin editorin nimeltä Decaf, joka uudelleenmuotoili 30 yleisintä Java-virheilmoitusta. Tutkimukseen osallistui satoja opiskelijoita, joille työkalu näytti yli 50 000 parannettua virheviestiä. Tulokset osoittivat, että työkalu vähensi opiskelijoiden tekemien virheiden määrää sekä näytti helpottavan opiskelijoiden ohjelmointiin liittyviä vaikeuksia (Becker, 2016). Decaf-editoria on käytetty sittemmin useassa tutkimuksessa, joista useat ovat osoittaneet tilastollisesti merkitseviä, mutta eivät erityisen ylivoimaisia virheilmoitusten parantamisen puolesta puhuvia tuloksia (Becker ym. 2019). Ahmed ym. (2019) kehittivät työkalun nimeltä TEGGER, jonka kohderyhmä oli aloittelevat ohjelmoijat. Työkalu liitti virheilmoituksiin korjausehdotuksen, joka perustui 15 000 virhe-korjausparilla opetettuun neuroverkkoon. Työkalun tarkkuus oli 97,7%. Yli 230 osallistujan käytettävyystutkimus osoitti, että työkalua käyttävät opiskelijat ratkaisivat virheen keskimäärin 25% nopeammin kuin opiskelijat, joita avusti ihminen.

Teknisessä luokassa Becker ym. (2019) tunnisti joukon yleisiä ongelmia, joihin kirjallisuudessa on pyritty vastaamaan. Nämä ongelmat Becker ym. 2019 mukaan ovat: 1) täydellisyysongelma (completeness problem): Kaikkea ohjelman virhetoimintaa ei voida havaita luotettavasti. 2) paikannusongelma (locality problem): virheet ilmenevät säännöllisesti etäällä niiden syntypaikasta. 3) yhdistämisongelma (mapping problem): virheet havaitaan esitystavoissa, joita ei voida suoraviivaisesti kytkeä alkuperäiseen koodiin. [esim. virheet generoidussa koodissa]. 4) virheenkäsittelyongelma (engineering problem): Vahva tai kattava virheenkäsittely johtaa hankalasti ylläpidettävään koodiin ja heikompaan suorituskykyyn. 5) etenemisongelma (liveness problem): joissain tilanteessa käännöstyökalut saavat syötteinä osittain suoritettuja ohjelmia, joita ne eivät ole perinteisesti suunniteltu käsittelemään. Edellä kuvatut haasteet ovat läsnä käännösprosessin (compilation pipeline) kaikissa vaiheissa. Aiheiden tutkimus käsittelee mm. parsereiden kehitystä, tyyppipäättelyä ja metaohjelmointia. (Becker ym., 2009)

Suosituksia tarjoava tutkimus on Becker ym. (2019) mukaan edennyt 1960-1990 lukujen lähes pelkästään anekdoottista näyttöä sisältävistä tutkimuksista kohti empiirisesti validoituja tuloksia. Becker ym. (2019) kuitenkin toteaa, että viimeisen 60 vuoden aikana tutkijat ovat tarjonneet pääasiassa samantyyppisiä ohjeita ohjelmointikielten kehittäjille virheilmoitusten parantamiseksi. Kuitenkin vuoden aikana ennen Becker ym. (2019) julkaisua, he toteavat tutkimukseen ilmestyneen uusia suuntia kuten kognitio, rationaalinen argumentointi sekä virheviestien aikajanaesittäminen (timeliness of the presentation of error messages). Becker ym. (2019) tunnisti kirjallisuudesta 10 kategoriaa, joihin suositusta käsittelevä kirjallisuus voidaan jakaa. Taulukko 3 listaa nämä kategoriat sekä kuhunkin kategoriaan kuuluvien tutkimusten määrän. Tutkimus voi kuulua useampaan kuin yhteen kategoriaan eli tarjota suosituksia useammassa kuin yhdessä

kategoriassa. ”Historiallinen” viittaa tutkimuksiin, jotka on julkaistu ennen vuotta 2000.

Taulukko 4 Virheilmoitusten parantamiseen liittyviä suosituksia käsittelevien tutkimusten määrä kategorioittain Becker ym. (2019) mukaan.

	Historiallinen	Anekdoottinen	Empiirinen
Luettavuuden parantaminen	7	14	12
Kognitiivisen kuorman pienentäminen	5	3	11
Kontekstin tarjoaminen	1	13	6
Positiivisen sävyn käyttäminen	7	7	2
Esimerkkien näyttäminen	1	0	8
Ratkaisujen tai vihjeiden näyttäminen	2	8	13
Dynaamisen vuorovaikutuksen mahdollistaminen	3	5	10
Kognitiivisen tuen tarjoaminen (provide scaffolding)	2	3	10
Looginen argumentointi	0	0	2
Oikea-aikainen virheiden raportointi	0	2	2

3 TUTKIMUSASETELMA

3.1 Tutkimuksen tausta

Tässä tutkimuksessa tutkittaviksi tietokannanhallintajärjestelmiksi valittiin MySQL (8.0.12) ja PostgreSQL (12.1), jotka ovat perinteisistä ja maksuttomista relaatiotietokannanhallintajärjestelmistä suosituimmat (DB-Engines, 2020b). Syntaksivirheitä on lukuisia (Ahadi ym. 2016). Siinä missä Ahadi ym. (2016) tutkivat syntaksivirheitä PostgreSQL-tietokannanhallintajärjestelmässä, heidän raportoimansa syntaksivirhekategoriat eivät yleisty muihin tietokannanhallintajärjestelmiin. Tästä syystä syntaksivirheiden tutkiminen on käytetyssä datassa muodostettu tietokannanhallintajärjestelmäriippumattoman virhekategorisoinnin ympärille (Taipalus ym. 2018). Koska syntaksivirheitä on määrällisesti paljon, tässä tutkielmassa tutkitut syntaksivirheet rajattiin kuuteentoista yleisimpään (Taipalus ym. 2018; Taipalus & Perälä 2019). Nämä syntaksivirhetyypit on esitelty kysymyslomakkeella (Liite 1) esiintyvässä järjestyksessä esimerkin kera taulukossa 5. Taulukon 5 sarakkeet *luokka* ja *ID* viittaavat taulukossa 2 esitettyyn Taipalus ym. (2018) syntaksivirheluokittelun. Tässä tutkimuksessa analysoitu data perustuu aiemmin kerättyyn datajoukkoon (Taipalus, 2020).

Taulukko 5 Tutkitut syntaksivirhetyypit Taipalus ym. (2018) luokittelun mukaan

Kysymys	Luokka	ID	Virhe	Esimerkki
1	SYN-1	2	monitulkintainen sarake	*SELECT name FROM a join b ON (a.id = b.aid) WHERE <ehto>;* kun molemmat a ja b sisältävät sarakkeen name
2	SYN-2	11	puuttuvat lainausmerkit merkkijonossa	*SELECT * FROM a WHERE name = Pirkko;*
3	SYN-6	35	IS väärässä yhteydessä	*SELECT * FROM a WHERE name IS 'Pirkko';*
4	SYN-6	32	virhe avainsanan käytön syntaksissa	*SELECT * FROM a WHERE name LIKE ('P', 'K');*
5	SYN-6	31	virhe avainsanan käytön logiikassa	*SELECT * FROM a WHERE <ehto> GROUP BY name ASC;* Ryhmittelyn järjestämisen sijaan
6	SYN-6	26	liian monta saraketta alikyselyssä	*SELECT name FROM a* *WHERE id IN (SELECT id, price FROM B WHERE <ehto>);*
7	SYN-2	4	saraketta ei ole olemassa	*SELECT foobar FROM a WHERE <ehto>;* kun a ei sisällä saraketta foobar
8	SYN-2	9	kirjoitusvirheet	*SELECT * FROM a WHRE <ehto>;*
9	SYN-3	12	sarakkeen toistamatta jättäminen	*SELECT * FROM a WHERE name LIKE 'Pirkko' OR LIKE 'Eveliina';*
10	SYN-4	14	koostefunktion käyttö muussa yhteydessä kuin SELECT tai HAVING	*SELECT name FROM a WHERE age > AVG(age);*
11	SYN-5	15	asiaankuulumaton ryhmittelysarake	*SELECT profession, name, AVG(age)* *FROM a WHERE <ehto> GROUP BY profession;*
12	SYN-6	37	epästandardit operaattorit	*SELECT * FROM a WHERE name == 'Pirkko';*
13	SYN-6	19	WHERE avainsanan käyttö kahdesti	*SELECT * FROM a WHERE name = 'Pirkko' WHERE age = 35;*
14	SYN-6	36	epästandardit avainsanat tai käyttö väärässä yhteydessä	*SELECT name FROM a WHERE age = 35* *AND JOIN (SELECT * FROM b);*
15	SYN-2	10	synonyymit	*SELECT fname FROM a WHERE <ehto>;* kun a ei sisällä saraketta fname vaan name;
16	SYN-6	34	Aalto-, haka- tai puuttuvat sulkeet.	*SELECT name FROM a* *WHERE id IN (SELECT id FROM B WHERE <ehto>;* *Sulkeva sulje puuttuu↑*

Koehenkilöt (n = 75) vastasivat kysymyksiin verkkovälitteisesti. Jokainen koehenkilö vastasi kysymyksiin koskien 16 syntaksivirhettä. Ennen kysymyksiin vastaamista koehenkilöt olivat osallistuneet koulutukseen relaatiotietokantojen rakenteesta, relaatioalgebrasta ja SQL-kielestä. SQL-kielen koulutuksessa käytettiin SQLite-tietokannanhallintajärjestelmää. Tutkimukseen osallistuminen oli vapaaehtoista, eikä koehenkilöitä palkittu tutkimukseen osallistumisesta. Ennen kysymyksiin vastaamista koehenkilöille tiedotettiin datan käyttötarkoituksesta, tallentamisesta ja anonymisoinnista.

3.2 Tutkimuskysymykset

Tässä tutkielmassa tarkastellaan relaatietokannanhallintajärjestelmien ilmoittamien virheilmoitusten hyödyllisyyttä kolmesta eri näkökulmasta.

Tutkimuskysymykset ovat: Miten hyödyllisiä tietokannanhallintajärjestelmien SQL-virheilmoitukset koetaan 1) virheen havaitsemisen kannalta 2) virheen korjaamisen kannalta sekä 3) korjausvarmuuden kannalta.

Korjausvarmuus viittaa varmuuteen, jolla virheen korjaaja kokee korjatun SQL-lauseen olevan toimiva. Valituista tutkimuskysymyksistä voidaankin todeta, että ne mittaavat virheilmoitusten ominaisuuksia koehenkilön subjektiiviseen kokemukseen perustuen. Kyselylomakkeen rakenne on esitetty Liitteessä 1.

3.3 Tutkimusmenetelmä

Analysoitavan datan ominaisuudet ja datan keräystapa vaikuttavat tutkimusmenetelmän valintaan. Tässä tutkielmassa menetelmänä käytetään epäparametrista Mann-Whitneyn U-testiä, koska *1)* tavoitteena on vertailla ryhmien välisiä eroja, *2)* datassa on (testiä kohden) yksi järjestysasteikollinen riippuva muuttuja, *3)* datassa on yksi kaksiryhmäinen riippumaton muuttuja ja *4)* havainnot ovat toisistaan riippumattomia. Testi ajetaan kullekin tutkimuskysymykselle (virheen *havaitseminen*, *korjaaminen* ja *varmuus*) käyttäen jokaisesta syntaksivirhetyypistä muodostettuja summamuuttujaa. Näiden testien lisäksi raportoidaan deskriptiivisiä tilastoja syntaksivirhekohtaisesti.

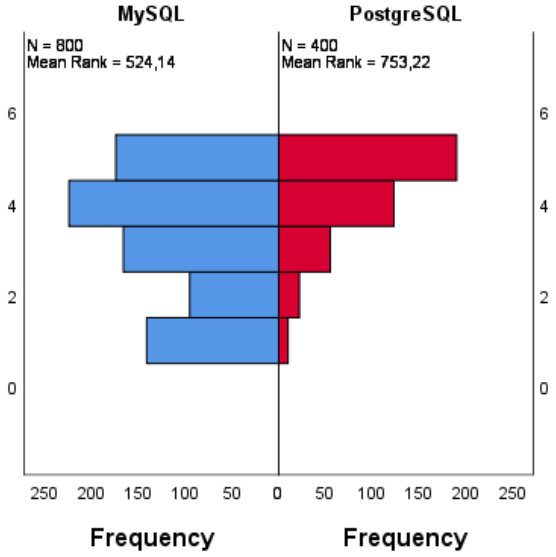
4 TULOKSET

4.1 Vertailu tietokannanhallintajärjestelmittäin

Mann-Whitneyn U-testin tulosten tulkintaan vaikuttaa ratkaisevasti vertailtavien ryhmien mediaanien jakauma. Summamuuttujien (*h, k, v*) mediaanien jakaumien silmämääräisen tarkastelun perusteella jakaumat eivät olleet muodoltaan samat, minkään kolmen summamuuttujan osalta (Kuviot 7, 8 ja 9). Tästä johtuen valittua testiä käytettiin jakaumien erojen tarkasteluun:

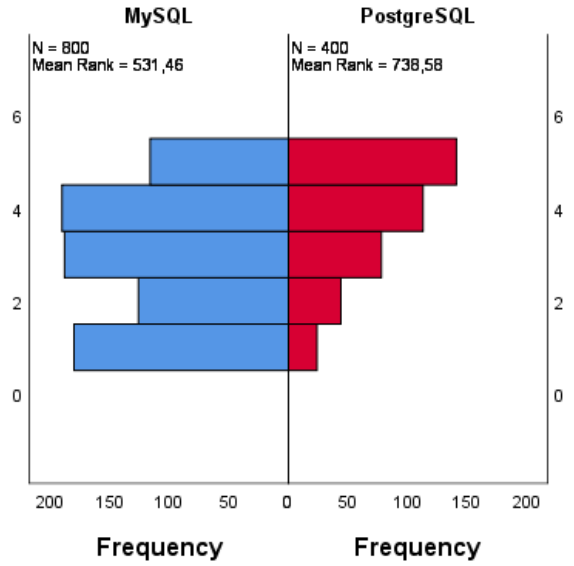
H_0: populaatioiden jakaumat ovat yhtäläiset.

Mann-Whitneyn U-testiä käytettiin selvittämään, onko syntaksivirheen koetussa *havaitsemisessa (h)* eroja MySQL- ja PostgreSQL-tietokannanhallintajärjestelmien virheilmoitusten välillä. Havaitsemisien jakaumat olivat silmämääräisen tarkastelun perusteella erilaiset. Havaitsemiset MySQL-tietokannahallintajärjestelmässä (järjestyksen keskiarvo 524,14) olivat tilastollisesti merkitsevästi pienempiä kuin PostgreSQL-tietokannahallintajärjestelmässä (järjestyksen keskiarvo 753,22), U = 221089, z = 11,141, p < .001.



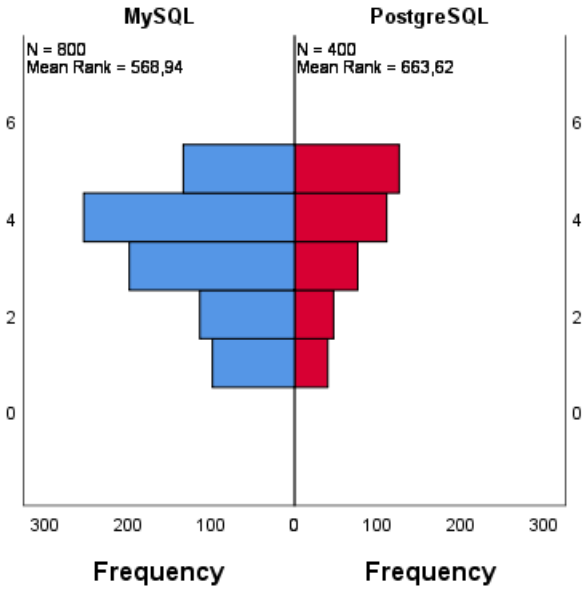
Kuvio 7 Mann-Whitneyn U-testin tulokset virheilmoituksen koetulle havaitsemiselle

Mann-Whitneyn U-testiä käytettiin selvittämään, onko syntaksivirheen koetussa *korjaamisessa (k)* eroja MySQL- ja PostgreSQL-tietokannanhallintajärjestelmien virheilmoitusten välillä. Korjaamisien jakaumat olivat silmämääräisen tarkastelun perusteella erilaiset. Korjaamiset MySQL-tietokannahallintajärjestelmässä (järjestyksen keskiarvo 531,46) olivat tilastollisesti merkitsevästi pienempiä kuin PostgreSQL-tietokannahallintajärjestelmässä (järjestyksen keskiarvo 738,58), U = 215231, z = 9,985, p < .001.



Kuvio 8 Mann-Whitneyn U-testin tulokset virheilmoituksen koetulle korjaamiselle.

Mann-Whitneyn U-testiä käytettiin selvittämään, onko syntaksivirheen koetussa korjaus*varmuudessa (v)* eroja MySQL- ja PostgreSQL-tietokannanhallintajärjestelmien virheilmoitusten välillä. Varmuuksien jakaumat olivat silmämääräisen tarkastelun perusteella erilaiset. Varmuudet MySQL-tietokannahallintajärjestelmässä (järjestyksen keskiarvo 568,94) olivat tilastollisesti merkitsevästi pienempiä kuin PostgreSQL-tietokannahallintajärjestelmässä (järjestyksen keskiarvo 663,62), $U = 185249$, $z = 4{,}588$, $p < .001$.

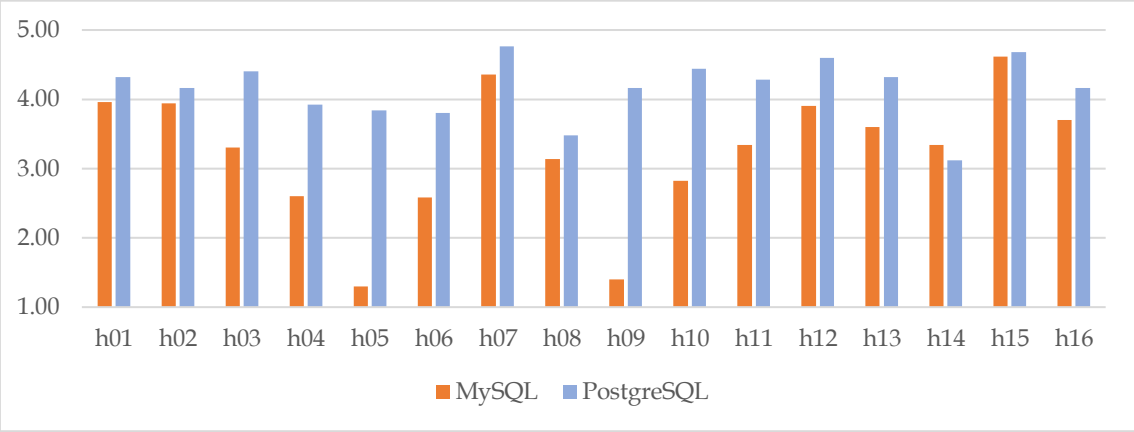


Kuvio 9 Mann-Whitneyn U-testin tulokset virheilmoituksen koetulle korjausvarmuudelle.

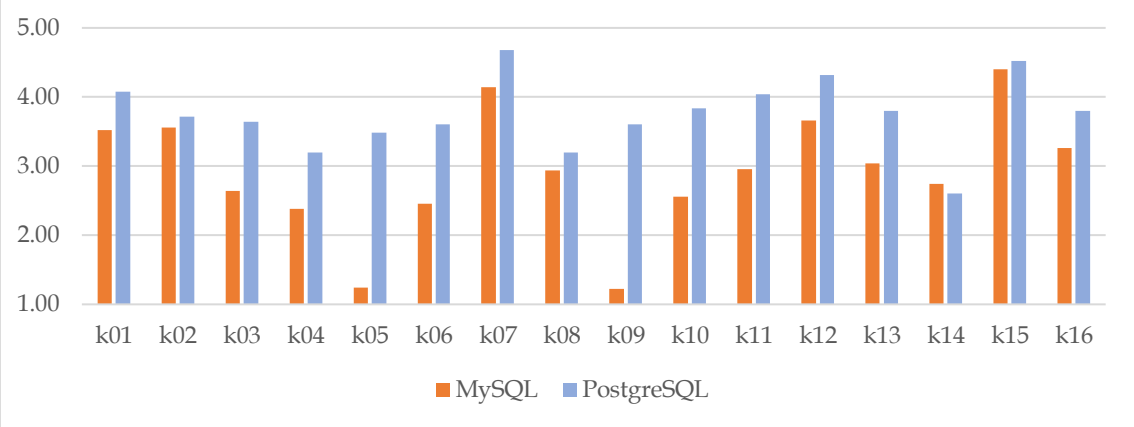
Jokaisessa testissä *p*-arvo on ilmoitettu eksaktina otantajakaumana *U*:n suhteen. Yhteenvetona Mann-*Whitneyn* U-testin nollahypoteesi voidaan hylätä kaikkien kolmen tutkimuskysymyksen osalta. MySQL-tietokannahallintajärjestelmän virheilmoitukset koettiin kaikilla kolmella mittarilla mitattuna PostgreSQL-tietokannahallintajärjestelmän virheilmoituksia heikommiksi, ja ryhmien väliset erot olivat tilastollisesti merkitseviä.

4.2 Vertailu syntaksivirheittäin

Tähän alalukuun on koottu virheiden havaitsemiseen, korjaamiseen ja korjausvarmuuteen liittyvien, Likert-asteikolla mitattujen riippuvien muuttujien keskiarvojen vertailut syntaksivirheittäin (Kuviot 10, 11 ja 12). Kuvioista havaitaan, että PostgreSQL:n virheilmoitukset koettiin kaikilla kolmella mittarilla lähes jokaisessa syntaksivirheessä hyödyllisimmäksi kuin MySQL:n.



Kuvio 10 Virheilmoituksen koettu hyöty virheen havaitsemiseen - keskiarvot syntaksivirheittäin.



Kuvio 11 Virheilmoituksen koettu hyöty virheen korjaamiseen - keskiarvot syntaksivirheittäin.

Kuvio 12 Virheilmoituksen koettu hyöty virheen korjaamisvarmuuteen - keskiarvot syntaksivirheittäin.

5 POHDINTA

Tämä tutkimus osoitti, että avoimen lähdekoodin MySQL ja PostgreSQL tietokannanhallintajärjestelmien raportoimissa virheilmoituksissa on selkeästi havaittavia eroja. PostgreSQL oli parempi virheen koetussa havaitsemisessa ja korjaamisessa sekä korjausvarmuudessa. Ero olivat jopa yllättävän suuri ja konsistentti kaikkien tutkimuskysymysten osalta.

Tutkimuksen aineiston vahvuutena voidaan pitää sitä, että käytetyn SQL-standardia noudattavien tehtävien vuoksi kysymykset oli mahdollista vakioida, siten, että virheilmoitusten vaikutus tuloksiin voitiin eristää luotettavasti. Sekä tehtävänanto että oikeat vastaukset olivat molempien TKHJiden osalta täsmälleen samat - ainoa ero olivat TKHJ:n raportoimat virheilmoitukset. Näyttö oli puhtaasti empiiristä eikä anekdoottista tai asiantuntija-arvioon perustuvaa, kuten osassa aiempaa virheilmoituskirjallisuutta Becker ym. (2009) on ollut.

5.1 Tulosten merkitys tieteelle

Becker ym. (2009) mukaan eniten käsitelty virheilmoitustutkimuksen osa-alue on virheilmoitusten parantaminen. Huomiota ovat saanut myös virheilmoitusten raportointiin liittyvät tekniset haasteet, virheilmoitusten nykytila ja erilaiset suositukset parempien virheilmoitusten luomiseksi. Vaikuttaa siltä, että kattava selvitys virheilmoituksia tuottavien kääntäjien keskinäisistä eroista ei ole saanut yhtä paljoa huomioita. Erityisesti kirjallisuuskatsauksessa ei löytynyt vastaavia tutkimuksia suosittujen tietokannanhallintajärjestelmien virheilmoitusten välisistä eroista. Tässä mielessä tämä tutkimus tuotti uutta tietoa. Laajempi selvitys on kuitenkin paikallaan paremman kokonaiskuvan saamiseksi suosituimpien tietokannanhallintajärjestelmien välisistä eroista. Tämä tutkimus toimii hyvänä alkuna.

Tulokset vahvistivat osaltaan monen lähteen, mm Becker ym. (2009) huomiota siitä, että opiskelijoiden virheilmoitusten kautta saama palaute on tärkeää. Tulokset tukivat myös suoraan Kölling (1999, s.145-146) huomiota siitä, että virheilmoituksilla on merkitystä siinä, onnistuuko opiskelija korjaamaan virheen itsenäisesti.

5.2 Tulosten merkitys teollisuudelle

Tässä tutkimuksessa tarkastellut virheiden havaitseminen, korjaaminen ja korjausvarmuus ovat kaikki asioita, joiden kehittämisestä jokainen TKHJ hyötyisi. On helppo arvioida, että tehokkaampi virheiden havaitseminen ja korjaaminen johtaisi positiivisiin tehokkuus- ja tulosparannuksiin, niin yksittäisten käyttäjien kuin TKHJ:iä hyödyntävien organisaatioiden osalta. Erityisen suoraa hyötyä

näistä tuloksista on TKHJ-kohtaisesti ongelmallisiksi todettujen virheilmoitusten kehittämisessä.

On toivottavaa, että tämänkaltaiset tulokset sekä rohkaisisivat, että loisivat painetta parantaa nykyisiä kääntäjätoteutuksia teollisuudessa koskien yhtä lailla TKHJ:den DML-kääntäjiä kuin perinteisten ohjelmointikielten kääntäjiä. Rohkaisisivat sikäli, että hyvät tulokset kannustavat panostamaan virheilmoituksiin jatkossakin ja loisivat positiivista painetta siinä mielessä, että jos on selkeästi toteen näytettävissä, että kilpailevat tuotteet ja ratkaisut ovat näissä käyttäjien suoriutumiseen vaikuttavissa ominaisuuksissa parempia, on TKHJ-kehittäjillä motivaatiota saattaa ratkaisunsa vähintään kilpailijoiden tasolle.

5.3 Tutkimuksen rajoitteet

On huomattava, että vaikka koehenkilöitä oli kokonaisuudessaan 75, vastaajat eivät jakautuneet tasaisesti kummankin tarkastellun TKHJ:n osalta. PostgreSQL-vastaajia (n = 50) oli kaksinkertainen määrä MySQL-vastaajiin nähden (n = 25). Ryhmien erisuuruus vaikuttaa tulosten yleistettävyyteen, mutta valitulle Mann-Whitney U-testille ryhmien yhtäsuurta kokoa ei vaadita.

Tässä tutkimuksessa tutkittiin vain yhtä versiota kummastakin TKHJ:sta ottamatta huomioon mahdollisia eroja saman TKHJ:n eri versioiden välillä. Toisin sanoen tulosten pohjalta ei voida aukottomasti väittää, että PostgreSQL on tutkimuskysymysten osalta kaikkien tai useimpien versioiden osalta vahvempi kuin MySQL.

Lisäksi on mainitsemisen arvoista, että kolmannen tutkimuskysymyksen osalta (*korjausvarmuus*) kysymyksen muotoilu vastauslomakkeella ei noudattanut samaa kaavaa kuin kahdessa ensimmäisessä (*havaitseminen* ja *korjaaminen*). Vertaa *"Miten hyödyllinen virheilmoitus oli virheen havaitsemisen/korjaamisen kannalta?"* ja *"Miten varma olet oman korjatun SQL-lauseesi toimivuudesta?"*. Osallistujia ei siis tarkasti ottaen pyydetty arvioimaan korjausvarmuutta suhteessa virheilmoitukseen samaan tapaan kuin havaitsemista ja korjaamista. Toisaalta kuten jo todettua, ainoa ero kussakin kysymyksessä MySQL- ja PostgreSQL-ryhmien välillä oli TKHJ:n virheilmoitus, joten voidaan olettaa, että muita korjausvarmuuteen vaikuttavia tekijöitä ei ryhmien välillä ollut. Vaikka tutkimuksessa päädyttiin mittaamaan virheilmoitusten ominaisuuksia vain osallistujien subjektiiviseen kokemukseen perustuen, on varmuuden osoitettu tavallisesti korreloivan objektiivisen onnistumisen kanssa (Carlebach, 2020). Tästä huolimatta virheilmoitusten hyödyllisyyden mittaaminen myös objektiivisilla mittareilla olisi kuitenkin vahvistanut tutkimuksen tuloksia.

5.4 Jatkotutkimus

Koska selvää näyttöä virheilmoitusten hyödyllisyyden eroista löytyi ja käsitys TKHJ-ratkaisujen keskinäisistä eroista aiheesta on puutteellinen, olisi hyödyllistä jatkaa vertailua muiden TKHJ:den osalta. Ensin suosittujen SQL-standardia noudattavien RTKHJ-ratkaisujen osalta, missä looginen seuraava askel olisi tutkia, kuinka kärkinelikon (Taulukko 1) kaksi muuta tuotetta Microsoft SQL Server ja Oracle suoriutuisivat vastaavassa tutkimuksessa. Tämän jälkeen selvitettävää on paljon. Merkittävä osa db-engines (2020b) listaamista 345:sta suosituimmista TKHJ:sta (db-engines, 2020b) on RTKHJ-ratkaisuja. Lisäksi viimeaikainen NewSQL-tietokantojen ilmaantuminen (Taipalus ym., 2018) laajentaa kenttää entisestään. NewSQL-tietokantojen virheilmoitusten hyödyllisyyden arvioinnissa voitaisiin hyödyntää samaa Taipalus ym. (2018) luomaa TKHJ-riippumatonta virheluokittelua kuin tässä tutkimuksessa. Kunnianhimoisempi tavoite olisi laajentaa selvitystyö suosituimpien tietokantojen osalta muihinkin kuin relaatiotietokantoihin. Tässä yhteydessä olisi myös arvokasta selvittää, onko tai missä määrin Taipalus ym. (2018):n TKHJ-riippumaton virheluokittelu on yleistettävissä muille loogisille malleille pohjautuville TKHJ:lle.

Tässä tutkimuksessa selvitettiin virheilmoitusten hyödyllisyyttä Taipalus ym. (2018) syntaksivirheluokittelusta (Taulukko 2) poimittujen 16 syntaksivirheen osalta. Saman selvityksen tekeminen tämän tutkimuksen ulkopuolelle jätetyille virheilmoituksille (Taulukko 2) voisi myös tuoda uutta tietoa. Etenkin olisi arvokasta selvittää onko eri syntaksivirheillä keskinäisiä merkityseroja. Tähän tutkimukseen virheet valittiin esiintymistiheyden perusteella, mutta onko jotain muita oleellisia kriteereitä kuin yleisyys? Onko jokin virhe esiintyessään erityisen vaikea korjattava tai aiheutuuko siitä muuta poikkeuksellista haittaa? Tällaisesta tiedosta voisi olla hyötyä paitsi opetuksessa, mutta myös priorisointiapuna virheilmoitusten kehittämisessä.

Kirjallisuuskatsauksen perusteella akatemiassa on tutkittu erityisen paljon vasta-alkajia. Tämä on ymmärrettävää ja arvokasta. Ymmärrettävää sikäli, että ohjelmistotuotantoa opetetaan yliopistoissa ja osallistujia on runsaasti saatavilla. Arvokasta siksi, että oppimisen ymmärtäminen ja kehittäminen auttaa parantamaan opetusta. Tämän lisäksi olisi kuitenkin arvokasta selvittää paremmin, kuinka virheilmoitusten kehittämisellä, mukaan lukien havaitseminen, korjaaminen ja korjausvarmuus, voitaisiin hyödyttää ammattilaisia, joiden työhön kääntäjävirheiden käsittely kuuluu olennaisesti. Tähän on helppo esittää syitä. Yksi on puhtaasti taloudellinen. Ohjelmistot ja tietokannat ovat läsnä kaikkialla. Pienelläkin tehokkuusparannuksella voisi olla tuntuvia vaikutuksia. Toisenlainen lähestymiskulma voisi olla, miten koetut virheet vaikuttavat ammattilaiskäyttäjiin. Mm. Lee & Ko (2011) sivusivat virheilmoitusten negatiivista vaikutusta ohjelmoijien psyykeen. Jokainen ohjelmistokehityksen parissa uraa tehnyt tietää kuinka stressaavaa virheiden selvittely voi kiireisessä ohjelmistoprojektissa olla. Tätä taustaa vasten kaikki parannukset olisivat tervetulleita.

Taipalus ym. (2018) virheluokittelu perustuu tutkimukseen, jossa koehenkilöinä toimivat opiskelijat. Teollisuutta hyödyttävä ammattilaisten työtä potentiaalisesti auttava tutkimuskohde olisi selvittää, mitä virheitä ammattilaiset tekevät eniten ja mitkä heidän tekemistä virheistä ovat erityisen haitallisia ja edelleen minkälaiset virheilmoitukset parhaiten hyödyttäisivät tätä joukkoa.

Becker ym. (2009) mukaan virheilmoitukset ovat tärkeitä järjestelmän kanssa käytävän vuorovaikutuksen tehostajia, mutta käytössä on hyvin vähän keinoja tämän vuorovaikutukseen mittaamiseen (Becker ym., 2009). Mahdollinen tutkimuksen aihe olisi selvittää, onko jo tämän tutkimuksen aineiston perusteella löydettävissä vastauksia siihen, mitkä seikat tutkittujen TKHJ:den virheilmoituksissa aiheuttavat eroja havaitsemisessa, korjaamisessa ja korjaamisvarmuudessa.

6 YHTEENVETO

Virheilmoituksilla on todellisia vaikutuksia todellisille ihmisille. Erityisen kriittisiä ne ovat aloittelijoille, mutta tärkeitä myös ammattilaisille. Virheilmoitusten hyödyllisyys jättää edelleen paljon toivomisen varaa, vaikka aihetta on tutkittu pitkään. Suurin osa tutkimuksesta keskittyy ohjelmointikielten virheilmoituksiin. Tietokannanhallintajärjestelmien raportoimia virheilmoituksia on tutkittu vähemmän. Tässä tutkimuksessa tutkittiin kahden suosituimman relaatiotietokannanhallintajärjestelmän, PostgreSQL:n ja MySQL:n virheilmoitusten hyödyllisyyttä. Tutkimus suoritettiin verkkopohjaisena kyselynä, jossa opiskelijoille esitettiin virheellinen SQL-lause ja tietokannanhallintajärjestelmän raportoima virheilmoitus. Vastaajien tehtävä oli korjata SQL-lause ja arvioida virheilmoituksen hyödyllisyyttä virheen havaitsemisessa ja korjaamisessa sekä arvioida kuinka varmoja he olivat korjauksen oikeellisuudesta. Kerätyn datan tilastollinen analyysi osoitti, että erot tietokannanhallintajärjestelmien välillä olivat tilastollisesti merkitseviä ja että PostgreSQL:n virheilmoitukset koettiin hyödyllisemmäksi virheen havaitsemisessa ja korjaamisessa ja myös korjausvarmuus oli parempi PostgreSQL:n osalta.

LÄHTEET

Abiteboul, S., Hull, R. & Vianu, V. (1995). *Foundations of Databases.* Boston: Addison Wesley.

Ahadi, A., Behbood, V., Vihavainen, A., Prior, J. & Lister, R. (2016). Students' Syntactic Mistakes in Writing Seven Different Types of SQL Queries and its Application to Predicting Students' Success. Teoksessa *SIGCSE '16: Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (401-406). New York, NY, USA.

Ahmed, U., Sindhgatta, R., Srivastava, N. & Karkare. A. (2019). Targeted Example Generation for Compilation Errors. Teoksessa *Proceedings of the 34th ACM/IEEE International Conference on Automated Software Engineering (ASE '19)* (78-87). New York, NY, USA.

Beaulieu, A. (2020). *Learning SQL: Generate Manipulate and Retrieve data* (3. painos) Sebastobol: O'Reilly.

Barik, T., Smith, J., Lubick, K., Holmes, E., Feng, J., Murphy-Hill, E. & Parnin, C. (2017). Do Developers Read Compiler Error Messages? Teoksessa *Proceedings of the IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. Buenos Aires, Argentina.

Becker, A., Denny, P., Pettit, R., Bouchard, D., Bouvier, D., Harrington, B., Kamil, A., Karkare, A., McDonald, C., Osera, P., Pearce, J. & Prather, J. (2019). Compiler Error Messages Considered Unhelpful: The Landscape of Text-Based Programming Error Message Research. *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '19)* (177-210). Aberdeen, Scotland UK.

Becker, A. (2016). An Effective Approach to Enhancing Compiler Error Messages. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)* 126–131. New York, NY, USA.

Brass, S., & Goldberg, C. (2005). Semantic Errors in SQL Queries: A Quite Complete List. *Journal of Systems and Software*, 79(5), 630-644.

Carlebach, N., & Yeung, N. (2020). Subjective confidence acts as an internal cost-benefit factor when choosing between tasks. *Journal of Experimental Psychology: Human Perception and Performance* 46(7), 729–748.

Codd, E. (1970). A Relational model of Data for Large Shared Data Banks. *Communications of the ACM, 13(6),* 377-387.

Darwen, H. (2009). *An introduction to Relational Database Theory*. Lontoo: Bookboon. Haettu osoitteesta https://bookboon.com/en/an-introduction-to-relational-database-theory-ebook

DB-engines (2020a, lokakuu). Method of calculating the scores of the DB-Engines Ranking. Haettu 31.10.2020 osoitteesta https://db-engines.com/en/ranking_definition

DB-engines (2020b). DB-Engines ranking. Haettu 31.10.2020 osoitteesta https://db-engines.com/en/ranking

DB-engines (2020c). DB-Engines Ranking - Trend Popularity. Haettu 31.10.2020 osoitteesta https://db-engines.com/en/ranking_trend

Denny, P., Luxton-Reilly, A. & Carpenter, D. (2014). Enhancing syntax error messages appears ineffectual. Teoksessa *ITiCSE '14: Proceedings of the 2014 conference on Innovation & technology in computer science education* (273-278). Uppsala Sweden, June, 2014

Elmasri, R. & Navathe, S. (2015). *Fundamentals of Database Systems* (7. painos) Pearson.

Horning, J. (1976). *What the compiler should tell the user*. Kirjassa *Compiler construction: an advanced course*. Berlin-Heidelberg: Springer-Verlag

International Organization for Standardization. (2016a). SQL—Part 1: Framework. (ISO/IEC 9075-1:2016). Haettu osoitteesta https://www.iso.org/standard/63555.html

International Organization for Standardization. (2016b). SQL—Part 2: Foundation. (ISO/IEC 9075-2:2016). Haettu osoitteesta https://www.iso.org/standard/63556.html

Kölling, M. (1999). *The Design of an Object-Oriented Environment and Language for Teaching* (Väitöskirja). Basser Department of Computer Science, University of Sydney. Haettu osoitteesta https://kar.kent.ac.uk/21868/

Lee, M. & Ko, A. (2011). Personifying Programming Tool Feedback Improves Novice Programmers' Learning. Teoksessa *Proceedings of the Seventh International Workshop on Computing Education Research (ICER '11)* (109–116). New York,NY, USA.

Marceau, G., Fisler, K. & Krishnamurthi, S. (2011) Measuring the Effectiveness of Error Messages Designed for Novice Programmers. Teoksessa *Proceedings of the 42nd ACM technical symposium on Computer science education (SIGCSE '11)* (499-504). Dallas, TX, USA, March, 2011.

McCall, D. & Kölling, M. (2019). A New Look at Novice Programmer Errors. *ACM Transactions on Computing Education, 19(4)*, Article 38

McCall, D. (2016). *Novice Programmer Errors – Analysis And Diagnostics* (Väitöskirja). University of Kent. Haettu osoitteesta https://kar.kent.ac.uk/61340/

MySQL. (2020). MySQL Differences from Standard SQL. Haettu 22.11.2020 osoitteesta https://dev.mysql.com/doc/refman/8.0/en/differences-from-ansi.html

Oracle. (2020). Oracle Compliance to Core SQL. Haettu 22.11.2020 osoitteesta https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Oracle-Compliance-To-Core-SQL2011.html#GUID-D372D906-805B-49B8-824A-D4697B05B7F8

Oracle. (2017, 24 heinäkuuta). Database SQL Tuning Guide: SQL Processing. Haettu 16.11.2020 osoitteesta https://docs.oracle.com/database/121/TGSQL/tgsql_sqlproc.htm

PostgreSQL. (2020). PostgreSQL vs SQL Standard. Haettu 22.11.2020 osoitteesta https://wiki.postgresql.org/wiki/PostgreSQL_vs_SQL_Standard

Simsion, G. & Witt, G. (2004). *Data Modeling Essentials*. (3. painos). San Francisco: Elsevier.

Smelcer, J. (1995) User errors in database query composition. *International journal of Human-Computer studies, 42(4)*, 353-381.

Taipalus T. & Perälä, P. (2019). What to Expect and What to Focus on in SQL Query Teaching. Teoksessa *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)* (198-203). Minneapolis, MN, USA. February 27th–March 2nd, 2019.

Taipalus, T. (2020). Error messages in relational database management systems. Julkaisematon käsikirjoitus.

Taipalus, T. & Seppänen, V. (2020). SQL Education: A Systematic Mapping Study and Future Research Agenda. *ACM Transactions on Computing Education, 20(3), Article 20.*

Taipalus T., Siponen M. & Vartiainen, T. (2018). Errors and Complications in SQL Query Formulation. *ACM Transactions on Computing Education* 18, 3, Article 15.

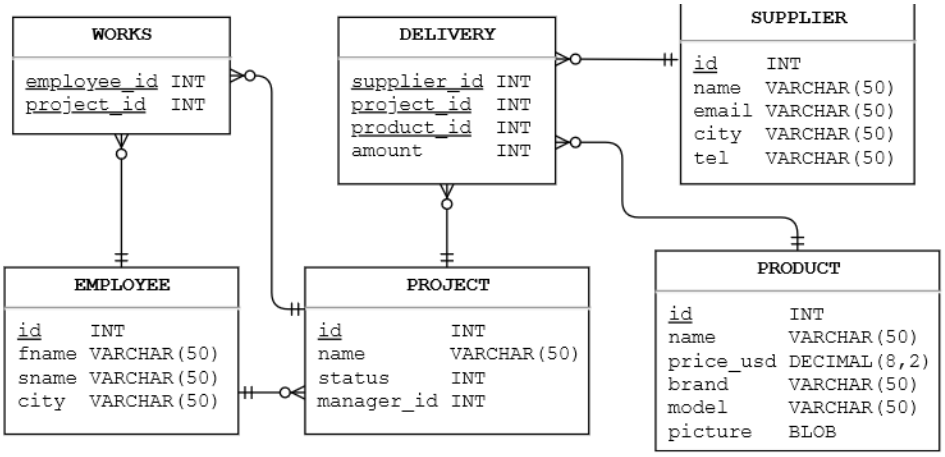
Traver, J. (2010). On Compiler Error Messages: What They Say and What They Mean. *Advances in Human-Computer Interaction, Article 3*. Haettu osoitteesta https://www.hindawi.com/journals/ahci/2010/602570/

Weinberg, P., Groff, J. & Oppen, A. (2009). *SQL The Complete Reference* (3. painos) McGraw-Hill.

LIITE 1 KYSELYLOMAKKEET

Kyselylomake sisälsi yhden sivun kutakin syntaksivirhettä kohden. Tutkittuja syntaksivirheitä oli 16 kappaletta. Jokainen sivu sisälsi:

- tietokannan skeeman (kuvana),
- tietotarpeen
- virheen sisältävän SQL-lauseen,
- tietokannanhallintajärjestelmän (MySQL tai PostgreSQL) kyselystä antaman virheilmoituksen ja
- kolme Likert-asteikolla vastattavaa kysymystä (tutkimuskysymykset).



Find the names and manager ids of projects to which a supplier from Sydney has delivered at least one delivery. Sort the results according to project name in ascending order.

```
SELECT name, manager_id
FROM   project
WHERE  id IN
        (SELECT project_id
        FROM    delivery
        WHERE   supplier_id IN
            SELECT id
            FROM    supplier
            WHERE   city = 'Sydney')
        );
```

```
ERROR:  syntax error at or near "SELECT"
LINE 7:             SELECT id
                    ^
```

	1	2	3	4	5
The error message was useful in finding the error	□	□	□	□	□
The error message was useful in fixing the error	□	□	□	□	□
The error message increased my confidence in error recovery	□	□	□	□	□

Kuvio 13 Kyselylomakkeen sivu yhden testin osalta (Taipalus, 2020).

LIITE 2 DB-ENGINES -VERKKOPALVELUN KÄYTTÄMÄ TKHJ-TUOTTEIDEN LISTAUSMENETELMÄ

db-engines.com on itävaltalaisen IT-konsulttiyritys Solid IT:n ylläpitämä listaus suosituimmista TKHJ-ratkaisuista.

db-engines (2020a) mukaan listausmenetelmä tutkii 6 parametria, jotka ovat:

1) Mainintojen määrä verkkosivuilla
    - Mainintojen määrän mittaamiseen käytetään Google ja Bing hakukoneita.
    - Hakuterminä käytetään <TKHJ-nimi> ja ”database”.
        - Esim. ”Oracle” ja ”database”
2) Yleinen kiinnostus kutankin TKHJ:ta kohtaan Google Trends (https://trends.google.com/) hakujen tiheyden perusteella.
3) Teknisten keskustelujen määrä tunnetuilla IT-kysymys ja vastauspalstoilla StackOverflow (https://stackoverflow.com) ja DB STack Exhange (https://dba.stackexchange.com)
4) Työtarjousten määrä, joissa TKHJ mainitaan.
    a. Käytetyt työnhakupalvelut ovat Indeed (https://www.indeed.com) ja Simpley Hired (https://www.simplyhired.com)
5) Julkisten profiilien määrä, joissa TKHJ on mainittu ammattilaisverkostopalveluissa
    a. Käytetty ammattilaisverkostopalvelu: LinkedIn (https://www.linkedin.com)
6) Relevanssi sosiaalisissa verkostoissa
    a. Maininta Twitter-palvelun (https://twitter.com) viesteissä eli tweeteissä.

Menetelmän kuvaus (db-engines, 2020a) suomeksi käännettynä.

> Laskemme järjestelmän suosioarvon standardoimalla ja keskiarvoistamalla yksittäiset parametrit. Nämä matemaattiset muunnokset tehdään tavalla, jolla yksittäisten järjestelmien etäisyys säilyy. Tämä tarkoittaa, että kun järjestelmällä A on kaksi kertaa suurempi arvo DB-Engines Rankingissa kuin järjestelmällä B, se on kaksi kertaa niin suosittu, kun keskiarvo lasketaan yksittäisten arviointikriteerien perusteella.
>
> Itse tietolähteiden muuttuvien määrien aiheuttamien vaikutusten poistamiseksi suosioarvot ovat aina suhteellinen arvo, jota tulisi tulkita vain muihin järjestelmiin verrattuna.
>
> DB-Engines Ranking ei mittaa järjestelmien asennusten määrää tai niiden käyttöä IT-järjestelmissä. Voidaan odottaa, että järjestelmän

suosion kasvu DB-Engines Rankingin avulla mitattuna (esim. Keskusteluissa tai työtarjouksissa) edeltää vastaavaa laajaa järjestelmän käyttöä tietyllä aikakertoimella. Tämän vuoksi DB-Engines Ranking voi toimia varhaisena indikaattorina.