**Ville-Pekka Vahteala**

# Tietokannat ja versionhallinta

Tietotekniikan kandidaatintutkielma

6. toukokuuta 2015

Jyväskylän yliopisto

Tietotekniikan laitos

**Tekijä:** Ville-Pekka Vahteala

**Yhteystiedot:** vahtis@student.jyu.fi

**Työn nimi:** Tietokannat ja versionhallinta

**Title in English:** Databases and Version controlling systems

**Työ:** Kandidaatintutkielma

**Sivumäärä:** 20+0

**Tiivistelmä:** Tämän tutkielman taustalla on työelämässä havaitut ongelmat tietokantojen rakenteiden ylläpitoon. Tavoitteena on löytää ratkaisumalli, joka mahdollisimman hyvin ratkaisisi havaitut ongelmat. Tietokannan rakennetta muokkaavien SQL–lauseiden järjestelmällinen tallentaminen version hallintaan näyttää ratkaisevan ongelmia, vaikkakin rinnakkaisen kehitystyön osalta kaikkia ongelmia ei saatu ratkaistua.

**Avainsanat:** Tietokanta, versionhallinta

**Abstract:** This study is based on problems acknowledged during development work of databases. Study tries to find solutions to these problems. It seems that storing database structure altering SQL batches into version controlling system will solve some of the problems. However parallel development of database still has some issues.

**Keywords:** Database, Version controlling systems

# Kuviot

# Sisältö

# 1 Johdanto

Tämän tutkielman aiheena on tietokannan rakenteen ylläpito ja kehittäminen versionhallintaa hyödyntäen. Tarkoituksena on löytää malli, jonka avulla tietokannan rakenteeseen tehtävät muutokset ovat toistettavissa ja hallittavissa suurempina kokonaisuuksina.

Nykyisessä käytännössä tietokantamuutokset ovat yksittäisiä verkkolevylle tallennettuja tiedostoja. Tiedostojen hallinta on hoidettu excel–tiedoston avulla, johon jokainen kehittäjä on lisännyt omat tiedostonsa. Tästä on aiheutunut monenlaisia ongelmia.

Tehtäessä versiopäivitystä testi- tai tuotantoympäristöön on havaittu ongelmia tietokantaobjektien toiminnassa tai päivitysrutiini on keskeytynyt virheeseen. Näissä tapauksissa on usein ollut kyse siitä, että kehittäjät ovat unohtaneet lisätä luomansa tiedostot excel–tiedostoon. Ongelma voidaan ratkaista tallentamalla tiedostot versionhallintaan, jolloin excel–tiedostoa ei enää tarvitse käyttää.

Ajoittain on sattunut tilanteita, joissa kahdella kehittäjällä on ollut tarve muuttaa samanaikaisesti tietokannan objektia. Nykyisessä käytännössä tällaista tilannetta ei ole huomattu ja näin ollen excel–tiedostossa jälkimmäisenä oleva muutos on jäänyt voimaan ja toinen on saattanut kadota kokonaan. Ratkaisussa jokaista objektia muokataan vain yhden versionhallinnassa olevan tiedoston avulla. Tällöin ristiriita tulee ilmi tallennettaessa objektia versionhallintaan.

Nykyinen käytäntö mahdollistaa muutostiedoston luomisen objektille, joka on riippuvainen toisesta kehitysympäristössä muokatusta objektista, jolle ei ole vielä tehty muutostiedostoa. Tällöin riippuvaisen objektin muutos epäonnistuu päivitysrutiinissa. Tätä ongelmaa voidaan ratkaista ainakin kahdella tavalla. Ensimmäinen tavassa jokaisella kehittäjällä on oma tietokantansa, jolloin keskeneräisiin objekteihin viittaaminen on mahdotonta. Toisessa tavassa muutokset versioidaan muokkauspäivämäärän perusteella, jolloin objektit luodaan tietokantaan oikeassa järjestyksessä.

Ohjelmiston julkaisussa on ongelmia aiheutunut myös siitä, että tietokantaa ei ole voitu testata osana jatkuvan integraation mallia muiden ohjelmiston osioiden tapaan. Tästä syystä tietokantaobjekteihin liittyvät ongelmat on havaittu vasta julkaisujen yhteydessä ja ongelmien syiden etsiminen on ollut haastavaa sekä aikaa vievää. Pahimmillaan tämä on aiheuttanut jopa julkaisun viivästymisiä. Ratkaisumallissa tulee ottaa huomioon, että tietokanta voidaan luoda ilman manuaalista työtä osana jatkuvan integraation mallia.

Ohjelmistojen elinkaaren aikana on tullut vastaan tilanteita, joissa tuotantoympäristössä ilmenevää ongelmaa ei saada toistettua muissa ympäristöissä. Näissä tilanteissa ainoita vaihtoehtoja ovat olleet ongelman tutkiminen tuotantoympäristössä tai tuotantoympäristön palauttaminen kehitysympäristöön. Ensimmäinen vaihtoehto saattaa haitata ohjelmiston päivittäistä käyttöä. Toinen vaihtoehto taas vie paljon aikaa ja levytilaa. Jotta tämä ongelma voidaan ratkaista, tulee ratkaisussa voida palata mihin tahansa tietokannan versioon.

Tutkielman tulevissa luvuissa käydään ensin läpi tarkemmin läpi tietokantaa ja sen rakennetta sekä kuinka rakennetta ja tietoa muokataan. Versionhallintaa käsittelevässä luvussa käydään läpi tutkielman kannalta keskeisiä versiohallinnan käsitteitä kuten versiot ja haarat. Peruskäsitteiden läpikäynnin jälkeen esitellään ratkaisumalli. Viimeisessä luvussa on tutkielman yhteenveto.

# 2 Tietokanta

Martinin (1977) mukaan tietokanta on kokoelma toisiinsa liittyviä tietoja, jotka ovat tallennettu ohjelmien käytettäviksi. Tallennustapa on riippumaton tietoa käyttävästä ohjelmasta ja tietojen tallennusrakenne mahdollistaa jatkuvan ohjelmistokehityksen (Martin 1977).

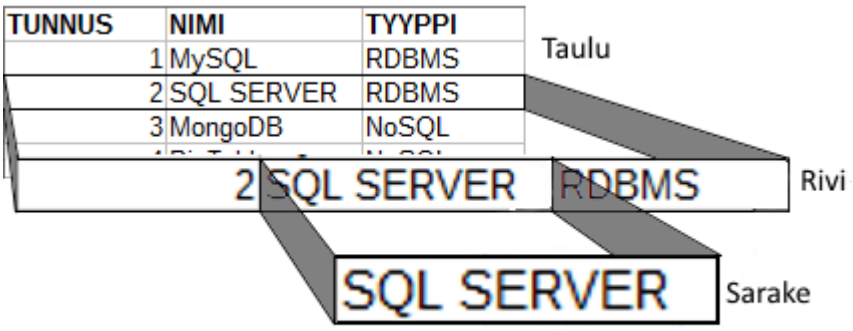
Mannila ja Räihä (1992) toteavat tietokantojen elinikien olevan pitkiä ja tietokannan uudelleen suunnittelemisen olevan hankalaa, ellei jopa mahdotonta, koska alkuperäiset ehdot usein muuttuvat ohjelmiston elinkaaren aikana. Tällaisia ehtoja ovat esimerkiksi käytettävä tietokantajärjestelmä tai tietokantaa hyödyntävien sovellusten vaihtuminen. Martin (1977) kertoo tietokantojen kehittyvän jatkuvasti ja uusien ohjelmien käyttävän niitä. Tietokantoihin lisätään uusia rakenteita ja käyttäjät hakevat niistä tietoja uusilla tavoilla. Tämän vuoksi on tärkeää, että tietokannan rakennetta pystytään muokkaamaan muuttuvien tarpeiden mukaan.

Tässä tutkielmassa keskitytään erityisesti relaatiotietokantoihin, joskin malli on sovellettavissa myös muihin tietokantoihin. Relaatiotietokannat on valittu ominaisuuksiensa takia. Ominaisuuksista kaksi merkittävintä ovat tietojen tallennus tauluihin ja taulujen relaatiot toisiinsa yhteisten kenttien kautta (Wang ja Tan 2005).

## 2.1 Tietokannan rakenne

Tietokannan tiedot tallennetaan tauluihin, jotka ovat kokoelmia riveistä ja sarakkeista (Wang ja Tan 2005). Tietokannan taulut voidaan edelleen järjestellä virtuaalisiksi tauluiksi eli näkymiksi. Näkymät koostuvat todellisten taulujen osajoukosta ja ne voivat sisältää vain osan taulujen sisällöstä, jolloin ne mahdollistavat tietoturvan tavoitteet (Wang ja Tan 2005). Rakennetta on kuvattu kuviossa 1.

Tietokantaan suoritettavia kyselyitä voidaan yhdistää aliohjelmiksi. Eisenberg (1996) määrittelee kolme laajaa mallia aliohjelmien toteuttamiseen: monen kyselyn proseduurit, tallennetut rutiinit ja moduulit sekä ulkoiset rutiinit.




Kuvio 1. Tietokannan rakenne

## 2.2 SQL–kieli

Structured Query Language (SQL) on IBM:n kehittämä kyselykieli, joka perustuu Edgar F. Coddin vuonna 1970 julkaistuun artikkeliin *A Relational Model of Data for Large Shared Data Banks* (Chamberlin ja Boyce 1974; Codd 1970). Kielen alkuperäinen tavoite on ollut tarjota yksinkertainen kieli tehtävien tekemiselle. Vuosien saatossa ja tietojärjestelmien erilaisten tarpeiden vaatiessa kielestä on tullut hyvin laaja ja monimutkainen.

SQL–kieli voidaan jakaa useisiin alakieliin, kuten Data Query Language (DQL), Data Control Language (DCL), Data Manipulation Language (DML) ja Data Definition Language (DDL) (Chamberlin ja Boyce 1974). Tämän tutkielman osalta keskitytään kahteen jälkimmäisenä mainittuun.

Tiedon hakuun ja muokkaukseen tietokannassa käytetään DQL– ja DML–kieliä. Yleisesti DQL–kieltä ei eritellä omakseen vaan se sisällytetään DML–kieleen. Tärkeimmät komennot edellämainittuihin tarkoituksiin ovat: SELECT, INSERT, UPDATE ja DELETE (Microsoft 2015). Komentoja yhdistelemällä voidaan muodostaa luvussa 2.1 mainittuja aliohjelmia ja näkymiä.

Data Definition Language (DDL)–lausekkeilla muokataan tietokannan tietorakenteita, näkymiä ja aliohjelmia (Microsoft 2015). Tutkielman kannalta merkittävimpiä komentoja ovat CREATE, DROP ja ALTER, joiden avulla voidaan luoda, poistaa ja

muuttaa tietokannan rakenteita ja aliohjelmia. Listauksessa 2.1 on yksinkertainen tietokantataulun luontilause.

```
CREATE TABLE Asiakas (
  tunnus INT
  , nimi VARCHAR(50)
  , osoite VARCHAR(100)
  );
GO
```

Listaus 2.1. Tietokantataulun luonti

# 3 Versionhallinta

Versionhallinnan avulla voidaan seurata ohjelmiston osien muutoshistoriaa sovelluksen kehityksen aikana. Fonseca (1996) kertoo versionhallinnasta olevan hyötyä erityisesti, kun halutaan palata aikaisempaan tilaan tai selvittää, kuinka kehitys on johtanut nykyiseen tilaan. Versionhallinta mahdollistaa myös kehityshaarojen luomisen sovelluksen eri versioiden ylläpidon vuoksi (Midha 1997).

## 3.1 Versiot

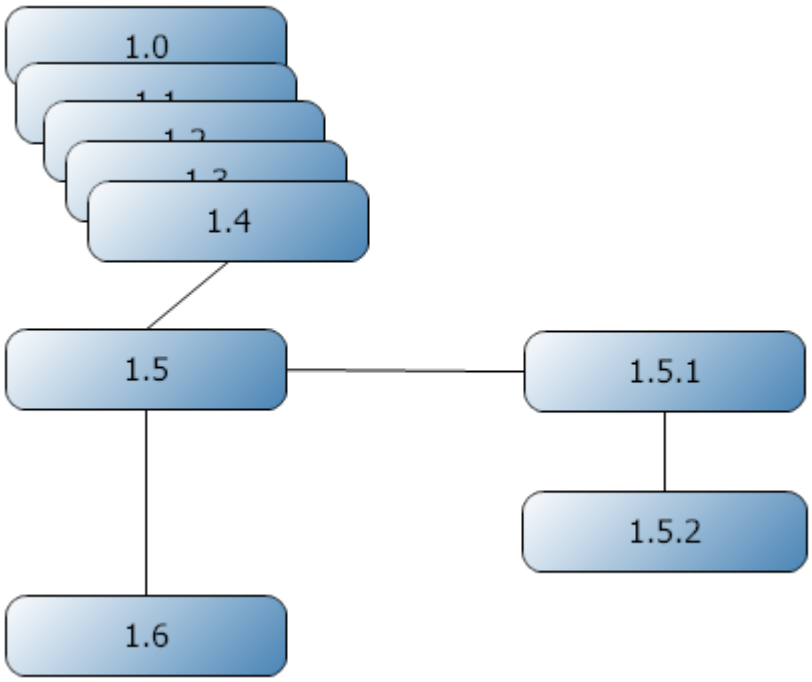
Alkeellisimmallaan versionhallinnassa tiedostojen versiot erotetaan toisistaan lisäämällä tiedostoon nimeen versionumero. Esimerkiksi create_table.sql–tiedoston ensimmäinen versio voitaisiin nimetä create_table.sql.1 ja sitä seuraava versio create_table.sql.2.

Edellisen kappaleen tallennusmuoto vaatii kuitenkin suhteellisen paljon tallennustilaa. Kehittyneemmissä järjestelmissä tiedostoista tallennetaan ensimmäinen versio ja erot siihen nähden. Kahden version välistä eroa kutsutaan *deltaksi*. Tämä matematiikasta lainattu termi viittaa siihen, että versioiden tulisi erota vain vähän toisistaan, käytännössä näin ei kuitenkaan aina ole (Conradi ja Westfechtel 1998).

Pelkkä versio ei riitä kuvaamaan kaikkia kehityksessä olevia tiedostoja. Tiedoston versiota, jonka tarkoitus on korvata aiempi versio, kutsutaan revisioksi (Fogel 1999). Kuviossa 2 tiedoston versio 1.6 on version 1.5 revisio. Tiedostojen samanaikaisesti kehitettäviä versioita kutsutaan varianteiksi (Conradi ja Westfechtel 1998). Kuvion 2 tiedoston versio 1.5.1 ja 1.5.2 ovat versioiden 1.5 ja 1.6 variantteja sekä toistensa revisioita.

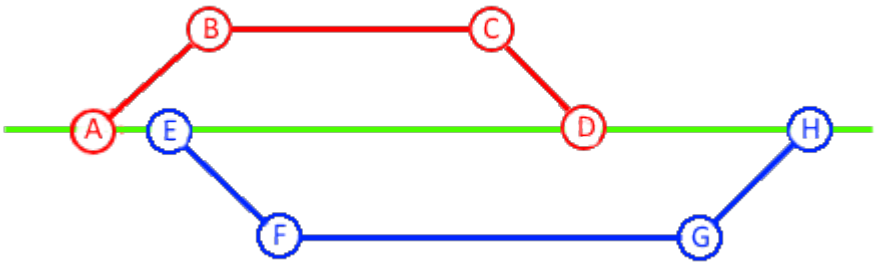
## 3.2 Haarat

Ohjelmistosta voi olla samaan aikaan kehityksessä useampi ominaisuus ja näiden osalta yleinen käytäntö on muodostaa ominaisuutta vastaava haara. Näin kehittä-




Kuvio 2. Tiedoston versiointi graafisesti esitettynä

jät pystyvät keskittymään yhden ominaisuuden tekemiseen kerrallaan muiden kehittäjien keskittyessä muihin ominaisuuksiin. Collins-Sussman, Fitzpatrick ja Pilato (2011) määrittelevät haaran olevan muista kehityslinjoista riippumattoman kehityslinjan. Kuviossa 3 on kuvattu vihreällä värillä alkuperäistä kehityshaaraa, josta on erkaantunut sinisellä ja punaisella kuvatut kehityshaarat.



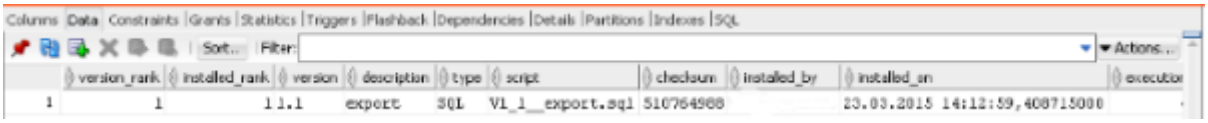

Kuvio 3. Rinnakkaiset kehityshaarat versionhallinnassa

# 4 Eräs ratkaisumalli

Tässä esitellään yksi ratkaisumalli, jolla tietokannan rakenne voidaan tallentaa versionhallintaan ja kuinka tietokannasta voidaan muodostaa versioita. Mallin pohjana ovat K. Scott Allenin työssään havaitsemat kolme sääntöä. Ensimmäinen sääntö kieltää keskitetyn tietokannan käytön kehitystyössä, koska käytäntö johtaa virheisiin ja on aikaa vievää. Toinen sääntö kehottaa ylläpitämään yhtä lähdettä tietokannalla, jotta tietokanta voidaan helposti asentaa mihin tahansa ympäristöön. Kolmannen säännön mukaan tietokanta tulee versioida, jotta muutokset voidaan hallitusti viedä kehitysympäristöstä testausympäristön kautta tuotantoympäristöön (Allen 2008a).

## 4.1 Tietokannan versiointi

Tietokannan versioinnin avulla pystytään saattamaan tietokanta haluttuun versioon. Tämä versio voi olla esimerkiksi tuotantoa vastaava versio, jolloin johdannossa 1 esitetty ongelma saatetaan pystyä toistamaan kehitysympäristössä. Allen (2008a) kertoo lisäksi versioinnin mahdollistavan hallitun muutosten tekemisen kehityksestä tuotantoon asti.




Kuvio 4. Flyway–järjestelmän perusversio tietokannassa

## 4.2 Perusversio

Allen (2008c) määrittelee perusversion sisältävän kaikki tietokannan luomiseksi tarvittavat komennot mukaan lukien koodistotaulujen sisällön tuottavat lisäyslauseet. Flyway–järjestelmässä perusversio voidaan muodostaa myös jo olemassa olevan tietokannan perusteella. Perusversioksi suositellaan käytettäväksi tuotannon tietokantaa (Fontaine 2015).

Perusversio voisi sisältää listauksen 4.1 kaltaisen tietokannan määrityksen. Esimerkkinä käytetään kuvitteellisen verkkokaupan tilaustenhallintaa ja asiakasrekisteriä.

```
CREATE TABLE Asiakas (
  tunnus INT
  , nimi VARCHAR(50)
  , osoite VARCHAR(100)
  );
GO
CREATE VIEW AsiakkaanOsoite AS SELECT DISTINCT osoite FROM Asiakas;
GO
CREATE TABLE TilauksenTilanne(tunnus INT, tilanne VARCHAR(50));
GO
```

Listaus 4.1. Perusversion tietokannan taulujen ja näkymien luonti

Flyway–järjestelmä tarjoaa mahdollisuuden luoda perusversio jo olemassa olevaan tietokantaan seuraavalla komennolle.

```
> flyway baseline -Dflyway.baselineVersion=1 -Dflyway.baselineDescription="Base version"
```

Kuvio 5. Perusversion luonti olemassa olevaan tietokantaan Flyway–järjestelmässä

## 4.3 Muutokset

Tietokannan elinkaaren aikana tulee vastaan tilanteita, joissa tietokannan rakennetta täytyy muokata. Tällaisia tilanteita ovat esimerkiksi muuttuneet vaatimukset sovelluksessa tai suorituskyvyn kannalta tärkeän indeksin lisääminen tauluun. Allen (2008b) luettelee muutoksien kohteiksi taulut, indeksit, avaimet ja rajoitteet. Erikseen käsiteltäviksi kohteiksi hän luettelee näkymät ja aliohjelmat (proseduurit ja funktiot).

Listauksessa 4.2 on tyypillinen esimerkki muutoksesta, jossa tietokantatauluun lisätään sarake ja koodistotauluun lisätään rivi.

```
ALTER TABLE Asiakas
  ADD KenganNumero int NOT NULL DEFAULT 0
GO
INSERT INTO TilauksenTilanne
   (Tunnus, Tilanne)
   VALUES(3, 'Peruttu')
GO
```

Listaus 4.2. Tietokannan tyypillinen muutos

Tärkeänä osana mallissa on virheiden esiintulo mahdollisimman aikaisessa vaiheessa. Tietokannassa on kuitenkin mahdollista muokata olemassa olevia rakenteita siten, että olemassa olevat kohteet eivät enää toimi. Tällaisesta muokkauksesta ei seuraa minkäänlaista virhettä tai varoitusta. Tämän vuoksi Allen (2008d) haluaa erottaa näkymien, proseduurien ja niiden kaltaisten kohteiden luonnin erillisiksi tiedostoiksi. Hän jatkaa vielä, että kyseisten kohteiden osalta tulee kirjoittaa aina kohteen poisto ja uudelleen luonti.

Esimerkiksi listauksen 4.3 kaltainen muutos Asiakas–tauluun aiheuttaisi AsikkaanOsoite–näkymän hajoamisen ilman virhettä. Tällainen virhe ei pääsisi tapahtumaan, jos tietokannan rakenteellisten muutosten jälkeen näkymät, proseduurit ja muut toisista kohteista riippuvaiset kohteet poistettaisiin ja luotaisiin uudelleen.

```
-- Normalisoidaan tauluja
CREATE TABLE Osoite (tunnus INT, osoite VARCHAR(100));
GO
-- tuodaan osoitteet asiakas-taulusta
INSERT INTO Osoite SELECT ROW_NUMBER() OVER (ORDER BY osoite),
   Osoite FROM Asiakas GROUP BY Osoite;
GO
ALTER TABLE Asiakas ADD osoiteTunnus INT;
GO
-- asiakkaan osoitteet kuntoon
```

```
UPDATE Asiakas SET osoiteTunnus = osoite.tunnus FROM Osoite WHERE
   osoite.osoite = Asiakas.osoite;
GO
-- poistetaan osoite - sarake
ALTER TABLE Asiakas DROP COLUMN osoite
GO
```

Listaus 4.3. Virheen aiheuttava rakenteellinen muutos tietokantaan

Näiden muutosten jälkeen listauksen 4.4 ajaminen epäonnistuisi, koska Asiakas–taulussa ei ole enää osoite–kenttää.

```
SELECT * FROM AsiakkaanOsoite;
Msg 207, Level 16, State 1, Procedure AsiakkaanOsoite, Line 28
Invalid column name 'osoite'.
Msg 4413, Level 16, State 1, Line 28
Could not use view or function 'AsiakkaanOsoite' because of
   binding errors.
```

Listaus 4.4. Epäonnistunut kysely

## 4.4 Jatkuva integraatio

Jatkuvan integraation puuttuminen tietokannan osalta on nykyisen käytännön haastavampia ongelmia. Tietokannan muutostiedostoissa olevat virheet tulevat esiin vasta versiopäivityksen yhteydessä, jolloin korjauksen löytäminen niihin on hankalaa. Järjestelmien jatkuva integraatio parantaa ohjelmiston laatua ja kehittäjien tuottavuutta kääntämällä ja testaamalla lähdekoodia sekä löytämällä virheet ennakolta (Kim et al. 2008).

Ratkaisumallissa on päädytty käyttämään Flyway–järjestelmää, jotta mallissa voidaan käyttää jatkuvaa integraatioa . Jenkins–alustalle flyway–järjestelmän liittäminen vaatii vain muutaman tunnin työn (Anderson 2012).

Luvussa 4.3 mainittu virhe paljastuisi jatkuvan integraation mallissa, koska näkymän luonti olisi vasta taulujen luonnin jälkeen ja sarakkeen puuttuminen johtaisi listauksen 4.5 mukaiseen virheeseen.

```
DROP VIEW AsiakkaanOsoite;
GO

CREATE VIEW AsiakkaanOsoite AS SELECT DISTINCT osoite FROM Asiakas;
GO

Msg 207, Level 16, State 1, Procedure AsiakkaanOsoite, Line 33
Invalid column name 'osoite'.
```

Listaus 4.5. Näkymän luonti lauseen virhe jatkuvassa integraatiossa

# 5 Yhteenveto

Tietokannoilla on jo pitkä historia osana ohjelmistoja. Perinteisesti ne ovat olleet kokonaan erillisiä ohjelmistokehityksestä, mutta niiden suhteen petää samat lainalaisuudet kuin muuhinkin ohjelmistokehitykseen laadun ja tuottavuuden osalta. Tietokantoja muokataan muuttuvien tarpeiden mukaan nyt ja tulevaisuudessa.

Tämä tutkielma lähti liikkeelle käytännön ongelmista työelämässä ja lopputuloksena tutkielmassa esitellään malli, jolla suurin osa esitetyistä ongelmista saadaan ratkaistua. Malli vaatii kehittäjiltä jonkin verran kurinalaisuutta ja tahtoa tehdä asiat sovitulla tavalla, mutta nämä vaatimukset näkyvät ohjelmiston parempana laatuna.

Tämä tutkielma tuntuu vain raapaisevan jatkuvan integraation osa-aluetta. Jatkossa kiinnostavia aiheita voisivat olla jatkuvan integraation ja käyttöönoton työkalujen vertailu tietokantojen näkökulmasta.

# Kirjallisuusluettelo


Allen, Scott K. 2008a. *Three Rules for Database Work.* Saatavilla WWW-muodossa, http://bit.ly/1D9KZ4O, viitattu 1.4.2015.

———. 2008b. *Versioning Databases – Change Scripts.* Saatavilla WWW-muodossa, bit.ly/1378LhO, viitattu 1.4.2015.

———. 2008c. *Versioning Databases – The Baseline.* Saatavilla WWW-muodossa, http://bit.ly/1B7VSPM, viitattu 1.4.2015.

———. 2008d. *Versioning Databases – Views, Stored Procedures, and the Like.* Saatavilla WWW-muodossa, http://bit.ly/1yrAXdZ, viitattu 14.4.2015.

Anderson, Lyle. 2012. *Using Maven to Integrate Flyway in an Existing Database – Part 1.* Saatavilla WWW-muodossa, http://bit.ly/1E7wkCE, viitattu 6.5.2015.

Chamberlin, Donald D., ja Raymond F. Boyce. 1974. "SEQUEL: A Structured English Query Language". Teoksessa *Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control,* 249–264. SIGFIDET '74. Ann Arbor, Michigan: ACM. doi:10.1145/800296.811515. http://doi.acm.org/10.1145/800296.811515.

Codd, E. F. 1970. "A Relational Model of Data for Large Shared Data Banks". *Commun. ACM* (New York, NY, USA) 13, numero 6 (kesäkuu): 377–387. ISSN: 0001-0782. doi:10.1145/362384.362685. http://doi.acm.org/10.1145/362384.362685.

Collins-Sussman, Ben, Brian W. Fitzpatrick ja C. Michael Pilato. 2011. *Version Control with Subversion.* Saatavilla PDF-muodossa, http://svnbook.red-bean.com/en/1.7/svn-book.pdf, haettu 1.4.2015.

Conradi, Reidar, ja Bernhard Westfechtel. 1998. "Version Models for Software Configuration Management". *ACM Comput. Surv.* (New York, NY, USA) 30, numero 2 (kesäkuu): 232–282. ISSN: 0360-0300. doi:10.1145/280277.280280. http://doi.acm.org/10.1145/280277.280280.

Eisenberg, Andrew. 1996. "New Standard for Stored Procedures in SQL". *SIGMOD Rec.* (New York, NY, USA) 25, numero 4 (joulukuu): 81–88. ISSN: 0163-5808. doi:10.1145/245882.245907. http://doi.acm.org/10.1145/245882.245907.

Fogel, Karl Franz. 1999. *Open Source Development with CVS.* Scottsdale, AZ, USA: Coriolis Group Books. ISBN: 1576104907.

Fonseca, Jonas. 1996. "GitTorrent: a P2P-based Storage Bakend for git". *SIGMOD Rec.* (Marraskuu). ISSN: 0163-5808.

Fontaine, Axel. 2015. *Documentation Flyway • Database Migrations Made Easy.* Saatavilla WWW-muodossa, http://flywaydb.org/documentation/, viitattu 1.4.2015.

Kim, Seojin, Sungjin Park, Jeonghyun Yun ja Younghoo Lee. 2008. "Automated Continuous Integration of Component-Based Software: An Industrial Experience". Teoksessa *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, 423–426. ASE '08. Washington, DC, USA: IEEE Computer Society. ISBN: 978-1-4244-2187-9. doi:10.1109/ASE.2008.64. http://dx.doi.org/10.1109/ASE.2008.64.

Mannila, Heikki, ja Kari-Jouko Räihä. 1992. *The Design of Relational Databases.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0-201-56523-4.

Martin, James. 1977. *Computer Database Organization, 2Nd Ed.* 2nd. Upper Saddle River, NJ, USA: Prentice Hall PTR. ISBN: 0131654233.

Microsoft. 2015. *Transact-SQL Reference (Database Engine).* Saatavilla WWW-muodossa, https://msdn.microsoft.com/en-us/library/bb510741.aspx, viitattu 14.3.2015.

Midha, Anil K. 1997. "Software configuration management for the 21st century". *Bell Labs Technical Journal* 2 (1): 154–165. ISSN: 1089-7089. doi:10.1002/bltj.2039.

Wang, Lingfeng, ja Kay CHen Tan. 2005. *Modern Industrial Automation Software Design.* Wiley-IEEE Press. ISBN: 0471683736.