Jere Moilanen

Entity Framework 6:n käyttäminen eri tietokantojen päällä

Tietotekniikan pro gradu -tutkielma

17. marraskuuta 2020

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Jere Moilanen

Yhteystiedot: moilasenjere@gmail.com

Ohjaaja: Ari Viinikainen

Työn nimi: Entity Framework 6:n käyttäminen eri tietokantojen päällä

Title in English: Using Entity Framework 6 on top of different databases

Työ: Pro gradu -tutkielma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Sivumäärä: 76+0

Tiivistelmä: ORM (object-relational mapping) toimii kehitettävän ohjelmiston ja tietokannan välissä, yksinkertaistaen kehittäjän työtä. Ohjelmistokehittäjä voi muokata ohjelmakoodissaan olioita ja ORM muuntaa olioiden muutokset tietokannan ymmärtämiksi käskyiksi. Entity Framework on Microsoftin kehittämä ORM, jota voi käyttää useiden eri tietokantojen päällä.

Tässä tutkielmassa tutkitaan Entity Frameworkille tehtyjä tietokantakohtaisia tuottajia ja niiden tukea Entity Frameworkin ominaisuuksille. Tuen laajuus testataan kirjoittamalla ohjelmakoodia, joka hyödyntää mahdollisimman laaja-alaisesti Entity Frameworkin ominaisuuksia ja tätä koodia suoritetaan erilaisia tietokantakohtaisia tuottajia vasten.

Avainsanat: Entity Framework, EF, tietokanta, ORM, ADO.NET, tuottaja, tuki, MS SQL, MS SQL CE, MySQL, MongoDB, Oracle, PostgreSQL

Abstract: ORM (object-relational mapping) works between a program and a database, simplifying the work of a developer. The developer can work with objects in the code and ORM translates these changes to the commands, that can be interpreted by the database. Entity Framework is an ORM that is developed by Microsoft and that can be used with many different databases.

With this thesis it will be studied what different database-specific providers have been deve-

loped for Entity Framework and what kind of support do they have for the features of Entity Framework. This support will be tested by writing test code that uses Entity Framework functionality and this code will be executed against different database-specific providers.

Keywords: Entity Framework, EF, databases, ORM, ADO.NET, provider, support, MS SQL, MS SQL CE, MySQL, MongoDB, Oracle, PostgreSQL

Termiluettelo

ADO.NET	.NET frameworkin osana toimiva kokoelma luokkia ja kirjastoja, jotka mahdollistavat tietolähteiden käsittelyn ylemmän tason koodista.
CRUD	Lyhenne sanoista Create, Read, Update ja Delete ja termillä kuvataan entiteettien hallinnan perustoiminnallisuuksia niiden lisäämiseen, lukemiseen, päivittämiseen ja poistamiseen liittyen.
DBMS	Lyhenne sanoista database management system ja se kuvaa tietokantaohjelmistoa, joka hallitsee tiedon määrittelyn, manipuloinnin ja noutamisen tietokannassa.
Entity Framework	Microsoftin kehittämä ORM.
Entity Framework 6	.NET frameworkin päällä toimivan Entity Frameworkin viimeisin versio (kirjoitushetkellä). Entity Framework 6 perustuu avoimeen lähdekoodiin.
Entity Frameworkin tuottaja	Entity Frameworkin ja tietokannan välissä toimiva kirjasto, joka muuttaa Entity Frameworkin sisäiset komennot tietylle tietokannalle sopiviksi komennoiksi.
MongoDB	MongoDB:n kehittämä dokumenttitietokanta.
MS SQL CE	Microsoftin kehittämä kevyt relaatiotietokanta.
MS SQL Server	Microsoftin kehittämä relaatiotietokanta.
MySQL	Avoimeen lähdekoodiin perustuva relaatiotietokanta, jonka nykyisin omistaa Oracle.
Oracle DB	Oraclen kehittämä relaatiotietokanta.
ORM	Lyhenne sanoista Object-Relational Mapping ja tämä tekniikka sallii sovelluskoodin manipuloivan dataa olioiden muodossa. Nämä olioiden käsittelyt ORM:n toteuttava kirjasto muuttaa tietokantakutsuiksi.
PostgreSQL	Kehittäjäyhteisön ylläpitämä avoimeen lähdekoodiin perustuva relaatiotietokanta.

UTC	Lyhenne sanoista Coordinated Universal Time ja sillä kuvataan referenssiaikaa, johon suhteutettuna aikaa esitetään globaalisti. Ei siirry kesä- tai talviaikaan.

Kuviot

Taulukot

Sisältö

1 Johdanto

Käsiteltävän tiedon tallentaminen ja lukeminen on tärkeä osa nykyisten ohjelmistojen toimintaa. Usein tiedon tallennuksen apuna käytetään ORM:ia (Object-Relational Mapper), joka hallitsee sekä ohjelmakoodissa käsiteltävien olioiden tallennuksen tietokantaan että niiden lukemisen tietokannasta. ORM:ja on olemassa lukuisia ja yksi näistä vaihtoehdoista on Microsoftin Entity Framework.

Entity Framework toimii tietokantariippumattomana ratkaisuna (Singh 2015, s. 120) sovellusten tietojen tallentamisessa ja se tarvitsee tietokantayhteyttä varten Entity Frameworkia tukevan tietokantakohtaisen ADO.NET tuottajan (engl. provider). Tuottaja huolehtii komentojen ja kyselyiden muutoksesta tietokantakohtaisiksi komennoiksi ja kyselyiksi. Entity Frameworkista on julkaistu useampia versioita, mutta tässä tutkielmassa keskitytään tuoreimpaan .NET frameworkille julkaistuun versioon 6 (“Entity Framework 6” 2016).

Oletuksena Entity Framework käyttää tiedon tallentamiseen Microsoftin omia tallennusratkaisuja, eli MS SQLtai MS SQL CE -tietokantaa. Näitä tietokantoja tukevat tuottajat tulevat Entity Frameworkin asennuspaketin mukana (“Git repository for Entity Framework 6” 2016). Alla olevaa tietokantaa kuitenkin pystytään vaihtamaan käyttämällä tietokannalle suunniteltua ja Entity Frameworkia tukevaa tuottajaa. Tutkimuksessani perehdyn eri tietokantojen käyttöön Entity Frameworkin tallennusratkaisuna ja pyrin vastaamaan seuraaviin kysymyksiin:

- Onnistuuko Entity Frameworkin alla toimivan tietokannan vaihto helposti tietokantakohtaista tuottajaa vaihtamalla?
- Miten laajasti eri tietokantakohtaiset tuottajat tukevat entiteettimallin mukaisia perustietotyyppejä?
- Miten laajasti eri tietokantakohtaiset tuottajat toteuttavat tuen Entity Frameworkin tukemille kanonisille funktioille?

Ajatus tutkielmasta syntyi työskennellessäni projektissa, jossa pohdittiin mahdollisuutta käyttää Entity Frameworkin päälle toteutettua ohjelmistoa MS SQL serverin sijaan Oraclen tietokannan päällä. Oraclen käyttöön ei koskaan päädytty, koska emme olleet varmoja kuinka

hyvä tuki Oraclen tietokannalle oli olemassa ja tulisiko vastaan kenties yllättäviä ongelmia. Kirjoittajalle jäi tästä tapauksesta kuitenkin mielenkiintoa selvittää asiaa tarkemmin myöhemmin.

Aiemmassa kirjallisuuskartoituksessa (Moilanen 2016) kartoitettiin Entity Frameworkille tehtyjä tuottajia ja selvitettiin millaisia rajoitteita niiden käyttöön on mainittu. Lopuksi todettiin todellisen tuen selvittämisen vaativan empiiristä tutkimusta. Tämän tutkielman tarkoituksena on testata käytännössä näiden aiemmin tutkittujen tuottajien tukemia toiminnallisuuksia käytännön testeillä. Tutkielma toteutetaan vertailevana tutkimuksena.

2 ORM (Object-Relational Mapper)

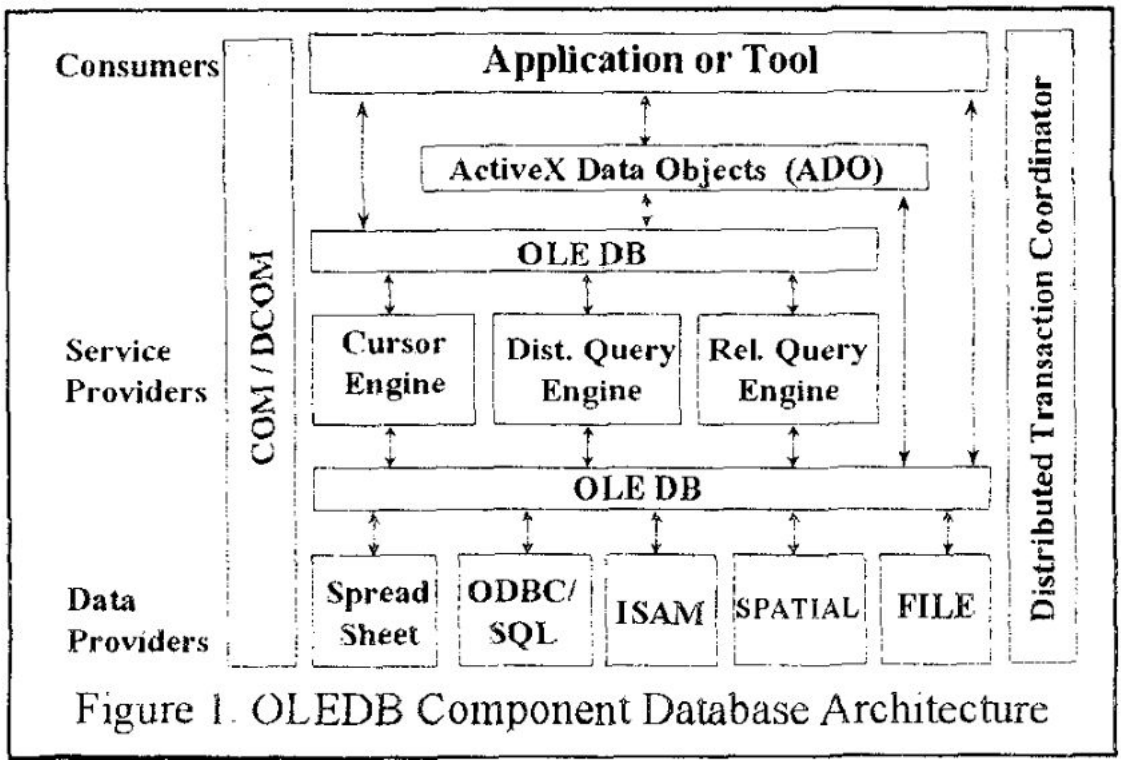
Olio-ohjelmoinnissa tietojen tallentaminen ei ole aivan yksinkertaista, johtuen olioiden viittauksista toisiinsa ja mahdollisen tietorakenteen kompleksisuudesta. Tähän ongelmaan avuksi on kehitetty erilaisia ORM:ja, jotka hallitsevat olioiden tallentamisen pysyväismuistiin ja lukemisen pysyväismusitista. ORM yhteensovittaa olioiden ja relaatiotietokannan väliset rakenteet ja osaa käskyttää tietokantaohjelmistoa (DBMS) sen omalla kielellä.

2.1 Mitä tehtiin ennen ORM:eja

Ennen ORM:ien olemassa oloa kehittäjät usein kirjoittivat itse tiedon tallentamisen ja lukemisen suorittavat tietokantakohtaiset komennot itse. Olemassa olevien tietokantojen paljous oli kuitenkin ongelma kehittäjille ja yhtenäisempi rajapinta tietokantoihin saatiinkin 1992, kun julkaistiin ODBC (Abdalhakim 2009). ODBC (Open DataBase Connectivity) salli kehittäjien muodostaa yhteyden tietokantaan entistä helpommin ja muutti käyttäjän syöttämät komennot tietolähteen ymmärtämään muotoon (tyypillisesti SQL) (Abdalhakim 2009). ODBC:n komennot tietokannan ymmärtämään muotoon muuttaa tietokantaspesifi ajuri (driver), joka huolehtii myös tietokantayhteydet avaamisesta ja sulkemisesta (Abdalhakim 2009).

Myöhemmin osittain ODBC:n päälle julkaistiin OLE DB (Object Linking and Embedding Database), joka salli pääsyn monipuolisempien tietolähteiden äärelle saman rajapinnan läpi, tietolähdekohtaisten tuottajien kautta. ODBC oli edelleen olemassa vaihtoehtoisena väylänä tietokantaan. OLE DB:n avulla tietolähteeksi voitiin valita relaatiotietokannan lisäksi vaikkapa tiedosto, OLAP, sähköpostipalvelin (MS Exchange) tai hakemistopalvelu (Active Directory) (Blakeley 1997). OLE DB oli askel eteenpäin ohjelmistokehittäjän arjen helpottamiseen, luomalla yhtenäisen rajapinnan erilaisten tallennusrakenteiden 1 äärelle (Blakeley 1997).

ADO (ActiveX Data Objects) on korkeamman tason rajapinta, joka toimii OLE DB:n päällä, tarjoten pääsyn lisäämään, poistamaan, muokkaamaan ja lukemaan tietoa OLE DB:n läpi (“ADO Overview and Benefits” 2017).



Kuvio 1. OLE DB -komponentin arkkitehtuurikuvaus (Blakeley 1997).

Tähän asti toiminnallisuudet rakentuivat aina osittain vanhemman teknologian päälle. Osittain ODBC:n aloittaman kehityksen päälle rakentui OLE DB, joka käytti osaan toiminnoista ODBC:n tietokantayhteyksiä. OLE DB:n jälkeen rakentunut ADO käytti OLE DB:n tarjoamia yhteyksiä tiedon käsittelyyn. ADO.NET sen sijaan on uudelleen kirjoitettu kokoelma työkaluja (luokkia ja kirjastoja), joka syntyi .NET Frameworkin osaksi. ADO.NET tarjosi uuden tavan linkittyä tietolähteisiin, eikä sortunut putkittamaan kutsuja aiempien teknologioiden lävitse. Eri valmistajien relaatiokannoille tuli olla omat tietolähdespesifit tuottajat, yleistettyjä tuottajia ei enää tuettu (Agarwal 2012, s. 173).

2.2 ORM:n hyvät puolet

ORM:ien esiinmarssi salli ohjelmistokehittäjien käsitellä olioita välittämättä juuri lainkaan siitä, minne olioita tallennetaan. Tämä säästää kehittäjältä aikaa ja vähentää ohjelman kompleksisuutta, koska ohjelmarivejä tarvitsee koodata vähemmän (Armas ym. 2017). Entiteettien käsittely CRUD-operaatioilla (Create, Read, Update, Delete) ORM:ia käyttäen usealla kut-

sulla peräkkäin on myös nopeampaa verrattuna SQL-kyselyjen ja -komentojen käyttämiseen (Armas ym. 2017). Tämä johtunee siitä, että ORM:t osaavat usein optimoida tietokantakutsuja sekä tallentavat tietoa välimuistiin. Esimerkiksi Entity Framework osaa tallentaa välimuistiin ("Performance considerations for EF 4, 5, and 6" 2014):

- olioita (Object caching),
- käännettyjä kyselyjä (Query plan caching) ja
- tietomallin metadataa (Metadata caching)

ORM:n käyttö saattaa tuoda tullessaan myös tietoturvaparannuksia. Esimerkiksi Entity Frameworkin kanssa SQL injektio -hyökkäys on mahdoton, kunhan käyttäjän syötteitä ei laita sellaisenaan Entity SQL:n sekaan. LINQ to Entities -kyselyt suoritetaan oliomallin lävitse ja sen kanssa SQL injektio -hyökkäys ei ole lainkaan mahdollista ("Security Considerations (Entity Framework)" 2017).

2.3 ORM:n huonot puolet

ORM:n käyttö tuo usein mukanaan kuitenkin joitain rajoitteita. Saattaa olla, että nopeampia kyselyjä / komentoja saavutettaisiin muilla keinoilla ja tällöin halutaan kyselyjä tehdä joko ilman ORM:ia, tai sitten käytössä olevan ORM:n ohi. Esimerkkejä tällaisista tapauksista voisivat olla esimerkiksi tietokannan kyselyjen optimointi. Mitä kaikkea ORM osaa? Osaako ORM:

- luoda oikeat indeksit, jotta kyselyt ovat tarpeeksi nopeita?
- luoda riittävän vähän indeksejä, jotta tietojen päivitysnopeus ja poistonopeus eivät kärsi?
- käyttää hyväkseen tietokannan omia optimoituja tapoja toimia, esimerkiksi MS SQL:n stored procedureja?

ORM piilottaa toiminnallisuutta ja vähänkin isommissa sovelluksissa käytännössä aina joudutaan kuitenkin miettimään ORM:n alla toimivan tietokannan käyttöä kyselyjen ja komentojen nopeutta ajatellen. ORM:n tuottamat SQL-kyselyt saattavat myös olla huonosti muodostettuja, ja harkitsematon ORM:n käyttäjä saattaakin tehdä tarkoitettua raskaampia kyse-

lyitä oliomallin tiedustelun piilottaessa todellisen luodun kyselyn.

ORM:eja on myös erilaisia ja mukana on varmaankin sekä hyvin että huonosti toteutettuja ratkaisuja. Tämä ORM:n hyvyys tai huonous harvoin näkyy päälle päin, vaan käytännössä ORM:n toiminnallisuuteen tutustutaan parhaiten vasta kokeilemalla sen käyttöä.

2.4 Erilaisia ORM-toteutuksia

ORM-toteutuksia löytyy nykyisin lukuisia, eri kielille ja alustoille suunnattuina. Wikipedia listaa gradun kirjoitushetkellä kaikkiaan 48 vaihtoehtoa ja .NET Frameworkille 8 vaihtoehtoa (“List of object-relational mapping software” 2020):

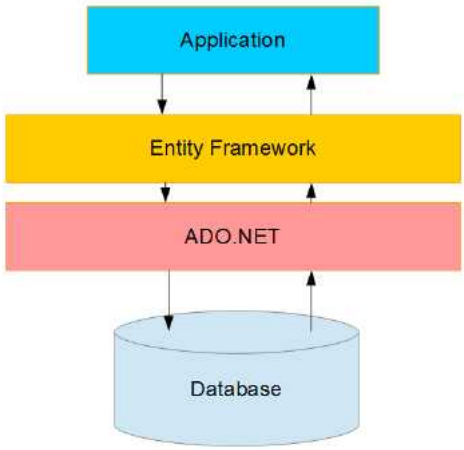
- Base One Foundation Component Library
- Dapper
- Entity Framework
- iBATIS
- LINQ to SQL
- NHibernate
- nHydrate
- Quick Objects

3 ADO.NET Entity Framework

Tässä luvussa käydään läpi mikä on ADO.NET Entity Framework, kuinka sovelluksen luomat yhteydet tietokantaan toimivat ADO.NET Entity Frameworkin lävitse ja mitä eri kerroksia siihen sisältyy. Entity Frameworkista on olemassa myös versio Entity Framework Core, joka toimii alustariippumattoman .NET Coren päällä (“Entity Framework Core” 2020). Tässä tutkielmassa keskitytään kuitenkin pidempään käytössä olleeseen, .NET Frameworkin päällä toimivaan versioon.

3.1 Yleiskuvaus

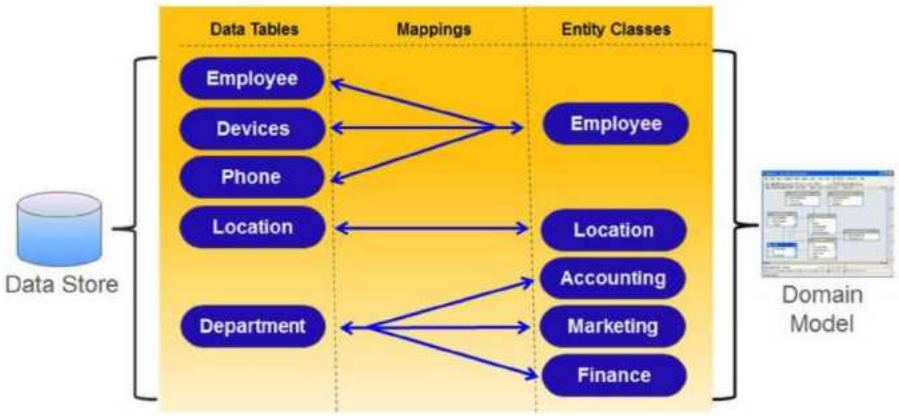
Entity Frameworkia käytettäessä kehitettävä sovellus käyttää Entity Frameworkia, joka puolestaan käyttää tietokantaa ADO.NET:n lävitse (kuva 2). Entity Frameworkin tuottajamalli rakentuu ADO.NET:n tuottajamallin päälle ja täydentää toiminnallisuutta mm. entiteettitietomallilla (EDM) sekä uudella tiedon manipulointiin tarkoitetulla kielellä (DML, Data Manipulation Language), Entity SQL:llä (“Anatomy of the ADO.NET entity framework” 2007).



Kuvio 2. Entity Framework käyttää tietokantaa ADO.NET:n lävitse (Singh 2015, s.8).

3.2 Entity Framework

Avoimen lähdekoodin (“Git repository for Entity Framework 6” 2016) Entity Framework on Microsoftin ADO.NET:in päällä toimiva laajennus(Mueller 2013, s. 3), joka tarjoaa korkeamman abstraktiotason (Singh 2015, s. 7) tietojen käsittelyyn. Microsoftin Entity Framework on ORM (Object Relational Mapper), joka sallii tietokantaa hyödyntävien ohjelmien toteuttamisen kiinnittämättä juurikaan huomiota siihen, miten ja minne tiedot tallennetaan. Entity Frameworkia käytettäessä tietokantakyselyitä tai -komentoja ei tarvitse kirjoittaa itse, vaan Entity Framework muuttaa kehittäjän korkeamman abstraktiotason oliotietomallia (engl. EDM, Entity Data Model) käyttävän koodin tietokannan ymmärtämiksi komennoiksi (kuva 3). Entity Frameworkin piilottaessa käytettävän tietokantaratkaisun, ei kehittäjän peruskäytössä periaatteessa tarvitsisi edes tietää mikä tietokanta on käytössä tietojen tallennukseen. Entity Frameworkin ensimmäinen CTP-versio (Community Technical Preview) julkaistiin vuonna 2006 (Jennings 2009, s. 353).



Kuvio 3. Entiteettitietomallin kuvaus. Kuvassa esitetään, kuinka vasemmalla sijaitsevat tietokannan taulujen tiedot linkitetään oikealla sijaitseviin entiteettien luokkakuvauksiin. (Hirani 2013, s. 3).

Entity Frameworkin entiteettitietomallin tietoja kysytään ja muokataan pääsääntöisesti joko käyttämällä LINQ:a (engl. language-integrated query) entiteeteille tai kirjoittamalla komennot Entity SQL -kielellä (“Anatomy of the ADO.NET entity framework” 2007).

LINQ to entities suorittaa vahvasti tyypitettyjä kyselyitä ja komentoja Entity Frameworkin entiteettitietomalliin. ADO.NET:n ja tietokantakohtaisen tuottajan vastuulle jää muuntaa en-

titeettitietomallia vasten muodostetut kyselyt tietokantaan kohdistuviksi kyselyiksi. LINQ to entities ei tue kaikkia LINQ:n ominaisuuksia vaan sillä on joitain rajoitteita (“Known Issues and Considerations in LINQ to Entities” 2017). Näihin rajoitteisiin ei keskitytä tämän tutkielman osalta, koska rajoitteet ovat samoja kaikille tietokantakohtaisille tuottajille.

Entity SQL suorittaa SQL-kyselyitä (engl. structured query language) Entity Frameworkin entiteettitietomalliin. ADO.NET ja tietokantakohtainen tuottaja muuntavat Entity SQL -kyselyt ja -komennot tietokantakohtaisiksi lauseiksi.

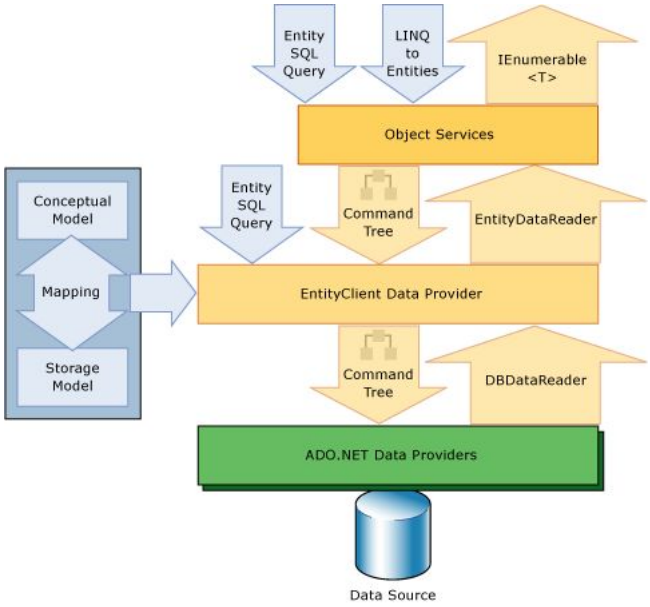
3.3 Kanoniset funktiot

Molemmilla kyselytavoilla (LINQ to entities ja entity SQL) voidaan kyselyissä ja komennoissa käyttää myös kanonisia funktioita (engl. canonical functions). Kanoniset funktiot ovat funktioita, jotka tuottaja muuntaa tietokannan suoritettavaksi funktioksi (Hirani 2013, s. 403). Tietokannan suorittamat funktiot ovat optimoitu niitä suorittaville tietokannalle ja tästä syystä niitä kannattaa suosia. Tuettuihin kanonisiin funktioihin kuuluu esimerkiksi koostefunktioita, matemaattisia funktioita, merkkijonojen käsittelyyn liittyviä funktioita, ajan käsittelyyn liittyviä funktioita, spatiaalisia funktioita ja bittioperaatiofunktioita. Kanoniset funktiot vaativat tietokannoilta laajaa tukea funktioille ja on oletettavaa, etteivät kaikki tietokannat tue kaikkia funktioita.

3.4 Tuottaja

Entity Frameworkia tukeva ADO.NET tuottaja huolehtii Entity Frameworkin alla sijaitsevan ADO.NET:n ja tietokannan välisestä liikenteestä (kuva 4).

Tuottaja muuttaa Entity Frameworkin kyselyt ja komennot tietokantakohtaisiksi kyselyiksi ja komennoiksi. Suurin osa toteutettavista ominaisuuksista on valinnaisia, joten tuottajan toimittaja voi toteuttaa ne ominaisuudet, jotka kyseisessä tietokannassa ovat mahdollisia ja jättää loput toteuttamatta. Tästä johtuen kaikki tuottajat eivät toteuta välttämättä samoja ominaisuuksia, eivätkä samaa tietokantaa tukevat tuottajat välttämättä toteuta samoja toiminnallisuuksia samalla tavalla.



Kuvio 4. Käytettävää tietokantaa voidaan vaihtaa vaihtamalla tietokantakohtaisen ADO.NET tuottajan (“Entity Framework overview” 2018).

3.5 Mallintamistavan valinta

Entity Frameworkin käyttöönotossa tehdään valinta siitä, miltä pohjalta entiteettitietomalli luodaan. Mallintamistavan valintaan vaikuttavat sekä kehittäjän mieltymykset (koodilähtökohtainen vai mallinnuslähtökohtainen ajattelutapa) että tietokannan olemassa ole (onko kehitettävän sovelluksen tietokanta jo olemassa vai ei) 5. Entiteettitietomalli sisältää sekä entiteettejä kuvaavat luokat että tietokantakontekstia kuvaavan luokan. Kolme mahdollista lähestymistapaa 5 ovat (“Development Approaches with Entity Framework” 2020):

- Database-first, jossa entiteettimalli luodaan olemassa olevan tietokannan pohjalta. Entiteettejä kuvaavat luokat generoidaan mallin pohjalta.
- Code-first, jossa entiteettimallin muodostuu kirjoittamalla entiteettejä kuvaavat luokat ja määritykset koodiin (Mueller 2013, s. 52). Tietokanta luodaan tai päivitetään tuoreimpaan versioon entiteettimallin pohjalta luotujen migraatioiden avulla.
  - Tämä lähtötapa on mahdollinen myös jo olemassa olevan tietokannan kanssa, jolloin koodi muodostetaan yleensä tarkoitukseen suunniteltuja työkaluja avuksi käyttäen.

- Model-first, jossa entiteettimalli luodaan tietorakennetta kuvaavan mallin pohjalta. Tietomallista tuotetaan sekä entiteettejä kuvaavat luokat että tietokanta.

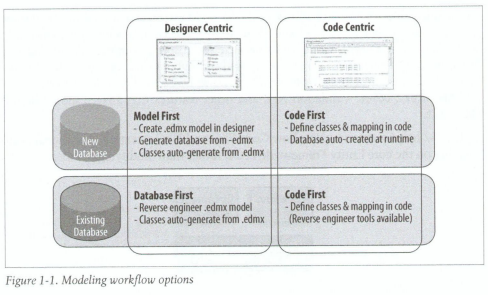


Figure 1-1. Modeling workflow options

Kuvio 5. Vaihtoehtoisia tapoja mallintaa Entity Frameworkin kanssa (Lerman 2011).

Tämän tutkielman testiosuudessa Entity Frameworkin kanssa käytetään lähestymistapaa Code-first. Entiteettejä kuvaavat luokat ja tietokantakontekstia kuvaava luokka luodaan ensin ja annetaan tietokantakohtaisten tuottajien luoda automaattisesti migraatioilla tarvittavat rakenteet tietokantoihin testien alkaessa.

4 Tutkimuksen toteuttaminen

Tässä tutkimuksessa tarkastellaan Entity Frameworkia käyttävän ohjelmakoodin toimintaa eri tietokantojen päällä tietokantakohtaisia tuottajia käyttäen. Tutkimus suoritetaan testaamalla tuottajien tukea olioiden perustietotyypeille, spatiaalisille tietotyypeille ja funktioille sekä muille kanonisille funktioille, kun komentoja suoritetaan Entity Frameworkin läpi.

Tietokantaan tallennettavat entiteetit mallinnetaan oliotietomalliin ja konfiguroidaan fluent API -menetelmää hyödyntäen ("Fluent API - Configuring and Mapping Properties and Types" 2016). Tietomalli konfiguroidaan myös luomaan tarvitsemansa kanta sisäisine rakenteineen automaattisesti pyydettäessä Code-first -lähestymistapaa hyödyntäen.

Tietokantakohtaisten tuottajien tukemien ominaisuuksien testausta varten testit kirjoitetaan yksikkötestimuotoon (NUnit), joilla testin eri osa-alueita on mahdollista pienellä vaivalla ajaa automaattisesti eri tuottajia vasten (Hamilton 2004). Testit ovat funktionaalisesti kuitenkin hyväksyntätestejä (acceptance testing), koska testissä mennään sovelluskerrokselta tietokantaan asti ja testataan koko putken toimintaa (Hamilton 2004, s. 5). Testin eri osa-alueilla testataan seuraavia asioita:

- Tuki olioiden perustietotyypeille
- Tuki spatiaalisille tietotyypeille ja funktioille
- Tuki koostefunktiolle
- Tuki matemaattisille funktioille
- Tuki merkkijonofunktioille
- Tuki ajankäsittelyn funktioille
- Tuki bittioperaatioille

Jokainen ominaisuus (perustietotyyppi tai funktio) testataan omassa testimetodissaan. Ennen jokaista testimetodin suoritusta, tietokanta alustetaan ennalta määrättyyn tilaan haluttuine testidatoineen, jotta testit eivät vahingossakaan pääse vaikuttamaan toisiinsa.

Tutkimuksessa tarkastellaan vain tietokantakohtaisten tuottajien ja tietokantojen toimintaa kokonaisuutena, ottamatta kantaa erikseen tietokantojen tai tuottajien aiheuttamiin rajoittei-

siin.

4.1 Testattavien tuottajien valinta

Tässä tutkielmassa testataan aiemmassa kirjallisuuskartoituksessa (Moilanen 2016) läpikäydyt tuottajat, joita ovat:

- Microsoftin MS SQL -tuottaja (MS SQL)
- Microsoftin MS SQL CE -tuottaja (MS SQL CE)
- Devartin dotConnect for Oracle (Oracle)
- Oraclen ODP.NET -tuottaja (Oracle)
- MySQL connector/NET (MySQL)
- Devartin dotConnect for MySQL (MySQL)
- CData MySQL ADO.NET provider (MySQL)
- CData MongoDB ADO.NET provider (MongoDB)
- Devartin dotConnect for PostgreSQL (PostgreSQL)
- Npgsql (PostgreSQL)

Tuottajista on valittu tähän vertailuun viimeisimmät testihetkellä saatavilla olevat versiot.

4.2 Tutkimuksen kulku

ORM:n perusominaisuuksiin kuuluu olioiden lisäys-, luku-, päivitys- ja poisto-operaatiot (Lipitsäinen 2010). Perustietotyyppien tuen testaamiseksi erilaisia tietotyyppejä sisältäviä entiteettejä tullaan lisäämään, lukemaan ja poistamaan tietokannoista Entity Frameworkin lävitse. Kanonisten funktioiden testaamiseksi ennen testejä lisätään tietokantaan testidataa, jota vasten funktioita suoritetaan.

4.3 Perustietotyypit

Tuottajien tukea kaikille Microsoftin entiteettitietomallissa (Entity Data Model) määrittelemille primitiivisille tietotyypeille (“Entity Data Model: Primitive Data Types” 2017) testa-

taan luomalla testattavalle entiteetille ominaisuuksia kullakin testattavalla tietotyypillä. Nämä primitiiviset tietotyypit ovat:

- Binary (byte[])
- Boolean (bool)
- Byte
- DateTime
- DateTimeOffset
- Decimal
- Double
- Float
- Guid
- Int16 (short)
- Int32 (int)
- Int64 (long)
- SByte
- String
- Time (TimeSpan)

Jokainen perustietotyyppi testataan erikseen tallentamalla kyseistä tietotyyppiä oleva attribuutti kantaan taulukon 1 kuvaamilla arvoilla, lukemalla tallennetut arvot tämän jälkeen kannasta ja vertailemalla kannasta palautettua arvoa alun perin lisättyyn arvoon. Osalla tietotyypeistä tiedon käsittelyn suuruusluokkaa (scale) ja tarkuuttaa (precision) on mahdollista konfiguroida ja konfigurointia tehdään tarvittaessa testitapausten sitä vaatiessa.

Perustietotyypien testiarvot				
Funktio	Arvo 1	Arvo 2	Arvo 3	Arvo 4
Boolean	true	false		
Byte	Byte.MinValue	0	Byte.MaxValue	
Byte[]	null	new byte[]	{0x01, 0x02, 0x03}	
DateTime	DateTime.MinValue	DateTime.UtcNow	DateTime.MaxValue	
DateTimeOffset	2020-01-01 T00:00:00-11:59	2020-01-01 T00:00:00	2020-01-01 T00:00:00+11:59	
Decimal	Decimal.MinValue	0	9876543.21m	Decimal.MaxValue
Double	Double.MinValue	0	123.456	Double.MaxValue
Float	float.MinValue	0	123.456f	float.MaxValue
Guid	Guid.Empty	Guid.NewGuid()		
Int16	Int16.MinValue	0	Int16.MaxValue	
Int32	Int32.MinValue	0	Int32.MaxValue	
Int64	Int64.MinValue	0	Int64.MaxValue	
SByte	SByte.MinValue	0	SByte.MaxValue	
String	null	String.Empty	"ääkkönen åland"	
Time	-23:59:59.999	TimeSpan.Zero	+23:59:59.999	

Taulukko 1: Perustietotyypien testiarvot

Lisäksi testattavilla entiteeteillä on määriteltynä tunnistekenttä "Id"(int), joka toimii oletuksena entiteetin uniikkina tunnisteena, ollen samalla entiteetin yksilöivä primääriavain (primary key) tietokantataulussa.

4.4 Kanoniset funktiot

Entity framework tukee kyselyissä seuraavia kanonisia funktioita (“Canonical Functions” 2017):

- Koostefunktiot (aggregate canonical functions)
- Matemaattiset funktiot (Math canonical functions)

- Merkkijonofunktiot (string canonical functions)
- Ajankäsittelyn funktiot (date and time canonical functions)
- Bittioperaatiot (bitwise canonical functions))
- Spatiaaliset funktiot (spatial functions)
- Muut kanoniset funktiot (other canonical functions)

Tässä tutkielmassa testataan tuottajien tukea valituille kanonisille funktioille valiten toiminnallisuuksia ylläolevan listauksen jokaisesta luokasta, poislukien "muut kanoniset funktiot".

4.4.1 Koostefunktiot

Koostefunktioiden testaamiseksi kantaan syötetään joukko listauksessa 4.1 näkyviä arvoja. Tätä joukkoa vasten kutakin koostefunktiota testataan.

```
[SetUp]
public void Setup()
{
    _aggregateFunctions = new AggregateFunctions();
    _aggregateFunctions.DeleteAll();

    _aggregateFunctions.Add(new AggregateCanonicalTestEntity { Value = -0.5m });
    _aggregateFunctions.Add(new AggregateCanonicalTestEntity { Value = 0m });
    _aggregateFunctions.Add(new AggregateCanonicalTestEntity { Value = 1m });
    _aggregateFunctions.Add(new AggregateCanonicalTestEntity { Value = 1m });
    _aggregateFunctions.Add(new AggregateCanonicalTestEntity { Value = 1.5m });
    _aggregateFunctions.Add(new AggregateCanonicalTestEntity { Value = 2m });
    _aggregateFunctions.Add(new AggregateCanonicalTestEntity { Value = 2m });
    _aggregateFunctions.Add(new AggregateCanonicalTestEntity { Value = 2m });
    _aggregateFunctions.Add(new AggregateCanonicalTestEntity { Value = 2m });
    _aggregateFunctions.Add(new AggregateCanonicalTestEntity { Value = 4m });
}
```

Esimerkkikoodi 4.1. Koostefunktioiden testidatan alustus

Koostefunktioita testataan seuraavasti:

- Avg (keskiarvo)

```
[Test]
public void Test_Avg()
{
    Assert.AreEqual(1.5m, _aggregateFunctions.Avg());
}
```

Esimerkkikoodi 4.2. Testi Avg-funktiolle

- BigCount (lukumäärä Int64)

```
[Test]
public void Test_BigCount()
{
    Assert.AreEqual(10, _aggregateFunctions.BigCount());
}
```

Esimerkkikoodi 4.3. Testi BigCount-funktiolle

- Count (lukumäärä Int32)

```
[Test]
public void Test_Count()
{
    Assert.AreEqual(10, _aggregateFunctions.Count());
}
```

Esimerkkikoodi 4.4. Testi Count-funktiolle

- Max (maksimi)

```
[Test]
public void Test_Max()
{
    Assert.AreEqual(4m, _aggregateFunctions.Max());
}
```

Esimerkkikoodi 4.5. Testi Max-funktiolle

- Min (minimi)

```
[Test]
public void Test_Min()
{
    Assert.AreEqual(-0.5m, _aggregateFunctions.Min());
}
```

Esimerkkikoodi 4.6. Testi Min-funktiolle

- StDev (keskihajonnan arvio näytteelle)

```
[Test]
public void Test_StDev()
{
    Assert.IsTrue(_aggregateFunctions.StDev()-1.2472 < 0.0001);
}
```

Esimerkkikoodi 4.7. Testi StDev-funktiolle

- StDevP (keskihajonnan lasku populaatiolle)

```
[Test]
public void Test_StDevP()
{
    Assert.IsTrue(_aggregateFunctions.StDevP() - 1.1832 < 0.0001);
}
```

Esimerkkikoodi 4.8. Testi StDevP-funktiolle

- Sum (summa)

```
[Test]
public void Test_Sum()
{
    Assert.AreEqual(15m, _aggregateFunctions.Sum());
}
```

Esimerkkikoodi 4.9. Testi Sum-funktiolle

- Var (varianssi)

```
[Test]
public void Test_Var()
{
    Assert.IsTrue(_aggregateFunctions.Var() - 1.5555 < 0.0001);
}
```

Esimerkkikoodi 4.10. Testi Var-funktiolle

- VarP (varianssin lasku populaatiolle)

```
[Test]
public void Test_VarP()
{
    Assert.IsTrue(_aggregateFunctions.VarP() - 4 < 0.0001);
}
```

Esimerkkikoodi 4.11. Testi VarP-funktiolle

4.4.2 Matemaattiset funktiot

Matemaattisten funktioiden testaamiseksi kantaan syötetään joukko listauksessa 4.12 näkyviä arvoja. Näitä arvoja vasten matemaattisia funktioita testataan.

```
[SetUp]
public void Setup()
{
    _mathFunctions = new MathFunctions();
    _mathFunctions.DeleteAll();

    _idOnePointFive = _mathFunctions.Add(new MathCanonicalTestEntity { Value = 1.5m });
    _idAlmostOnePointFive = _mathFunctions.Add(new MathCanonicalTestEntity { Value = 1.49m });
    _idAlmostOne = _mathFunctions.Add(new MathCanonicalTestEntity { Value = 0.99m });
    _idZero = _mathFunctions.Add(new MathCanonicalTestEntity { Value = 0m });
    _idMinusOnePointFive = _mathFunctions.Add(new MathCanonicalTestEntity { Value = -1.5m });
    _idMinusAlmostOnePointFive = _mathFunctions.Add(new MathCanonicalTestEntity { Value = -1.49m });
    _idLittleBitOverZero = _mathFunctions.Add(new MathCanonicalTestEntity { Value = 0.01m });
    _idTwo = _mathFunctions.Add(new MathCanonicalTestEntity { Value = 2m });
}
```

Esimerkkikoodi 4.12. Matemaattisten funktioiden testidatan alustus

Matemaattisia funktioita testataan seuraavasti:

- Abs (absoluuttinen arvo)

```
[Test]
public void Test_Abs()
{
    Assert.AreEqual(1.5m, _mathFunctions.Abs(_idMinusOnePointFive));
    Assert.AreEqual(1.49m, _mathFunctions.Abs(_idMinusAlmostOnePointFive));
    Assert.AreEqual(1.5m, _mathFunctions.Abs(_idOnePointFive));
    Assert.AreEqual(0m, _mathFunctions.Abs(_idZero));
}
```

Esimerkkikoodi 4.13. Testimetodi Abs-funktiolle

- Ceiling (pienin kokonaisluku, joka on annettua arvoa suurempi)

```
[Test]
public void Test_Ceiling()
{
    Assert.AreEqual(0m, _mathFunctions.Ceiling(_idZero));
    Assert.AreEqual(1m, _mathFunctions.Ceiling(_idLittleBitOverZero));
    Assert.AreEqual(-1m, _mathFunctions.Ceiling(_idMinusOnePointFive));
}
```

Esimerkkikoodi 4.14. Testimetodi Ceiling-funktiolle

- Floor (suurin kokonaisluku, joka on annettua arvoa pienempi)

```
[Test]
public void Test_Floor()
{
    Assert.AreEqual(0m, _mathFunctions.Floor(_idZero));
    Assert.AreEqual(0m, _mathFunctions.Floor(_idAlmostOne));
    Assert.AreEqual(1m, _mathFunctions.Floor(_idOnePointFive));
    Assert.AreEqual(-2m, _mathFunctions.Floor(_idMinusOnePointFive));
}
```

Esimerkkikoodi 4.15. Testimetodi Floor-funktiolle

- Power (potenssi)

```
[Test]
public void Test_Power()
{
    Assert.AreEqual(0m, _mathFunctions.Power(_idZero, 1));
    Assert.AreEqual(0m, _mathFunctions.Power(_idZero, 2));
    Assert.AreEqual(1m, _mathFunctions.Power(_idZero, 0));
    Assert.AreEqual(2m, _mathFunctions.Power(_idTwo, 1));
    Assert.AreEqual(4m, _mathFunctions.Power(_idTwo, 2));
    Assert.AreEqual(8m, _mathFunctions.Power(_idTwo, 3));
    Assert.AreEqual(16m, _mathFunctions.Power(_idTwo, 4));
    Assert.AreEqual(2.25m, _mathFunctions.Power(_idOnePointFive, 2));
}
```

Esimerkkikoodi 4.16. Testimetodi Power-funktiolle

- Round (pyöristys)

```
[Test]
public void Test_Round()
{
    Assert.AreEqual(0m, _mathFunctions.Round(_idZero, 0));
    Assert.AreEqual(1m, _mathFunctions.Round(_idAlmostOnePointFive, 0));
    Assert.AreEqual(1.5m, _mathFunctions.Round(_idAlmostOnePointFive, 1));
    Assert.AreEqual(1.49m, _mathFunctions.Round(_idAlmostOnePointFive, 2));
    Assert.AreEqual(0m, _mathFunctions.Round(_idLittleBitOverZero, 0));
    Assert.AreEqual(0m, _mathFunctions.Round(_idLittleBitOverZero, 1));
    Assert.AreEqual(0.01m, _mathFunctions.Round(_idLittleBitOverZero, 2));
    Assert.AreEqual(-1m, _mathFunctions.Round(_idMinusAlmostOnePointFive, 0));
    Assert.AreEqual(-1.5m, _mathFunctions.Round(_idMinusAlmostOnePointFive, 1));
    Assert.AreEqual(2m, _mathFunctions.Round(_idOnePointFive, 0));
}
```

Esimerkkikoodi 4.17. Testimetodi Round-funktiolle

- Truncate (luvun tarkkuuden pienentäminen ilman pyöristystä)

```
[Test]
public void Test_Truncate()
{
    Assert.AreEqual(0m, _mathFunctions.Truncate(_idZero, 0));
    Assert.AreEqual(0m, _mathFunctions.Truncate(_idAlmostOne, 0));
    Assert.AreEqual(0.9m, _mathFunctions.Truncate(_idAlmostOne, 1));
    Assert.AreEqual(-1m, _mathFunctions.Truncate(_idMinusAlmostOnePointFive, 0));
    Assert.AreEqual(-1.4m, _mathFunctions.Truncate(_idMinusAlmostOnePointFive, 1));
    Assert.AreEqual(-1.49m, _mathFunctions.Truncate(_idMinusAlmostOnePointFive, 2));
}
```

Esimerkkikoodi 4.18. Testimetodi Truncate-funktiolle

4.4.3 Merkkijonofunktiot

Merkkijonofunktioiden testaamiseksi kantaan syötetään joukko listauksessa 4.19 näkyviä arvoja. Näitä arvoja vasten merkkijonofunktioita testataan.

```
[SetUp]
public void Setup()
{
    _stringFunctions = new StringFunctions();
    _stringFunctions.DeleteAll();

    _stringAbcId = _stringFunctions.Add(new StringCanonicalTestEntity {Value = "abc"});
    _stringAbcUpperCaseId = _stringFunctions.Add(new StringCanonicalTestEntity { Value = "ABC" });
    _stringAbcWithSpacesId = _stringFunctions.Add(new StringCanonicalTestEntity { Value = " abc " });
    _stringAbcWithDoubleSpacesId = _stringFunctions.Add(new StringCanonicalTestEntity { Value = "  abc  " });
    _stringEmptyId = _stringFunctions.Add(new StringCanonicalTestEntity { Value = "" });
    _stringAbcWithSpaceInBegin = _stringFunctions.Add(new StringCanonicalTestEntity { Value = " abc" });
    _stringAbcWithSpaceInEnd = _stringFunctions.Add(new StringCanonicalTestEntity { Value = "abc " });
}
```

Esimerkkikoodi 4.19. Merkkijonofunktioiden testidatan alustus

Merkkijonofunktioita testataan seuraavasti:

- Concat (yhdistäminen)

```
[Test]
public void Test_Concat()
{
    Assert.AreEqual("abcxyz", _stringFunctions.Concat(_stringAbcId, "xyz"));
    Assert.AreEqual("abc", _stringFunctions.Concat(_stringAbcId, ""));
    Assert.AreEqual("xyz", _stringFunctions.Concat(_stringEmptyId, "xyz"));
    Assert.AreEqual("", _stringFunctions.Concat(_stringEmptyId, ""));
}
```

Esimerkkikoodi 4.20. Testimetodi Concat-funktiolle

- Contains (sisältyvyys)

```
[Test]
public void Test_Contains()
{
    Assert.AreEqual(true, _stringFunctions.Contains(_stringAbcId, "abc"));
    Assert.AreEqual(true, _stringFunctions.Contains(_stringAbcId, "bc"));
    Assert.AreEqual(true, _stringFunctions.Contains(_stringAbcId, "c"));
    Assert.AreEqual(true, _stringFunctions.Contains(_stringAbcId, "a"));
    Assert.AreEqual(true, _stringFunctions.Contains(_stringAbcWithDoubleSpacesId, " "));
    Assert.AreEqual(true, _stringFunctions.Contains(_stringAbcWithDoubleSpacesId, "  "));
    Assert.AreEqual(true, _stringFunctions.Contains(_stringAbcWithDoubleSpacesId, "  a"));
    Assert.AreEqual(true, _stringFunctions.Contains(_stringAbcWithDoubleSpacesId, "c  "));
    Assert.AreEqual(true, _stringFunctions.Contains(_stringAbcWithSpacesId, " "));

    Assert.AreEqual(false, _stringFunctions.Contains(_stringAbcId, "cb"));
    Assert.AreEqual(false, _stringFunctions.Contains(_stringAbcId, "xyz"));
    Assert.AreEqual(false, _stringFunctions.Contains(_stringAbcId, "ac"));
    Assert.AreEqual(false, _stringFunctions.Contains(_stringAbcId, " "));
    Assert.AreEqual(false, _stringFunctions.Contains(_stringEmptyId, "a"));
    Assert.AreEqual(false, _stringFunctions.Contains(_stringEmptyId, "b"));
    Assert.AreEqual(false, _stringFunctions.Contains(_stringEmptyId, " "));
}
```

Esimerkkikoodi 4.21. Testimetodi Contains-funktiolle

- EndsWith (merkkijonon päättyminen)

```
[Test]
public void Test_EndsWith()
{
    Assert.AreEqual(true, _stringFunctions.EndsWith(_stringAbcId, "abc"));
    Assert.AreEqual(true, _stringFunctions.EndsWith(_stringAbcId, "bc"));
    Assert.AreEqual(true, _stringFunctions.EndsWith(_stringAbcId, "c"));
    Assert.AreEqual(true, _stringFunctions.EndsWith(_stringAbcWithDoubleSpacesId, " "));
    Assert.AreEqual(true, _stringFunctions.EndsWith(_stringAbcWithDoubleSpacesId, "  "));

    Assert.AreEqual(false, _stringFunctions.EndsWith(_stringAbcId, "b"));
    Assert.AreEqual(false, _stringFunctions.EndsWith(_stringEmptyId, "a"));
    Assert.AreEqual(false, _stringFunctions.EndsWith(_stringAbcWithDoubleSpacesId, "c"));
}
```

Esimerkkikoodi 4.22. Testimetodi EndsWith-funktiolle

- IndexOf (merkkijonon sijainnin etsiminen toisen merkkijonon sisältä)

```
[Test]
public void Test_IndexOf()
{
    Assert.AreEqual(1, _stringFunctions.IndexOf(_stringAbcId, "abc"));
    Assert.AreEqual(2, _stringFunctions.IndexOf(_stringAbcId, "bc"));
    Assert.AreEqual(2, _stringFunctions.IndexOf(_stringAbcId, "b"));
    Assert.AreEqual(3, _stringFunctions.IndexOf(_stringAbcId, "c"));
    Assert.AreEqual(1, _stringFunctions.IndexOf(_stringAbcWithDoubleSpacesId, " "));
    Assert.AreEqual(1, _stringFunctions.IndexOf(_stringAbcWithDoubleSpacesId, "  "));
    Assert.AreEqual(0, _stringFunctions.IndexOf(_stringAbcId, "d"));
    Assert.AreEqual(0, _stringFunctions.IndexOf(_stringEmptyId, "cb"));
    Assert.AreEqual(0, _stringFunctions.IndexOf(_stringEmptyId, "xyz"));
}
```

Esimerkkikoodi 4.23. Testimetodi IndexOf-funktiolle

- Left (uuden annetun mittaisen merkkijonon luonti vasemmalta alkaen)

```
[Test]
public void Test_Left()
{
    Assert.AreEqual("abc", _stringFunctions.Left(_stringAbcId, 4));
    Assert.AreEqual("abc", _stringFunctions.Left(_stringAbcId, 3));
    Assert.AreEqual("ab", _stringFunctions.Left(_stringAbcId, 2));
    Assert.AreEqual("a", _stringFunctions.Left(_stringAbcId, 1));
    Assert.AreEqual("", _stringFunctions.Left(_stringAbcId, 0));
    Assert.AreEqual(" ", _stringFunctions.Left(_stringAbcWithDoubleSpacesId, 1));
    Assert.AreEqual("  ", _stringFunctions.Left(_stringAbcWithDoubleSpacesId, 2));
}
```

Esimerkkikoodi 4.24. Testimetodi Left-funktiolle

- Length (merkkijonon pituus)

```
[Test]
public void Test_Length()
{
    Assert.AreEqual(3, _stringFunctions.Length(_stringAbcId));
    Assert.AreEqual(4, _stringFunctions.Length(_stringAbcWithSpaceInBegin));
    Assert.AreEqual(4, _stringFunctions.Length(_stringAbcWithSpaceInEnd));
    Assert.AreEqual(5, _stringFunctions.Length(_stringAbcWithSpacesId));
    Assert.AreEqual(7, _stringFunctions.Length(_stringAbcWithDoubleSpacesId));
    Assert.AreEqual(0, _stringFunctions.Length(_stringEmptyId));
}
```

Esimerkkikoodi 4.25. Testimetodi Length-funktiolle

- LTrim (poistaa välilyönnit merkkijonon alusta)

```
[Test]
public void Test_LTrim()
{
    Assert.AreEqual("abc", _stringFunctions.LTrim(_stringAbcId));
    Assert.AreEqual("", _stringFunctions.LTrim(_stringEmptyId));
    Assert.AreEqual("abc  ", _stringFunctions.LTrim(_stringAbcWithDoubleSpacesId));
    Assert.AreEqual("abc ", _stringFunctions.LTrim(_stringAbcWithSpacesId));
}
```

Esimerkkikoodi 4.26. Testimetodi LTrim-funktiolle

- Replace (korvaaminen)

```
[Test]
public void Test_Replace()
{
    Assert.AreEqual("adec", _stringFunctions.Replace(_stringAbcId, "b", "de"));
    Assert.AreEqual("xyzabcxyz", _stringFunctions.Replace(_stringAbcWithDoubleSpacesId, "  ", "xyz"));
    Assert.AreEqual("  xyz  ", _stringFunctions.Replace(_stringAbcWithDoubleSpacesId, "abc", "xyz"));
}
```

Esimerkkikoodi 4.27. Testimetodi Replace-funktiolle

- Reverse (käänteinen järjestys)

```
[Test]
public void Test_Reverse()
{
    Assert.AreEqual("cba", _stringFunctions.Reverse(_stringAbcId));
    Assert.AreEqual("  cba  ", _stringFunctions.Reverse(_stringAbcWithDoubleSpacesId));
    Assert.AreEqual("", _stringFunctions.Reverse(_stringEmptyId));
}
```

Esimerkkikoodi 4.28. Testimetodi Reverse-funktiolle

- Right (uuden annetun mittaisen merkkijonon luonti merkkijonon viimeisistä kirjaimista)

```
[Test]
public void Test_Right()
{
    Assert.AreEqual("abc", _stringFunctions.Right(_stringAbcId, 4));
    Assert.AreEqual("abc", _stringFunctions.Right(_stringAbcId, 3));
    Assert.AreEqual("bc", _stringFunctions.Right(_stringAbcId, 2));
    Assert.AreEqual("c", _stringFunctions.Right(_stringAbcId, 1));
    Assert.AreEqual("", _stringFunctions.Right(_stringAbcId, 0));
    Assert.AreEqual(" ", _stringFunctions.Right(_stringAbcWithDoubleSpacesId, 1));
    Assert.AreEqual("  ", _stringFunctions.Right(_stringAbcWithDoubleSpacesId, 2));
}
```

Esimerkkikoodi 4.29. Testimetodi Right-funktiolle

- RTrim (poistaa välilyönnit merkkijonon lopusta)

```
[Test]
public void Test_RTrim()
{
    Assert.AreEqual("abc", _stringFunctions.RTrim(_stringAbcId));
    Assert.AreEqual("", _stringFunctions.RTrim(_stringEmptyId));
    Assert.AreEqual("  abc", _stringFunctions.RTrim(_stringAbcWithDoubleSpacesId));
    Assert.AreEqual(" abc", _stringFunctions.RTrim(_stringAbcWithSpacesId));
}
```

Esimerkkikoodi 4.30. Testimetodi RTrim-funktiolle

- StartsWith (merkkijonon alkaminen)

```
[Test]
public void Test_StartsWith()
{
    Assert.AreEqual(true, _stringFunctions.StartsWith(_stringAbcId, "abc"));
    Assert.AreEqual(true, _stringFunctions.StartsWith(_stringAbcId, "ab"));
    Assert.AreEqual(true, _stringFunctions.StartsWith(_stringAbcId, "a"));
    Assert.AreEqual(true, _stringFunctions.StartsWith(_stringAbcWithDoubleSpacesId, " "));
    Assert.AreEqual(true, _stringFunctions.StartsWith(_stringAbcWithDoubleSpacesId, "  "));

    Assert.AreEqual(false, _stringFunctions.StartsWith(_stringAbcId, "b"));
    Assert.AreEqual(false, _stringFunctions.StartsWith(_stringEmptyId, "c"));
    Assert.AreEqual(false, _stringFunctions.StartsWith(_stringAbcWithSpacesId, "a"));
}
```

Esimerkkikoodi 4.31. Testimetodi StartsWith-funktiolle

- Substring (uuden annetun mittaisen merkkijonon luonti annetusta indeksistä alkaen)

```
[Test]
public void Test_Substring()
{
    Assert.AreEqual("abc", _stringFunctions.SubString(_stringAbcId, 1, 3));
    Assert.AreEqual("c", _stringFunctions.SubString(_stringAbcId, 3, 1));
    Assert.AreEqual("bc", _stringFunctions.SubString(_stringAbcId, 2, 2));
    Assert.AreEqual("  ", _stringFunctions.SubString(_stringAbcWithDoubleSpacesId, 1, 2));
    Assert.AreEqual("  ab", _stringFunctions.SubString(_stringAbcWithDoubleSpacesId, 1, 4));
    Assert.AreEqual("  ", _stringFunctions.SubString(_stringAbcWithDoubleSpacesId, 6, 2));
}
```

Esimerkkikoodi 4.32. Testimetodi Substring-funktiolle

- ToLower (muuntaa merkkijonon kirjaimet pieniksi kirjaimiksi)

```
[Test]
public void Test_ToLower()
{
    Assert.AreEqual("abc", _stringFunctions.ToLower(_stringAbcId));
    Assert.AreEqual("abc", _stringFunctions.ToLower(_stringAbcUpperCaseId));
    Assert.AreEqual("", _stringFunctions.ToLower(_stringEmptyId));
    Assert.AreEqual("  abc  ", _stringFunctions.ToLower(_stringAbcWithDoubleSpacesId));
    Assert.AreEqual(" abc ", _stringFunctions.ToLower(_stringAbcWithSpacesId));
}
```

Esimerkkikoodi 4.33. Testimetodi ToLower-funktiolle

- ToUpper (muuntaa merkkijonon kirjaimet isoiksi kirjaimiksi)

```
[Test]
public void Test_ToUpper()
{
    Assert.AreEqual("ABC", _stringFunctions.ToUpper(_stringAbcId));
    Assert.AreEqual("ABC", _stringFunctions.ToUpper(_stringAbcUpperCaseId));
    Assert.AreEqual("", _stringFunctions.ToUpper(_stringEmptyId));
    Assert.AreEqual("  ABC  ", _stringFunctions.ToUpper(_stringAbcWithDoubleSpacesId));
    Assert.AreEqual(" ABC ", _stringFunctions.ToUpper(_stringAbcWithSpacesId));
}
```

Esimerkkikoodi 4.34. Testimetodi ToUpper-funktiolle

- Trim (poistaa välilyönnit sekä alusta että lopusta)

```
[Test]
public void Test_Trim()
{
    Assert.AreEqual("abc", _stringFunctions.Trim(_stringAbcId));
    Assert.AreEqual("", _stringFunctions.Trim(_stringEmptyId));
    Assert.AreEqual("abc", _stringFunctions.Trim(_stringAbcWithDoubleSpacesId));
    Assert.AreEqual("abc", _stringFunctions.Trim(_stringAbcWithSpacesId));
}
```

Esimerkkikoodi 4.35. Testimetodi Trim-funktiolle

4.4.4 Ajankäsittelyn funktiot

Ajankäsittelyn funktioiden testaamiseksi kantaan syötetään listauksessa 4.36 kuvattu arvo, jota vasten funktioita testataan (1.1.2020 klo 0:00, aikavyöhyke UTC).

```
[SetUp]
public void Setup()
{
    _dateTimeFunctions = new DateTimeFunctions();

    _dateTimeFunctions.DeleteAll();
    _baseValue = new DateTime(2010, 1, 1, 0, 0, 0, DateTimeKind.Utc);
    _idBaseValue = _dateTimeFunctions.Add(new DateTimeCanonicalTestEntity {Value = _baseValue});
}
```

Esimerkkikoodi 4.36. Ajankäsittelyn funktioiden testidatan alustus

Ajankäsittelyn funktioiden toteutusta testataan .NET frameworkin vastaavaa toiminnallisuutta vasten. Ajankäsittelyn funktioita testataan seuraavasti:

- AddMilliseconds (lisää annetun määrän millisekunteja tietokannasta löytyvään aikaan ja palauttaa tuloksen)

```
[Test]
public void Test_AddMilliseconds()
{
    for (int i = -1; i <= 1; i++)
    {
        var expected = _baseValue.AddMilliseconds(i);
        Assert.AreEqual(expected, _dateTimeFunctions.AddMilliseconds(_idBaseValue, i));
    }
}
```

Esimerkkikoodi 4.37. Testimetodi AddMilliseconds-funktiolle

- AddSeconds (lisää annetun määrän sekunteja tietokannasta löytyvään aikaan ja palauttaa tuloksen)

```
[Test]
public void Test_AddSeconds()
{
    for (int i = -1; i <= 1; i++)
    {
        var expected = _baseValue.AddSeconds(i);
        Assert.AreEqual(expected, _dateTimeFunctions.AddSeconds(_idBaseValue, i));
    }
}
```

Esimerkkikoodi 4.38. Testimetodi AddSeconds-funktiolle

- AddMinutes (lisää annetun määrän minuutteja tietokannasta löytyvään aikaan ja palauttaa tuloksen)

```
[Test]
public void Test_AddMinutes()
{
    for (int i = -1; i <= 1; i++)
    {
        var expected = _baseValue.AddMinutes(i);
        Assert.AreEqual(expected, _dateTimeFunctions.AddMinutes(_idBaseValue, i));
    }
}
```

Esimerkkikoodi 4.39. Testimetodi AddMinutes-funktiolle

- AddHours (lisää annetun määrän tunteja tietokannasta löytyvään aikaan ja palauttaa tuloksen)

```
[Test]
public void Test_AddHours()
{
    for (int i = -1; i <= 1; i++)
    {
        var expected = _baseValue.AddHours(i);
        Assert.AreEqual(expected, _dateTimeFunctions.AddHours(_idBaseValue, i));
    }
}
```

Esimerkkikoodi 4.40. Testimetodi AddHours-funktiolle

- AddDays (lisää annetun määrän päiviä tietokannasta löytyvään aikaan ja palauttaa tuloksen)

```
[Test]
public void Test_AddDays()
{
    for (int i = -1; i <= 1; i++)
    {
        var expected = _baseValue.AddDays(i);
        Assert.AreEqual(expected, _dateTimeFunctions.AddDays(_idBaseValue, i));
    }
}
```

Esimerkkikoodi 4.41. Testimetodi AddDays-funktiolle

- AddMonths (lisää annetun määrän kuukausia tietokannasta löytyvään aikaan ja palauttaa tuloksen)

```
[Test]
public void Test_AddMonths()
{
    for (int i = -1; i <= 1; i++)
    {
        var expected = _baseValue.AddMonths(i);
        Assert.AreEqual(expected, _dateTimeFunctions.AddMonths(_idBaseValue, i));
    }
}
```

Esimerkkikoodi 4.42. Testimetodi AddMonths-funktiolle

- AddYears (lisää annetun määrän vuosia tietokannasta löytyvään aikaan ja palauttaa tuloksen)

```
[Test]
public void Test_AddYears()
{
    for (int i = -1; i <= 1; i++)
    {
        var expected = _baseValue.AddYears(i);
        Assert.AreEqual(expected, _dateTimeFunctions.AddYears(_idBaseValue, i));
    }
}
```

Esimerkkikoodi 4.43. Testimetodi AddYears-funktiolle

- CurrentDateTime (kysyy tietokannasta tämänhetkisen aikaleiman lokaalissa ajassa)

```
[Test]
public void Test_CurrentDateTime()
{
    var dbTime = _dateTimeFunctions.CurrentDateTime();
    var ownTime = DateTime.Now;
    var difference = Math.Abs(dbTime.Ticks - ownTime.Ticks);

    Assert.LessOrEqual(difference, 10000); // times should be within 1 ms
}
```

Esimerkkikoodi 4.44. Testimetodi CurrentDateTime-funktiolle

- CurrentUtcDateTime (kysyy tietokannasta tämänhetkisen aikaleiman UTC-ajassa)

```
[Test]
public void Test_CurrentUtcDateTime()
{
    var dbTime = _dateTimeFunctions.CurrentUtcDateTime();
    var ownTime = DateTime.UtcNow;
    var difference = Math.Abs(dbTime.Ticks - ownTime.Ticks);

    Assert.LessOrEqual(difference, 10000); // times should be within 1 ms
}
```

Esimerkkikoodi 4.45. Testimetodi CurrentUtcDateTime-funktiolle

- Millisecond (palauttaa tietokannassa sijaitsevan aikaleiman millisekunnin)

```
[Test]
public void Test_Millisecond()
{
    Assert.AreEqual(_baseValue.Millisecond, _dateTimeFunctions.Millisecond(_idBaseValue));
}
```

Esimerkkikoodi 4.46. Testimetodi Millisecond-funktiolle

- Second (palauttaa tietokannassa sijaitsevan aikaleiman sekunnin)

```
[Test]
public void Test_Second()
{
    Assert.AreEqual(_baseValue.Second, _dateTimeFunctions.Second(_idBaseValue));
}
```

Esimerkkikoodi 4.47. Testimetodi Second-funktiolle

- Minute (palauttaa tietokannassa sijaitsevan aikaleiman minuutin)

```
[Test]
public void Test_Minute()
{
    Assert.AreEqual(_baseValue.Minute, _dateTimeFunctions.Minute(_idBaseValue));
}
```

Esimerkkikoodi 4.48. Testimetodi Minute-funktiolle

- Hour (palauttaa tietokannassa sijaitsevan aikaleiman tunnin)

```
[Test]
public void Test_Hour()
{
    Assert.AreEqual(_baseValue.Hour, _dateTimeFunctions.Hour(_idBaseValue));
}
```

Esimerkkikoodi 4.49. Testimetodi Hour-funktiolle

- Day (palauttaa tietokannassa sijaitsevan aikaleiman päivän)

```
[Test]
public void Test_Day()
{
    Assert.AreEqual(_baseValue.Day, _dateTimeFunctions.Day(_idBaseValue));
}
```

Esimerkkikoodi 4.50. Testimetodi Day-funktiolle

- DayOfYear (palauttaa tietokannassa sijaitsevan aikaleiman kuluvan vuoden päivän)

```
[Test]
public void Test_DayOfYear()
{
    Assert.AreEqual(_baseValue.DayOfYear, _dateTimeFunctions.DayOfYear(_idBaseValue));
}
```

Esimerkkikoodi 4.51. Testimetodi DayOfYear-funktiolle

- Month (palauttaa tietokannassa sijaitsevan aikaleiman kuukauden)

```
[Test]
public void Test_Month()
{
    Assert.AreEqual(_baseValue.Month, _dateTimeFunctions.Month(_idBaseValue));
}
```

Esimerkkikoodi 4.52. Testimetodi Month-funktiolle

- Year (palauttaa tietokannassa sijaitsevan aikaleiman vuoden)

```
[Test]
public void Test_Year()
{
    Assert.AreEqual(_baseValue.Year, _dateTimeFunctions.Year(_idBaseValue));
}
```

Esimerkkikoodi 4.53. Testimetodi Year-funktiolle

4.4.5 Bittioperaatiot

Bittioperaatioiden testaamiseksi kantaan syötetään tavun (byte, 0-255) kokoisten lukujen pareja, joita vasten testejä suoritetaan. Testatessa funktioita AND, OR ja XOR, testi kohdistuu molempiin kannassa sijaitseviin tavuihin. Testatessa funktiota NOT, testi kohdistuu tavuparin ensimmäiseen tavuun. Tietokantaan lisätään ennen testien suorittamista listauksessa 4.54 kuvatut tavuparit.

```
[SetUp]
public void Setup()
{
    _bitwiseFunctions = new BitwiseFunctions();

    _bitwiseFunctions.DeleteAll();

    _byte0And15Id = _bitwiseFunctions.Add(new BitwiseCanonicalTestEntity {Byte1 = 0, Byte2 = 15}); // 00000000 & 00001111
    _byte0And60Id = _bitwiseFunctions.Add(new BitwiseCanonicalTestEntity {Byte1 = 0, Byte2 = 60}); // 00000000 & 00111100
    _byte0And255Id = _bitwiseFunctions.Add(new BitwiseCanonicalTestEntity {Byte1 = 0, Byte2 = 255}); // 00000000 &
        11111111

    _byte15And60Id = _bitwiseFunctions.Add(new BitwiseCanonicalTestEntity {Byte1 = 15, Byte2 = 60}); // 00001111 &
        00111100
    _byte15And255Id = _bitwiseFunctions.Add(new BitwiseCanonicalTestEntity {Byte1 = 15, Byte2 = 255}); // 00001111 &
        11111111

    _byte60And255Id = _bitwiseFunctions.Add(new BitwiseCanonicalTestEntity {Byte1 = 60, Byte2 = 255}); // 00111100 &
        11111111
}
```

Esimerkkikoodi 4.54. Bittioperaatioiden testidatan alustus

Bittioperaatioita testataan seuraavasti:

- AND (BitWiseAnd / &)

```
[Test]
public void Test_BitWiseAnd()
{
    Assert.AreEqual(0, _bitwiseFunctions.BitWiseAnd<byte>(_byte0And15Id)); // 00000000 & 00001111 => 00000000
    Assert.AreEqual(0, _bitwiseFunctions.BitWiseAnd<byte>(_byte0And60Id)); // 00000000 & 00111100 => 00000000
    Assert.AreEqual(0, _bitwiseFunctions.BitWiseAnd<byte>(_byte0And255Id)); // 00000000 & 11111111 => 00000000
    Assert.AreEqual(12, _bitwiseFunctions.BitWiseAnd<byte>(_byte15And60Id)); // 00001111 & 00111100 => 00001100
    Assert.AreEqual(15, _bitwiseFunctions.BitWiseAnd<byte>(_byte15And255Id)); // 00001111 & 11111111 => 00001111
    Assert.AreEqual(60, _bitwiseFunctions.BitWiseAnd<byte>(_byte60And255Id)); // 00111100 & 11111111 => 00111100
}
```

Esimerkkikoodi 4.55. Testimetodi AND-funktiolle

- NOT (BitWiseNot / $\sim$)

```
[Test]
public void Test_BitWiseNot()
{
    Assert.AreEqual(255, _bitwiseFunctions.BitWiseNot<short>(_byte0And15Id)); // 00000000 => 11111111
    Assert.AreEqual(240, _bitwiseFunctions.BitWiseNot<byte>(_byte15And60Id)); // 00001111 => 11110000
    Assert.AreEqual(195, _bitwiseFunctions.BitWiseNot<byte>(_byte60And255Id)); // 00111100 => 11000011
}
```

Esimerkkikoodi 4.56. Testimetodi NOT-funktiolle

- OR (BitWiseOr / |)

```
[Test]
public void Test_BitWiseOr()
{
    Assert.AreEqual(15, _bitwiseFunctions.BitWiseOr<byte>(_byte0And15Id)); // 00000000 & 00001111 => 00001111
    Assert.AreEqual(60, _bitwiseFunctions.BitWiseOr<byte>(_byte0And60Id)); // 00000000 & 00111100 => 00111100
    Assert.AreEqual(255, _bitwiseFunctions.BitWiseOr<byte>(_byte0And255Id)); // 00000000 & 11111111 => 11111111
    Assert.AreEqual(63, _bitwiseFunctions.BitWiseOr<byte>(_byte15And60Id)); // 00001111 & 00111100 => 00111111
    Assert.AreEqual(255, _bitwiseFunctions.BitWiseOr<byte>(_byte15And255Id)); // 00001111 & 11111111 => 11111111
    Assert.AreEqual(255, _bitwiseFunctions.BitWiseOr<byte>(_byte60And255Id)); // 00111100 & 11111111 => 11111111
}
```

Esimerkkikoodi 4.57. Testimetodi OR-funktiolle

- XOR (BitWiseXor / ^)

```
[Test]
public void Test_BitWiseXor()
{
    Assert.AreEqual(15, _bitwiseFunctions.BitWiseXor<byte>(_byte0And15Id)); // 00000000 & 00001111 => 00001111
    Assert.AreEqual(60, _bitwiseFunctions.BitWiseXor<byte>(_byte0And60Id)); // 00000000 & 00111100 => 00111100
    Assert.AreEqual(255, _bitwiseFunctions.BitWiseXor<byte>(_byte0And255Id)); // 00000000 & 11111111 => 11111111
    Assert.AreEqual(51, _bitwiseFunctions.BitWiseXor<byte>(_byte15And60Id)); // 00001111 & 00111100 => 00110011
    Assert.AreEqual(240, _bitwiseFunctions.BitWiseXor<byte>(_byte15And255Id)); // 00001111 & 11111111 => 11110000
    Assert.AreEqual(195, _bitwiseFunctions.BitWiseXor<byte>(_byte60And255Id)); // 00111100 & 11111111 => 11000011
}
```

Esimerkkikoodi 4.58. Testimetodi XOR-funktiolle

4.4.6 Spatiaaliset funktiot

Entity Framework 6 tukee sekä spatiaalisen (esim. maantieteellisen ja kartografisen) tiedon tallentamista että spatiaalisten funktioiden käyttöä .NET framework:n versiosta 4 alkaen (“Provider Support for Spatial Types” 2016).

Testin aikana tallennetaan spatiaalista tietoa tietotyyppejä DBGEOMETRY ja DBGEOGRAPHY käyttäen. Näistä DBGEOMETRY kuvaa geometrista kuviota (esim. piste, viiva tai monikulmio) ja DBGEOGRAPHY kuvaa muotoja tai kuviota geodeettisessä koordinaatiojärjestelmässä (Mueller 2013, s. 366). Spatiaalisia funktioita testattaessa tietokantaan lisätään ensin spatiaalista tietoa sisältäviä entiteettejä, jotka määritellään Well-Known Text -representaatiolla (“OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture” 2011). Entiteettien lisäämisen jälkeen tietokantaa tukevaa tuottajaa vasten suoritetaan kyselyitä spatiaalisia funktioita hyödyntäen. Jokaisen testattavan spatiaalisen funktion yhteydessä mainitaan, kuinka sitä tullaan testaamaan. Tämän tutkimuksen osalta testeissä käytetään seuraavia muotoja:

- POINT (piste)
- LINESTRING (viiva kahden pisteen välillä)
- POLYGON (monikulmio)

Testejä varten tietokantaan lisätään listauksessa 4.59 mainitut kuviot, joita vasten spatiaalisia funktioita testataan:

- DbGeometry - Piste (1 1)
- DbGeometry - Ympyrä (keskipiste (3 2), säde 2)
- DbGeometry - Neliö ((0 0, 0 10, 10 10, 10 0, 0 0))
- DbGeometry - Viiva (0 0, 0 2)
- DbGeometry - Viiva (0 0, 1 2)
- DbGeography - Piste (25.736675, 62.232186, 79)

```
[SetUp]
public void Setup()
{
    _spatialGeometryFunctions = new SpatialGeometryFunctions();
    _spatialGeographyFunctions = new SpatialGeographyFunctions();

    _spatialGeometryFunctions.DeleteAll();
    _spatialGeographyFunctions.DeleteAll();

    _idPointOneOne = _spatialGeometryFunctions.Add(new DbGeometryTestEntity { Value = DbGeometry.FromText("POINT(1 1)")
        });
    _idCircle = _spatialGeometryFunctions.Add(new DbGeometryTestEntity { Value = DbGeometry.FromText("POINT(3
        2)").Buffer(2) });
    _idSquare = _spatialGeometryFunctions.Add(new DbGeometryTestEntity { Value = DbGeometry.FromText("POLYGON((0 0, 0 10,
        10 10, 10 0, 0 0))") });
    _idLine = _spatialGeometryFunctions.Add(new DbGeometryTestEntity { Value = DbGeometry.FromText("LINESTRING(0 0, 0
        2)") });
    _idLine2 = _spatialGeometryFunctions.Add(new DbGeometryTestEntity { Value = DbGeometry.FromText("LINESTRING(0 0, 1
        2)") });

    _idAgora = _spatialGeographyFunctions.Add(new DbGeographyTestEntity { Value =
        DbGeography.FromText($"POINT({AgoraLongitude.ToString(NumberFormatInfo.InvariantInfo)}
        {AgoraLatitude.ToString(NumberFormatInfo.InvariantInfo)} {AgoraElevation})", 4326) });
}
```

Esimerkkikoodi 4.59. Spatiaalisten entiteettien lisääminen tietokantaan ennen testien ajamista

Testidatan alustuksessa viitataan muuttujiin *AgoraLatitude*, *AgoraLongitude* ja *AgoraElevation*, jotka alustettiin esimerkkikoodin 4.60 mukaisesti.

```
private const decimal AgoraLatitude = 25.736675m;
private const decimal AgoraLongitude = 62.232186m;
private const decimal AgoraElevation = 79m;
```

Esimerkkikoodi 4.60. Agoran sijainnin alustaminen testiluokassa

Seuraavia spatiaalisia funktioita testataan tekemällä kyselyitä tietokantaan spatiaalisia tietotyyppejä ominaisuuksinaan sisältäviä entiteettejä vasten:

- Distance (etäisyys annetusta pisteestä)

```
[Test]
public void Test_Distance()
{
    Assert.AreEqual(1, _spatialGeometryFunctions.Distance(_idPointOneOne, DbGeometry.FromText("POINT(1 2)")));
    Assert.AreEqual(0, _spatialGeometryFunctions.Distance(_idLine, DbGeometry.FromText("POINT(0 1)")));
    Assert.AreEqual(1, _spatialGeometryFunctions.Distance(_idLine, DbGeometry.FromText("POINT(1 1)")));
    Assert.AreEqual(0, _spatialGeometryFunctions.Distance(_idSquare, DbGeometry.FromText("POINT(2 2)")));
    Assert.Less(0, _spatialGeometryFunctions.Distance(_idCircle, DbGeometry.FromText("POINT(1 2)")));
    Assert.AreEqual(1, _spatialGeometryFunctions.Distance(_idCircle, DbGeometry.FromText("POINT(0 2)")));
}
```

Esimerkkikoodi 4.61. Testimetodi Distance-funktiolle

- Centroid (keskipiste)

```
[Test]
public void Test_Centroid()
{
    var expected = DbGeometry.FromText("POINT(5 5)");
    Assert.AreEqual(expected, _spatialGeometryFunctions.Centroid(_idSquare));
    expected = DbGeometry.FromText("POINT(3 2)");
    Assert.AreEqual(expected, _spatialGeometryFunctions.Centroid(_idCircle));
}
```

Esimerkkikoodi 4.62. Testimetodi Centroid-funktiolle

- Area (alue)

```
[Test]
public void Test_Area()
{
    // Area of a circle: a = pi * r^2
    var expected = Math.PI * Math.Pow(2, 2);
    var area = _spatialGeometryFunctions.Area(_idCircle);
    Assert.IsTrue(area.HasValue);
    Assert.IsTrue(Math.Abs(expected - area.Value) < 0.01); // Close enough

    Assert.AreEqual(100, _spatialGeometryFunctions.Area(_idSquare));
}
```

Esimerkkikoodi 4.63. Testimetodi Area-funktiolle

- Elevation (korkeus merenpinnasta)

```
[Test]
public void Test_Elevation()
{
    Assert.AreEqual(AgoraElevation, _spatialGeographyFunctions.Elevation(_idAgora));
}
```

Esimerkkikoodi 4.64. Testimetodi Elevation-funktiolle

- Latitude (leveyspiiri)

```
[Test]
public void Test_Latitude()
{
    Assert.AreEqual(AgoraLatitude, _spatialGeographyFunctions.Latitude(_idAgora));
}
```

Esimerkkikoodi 4.65. Testimetodi Latitude-funktiolle

- Longitude (pituuspiiri)

```
[Test]
public void Test_Longitude()
{
    Assert.AreEqual(AgoraLongitude, _spatialGeographyFunctions.Longitude(_idAgora));
}
```

Esimerkkikoodi 4.66. Testimetodi Longitude-funktiolle

- SpatialContains (sisältyvyys)

```
[Test]
public void Test_Contains()
{
    Assert.IsTrue(_spatialGeometryFunctions.Contains(_idSquare, DbGeometry.FromText("POINT(0.001 0.001)")));
    Assert.IsTrue(_spatialGeometryFunctions.Contains(_idSquare, DbGeometry.FromText("POINT(5 5)")));
    Assert.IsTrue(_spatialGeometryFunctions.Contains(_idLine, DbGeometry.FromText("POINT(0 1)")));
    Assert.IsTrue(_spatialGeometryFunctions.Contains(_idCircle, DbGeometry.FromText("POINT(3 2)")));
    Assert.IsTrue(_spatialGeometryFunctions.Contains(_idCircle, DbGeometry.FromText("POINT(3 0.001)")));
    Assert.IsFalse(_spatialGeometryFunctions.Contains(_idCircle, DbGeometry.FromText("POINT(3 0)")));
    Assert.IsFalse(_spatialGeometryFunctions.Contains(_idSquare, DbGeometry.FromText("POINT(0 0)")));
}
```

Esimerkkikoodi 4.67. Testimetodi SpatialContains-funktiolle

- SpatialCrosses (risteävyys)

```
[Test]
public void Test_Crosses()
{
    Assert.IsTrue(_spatialGeometryFunctions.Crosses(_idLine, DbGeometry.FromText("LINESTRING(-1 1, 1 1)")));
    Assert.IsFalse(_spatialGeometryFunctions.Crosses(_idLine, DbGeometry.FromText("LINESTRING(-2 0, 2 0)")));
    Assert.IsTrue(_spatialGeometryFunctions.Crosses(_idSquare, DbGeometry.FromText("LINESTRING(-1 1, 11 1)")));
    Assert.IsFalse(_spatialGeometryFunctions.Crosses(_idSquare, DbGeometry.FromText("LINESTRING(-2 0, 12 0)")));
    Assert.IsFalse(_spatialGeometryFunctions.Crosses(_idSquare, DbGeometry.FromText("LINESTRING(1 1, 4 4)")));
}
```

Esimerkkikoodi 4.68. Testimetodi SpatialCrosses-funktiolle

- SpatialDimension (ulottuvuus)

```
[Test]
public void Test_Dimension()
{
    Assert.AreEqual(0, _spatialGeometryFunctions.Dimension(_idPointOneOne));
    Assert.AreEqual(1, _spatialGeometryFunctions.Dimension(_idLine));
    Assert.AreEqual(2, _spatialGeometryFunctions.Dimension(_idSquare));
    Assert.AreEqual(2, _spatialGeometryFunctions.Dimension(_idCircle));
}
```

Esimerkkikoodi 4.69. Testimetodi SpatialDimension-funktiolle

- SpatialEnvelope (kuviot sisältävän laatikon laskenta)

```
[Test]
public void Test_Envelope()
{
    Assert.AreEqual(DbGeometry.FromText("POLYGON((1 0, 5 0, 5 4, 1 4, 1 0))"),
          _spatialGeometryFunctions.Envelope(_idCircle));
    Assert.AreEqual(DbGeometry.FromText("POLYGON((0 0, 1 0, 1 2, 0 2, 0 0))"),
          _spatialGeometryFunctions.Envelope(_idLine2));
}
```

Esimerkkikoodi 4.70. Testimetodi SpatialEnvelope-funktiolle

- SpatialIntersects (leikkaavuus)

```
[Test]
public void Test_Intersects()
{
    Assert.IsTrue(_spatialGeometryFunctions.Intersects(_idLine, DbGeometry.FromText("LINESTRING(-1 1, 1 1)")));
    Assert.IsFalse(_spatialGeometryFunctions.Intersects(_idLine, DbGeometry.FromText("LINESTRING(-1 3, 1 3)")));
    Assert.IsTrue(_spatialGeometryFunctions.Intersects(_idSquare, DbGeometry.FromText("LINESTRING(-1 1, 1 1)")));
}
```

Esimerkkikoodi 4.71. Testimetodi SpatialIntersects-funktiolle

- SpatialOverlaps (päällekkäisyys)

```
[Test]
public void Test_Overlaps()
{
    Assert.IsTrue(_spatialGeometryFunctions.Overlaps(_idLine, DbGeometry.FromText("LINESTRING(0 1, 0 3)")));
    Assert.IsFalse(_spatialGeometryFunctions.Overlaps(_idLine, DbGeometry.FromText("LINESTRING(0 1, 0 2)")));

    Assert.IsTrue(_spatialGeometryFunctions.Overlaps(_idSquare, DbGeometry.FromText("POLYGON((1 1, 1 11, 11 11,
          11 1, 1 1))")));
    Assert.IsFalse(_spatialGeometryFunctions.Overlaps(_idSquare, DbGeometry.FromText("POLYGON((0 0, 0 5, 5 5, 5
          0, 0 0))")));
    Assert.IsTrue(_spatialGeometryFunctions.Overlaps(_idSquare, DbGeometry.FromText("POINT(1 1)").Buffer(3)));

    Assert.IsTrue(_spatialGeometryFunctions.Overlaps(_idCircle, DbGeometry.FromText("POINT(4 2)").Buffer(2)));
    Assert.IsFalse(_spatialGeometryFunctions.Overlaps(_idCircle, DbGeometry.FromText("POINT(3 2)").Buffer(1)));
}
```

Esimerkkikoodi 4.72. Testimetodi SpatialOverlaps-funktiolle

- SpatialTouches (koskettavuus)

```
[Test]
public void Test_Touches()
{
    Assert.IsTrue(_spatialGeometryFunctions.Touches(_idSquare, DbGeometry.FromText("LINESTRING(-1 1, 1 -1)")));
    Assert.IsTrue(_spatialGeometryFunctions.Touches(_idCircle, DbGeometry.FromText("LINESTRING(1 0, 4 0)")));
    Assert.IsTrue(_spatialGeometryFunctions.Touches(_idLine, DbGeometry.FromText("LINESTRING(0 1, 3 1)")));
}
```

Esimerkkikoodi 4.73. Testimetodi SpatialTouches-funktiolle

- SpatialUnion (yhdistäminen)

```
[Test]
public void Test_Union()
{
    Assert.AreEqual(DbGeometry.FromText("LINESTRING(4 2, 0 2, 0 0)"), _spatialGeometryFunctions.Union(_idLine,
        DbGeometry.FromText("LINESTRING(0 2, 4 2)")));
    Assert.AreEqual(DbGeometry.FromText("POLYGON((0 0, 10 0, 10 5, 10 10, 10 15, 0 15, 0 10, 0 5, 0 0))"),
        _spatialGeometryFunctions.Union(_idSquare, DbGeometry.FromText("POLYGON((0 5, 0 15, 10 15, 10 5, 0
        5))")));
}
```

Esimerkkikoodi 4.74. Testimetodi SpatialUnion-funktiolle

- SpatialWithin (sisälläolevuus)

```
[Test]
public void Test_Within()
{
    Assert.AreEqual(true, _spatialGeometryFunctions.Within(_idSquare, DbGeometry.FromText("POINT(3 3)")));
    Assert.AreEqual(true, _spatialGeometryFunctions.Within(_idSquare, DbGeometry.FromText("LINESTRING(1 1, 9
        9)")));
    Assert.AreEqual(true, _spatialGeometryFunctions.Within(_idSquare, DbGeometry.FromText("POLYGON((1 1, 9 1, 9
        9, 1 9, 1 1))")));
    Assert.AreEqual(true, _spatialGeometryFunctions.Within(_idSquare, DbGeometry.FromText("POINT(4
        4)").Buffer(2)));

    Assert.AreEqual(false, _spatialGeometryFunctions.Within(_idSquare, DbGeometry.FromText("POINT(11 11)")));
    Assert.AreEqual(false, _spatialGeometryFunctions.Within(_idSquare, DbGeometry.FromText("LINESTRING(9 9, 11
        11)")));
    Assert.AreEqual(false, _spatialGeometryFunctions.Within(_idSquare, DbGeometry.FromText("POLYGON((9 9, 11 9,
        11 11, 9 11, 9 9))")));
    Assert.AreEqual(false, _spatialGeometryFunctions.Within(_idSquare, DbGeometry.FromText("POINT(1
        1)").Buffer(2)));
}
```

Esimerkkikoodi 4.75. Testimetodi SpatialWithin-funktiolle

5 Testiympäristö

Tässä luvussa esitellään testien ajossa käytettävä ympäristö (ajoalusta + käyttöjärjestelmä), tietokantojen versiot, tietokantakohtaisten tuottajien versiot sekä käytetyn Entity Frameworkin versio.

5.1 Testeissä käytettävien tuottajien versiot

Testeissä käytetään tietokantakohtaisista tuottajista viimeisintä testiajankohtana julkaistua versiota, jotka ovat listattuna:

- Microsoftin MS SQL -tuottaja versio 6.4.0
- Microsoftin MS SQL CE -tuottaja versio 6.4.0
- Devartin dotConnect for Oracle (Oracle) 9.13.1127 professional trial
- Oraclen ODP.NET -tuottaja Oracle.ManagedDataAccess.EntityFramework 6.122.19.1
- MySQL connector/NET (MySQL) 8.0.19 MySql.Data.Entity.EF6 version 6.10.9.0
- Devartin dotConnect for MySQL (MySQL) 8.18. professional trial
- CData MySQL ADO.NET provider (MySQL) versio 19.0.7354
- CData MongoDB ADO.NET provider versio 19.0.7354
- Devartin dotConnect for PostgreSQL (PostgreSQL) 7.17.1666 professional trial
- Npgsql (PostgreSQL) EntityFramework6.Npgsql 6.4.1.0

5.2 Testeissä käytettävät tietokannat

Lähtökohtaisesti kustakin tietokannasta valitaan testattavaksi joko viimeisin saatavilla oleva versio, tai muu tuottajan julkaisijan suosittelema versio. Versiot, joilla nämä testit suoritetaan:

- Microsoft SQL server 2016
- SQL Server Compact 4.0 (4.0.8876.1)
- Oracle - Oracle Database Express Edition (XE) Release 18.4.0.0.0 (18c)
- MySQL - MySQL Server 8.0.19

- MongoDB - MongoDB Community Server 4.2.5
- PostgreSQL 10.12

5.3 Ajoalusta

Kaikki testit suoritetaan samalla kannettavalla tietokoneella, jonka tarkemmat tiedot ovat ohessa:

- Prosessori (CPU) - i7-6820HQ @ 2.70GHz
- Keskusmuisti (RAM) - 64 GB
- Käyttöjärjestelmä (OS) - Windows 10 Enterprise
- Massamuisti (OS + tietokannat) - Samsung NVMe 512GB M.2 SSD

5.4 Entity Framework

Testien ajossa käytetään Entity Frameworkin versiota 6.4.0, joka oli viimeisin saatavilla oleva versio testien suorituksen alkaessa.

6 Testien tulokset

Tässä luvussa käydään läpi sekä ongelmat, joihin törmättiin testien ajossa, että ajettujen testien tulokset.

Jokaisesta testattavasta kokonaisuudesta koostetaan testitulokset tuottajittain taulukkoon. Tulostaulukoissa käytetään seuraavaa merkintätapaa:

- X -> Tuottaja tukee kyseistä ominaisuutta (testi meni läpi ilman ongelmia)
- * -> Tuottaja tukee kyseistä ominaisuutta osittain (nämä tulokset käydään läpi tapauskohtaisesti)
- - -> Tuottaja ei tue kyseistä ominaisuutta

Ongelmatilanteet testien ajon aikana käsitellään tapauskohtaisesti. Jos jokin testattavista tuottajista ei tue jonkin tietyn tietotyypin käyttämistä tai jotain tiettyä funktiota, ei-tuettu tietotyyppi tai funktio poistetaan tai muokataan sopivaksi testin ajon ajaksi ja tästä raportoidaan tuloksissa.

6.1 Testeistä hylätyt tuottajat

Osaa tuottajista ei pystytty onnistuneesti testaamaan, koska tuottajat eivät osanneet luoda tarvitsemaansa tietokantaa. Devartin tuottajat yrittivät avata nimettyä tietokantaa, jota ei vielä ollut olemassa. CDatan tuottajat ilmoittivat suoraan, ettei toimintoa CreateDatabase tueta. Seuraavat tuottajat jouduttiin tästä syystä jättämään pois testeistä:

- Devartin dotConnect for MySQL
  - Tuottajaa yritettiin käyttää MySQL:n tukemalla connectionstringillä, mutta tuottaja heittää poikkeuksen: *"Devart.Data.MySql.MySqlException : Unknown database 'devartmysql'"*.
- CData MySQL ADO.NET provider
  - Tuottajaa yritettiin käyttää MySQL:n tukemalla connectionstringillä, mutta tuottaja heittää poikkeuksen: *"CreateDatabase is not supported by the provider."*

- CData MongoDB ADO.NET provider
  - Tuottajaa yritettiin käyttää MongoDB:n tukemalla connectionstringillä, mutta tuottaja heittää poikkeuksen: *"CreateDatabase is not supported by the provider."*
- Devartin dotConnect for PostgreSQL
  - Tuottajaa yritettiin käyttää PostgreSQL:n tukemalla connectionstringillä, mutta tuottaja heittää poikkeuksen: *"Devart.Data.PostgreSql.PgSqlException: database "devartPostgre"does not exist"*.

6.2 Testeistä hyväksytysti suoriutuneet tuottajat

Testit ajettiin onnistuneesti seuraaville tuottajille, joista käytetään annettuja merkintöjä tulostaulukoissa:

- MS SQL -> Microsoftin MS SQL -tuottaja
- MS CE -> Microsoftin MS SQL CE -tuottaja
- D Oracle -> Devartin dotConnect for Oracle
- Oracle -> Oraclen ODP.NET -tuottaja
- MySQL -> MySQL connector/NET
- PostgreSQL -> Npgsql (PostgreSQL)

Oraclen tietokannoille suunnatuille tuottajille ajettiin testit omia instanssejaan vasten (Oraclen ODP.NET -tuottaja ja Devartin dotConnect for Oracle -tuottaja), koska käsittääkseni saman instanssin sisälle ei voi luoda useampaa tietokantaa. Näiden tuottajien osalta tuottajan tietokantayhteydet osoittivat Oracle-instanssiin ilman tietokannan nimeä, käyttäen vain määritystä tietolähteelle (Data Source=localhost).

6.3 Tuki olioiden perustietotyypeille

Taulukossa 2 esitetään tuottajien tuki erilaisille tallennettaville perustietotyypeille.

Tuki perustietotyypeille						
Funktio	MS SQL	MS CE	D Oracle	Oracle	MySQL	PostgreSQL
Boolean	X	X	X	X	X	X
Byte	X	X	X	X	X	X
Byte[]	X	X	X	X	X	X
DateTime	X	*	*	*	*	X
DateTimeOffset	X	-	X	X	-	X
Decimal	*	*	*	*	*	*
Double	X	X	X	*	X	X
Float	X	X	X	*	*	X
Guid	X	X	X	X	X	X
Int16	X	X	X	X	X	X
Int32	X	X	X	X	X	X
Int64	X	X	X	X	X	X
SByte	-	-	X	-	X	-
String	X	X	*	X	X	X
Time	*	-	X	-	*	X

Taulukko 2: Tuottajien tuki perustietotyypeille

6.3.1 Microsoftin MS SQL -tuottaja

Microsoftin MS SQL -tuottaja ei tukenut seuraavia perustietotyyppejä:

- Etumerkillinen tavu (SByte)

Muiden tietotyyppien osalta testit menivät Microsoftin MS SQL -tuottajalla läpi lukuunottamatta seuraavia poikkeuksia:

- Aika (Timespan) toimi testeissä oikein aikavälillä 0:00:00 - 23:59:59. Negatiivisia arvoja ei tuettu, vaan ne tallentuivat kantaan arvona 0:00:00.

- Desimaaliluvun (Decimal) käsittely toimi testeissä oikein muutoin, mutta pienimmän ja suurimman mahdollisen arvon (*Decimal.MinValue* ja *Decimal.MaxValue*) kantaan tallentaminen epäonnistui aiheuttaen poikkeuksen: *"System.OverflowException : Conversion overflows."*

Tietokanta määriteltiin tallentamaan ajanhetken kuvaus (DateTime) tietokantaan datetime2-tyyppisenä.

6.3.2 Microsoftin MS SQL CE -tuottaja

Microsoftin MS SQL CE -tuottaja ei tukenut seuraavia perustietotyyppejä:

- Etumerkillinen tavu (SByte)
- Ajanhetken kuvaus UTC-aikaan verrannollisena (DateTimeOffset)
- Aika (Timespan)

Muiden tietotyyppien osalta testit menivät Microsoftin MS SQL CE -tuottajalla läpi lukuunottamatta seuraavia poikkeuksia:

- Ajanhetken kuvaus (DateTime) toimi testeissä oikein muutoin, mutta pienimmän ja suurimman mahdollisen ajanhetken (*DateTime.MinValue* ja *DateTime.MaxValue*) kantaan tallentaminen epäonnistui aiheuttaen poikkeuksen: *"System.Data.SqlServerCe.SqlCeException : An overflow occurred while converting to datetime."*
- Desimaaliluvun (Decimal) käsittely toimi testeissä oikein muutoin, mutta pienimmän ja suurimman mahdollisen arvon (*Decimal.MinValue* ja *Decimal.MaxValue*) kantaan tallentaminen epäonnistui aiheuttaen poikkeuksen: *"System.Data.SqlServerCe.SqlCeException : Data conversion failed. [OLE DB status value (if known) = 0]"*.

6.3.3 Devartin dotConnect for Oracle -tuottaja

Devartin dotConnect for Oracle -tuottaja tuki kaikkia perustietotyyppejä. Testien aikana törmättiin seuraaviin ongelmiin:

- Ajanhetken kuvaus (DateTime) toimi oikein, ja tarkkuuden sai konfiguroitua milli-

sekunnin tarkkuudelle asti. Kuitenkin millisekunnin tarkkuuden konfiguroinnin jälkeen, tietokantaan tallennettua suurinta mahdollista arvoa *DateTime.MaxValue* ei enää voinut lukea kannasta, vaan yrittäessä tiedon lukua kannasta, saatiin poikkeus: *"System.ArgumentOutOfRangeException : Year, Month, and Day parameters describe an un-representable DateTime."*

- Merkkijonon (string) käsittely toimii testeissä muuten oikein, mutta tyhjä merkkijono palautuu tietokannasta olemattomana (null).
- Desimaaliluvun (Decimal) käsittely toimi testeissä oikein muutoin, mutta pienimmän ja suurimman mahdollisen arvon (*Decimal.MinValue* ja *Decimal.MaxValue*) kantaan tallentaminen epäonnistui aiheuttaen poikkeuksen: *"Devart.Data.Oracle.OracleException : ORA-01438: value larger than specified precision allowed for this column"*.

6.3.4 Oraclen ODP.NET -tuottaja

Oraclen ODP.NET -tuottaja ei tukenut seuraavia perustietotyyppejä:

- Etumerkillinen tavu (SByte)
- Aika (Timespan)

Muiden tietotyyppien osalta testit menivät Oraclen ODP.NET -tuottajalla läpi lukuunottamatta seuraavia poikkeuksia:

- Ajanhetken kuvauksen (DateTime) käsittelyssä törmättiin samoihin ongelmiin kuin Devartin Oraclen toimittajan kanssa. Konfiguroimalla saatiin lisätarkkuutta (millisekunnin tasolle), mutta tämän jälkeen suurimman mahdollisimman arvon tallentaminen kantaan ja lukeminen sieltä ei enää onnistunut.
- Liukuluku (Double) toimii testeissä muuten oikein, mutta minimin (*Double.MinValue*) ja maksimin (*Double.MaxValue*) tallennus aiheuttaa poikkeuksen: *"System.OverflowException : Arithmetic operation resulted in an overflow."*.
- Liukuluku (float) toimii testeissä muuten oikein, mutta tallennettu arvo on tarkka vain seitsemään merkitsevään numeroon asti. Tämä kävi ilmi kun testissä tallennettiin ja noudettiin kannasta vertailua varten minimi- (*Float.MinValue*) ja maksimi-arvot (*Float.MaxValue*).

- Desimaaliluvun (Decimal) käsittely toimi testeissä oikein muutoin, mutta pienimmän ja suurimman mahdollisen arvon (*Decimal.MinValue* ja *Decimal.MaxValue*) kantaan tallentaminen epäonnistui aiheuttaen poikkeuksen: *"Oracle.ManagedDataAccess.Types.OracleTruncateException : ORA-16550: truncated result"*.

6.3.5 MySQL connector/NET -tuottaja

MySQL connector/NET -tuottaja ei tukenut seuraavaa perustietotyyppiä:

- Ajanhetken kuvaus UTC-aikaan verrannollisena (DateTimeOffset)

Muiden tietotyyppien osalta testit menivät MySQL connector/NET -tuottajalla läpi lukuunottamatta seuraavia poikkeuksia:

- Ajanhetken kuvaus (DateTime) toimi oikein suurinta mahdollista arvoa (DateTime.MaxValue) lukuunottamatta. Suurin mahdollinen ajanhetki tallentuu kantaan väärin (aikana 0001-01-01T00:00:00).
- Liukuluku (float) toimii testeissä muuten oikein, mutta tallennettu arvo on tarkka vain kuuteen desimaaliin asti. Tämä kävi ilmi kun testissä tallennettiin ja noudettiin kannassa vertailua varten minimi- (*Float.MinValue*) ja maksimi-arvot (*Float.MaxValue*).
- Aika (Timespan) tallentuu suurimmassa osassa tapauksista väärin kantaan. Aika 0:00:00 tallentuu kantaan oikein, mutta sekä -23:59:59 että 23:59:59 tallentuvat kantaan väärin (arvoina -1:00:00 ja 1:00:00).
- Desimaaliluvun (Decimal) käsittely toimi testeissä oikein muutoin, mutta pienimmän ja suurimman mahdollisen arvon (*Decimal.MinValue* ja *Decimal.MaxValue*) kantaan tallentaminen epäonnistui aiheuttaen poikkeuksen, jossa tuottaja ilmoitti tuetuksi arvoalueeksi -9999999999999999.99m ja 9999999999999999.99m välisen alueen.

6.3.6 PostgreSQL-tuottaja

Npgsql-tuottaja ei tukenut seuraavaa perustietotyyppiä:

- Etumerkillinen tavu (SByte)

Muiden tietotyyppien osalta testit menivät Npgsql-tuottajalla läpi lukuunottamatta seuraavaa poikkeusta:

- Desimaaliluvun (Decimal) käsittely toimi testeissä oikein muutoin, mutta pienimmän ja suurimman mahdollisen arvon (*Decimal.MinValue* ja *Decimal.MaxValue*) kantaan tallentaminen epäonnistui aiheuttaen poikkeuksen: *"Npgsql.PostgresException : 22003: numeric field overflow"*.

6.3.7 Yhteenveto tuesta perustietotyypeille

Tuottajilla oli yllättävän paljon ongelmia jo perustietotyyppien tallentamisessa ja lukemisessa. Erityisesti ääriarvot ja liukulukujen tarkuus aiheuttivat ongelmia. Desimaalilukujen ääriarvojen tallentaminen ei onnistunut yhdeltäkään tuottajalta. Desimaaliluvun tallentavaa kenttää ei konfiguroitu yhdelläkään tuottajalla, vaan kaikki menivät tuottajan oletusasetuksilla. Vaarallisinta tuloksissa on mielestäni se, että osa tuottajista ei heitä virhetapauksissa poikkeusta, vaan tallentaa kantaan väärän arvon. Tämä voi aiheuttaa yllättäviä ongelmia, koska tallentamisen voisi olettaa onnistuvan ongelmitta.

Oraclen tietokantaa vasten tehdyillä tuottajilla oli identtisiä ongelmia ajanhetken kuvauksen (DateTime) tallentamisen kanssa, mutta yllättäen tuottajissa oli myös eroja. Eroavista ongelmista huomataan, että kyseessä eivät voi olla tietokannan rajoitteet, vaan tuottajien toteutukset ovat vain erilaiset. Devartin tuottaja palautti tyhjän merkkijonon olemattomana (null), kun taas ODP.NET-tuottaja palautti sen oikein (tyhjänä merkkijonona). ODP.NET-tuottaja ei tukenut lainkaan tietotyyppejä *SByte* ja *TimeSpan*, vaikka Devartin tuottaja hallitsi ne ongelmitta. Myös liukulukujen tarkkuuden tallentamisessa oli eroja näiden tuottajien välillä.

Npgsql-tuottajalla oli tässä testikategoriassa lukumääräisesti mitattuna vähiten ongelmia. Ainoastaan *SByte* ei ollut tuettuna ja desimaaliluvun ääriarvot aiheuttivat ongelmia. Tämä oli yllättävä tulos, sillä olisi voinut kuvitella Microsoftin de facto -tuottajalla (MS SQL) olevan vähiten ongelmia.

6.4 Tuki koostefunktioille

Taulukossa 3 esitetään tuottajien tuki erilaisille koostefunktioille.

Tuki koostefunktioille						
Funktio	MS SQL	MS CE	D Oracle	Oracle	MySQL	PostgreSQL
Avg	X	X	X	*	X	X
BigCount	X	X	X	*	X	X
Count	X	X	X	*	X	X
Max	X	X	X	*	X	X
Min	X	X	X	*	X	X
StDev	X	-	X	X	-	X
StDevP	X	-	X	X	-	X
Sum	X	X	X	*	X	X
Var	X	-	X	X	-	X
VarP	X	-	X	X	-	X

Taulukko 3: Tuottajien tuki koostefunktioille

6.4.1 Microsoftin MS SQL -tuottaja

Microsoftin MS SQL -tuottajalla ajetut testit menivät läpi ja kaikki koostefunktiot olivat tuettuja.

6.4.2 Microsoftin MS SQL CE -tuottaja

MS SQL CE -tuottaja ei tukenut varianssiin ja keskihajontaan liittyviä funktioita. Muilta osin testit menivät läpi.

6.4.3 Devartin DotConnect for Oracle -tuottaja

Devartin Oracle -tuottajalla ajetut testit menivät läpi ja kaikki koostefunktiot olivat tuettuja.

6.4.4 Oraclen ODP.NET -tuottaja

Oraclen ODP.NET -tuottajalla ajetut testit menivät läpi varianssin ja keskihajonnan osalta. Muilta osin testit epäonnistuivat ja näyttäisikin siltä, että funktiot palauttavat kaikkeen vastauksena arvon 1. Tuottajan testeissä palauttamat arvot:

- Keskiarvo (Avg) = 1 (pitäisi olla 1.5)
- Lukumäärä (sekä *Count* että *BigCount*) = 1 (pitäisi olla 10)
- Maksimi (Max) = 1 (pitäisi olla 4)
- Minimi (Min) = 1 (pitäisi olla -0.5)
- Summa (Sum) = 1 (pitäisi olla 15)

6.4.5 MySQL connector/NET -tuottaja

MySQL:n tuottaja ei tukenut varianssiin ja keskihajontaan liittyviä funktioita. Muilta osin testit menivät läpi.

6.4.6 PostgreSQL-tuottaja

Npgsql-tuottajalla ajetut testit menivät läpi ja kaikki koostefunktiot olivat tuettuja.

6.4.7 Yhteenveto tuesta koostefunktioille

MS SQL -tuottaja, Devartin Oracle-tuottaja ja PostgreSQL:n Npgsql-tuottaja selvisivät kaikista funktioista ongelmitta. Toinen Oraclea tukeva tuottaja (ODP.NET) palautti lukeman kaikista funktiokutsuista, mutta paluuarvona tuli useammin vääriä kuin oikeita vastauksia. Oraclea tukevien tuottajien tuessa koostefunktioille oli siis selkeä ero. Varianssin ja keskihajonnan laskenta aiheutti kahdelle tuottajalle (MS SQL CE ja MySQL) ongelmia.

6.5 Tuki matemaattisille funktioille

Taulukossa 4 esitetään tuottajien tuki erilaisille matemaattisille funktioille.

Tuki matemaattisille funktioille						
Funktio	MS SQL	MS CE	D Oracle	Oracle	MySQL	PostgreSQL
Abs	X	X	X	X	X	X
Ceiling	X	X	X	X	X	X
Floor	X	X	X	X	X	X
Power	X	X	X	X	X	X
Round	X	X	X	X	X	X
Truncate	X	X	X	X	X	X

Taulukko 4: Tuottajien tuki matemaattisille funktioille

6.5.1 Microsoftin MS SQL -tuottaja

Microsoftin MS SQL -tuottajalla ajetut testit menivät läpi ja kaikki matemaattiset funktiot olivat tuettuja.

6.5.2 Microsoftin MS SQL CE -tuottaja

Microsoftin MS SQL CE -tuottajalla ajetut testit menivät läpi ja kaikki matemaattiset funktiot olivat tuettuja.

6.5.3 Devartin DotConnect for Oracle -tuottaja

Devartin Oracle-tuottajalla ajetut testit menivät läpi ja kaikki matemaattiset funktiot olivat tuettuja.

6.5.4 Oraclen ODP.NET -tuottaja

Oraclen ODP.NET -tuottajalla ajetut testit menivät läpi ja kaikki matemaattiset funktiot olivat tuettuja.

6.5.5 MySQL connector/NET -tuottaja

MySQL-tuottajalla ajetut testit menivät läpi ja kaikki matemaattiset funktiot olivat tuettuja.

6.5.6 PostgreSQL-tuottaja

Npgsql-tuottajalla ajetut testit menivät läpi ja kaikki matemaattiset funktiot olivat tuettuja.

6.5.7 Yhteenveto tuesta matemaattisille funktioille

Kaikki tuottajat tukivat kaikkia matemaattisia funktiota.

6.6 Tuki merkkijonofunktioille

Taulukossa 5 esitetään tuottajien tuki erilaisille spatiaalisille funktioille.

Tuki merkkijonofunktioille						
Funktio	MS SQL	MS CE	D Oracle	Oracle	MySQL	PostgreSQL
Concat	X	X	*	X	X	X
Contains	X	X	X	X	-	X
EndsWith	X	X	*	X	-	-
IndexOf	X	X	X	X	X	X
Left	X	X	*	*	X	X
Length	*	*	X	X	X	X
LTrim	X	X	*	X	X	X
Replace	X	X	X	X	X	X
Reverse	X	-	-	-	X	-
Right	X	X	*	*	X	X
RTrim	X	X	*	X	X	X
StartsWith	X	X	X	X	-	X
Substring	X	X	X	X	X	X
Trim	X	X	*	X	X	X
ToUpper	X	X	*	X	X	X
ToLower	X	X	*	X	X	X

Taulukko 5: Tuottajien tuki merkkijonofunktioille

6.6.1 Microsoftin MS SQL -tuottaja

Microsoftin MS SQL -tuottaja tukee kaikkia merkkijonofunktioita ja testit menivät läpi lukuun ottamatta funktiota *Length*, joka ei laskenut merkkijonon lopussa olevia välilyöntejä mukaan merkkijonon pituuteen.

6.6.2 Microsoftin MS SQL CE -tuottaja

Microsoftin MS SQL CE -tuottaja tukee *Reverse*-funktiota lukuun ottamatta kaikkia merkkijonofunktioita. Muiden funktioiden osalta testit menivät läpi lukuun ottamatta funktiota *Length*, joka ei laskenut merkkijonon lopussa olevia välilyöntejä mukaan merkkijonon pituuteen.

6.6.3 Devartin DotConnect for Oracle -tuottaja

Devartin DotConnect for Oracle -tuottaja tukee kaikkia merkkijonofunktioita, mutta seuraavien funktioiden kutsut heittivät poikkeuksen:

- *Reverse*-funktion kutsu heitti poikkeuksen: *"Devart.Data.Oracle.OracleException : ORA-00932: inconsistent datatypes: expected CHAR got NCLOB"*.
- *RTrim*-funktion kutsu heitti poikkeuksen: *"Devart.Data.Oracle.OracleException : ORA-22275: invalid LOB locator specified"*.

Yllä olevien lisäksi *EndsWith*-funktio ei tunnista merkkijonon lopussa olevia välilyöntejä. Muut taulukossa 5 mainitut funktioiden osittaiset ongelmat (*) johtuvat siitä, että tyhjä merkkijono palautetaan testitapauksissa kannasta olemattomana (null) kutsujalle.

6.6.4 Oraclen ODP.NET -tuottaja

Oraclen ODP.NET tuottaja palauttaa *Left*- ja *Right*-funktioista kysyttäessä nollan mittaista merkkijonoa olemattoman (null) tyhjän merkkijonon sijaan. Näiden lisäksi funktio *Reverse* heittää poikkeuksen: *"Oracle.ManagedDataAccess.Client.OracleException : ORA-00932: epäyhtenäiset tietotyypit: odotettiin CHAR, saatiin NCLOB"*.

6.6.5 MySQL connector/NET -tuottaja

MySQL-tuottaja ei tukenut funktioita *Contains*, *EndsWith* ja *StartsWith*. Muiden funktioiden osalta testit menivät läpi.

6.6.6 PostgreSQL-tuottaja

PostgreSQL-kannan Npgsql-tuottaja ei tukenut funktioita *EndsWith* ja *Reverse*. Muiden funktioiden osalta testit menivät läpi.

6.6.7 Yhteenveto tuesta merkkijonofunktioille

De facto -tuottajalla (MS SQL) oli lähes virheetön suoritus, testeissä tuli vastaan vain yksi puute yhden funktion (*Length*) testissä. Muidenkin tuottajien osalta tuet olivat hyvällä tasolla. Hieman yllättäen kaupallisella Devartin DotConnect for Oracle -tuottajalla oli useimman testin kanssa ongelmia. Suurin osa tämän tuottajan ongelmista toisaalta liittyivät samaan asiaan, eli tyhjän merkkijonon palauttamiseen olemattomana (null). Oraclen ODP.NET -tuottaja palautti myös *Left*- ja *Right*-funktioista tyhjän merkkijonon nullina, mutta muista funktioista joissa Devartin tuottajalla oli ongelmia, tyhjät merkkijonot palautuvat oikein. Kyseessä ei siis voi olla ainakaan niiltä osin kannan rajoite.

6.7 Tuki ajankäsittelyn funktioille

Taulukossa 6 esitetään tuottajien tuki erilaisille ajankäsittelyn funktioille.

Tuki ajankäsittelyn funktioille						
Funktio	MS SQL	MS CE	D Oracle	Oracle	MySQL	PostgreSQL
AddMilliseconds	X	*	X	X	-	X
AddSeconds	X	X	X	X	-	X
AddMinutes	X	X	X	X	-	X
AddHours	X	X	X	X	-	X
AddDays	X	X	X	X	-	X
AddMonths	X	X	X	X	-	X
AddYears	X	X	X	X	-	X
CurrentDateTime	X	*	*	*	*	X
CurrentUtcDateTim	X	-	*	*	-	X
Millisecond	X	X	X	X	-	X
Second	X	X	X	X	X	X
Minute	X	X	X	X	X	X
Hour	X	X	X	X	X	X
Day	X	X	X	X	X	X
DayOfYear	X	X	X	X	X	-
Month	X	X	X	X	X	X
Year	X	X	X	X	X	X

Taulukko 6: Tuottajien tuki ajankäsittelyn funktioille

6.7.1 Microsoftin MS SQL -tuottaja

Microsoftin MS SQL -tuottajalla ajetut testit menivät läpi ja kaikki ajankäsittelyn funktiot olivat tuettuja.

6.7.2 Microsoftin MS SQL CE -tuottaja

Microsoftin MS SQL CE -tuottaja ei tukenut funktiota *CurrentUtcDateTime*. Funktio *AddMilliseconds* ei palauttanut tietokannassa olevaa ajanhetkeä siirrettynä annetulla määrällä millisekunteja (-1), vaan palautti suoraan tietokannassa olevan ajanhetken. Myös *CurrentDateTime* palautti ajan joskus millisekunnin sisään kyselyhetkestä ja joskus ei. Syy tähän löytyy tallennusrakenteen käyttämästä tietotyypistä *datetime (Transact-SQL)*, joka ei kykene millisekunnin tarkkuuteen, vaan pyöristää sekunnit aina lähimpään inkrementtiin arvoista .000, .003, ja .007 ("datetime (Transact-SQL)" 2017). MS SQL -tuottajalle konfiguroitu tarkempi *datetime2* ei valitettavasti ole saatavilla MS SQL CE -kannalle.

6.7.3 Devartin DotConnect for Oracle -tuottaja

Devartin DotConnect for Oracle -tuottaja tuki kaikkia ajankäsittelyn funktioita, mutta seuraavien funktioiden tarkkuuden kanssa ei ylletty millisekunnin tasolle testeissä:

- CurrentDateTime: Tuottaja palautti tämänhetkisen ajan oikein, mutta sekunnin tarkkuudella
- CurrentUtcDateTime: Tuottaja palautti tämänhetkisen ajan oikein, mutta kyselyn palautumisessa kesti liian kauan, jotta tietokannasta saatu aika olisi ollut millisekunnin sisällä vertailuajasta. Ajanhetkien ero vaihteli useamman kerran testatessa 1,2 ja 9,6 millisekunnin välillä.

6.7.4 Oraclen ODP.NET -tuottaja

Oraclen ODP.NET -tuottaja tuki kaikkia ajankäsittelyn funktioita, mutta tätä ajanhetkeä kyselevien funktioiden (*CurrentDateTime* ja *CurrentUtcDateTime*) kanssa ei ylletty millisekunnin tarkkuuteen, johtuen kyselyn itsensä ajankestosta (vastaava tilanne kuin Devartin tuottajalla oli funktion *CurrentUtcDateTime* kanssa).

6.7.5 MySQL connector/NET -tuottaja

MySQL connector/NET -tuottaja ei tukenut seuraavia ajankäsittelyn funktioita: *AddMilliseconds*, *AddSeconds*, *AddMinutes*, *AddHours*, *AddDays*, *AddMonths*, *AddYears*, *CurrentUtcDateTime* ja *Millisecond*.

CurrentDateTime palautti tämänhetkisen ajan sekunnin tarkkuudella.

6.7.6 PostgreSQL-tuottaja

PostgreSQL-tuottaja ei tukenut funktiota *DayOfYear*, mutta muut funktiot olivat tuettuja ja testit menivät niiden osalta läpi.

6.7.7 Yhteenveto tuesta ajankäsittelyn funktioille

MS SQL -tuottaja selvisi näistä testeistä puhtain paperein ja PostgreSQL-tuottajakin vain yhdellä huomautuksella (funktiota *DayOfYear* ei tuettu). MS SQL CE ei päässyt millisekunnin tarkkuuteen, johtuen tietokannan rajoitteista.

Oraclen päällä toimivat tuottajat tukivat kaikkea, mutta muutamissa kohdin oli suorituskykyongelmia (muutaman millisekunnin viiveitä kyselyissä). Koska molemmilla Oraclen tuottajilla oli sama ongelma, voisi arvella kyseessä olevan tietokannan suorituskykyongelma. Normaalissa käytössä näin pienet viiveet eivät tosin aiheuta ongelmia, mutta niiden olemassa olo on hyvä tiedostaa.

6.8 Tuki bittioperaatioille

Taulukossa 7 esitetään tuottajien tuki erilaisille bittioperaatioille.

Tuki bittioperaatioille						
Funktio	MS SQL	MS CE	D Oracle	Oracle	MySQL	PostgreSQL
And	X	X	X	X	*	X
Or	X	X	X	-	*	X
Not	X	X	-	X	*	*
Xor	X	X	X	X	*	X

Taulukko 7: Tuottajien tuki bittioperaatioille

6.8.1 Microsoftin MS SQL -tuottaja

Microsoftin MS SQL -tuottajalla ajetut testit menivät läpi ja kaikki bittioperaatiot olivat tuettuja.

6.8.2 Microsoftin MS SQL CE -tuottaja

Microsoftin MS SQL CE -tuottajalla ajetut testit menivät läpi ja kaikki bittioperaatiot olivat tuettuja.

6.8.3 Devartin DotConnect for Oracle -tuottaja

Devartin DotConnect for Oracle -tuottajalla ajetut testit menivät läpi Not-operaatiota lukuunottamatta. Not-operaatio ilmeisesti ei ole tuettuna lainkaan, koska operaation kutsu aiheutti seuraavan poikkeuksen: *"ORA-00904: "BITWISENOT": invalid identifier"*.

6.8.4 Oraclen ODP.NET -tuottaja

Oraclen ODP.NET -tuottajalla ajetut testit menivät läpi Or-operaatiota lukuunottamatta. Or-operaation kutsu aiheutti seuraavan poikkeuksen: *"Value was either too large or too small for a UInt32"*.

6.8.5 MySQL connector/NET -tuottaja

Ajettavaan testikoodiin piti tehdä muutamia muutoksia MySQL connector/NET -tuottajan osalta, sillä se ei tukenut bittioperaatioiden palautusarvona tavuja (byte). Tätä tuottajaa käytettäessä sallittiin bittioperaatioiden paluuarvona Int32.

Int32-paluuarvoa käyttäen, MySQL connector/NET -tuottajalla ajetut testit menivät läpi Not-operaatiota lukuunottamatta. Tuottaja tuki myös Not-operaatiota, mutta palautti funktiota käytettäessä vääriä arvoja. Esimerkiksi tavulle *00000000* MySQL-tuottaja palautti NOT-operaatiolla arvon 1, ja muihin testitapauksiin vastaus oli 0, joten voisi olettaa MySQL:n tuottajan toimivan tässä binäärisesti (palauttaa nollalle arvon 1 ja kaikille muille arvon 0).

6.8.6 PostgreSQL-tuottaja

Npgsql-tuottajalla ajetut testit menivät läpi Not-operaatiota lukuunottamatta. Not-operaation kutsu aiheutti seuraavan poikkeuksen: *"System.OverflowException : Arithmetic operation resulted in an overflow."*

6.8.7 Yhteenveto tuesta bittioperaatioille

Microsoftin tuottajat selvisivät tästä testikategoriasta ongelmitta, muilla tuottajilla oli ongelmia. Erityisesti NOT-operaatio tuntui olevan haastava, koska se aiheutti ongelmia kolmelle tuottajalle (Devart Oracle, MySQL ja PostgreSQL).

6.9 Tuki spatiaalisille funktioille

Taulukossa 8 esitetään tuottajien tuki erilaisille spatiaalisille funktioille.

Tuki spatiaalisille funktioille						
Funktio	MS SQL	MS CE	D Oracle	Oracle	MySQL	PostgreSQL
Area	X	-	*	-	-	-
Centroid	*	-	X	-	-	-
Contains	X	-	*	-	-	-
Crosses	*	-	*	-	-	-
Dimension	X	-	X	-	-	-
Distance	*	-	X	-	-	-
Elevation	X	-	*	-	-	-
Envelope	*	-	X	-	-	-
Intersects	X	-	X	-	-	-
Latitude	X	-	X	-	-	-
Longitude	X	-	X	-	-	-
Overlaps	X	-	*	-	-	-
Touches	*	-	X	-	-	-
Union	X	-	X	-	-	-
Within	X	-	*	-	-	-

Taulukko 8: Tuottajien tuki spatiaalisille funktioille

6.9.1 Microsoftin MS SQL -tuottaja

Microsoftin MS SQL -tuottaja tuki kaikkia funktioita, mutta tallennustarkkuus ei ollut kaikissa funktioissa matemaattisesti täysin tarkka. Tarkkuus on varmasti riittävä kuvioiden esittämiseen, mutta osa testeistä (*Centroid*, *Distance*, *Envelope*, *Touches*) epäonnistui, koska kantaan tallennettu arvo oli tarkka "vain"13. tai 14. desimaaliin asti. Funktion *Crosses* testit menivät läpi kahdella viivalla testatessa, mutta eivät viivan ja neliön interaktiota testatessa.

6.9.2 Microsoftin MS SQL CE -tuottaja

MS SQL CE -tuottaja ei tukenut spatiaalisia muotoja.

6.9.3 Devartin DotConnect for Oracle -tuottaja

Devartin DotConnect for Oracle -tuottaja tuki spatiaalisia muotoja ja funktioita listauksen 6.1 mukaisen konfiguroinnin jälkeen (“Spatial Service Type” 2020).

```
var config = OracleEntityProviderConfig.Instance;
config.SpatialOptions.SpatialServiceType = SpatialServiceType.NetTopologySuiteSigned;
```

Esimerkkikoodi 6.1. Tuen lisääminen spatiaaliselle datalle

Funktiot *Overlaps* ja *Within* palauttivat kaikille testitapauksille paluuarvon false, joten nämä funktiot eivät välttämättä toimi lainkaan. *Elevation* palautti olemattoman arvon (null) ja *Crosses* palautti truen myös täysin neliön sisällä sijaitsevalle viivalle. Funktiot *Area* ja *Contains* antavat ymmärtää, ettei kaikkia arvoja lasketa kovin tarkasti. *Area*-testissä ympyrän ala heitti todellisesta jo ensimmäisen desimaalin kohdalla. *Contains*-testissä neliön ja ympyrän sisäpuolella sijaitsevat pisteet (0.001 yksikön marginaalilla) palauttivat falsen.

6.9.4 Oraclen ODP.NET -tuottaja

ODP.NET-tuottaja ei tukenut spatiaalisia muotoja.

6.9.5 MySQL connector/NET -tuottaja

MySQL-tuottaja ei tukenut spatiaalisia muotoja.

6.9.6 PostgreSQL-tuottaja

Npgsql-tuottaja ei tukenut spatiaalisia muotoja.

6.9.7 Yhteenveto tuesta spatiaalisille funktioille

MS SQL -tuottaja ja Devartin DotConnect for Oracle -tuottaja olivat ainoat, jotka tukivat tietokannan spatiaalisia muotoja (*DbGeometry* ja *DbGeography*).

Molempia voisi varmasti hyödyntää tähän käyttötarkoitukseen, rajoitteet huomioon ottaen. Molemmilla oli hieman epätarkkuutta funktioissa ja Devartin DotConnect for Oracle -tuottajalla oli enemmän varsinaisesti väärin toimivia funktiototeutuksia.

7 Pohdinta

Entity Framework 6 käyttää oletuksena Microsoftin omaa MS SQL -tietokantaa olioiden tallentamiseen. Tässä tutkielmassa tutkittiin mahdollisuutta vaihtaa Entity Framework 6:n alla käytettävää tietokantaa tietokantakohtaista tuottajaa vaihtamalla. Tutkittaviksi tietokannoiksi valittiin MS SQL, MS SQL CE, Oracle, MySQL, PostgreSQL ja MongoDB. Kriteerit testattavien tietokantojen valinnalle on käyty läpi aiemmassa kirjallisuuskartoituksessa (Moilanen 2016, s. 8).

Microsoft ylläpitää listausta Entity Framework 6:tta tukevista tietokantakohtaisista tuottajista (“Entity Framework 6 Providers” 2018). Tutkimukseen valittiin tältä listalta valituille tietokannoille mainittuja tuottajia ja näin tutkimukseen tuli testattavia tuottajia sekä tietokantojen toimittajilta itseltään että kaupallisilta toimijoilta (CData, Devart). Tutkimukseen valituista kymmenestä tietokantakohtaisesta tuottajasta neljä jouduttiin hylkäämään jo tutkimuksen alkumetreillä, koska ne eivät tukeneet tietokannan luontia (CDatan molemmat tuottajat ja kaksi Devartin tuottajaa).

Jäljelle jäävillä kuudella tuottajalla oli yllättävän paljon eroavaisuuksia tuettujen toiminnallisuuksien suhteen. Täysin sokkona Entity Frameworkin alla olevaa tietokantaa ei voi tämän tutkimuksen perusteella suositella vaihdettavaksi, mutta tietokantakohtaisten tuottajien ongelmakohdat huomioiden tietokannan vaihto on kyllä mahdollista.

Tuottajien ongelmat voidaan jakaa kahteen osa-alueeseen: ei-tuetut toiminnallisuudet ja väärin toimivat toiminnallisuudet. Näistä vaarallisempia ovat väärin toimivat toiminnallisuudet, koska ne antavat käyttäjälle valheellisen tunteen toiminnallisuuden tukemisesta, mutta saattavat aiheuttaa ongelmia (bugeja) myöhemmin. Tässä kategoriassa kunniamaininnan ansaitsee PostgeSQL-kannalle suunnattu Npgsql-tuottaja, joka ei tukenut kaikkea toiminnallisuutta eikä sallinut desimaalilukujen ääriarvojen tallentamista, mutta ne toiminnallisuudet joita tuettiin ja arvot joiden tallennus kantaan onnistui, toimivat näiden testien valossa virheettömästi. Tähän suoritukseen ei pystynyt edes Entity Frameworkin mukana toimitettava de facto -tuottaja (MS SQL).

Laajimman tuen toiminnallisuuksille tarjosivat MS SQL -tuottaja ja Devartin DotConnect

for Oracle -tuottaja. Nämä olivat myös testijoukon ainoita tuottajia, jotka tukivat spatiaalisia tietotyyppejä ja funktioita. Molemmilla oli omat ongelmansa spatiaalisten funktioiden kanssa ja Devartin tuottajalla oli lisäksi toistuvana ongelmana tyhjän merkkijonon palauttaminen olemattomana (null). Muilta osin tuki oli laajaa, joskaan ei täysin virheetöntä.

MySQL-tuottajalla oli eniten haasteita testien kanssa, lopputuloksena sekä lukuisia ei-tuettuja ominaisuuksia että rajoitetusti tai virheellisesti toimivia ominaisuuksia. Näkisin kuitenkin, että minkä tahansa testeistä suoriutuneista tuottajista voisi ottaa käyttöön, kunhan ottaa kehitystyön aikana tuottajan rajoitteet huomioon.

8 Yhteenveto

Entity Framework tukee useampia erilaisia tallennusratkaisuja tietojen tallentamiseen, tietolähdekohtaisia tuottajia käyttäen. Tutkielman kirjoittajaa kiinnosti, onnistuuko vaihdos käytännössä kuinka helposti. Tässä tutkielmassa haettiin vastauksia seuraaviin kysymyksiin:

1. Onnistuuko Entity Frameworkin alla toimivan tietokannan vaihto helposti tietokantakohtaista tuottajaa vaihtamalla?
2. Miten laajasti eri tietokantakohtaiset tuottajat tukevat entiteettimallin mukaisia perustietotyyppejä?
3. Miten laajasti eri tietokantakohtaiset tuottajat toteuttavat tuen Entity Frameworkin tukemille kanonisille funktioille?

Ensimmäiseen kysymykseen vastauksena voisi olla "Ehkä". Tämä riippuu täysin kehitettävän sovelluksen käyttämistä Entity Frameworkin ominaisuuksista, mutta oletusta ongelmattomasta vaihdosta ei voida tämän tutkielman testitulosten perusteella tehdä. Vaihdos voidaan tehdä toteuttajien rajoitteet huomioiden ja testaamalla ohjelman toiminnallisuus hyvin vaihdon yhteydessä.

Entiteettimallin perustietotyyppejä tuettiin melko hyvin, mutta myös ongelmia oli. Enimmäkseen ongelmat ilmenivät testeissä, joissa testattiin tietotyyppien ääriarvojen tallennusta. Nämä rajoitteet on kuitenkin syytä ottaa huomioon tietokantakohtaisen tuottajan valintaa tehdessä.

Tuki kanonisille funktioille oli hyvin vaihtelevaa. Jos spatiaalisille funktioille on käyttöä, tässä tutkielmassa testatuista tuottajista vaihtoehdoiksi jäävät Microsoftin de facto MS SQL -tuottaja tai Devartin DotConnect for Oracle -tuottaja.

Kaiken kaikkiaan voisi todeta, että tuottajilta olisi voinut odottaa tasaisempaa suorittamista. Tuottajien toteuttajat vaikuttaisivat mainostavan tuottajiensa hyviä puolia, mutta huonosti toimivia tai toimimattomia ominaisuuksia ei mainita. Tässä on riski käyttäjän kannalta ja näkisinkin, että osuu valinta mihin tuottajaan tahansa, laajat ja tietokantaan asti tietoa tallentavat hyväksyntätestit ovat tarpeen, jotta yllätyksiltä vältytään.

Jatkotutkimuksena tämän tutkielman pohjalta erityisen kiinnostava aihealue voisi olla tuottajien suorituskyvyn testaaminen. Sovelluskehityksessä suorituskyvyllä on kuitenkin isoja vaikutuksia ohjelmiston toimintaan ja olisi mielenkiintoista tietää, voisiko oikean tuottajan valinnalla kenties välttyä suorituskykyongelmilta laajoja tietomassoja käsitellessä.

Lähteet

Abdalhakim, H. 2009. “Addressing Burdens of Open Database Connectivity Standards on the Users”. Teoksessa *2009 Third International Symposium on Intelligent Information Technology Application Workshops,* 305–308. doi:10.1109/IITAW.2009.40.

“ADO Overview and Benefits”. 2017. Viitattu 10. marraskuuta 2020. https://docs.microsoft.com/en-us/sql/ado/guide/ado-introduction.

Agarwal, Vidya Vrat. 2012. “Introduction to ADO.NET”. Teoksessa *Beginning C# 5.0 Databases,* 171–182. Berkeley, CA: Apress. ISBN: 978-1-4302-4261-1. https://doi.org/10.1007/978-1-4302-4261-1.

“Anatomy of the ADO.NET entity framework”. 2007. Viitattu 24. lokakuuta 2019. http://dl.acm.org/citation.cfm?id=1247580.

Armas, J., P. Navas, T. Mayorga, P. Rengifo ja B. Arévalo. 2017. “Optimization of code lines and time of access to information through object-relational mapping (ORM) using alternative tools of connection to database management systems (DBMS)”. Teoksessa *2017 2nd International Conference on System Reliability and Safety (ICSRS),* 500–504. doi:10.1109/ICSRS.2017.8272872.

Blakeley, J. A. 1997. “Universal data access with OLE DB”. Teoksessa *Proceedings IEEE COMPCON 97. Digest of Papers,* 2–7. doi:10.1109/CMPCON.1997.584662.

“Canonical Functions”. 2017. Viitattu 12. joulukuuta 2019. https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/language-reference/canonical-functions.

“datetime (Transact-SQL)”. 2017. Viitattu 12. marraskuuta 2020. https://docs.microsoft.com/en-us/sql/t-sql/data-types/datetime-transact-sql.

“Development Approaches with Entity Framework”. 2020. Viitattu 10. marraskuuta. https://www.entityframeworktutorial.net/choosing-development-approach-with-entity-framework.aspx.

“Entity Data Model: Primitive Data Types”. 2017. Viitattu 28. marraskuuta 2019. https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/entity-data-model-primitive-data-types.

“Entity Framework 6”. 2016. Viitattu 24. lokakuuta 2019. https://docs.microsoft.com/en-us/ef/ef6.

“Entity Framework 6 Providers”. 2018. Viitattu 13. marraskuuta 2020. https://docs.microsoft.com/en-us/ef/ef6/fundamentals/providers/.

“Entity Framework Core”. 2020. Viitattu 10. marraskuuta 2020. https://docs.microsoft.com/en-us/ef/core/.

“Entity Framework overview”. 2018. Viitattu 10. marraskuuta 2020. https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/overview.

“Fluent API - Configuring and Mapping Properties and Types”. 2016. Viitattu 10. marraskuuta 2020. https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/fluent/types-and-properties.

“Git repository for Entity Framework 6”. 2016. Viitattu 9. marraskuuta 2020. https://github.com/dotnet/ef6.

Hamilton, Bill. 2004. *NUnit Pocket Reference: Up and Running with NUnit.* "O’Reilly Media, Inc.".

Hirani, ym. 2013. *Entity Framework 6 Recipes.* New York: Apress.

Jennings, Roger. 2009. *Professional ADO.NET 3.5 with LINQ and the Entity Framework.* John Wiley / Sons, Incorporated.

“Known Issues and Considerations in LINQ to Entities”. 2017. Viitattu 24. lokakuuta 2019. https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/language-reference/known-issues-and-considerations-in-linq-to-entities.

Lerman, Miller. 2011. *Programming Entity Framework: Code First.* O’Reilly Media.

Lipitsäinen, A. 2010. “DBTechNet Tutorial on: ORM–Object Relational Mapping”. Viitattu 5. joulukuuta 2019. http://myy.haaga-helia.fi/~dbms/dbtechnet/labs/dae_lab/Orm.pdf.

“List of object-relational mapping software”. 2020. Viitattu 10. marraskuuta. https://en.wikipedia.org/wiki/List_of_object-relational_mapping_software.

Moilanen, Jere. 2016. “Entity Framework 6:n tuki eri tietokannoille”. Viitattu 24. lokakuuta 2019. https://jyx.jyu.fi/handle/123456789/50055.

Mueller, John Paul. 2013. *Microsoft ADO.NET Entity Framework Step by Step.* Yhdysvallat: Microsoft Press.

“OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture”. 2011. Viitattu 5. marraskuuta 2020. http://portal.opengeospatial.org/files/?artifact_id=25355.

“Performance considerations for EF 4, 5, and 6”. 2014. Viitattu 9. marraskuuta 2020. https://docs.microsoft.com/en-us/ef/ef6/fundamentals/performance/perf-whitepaper.

“Provider Support for Spatial Types”. 2016. Viitattu 5. joulukuuta 2019. https://docs.microsoft.com/en-us/ef/ef6/fundamentals/providers/spatial-support.

“Security Considerations (Entity Framework)”. 2017. Viitattu 9. marraskuuta 2020. https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/security-considerations.

Singh, Rahul Rajat. 2015. *Mastering Entity Framework.* Birmingham: Packt Publishing Ltd.

“Spatial Service Type”. 2020. Viitattu 13. marraskuuta. https://www.devart.com/dotconnect/oracle/docs/SpatialServiceType.html.