Tampereen ammattikorkeakoulu

SQL-tietokantaan perustuva datankeräys osana logiikkaohjelmaa

Eero Taittonen

OPINNÄYTETYÖ
Toukokuu 2021

Sähkö- ja automaatiotekniikka
Automaatiotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Sähkö- ja automaatiotekniikka
Automaatiotekniikka

TAITTONEN, EERO:
SQL-tietokantaan perustuva datankeräys osana logiikkaohjelmaa

Opinnäytetyö 45 sivua, joista liitteitä 0 sivua
Toukokuu 2021

Opinnäytetyön tarkoituksena oli tutkia prosessidatan keräystä ohjelmoitavalta logiikalta SQL-tietokantaan. Tavoitteena oli selvittää, soveltuuko SQL-tietokanta datankeruualustaksi ja onko tiedonsiirto vaadittavalla tasolla. Tiedonsiirtoon suunniteltujen ratkaisujen täytyy toimia siten, että ohjelmoitavalta logiikalta siirretty data olisi reaaliaikaista ja keräys suoritettaisiin luotettavasti.

Järjestelmän toteutuksen perustana toimi Siemensin laatima alustusohjelma, joka siirtää dataa SQL-tietokantaan. Tämän alustusohjelman ympärille luotiin muut tiedonsiirtoon vaadittavat toimet, joiden toimintaa testattiin Siemensin simulointiympäristössä. Valmis logiikkaohjelma yhdistää tarvittavat komennot ja prosessiarvot yhdeksi komentoriviksi, joka lähetetään SQL-palvelimelle. Palvelimella data siirtyy tietokantaan luotuun tauluun komentorivin mukaisesti.

Tutkimuksen päämääränä oli selvittää tiedonsiirto-ohjelman toimivuutta sekä kuormittavuutta. Tätä testattiin simuloimalla ohjelmoitavaa logiikkaa ja suorittamalla tiedonsiirtokomentoja tietokantaan. Testauksessa tarkasteltiin tiedonsiirron nopeutta ja määrää, mistä voitiin päätellä datankeräyksen käytettävyyttä voimalaitosprosesseissa.

Työn tuloksista todettiin, että datan keräys SQL-tietokantaan on luotettava ja helppo tapa taltioida dataa reaaliajassa. Työssä käytettyjen ohjelmien välinen tiedonsiirtonopeus ja -määrät olivat riittävällä tasolla, joten tämän menetelmän käyttäminen ja kehittäminen datan keräykseen oli perusteltua.

Asiasanat: SQL, tietokantaohjelmointi, logiikkaohjelmointi, tiedonsiirto

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Electrical Engineering
Automation Engineering

TAITTONEN, EERO:
SQL-based Data Harvest as a Part of Logic Programming

Bachelor's thesis 45 pages, appendices 0 pages
May 2021

The purpose of this thesis was to study the harvesting of process data from programmable logic to an SQL database. The aim was to find out whether the SQL database is suitable as a data collection platform and whether the data transfer is at the required level. Solutions designed for data transfer must work in such a way that the data transferred from the programmable logic control is real-time and that the collection is performed reliably.

The system was based on an initialization program developed by Siemens, which transfers data to an SQL database. Other actions required for data transfer were created around this initialization program, and they were tested in the Siemens simulation environment. The ready-made logic program combines the necessary commands and process values into a single command line that is sent to the SQL server. Data is transferred to the table created in the database according to the command line.

The aim of this study was to find out the functionality and workload of the data transfer program. This was tested by simulating programmable logic control and transferring communication commands to the database. The study examined the speed and volume of data transfer, which enabled deducing the usability of data harvesting in power plant processes.

The conclusion of the study is that collecting data into an SQL database is a reliable and easy way to store data in real-time. Data transfer speed and data transfer volumes between the programs used were at a sufficient level, so the use and development of this method for data collection was justified.

Key words: SQL, database programming, logic programming, data transmission

SISÄLLYS

LYHENTEET JA TERMIT

API	sovellusohjelmointirajapinta (API, Application Programming Interface)
BOOL	totuusarvot (tosi/epätosi)
CLI	kutsuntarajapinta (Call-Level Interface)
CPU	suoritin (Central Processing Unit)
DATA	tieto
DB	tietokanta (SQL, Database)
DB	datalohko (PLC, Data Block)
DCL	tietojen ohjauskieli (Data Control Language)
DDL	tietojen määrityskieli (Data Definition Language)
DML	tietojenkäsittelykieli (Data Manipulation Language)
EPC	avaimet käteen -toimitus (Engineering, Procurement, Construction)
FB	toimilohko (Function Block)
Fuzzy Control	sumea säätö
IEC	kansainvälinen standardijärjestö (International Electrotechnical Commission)
IP	internetprotokolla (Internet Protocol)
NULL	tuntematon arvo
OB	organisaatiolohko (Organisation Block)
PLC	ohjelmoitava logiikka (Programmable Logic Controller)
PN	väyläteknologia (Profinet)
RDBMS	relaatiotietokantajärjestelmä (Relational Database Management System)
SDCI	singledrop-digitaalisen viestinnän rajapinta pienille antureille ja toimilaitteille (Single-drop Digital Communication Interface)
Server	palvelin
SQL	kyselykieli (Structured Query Language)
TAG	nimike
TCP	tietoliikenneprotokolla (Transmission Control Protocol)

1 JOHDANTO

Opinnäytetyön teettää VEO Oy. Työn tarkoituksena on selvittää, miten prosessidataa pystytään siirtämään ohjelmoitavalta logiikalta SQL-tietokantaan. Datan keräys mahdollistaa prosessidatan analysoinnin ja seurannan. Datan keräykseen suunniteltujen ratkaisujen täytyy toimia siten, että automaatiojärjestelmästä siirretty data on reaaliaikaista ja keräys suoritetaan luotettavasti. Työssä käydään läpi tiedonsiirtoon vaadittavat elementit niiltä osin, mitä käyttäjän tulee tietää ohjelmien kulusta. Tätä datan keräyksen mallia yritys voi soveltaa esimerkiksi takuuaikaisten tuotteiden seurantaan ja raportointiin.

Opinnäytetyön tarkastelussa olevat automaatiojärjestelmät rakennetaan Siemens SIMATIC S7-1500 -ohjelmoitavan logiikan ympärille, joka mahdollistaa datan keräykselle kattavan valikoiman erilaisia sovelluksia. Datan keräyksen sovellukseksi valikoituu SQL-tietokantaan pohjautuva ilmainen Microsoft SQL Server Express, mikä löytyy yrityksen tietokannasta valmiiksi ladattuna. Tiedonsiirron alustana toimii Siemensin laatima alustusohjelma, joka on tehty toimimaan kyseisen SQL-palvelimen kanssa. Opinnäytetyössä selvitetään, miten sovellukset liitetään toisiinsa ja miten dataa siirretään SQL-palvelimelle.

Datan keräyksessä, eli tiedonsiirrossa, käytetään pääasiallisesti kahta työkalua: TIA Portal V16-ohjelmointiympäristöä, jolla rakennetaan sovellus ohjelmoitavalle logiikalle ja Microsoft SQL Server Management Studio-tietokantatyökalua, jolla käsitellään tietokantoja. Datan keräystä varten täytyy luoda kolme päärakennetta: yhteys, tietokanta sekä sovellus. Yhteys luodaan SQL Server 2016 Express ja ohjelmoitavan logiikan välille siten, että sovelluksien välinen tiedonsiirto onnistuu. Tietovirta sovellusten välillä tapahtuu taulukkomuotoisena, mikä mahdollistaa logiikan suoran yhteyden SQL-palvelimelle ja sitä kautta lähettää dataa tietokantaan. Tietokanta luodaan SQL-tietokantatyökalulla, minne kerättävä data taltioidaan ja missä sitä päästään analysoimaan. Sovellus luodaan TIA Portal -ohjelmointiympäristössä, jossa data muokataan ensin siirrettävään muotoon, eli komentoriviksi. Tämän jälkeen komentorivi lähetetään tietokantaan Siemensin laatiman tiedonsiirtolohkon avulla. Tietokannasta data voidaan tulostaa tauluun, jonka jälkeen se on käsiteltävissä tietokantatyökalussa.

Opinnäytetyö pyrkii vastaamaan VEO Oy:n tarpeisiin kartoittaa tiedonsiirtoon vaadittavien ohjelmien ja sovellusten käytettävyyttä. Tutkimuksen tehtävänä on selvittää SQL-tietokannan tiedonsiirtonopeudet sekä tiedonsiirtomäärät, eli millä syklillä ja kuinka paljon dataa voi ohjelmoitavalta logiikalta SQL-tietokantaan siirtää. Toisena tutkimuskohteena on tiedonsiirtoon vaadittavien sovellusten käytettävyys tämän kaltaisissa prosesseissa. Edellä mainittujen seikkojen avulla tämän tutkimuksen päämääränä on selvittää SQL-tietokantaan perustuvan datankeräyksen käytettävyyttä osana automaatiojärjestelmää. Työn lopputuloksia käytetään datan keräyksen kehityksessä ja mahdollisesti asiakkaille tarjottavien palveluiden tukena.

2 VEO OY

VEO Oy on vuonna 2000 perustettu vaasalainen osakeyhtiö, jonka toimitusjohtajana toimii Timo Ala-Heikkilä. Yrityksellä on toimipisteitä Suomen lisäksi Norjassa, Ruotsissa ja Isossa-Britanniassa, joissa se työllistää 385 henkilöä. Yrityksen pääkonttori ja tuotantotehdas sijaitsevat Vaasassa ja muita Suomen toimipisteitä on Paimiossa, Seinäjoella ja Rovaniemellä. Vuoden 2019 liikevaihto oli noin 95,3 miljoonaa euroa. Tätä edeltävällä vuodella yrityksen liikevaihto oli noin 87,5 miljoonaa euroa, joten yritys kasvatti liikevaihtoa noin 8,9%. (Asiakastieto 2019.)

VEO:n pääasiallinen toimiala on sähköautomaatio. Se tarjoaa automaatio- ja sähköistysratkaisuja energiantuotantolaitoksiin eripuolille maailmaa. Yrityksellä on kolme pääasiallista liiketoimintayksikköä: sähköntuotanto, -jakelu ja -käyttö.

Sähköntuotannossa VEO tarjoaa automaatio- ja sähköistysratkaisuja kaikille voimalaitostyypeille vesi-, tuuli- ja lämpövoimalaitoksista moottori- ja hybridivoimalaitoksiin. Ratkaisuihin kuuluu vanhojen järjestelmien päivittäminen ja muokkaaminen nykyaikaisiin järjestelmiin sekä uusiutuvien energialähteiden osuuden kasvattaminen. Konkreettisia esimerkkejä VEO:n tuottamista laitteistoista ovat pien- ja keskijännitekojeistot, muuntoasemat ja turbiinin ohjainjärjestelmät. (VEO Oy 2021.)

Sähkönjakelussa VEO on yksi pohjoismaiden suurimmista sähköasemien EPC-toimittajista (Engineering-Procurement-Construction, suunnittelu-hankinta-rakentaminen). EPC-malli kuvastaa liiketoimintaa, missä yritys tarjoaa kaiken tarvittavan asiakkaalleen, eli niin sanottua avaimet käteen -toimitusta. VEO tarjoaa EPC-toimitusta keski- ja suurjänniteratkaisuille alue- ja kantaverkoissa sekä teollisuudessa, mitkä sisältävät älykkäät ohjaus- ja suojausjärjestelmät, maa- ja vesirakennustyöt sekä keskijännitekojeistojen valmistuksen. (VEO Oy 2021.)

Sähkönkäytön ratkaisuja VEO tarjoaa prosessiteollisuudelle, nostureille sekä laivoille. EPC-toimitukseen sisältyy pien- tai keskijännitekojeistot, virityksen moottoreille ja generaattoreille sekä automaatio- ja visualisointijärjestelmät. Toimitukseen kuuluu myös suunnittelu ja käyttöönotto. (VEO Oy 2021.)

3 TIETOKANTAOHJELMOINTI

SQL (Structured Query Language) on virallinen ja standardoitu ohjelmointikieli relaatiotietokantojen luomiseen, muokkaamiseen ja noutamiseen. Relaatiotietokannat koostuvat tauluista, joissa on sarakkeita (columns) ja rivejä (rows). Ne sijaitsevat relaatiotietokantojen hallintajärjestelmässä (RDBMS). Tietokantojen hallintajärjestelmä on SQL-palvelimessa toimiva järjestelmä, missä tietokannat sijaitsevat ja missä niitä voidaan muokata SQL-kyselyjen (query) avulla. SQL-tietokanta tarvitsee toimiakseen aina relaatiotietokannan hallintajärjestelmän. Eri valmistajien hallintajärjestelmiä ovat esimerkiksi Oracle, MySQL, DB2, Sybase ja SQL Server. (Oppel 2016, 4.)

3.1 SQL-kyselykielen toiminnot

SQL-kielen toimintojen jako voidaan suorittaa usealla eri tavalla riippuen siitä, mikä ominaisuus on tarkastelussa. SQL-standardi jakaa toiminnot kymmeneen eri alalajiin, kun taas yleisesti suositellaan, että toiminnot jaetaan kolmeen alalajiin (Oppel 2016, 20). Tässä työssä tarkasteluun otetaan yleisessä dokumentaatiossa käytettävät kolme alalajia: tietokantojen rakenteen määrityskieli, tietojenkäsittelykieli ja tietojen ohjauskieli. Tämä jako määräytyy toimintojen käyttötarkoituksen perusteella.

Tietokannan rakenteen määrittelykieli

DLL (Data Definition Language) koostuu tietokannan rakenteen määrittämiseen käytettävistä komennoista. Tällaisia komentoja ovat esimerkiksi:

- `CREATE`, millä luodaan tietokantoja tai objekteja.
- `ALTER`, millä muutetaan tietokantaa tai objektia.
- `DROP`, millä poistetaan tietokanta tai objekti.
- `TRUNCATE`, millä poistetaan välittömästi taulun tiedot.

(Oppel 2016, 19.)

Tietojenkäsittelykieli

DML (Data manipulation Language) koostuu tietokannan dataan kohdistuvista kysely- ja muokkauskomennoista. Tällaisia komentoja ovat esimerkiksi:

- `SELECT`, millä haetaan (tulostetaan) dataa tietokannasta.
- `INSERT`, millä lisätään dataa tietokantaan.
- `UPDATE`, millä päivitetään dataa tietokannassa.
- `DELETE`, millä poistetaan dataa tietokannasta.

(Oppel 2016, 20.)

Tietojen ohjauskieli

DCL (Data Control Language) koostuu tietokannan käyttöoikeus- ja tapahtumien hallintakomennoista. Tällaisia komentoja ovat esimerkiksi:

- `GRANT`, millä annetaan käyttäjälle käyttöoikeuksia tietokantaan.
- `REVOKE`, millä poistetaan käyttäjältä käyttöoikeuksia tietokantaan.
- `COMMIT`, millä hyväksytään tapahtuma.
- `ROLLBACK`, millä hylätään tapahtuma.

(Oppel 2016, 19.)

Tämän työn kannalta merkittävimpiä komentoja ovat tietojenkäsittelykielen komennot. Tämä johtuu siitä, että ohjelman perustana toimii tiedonsiirto, missä dataa lisätään `INSERT`-komennolla tietokannassa sijaitsevaan tauluun. Toinen tärkeä komento on `SELECT`, millä taulun tiedot saadaan tulostettua.

3.2 SQL-datatyypit

Tietokantaa luotaessa jokaiselle sarakkeen muuttujalle täytyy antaa nimi ja datatyyppi. Datatyyppi annetaan sen mukaan, mitä kyseisen sarakkeen muuttujalla halutaan ilmaista. Datatyyppi rajaa arvon, minkä käyttäjä voi syöttää muuttujalle. Datatyyppi voi rajoittaa muuttujalle syötettävän arvon esimerkiksi pelkkiin numeroihin tai se voi sallia kaikkien merkkien käytön. Tietokannan sarakkeisiin tulevien muuttujien merkkien maksimi määrä täytyy tietää ennen datatyypin valitsemista.

Tämä johtuu siitä, että datatyypit tarvitsevat eri määrän muistia riippuen datatyypille varattujen merkkien määrästä. Kannattaa valita sellainen datatyyppi, mikä on mahdollisimman lähellä tarvittavaa merkkimäärää, sillä ylimääräinen tila hidastaa ohjelmia ja toisaalta liian pieni tila vääristää lopputulosta. (Bush 2020.) SQL tukee kolmea eri datatyyppiä: ennalta määritetyt-, rakennetut-, ja käyttäjän määrittämät datatyypit (Oppel 2016, 55).

Ennalta määritetyt datatyypit

Ennalta määritetyt datatyypit, eli niin sanotut sisään rakennetut datatyypit ovat yleisimpiä nimettyjä elementtejä SQL-kielessä. Jokaiselle datatyypille on omat ominaisuudet, kuten merkkien määrät ja rajaukset. Tällaisia ennalta määritettyjä datatyyppejä on viisi kappaletta: boolean, merkkijono, numeerinen, päivämäärä ja aikaväli. (Oppel 2016, 55.)

- Boolean (`BOOL`) datatyyppi on yksinkertainen, koska sillä on vain kolme mahdollista tilatietoa: tosi, epätosi ja tuntematon. Booleania käytetään totuusarvoina esimerkiksi kytkinohjauksessa, missä kytkimellä ei ole kuin kaksi arvoa: päällä ja pois päältä. Tuntematonta arvo tulee silloin, kun ohjelma ei tiedä mikä tila kyseisellä boolean muuttujalla on. Tällöin SQL-kielessä muuttuja saa arvon `NULL`. (Oppel 2016, 58.)
- Merkkijono (`STRING`) datatyypit koostuvat merkistöihin perustuvista arvoista. Esimerkkinä kirjainten, numeroiden ja erikoismerkkien yhdistelmät. Eri datatyypit määrittelevät sallitun pituuden ja merkit, joita muuttujalle voi syöttää. SQL-kielessä on kolme eri merkkijonotyyppiä:
    - Merkkijonot, jotka sallivat arvot tietystä merkistöstä. Merkit rajataan joko oletusjoukosta tai muuttujalle määritetään joku tietty merkistö. SQL-kielen merkkijonotyypit ovat: `CHARACTER`, `CHARACTER VARYING`, `CHARACTER LARGE OBJECT` ja `XML`.
    - Kansalliset merkkijonot, jotka sallivat merkistöt, jotka ovat relaatiotietokantojen hallintajärjestelmän merkistöjen joukossa. Näitä hyödynnetään, jos tietokantaan halutaan dataa usealla eri kielellä. SQL-kielen kansalliset merkkijonot ovat: `NATIONAL CHARACTER`, `NATIONAL CHARACTER VARYING` ja `NATIONAL CHARACTER LARGE OBJECT`.

  - Binaarijoukot, jotka sallivat binaariluvuista muodostetut merkkijonot (bittijonot). Tämä tarkoittaa sitä, että datatyypit sallivat vain merkit 1 ja 0. Datatyyppi jäsentelee bitit okteteiksi, mitkä ovat kahdeksan bitin pituisia. Näitä käytetään kuvien ja ääninäytteiden tallentamiseen tietokantaan. SQL-kielen binaarijoukot ovat: `BINARY`, `BINARY VARYING` ja `BINARY LARGE OBJECT`.

(Oppel 2016, 56.)

- Numeeriset (`NUMERIC`) datatyypit koostuvat numeroista, mutta ne eroavat toisistaan tarkkuuden ja asteikon myötä. Tarkkuus tarkoittaa numeroiden määrää kokonaisuudessaan, kun taas asteikko määrittää desimaalipilkun paikan muuttujassa. Esimerkki: muuttujan tarkkuus on 5 ja asteikko 2, jolloin numero voi olla väliltä -999,99…999,99. SQL-kielessä on kahta eri numeerista datatyyppiä:
  - Tarkat numerot, jotka sallivat arvot, joilla on asetettu tarkkuus ja asteikko. Tällöin desimaalipilkku on aina sille määritetyllä paikalla. SQL-kielen tarkat numeeriset datatyypit ovat: `NUMERIC`, `DECIMAL`, `BIGINT` ja `SMALLINT`.
  - Likimääräiset numerot, jotka sallivat arvot, joilla on tarkkuus ilman asteikkoa. Tällöin desimaalipilkku on niin sanottu kelluva, eli se voi sijaita missä tahansa kohtaa muuttujaa. SQL-kielen likimääräisten numerojen datatyypit ovat: `REAL`, `DOUBLE PRECISION` ja `FLOAT`.

(Oppel 2016, 56.)

- Päivämäärädatatyypit (`DATETIME`) koostuvat päivämääriin ja kellonaikoihin liittyvistä arvoista. SQL-kielen perus päivämäärädatatyypit ovat: `DATE`, `TIME` ja `TIMESTAMP`. Näiden lisäksi on useita eri variaatioita, joissa on huomioitu eri aikavyöhykkeet sekä aikaerot. Tällöin lisätään datatyypin perään lause: `WITH TIME ZONE`. Eri relaatiotietokantojen valmistajilla on suuria eroja päivämäärädatatyyppien variaatioissa, joten käyttäjän täytyy tutustua oman tuotevalmistajan käytäntöihin ja määrityksiin. (Oppel 2016, 58.)
- Aikavälidatatyypit (`INTERVAL`) koostuvat kahden eri päivämäärän välisestä erosta. Nämä perustuvat päivämäärädatatyyppeihin, joten tuotekohtaisia eroja voi olla. SQL-kielessä on kahta eri aikavälidatatyyppiä:
  - Vuosi-kuukausi aikavälit, jotka määrittävät vuosien, kuukausien tai molempien väliset aikamääreet. SQL-kielen vuosi-kuukausi aikavälit ovat: `YEAR`, `MONTH`.

- Päivä-aika aikavälit, jotka määrittävät päivien, tuntien, minuuttien tai sekuntien aikavälit. SQL-kielen päivä-aika aikavälit ovat: `DAY`, `HOUR`, `MINUTE` ja `SECOND`.

Näissä täytyy huomioida, että aikavälityyppejä ei voida sekoittaa keskenään toistensa kanssa. (Oppel 2016, 60.)

Rakennetut datatyypit

Rakennetut datatyypit ovat myös nimettyjä elementtejä SQL-kielessä. Ero ennalta määritettyihin datatyyppeihin on se, että rakennetut datatyypit ovat usein monimutkaisia ja niihin mahtuu useita eri arvoja. (Oppel 2016, 55.)

Käyttäjän määrittämät datatyypit

Käyttäjän määrittämät datatyypit ovat käyttäjän itse luomia eikä niistä löydy valmiita elementtejä SQL-kielessä. Ne jaetaan kahteen eri alalajiin: rakenteellisiin ja erillisiin datatyyppeihin. (Oppel 2016, 55.)

- Rakenteelliset datatyypit koostuvat yhdestä tai useammasta tunnusmerkistä, jotka perustuvat muihin datatyyppeihin. käyttäjä rakentaa funktion, joka toimii aina samalla tavalla. Funktioon syötetään parametrit, jonka jälkeen ohjelma palauttaa yhden arvon parametrien perusteella.
- Erilliset datatyypit perustuvat ennalta määritettyihin datatyyppeihin, missä käyttäjä muokkaa datatyypin haluamaansa muotoon.

SQL-kieli tarjoaa `CREATE TYPE` -komennon, jolla käyttäjä voi luoda omia datatyyppejä. Tämä komento toimii siten, että komennon jälkeen tulee uuden datatyypin nimi, datatyyppi, mihin se perustuu ja funktio, miten merkit sijoittuvat sarakkeeseen. Esimerkiksi: `CREATE TYPE testi AS NUMERIC (7,3) FINAL;`. Esimerkissä uuden datatyypin nimi on testi ja se perustuu numeeriseen datatyyppiin. Sulkeiden sisälle laitettavat arvot kertovat datatyypin tarkkuuden ja asteikon, eli datatyyppi on väliltä -9999,999…9999,999. kyselyn perään kirjoitetaan `FINAL`-komento, mikä kertoo, ettei datatyypille tule alatyyppejä. Kun datatyyppi on luotu, voi sitä käyttää samaan tapaan kuin ennalta määritettyjä datatyyppejä. (Oppel 2016, 64.)

3.3 SQL-lausekkeiden suorittaminen

Tietokantaan tehtäviä SQL-kyselyitä voidaan suorittaa usealla eri tavalla. SQL-standardi määrittää neljä eri toteutustapaa sille, miten ja millä kyselyitä tietokantaan voidaan suorittaa. Tässä luvussa esitellään eri toteutustavat.

Interaktiivinen SQL

Interaktiivinen SQL -menetelmällä (Direct Invocation) pystytään kommunikoimaan tietokantoihin suoraan käyttöliittymäsovelluksessa. Tämä tarkoittaa sitä, että kyselyt voidaan suorittaa suoraan sovellusikkunassa ja sovellus palauttaa tulokset niin nopeasti kuin suorittimen (CPU) kapasiteetti sallii. Tämä on helppo ja nopea keino tarkastella ja testata tietokantoja. Tällaisia tietokantatyökaluja tarjoavat muun muassa Microsoft SQL Serverin Management Studio sekä Oraclen SQL Delveloper. (Oppel 2010, 20.)

Upotettu SQL

Upotettu SQL -menetelmällä (Embedded SQL) kyselyt koodataan suoraan isäntäohjelmointikielelle. Tämä tarkoittaa sitä, että kyselyt niin sanotusti upotetaan ohjelmakoodiin, jonka jälkeen ohjelmakoodi esikäsitellään ja kootaan. Tämä on yksi yleisimmistä toteutustavoista suorittaa kyselyjä tietokantaan. Tyypillisiä ohjelmointikieliä ovat Java, C tai COBOL. (Oppel 2010, 20.)

Moduulinen SQL

Moduulinen SQL -menetelmällä (Module Binding) kyselyt muodostetaan itsenäisillä moduuleilla, jotka ovat erillään isäntäohjelmointikielestä. Tietokantaan suoritettavat kyselyt operoidaan isäntäohjelmointikielessä kutsumalla haluttua moduulia, joka suorittaa kyselyn tietokantaan. Tämä menetelmä on harvinainen eikä sitä juurikaan tueta SQL-toteutuksissa. (Oppel 2010, 20.)

Kutsuntarajapinta

CLI (Call-Level Interface) on menetelmä, jolla voidaan olla yhteydessä SQL-tietokantoihin sovellusohjelmointirajapinnan (API, Application Programming Interface) avulla. Sovellusohjelmointirajapinnassa on ennalta määriteltyjä rutiineja yhteyden muodostamiseen sekä tietojen muokkaamiseen ja hakuun. Ohjelmointikielellä kutsutaan rutiineja, joiden mukaan tietokantaan tehdyt kyselyt suoritetaan. Suoritteen jälkeen tulokset palautetaan ohjelmalle. Tällaista toteutusta tarjoaa esimerkiksi Microsoft Open Database Connectivity. (Oppel 2010, 20.)

4 LOGIIKKAOHJELMOINTI

Logiikkaohjelmointiin tarvittavia laitteita ja sovelluksia ovat ohjelmoitava logiikka, ohjelmointiympäristö ja kohde, mitä ohjataan. Näitä voivat olla esimerkiksi Siemensin SIMATIC S7 PLC, TIA Portal -ohjelmointiympäristö ja vesivoimalaitos. Logiikkaohjelmoinnissa käyttäjä rakentaa ohjelmointiympäristössä ohjelman PLC:hen, millä ohjataan kohdetta, eli prosessia käyttäjän luomien ehtojen perusteella. Opinnäytetyön laitteistot ja sovellukset pohjautuvat kansainvälisen standardijärjestön IEC (International Electrotechnical Commission) standardisarjaan. (Siemens 2013.)

IEC 61131 standardisarja käsittelee elektronisesti ohjelmoitavien logiikoiden ja oheislaitteiden määritteitä ja vaatimuksia. Standardi koostuu ohjeista, määrityksistä ja vaatimuksista, jotka jaetaan yhdeksään osaan niiden käyttötarkoituksen perusteella.

- IEC 61131-1, johdanto standardeihin, määritelmät ja perusominaisuudet.
- IEC 61131-2, laitevaatimukset ja testaukset.
- IEC 61131-3, ohjelmointikieliin liittyvät määritykset.
- IEC 61131-4, käyttöohjeisiin liittyvät määritykset.
- IEC 61131-5, tietoliikenteen määritykset laitteiden välille.
- IEC 61131-6, toiminnallisen turvallisuuden määritykset.
- IEC 61131-7, sumean ohjauksen (Fuzzy Control) ohjelmointikielet.
- IEC 61131-8, ohjeet ohjelmointikielen soveltamiseen ja käyttöönottoon.
- IEC 61131-9, SDCI (Single-drop digital communication interface), eli OI-Linkki pienten laitteiden ja sensorien digitaaliseen liitäntään.

(John & Tiegelkamp 2010, 14.)
Tässä työssä pääprioriteetti on standardin IEC 61131-3 sisältö, eli määritykset ja perusominaisuudet logiikkaohjelmointiin.

4.1 Ohjelmointikielet

Ohjelmoitavien logiikoiden standardin kolmas osa (IEC 61131-3) käsittelee ohjelmointikieliin liittyviä vaatimuksia ja määrityksiä (John & Tiegelkamp 2010, 14). Tämä standardi on opinnäytetyössä käytettävien ratkaisujen perustana. Standardoituja kieliä ohjelmoinnille on viisi kappaletta. Tässä luvussa esitellään ohjelmointikielet pääpiirteittäin ja jokaisella on rakennettu esimerkkiohjelma kuvaamaan käsittelyssä olevaa ohjelmointikieltä.

Esimerkkiohjelmien toimintaperiaate on sama, ohjelmassa on kolme tulomuuttujaa (`A`, `B` ja `C`) ja yksi lähtömuuttuja (`X`). Jokainen muuttuja on bool-muuttuja, eli sen tila on joko 1 (tosi) tai 0 (epätosi). Ohjelman lähtömuuttuja `X` saa tilan 1, kun tulomuuttujat `A` tai `B` on tilassa 1 ja `C` on tilassa 1. Esimerkkiohjelmassa on siis kaksi toimintoa: JA (`&`, `AND`) sekä TAI (`≥1`, `OR`).

Käskylista

IL (Instruction List) on tekstimuotoinen ohjelmointikieli, mikä muistuttaa tietokoneohjelmoinnista tuttua Assembly-konekieltä (kuvio 1). Tämä ohjelmointikieli on tarjolla suurimmalla osalla ohjelmointijärjestelmistä (John & Tiegelkamp 2010, 100). Siemens-ohjelmointiympäristön vastaava kieli on STL (Statement List) (Siemens 2013).

```
LD      A
OR      B
AND     C
ST      X
```

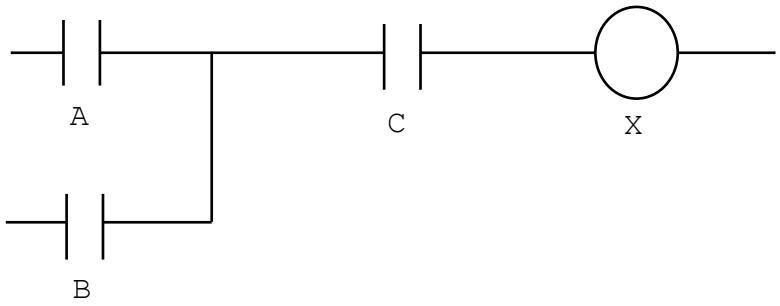
KUVIO 1. Käskylistaesimerkki

Käskylistaohjelmoinnissa toiminnot muodostetaan riveittäin siten, että vasemmalle tulee suoritettava toiminto ja oikealle muuttuja tai arvo. Ohjelmoitava lo-

giikka suorittaa ohjelmaa vasemmalta oikealle ja ylhäältä alas rivi kerrallaan. Esimerkkiohjelma (kuvio 1) aloitetaan lataamalla muuttuja `LD`-komennolla, minkä jälkeen tehdään ehdot ohjelman kululle ja siirretään lopputulos `ST`-komennolla muuttujaan `X`. Tämä on melko selkeä tapa esittää yksinkertaisia ohjelmia.

Tikapuukaavio

LD (Ladder Diagram) on graafinen ohjelmointikieli, minkä elementit muistuttavat releohjauksen geometriaa (kuvio 2). Tämä ohjelmointikieli tarjoaa selkeän käyttökokemuksen, sillä siinä kuvataan bool-muuttujia koskettimien avulla graafisesti (John & Tiegelkamp 2010, 147). Siemens-ohjelmointiympäristön vastaava kieli on LAD (Ladder Logic) (Siemens 2013).



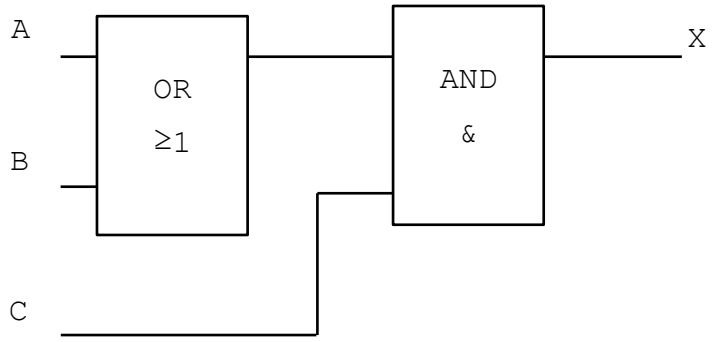
KUVIO 2. Tikapuukaavioesimerkki

Tikapuukaavio-ohjelmoinnissa toiminnot muodostetaan vaakatasossa siten, että vasemmalla on tulomuuttujat ja oikeassa päädyssä lähtömuuttujat. Ohjelmoitava logiikka suorittaa ohjelmaa vasemmalta oikealle ja ylhäältä alas rivi kerrallaan. Esimerkkiohjelmassa (kuvio 2) muuttujista muodostetaan kaavio, missä ehdot muodostetaan graafisesti. Tämä on erittäin selkeä ohjelmointikieli etenkin simuloituna, sillä silloin käyttäjä näkee selkeästi ohjelman kulun.

Toimilohkokaavio

FBD (Function Block Diagram) on graafinen ohjelmointikieli, missä elementit koostuvat toiminnallisista lohkoista (kuvio 3). Tämä kieli tarjoaa valmistajien suunnittelemat funktiot, mihin käyttäjä kytkee tarvittavat tulo- ja lähtöparametrit.

Tämän ansiosta monimutkaisetkin toiminnot ovat helposti käyttäjän ohjelmoitavissa. (John & Tiegelkamp 2010, 134.)



KUVIO 3. Toimilohkokaavio

Toimilohkokaavio-ohjelmoinnissa toiminnot muodostetaan vaakatasossa siten, että vasemmalla on tulomuuttujat, keskellä toimilohkot ja oikealla lähtömuuttujat. Ohjelmoitava logiikka suorittaa ohjelmaa vasemmalta oikealle ja ylhäältä alas rivi kerrallaan. Esimerkkiohjelmassa (kuvio 3) muodostetaan kaavio, missä muuttujat yhdistetään toimilohkoihin, joiden mukaan ehdot rakentuvat. Toimilohkojen lähtöpaikkaan sijoitetaan seuraava ehto tai ohjelman lähtöarvo. Tämä on erittäin selkeä ohjelmointikieli etenkin simuloituna, sillä silloin käyttäjä näkee selkeästi ohjelman kulun.

Rakenteellinen teksti

ST (Structured Text) on ohjelmointikieli, millä käyttäjä onnistuu rakentamaan monimutkaisia laskentatoimia ja ohjelmasilmukoita logiikalle (kuvio 4). Tyyliltään ST muistuttaa Pascal ja C ohjelmointikieliä. (John & Tiegelkamp 2010, 116.) Siemens-ohjelmointiympäristön vastaava kieli on SCL (Structured Control Language) (Siemens 2013).

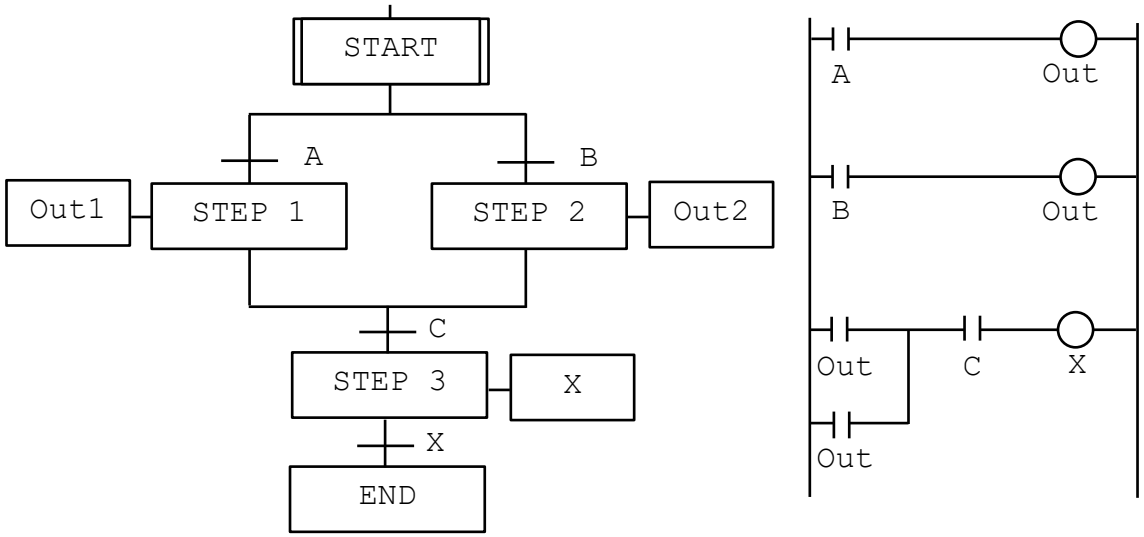
```
(A OR B)AND C :=X;
```

KUVIO 4. Rakenteellisen tekstikielen esimerkki

Rakenteellisessa tekstissä toiminnot muodostetaan vaakatasoon siten, että lausekkeessa on tulomuuttujat, ehdon muodostavat elementit, lähtömuuttujat ja puolipiste, joka päättää lauseen ja erottaa lausekkeet toisistaan. Ohjelmoitava logiikka suorittaa ohjelmaa vasemmalta oikealle ja ylhäältä alas rivi kerrallaan. Esimerkkiohjelmassa (kuvio 4) muuttujista muodostetaan lauseke, missä ehdot muodostetaan kirjoittamalla muuttujat ja komennot lauseeksi, mikä muistuttaa matemaattista kaavaa. Tämä ohjelmointikieli on hyvä monimutkaisissa ohjelmissa.

Sekvenssikaavio

SFC (Sequential Function Chart) on graafinen ohjelmointikieli, missä elementit rakennetaan sekvenssimuotoon (kuvio 5). Sekvenssi koostuu lohkoista, joiden toiminnot suoritetaan sekvenssin mukaisessa järjestyksessä kunkin ehdon täyttyessä. Tämä on selkeä tapa suorittaa samalla periaatteella toistuva ohjelma, sillä ohjelma toimii aina samalla tavalla askelma (step) kerrallaan. Ohjelman askeleiden toiminnot voidaan ohjelmoida millä tahansa edellä mainituilla kielillä sekä graafisesti, että tekstimuodossa. (John & Tiegelkamp 2010, 169.) Siemens-ohjelmointiympäristön vastaava kieli on S7-GRAPH (Siemens 2013).



KUVIO 5. Sekvenssikaavio

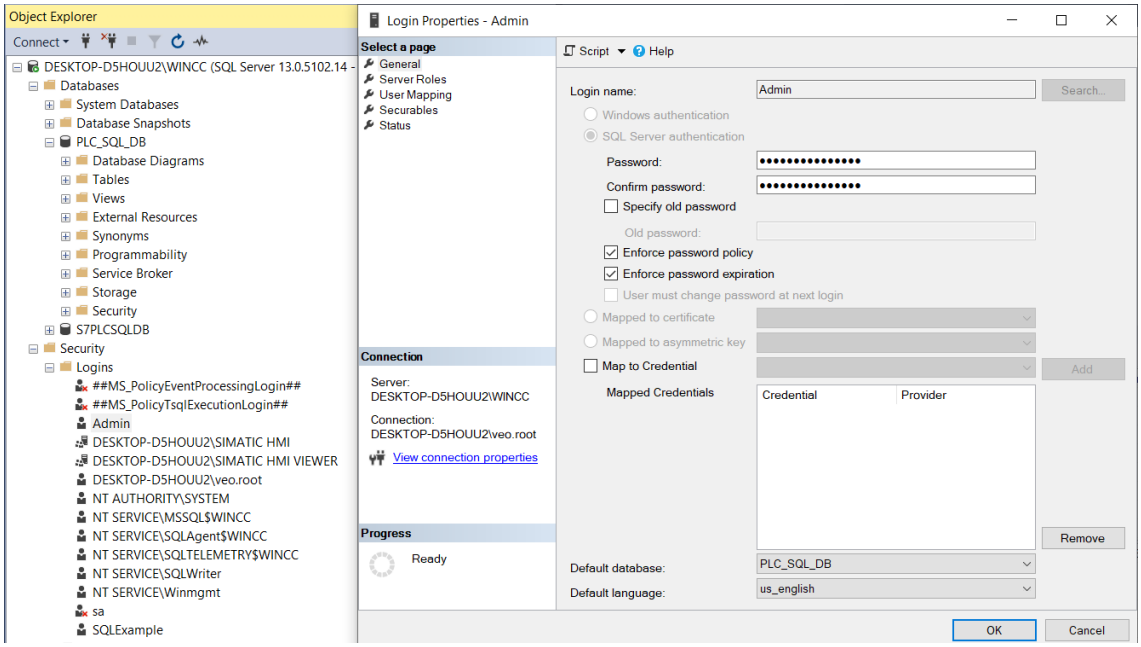
Sekvenssikaavio muodostetaan askelista, joissa jokaisella on ehto, millä ohjelma siirtyy seuraavaan askeleeseen. Sekvenssikaavio rakennetaan pystysuoraan siten, että sekvenssin aloittava askel on ylimpänä ja sekvenssin lopettava askel on alimpana. Alku- ja loppuaskelien väliin rakennetaan askeleet jokaiselle ohjelman vaiheelle. Ohjelmoitava logiikka suorittaa ohjelmaa ylhäältä alas askel kerrallaan. Esimerkkiohjelmassa (kuvio 5) oikealla puolella on tikapuukaaviolla tehty ohjelma, mikä on rakennettu sekvenssiohjelmaksi vasemmalle. Tämä on erittäin selkeä ohjelmointikieli etenkin simuloituna, sillä silloin käyttäjä näkee selkeästi ohjelman kulun.

5 TIEDONSIIRTOALUSTAN LUOMINEN

Tiedonsiirto tapahtuu relaatiotietokantaan, missä kyselyillä lisätään vesivoimalaitoksen prosessiarvoja tietokantaan tietyn syklin välein. Tiedonsiirto tapahtuu TIA Portal ohjelmointiympäristössä luodulla kyselyllä, joka lisää määriteltyyn tauluun tarvittavat prosessiarvot. Tätä varten luodaan tiedonsiirtoalusta, jonne prosessista saatu data tallennetaan.

5.1 Käyttäjän luominen

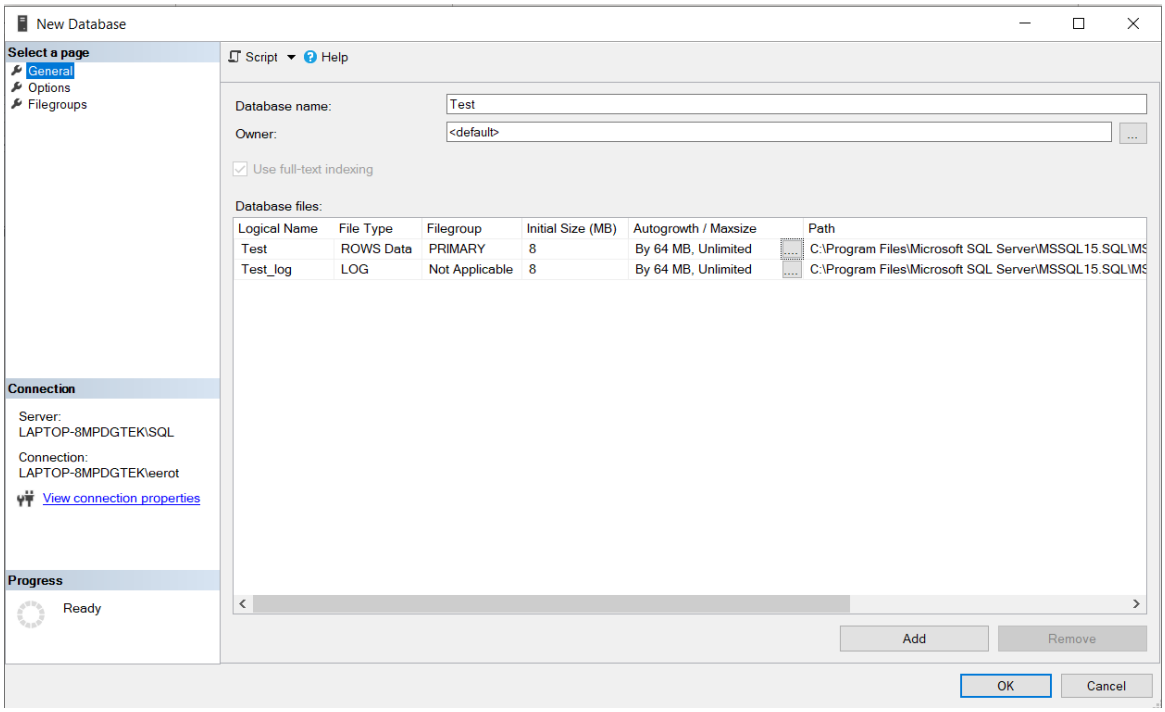
Käyttäjä luodaan SQL-palvelimen projektipuussa ”Object Explorer” painamalla ”security”-kansiossa hiiren kakkospainikkeella ja valitsemalla ”New login…”-toiminto. Käyttäjälle luodaan ”SQL Server authentication”, missä asetetaan nimi, salasana sekä oletustietokanta (kuva 1) (Microsoft 2016). Käyttäjätietoja tarvitaan TIA Portal -ohjelmointiympäristössä kirjautumisasetuksiin, minkä avulla ohjelmoitava logiikka saa väylän käyttäjäkohtaiseen tietokantaan (Siemens 2020, 8).



KUVA 1. Käyttäjän luominen

5.2 Tietokannan luominen

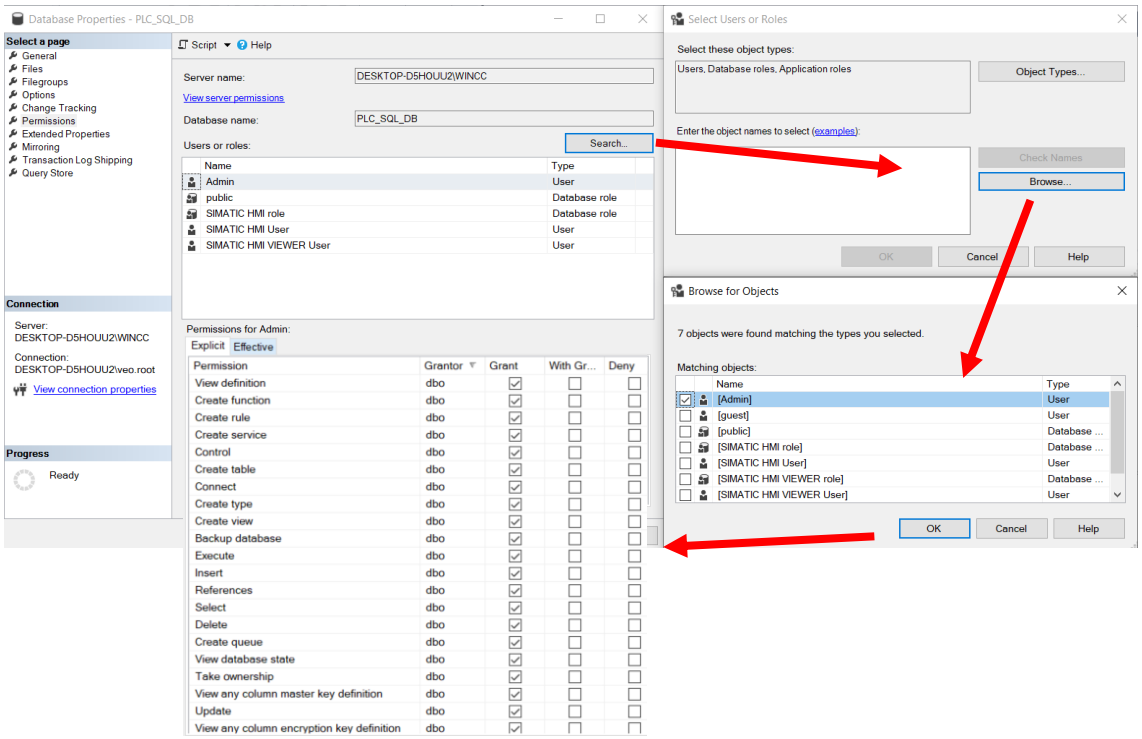
Prosessiarvoille täytyy luoda tietokanta (DB), mihin dataa siirretään. Tietokanta luodaan SQL-palvelimen projektipuussa painamalla ”Databases”-kansiossa hiiren kakkospainikkeella ja valitsemalla ”New database…”-toiminto. Tietokannalle määritellään tietokannan nimi, käyttäjä sekä muistiasetukset (kuva 2). Toinen tapa luoda tietokanta on luoda se manuaalisesti uudella kyselyllä. Tietokannan voi luoda komennolla `CREATE DATABASE Test`. (Oppel 2016, 44-46.) Tämä ei ole yksinkertaisin keino, sillä kaikki yksityiskohdat täytyy osata koodata kyselyyn oikein, kun taas ensimmäisellä keinolla tietokannan eri asetukset tulevat sovellusikkunaksi käyttäjän ruudulle.



KUVA 2. Tietokannan luominen käyttäjälle

Luodulle käyttäjälle ”Admin” (kuva 1) myönnetään käyttöoikeudet tietokantaan, jotta käyttäjä pääsee syöttämään tietoja ohjelmoitavalta logiikalta. Tämä tapahtuu tietokannan ominaisuuksissa, minne päästään painamalla tietokannan päältä hiiren kakkospainiketta ja valitsemalla listasta ”Properties”. Ominaisuusikkunasta valitaan ”Permissions”-välilehti, missä käyttäjä lisätään tietokantaan (kuva 3). Käyttäjätietojen päivityksen jälkeen käyttäjälle annetaan oikeudet käsitellä tieto-

kantaa. Tämä on yksinkertaisin keino myöntää oikeuksia käyttäjille, sillä sovellusikkunaan tulee kaikki käyttöoikeuksiin vaadittava tieto. Toinen keino myöntää käyttöoikeuksia on GRAND-komento.



KUVA 3. Käyttäjälle määritetyt oikeudet

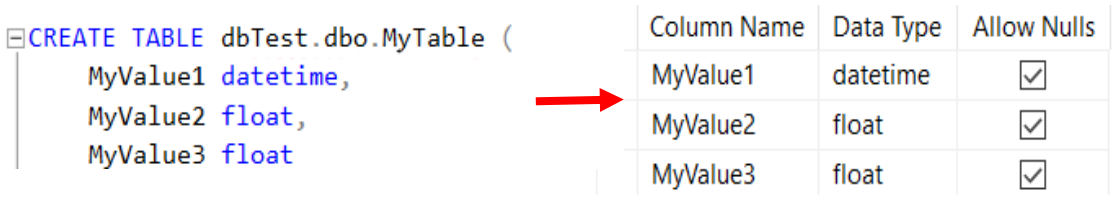
5.3 Hyödyllisiä komentoja

Tässä tapauksessa normaalin SQL-kyselykielen operoinnin rinnalle kyselyjä suoritetaan ohjelmoitavalle logiikalle ladatun sovelluksen avulla. Käyttäjä voi halutessaan luoda ja suorittaa kyselyitä tietokantaan, mutta tiedonsiirtoon käytettävät komennot on luotu TIA Portal -ympäristössä. Tällöin data siirtyy ohjelman mukaisella syklillä automaattisesti SQL-palvelimelle. Näin ollen käyttäjä voi tulostaa haluttujen taulujen datan milloin vain, kunhan tiedonsiirtolohko on käytössä.

Taulun luominen

Tiedonsiirtoa varten luodaan taulu (Table) tietokantaan (Database). Taulun voi luoda kirjoittamalla komentoriviin kyselyn (Query), jolla taulu luodaan. Kysely,

jolla taulun voi luoda on muotoa: `CREATE TABLE dbTest.dbo.MyTable (MyValue1 datetime, MyValue2 float, MyValue3 float)` (kuva 4). Taulun voi luoda myös painamalla ”table”-kansion päältä hiiren kakkospainikkeella ja valita luettelosta ”Table…”, jolloin taulu voidaan luoda manuaalisesti. (Oppel 2016, 326.)



KUVA 4. Komentorivi uuden taulun luomiseen

Datan lisääminen tauluun

Ohjelmoitavalle logiikalle tuleva data halutaan siirtää eteenpäin SQL-tietokantaan. Käytännössä TIA Portal -toimilohkot lähettävät tiedon komentorivinä, minkä mukaan SQL-palvelin vastaanottaa tiedot. Komentorivi on muotoa: ”`INSERT INTO MyTable VALUES( MyValue1, MyValue2, MyValue3)`”, missä ”MyTable” on esimerkkitaulu, johon tiedot siirretään ja ”MyValue1-3” on sarakeisiin lisättävä data (kuva 5). Komentorivin datan määrä ja datatyyppi tulee vastata luodun taulun sarakkeiden määrää ja datatyyppiä (kuva 4). Data erotellaan sarakkeisiin komentorivissä pilkuilla. (Oppel 2016, 328.)

```
/* Esimerkki: TIA Portal:n lähettämä komentorivi */
insert into MyTable Values (CURRENT_TIMESTAMP,1.23,4.56)
```

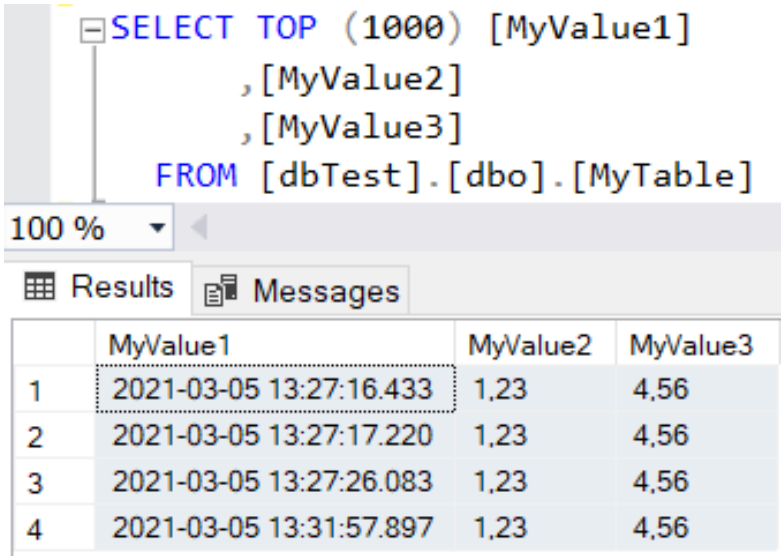
KUVA 5. Komentorivi datan lisäämiseen

Edellä mainittu komentorivi on merkittävin osa tiedonsiirtoa, sillä komento luodaan TIA Portal -ympäristössä ja siihen sisältyy kaikki tarvittava data prosessista.

Datan tulostus taulukoksi

Kun data on vastaanotettu SQL-palvelimelle, se tallentuu muistiin, jonka jälkeen käyttäjä voi tulostaa tietokannasta kyseisen taulun (table) sisällön SQL Server

Management Studiolla. Datan tulostus tehdään komennolla: `SELECT TOP (1000) [MyValue1] ,[MyValue2] ,[MyValue3] FROM [dbTest].[dbo]. [MyTable]` (kuva 6). Komentorivin sulkeissa oleva numero määrittää kuinka monta riviä tauluun tulostetaan. (Oppel 2016, 151.)



KUVA 6. Komentorivi datan tulostukseen

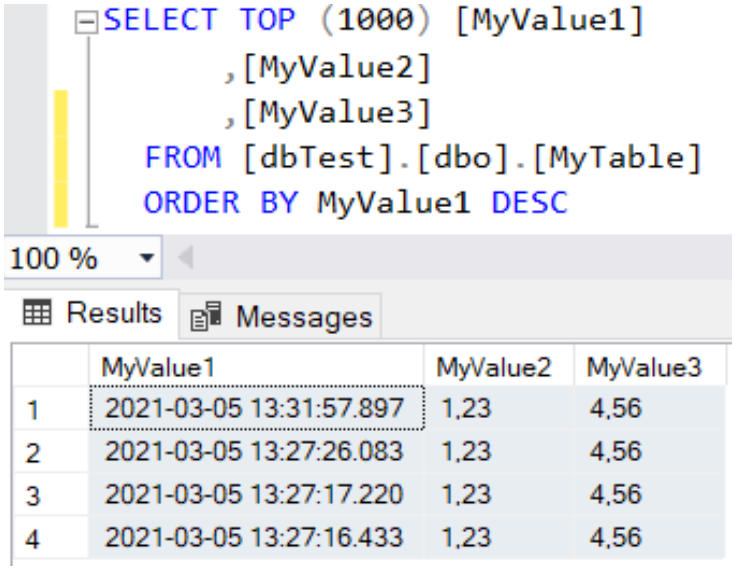
Tämän jälkeen käyttäjä voi suorittaa (execute) komentoriviä, jolloin data päivittyy SQL-palvelimelta tauluun. Jokaisen suorituksen jälkeen tauluun tulevat ne tiedot, jotka tietokannassa sillä hetkellä on.

SQL palvelimen aikaleimauskomennot

Prosessiteollisuudessa on tärkeää tietää, milloin prosessista tullut data on mitattu. SQL-kielessä tämän tiedon saa kirjoittamalla `CURRENT_TIMESTAMP`, jolloin SQL-tietokantaan tulee kyseisen komennon suoritushetken aikaleima. Aikaleima, joka tietokantaan tulee, on se aika, jolloin kyseinen suoritus on kirjattu SQL-palvelimelle. Tämä täytyy ottaa huomioon, mikäli aikaleima halutaan useamman millisekunnin tarkkuudella, sillä prosessin tiedonsiirrossa tapahtuu aina pientä viivettä. `CURRENT_TIMESTAMP` -aikaleiman datatyyppi on sille määritetyn sarakkeen datatyyppi. Työssä käytettävä datatyyppi on ”datetime”, jolloin aikaleima tulee muotoon: YYYY-MM-DD hh:mm:ss[.nnn], (esimerkiksi 2021-02-01 12:34:56.789). (Oppel 2016, 251.)

Datan järjestäminen

Tietokantaan tuleva data halutaan saada aikajärjestykseen siten, että viimeisin mittaus tulee taulun ensimmäiseksi arvoksi. Tämä tehdään siitä syystä, että taulussa olisi aina viimeisimmät prosessista saadut arvot. Ilman tätä esimerkkitauluun tulostuisi 1000 ensimmäistä arvoa, jonka jälkeen tauluun ei päivittyisi enää uusia arvoja. Kun aikajärjestys on käänteinen, tauluun tulostuu aina 1000 viimeisintä prosessiarvoa, ja näin ollen taulu pysyy ajan tasalla. Tämä onnistuu lisäämällä tulostuskomennon (kuva 6) perään komento `ORDER BY MyValue1 DESC` (kuva 7) (Oppel 2016, 173). Pääte `DESC` muuntaa valitun muuttujan arvojärjestyksen käänteiseksi.



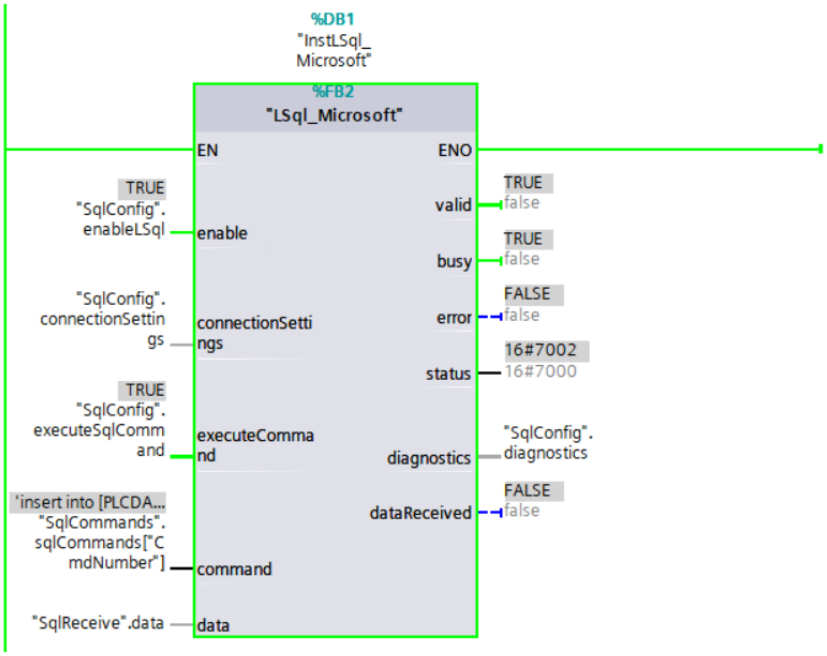
KUVA 7. Komentorivi järjestyksen muuttamiseen

6 TIEDONSIIRTO

TIA Portal-ohjelmointiympäristössä luodaan toimilohkot tiedonsiirtoa varten. Tämä toteutetaan niin, että SQL-palvelin pystyy vastaanottamaan tarvittavan datan. Työssä käyttöön saatiin Siemensin oma alustusohjelma, jossa tiedonsiirtoon tarvittavat lohkot sijaitsevat. Microsoft SQL-alustusohjelma koostuu ohjelmalohkokansiosta (Program blocks) ja PLC datatyyppikansiosta (PLC data types). Näissä kansioissa on tiedonsiirtoon vaadittavat toimi- (FB), data- (DB) sekä organisaatiolohkot (OB), jotka käyttäjä kopioi omaan projektiinsa. (Siemens 2020, 10.)

6.1 Tiedonsiirtolohko

Alustusohjelmassa on yksi toimilohko *LSql_Microsoft* (kuva 8), joka toimii koko ohjelman päälohkona organisaatiolohkossa (Main OB1) (Siemens 2020, 7). Se on rakennettu toimimaan yhteydessä SQL-palvelimen kanssa ja sisältää kaikki tiedonsiirtoon vaadittavat elementit. Tässä opinnäytetyössä ei avata lohkoa kokonaisuudessaan, vaan vain siltä osin, mitä käyttäjän täytyy tietää lohkon toiminnasta.



KUVA 8. *LSql_Microsoft* -tiedonsiirtotoimilohko (Siemens 2020, muokattu)

Toimilohkossa on datan lähettämistä ja vastaanottamista varten tulo- ja lähtöparametreja. Datan vastaanottamista SQL-palvelimelta ei tässä opinnäytetyössä tarvita, joten datan vastaanottamiseen liittyviä asioita ei tässä työssä käsitellä. Toimilohkon toimintaan ja tiedonsiirtoon tarvittavat parametrit ja niiden datatyypit ovat: *enable (Bool), ConnectionSettings (LSql_typeConnectionSettings), executeCommand (Bool), command (String), valid (Bool), busy (Bool), error (Bool), status (Word) ja diagnostics (LSql_typeDiagnostics)* (taulukko 1) (Siemens 2020, 7).

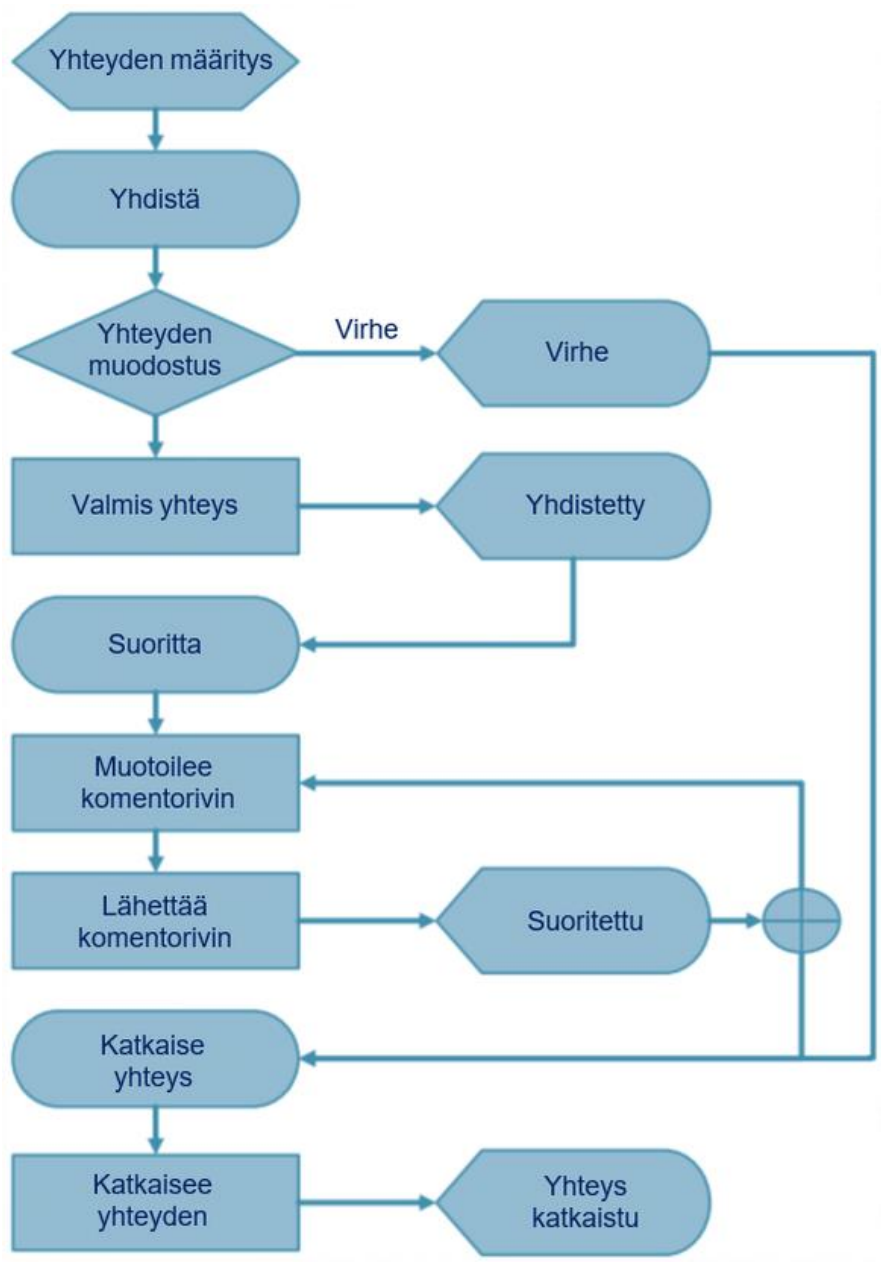
TAULUKKO 1. Tiedonsiirtolohkon signaalit ja niiden merkitys (Siemens 2020, 7).

Nimi	Selite
enable	Toimilohkon käyttöönotto
connectionSettings	Kirjautuminen tietokantaan
executeCommand	Suorittaa komennon tietokantaan kerran
command	Komentorivi, mikä tietokantaan suoritetaan
valid	Lähtöarvot käyttökelpoisia
busy	Toimilohkon valmiustila
error	Virhe suoritettaessa
status	Tila-/virhekoodi
diagnostics	Diagnostiikkatiedot toimilohkosta

Käyttäjälle merkittäviä parametreja ovat *enable, executeCommand, command, error* ja *status*. *Enable* määrittää toimilohkon käytön, *executecommand* määrittää datan lähettämisen ja komentorivin suorittamisen SQL-palvelimelle, *command* näyttää lähetettävän komentorivin, *error* määrittää toimilohkon virheen ja *status* virheen tilatiedon.

Toimilohkon (kuva 8) toiminta perustuu komentorivin lähettämisestä ja suorittamisesta SQL-palvelimella. Tämä tapahtuu yksinkertaisimmillaan siten, että ohjelma on ladattu onnistuneesti PLC:lle ja toimilohkon *enable*-parametri on tilassa 1 (true), jolloin toimilohko on käytettävissä. Mikäli *error*-parametri on tilassa 0 (false), on toimilohkon toiminta ja yhteys SQL-plavelimelle virheetön. Tämän jälkeen käyttäjä syöttää *command*-parametriin tietokantaan syötettävän komentoriviin ja lähettää sen SQL-palvelimelle asettamalla *executeCommand*-parametrin tilatiedon arvoon 1 (true). (Siemens 2020, 5.)

Tiedonsiirrossa toimilohko käyttää aina samaa toimintaperiaatetta (kuva 9). Ensimmäisenä SQL palvelimen ja toimilohkon välille muodostetaan yhteys. Mikäli yhteyden muodostus ei onnistu tapahtuu virhe (*error*) ja toiminto keskeytetään. Onnistuneen yhteyden muodostuksen jälkeen toimilohkolla voidaan lähettää dataa tietokantaan suorittamalla komentoriviä. Komentoriviä voidaan suorittaa niin kauan, kunnes yhteys katkaistaan tai toimilohkossa tapahtuu jokin virhe. (Siemens 2020, 5.)



KUVA 9. Tiedonsiirron toimintaperiaate (Siemens 2020, 5)

6.2 Asetuksien datalohko

SqlConfig-datalohkossa määritellään kaikki tarvittavat tiedot tiedonsiirtoa varten. Datalohkoon asetetaan kommunikaatioasetukset, sisäänkirjautumistiedot ja sieltä nähdään toimilohkon parametrien eri tila- ja statustietoja (kuva 10).

SqlConfig

	Name	Data type	Start value	Monitor value
1	Static			
2	connectionSettings	"LSql_typeConn...		
3	interfaceSettings	TCON_IP_v4		
4	InterfaceId	HW_ANY	64	64
5	ID	CONN_OUC	16#10	16#0010
6	ConnectionType	Byte	16#0B	16#0B
7	ActiveEstablish...	Bool	true	TRUE
8	RemoteAddress	IP_V4		
9	RemotePort	UInt	1433	1433
10	LocalPort	UInt	0	0
11	loginInformation	"LSql_typeLoginInf...		
12	hostName	String	''	''
13	userName	String	'Admin'	'Admin'
14	password	String	'Admin'	'Admin'
15	appName	String	''	''
16	serverName	String	'DESKTOP-D5H...	'DESKTOP-D5HOUU2\WINCC'
17	libraryName	String	''	''
18	local	String	''	''
19	databaseName	String	'PLC_SQL_DB'	'PLC_SQL_DB'
20	sspi	String	''	''
21	attachDbfile	String	''	''
22	changePassword	String	''	''
23	enableLSql	Bool	false	FALSE
24	diagnostics	"LSql_typeDiagnost...		
25	status	Word	16#0	16#0000
26	subfunctionStatus	Word	16#0	16#0000
27	stateNumber	DInt	0	0
28	executeSqlCommand	Bool	false	FALSE

KUVA 10. *SqlConfig*-datalohko (Siemens 2020, 12, muokattu)

Sisäänkirjautumistietoihin kirjataan halutun palvelimen ja tietokannan tiedot, mihin dataa halutaan siirtää. Pakolliset tiedot ovat: käyttäjätunnus (*userName*), salasana (*password*), palvelimen nimi (*serverName*) ja tietokannan nimi (*databaseName*). Käyttöliittymätiedot ovat oletusasetuksina kuten portti 1344, joka on SQL-palvelimen oletusportti. (Siemens 2020, 13.)

6.3 Komentorivien datalohko

SqlCommands-lohkossa määritetään SQL-palvelimelle lähetettävä komentorivi (kuva 11). Alustusohjelmassa komennoille on luotu kuuden rivin komentojoukko, missä jokaiselle riville voidaan määritellä komentorivi. Komentorivi on merkkijonomuodossa ja dataa lähetettäessä tiettyyn tauluun, täytyy komentorivin vastata kyseisen taulun tietoja. Vaadittavia tietoja ovat nimi, sarakkeiden määrä ja datatyyppi. Sarakkeet erotellaan komentorivissä pilkuilla ja niiden tulee vastata tietokantaan luodun taulun sarakkeiden lukumäärää. Myös komentorivin arvot täytyy vastata luodun taulun datatyyppejä, että tiedonsiirto onnistuu.

SqlCommands

	Name	Data type	Start ...	Monitor value
1	Static			
2	sqlCommands	Array[0..5] of String		
3	sqlCommands[0]	String	'insert...	''
4	sqlCommands[1]	String	'Upda...	'insert into PLCDATA_1 values (CURRENT_TIMESTAMP, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100)'
5	sqlCommands[2]	String	'Selec...	'insert into PLCDATA_2 values (CURRENT_TIMESTAMP, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200)'
6	sqlCommands[3]	String	'Selec...	'insert into PLCDATA_3 values (CURRENT_TIMESTAMP, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300)'
7	sqlCommands[4]	String	'Selec...	'insert into PLCDATA_4 values (CURRENT_TIMESTAMP, 310, 320, 330, 340, 350, 360, 370, 380, 390, 400)'
8	sqlCommands[5]	String	'insert...	'insert into PLCDATA_5 values (CURRENT_TIMESTAMP, 410, 420, 430, 440, 450, 460, 470, 480, 490, 500)'

KUVA 11. *SqlCommands*-datalohko (Siemens 2020, 15, muokattu)

Lsql Microsoft -toimilohko pystyy lähettämään yhden komentorivin ohjelmakierroksella, joten ohjelmassa määritellään, mikä rivi komentojoukosta suoritetaan tietokantaan. Määritys tapahtuu valitsemalla *lsql Microsoft* -toimilohkon *command*-tulopaikkaan halutun komentorivin nimike (TAG).

7 DATANKÄSITTELYOHJELMA

Lsql Microsoft -toimilohkon rinnalle täytyy luoda komentorivin muodostamista varten omat lohkot, jotta tiedonsiirrosta saadaan automaattinen. Ilman datankäsittelytoimilohkoja käyttäjän tulisi itse kirjoittaa haluamansa tiedot tietokantaan. Työssä halutaan saada prosessiarvot helposti siirrettävään muotoon ja siirretyksi. Tämän toteuttamiseksi työssä täytyy luoda datankäsittelyohjelma, joka koostuu muunnos- ja yhdistämistoimilohkoista.

7.1 Prosessiarvojen datalohko

Työn yhtenä tavoitteena on selvittää tiedonsiirtomääriä SQL-tietokantaan. Prosessidatan hallinnoinnin tueksi luotiin datalohko, mihin kaikki halutut prosessiarvot sijoitetaan (kuva 12). Prosessiarvot ovat kokonaislukumuodossa selkeyttääkseen ja helpottaakseen tiedonsiirtoa. Prosessiarvot ovat yleisesti reaalilukuina, mutta se aiheuttaa ongelmia tiedonsiirtoon, sillä datanmuunnoksessa reaaliluvun desimaaliksi jakava piste muuttuu muunnoksen seurauksena pilkuksi, mikä sekoittaa komentorivin sarakkeiden paikat. Tämän takia päätettiin, että esimerkkiprosessin arvot syötetään kokonaislukuina, minkä jälkeen ongelmia ei ilmennyt.

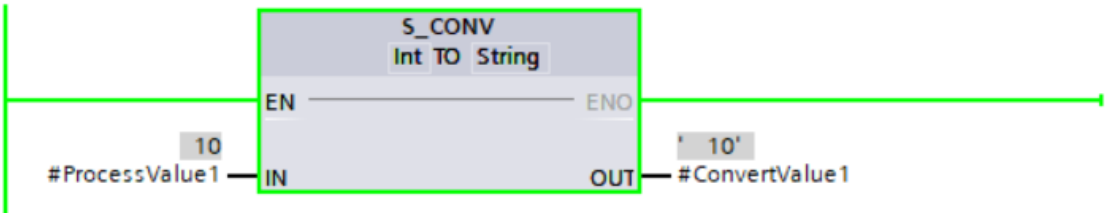
ProcessValues

	Name	Data type	Start value	Monitor value	Retain
1	Static				
2	ProcessValue	Array[1..50] of Int			
3	ProcessValue[1]	Int	0	10	
4	ProcessValue[2]	Int	0	20	
5	ProcessValue[3]	Int	0	30	
6	ProcessValue[4]	Int	0	40	
7	ProcessValue[5]	Int	0	50	
8	ProcessValue[6]	Int	0	60	
9	ProcessValue[7]	Int	0	70	
10	ProcessValue[8]	Int	0	80	
11	ProcessValue[9]	Int	0	90	
12	ProcessValue[10]	Int	0	100	
13	ProcessValue[11]	Int	0	110	
14	ProcessValue[12]	Int	0	120	
15	ProcessValue[13]	Int	0	130	
16	ProcessValue[14]	Int	0	140	
17	ProcessValue[15]	Int	0	150	
18	ProcessValue[16]	Int	0	160	
19	ProcessValue[17]	Int	0	170	
20	ProcessValue[18]	Int	0	180	
21	ProcessValue[19]	Int	0	190	
22	ProcessValue[20]	Int	0	200	
23	ProcessValue[21]	Int	0	210	
24	ProcessValue[22]	Int	0	220	
25	ProcessValue[23]	Int	0	230	
26	ProcessValue[24]	Int	0	240	
27	ProcessValue[25]	Int	0	250	
28	ProcessValue[26]	Int	0	260	
29	ProcessValue[27]	Int	0	270	

KUVA 12. Esimerkkiprosessin datalohko

7.2 Prosessiarvot merkkijonoksi

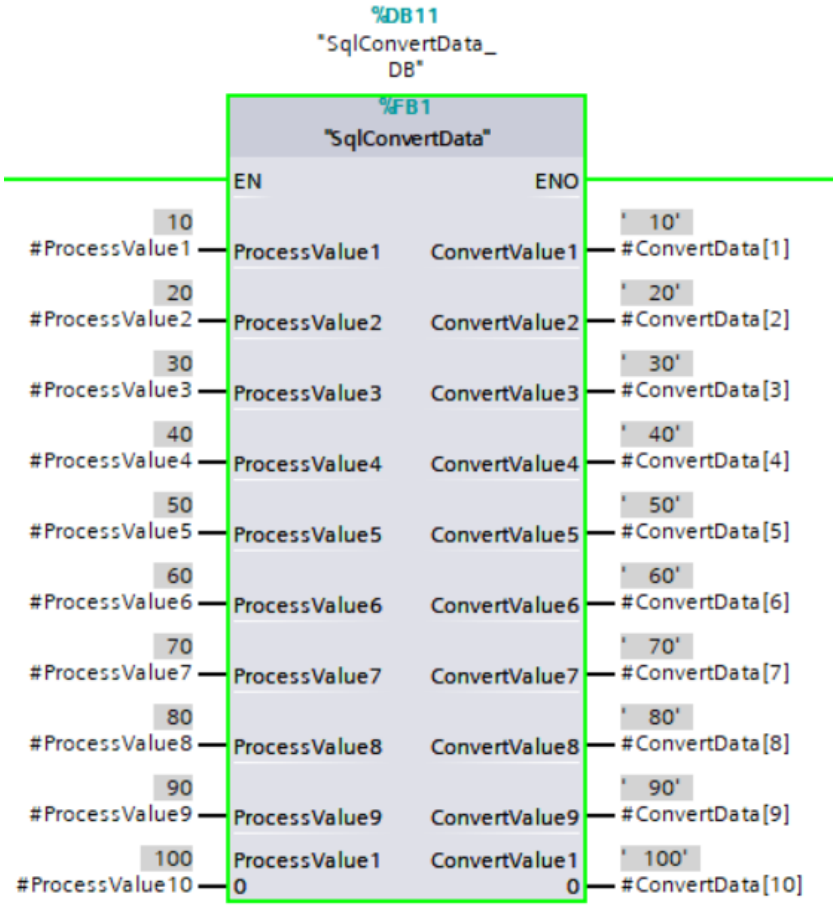
Lsql Microsoft -toimilohkoa käytettäessä prosessista vastaanotettava data on kokonaisluku muodossa (*Int*), joten se täytyy muuntaa merkkijonomuotoon (*String*). Jokainen prosessista tuleva arvo täytyy muuntaa merkkijonomuotoon, jotta komentorivi muodostuu oikein. Muunto tapahtuu *S_CONV*-lohkolla, missä tulopaikkaan laitetaan prosessista tuleva kokonaislukumuodossa oleva arvo ja lähtöpuolelle lohko muuntaa kyseisen arvon merkkijonoksi (kuva 13) (John & Tiegelkamp 2010, 216). Merkkijonomuodossa olevan arvon tunnistaa suorasta heittomerkeistä arvon molemmin puolin.



KUVA 13. Datan muuntaminen kokonaisluvusta merkkijonoksi

S_CONV-lohkoista luodaan toimilohko (kuva 14), millä pystytään muuntamaan kerrallaan kymmentä prosessiarvoa. Tämä tehdään siitä syystä, että saadaan luotettavasti prosessiarvot tietokantaan.

Työn tavoitteena oli saada useita kymmeniä prosessiarvoja tietokantaan. Se osoittautui mahdottomaksi yhdellä toimilohkolla, sillä Siemensin merkkijonodatatyypin maksimi merkkimäärä on 256. Tämän vuoksi tietokantaan täytyy luoda useampi taulu, mihin dataa siirretään useammalla toimilohkolla. Työssä päädyttiin käyttämään viittä eri taulua, joille jokaiselle siirretään aikaleiman lisäksi kymmenen prosessiarvoa. Näin ollen päästään 50 prosessiarvon suuruiseen tietokantaan. Tällä tyylillä tauluja ja toimilohkoja voidaan tehdä käyttäjän tarpeiden mukaan niin paljon kuin on tarve. Täytyy muistaa, että *Lsql Microsoft* -toimilohko voi suorittaa vain yhtä komentoriviä kerrallaan. Tämä aiheuttaa sen, että useita kymmeniä prosessiarvoja ei voida siirtää tietokantaan yhdellä ohjelmakierroksella, vaan kierroksia tarvitaan niin monta kuin on tauluja, mihin dataa siirretään.



KUVA 14. Lopullinen datan muuntolohko

Toimilohkolle luotiin tulo- ja lähtöpaikat kymmenelle prosessiarvolle sekä muunnetuille arvoille, joten käyttäjän on helppo sijoittaa prosessiarvot haluamalleen paikalle toimilohkossa (kuva 15).

SqlConvertData_DB

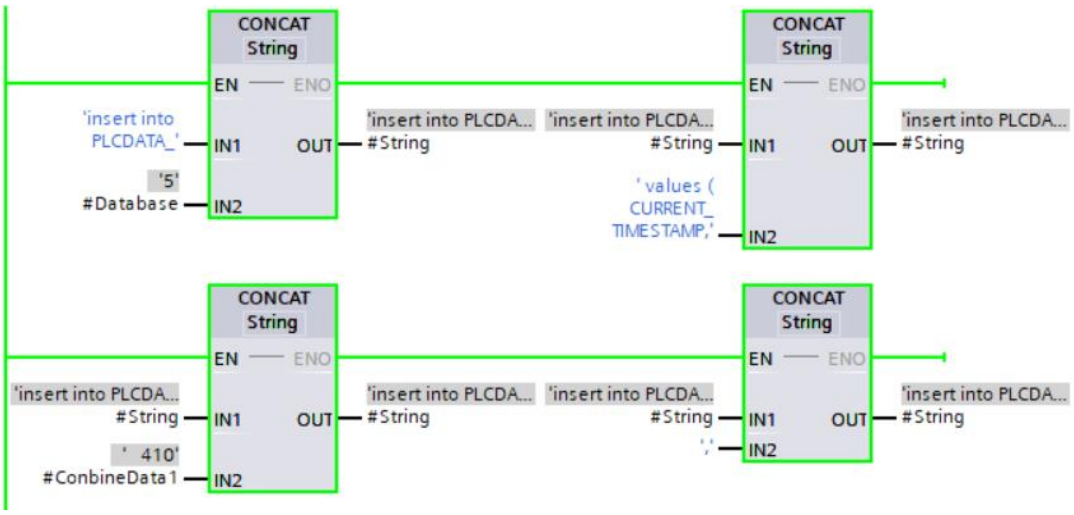
	Name	Data type	Start value	Monitor value
1	Input			
2	ProcessValue1	Int	0	410
3	ProcessValue2	Int	0	420
4	ProcessValue3	Int	0	430
5	ProcessValue4	Int	0	440
6	ProcessValue5	Int	0	450
7	ProcessValue6	Int	0	460
8	ProcessValue7	Int	0	470
9	ProcessValue8	Int	0	480
10	ProcessValue9	Int	0	490
11	ProcessValue10	Int	0	500
12	Output			
13	ConvertValue1	String	''	' 410'
14	ConvertValue2	String	''	' 420'
15	ConvertValue3	String	''	' 430'
16	ConvertValue4	String	''	' 440'
17	ConvertValue5	String	''	' 450'
18	ConvertValue6	String	''	' 460'
19	ConvertValue7	String	''	' 470'
20	ConvertValue8	String	''	' 480'
21	ConvertValue9	String	''	' 490'
22	ConvertValue10	String	''	' 500'

KUVA 15. Muuntolohkon tulo- ja lähtöpaikat

7.3 Merkkijonojen yhdistäminen

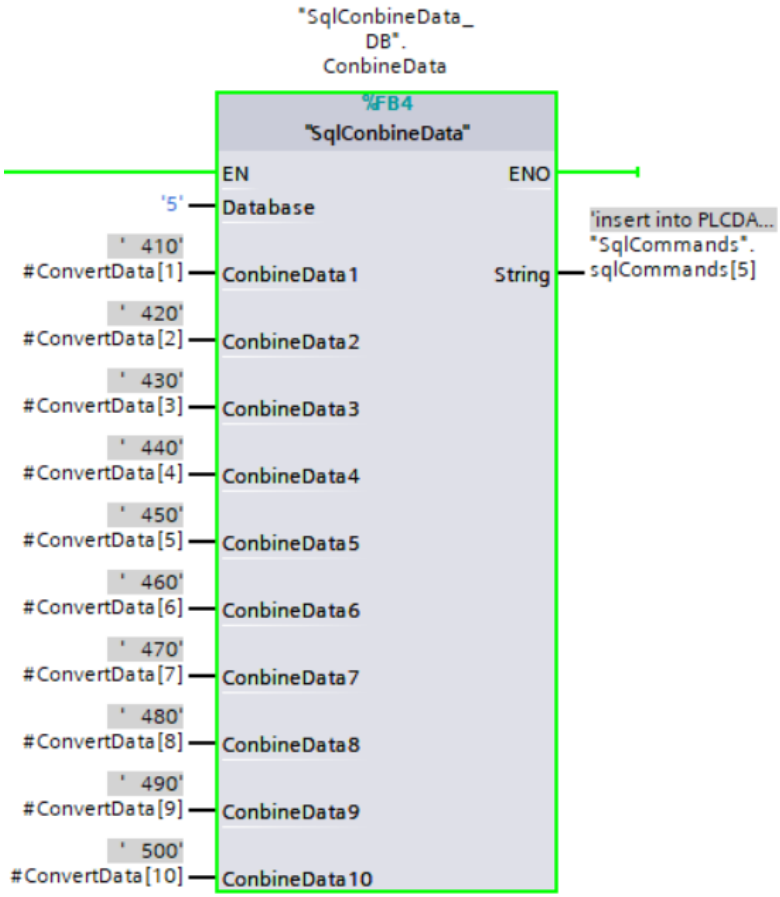
Lopullinen komentorivi *Lsql Mmicrosoft* -toimilohkolle luodaan *CONCAT*-lohkolla, missä muunnetut prosessiarvot yhdistetään osaksi komentoriviä (kuva 16). *CONCAT*-lohkossa on kaksi tulopaikkaaja *(IN1* ja *IN2*), joista *IN1* ja *OUT* laitetaan muodostuvan komentorivin nimike ja *IN2* komentoriviin lisättävän merkkijono tai merkkijonon nimike (John & Tiegelkamp 2010, 320).

Opinnäytetyössä komentorivin muodostus aloitetaan komennolla ”*insert into*”, mikä lisää dataa tauluun. Tietokantaan on luotu viisi eri taulua, mihin dataa lisätään (*PLCDATA_1…PLCDATA_5*), joten käyttäjän täytyy päättää mihin tauluun data lisätään. Tämä tehdään asettamalla halutun taulun numero (1-5) lähtöpaikkaan ”*Database*”. Komento jatkuu arvojen kirjauksella, mikä alkaa lauseella ”*values (CURRENT_TIMESTAMP,*”, missä ensimmäinen arvo on tiedonsiirron aikaleima. Tämän jälkeen ohjelma laittaa prosessiarvot samaan järjestykseen allekkain kuin tietokantaan luodun taulun sarakkeet. Näin ollen prosessiarvot tulevat niille määritetyille paikoille tauluun.



KUVA 16. Merkkijonojen yhdistämisen toimilohkoja

CONCAT-lohkoista luodaan toimilohko (kuva 17), missä tulopaikkoina on tietokannan numero sekä muunnetut prosessiarvot ja lähtöpaikkana *sqlCommands*-datalohkon nimike.



KUVA 17. Lopullinen yhdistämislohko

Yhdistämislohkon tulopaikoille sijoitetaan muunnoslohkon lähtöpaikkojen nimikkeet (kuva 14). Tällöin prosessiarvot tulevat siihen järjestykseen, mihin ne on sijoitettu muunnoslohkossa. Tietokantaan prosessiarvot tulevat tässä järjestyksessä, joten jokaisen taulun ensimmäinen arvo on aikaleima. Lopullinen komentorivi muodostuu toimilohkon lähtöpaikkaan *String*, minkä *Lsql Microsoft* -toimilohko lähettää SQL-palvelimelle ja sitä kautta tietokantaan (kuva 18).

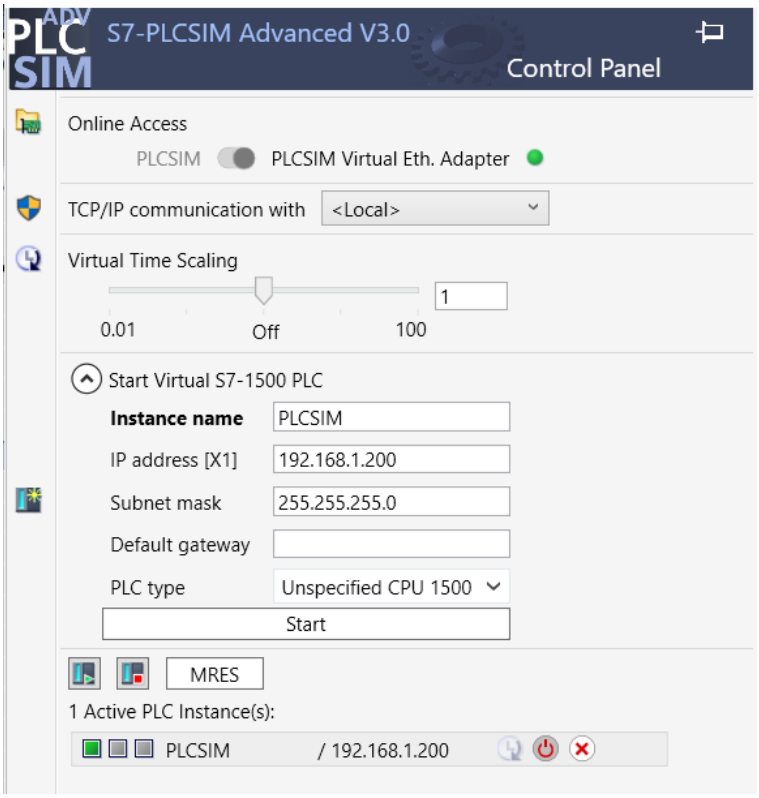
SqlConbineData_DB

	Name	Data ..	...	Monitor value
1	Input			
2	Output			
3	InOut			
4	Static			
5	ConbineData	"SqlC...		
6	Input			
7	Database	String	''	'5'
8	ConbineData1	String	''	' 410'
9	ConbineData2	String	''	' 420'
10	ConbineData3	String	''	' 430'
11	ConbineData4	String	''	' 440'
12	ConbineData5	String	''	' 450'
13	ConbineData6	String	''	' 460'
14	ConbineData7	String	''	' 470'
15	ConbineData8	String	''	' 480'
16	ConbineData9	String	''	' 490'
17	ConbineData10	String	''	' 500'
18	Output			
19	String	String	''	'insert into PLCDATA_5 values (CURRENT_TIMESTAMP, 410, 420, 430, 440, 450, 460, 470, 480, 490, 500)'

KUVA 18. Yhdistämislohkon tulo- ja lähtöpaikat

8 TOIMINNAN TESTAUS

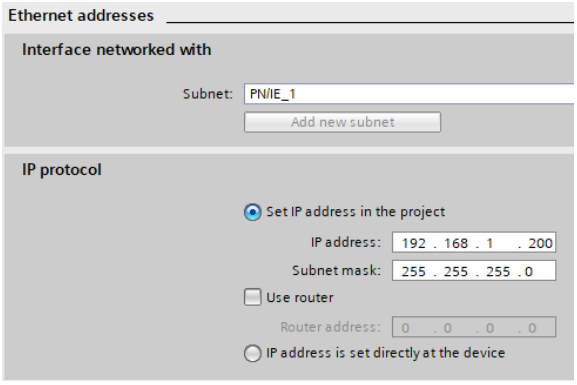
Toiminnan testaus suoritetaan simulointiympäristössä, jossa valmis ohjelma ladataan virtuaaliseen ohjelmoitavaan logiikkaan (PLC). Käytössä on Siemensin 27-PLCSIM Advanced V3.0 -simulointiympäristö (kuva 19), missä käyttäjä pystyy simuloimaan ohjelmoitavaa logiikkaa omalta tietokoneeltaan. Tämä helpottaa sovellustason testaamista, sillä todellista ohjelmoitavaa logiikkaa ei tarvita. Tässä täytyy kuitenkin huomioida todellisen ja virtuaalisen PLC:n erot, sillä esimerkiksi suorituskyky laitteistojen välillä voi olla suuri.



KUVA 19. PLCSIM-simulointiympäristö (Siemens 2019b, 9, muokattu)

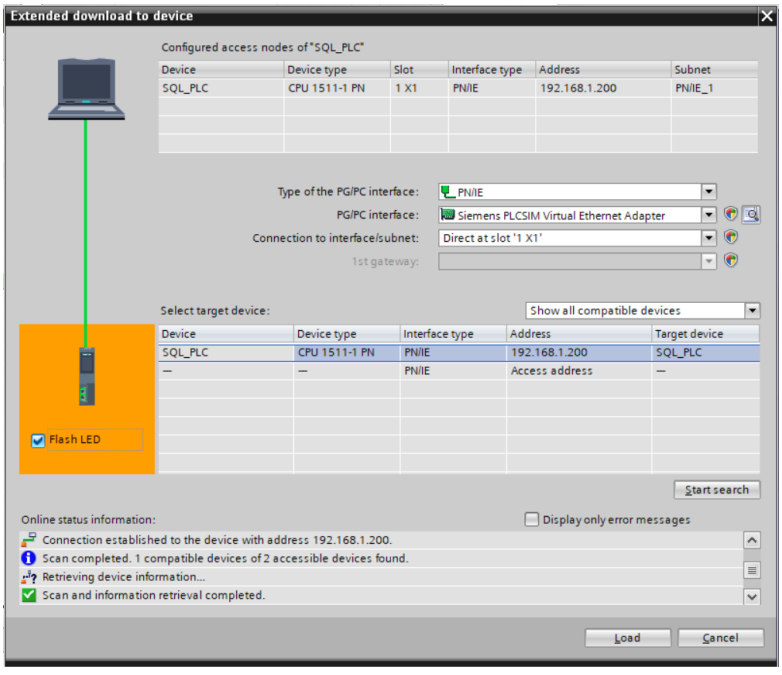
Simulointiympäristössä luodaan esimerkkiprosessille oma PLC, mille rakennetut ohjelmat ladataan ja toimintaa testataan. Simuloinnissa käytetään PLCSIM Virtual Eth. -adapteria, jolloin PLC päästään luomaan. PLC:lle annetaan nimi, IP-osoite, aliverkon peite sekä PLC tyyppi (kuva 19) (Siemens 2019b, 40).

TIA Portal -ympäristössä Ethernet-asetukset täytyy vastata PLCSIM-ympäristöön luodun PLC kanssa. Esimerkkiprosessin PLC:n IP-osoite on 192.168.1.200, joten suorittimen (CPU) Ethernet-kortin IP-osoite täytyy olla vastaava (kuva 20).



KUVA 20. Ethernet kortin asetukset

Verkkoasetusten ollessa oikein voidaan rakennettu ohjelma ladata PLC:lle (kuva 21). Ensimmäisellä latauskerralla muodostetaan yhteys haluttuun PLC:hen, jolloin täytyy valita oikeat käyttöliittymätyypit. Käyttöliittymätyyppinä esimerkkiprosessissa on Profinet/Industrial Ethernet (PN/IE) ja käyttöliittymänä sama kuin simulointiympäristössä luodun PLC:n, eli Siemens PLCSIM Virtual Ethernet Adapter. Tämän jälkeen painetaan ”Start search” -painiketta, jolloin luotu PLC tulee valittavaksi. Ladataan ohjelma valitsemalla luotu PLC ja painamalla ”Load”. Kun ohjelma on ladattu PLC:lle, voidaan alkaa testaamaan ohjelman toimintaa.



KUVA 21. Yhdistäminen PLC:hen ja ohjelman lataus

Testauksen tarkoituksena on saada PLC:lle määritetyt arvot siirrettyä SQL-palvelimelle ja sitä kautta tietokantatyökalulle. Tämän jälkeen tietokantatyökalulla (SQL Server Management Studio) tulostetaan simuloidut arvot niille määritettyihin tauluihin. Testauksen tavoitteena on selvittää, onko datansiirto riittävällä tasolla.

Testauksessa annetaan prosessiarvot manuaalisesti siten, että käyttäjän on helppo nähdä tulevatko prosessiarvot niille määritetyille paikoille tauluihin. Tässä annetaan 50 eri prosessiarvoa 10…500, eli kymmenen yksikön välein (kuva 11). Testauksen ajaksi rakennetaan alusta, mikä suorittaa komentorivejä SQL-palvelimelle. Ohjelmassa ohjelmakierto kestää 20 ms ja jokaisella ohjelma kierrolla vaihdetaan taulua, mihin dataa siirretään. Näin ollen saadaan hyödynnettyä useampaa taulua ja päästään suurempiin datansiirtomääriin. Tämän jälkeen ohjelma ladataan PLC:lle ja aloitetaan testaus. tiedonsiirto tapahtuu laittamalla *enable*-parametri tilaan 1, jolloin testiohjelma alkaa suorittaa komentorivejä. Tämän jälkeen voidaan avata tietokantatyökalu, josta nähdään työn lopputulos (kuva 22).

Results | Messages

	Time_Stamp	Value_1	Value_2	Value_3	Value_4	Value_5	Value_6	Value_7	Value_8	Value_9	Value_10
1	2021-04-26 13:23:50.327	10	20	30	40	50	60	70	80	90	100
2	2021-04-26 13:23:50.180	10	20	30	40	50	60	70	80	90	100
3	2021-04-26 13:23:50.050	10	20	30	40	50	60	70	80	90	100

	Time_Stamp	Value_11	Value_12	Value_13	Value_14	Value_15	Value_16	Value_17	Value_18	Value_19	Value_20
1	2021-04-26 13:23:50.343	110	120	130	140	150	160	170	180	190	200
2	2021-04-26 13:23:50.200	110	120	130	140	150	160	170	180	190	200
3	2021-04-26 13:23:50.070	110	120	130	140	150	160	170	180	190	200

	Time_Stamp	Value_21	Value_22	Value_23	Value_24	Value_25	Value_26	Value_27	Value_28	Value_29	Value_30
1	2021-04-26 13:23:50.367	210	220	230	240	250	260	270	280	290	300
2	2021-04-26 13:23:50.230	210	220	230	240	250	260	270	280	290	300
3	2021-04-26 13:23:50.097	210	220	230	240	250	260	270	280	290	300

	Time_Stamp	Value_31	Value_32	Value_33	Value_34	Value_35	Value_36	Value_37	Value_38	Value_39	Value_40
1	2021-04-26 13:23:50.387	310	320	330	340	350	360	370	380	390	400
2	2021-04-26 13:23:50.253	310	320	330	340	350	360	370	380	390	400
3	2021-04-26 13:23:50.110	310	320	330	340	350	360	370	380	390	400

	Time_Stamp	Value_41	Value_42	Value_43	Value_44	Value_45	Value_46	Value_47	Value_48	Value_49	Value_50
1	2021-04-26 13:23:50.407	410	420	430	440	450	460	470	480	490	500
2	2021-04-26 13:23:50.273	410	420	430	440	450	460	470	480	490	500
3	2021-04-26 13:23:50.130	410	420	430	440	450	460	470	480	490	500

Query executed successfully. | DESKTOP-D5HOUU2\WINCC (13.0... | DESKTOP-D5HOUU2\veo.ro... | PLC_SQL_DB

KUVA 22. Tulostettu data

Kuten kuvasta nähdään, datansiirto onnistui moitteettomasti, sillä yhteys PLC:hen onnistuu, jokainen prosessiarvo on omalla paikallaan ja datansiirtonopeus on riittävällä tasolla.

9 POHDINTA

Opinnäytetyö tehtiin yhteistyössä VEO Oy:n kanssa. Työn tavoite oli selvittää, miten prosessista saadaan siirrettyä dataa SQL-tietokantaan ja ovatko siirtoon käytettävät sovellukset ja ohjelmat käytännöllisiä. Datan keräykseen suunniteltujen ratkaisujen täytyy toimia siten, että automaatiojärjestelmästä siirretty data olisi reaaliaikaista ja keräys suoritettaisiin luotettavasti. Työn lopputulosta yritys voi soveltaa ja kehittää esimerkiksi takuuaikaisten tuotteiden seurantaan ja raportointiin.

Työ rakennettiin kahteen erilliseen sovellustasoon: SQL-tietokantaan ja logiikkaohjelmaan. Käytössä olleet sovellukset olivat Microsoft SQL Serverin -tietokantasovellukset ja Siemensin ohjelmointisovellukset. Microsoftin tarjoamat SQL-sovellukset ovat SQL Server, eli tietokantapalvelin sekä SQL Server Management Studio, eli tietokantatyökalu. Microsoft SQL Server käyttää interaktiivista SQL:ää, mikä soveltuu hyvin tähän työhön, sillä käyttäjä pääsee suoraan sovellusikkunasta tekemään hakuja ja muokkauksia. Etenkin testausvaiheessa tästä oli hyötyä, sillä tiedonsiirron toimivuuden pystyi toteamaan nopeasti. Siemensin tarjoamat logiikkasovellukset ovat TIA Portal -ohjelmointiympäristö, jolla rakennettiin logiikkaohjelma sekä PLCSIM-simulointiympäristö, millä simuloitiin ohjelmoitavaa logiikkaa. Näille sovellustasoille piti luoda yhteys, tietokanta ja logiikkaohjelma, joiden avulla tiedonsiirto onnistuu työn tavoitteiden mukaisesti. Yhteys luotiin SQL Server 2016 Expressin ja ohjelmoitavaa logiikkaa simuloivan PLCSIM:n välille virtuaalisen Ethernet-adapterin avulla, mikä sujui vaivattomasti, sillä todellisia laitteita ei tarvittu testauksen toteutukseen.

Yritystä kiinnostavia asioita opinnäytetyössä oli SQL-tietokannan tiedonsiirtonopeudet sekä tiedonsiirtomäärät, eli millä syklillä ja kuinka paljon tietoa voi ohjelmoitavalta logiikalta SQL-tietokantaan lähettää. Toisena tutkimuskohteena oli tiedonsiirtoon vaadittavien sovellusten käytettävyys tämän kaltaisissa prosesseissa. Edellä mainittujen seikkojen avulla tämän tutkimuksen päämääränä oli selvittää SQL-tietokantaan perustuvaa datankeräyksen käytettävyyttä osana logiikkaohjelmaa.

Tiedonsiirto SQL-tietokantaan ohjelmoitavalta logiikalta antaa paljon mahdollisuuksia tiedon jatkojalostamiseen, sillä tällaisessa datan keräyksessä on kaksi ohjelmointityökalua, millä dataa voidaan käsitellä. Tämä mahdollistaa esimerkiksi sen, että dataa voi siirtää yksinkertaisimmassa muodossa ohjelmoitavalta logiikalta tietokantaan ja vasta tietokannassa muokata data haluttuun muotoon. SQL-kielessä voidaan esimerkiksi tulostaa vain tiettyjen raja-arvojen ylittävät prosessiarvot, jolloin raportointi ja analysointi on helpompaa. Etenkin Microsoftin SQL Server Express -tietokantatyökalu on erinomainen tällaiseen datankeräykseen, sillä se käyttää interaktiivista SQL:ää kyselyjen suorittamisessa, mikä mahdollistaa erittäin nopean vasteajan datan hakuun ja käsittelyyn. Tämä sovellus on myös ilmainen, mikä osaltaan helpottaa sen käytettävyyttä.

Simuloituna tiedonsiirto ohjelmoitavalta logiikalta ei tuottanut ongelmia. Yleisesti todellisten ohjelmoitavien logiikoiden prosessorit ovat suorituskyvyltään tehokkaampia. Tästä voidaan todeta, että ohjelmat toimivat myös todellisessa prosessissa vähintään yhtä hyvin. Ohjelma, jolla tiedonsiirto saatiin onnistumaan, koostuu Siemensin tiedonsiirto-ohjelmasta ja työtä varten rakennetusta datan muokkausohjelmasta. Muokkausohjelmassa vaikeuksia aiheutti Siemensin ohjelmointiympäristössä olevan merkkijono datatyyppi, sillä se rajoitti käytettävien merkkien määrän 256:een, mikä tiputti prosessiarvojen määrän noin 10-15. Tämä ratkaistiin sillä, että luotiin lisää tauluja, joihin dataa siirrettiin. Näin ollen ylärajana prosessiarvojen määrälle on teoriassa tietokannan muisti, suorittimen kapasiteetti tai käytännöllisyys, sillä jokaiselle yhdentoista arvon ryppäälle täytyi rakentaa tietokantaan oma taulu ja logiikkaohjelmaan omat toimilohkot. Lopulta rakennettiin ohjelma, millä pystyi siirtämään 55 arvoa viidelle eri taululle. Tätä, aluksi ongelmia aiheuttanutta datanjakoa eri tauluihin, voidaan pitää hyvänä ominaisuutena, sillä usein prosesseissa on eri mittauskohteita, joilla on omat näytteenottovälit. Jaon ansioista käyttäjä voi jakaa prosessiarvot eri tauluihin, niiden näytteenottovälien perusteella ja rakentaa logiikkaohjelman suorittamaan datansiirron eri ajankohtina. Tällöin prosessiarvot voidaan kerätä tietokantaan niille sopivin aikavälein.

Datan jatkojalostusmahdollisuuksia SQL Server Management Studiossa on useita. SQL-komentojen lisäksi datan käsittelylle ja raporttien tekemiselle on vi-

sualisointiohjelmia. Sovelluksessa itsessään on nettipohjainen Azure Data Studio, jolla tietokantaan siirretystä datasta voidaan tehdä erilaisia trendejä ja taulukoita. Tietokannan data voidaan esimerkiksi viedä yleisesti käytettyyn Microsoft Excel -ohjelmaympäristöön, jossa saadaan tämän ohjelman edut käyttöön.

Työn toteutus onnistui hyvin, sillä käytössä olleet laitteet ja sovellukset toimivat läpi tutkimuksen moitteetta. Työn lopputulosta voidaan soveltaa useaan eri tarkoitukseen ja sitä voidaan muokata projektien vaatimusten mukaan. Etenkin datan jatkojalostus visualisointiohjelmassa tuo paljon eri mahdollisuuksia monenlaiseen datan analysointiin.

LÄHTEET

Asiakastieto 2019. VEO Oy. Luettu 7.4.2021 https://www.asiakastieto.fi/yritykset/fi/veo-oy/15719966/taloustiedot

Bush, J. 2020. Learn SQL Database Programming. Birmingham: Packt Publishing.

John, K-H. & Tiegelkamp, M. 2010. IEC 61131-3: Programming Industrial Automation Systems. Forchheim: Springer.s

Microsoft 2016. Create a Login. Luettu 19.11.2020. https://docs.microsoft.com/en-us/sql/relational-databases/security/authentication-access/create-a-login?view=sql-server-ver15

Oppel, A. 2016. SQL: A Beginner's Guide Fourth Edition. Yhdysvallat: McGraw-Hill Education.

Siemens 2019. SIMATIC S7-1500 S7-PLCSIM Advanced. Luettu 03.02.2021. https://support.industry.siemens.com/cs/mdm/109773484?c=128470077195&lc=en-WW

Siemens 2013. Standards according to IEC 61131-3. Luettu 7.4.2021. https://support.industry.siemens.com/cs/document/50204938/how-do-you-program-the-plc-with-step-7-(tia-portal)-in-compliance-with-the-iec-61131-3-standard-?dti=0&lc=en-WW

Siemens 2020. Connecting a S7-1500 to a SQL Database V2.0. Luettu 15.10.2020. https://support.industry.siemens.com/cs/document/109779336/connecting-a-s7-1200-s7-1500-to-a-sql-database-?dti=0&lc=en-WW

VEO Oy 2021. VEO Oy kotisivut. Luettu 7.4.2021. https://www.veo.fi/