Niko Koivisto

MYSQL-TIETOKANTA CX-LAITTEELLA

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Sähkö- ja automaatiotekniikka
Helmikuu 2021

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Helmikuu 2021	**Tekijä/tekijät** Niko Koivisto
Koulutus Sähkö- ja automaatiotekniikka		☒ AMK ☐ YAMK
Työn nimi MYSQL-TIETOKANTA CX-LAITTEELLA		
Työn ohjaaja Hannu Puomio		**Sivumäärä** 23 + 8
Työelämäohjaaja Taneli Mäkitalo		

Tämä opinnäytetyö tehtiin toimeksiantona Aitomaatio OY:lle. Taneli Mäkitalo toimi työelämäohjaajani koko työn ajan. Työ on tarkoitus tulla oikeisiin kohteisiin helpottamaan tietokantatallennusta. Opinnäytetyön tarkoituksena oli tehdä tietokantatallennus CX-laitteelle, jossa on ARM-prosessori ja Windows CE -käyttöjärjestelmä. CE-käyttöjärjestelmä ja CX8180-laitetta ovat yhdessä liian epävakaat tietokannan käyttämiseen, joten päädyttiin käyttämään verkkokovalevyä tietokantatallennuksessa. Käytännössä tässä työssä tarkasteltiin tietokannan toimivuutta vain normaalilla tietokoneella. Tarkoituksena oli saada toimiva ohjelma ennen kuin siirrytään käyttämään ohjelmaa logiikan ja verkkokovalevyn kanssa. Verkkokovalevynä käytettiin QNAP-tuotetta, jossa on MySQL-tietokanta valmiina.

Tietokantaan oli tarkoitus tallentaa mittaushistoria ja hälytyshistoria. Mittaushistoria tallennus tehdään minuutin sykleissä ja hälytystallennus aina, kun raja-arvot ylittyvät. Hälytyshistoria tallennetaan jokaisen anturin kohdalta omaan tauluunsa, toisin kuin mittaushistoria, jonka tallennus tapahtuu yhteiseen tauluun. Tietokanta luotiin käyttämällä MySQL-kieltä. Työssä oli tarkoitus käydä yleisesti läpi myös tietokantojen luomista SQL-kielellä sekä tietokantatallennukseen käytettyjen työkalujen MySQL-, TwinCAT- ja TF6420-ohjelmien käyttöä.

Tavoitteena oli saada pienen mittakaavan tietokantatallennus edullisesti ja toimintavarmasti. Työ oli onnistunut ja käyttökelpoinen.

Asiasanat
Automaatio, MySQL, relaatiotietokanta, tietokanta.

ABSTRACT

Centria University of Applied Sciences	**Date** February 2021	**Author** Niko Koivisto
Degree programme Electrical and Automation Engineering		
Name of thesis MYSQL DATABASE ON CX DEVICE		
Instructor Hannu Puomio		**Pages** 23 + 8
Supervisor Taneli Mäkitalo		

The purpose of the thesis was to make a database storage for a CX device with an ARM processor and a Windows CE operating system. The CE operating system and the CX8180 together are too unstable to run the database, so it was decided that a network hard drive was used for database storage. In practice, this work looked at the functionality of the database only on a normal computer. The intention was to get a working program before switching to using the program with logic and a network hard drive. A QNAP product with a MySQL database ready was used as the network hard drive.

The database was intended to store measurement history and alarm history. The measurement history is recorded in one-minute cycles and the alarm is recorded whenever the limit values are exceeded. The alarm history is stored for each sensor in its own table, unlike the measurement history, which is stored in a shared table. The database was created using the MySQL language. The aim of the work was also to go through the creation of databases in SQL and the use of MySQL, TwinCAT and TF6420 tools for database storage.

The goal was to obtain small-scale database storage money wisely and to make it reliably. The work was successful and usable.

Key words
Automation, database, MySQL, relational database.

KÄSITTEIDEN MÄÄRITTELY

CX-laite

Automaatiossa käytettävä pienoistietokone

Funktion blokki

TwinCAT-ohjelman aliohjelma

Method

Funktion blokin tai ohjelman kutsuma aliohjelma

MySQL

SQL-ohjelmointi kieleen perustuva relaatiotietokantaohjelmisto

SQL

Standardoitu ohjelmointikieli **(**Structured Query Language)

ST-koodi

TwinCAT-ohjelmointikieli (Structured Text)

Windows CE

Microsoftin luoma pienitehoisiin laitteisiin suunniteltu käyttöjärjestelmä. Nykyään Windows Embedded Compact

KUVIOT

KUVAT

1 JOHDANTO

Tänä päivänä on suotavaa, että prosesseja voidaan seurata reaaliajassa sekä tarkistaa tarvittaessa tapahtumien historiaa pidemmältäkin aikaväliltä. Isoissa laitoksissa seurantalaitteet, tallennuslaitteet ja mittakaavat ovat suuremmat kuin pienemmän luokan laitoksissa. Reaaliaikainen seuranta ei tarvitse tietokantaa toimiakseen, vaan tieto johdetaan suoraa automaatiokeskuksen näyttöön tai valvomoon tietokoneen näytölle, jossa sitä voidaan tarkastella. Kun tietoa halutaan tallentaa, se pitää kerätä ohjelman avulla ja tallentaa tietokantaan. Tietokannat ovat yleistyneet paljon ja ne helpottavat vian selvityksessä jälkikäteen. Yöllä sattunut hälytys saadaan kirjattua sekunnin tarkasti tietokantaa ja samalla saadaan tallennettua tietokantaa kaikki tarvittava tieto siitä, mikä hälytykseen on johtanut.

Tässä opinnäytetyössä tavoitteena oli saada toimiva tietokantajärjestelmä käyttämällä MySQL-tietokantaa, joka olisi käyttäjäystävällinen ja se olisi suunniteltu myös pienen mittaluokan toimintaan. Tavoitteena oli saada mittaustieto tallennettua tietyin aikavälein tietokantaan ja aina halutessa luettua se. Tarkoituksena oli saada toimiva tietokantatallennus yhdelle mittausarvolle, jota voisi tarvittaessa laajentaa helposti useamman mittauksen tallentamiseen. MySQL-tietokanta käyttää taulukkomuotoa tallennukseen. Ylimmillä sarakkeilla on tallennettavien tietojen nimet, jotka luodaan taulua tehdessä. Tieto kerätään aina oman sarakkeen alapuolelle. Tässä opinnäytetyössä oli järkevää käyttää aikaleimaa tallennusta tehdessä, jotta tietokannasta lukeminen onnistuu tarkasti halutulla aikavälillä.

Tallennukseen käytettiin Beckhoff TwinCAT 3.1 XAE- ja TF6420 database – ohjelmia. TF6420 database – ohjelmalla luotiin tietyn aikavälein tietoa tallentava ohjelma ja TwinCAT-ohjelmalla luotiin ST-koodia käyttäen ohjelma, jolla tietokannasta voidaan lukea tietoa. Lisäksi asensin omalla koneelleni MySQL server – ohjelman, jonne tieto tallennettiin. Ongelmakohtina ilmeni pääsy tietokantaa muutoin kuin paikallisena käyttäjänä sekä tallennettu menettely – toiminnon käyttäminen kahdella IN-muuttujalla. Näihin ongelmiin paneudun toimintaohjeessa.

2 TIETOKANTOJEN MERKITYS

2000-luvulla tietokantojen merkitys on kasvanut merkittävästi käyttäjien ja datan räjähdysmäisen kasvun takia. Se tuo ongelmaksi sen, että yhden laitteen kapasiteetti tai suorituskyky ei yksinkertaisesti riitä. Ongelmaan ratkaisuksi käytetään yleisesti pilvipalveluita tai ulkoisia tallennusasemia. Käytännössä nämä ajavat saman asian sillä erotuksella, että ulkoiseen tallennusasemaan voidaan päästä käsiksi myös johdon kautta. Tietokantatallennus mahdollistaa datan käytön tehokkaasti ja suuret yritykset käyttävät sitä itse tai myyvät sitä mainontaa varten. Datasta on luotu suuret markkinat. (Taipalus 2017.)

Varsinkin mobiililaitteet, jotka käyttävät internet-yhteyttä, ovat aiheuttaneet erilaisia haasteita tietokannoille käyttäville suurille yrityksille. Ne ovat suosineet nykyään SQL-pohjaisia tietokantoja, koska niitä edeltävät tietokannat pitivät sisällään seuraavia ongelmia:

- Tietokannat eivät mukaudu tarpeeksi tehokkaasti suurille datamäärille
- Tietokannat eivät mukaudu tarpeeksi tehokkaasti suureen määrään käyttäjiä
- Tietokantojen hajautus ja muokkaaminen on työlästä.

Juuri näistä syistä suurimmat yritykset ovat siirtyneet käyttämään SQL-pohjaisia ja niiden johdannaisia tietokantatallennuksessa ja tietokantojen käsittelyssä. (Taipalus 2017.)

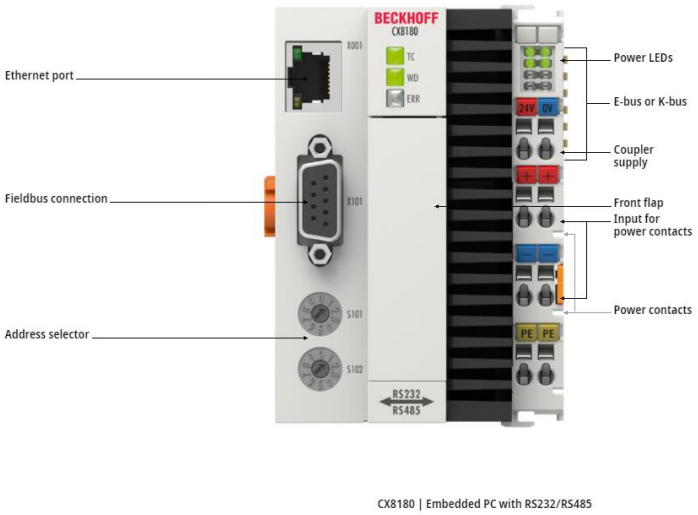
Pienillekin yrityksille tiedot ovat tärkeä resurssi ja niiden varastointi saattaa aiheuttaa investointeja. Investointeja helpottaa tallennuslaitteiden halpeneminen, mikä mahdollistaa uusien laitteiden hankinnan. Investointeja tehdäänkin nykyään kehityksen ehdoilla, eikä niinkään hinnan perusteella. Monet yritykset tukeutuvat johtoportaassa katsomaan yrityksen kehitystä nimenomaan tietokantoihin perustuen. Tietokannoista voidaan katsoa, kuinka yritys on kehittynyt ja mitkä toimet ovat siihen vaikuttaneet. On siis tärkeää, että tiedot ovat tallennettu oikein ja järkevässä muodossa. Tietokanta tarvitsee lähes poikkeuksetta tietokannan hallintajärjestelmän. Tässä opinnäytetyössä käytettiin MySQL-tietokantaa ja siihen liittyviä ohjelmia. (Hovi, Huotari & Lahdenmäki. 2005, 2, 4.)

3 SULAUTETTU JÄRJESTELMÄ

Sulautettu järjestelmä on yleensä jonkin tehtävän suorittamista varten suunniteltu laite. Se sisältää yleensä elektroniikkaa ja siihen on myös mahdollisesti liitetty mekaniikkaa sekä nimenomaiselle laitteelle suunnitellun ohjelmiston. Sulautettujen järjestelmän kehittymisen on mahdollistanut eri alojen tiivis yhteistyö. Järjestelmässä oleellista on se, että käyttäjän ei tarvitse välttämättä tiedostaa ohjelman luonnetta tai olemassaoloa lainkaan. Sulautetun järjestelmän ohjelmisto yleisesti toimii reaaliajassa. Ohjelmiston oletetaan reagoivan välittömästi muuttuviin tilanteisiin. Hyvin olennaista on myös sulautetun järjestelmän lähes olematon virrankulutus. (Lehtonen 2014.)

Sulautettujen järjestelmien koko voi vaihdella hyvin pienistä laitteista, kuten herätyskellosta, suuriin laitteisiin, kuten lentokoneisiin. Nykypäivänä autoteollisuudessa sulautettujen järjestelmien kehitys on ollut valtavaa. Viime vuosina yleiseksi tulleet autojen turvajärjestelmät, kuten ajanvakautusjärjestelmä tai kaistavahti, perustuvat sulautettuihin järjestelmiin. Sulautettujen järjestelmien ominaisuuksiin kuuluu lähes poikkeuksetta niiden ohjelmien muokattavuus. Ne voidaan ohjelmoida kokonaan uudestaan tai niitä voidaan muokata haluttuun suuntaan. (Lehtonen 2014.)

Tämän opinnäytetyön tarkoituksena oli luoda tietokanta tallennus CX-laitteelle, jolla on Windows CE-käyttöjärjestelmä. Käytettävänä logiikkana toimii CX8180, jonka ARM-prosessorin tehon rajallisuus käytännössä vaatii sen, että MySQL-tietokanta on asennettava ulkoiselle kovalevylle, jonne tallennus tapahtuu. Näin voidaan taata, että toimintavarmuus säilyy. Logiikka ja sen ohjelmisto on siis sulautettu järjestelmä. Ohjelmoitavaksi logiikaksi valikoitui Beckhoff CX8180 (KUVA 1). Tässä opinnäytetyössä ei ollut tarpeellista käyttää isoa ja tehokasta tietokonetta sen mahdollisen käyttökohteen takia. Tarkoituksena oli edullinen ja käyttäjäystävällinen vaihtoehto laadusta tinkimättä. Logiikan ohjelma toteutettiin TwinCAT 3.1 -ohjelmalla, joka voitiin asentaa Ethernet-käyttöliittymän kautta. Ohjelmien pidempiaikainen toimivuus varmistettiin asentamalla ohjelmiin tarvittavat lisenssit, ja ohjelma testattiin huolellisesti, jotta se toimi moitteettomasti. Testausvaiheessa voitiin luoda seitsemän päivän kokeilulisenssi ja testata ohjelman toimivuus sen avulla.



KUVA 1. Beckhoff CX8180 kuvattuna edestäpäin (Beckhoff – CX8180. 2020)

Laitteen RS232-sarjaliikenneväylä on kytketty tekstiviestimodeemille, joka toimii varaohjauksena. RS485 I/O-kortti on taas puolestaan kytkettynä virtausmittarin modbus-väylään. Sarjaliikenneväylä RS-485:ttä käytetään teollisuuden mittaus- ja ohjaussovelluksissa ja muissa automaatiojärjestelmissä, joissa väylälaitteiden etäisyydet ovat melko suuria, enimmillään noin 1200 metriä. Siirtonopeudet vaihtelevat 10 m:n 35 Mbps ja 1200 m:n 100 kbps väliltä. (Wikipedia 2020c.)

4 RELAATIOTIETOKANTA

Tietokannan voidaan yleisesti ajatella olevan kokoelma tiedoista, jotka voidaan loogisesti yhdistää toisiinsa. Lähes kaikkia tietokantoja yhdistävät seuraavat asiat:

- Tietokanta sisältää ennalta määrittelemättömän määrän yhteen kuuluvaa tietoa
- Tietokanta on suunniteltu ja toteutettu tiettyä ennalta määritettyä tapausta varten
- Tietokantaan voidaan lisätä ja poistaa käyttäjän haluamaa tietoa, ja eri käyttäjät voivat muokata olemassa olevia tietoja
- Tietokantaan voidaan luoda oikeuksia eri käyttäjiä varten
- Tietokantaa tallennetaan tieto vain kerran, tiedoissa ei esiinny toistoa
- Tietokannasta voidaan hakea tietoa joustavasti eri kyselyjen avulla. Hyvässä tietokannassa on mahdollisuus tallentaa muistiin hakuehtoja ja toteuttaa ne nopeasti sekä luoda uusia kyselyitä. (VIOPE 2020)

Vuonna 1970 Edgar F. Codd esitteli relaatiomallin, joihin nykyisetkin relaatiotietokannat perustuvat. Relaatiomalli perustuu joukko-oppiin, matematiikkaan ja predikaattilogiikkaan. Relaatiomalli ei ota kantaa relaatiokannan fyysiseen toteutukseen, vaan se on jätetty tietokantatuotteiden toimittajille (Hovi ym. 2005, 7). Relaatiotietokannat yleistyivät sen jälkeen nopeasti ja ovatkin nykyään käytetyimpiä tietokantoja. Suosio pohjautuu lähes yksistään siihen, että ne ovat yksinkertaisia, todella joustavia ja vastaavat hyvin tietokannalle ominaisiin vaatimuksiin. Uudempiakin tietokantamalleja löytyy, kuten oliotietokanta ja graafitietokanta, mutta niiden käyttö ei ole saavuttanut relaatiotietokannan saamaa suosiota. (Wikipedia 2020b; VIOPE 2020.)

Yksinkertainen tietokanta voisi olla puhelinluettelo, jossa on kerrottu nimi, osoite ja numero. Monimutkaisimmissa tietokannoissa voidaan käyttää useampaa tallennustaulua, jotka ovat yhdistetty eri komennoilla. Esimerkiksi työpaikalla A voisi olla tietokanta, jossa olisi työntekijöiden taulu, jossa olisi tunnus, nimi, syntymäaika, palkka, osasto, sukupuoli ja työvuodet sarakkeet. Tämän taulun voisi liittää keräilytauluun, jossa näkyisi keräilytunnus, kerääjä, kilot ja reitti. Reitit-taulussa voisi olla reitin nimi ja tunnus sarakkeet. Lisäksi voisi olla lähtevän tavaran taulu, jossa olisi Lähtevän tavaran tunnus, reitin valmius,

keräilijät ja kuljetusfirma ja nämä kaikki neljä taulua voisi liittää toisiinsa komennoilla ja saisimme kokonaiskuvan siitä, mitä työpaikalla A tapahtuu, kuten kuviossa (KUVIO 1) näkyy. Punaisella värillä on korostettu taulujen primäärejä avaimia.

TYÖNTEKIJÄT

T-Tunnus	Etu-nimi	Sukunimi	Syntymäaika	Palkka	Sukupuoli	Työvuodet
1	Asko	Asikainen	2.4.1965	35000	M	15
2	Jorma	Jokamies	14.6.1977	32000	M	12
3	Maija	Meikäläinen	8.12.1997	30000	N	2

REITIT

Reitin nimi	Reitin tunnus
Kannus	69100
Sievi	85410
Ylivieska	84100

KERÄILY

Keräily tunnus	Kerääjä	Kilot	Reitti
1	1	500	69100
2	2	500	85410
3	3	500	84100

LÄHTEVÄ TAVARA

Lähtevä tavara	Valmiina noutoon	Keräilijät	Nou-dettu	Kuljetusliike
1	11.12.2020 11:12	1,2	ON	Firma-A
2	11.12.2020 11:29	3	EI	Firma-B

KUVIO 1. Monimutkaisempi tietokanta

4.1 SQL-kieli

SQL on tietokonekieli, jolla voi rakentaa, ylläpitää ja käyttää hyväksi eri tarkoituksiin suunniteltuja relaatiotietokantoja. SQL on IBM:n alun perin kehittämä standardoitu kyselykieli, jolla relaatiotietokantaan voi tehdä erilaisia hakuja. SQL-kieli ei määrittele verkon protokollaa, jolla viestit välitetään, vaan se määriteetään käyttäjän toimesta. Tässä opinnäytetyössä käytetty MySQL ei ole ainoa ohjelma, joka käyttää SQL-kieltä. Suljetulla lähdekoodilla eli lisenssillä toimivia ohjelmia on saatavilla esim. Oracle, MS SQL server ja IBM DB2. Avointa lähdekoodia käyttäviä puolestaan suositaan niiden ilmaisuuden takia. Esimerkiksi MySQL, PostgreSQL ja MariaDB käyttävät avointa lähdekoodia. (Wikipedia 2020a; VIOPE 2020.)

4.2 MySQL-tietokanta

Tässä opinnäytetyössä tietokantaohjelmistoksi valikoitui MySQL, koska QNAP valmistaa verkkokovalevyjä, jossa MySQL-serveri on asennettuna valmiiksi. MySQL on ruotsalainen MySQL AB:n suunnittelema relaatiotietokantaohjelma. MySQL pohjautuu SQL-kieleen ja ensimmäinen versio julkaistiin vuonna 1996. Ohjelmoitavan logiikan takia ohjelman tekoon valitsimme TwinCAT-ohjelmiston. Lisäksi Beckhoffin TwinCAT-ohjelmistolla pystyttiin luomaan kaikki tarpeellinen projektin tietokantatallennusta varten. TwinCAT-projektiin voitiin lisätä TF6420 database server connectivity -projekti, jolla saatiin luotua automaattinen tallennus juuri sille aikavälille mille haluamme. Kyseisellä ohjelmalla vältyimme pitkältä PLC-koodilta ja saatiin yksinkertaistettua tietokantatallennusta.

MySQL on yksi maailman käytetyimmistä avoimen lähdekoodin relaatiotietokannoista. MySQL perustajat suomalainen Michael ”Monty” Widenius ja ruotsalainen David Axmark käyttivät omiin toimiinsa ensiksi mSQL-tietokantajärjestelmää ja totesivat, että se ei ollut heille tarpeeksi nopea ja joustava. Etuliite My tulee Montyn tyttären Myn mukaan. MySQL AB julkaisi ohjelmistonsa vuonna 1995 ja se perustui pelkästään MySQL-pohjaisten tuotteiden ja palvelujen ympäristöön. Vuonna 2008 Sun Microsystems osti MySQL AB:n, mutta jo seuraavana vuonna 2009 Oracle osti Sunin, jonka mukana siirtyi MySQL:n omistajuus Oraclelle. (MySQL 2020; Ahola 2011, 3.)

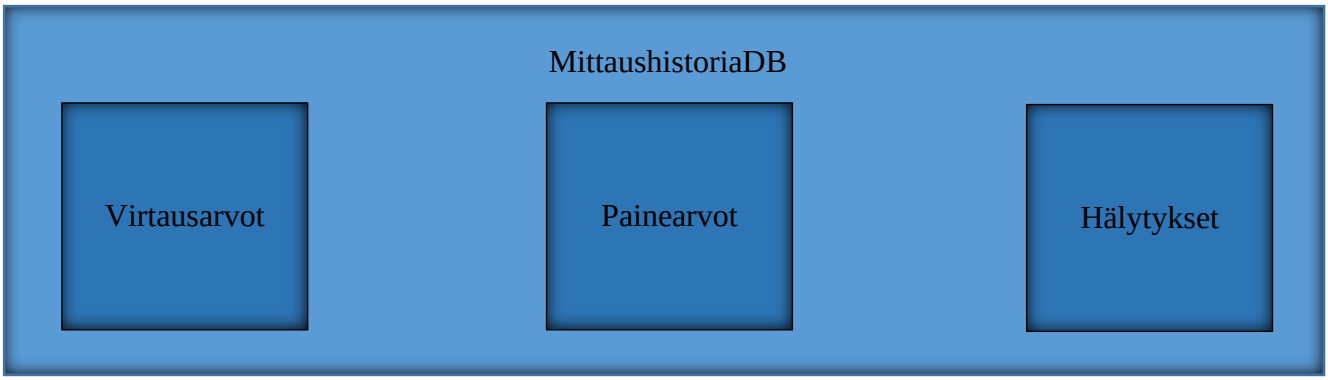
Kun haetaan yrityksille toimivia ratkaisuja, monesti katsotaan, mitä suuret yritykset tekevät. MySQL-tietokantaan turvautuu suurimmista firmoista mm. Facebook, Google ja Adobe. Tämä kertoo sen hyvin, kuinka MySQL soveltuu massiiviseen tietojen käsittelyyn yhtä hyvin kuin pienen mittakaavan projektiin. MySQL on kirjoitettu C- ja C++ -kielillä, mikä mahdollistaa laajan liitettävyyden eri alustoilla. (MySQL 2020)

Lisäksi tietoturva tuntuu olevan asiallisella tasolla. En päässyt ulkoisella logiikalla suoraa cat6-väyläkaapelin avulla kirjautumaan oman koneeni MySQL-serverille, vaikka kaikki Windows 10 -palomuurit oli laitettu pois päältä ja portti 3306 oli aikaistu vielä erikseen yhteyttä varten. MySQL vaati ulkopuoliselle käyttäjälle asetettavan käyttöoikeudet.

4.3 MySQL-tietokannan rakenne

MySQL, kuten monet muutkin relaatiotietokannat, tallentavat tiedot erillisiin tauluihin (tables) sen sijaan, että kaikkia tieto olisi yhdessä isossa varastossa. Tietokantarakenteet on järjestetty fyysisiin tiedostoihin optimoimaan nopeutta. Looginen malli, joka rakentuu tietokannoista, tauluista, näkymistä, riveistä ja sarakkeista, antaa joustavan ohjelmointi ympäristön. Käyttäjä voi määrittää säännöt, joilla määritellään tietokenttien välisiä suhteita. (MySQL 2020)

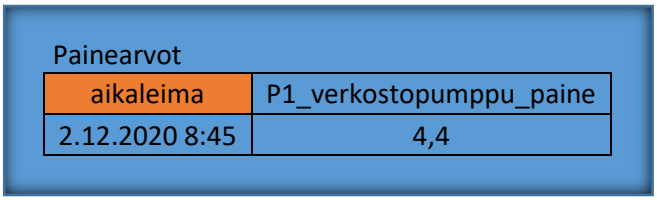
Ensimmäinen vaihe on tietokannan (database) luominen. Tietokannalle annetaan sopiva nimi ja nimeen on hyvä liittää loppuliite DB (database), jotta tietokanta ei sekoitu taulujen kanssa. Tietokannan luomisen jälkeen sen sisälle voidaan luoda taulu tai useampi tietokannan käyttötarkoituksen mukaan. Luodut taulut (KUVIO 2) Virtausarvot, Painearvot ja Hälytykset näkyvät vain tietokannan MittaushistoriaDB sisällä.



KUVIO 2. Esimerkki tietokannan rakenteesta

Taulua ei voi luoda ilman, että sille antaa rakenteen (KUVIO 3). Vähimmäisvaatimuksena taulu vaatii nimen ja tietotyypin. Yleensä tärkeille kentille annetaan pääavain, joka voidaan kuvitella uniikiksi luvuksi, jota mikään muu kentän tietue (record) ei voi saada. Tämän opinnäytetyön tietokannan pääavaimena toimii aika. Jokaiselle arvolle kirjoitetaan sen hetkinen aika eli päällekkäisyyksiä ei voi syntyä. Vaihtoehtona olisi ollut antaa juokseva ID-numero, joka toimisi primäärisenä avaimena, mutta tässä opinnäytetyössä sillä ei ole minun mielestäni merkitystä, koska tässä työssä tarkoituksena oli tallentaa syklisesti tietokantaan trendin piirtoa varten. Päällekkäisyyksiltä vältytään, kun ohjelma kirjoittaa joka minuutti sen hetkisen arvon tauluun ja tallentaa sille uniikin ajan. Hälytysarvon ylittyessä jokaisella mittausarvolla on oma hälytystaulunsa, jonne kirjoitetaan hälytysrajan ylittänyt arvo sen muuttuessa yhden sekunnin välein. Myöskään hälytystaulussa ei voi syntyä päällekkäisyyksiä, koska jokaisella mittausarvolla on oma ohjelma, joka tallentaa itsenäisesti tuloksiaan.

Tämän työn tarkoituksen mukaan suunniteltiin anturitallennuksien taulut. Hälytysarvotallennus olisi ollut mahdollista myös toteuttaa yhteen tauluun, jolloin samaan tauluun olisi tullut arvoja kuudelta eri anturilta. Tällöin tauluun olisi pitänyt lisätä jokainen mittausanturi ja tyhjiä tallennussarakkeita olisi syntynyt. Tiedot olisi voinut silloin saada luettavaksi nimeämällä esimerkiksi kenttien nimet uudelleen kyselyn tarkoitusperän mukaan. Luettavuuden kannalta oli kuitenkin paljon yksinkertaisempi tallentaa jokaista anturiarvoa erilliseen hälytystauluun, koska vain yksi anturiarvo saattaa ylittyä ja vältytään viideltä tyhjältä kentältä.



KUVIO 3. Esimerkki tietokannan painearvot-taulun rakenteesta

4.4 MySQL-rakenteen suhteet

Tässä opinnäytetyössä ei ollut tarpeellista käyttää taulujen välisiä suhteita. Tallennus tapahtuu jatkuvassa tallennuksessa yhteen tauluun minuutin aikavälein ja sitä on mahdoton sitoa yhteen hälytystaulun arvojen kanssa. Hälytysarvot kun voivat ylittyä milloin vain, eikä ainoastaan minuutin välein. Suhteilla pystytään yhdistämään yhden tai useamman taulun toisiinsa niin sanotuilla sidoksilla. Jos esimerkiksi tämän opinnäytetyön kaltaisia kohteita olisi useampia niin ne voitaisiin yhdistää sidoksilla toisiinsa. Vaihtoehtoisesti voitaisiin yhdistää venttiilien tilatieto taulu virtaus- tai paine-tauluihin. Yhteyksillä voidaan vähentää tiedon määrää välttämällä ylimääräistä tai kaksinkertaista dataa. Tässä työssä ylimääräisen datan tallennus vältettiin luomalla jokaiselle mitattavalle arvolle oma hälytystaulunsa niiden lukemisen selkeyttämiseksi. (VIOPE 2020)

4.5 MySQL-komennot

SQL komennoilla voidaan hakea, lisätä, päivittää, poistaa tai asettaa ehtoja haulle. SQL-komentoja ei voi muotoilla sanoiksi ja ne pitää toteuttaa SQL-kielellä. Yleisimpiä SQL-komentoja ovat seuraavat:

- SELECT – hakee tietoa tietokannasta
- UPDATE – päivittää tietoa tietokannassa

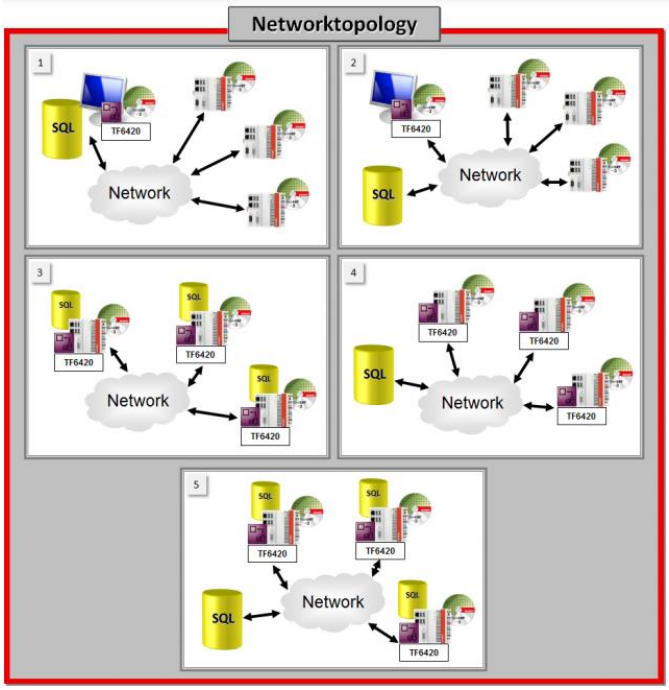
- DELETE – poistaa tietoa tietokannasta
- INSERT INTO – lisää tietoa tietokantaan
- CREATE DATABASE – luo uuden tietokannan
- ALTER DATABASE – muokkaa olemassa olevaa tietokantaa
- CREATE TABLE – luo taulun tietokantaan
- ALTER TABLE – muokkaa taulua
- DROP TABLE – poistaa taulun
- WHERE – ehto, jonka tietojen tulee täyttää.

SQL on monipuolinen ja voimakas tietokantakieli. Se on ns. ei-proseduraalinen kieli, mikä tarkoittaa, että sillä kerrotaan, mitä tietoa haetaan, ei sitä, miten tietoa haetaan. (Hovi ym. 2005, 10) Haku on hyvin samanlainen kuin Googlesta haettaessa. Kuitenkin normaalin kielen ”näytä kaikki arvot arvolle1” muotoutuu SQL-kielessä muotoon ”select arvo1 from virtausarvot”. Hakemisen lisäksi on tarpeellista myös pystyä luomaan uutta, lisäämään tietoa ja päivittämään sitä. SQL-tietokannasta haettaessa yleensä on helpompi antaa jokin ehto lauseelle, jonka sen tulee täyttää. Näin tiedämme heti, mitä hälytyksiä tullut 2.2.2020 - 3.2.2020 välisenä aikana, eikä tarvitse selata koko hälytyshistoriaa läpi.

Tietokanta tulisi suunnitella riittävän hyvin ja testata tarpeeksi sen toimivuutta, jolloin päästään tulokseen, missä CREATE-komennot ovat tarpeellisia ainoastaan tietokantoja tehdessä. Tähän tulokseen päästään keskustelemalla ja suunnittelemalla yhdessä yrityksen kanssa. On tärkeää kuunnella heidän tarpeensa ja toiveensa ja toteuttaa ne parhaalla mahdollisella tavalla. Voi kuitenkin olla, että myöhemmin tulee tarve lisätä esimerkiksi uusi tuote tai huomataan, että jokin tuote poistuu. Tämän opinnäytetyön tapauksessa, jokin venttiilin nimi voisi muuttua tai mittausantureita saattaisi tulla lisää. Silloin tarpeellisia toimintoja voidaan suorittaa UPDATE- ja DELETE-toiminnoilla.

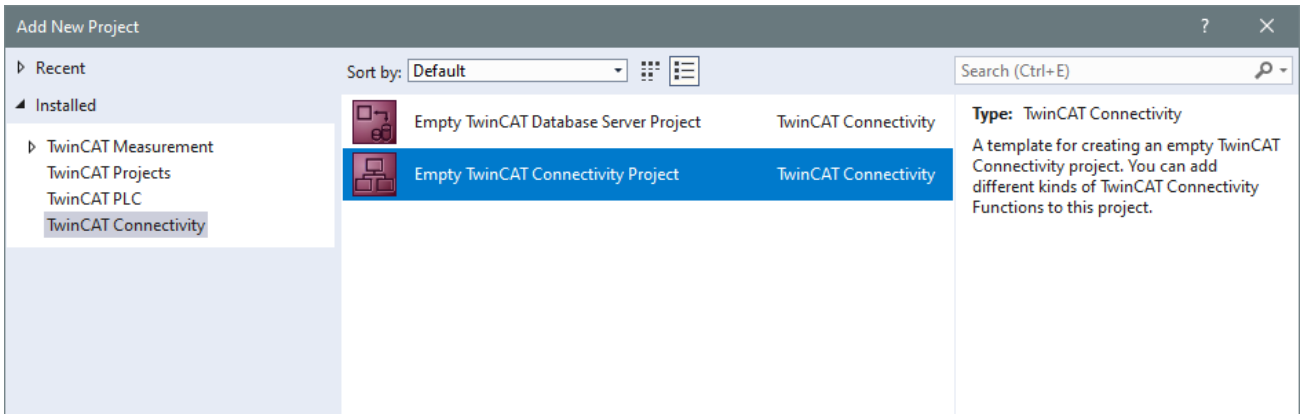
5 MYSQL SERVERIN JA TIETOKANTA TYÖKALUJEN ASENNUS

Suunnitelmien jälkeen tarvittiin tarpeelliset ohjelmat koneelle, jotta voitiin aloittaa tietokannan suunnittelu ja toteutus. MySQL-serveri on ladattavissa MySQL-sivulta ja on täysin ilmainen. Asennuksen yhteydessä tehdään MySQL-serverille root-salasana, jota käytetään, kun kirjaudutaan admin-tasolla MySQL serverille. TwinCAT 3.1 ohjelman asennuksen jälkeen asensin TF6420 database -serverin, jolla voidaan ottaa yhteys MySQ- serveriin. Asennukselle oli yksiselitteiset ohjeet TF6420-manuaalissa. Manuaalissa oli hyvä kuva (KUVA 2), kuinka verkkotopologioilla voidaan tietokantatallennusta toteuttaa. Testausvaiheessa käytimme esimerkkiä yksi ja oikeassa ympäristössä on tarkoitus käyttää tapausta neljä.



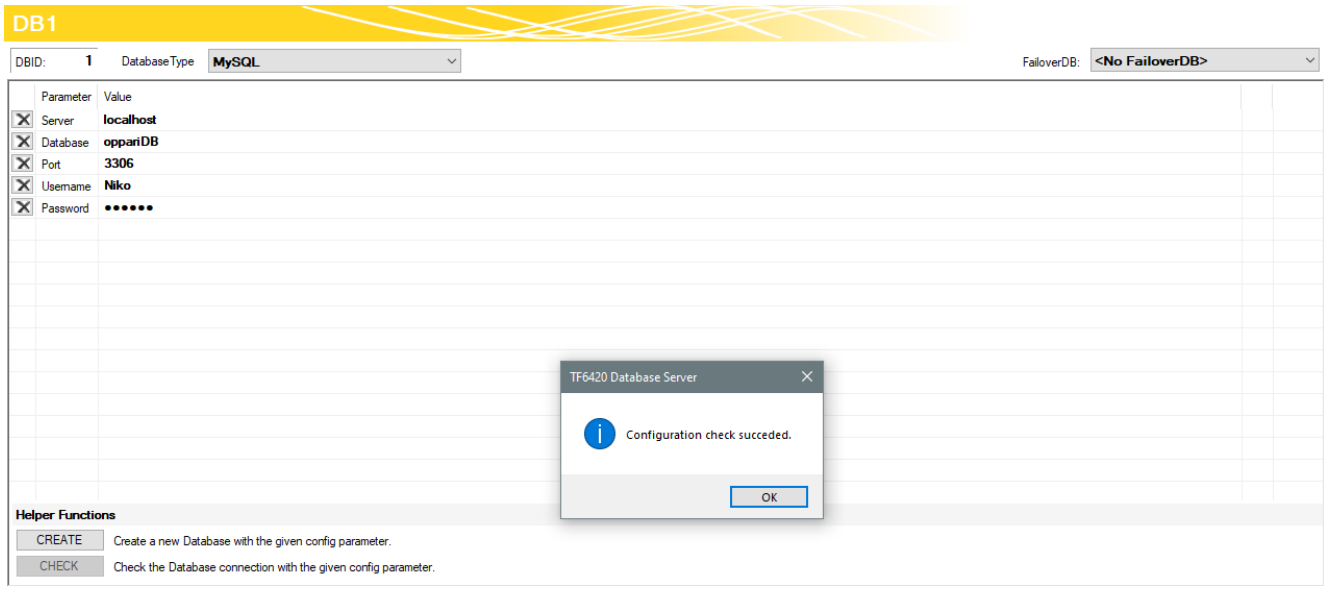
KUVA 2. Verkkotopologia (TF6420 TC3 Database Server 2020)

Yhteyden muodostaminen ei onnistu MySQL serverin kautta, vaan se luodaan TF6420-ohjelmalla. Käytin koko ohjelmoinnin ajan TF6420-ohjelmaa TwinCAT:in kautta, enkä erillisenä ohjelmana. TwinCAT-projektiin voitiin liittää luomalla TwinCAT connectivity projekti (KUVA 3), joka helpotti ohjelmointia. Näin saatiin kaikki toiminnot saman ohjelman sisälle. Lisäksi tietokantojen ja taulujen luominen ja poistaminen olivat minusta helpompia toteuttaa valmiilla ohjelmalla, kuin command linen avulla.



KUVA 3. TwinCAT Connectivity -projektin lisääminen

Yhteys tietokantaan muodostettiin (KUVA 4) syöttämällä oman koneeni tiedot. Samalla koneella oli myös MySQL-serveri, jota käytimme testausvaiheessa.



KUVA 4. Yhteyden muodostaminen MySQL-serveriin

Yhteyden luomisen jälkeen MySQL-tietokanta oli valmiina käytettäväksi. Yhteys pysyy päällä niin kauan kuin TwinCAT on päällä ja uudelleenkäynnistyksen jälkeen se luo yhteyden uudelleen automaattisesti.

Tätä vaihetta päätimme myös simuloida myös logiikan kanssa. Tietokoneelleni asennettu MySQL-serveri toimi tallennukseen ja lukemiseen ulkoisen verkkokovalevyn sijaan. Yhteyden luomisessa kuitenkin huomasimme heti ongelman. TwinCAT antoi virhekoodiksi käyttäjäntason puuttumiseen, vaikka ainoastaan paikallisen käyttäjän sijaan käytimme oikeita IP-osoitteita. Virhe toistui, vaikka portille 3306 oli

tehty aukko palomuuriin. Emme saaneet yhteyttä toimivaan, vaikka palomuuri otettiin kokonaan pois käytöstä.

Ongelman syyksi saimme paikannettua MySQL-serverin suojauksen, joka ei päästä ulkopuolisia tahoja käyttämään serveriä, vaikka ulkopuolelta annettaisiin oikea tieto päästä kirjautumaan sisälle paikallisena käyttäjänä. MySQL-serveri vaatii tehtäväksi käyttäjän, jolla on tunnukset käyttöä serverin varten. Paikallisen käyttäjän tiedoilla pääsee kirjautumaan ainoastaan samalta tietokoneelta käsin. Ratkaisuna voitiin luoda MySQL-serverille käyttäjä, joka sallii muistakin IP-osoitteista tulevan yhteyden. Käyttäjälle on myös syytä luoda kaikki oikeudet, että sillä voi tallentaa ja lukea tietokannasta.

MySQL-serverille käyttäjän luominen onnistuu seuraavasti:

```
mysql>CREATE USER 'käyttäjä'@'%' IDENTIFIED BY 'salainensalasana';
mysql>GRANT ALL ON *.* TO 'käyttäjä'@'%';
mysql>flush privileges;
```

Nyt käyttäjällä käyttäjä saa yhteyden MySQL-serverille ja se on valmis käyttöön. Tiedot tulee vaihtaa tietokannan yhdistämissivulle TF6420-ohjelmassa käyttäjän ja salasanan kohdalle.

6 MYSQL TIETOKANTA JA SEN TOIMINNOT

Tässä opinnäytetyössä keskitytään tallentamaan mittaustuloksia ja antamaan niille aikaleima, jotta niitä voidaan hakea tietokannasta halutulla aikavälillä. Vianmääritystä varten oli tarpeellista tehdä erillinen hälytystaulu, jotta tarkka aika jää talteen tallentaen anturin arvon. Tietokanta on siis hyvin yksinkertainen, mutta erittäin hyödyllinen ja kustannustehokas pienen mittakaavan toimijoille. Parhaassa tapauksessa se säästää aikaa ja rahaa, kun vika ja aika on selvillä.

Aloitin suunnittelun keräämällä tietoa yhdessä toimeksiantajani kanssa siitä, mitä tietokantaan olisi hyvä tallentaa. Kaikilta kuudelta anturilta tuleva tieto tulisi tallettaa mittaustauluun (KUVA 5), ja mikäli raja-arvot ylittyvät tai alittuvat, silloin ne tallennetaan myös hälytystauluihin (KUVA 6). Jokaiselle arvolle tallennetaan aikaleima sekä seurannan että vianmäärityksen helpottamiseksi. En kokenut tarpeelliseksi antaa juoksevaa ID-numeroa, koska tallennus jokaiselta anturilta tapahtuu päätauluun minuutin välein riippumatta arvoista. Hälytystauluihin tallennus tapahtuu aina kun raja-arvot ylittyvät tai alittuvat. Jokaiselle anturiarvolle on oma hälytystaulunsa ja oma ohjelmansa, jolla ne tallennetaan. Näin päästään ongelmasta, että päällekkäisyyksiä ja tyhjiä arvoja ei tarvitse tallentaa. Ohjelmat tallentavat hälytysarvon sekunnin välein, jos arvo on muuttunut.

aikaleima	rP1_verkostopumppu_nopeus	rP2_verkostopumppu_nopeus	rP3_verkostopumppu_nopeus	rLahtopaine	rSailion_pinnankorkeus	rVirtaus_verkostoon
2020-12-02 08:59:31	0	0	0	0	0	0
2020-12-02 08:59:36	0	0	0	0	0	0
2020-12-02 08:59:41	0	0	0	0	0	0
2020-12-02 08:59:46	0	0	0	0	0	0
2020-12-02 08:59:51	0	0	0	0	0	0
2020-12-02 08:59:56	0	0	0	0	0	0
2020-12-02 09:00:01	0	0	0	0	0	0
2020-12-02 09:00:06	0	0	0	0	0	0
2020-12-02 09:00:11	0	0	0	0	0	0
2020-12-02 09:00:16	0	0	0	0	0	0

10 rows in set (0.02 sec)

KUVA 5. Mittaustaulu

aikaleima	P1_hälytys
2020-12-08 13:54:20	6
2020-12-08 13:54:36	1
2020-12-08 13:54:39	2
2020-12-08 13:54:48	6

4 rows in set (0.00 sec)

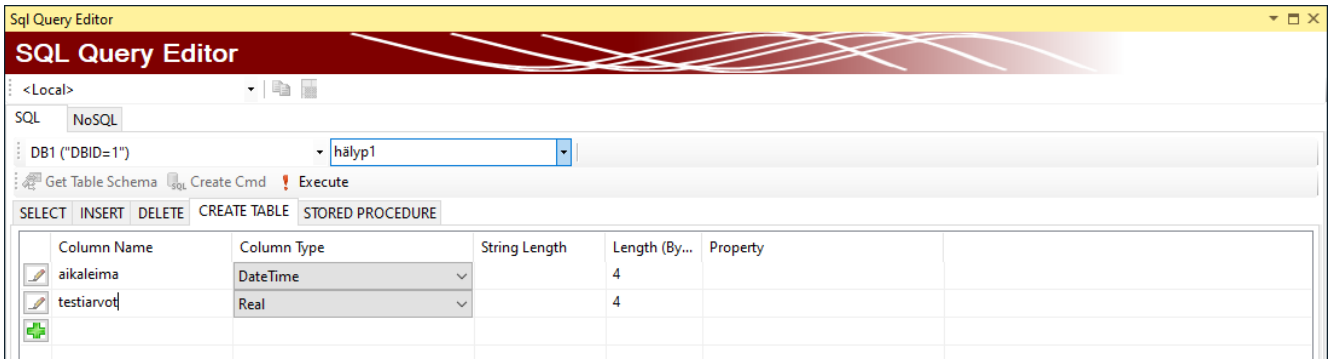
KUVA 6. Hälytystaulu

6.1 Tietokannan ja taulujen luominen

Tärkeimpinä MySQL tietokannan komentoina voidaan pitää hyvin pientä osaa komennoista. Tärkeimpinä voidaan pitää luomiskomentoja. Niiden lisäksi tärkeitä komentoja ovat kyselykomennot. Niitä yhdistelemällä voidaan tehdä tarpeeksi kattavia kyselyitä tietokantaan.

MySQL:n käyttö alkaa tietokannan luomisella ”Create database” – komennolla. Sillä saadaan luotua tietokanta ja ”use” – komennolla tietokanta saadaan MySQL-serverin käyttöön, jotta sinne voidaan edelleen luoda taulu. Taulu luodaan ”create table” – komennolla ja samalla sille täytyy antaa rivien pääsarakkeiden nimet. MySQL-serveri antaa virheilmoituksen, jos jokin komento menee väärin. Jokainen haluttu komento pitää lopettaa puolipisteeseen.

Samat toiminnot TF6420-ohjelmassa näyttävät tältä (Kuva 7). TwinCAT:issa näkymä on mieluisa ja ohjelma on käyttäjäystävällinen. Kirjoittamisen tarve on jäänyt minimiin ja komentoja voi tehdä tietämättä niistä tai niiden rakenteesta helposti. Näen tämän sekä uhkana että mahdollisuutena. Itse pidän siitä, että tiedän, mitkä komennot ovat oikeasti ohjelman takana. Esimerkiksi tallennettu menettely – toimintoa TwinCAT:in puolella voi vain kutsua, mutta ei luoda. Luominen tapahtuu command linen kautta.



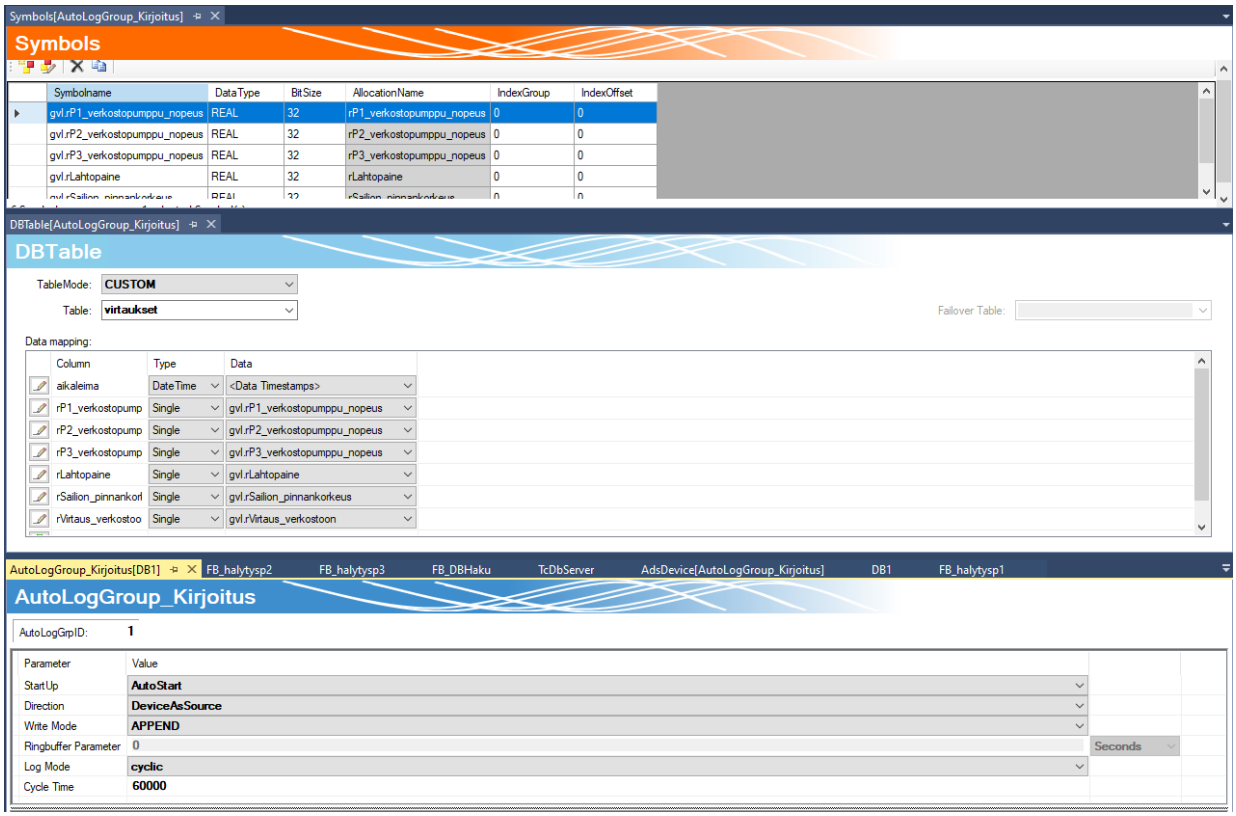
KUVA 7. TF6420-ohjelman MySQL-komentojen luominen ja toteutus

6.2 Tietokantaan tallennus

Tietokantaan tallennus tässä tapauksessa piti olla automatisoitua ja toistua tietyin väliajoin. Valitsin kahden hyvän, TwinCAT PLC puolen ST-koodin ja TF6420-ohjelman, väliltä. Tallennukseen valikoitui erinomaisen ohjelmoitavuuden ja helppouden vuoksi TF6420. Se tarjosi jatkuvan tallennuksen mahdollisuuden, on helposti muokattavissa ja on automaattisesti käynnistyvä uudelleen käynnistyksen jälkeen.

Sinne voidaan syöttää arvoja sekä PLC-koodin kautta luettavaksi että logiikan omasta muistista. Lisäksi sen konfigurointi (KUVA 8) oli mielekkäämpää ja selkeämpää kuin ST-koodin kirjoittaminen.

Hienon visuaalisen verhon takana TF6420 käyttää komentoa ”INSERT INTO [taulu] VALUES [sarakkeet, arvot]” tallentamiseen tietokantaan. Taulu kohtaan merkitään haluttu taulun nimi, joka on luotu aikaisemmin, sarakkeet-kohtaan tulee pääsarakkeiden nimi ja arvot-kohtaan niiden numeraaliset arvot. TF6420-ohjelmassa testausvaiheessa annettiin GVL-muuttujat ohjelman tallennukseen symbolitaulussa. DB-taulussa näkyy MySQL-tietokannassa olevan taulun rakenne, jonka se saa itse haettua yhteyden muodostuksen jälkeen.



KUVA 8. Tietokantatallennuksen konfigurointi TF6420-ohjelmassa

6.2.1 Jatkuva tallennus

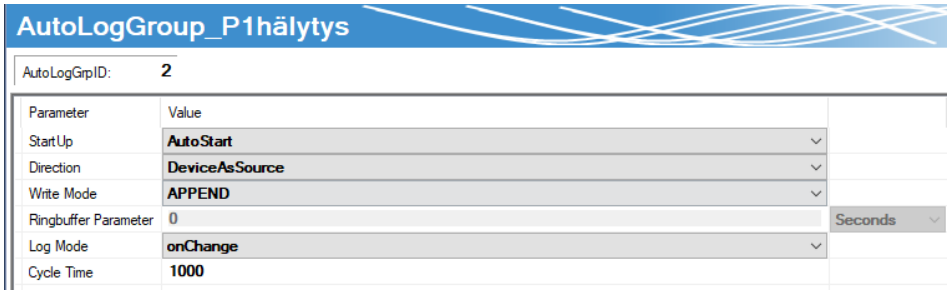
TF6420 tarjoaa loistavan pohjan automaattiselle jatkuvalle tallennukselle (KUVA8). Nimensäkin mukaan AutoLogGroup_kirjoitusohjelma aloittaa kirjoittamisen automaattisesti, kun ohjelma logiikassa käynnistyy. Se tallentaa logiikasta verkkokovalevylle ja kirjoittaa joka kerta oman uuden arvon. Kirjoittaminen tapahtuu jatkuvasti 60 sekunnin välein. Vaihtoehtona harkitsin, että tulokset olisi kirjattu ylös

sekunnin välein ja laskettu keskiarvo minuutin ajanjaksolle. Tulin kuitenkin siihen tulokseen, että tallennus minuutin välein riittää, koska keskiarvo ei kerro aina totuutta. Lisäksi raja-arvot ylittäville arvoille on olemassa hälytystaulu ja erillinen tallennus. Näin ollen minuutin välein tapahtuva tallennus on myös riittävä näyttämään selkeän kuvan halutun ajan tapahtumista käyttäjän taulukon graafissa.

Vaihtoehtona jatkuvalle tallennukselle olisi voinut käyttää arvon mukaan muuttuvaa tallennusta. Toisin sanoen, joka kerta kun arvo muuttuu, se kirjoitettaisiin tietokantaan. Tämä muodostuu ongelmaksi tämän kaltaisissa mittauksissa sen takia, että veden virtaus- ja painearvot muuttuvat alati. Mittaustuloksia olisi kertynyt sata-, ellei tuhatkertainen määrä minuutin välein tapahtuvaan tallennukseen verrattuna.

6.2.2 Hälytyshistorian tallennus

Mittausarvojen tallennus raja-arvojen ylittyessä oli tärkein osa mittauksien tallennuksen osalta. Toiveena oli, että saadaan tarkasti tietää, milloin raja-arvot ovat ylittyneet ja mikä sen hetkinen anturiarvo on ollut. Hälytysjärjestelmästä minun ei tarvinnut huolehtia, vaan keskityin ainoastaan tallentamaan tietoa. TF6420 tarjosi minulle hyvän mahdollisuuden tallentaa arvon aina muuttuessa (KUVA 9). Se mahdollisti sen, että pystyin ST-koodilla tekemään muuttujan x, jolle annetaan raja-arvot. X muuttuu vain, kun se ylittää raja-arvot ja silloin TF6420 tallentaa arvon x tietokantaan. Jokaiselle kuudelle anturiarvolle on oma ohjelma, joka tallentaa vain sen kyseisen anturin arvoa tietokantaan. Tallennusväliksi valitsin yhden sekunnin, koska anturiarvot saatavat muuttua koko ajan. Näin saadaan tarpeeksi selkeää ja riittävää tietoa, vaikka tallennus tapahtuukin vain yhden sekunnin välein



KUVA 9. P1-arvon tallennus hälytystauluun

Jokaiselle anturiarvolle on olemassa oma hälytystaulunsa (KUVA 7), jonne tieto tallennetaan. Näin vältymme tyhjiltä sarakkeilta tietokannassa, koska yksi arvo saattaa ylittyä, mutta kaikki muut ovat sallituissa rajoissa. Lisäksi tietokannasta hakeminen helpottuu WHERE-lauseen avulla, kun hakuun tarvitsee lisätä vain haluttu ajankohta.

Vaihtoehtoisesti olisin voinut toteuttaa tallennuksen ST-koodilla, mutta TF6420:n kautta se onnistui todella paljon helpommin ainakin omasta mielestäni. Ajatellen käyttötarkoitusta tässä tilanteessa TF6420:n kautta toteutus oli riittävä. Muokkaaminen on myös helppo tehdä myöhemmässä vaiheessa, jos sille on tarvetta.

6.3 Tietokannasta lukeminen

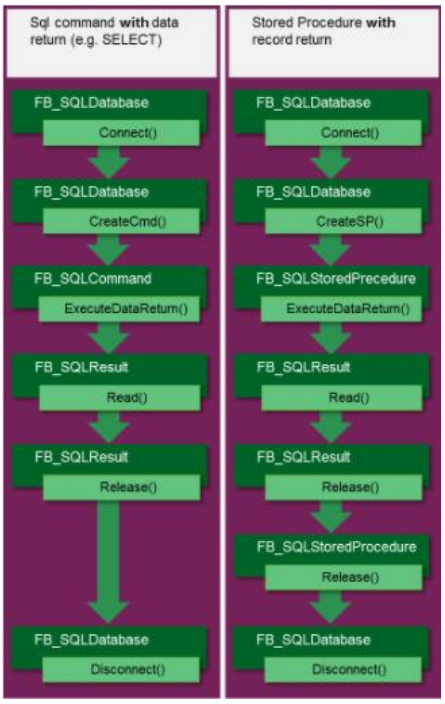
Tietokannasta lukeminen onnistuu hyvin TF6420-ohjelmalla SQL query-editorilla, jos haluaa kaikki tulokset esille yhdellä kertaa. SQL query editorissa ei voi käyttää WHERE-lauseketta, mitä ihmettelin kovasti ohjelman muutoin hyvän toteutuksen vuoksi. Melkein jokainen haku, joka toteutetaan, tarvitsee WHERE-lausekkeen, koska halutaan vain jotain tiettyä tietoa näkyville. Tässä työssä pitää pystyä aika kohdentamaan tietylle välille. Tätä varten rakensin ST-koodilla hakuohjelman funktion blokille (LIITE 1), jolle voi syöttää haun alku- ja loppuajan. Käyttäjälle tehdään myöhemmin näkymä, mistä hän voi itse asettaa haluansa ajat ja painaa HAKU-näppäintä ja nähdä graafin oikein skaalattuna omalla näytöllään.

6.4 SQL Expert mode

Beckhoffin manuaalissa (TF6420 TC3 Database Server. 2020) oli ohjeistettu yksiselitteisesti, miten funktion blokkeja voidaan hyödyntää. Lisäksi manuaalissa oli muutama hyvä esimerkki, joiden avulla ymmärsi hyvin, kuinka ne toimivat. ST-koodin valitsin lukemiseen, koska tarkoituksena on saada käyttäjälle näytettyä grafiikkaa, miten arvot ovat eläneet tiettynä aikana. ST-koodin osalta pystyttiin hyödyntämään ARRAY-toimintoa ja saamaan haun tulokset talteen.

Asiaa hieman tutkittuani, tulin siihen johtopäätökseen, että paras toteutus syntyy käyttämällä tallennettua menettelyä (Kuva 10). Tämä on toiminto, joka tallennetaan MySQL-serverille ja sitä voidaan kutsua useasti ja helposti. Vaihtoehtona olisi ollut SELECT-toiminnon käyttäminen (KUVA 10), mutta aikojen lisääminen STRING-muuttujaan ei ollut aivan yksiselitteistä, joten päädyimme käyttämään tallennettua

menettelyä. Ilman ongelmiakaan tässä ei selvitty, mutta kiitos Beckhoffin teknisen tuen, sain moneenkin asiaan vastauksia. Siellä asia otettiin nopeasti käsittelyyn ja palvelu oli todella hyvää, vaikkakin kyseessä oli ”vain” opinnäytetyö.



KUVA 10. SQL Expert mode (TF6420 TC3 Database Server. 2020)

Tein oman funktion blokin haulle, jossa oli triggerit neljälle toiminnolle: yhdistämiselle, suorittamiselle, lukemiselle ja yhteyden sulkemiselle. Nämä toiminnot olivat TC3-kirjaston sisältämiä funktion blokkeja (KUVA 10). Tarkoituksena on saada myöhemmin tehtyä käyttäjälle valikko, josta käyttäjä syöttää haluamansa ajat ja painaa HAKU-näppäintä. Tämän jälkeen funktion blokki suorittaa järjestyksessä edellä mainitut toiminnot ja näyttää grafiikan käyttäjälle.

FB_SQLDatabase funktion blokilla sain avattua yhteyden MySQL-serveriin käyttämällä connect()-metodia. Yhteyden avaamisen yhteydessä tarkistetaan virheet ja annetaan käyttäjälle virheilmoitus, jos jokin virhe ilmaantuu. Jos yhteyden avaaminen onnistuu, funktion blokki siirtyy suorittamaan createSP()-metodia. Tämä menetelmä tekee MySQL-serveriä varten komennon, joka sisältää kaksi IN-muuttujaa tallennettua menettelyä varten. IN-muuttuja on jokin datatyyppi ja tapauksessamme päivämäärä ja aika. MySQL-serverin tallennettu menettely käyttää näytä IN-muuttujia hakutoiminnossa. IN-muuttujaa en saanut toimimaan ilman, että sille antoi DATETIME(0)-arvon. Tämä tarkoittaa, että sekunnit annetaan vain yksikkö muodossa. Arvo voisi olla nollan ja kuuden väliltä, mikä lisää desimaaleja sekunnin jälkeen. Lopuksi kursori siirretään FB_SQLStoredProceduren päälle.

Yksi ongelmista oli, kuinka saan suoritettua kahden IN-muuttujan tallennetun menettelyn. Ongelma ratkesi sillä, kun näille kahdelle muuttujalle loi oman structin DUTsiin. ST-koodin ohjelma tarvitsi tiedon IN-muuttujille siirrettävästä tiedosta structin muodossa. Lisäksi kesti hetken, ennen kuin löysin ratkaisun IN-muuttujan toimimiseen.

Method, jota FB_SQLStoredProcedure funktion blokki käyttää, on ExecuteDataReturn(). Kursori siirtyy structuren päälle, jossa on molempien aikojen rakenne. Tämä method suorittaa kahden IN-muuttujan kyselyn MySQL-serverillä käyttäen tallennettua menettelyä (KUVA 11). Käytännössä se hakee kaikki arvot käyttäjän valitsemalta ajalta. BETWEEN-toiminto sisällyttää molemmat valitut ajat haun lopputuloksissa. Tämä metodi tarkistaa erikseen tehdystä structista, millaisen tilan se tarvitsee MySQL-serverin arvojen palautusta varten. Structi täytyy tehdä sen mukaan mitä haettavaan tauluun on tallennettu. Lukemista varten kursori siirretään FB_SQLResultin päälle.

FB_SQLResult funktion blokki käyttää read()-metodia. Read lukee palautetut arvot ST-koodin array-taulukkoon, joka on suunniteltu MySQL-taulun mukaiseksi. Sieltä ne on helppo lukea yksittäin arvoihin ja näyttää käyttäjälle näkyvässä taulukossa graafina. Lukeminen on helppo toteuttaa esimerkiksi ST-koodin loop-toiminnon avulla.

Release()-metodilla suljetaan funktio blokkien SQLResult ja SQLStoredProcedure avatut yhteydet. Ohjeissa kerrottiin, että ne vaativat sen, jos niitä halutaan käyttää uudestaan. Tämä toiminto nollaa parametrit, joita ne käyttivät hakemiseen tietokannasta. Disconnect()-metodi sulkee yhteyden SQLDatabase funktion blokilta. Yhteyden jättäminen päälle vie tehoa ja lisää tietoturvariskin mahdollisuutta. (TF6420 TC3 Database Server 2020)

6.5 MySQL:n Tallennettu menettely

SELECT-komennolla voidaan luoda kyselyitä tietokantaan. Se kuitenkin joudutaan tekemään joka kerta uudestaan, kun tietoa halutaan. Beckhoffin ST-koodin puolella ongelmaksi muodostui kyselylauseen luominen muuttujilla. Yhden arvon tai sanan lisääminen ei ole suurikaan ongelma, mutta päivämäärän muuttaminen DT_TO_STRING-komennolla lisää sanan alkuun ”DT#”. Sain asian ratkaistuksi, mutta en siististi ja haluamallani tavalla. Totesin siinä olevan liikaa muuttujia, joten päädyin etsimään toista vaihtoehtoa.

MySQL-tietokantaan on mahdollista tallentaa omia hakuja. Yksi tapa on toteuttaa SELECT-komentoa uudestaan ja uudestaan, mutta toinen tapa on tehdä tallennettu menettely – komento. Sekin käyttää SELECT-hakua, mutta aikamuuttujien lisääminen oli helpompaa tallennetun menettelyn IN-muuttujien kautta. TwinCAT-kirjastosta löytyi valmiina funktion blokkeja, jolla pystyi käyttämään tallennettuja menetelmiä. Totesin tämän siis olevan parempi vaihtoehto tehdä kyselyjä useammin ja varmemmin. Huonona puolena mainittakoon, että tallennettua menettelyä ei voi muokata jälkikäteen, vaan se pitää poistaa ja luoda uudelleen.

Seuraava ote on MySQL-komentosuoritteesta, kun loin tallennetun menettelyn:

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE SP_testihaku (IN aika1 DATETIME(0), IN aika2 DA-
TETIME(0))
   -> BEGIN
   -> SELECT * FROM testiarvot WHERE aikaleima BETWEEN aika1 AND aika2;
   -> END; //
mysql> DELIMITER ;
```

Tällä koodilla sain luotua tallennetun menettelyn MySQL-serverille ja valjastettua sen tähän käyttötarkoitukseen. DELIMITER-komento muuttaa lauseen päätöstä ;-merkistä //-merkkiin, ja koodin lopussa se tietenkin muutetaan takaisin ;-merkkiin, jotta komennot toimivat moitteetta. Tämä pitää tehdä sen takia, että MySQL-serverin komennot pitää päättyä ;-merkkiin. Tällä tekemälläni tavalla pystytään syöttämään kaksi ;-merkkiä tallennettuun menettelyyn, jolloin komennosta tulee toimiva. Sisään menevät arvot aika1 ja aika2 määritetään käyttäjän toimesta ja ne määräävät ajanjakson, jolta haku suoritetaan. *-merkki kertoo, että testiarvot-taulusta haetaan kaikkia arvot.

7 POHDINTA

Tässä työssä oli tarkoitus luoda toimiva tietokantajärjestelmä, jolla voidaan tallentaa haluttua tietoa sekä tarpeen tullen pystyä lukemaan sitä tietokannasta. Käytössämme ollut MySQL soveltui loistavasti tämän työn tietojen tallennukseen. TwinCAT osoittautui käyttäjäystävälliseksi ja sitä oli todella mieluisa käyttää tässä työssä. Tarkoituksena oli avata hieman tässä työssä käytettäviä tietokantatyökaluja ja tietokannan luomiseen liittyviä vaiheita ja kertoa niiden toiminnasta. Vaikka tietokannat luodaan aina projektin tarpeiden mukaan, niin mielestäni onnituin myös kertomaan hyvin yleisellä tasolla, kuinka tietokantatallennus tapahtuu. Kaiken kaikkiaan nykymaailman tietojen määrän vuoksi on todella hyödyllistä osata järjestää ja varastoida tietoa järkevästi. Lisäksi on tärkeää, että tietoja voidaan käyttää hyödyksi tarpeen mukaan. Tietokanta työkalut antavat siihen loistavan mahdollisuuden niiden hyvän käytettävyyden ja muokattavuuden takia.

Opinkin tässä työssä kuinka tärkeää on osata järjestää tietoa oikein ja suunnitella projekti, oli se sitten kuinka pieni tahansa, oikein ja huolellisesti. Huolellisen suunnittelun ansiosta voidaan vähentää datan määrää, vältytään turhilta tallennuksilta ja saadaan otettua käyttöön tehokkaasti tallennettu data komentojen avulla. Lisäksi tietokanta hausta on pystyttävä luomaan yksinkertainen ja selkeä käyttöinen käyttäjälle.

Toimeksiantajana tälle työlle toimi Automaatio Mäkitalo, jolla oli tarve pienen mittakaavan tietokantatallennukselle. Tekemäni työ jää hänen yrityksellensä käyttöön ja olen kiitollinen, että sain tehdä näin mielenkiintoisen projektin. Työn eri vaiheet tekivät työstä mielenkiintoisen ja lisäsivät tietoa ja ymmärrystä sekä tekijälle itselleen ja myös toimeksiantajalle. Toivon myös tämän opinnäytetyön tuovan ymmärrystä tietokantatallennuksesta kiinnostuneille ja siitä tietoa etsiville. Kiitoksen sana kuuluu myös opinnäytetyön ohjaavalle opettajalle Hannu Puomiolle erinomaisista näkökulmista, ideoista ja materiaaleista.

LÄHTEET

Ahola, H. 2011. Relaatiotietokannan suunnittelu ja toteutus Case: Kiinteistötietokanta. Opinnäytetyö. Ylivieska: Keski-pohjanmaan ammattikorkeakoulu. Saatavissa: http://urn.fi/URN:NBN:fi:amk-201105208865. Viitattu 3.12.2020.

Beckhoff – TF6420 TC3 Database Server. 2020. Saatavissa https://download.beckhoff.com/download/document/automation/twincat3/TF6420_TC3_Database_Server_EN.pdf. Viitattu 20.12.2020.

Beckhoff – TC3 database server. 2020. Saatavissa https://www.beckhoff.com/english.asp?twincat/tf6420.htm. Viitattu 20.12.2020.

Beckhoff – CX8180. 2020. Saatavissa https://www.beckhoff.com/fi-fi/products/ipc/embedded-pcs/cx8100-arm-cortex-a9/cx8180.html. Viitattu 20.12.2020.

Hovi, A., Huotari, J. & Lahdenmäki, T. 2005. Tietokantojen suunnittelu & indeksointi. Jyväskylä: Docendo.

MySQL – What is MySQL? 2020. Saatavissa https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. Viitattu 3.12.2020.

MySQL – MySQL Community downloads. 2020. Saatavissa https://dev.mysql.com/downloads/mysql/. Viitattu 3.12.2020.

MySQL – Chapter 3 Tutorial. 2020. Saatavissa https://dev.mysql.com/doc/refman/8.0/en/tutorial.html. Viitattu 3.12.2020.

Taipalus, T. 2017.Tietokannat ja tiedonhallinnan perusteet. Saatavissa https://tim.jyu.fi/view/kurssit/tktl/itka204/kurssimoniste#zqlbHVm1qlhQ. Viitattu 20.12.2020.

Lehtonen, T. 2014. Sulautettujen järjestelmien ketterä käsikirja. Saatavissa https://www.utupub.fi/handle/10024/99142. Viitattu 20.12.2020.

VIOPE. 2020. SQL-Tietokannat. https://vw4.viope.com/. Viitattu 3.12.2020.

W3schools. 2020. Tutorial. Saatavissa https://www.w3schools.com/sql/default.asp. Viitattu 3.12.2020.

Wikipedia. 2020a. Wikipedia - SQL. Saatavissa https://fi.wikipedia.org/wiki/SQL. Viitattu 20.12.2020.

Wikipedia. 2020b. Wikipedia - MySQL. Saatavissa https://fi.wikipedia.org/wiki/MySQL. Viitattu 20.12.2020.

Wikipedia. 2020c. Wikipedia - RS-485 Saatavissa. https://fi.wikipedia.org/wiki/RS-485. Viitattu 20.12.2020.