Metropolia Ammattikorkeakoulu
Elektroniikan koulutusohjelma

Tomi Järvinen

Tuotantotestauksen mittausdatan analysointi

Insinöörityö 9.6.2009

Ohjaaja: tuotantopäällikkö Robin Cavén
Ohjaava opettaja: yliopettaja Kari Salmi

Metropolia Ammattikorkeakoulu Insinöörityön tiivistelmä

Tekijä Otsikko Sivumäärä Aika	Tomi Järvinen Tuotantotestauksen mittausdatan analysointi 55 sivua 9.6.2009
Koulutusohjelma	elektroniikka
Tutkinto	insinööri (AMK)
Ohjaaja Ohjaava opettaja	tuotantopäällikkö Robin Cavén yliopettaja Kari Salmi

Tässä insinöörityössä tutkittiin Fastrax Oy:n oman tuotantotestauksen tuottamaa mittausdataa, joka on tallennettu tietokantaan. Tarkoituksena oli selvittää, miten tietokantaan tehtäviä hakuja saataisiin tehokkaammiksi ja nopeammiksi, jotta voitaisiin paremmin analysoida tietoa. Aluksi selvitettiin tietokannan rakennetta ja sitä miten sen taulut liittyvät toisiinsa sekä mitkä taulut mahdollisesti tuottavat ongelmia hakujen yhteydessä. Hakuaikojen parantamiseksi luotiin uudet indeksit ja optimoitiin tietokanta sekä kiinnitettiin huomiota hakutapaan. Vaadittavat haut tietokantaan suoritettiin SQL-kielellä tehdyillä koodeilla. Työn suorituksessa ohjelmana oli Microsoft SQL Server ja siihen kuuluvat SQL Query Analyzer sekä SQL Server Enterprise Manager.

Tietokantaan suoritettujen huoltotoimenpiteiden jälkeen huomattiin hakuaikojen yleisesti nopeutuneen huomattavasti. Kun jollain tietyllä sarjanumerolla suoritettiin haku, hakuaika oli pudonnut kolmesta tunnista noin minuuttiin. Toisaalta ryhmänumeron hakuaika, joka oli aluksi viisi tuntia, oli edelleen turhan korkea. Kun haku suoritettiin kolmessa eri osassa, päästiin noin kymmenestä minuutista alle viiden minuutin tulokseen. Jaetussa haussa hyödynnettiin ohjelmiston välimuistia, johon aina edellinen haku tallentui. Jaettu haku toteutettiin etenemällä tietokannassa askel kerrallaan, kunnes saavutettiin haluttu tieto.

Työn tavoitteet saavutettiin osittain. Sarjanumerolla tehdyt haut saatiin tarpeeksi nopeiksi mutta ryhmänumerolla suoritetut haut jäivät pidemmiksi kuin tavoitteena oli. Ryhmänumerolle suoritettu indeksointi ei onnistunut odotetusti, ja se vaatiikin vielä parantelua. Tietokannan kunnolla on todella suuri vaikutus hakuaikoihin, kuten tuloksista huomattiin, varsinkin jos tietokanta on useita kymmeniä gigatavuja. Ryhmänumeron hidas hakuaika johtui sen sisältämästä suuresta tietomäärästä. Hakuaikaa pystytään todennäköisesti parantamaan luomalla parempi indeksi. Paremman indeksin kehittämisessä tulee kiinnittää huomiota tietokannan rakenteeseen ja siihen, mitä tietoa halutaan saada nopeammin uudella indeksillä. Kun hakuajat pysyvät maksimissaan minuutissa, voidaan datan analysointia tehdä useammin, ja näin saadaan selville nopeammin tuotteissa olevat viat. Suuri tietokanta tulisi optimoida ainakin kaksi kertaa vuodessa.

Hakusanat	SQL, relaatiotietokanta, tietokannan indeksointi, tietokannan optimointi, monimuuttujamenetelmät

Helsinki Metropolia University of Applied Sciences **Abstract**

Author	Tomi Järvinen
Title	Measurement data analysis in production testing
Number of Pages	55
Date	9 June 2009
Degree Programme	Electronics Engineering
Degree	Bachelor of Engineering
Instructor	Robin Cavén, Production & Sourcing Manager
Supervisor	Kari Salmi, Principal Lecturer

The subject of this final year project was the measurement data analysis in production testing. The studied data is saved in the database of the company, Fastrax. The goal of the project was to produce a more effective way of analyzing data and a faster way of making searches to the database. At the beginning of the project, the structure of the database, how the databases panels are linked together and what panels might cause problems while doing searches were studied. New indexes and optimization of the database were carried out and more attention was paid to the search mode to result in a better access time. All the codes for demanded searches where executed with the SQL language. The project was realized by using Microsoft SQL Server, which included SQL Query Analyzer and SQL Server Enterprise Manager.

After making service operations to the database, it was noticed that access times were generally much faster than earlier. When the search was done with a certain serial number, the access time dropped from three hours to approximately one minute. However, the batch number access time, which was originally five hours, was still too long. When the search was done in three different sections, the results of the access time were dropped from ten minutes to approximately five minutes. In the section search, the software's cache memory was used and the latest search was saved there. The section search was executed by moving in the database step by step until, the wanted data was reached.

The goal of the project was achieved partly: searches done with a serial number were fast enough but the batch number access times were longer than expected. Indexes for the batch numbers were not created successfully and this part of the project still needs to be developed further. The condition of the database has a great effect on the access times, as the results show, especially if the database is as large as several dozens of gigabytes. The reason for the long access times in batch number searches was that these include a large amount of information. A better access time could possibly be reached by creating a better index. When creating a better index, focus should be on the structure of the database and on the information needed to be reached faster. When the access time takes a maximum of one minute, it is possible to execute analyses more often and to find the problems of the products more effectively. A large database should be optimized at least twice a year.

Keywords	SQL, relational database, indexing of database, optimization of database, multivariate analysis

Sisällys

Lyhenteet, määritelmät ja symbolit

A	Pxr-matriisi, joka on faktorianalyysissä faktorimatriisi. Sisältää faktorien ja muuttujien välillä olevat korrelaatiokertoimet.
$F = (F_1,F_2,...,F_r)$	Yhteisfaktoreita faktorianalyysissä.
GPS	Global Positioning System, maailmanlaajuinen paikannusjärjestelmä.
Kenttä	Tietokannassa olevan rivin yksi sarake, sisältää tarkat tiedot jostain aiheesta.
Microsoft SQL server	Tietojen hallintaohjelmisto.
Predikaattilogiikka	Tutkitaan tietynlaisia merkkijonojen joukkoja, on symbolisen logiikan osa-alue.
r	Faktoreiden lukumäärä.
SQL	Structured Query Language, standardoitu kyselykieli relaatiotietokannoille.
SQL Query Analyzer	Toteutetaan SQL-kyselyitä, SQL-serverin hallintatyökalu.
SQL Server Enterprise Manager	Toteutetaan SQL-kyselyitä, SQL-serverin hallintatyökalu.
T	Säännöllinen rxr-matriisi.

Taulu	Tietokannan näkyvin ja isoin osa käyttäjälle, sisältää tietueita ja kenttiä.
Tietue	Tietokannassa olevan taulun yksi rivi, joka sisältää yhteen aiheeseen liittyvät tiedot.
$U = (U_1,U_2,...,U_p)$	Ominaisfaktorit faktorianalyysissä.
$X_1,X_2,...,Xp$	Yleisesti havaitut muuttujat faktorianalyysissä.
Ψ^2	Alkioiden Ψ^2_1, Ψ^2_2,..., Ψ^2_p lävistäjämatriisi.
Φ	Faktorianalyysin perusyhtälöön kuuluva matriisi.
μ	Odotusarvovektori faktorianalyysissä.
$\sum$	Kovarianssimatriisi.

1 Johdanto

Tämän insinöörityön aiheena on tuotantotestauksen mittausdatan analysointi. Työssä käsitellään tietokantojen teoriaa sekä, sitä miten ne ovat kehittyneet, tutkitaan millaisia analysointimenetelmiä voi hyödyntää, pyritään indeksoimaan tietokantaa ja suoritetaan SQL-kielellä tehtyjä hakuja tietokantaan halutun tiedon saamiseksi. Työn alussa käsitellään tietokantoja ja niiden toimivuutta, ja lisäksi tutkitaan, millaisilla analysointimenetelmillä voidaan analysoida tietokannoista saatavaa tietoa, siten ettei tapahtuisi vääristymiä. Työn loppupuolella tutkitaan, onko indeksoinnista apua, kun suoritetaan raskaita SQL-kyselyitä ja tulkitaan saatua tietoa.

Työn aihe on Fastrax Oy:stä ja työ on suoritettu pääasiallisesti sen tiloissa. Mittausdata on Fastraxin omasta tuotantotestauksesta tulevaa tietoa, ja työn tarkoituksena on parantaa ja nopeuttaa tiedon analysointia. Fastrax Oy on perustettu vuonna 1999, ja sen tuotteet perustuvat GPS-teknologiaan. Sen tuotteet on suunniteltu vastaamaan vaativimpienkin asiakkaiden tarpeita, kun on kyse GPS-vastaanottimen koosta, virran kulutuksesta, suorituskyvystä ja ohjelmoitavuudesta. Fastrax Oy:n tuotteet soveltuvat niin teollisuuden kuin tavalliselle käyttäjälle suunnattujen laitteiden tarpeisiin. (6.)

Nykyään jokaisella yrityksellä on käytössään tietokantoja, ja niiden käyttö sekä tarve kasvavat koko ajan. Tiedon määrän kasvaessa pyritään kehittämään parempia ja nopeampia tapoja saada haluttu tieto esiin ja tarvittaessa analysoitavaksi. Kun tietokantojen koko on useita kymmeniä gigatavuja, ei tavallisilla menetelmillä kyetä enää hakemaan tietoa kohtuullisessa ajassa. Tätä varten tietokantaa pitää huoltaa, eli pitää luoda indeksejä ja optimoida tietokantaa. Myös SQL-kyselyissä kannattaa siirtyä alikyselyihin, koska se nopeuttaa tiedon hakua.

2 Tietokannat

2.1 Yleistä

Tietokannat ovat yleisesti ottaen tallennettujen, loogisten, yhteenkuuluvien tietojen joukko. Sitä voidaan analysoida helposti tietokantakielellä, kuten SQL:llä (Structured Query Language). Tietoja, jotka tietokanta sisältää, hallinnoi erityinen ohjelmisto, tietokannan hallintajärjestelmä. Näitä tietokannan hallintajärjestelmän ohjelmistoja ovat muun muassa MySQL, Microsoft SQL Server, Oracle, Access ja DB2. Yleensä tietokannan hallintajärjestelmät ovat isoja ohjelmistoja ja erittäin monimutkaisia. Ne tarjoavat kuitenkin käyttäjille ja ohjelmoijille monia hyödyllisiä palveluita. Tietokantaan tallennetaan tiedot tietoyhteyden turvaamiseksi, sovellusohjelmoinnin helpottamiseksi, muutosjoustavuuden lisäämiseksi sekä suorituskyvyn parantamiseksi. (1, s. 4–5; 2, s. 2–3.)

Mikäli tietokannan hallintajärjestelmää ei olisi, jouduttaisiin käyttämään tiedostoja. Silloin monimutkaisten tietokokonaisuuksien ohjelmointi olisi huomattavasti työläämpää, tietojen hakeminen olisi todella vaikeaa ja eheys tietokannan sisällä olisi heikompaa. Tietokanta voi siis tietokannan hallintajärjestelmän avulla olla:

- Yhteiskäyttöinen: yhteistä kantaa käyttävät useat eri sovellukset.
- Ei-toisteinen: tiedot on tallennettu kertaalleen.
- Ajantasainen: käyttäjien tekemät päivitykset ovat heti kaikkien näkyvillä.
- Eheä: tiedot kuvaavat mahdollisimman hyvin reaalimaailmaa ja ovat ristiriidattomia.

(1, s. 4–5; 2, s. 2–3.)

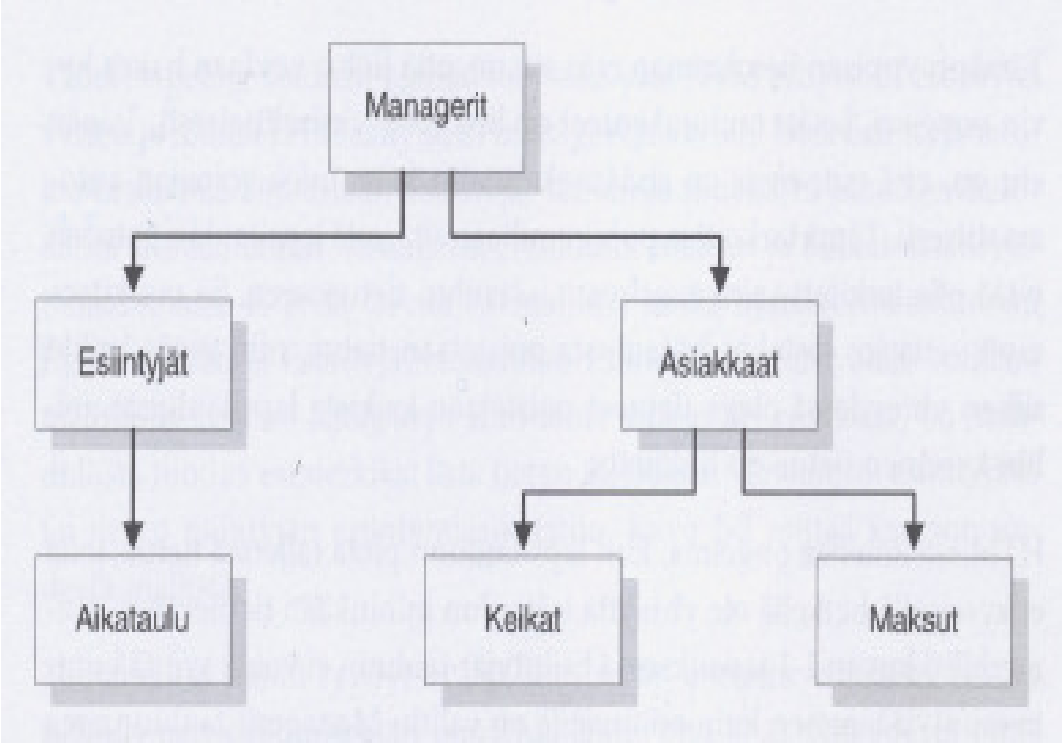
Ennen relaatiotietokantamallia tietojen käsittelyyn ja säilyttämiseen käytettiin yleisesti kahta mallia, hierarkkista ja verkkomallista. Melkein kaikki tietokannan hallintajärjestelmät ovat nykyään SQL-pohjaisia relaatiotietokantoja. Perinteiset

hierarkkiset ja verkkomalliset tietokannat ovat hankalampia muuttaa ja käyttää kuin relaatiotietokannat. Tietokantojen hallintajärjestelmissä käytetään nykyisin kahdenlaisia tietokantoja: analyyttisia tietokantoja ja käyttötietokantoja. (1, s. 5; 3, s. 3–4.)

Käyttötietokantoja käytetään päivittäin, ja yleensä niitä käytetään, kun pitää säilyttää, kerätä ja muokata tietoa. Data, jota tallennetaan tämäntyyppiseen tietokantaan, on dynaamista. Dynaaminen tarkoittaa sitä, että data heijastaa ajan tasalla olevaa tietoa, vaikka se muuttuu jatkuvasti. Käyttötietokantoja ovat esimerkiksi tilaustenhoitotietokannat ja inventaariotietokannat. Analyyttisia tietokantoja puolestaan käytetään ajasta riippuvien ja historiallisten tietojen seuraamiseen ja tallentamiseen. Mikäli yrityksen pitää tehdä suunnitelmia pitkälle aikavälille, tutkia pidemmän aikajakson tilastotietoja tai seurata kehityksen suuntaa, tallennetaan tarvittavat tiedot analyyttiseen tietokantaan. Tiedot analyyttisessä tietokannassa ovat staattisia, toisin sanoen tietokanta koskee jotain tiettyä ajankohtaa, eikä tietoa tietokannassa muuteta juuri koskaan. Analyyttisia tietokantoja ovat esimerkiksi kyselytutkimustietokannat ja kemiallisten testien tietokannat. (1, s. 5–6; 3, s. 3–4.)

2.2 Hierarkkinen tietokantamalli

Hierarkkisessa tietokannassa tiedot on järjestetty hierarkkisesti. Yleisesti tätä mallia kuvataan ylösalaisin käännettynä puuna. Hierarkkisessa tietokantamallissa yksi tietokannan taulu toimii ”juurena” ylösalaisin käännetylle puulle, ja toiset taulut ovat haaroja, jotka kasvavat juuresta. Hierarkkisessa tietokantamallissa sen rakenne on jäykkä, eikä sillä sen takia kyetä esittämään kaikkia rakenteita. Kuvassa 1 on esitetty kaavio hierarkkisesta tietokantamallista. (2, s. 3; 3, s. 4.)



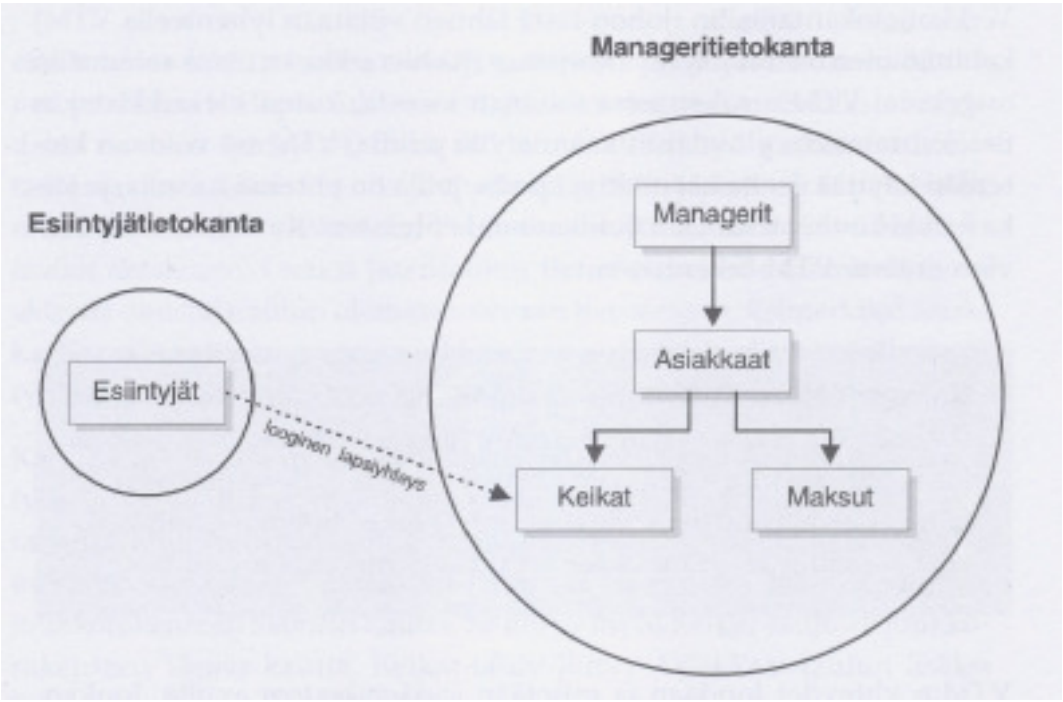
Kuva 1. Hierarkkisen tietokantamallin kaavio (3, s. 5).

Hierarkkisen tietokantamallin taulujen sisältämiä suhteita kuvataan käyttämällä käsitteitä ”isä” ja ”lapsi”. Tämä tarkoittaa sitä, että isätaulu voidaan yhdistää useisiin lapsitauluihin, mutta yksittäisellä lapsitaululla ei voi olla kuin yksi isätaulu. Kuvassa 1 isätauluna on Managerit ja lapsitauluja ovat Esiintyjä ja Asiakkaat, Esiintyjät-taulu toimii isätauluna lapsitaululle Aikataulu ja lapsitaulujen Keikat ja Maksut isätaulu on Asiakkaat. Taulut linkitetään toisiinsa osoittimien avulla tai tietueiden fyysisen järjestyksen perusteella taulun sisällä. Tässä mallissa käyttäjän pitää tuntea tietokannan rakenne hyvin, jotta haluttu tieto saadaan esiin, koska tiedon hakeminen pitää aloittaa aina juuritaulusta. (3, s. 5.)

Hierarkkisen tietokantamallin yksi etu onkin se, että tieto kyetään hakemaan todella nopeasti, koska taulurakenteet on linkitetty yksiselitteisesti. Toisena etuna voidaan pitää sitä, että viite-eheys on sisäänrakennettu ja voimassa automaattisesti. Pohjimmiltaan tämä tarkoittaa, että tietueen lapsitaulun pitää olla linkitetty olemassa olevaan tietueeseen isätaulussa. Tämä tarkoittaa myös sitä, että jos tietue poistetaan isätaulusta, myös lapsitaulusta poistetaan kaikki tietueet, joihin se oli linkitetty. (3, s. 6.)

Ongelmia hierarkkisessa tietokantamallissa ilmenee silloin, kun tietue pitäisi tallettaa lapsitauluun, jolla ei ole sillä hetkellä yhteyttä mihinkään tietueeseen isätaulussa. Ongelma voidaan ratkaista lisäämällä esimerkiksi kuvassa 1 olevaan Managerit-tauluun yksi näennäinen manageri, ja näin saadaan esimerkiksi esiintyjät lisättyä tietokantaan, ennen kuin niille löydetään oikea manageri. (2, s. 3; 3, s. 6–7.)

Toinen ongelma hierarkkisessa tietokantamallissa on ylimääräinen data, koska tässä mallissa ei voida muodostaa monimutkaisia suhteita. Esimerkiksi kuvassa 1 esiintyjien ja asiakkaiden välillä on monesta moneen -yhteys, eli asiakas voi palkata lukuisia esiintyjiä, ja esiintyjä voi esiintyä useille asiakkaille. Koska hierarkkisessa tietokantamallissa ei pystytä kuvaamaan tätä, pitää sekä Keikat- että Aikataulu-tauluun lisätä ylimääräistä dataa. Tästä seuraa päällekkäisyysongelma, joka mahdollistaa sen, että tietokannasta saadaan virheellistä tietoa. Ongelma voidaan kuitenkin ratkaista luomalla kaksi hierarkkista tietokantamallia, toinen managereita ja toinen esiintyjiä varten. (2, s. 3; 3, s. 6–7.)

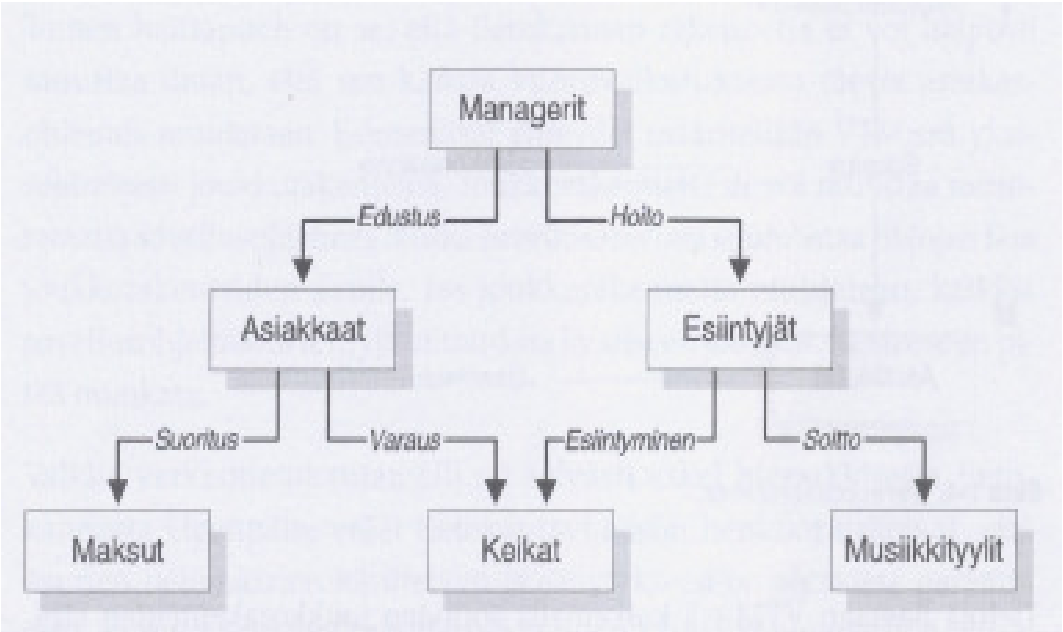


Kuva 2. Kahden hierarkkisen tietokantamallin avulla tehty ratkaisu monesta moneen -yhteyteen (3, s. 7).

2.3 Verkkotietokantamalli

Verkkotietokantamalli kehitettiin, jotta voitaisiin korjata hierarkkisen tietokantamallin ongelmia. Verkkotietokantamallia ja sen rakennetta voidaan kuvata samalla tavalla kuin hierarkkista tietokantamallia eli ylösalaisin olevana puuna. Verkkotietokantamallissa pystytään käyttämään useita käännettyjä puita, jotka kaikki sisältyvät samaan tietokantarakenteeseen ja joilla on yhteisiä haaroja. Kuvassa 3 on esitetty kaavio verkkotietokantamallista. (3, s. 8.)

Yhteydet esitetään ja luodaan verkkotietokantamallissa joukkorakenteen avulla. Rakenne joukkorakenteessa on läpinäkyvä, ja se liittää kaksi taulua yhteen niin, että toisesta tulee jäsentaulu ja toisesta omistajataulu. Yhdestä moneen -yhteydet ovat näin ollen mahdollisia joukkorakenteita. Toisin sanoen omistajataulun tietueet voivat olla yhteydessä lukuisiin jäsentaulun tietueisiin, mutta yksittäinen tietue jäsentaulusta voi olla yhteydessä ainoastaan yhteen tietueeseen omistajataulussa. Tämän lisäksi jäsentaulun tietue ei ole olemassa, jos sillä ei ole yhteyttä olemassaolevaan tietueeseen omistajataulussa. Esimerkiksi kuvassa 3 Managerit-taulu on omistajataulu ja Asiakkaat ja Esiintyjät ovat jäsentauluja. (3, s. 8–9.)



Kuva 3. Verkkotietokantamallin kaavio (3, s. 9).

Yksittäinen taulu voidaan liittää niin moneen muuhun joukkoon tietokannan muiden taulujen kanssa kuin halutaan, ja kahden taulun välille kyetään määrittelemään niin monta joukkoa kuin on tarvetta. Kuvassa 3 Suoritus on joukkorakenne ja yhdistää Asiakkaat-taulun Maksut-tauluun. Asiakkaat-taulu liittyy myös joukkorakenteen Varaus kautta Keikat-tauluun, ja Keikat-taulu liittyy myös Esiintyjät-tauluun joukkorakenteen Esiintyminen avulla. (3, s. 9.)

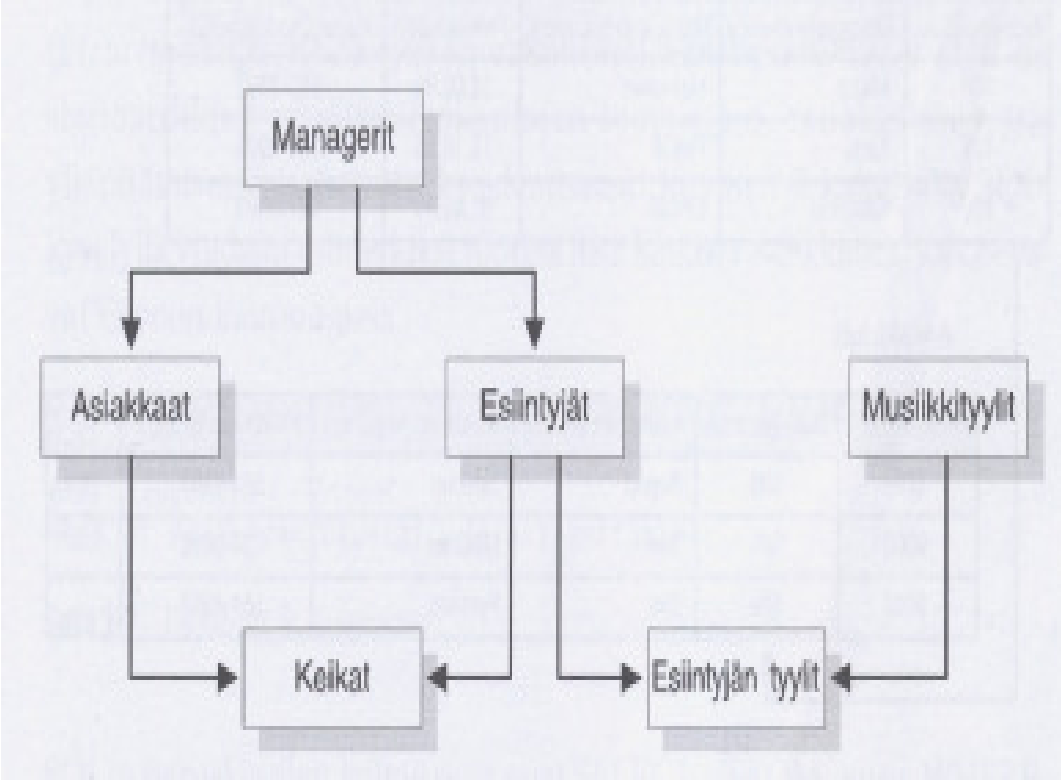
Verkkotietokantamallissa haetaan tietoa kulkemalla sopivien joukkorakenteiden läpi. Toisin kuin hierarkkisessa tietokantamallissa, jossa haku tulee aina aloittaa juuritaulusta, käyttäjä voi verkkotietokantamallissa aloittaa tiedon haun mistä taulusta tahansa ja edetä taakse- tai eteenpäin käyttäen joukkoja, jotka liittyvät toisiinsa. Verkkotietokantamallissa haut ovat hyvin nopeita, ja se onkin yksi mallin eduista. Lisäksi käyttäjä pystyy muodostamaan monimutkaisempia kyselyitä verkkotietokantamallissa kuin hierarkkisessa tietokantamallissa. (3, s. 10.)

Verkkotietokantamallin, aivan kuten hierarkkisenkin tietokantamallin, haittapuolena pidetään sitä, että jotta liikkuminen joukkorakenteiden läpi onnistuisi, tulisi käyttäjän tuntea tietokannan rakenne todella hyvin. Toisena haittapuolena pidetään sitä, että tietokannan rakenteen muuttaminen on vaikeaa, jos ei samalla muuteta asiakasohjelmia, jotka ovat vuorovaikutuksessa sen kanssa. Yhteydet verkkotietokantamallissa määritellään yksiselitteisesti joukkorakenteina. Sovellusohjelmaa muuttamatta ei voida muuttaa joukkorakennetta, koska joukkorakenteiden avulla sovellusohjelma kulkee tietojen läpi. Toisin sanoen mikäli joukkorakennetta ruvetaan muuttamaan, pitää kaikkia kyseiseen joukkorakenteeseen sovellusohjelmalla tehtyjä viittauksia muokata. (3, s. 10–11.)

2.4 Relaatiotietokantamalli

Relaatiotietokantamalli perustuu matematiikkaan ja tarkemmin katsottuna kahteen haaraan siinä, joukko-oppiin ja ensimmäisen kertaluvun predikaattilogiikkaan. Relaatiotietokantamallissa tiedot tallennetaan relaatioina, jotka näkyvät käyttäjille tauluina. Kaikki relaatiot muodostavat attribuuteista, eli kentistä, ja monikoista, eli

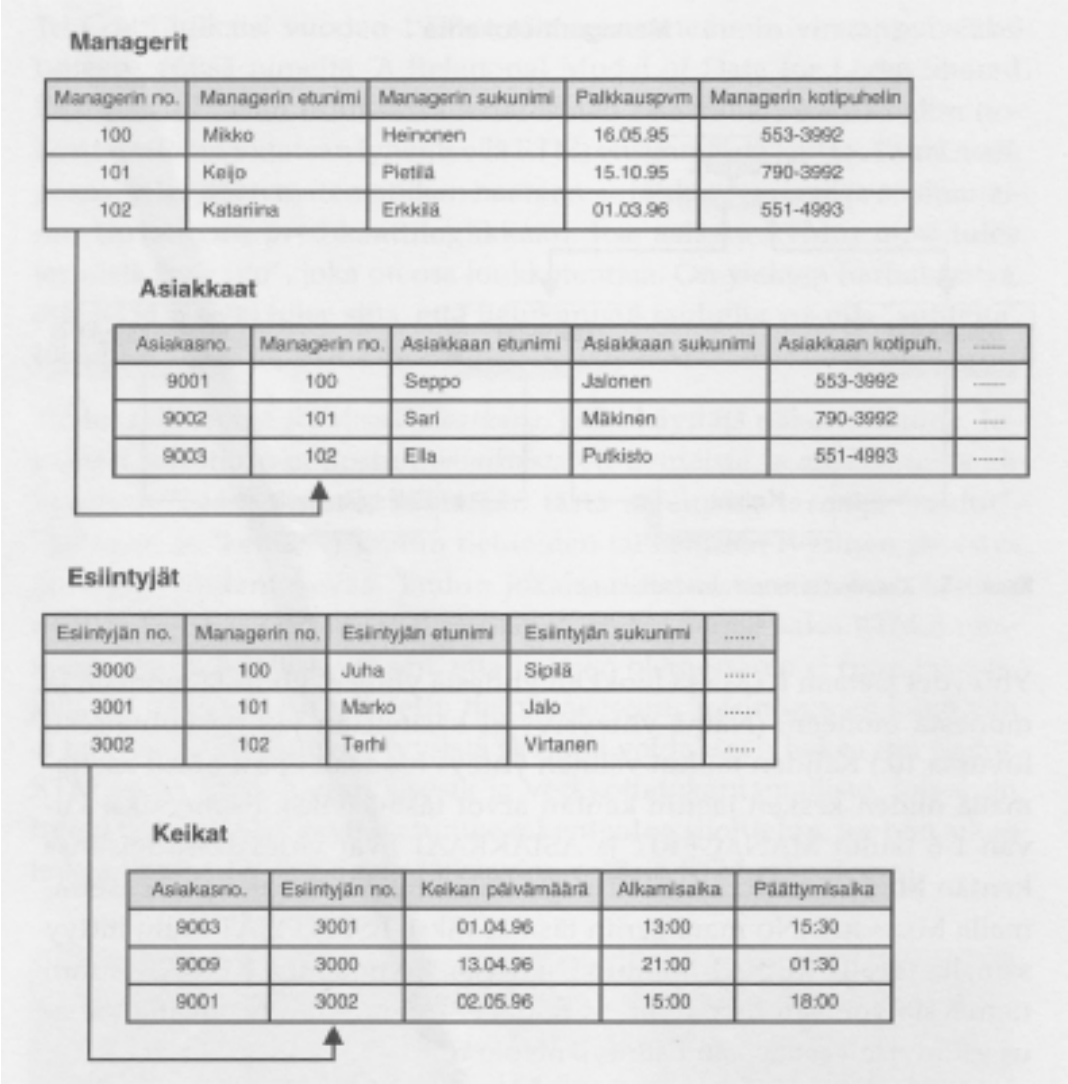
tietueista. Kenttien ja tietueiden fyysisellä järjestyksellä taulussa ei ole mitään merkitystä. Tietueet taulussa tunnistetaan kentän avulla, joka pitää sisällään muista poikkeavan arvon. Nämä kaksi ominaisuutta relaatiotietokantamallissa mahdollistavat sen, että tapa, jolla tiedot on fyysisesti tallennettu tietokoneeseen, ei vaikuta tiedon olemassaoloon. Toisin sanoen tietueen fyysistä sijaintia ei tarvitse tietää, kun haetaan tietoa. Juuri tässä relaatiotietokantamalli eroaa hierarkkisesta ja verkkotietokantamallista, joissa pitää tuntea tietokannan rakenne hakuja tehtäessä. (1, s. 7; 2, s. 4; 3, s. 12.)



Kuva 4. Relaatiotietokantamallin kaavio (3, s. 13).

Relaatiotietokantamallissa yhteydet jaetaan luokkiin ”monesta moneen”, ”yhdestä moneen” ja ”yhdestä yhteen”. Jotta taulujen välinen yhteys saadaan luotua, asetetaan niiden kesken jaettujen kenttien arvot epäsuorasti täsmääviksi.
Relaatiotietokantamallissa voidaan tietoa hakea melkein millä tavalla hyvänsä, käyttäjän pitää vain tuntea tietokannassa olevien taulujen väliset yhteydet. Tietoa voidaan siis hakea suoran yhteyden omaavista tauluista tai tauluista, joilla on epäsuora yhteys.

Tietoa haetaan siis valitsemalla halutut taulut ja kentät. Helpoin tapa tiedon etsimisessä on käyttää SQL-kieltä. SQL:ää pidetään standardikielenä relaatiotietokantojen muokkaamisessa, ylläpitämisessä, tietojen kyselemisessä ja luomisessa. (3, s. 13,15.)



Managerit

Managerin no.	Managerin etunimi	Managerin sukunimi	Palkkauspvm	Managerin kotipuhelin
100	Mikko	Heinonen	16.05.95	553-3992
101	Keijo	Pietilä	15.10.95	790-3992
102	Katariina	Erkkilä	01.03.96	551-4993

Asiakkaat

Asiakasno.	Managerin no.	Asiakkaan etunimi	Asiakkaan sukunimi	Asiakkaan kotipuh.	.....
9001	100	Seppo	Jalonen	553-3992	.....
9002	101	Sari	Mäkinen	790-3992	.....
9003	102	Ella	Putkisto	551-4993	.....

Esiintyjät

Esiintyjän no.	Managerin no.	Esiintyjän etunimi	Esiintyjän sukunimi	.....
3000	100	Juha	Sipilä	.....
3001	101	Marko	Jalo	.....
3002	102	Terhi	Virtanen	.....

Keikat

Asiakasno.	Esiintyjän no.	Keikan päivämäärä	Alkamisaika	Päättymisaika
9003	3001	01.04.96	13:00	15:30
9009	3000	13.04.96	21:00	01:30
9001	3002	02.05.96	15:00	18:00

Kuva 5. Relaatiotietokantamallissa olevia tauluja, joilla on yhteyksiä toisiinsa (3, s. 14).

Relaatiotietokantamallissa on monia etuja, joista neljä ehkä tärkeintä ovat

- datan looginen ja fyysinen riippumattomuus tietokantasovelluksissa
- tietojen taattu yhtäpitävyys ja oikeellisuus
- sisäänrakennettu monitasoinen eheys
- tietojen haun helppous.

Datan looginen ja fyysinen riippumattomuus tietokantasovelluksissa tarkoittaa sitä, että tietokantaohjelmiston myyjän tekemät fyysiset muutokset tietokannan toteutukseen tai käyttäjän tekemät muutokset tietokannan loogiseen rakenteeseen eivät vaikuta sen päälle rakennettuihin sovelluksiin haitallisesti. Tietojen taattu yhtäpitävyys ja oikeellisuus tarkoittavat sitä, että kaikki tiedot ovat oikeita ja yhtäpitäviä, koska tietokantaan voidaan asettaa useita eheystasoja. (3, s. 16.)

Sisäänrakennettu monitasoinen eheys puolestaan tarkoittaa sitä, että kenttätasolla tietojen eheys on rakennettu malliin sitä varten, että se varmistaa tietojen oikeellisuuden. Taulutasolla tämä tapahtuu sitä varten, ettei olisi moninkertaisia tietueita ja jotta havaittaisiin, jos pääavainten arvot puuttuvat. Yhteystasolla tämä varmistaa sen, että kahden taulun välillä on voimassaoleva yhteys, ja liiketoimintatasolla tämä varmistaa, että itse liiketoiminnan suhteen tiedot ovat oikeita. Tietojen haun helppoudella tarkoitetaan, että käyttäjä voi hakea tiedot jostain tietystä taulusta tai ihan mistä tahansa muusta taulusta, joihin siitä on yhteys. (3, s. 16.)

2.5 SQL

SQL (Structured Query Language) on relaatiotietokantojen käyttöön standardoitu kieli. SQL-kieltä voidaan käyttää tietokannan rakenteen muuttamiseen ja määrittelyyn, tietojen muuttamiseen, lisäämiseen ja poistamiseen, kyselyjen tekemiseen ja tapahtumankäsittelyn ohjaamiseen. SQL-kieli sisältää myös seuraavat ominaisuudet: upotetun SQL:n ja kohdistimien hallitsemisen ja poistamisen sekä turvallisuuden ja valtuuksien hoitamisen. Mikäli tietokannan rakennetta tai siihen kohdistuvia operaatioita halutaan muuttaa, tulee ennen sitä huolellisesti suunnitella tehtävä työ. Jos esimerkiksi rakennetta ei saada kerralla määriteltyä oikein, on sen muuttaminen oikeanlaiseksi jälkikäteen hyvin työlästä. SQL-kielen yleisimmin käytetty muoto on sillä suoritettavat kyselyt. Tietokantojen käsittely SQL:llä on pääasiassa erilaisten yhteenvetojen, kyselyiden ja raporttien luomista tietokannasta löytyvästä tiedosta. Kuvassa 6 on esitetty perus-SQL-kysely tietokantaan ja samalla tieto tallennetaan MEAS_TMP-tauluun. (1, s. 10; 2, s. 37–38.)

```
DELETE  FROM MEAS_TMP
INSERT INTO MEAS_TMP


SELECT     UNITS, DATA, NAME
FROM       MEAS_NUMERICLIMIT
```

Kuva 6. Perus SQL-kysely.

SQL on voimakas ja monipuolinen tietokannan käsittelykieli. Se on myös niin sanottu ei-proseduraalinen kieli eli, se kertoo, mitä tietoa halutaan etsiä eikä miten se tehdään. SQL:n peruskomentoja ovat SELECT...FROM, CREATE, INSERT ja DELETE. Jotta SQL-kieltä voitaisiin hyödyntää täydellisesti, on itse tietokannan oltava suunniteltu hyvin. SQL-kielellä on omanlaisensa rakenne. Alikyselyitä tehtäessä tulee muistaa, että koodia luetaan alhaalta ylöspäin. Harjaantuneet käyttäjät kykenevät tekemään monimutkaisia kyselyitä tietokantaan. (1, s. 10–11; 2, s. 38–39.)

3 Monimuuttujamenetelmät

3.1 Yleistä

Monimuuttujamenetelmät kuuluvat tilastollisiin analysointimenetelmiin, joilla tarkastellaan yhdenaikaisesti useiden eri muuttujien välisiä yhteyksiä. Monimuuttujamenetelmien päätarkoituksena on pelkistää laajan tai monimutkaisen aineiston tuottamaa informaatiota. Kun analysoitavissa kohteissa on useita kymmeniä muuttujia, rupeaa niiden suhteiden selvittäminen toisiinsa yksitellen olemaan melkein mahdotonta. Tällöin monimuuttuja-analyysi on suositeltava ja järkevä tapa saada hyvä katsaus analysoitavaan kohteeseen. Tarkoituksena tutkimuksessa saattaa olla etsiä esimerkiksi jokin kombinaatio, joka ennustaa muuttujan arvoja mahdollisimman tarkasti. Menetelmänä tähän sopisi regressioanalyysi. Toisaalta päämääränä saattaa olla suuren muuttujajoukon sisältämän tutkimusaineiston mahdollisimman säästeliäs kuvaaminen, jolloin voitaisiin käyttää faktorianalyysia. (4, s. 243.)

Monimuuttujamenetelmien rakenteeseen sisältyy monia monimutkaisia matemaattisia laskentaoperaatioita, ja teoria matemaattisille laskentaoperaatioille tunnetaan ainoastaan, kun vahvat rajoitukset ovat voimassa. Onneksi tietokoneiden ohjelmistot on valjastettu lähes kaikkia mahdollisia käyttötarpeita varten. Toisaalta kun käytetään tilasto-ohjelmia, pitää olla tarkkana, ettei sorru menetelmien käyttöön väärässä yhteydessä. Tämän vuoksi monimuuttujamenetelmiä tulisi käyttää vasta, kun ollaan varmoja muuttujia koskevista vaatimuksista. Useimmiten monimuuttujamenetelmissä on mukana useita selittäjiä ja yksi selitettävä muuttuja. Monimuuttajamenetelmät ovat yleensä parametrisiä eli jakaumasta riippuvia menetelmiä. Kun valitaan, menetelmää tulee kiinnittää huomiota muuttujien mitta-asteikkoon ja siihen, ovatko muuttujat jatkuvia. (4, s. 243–244.)

3.2 Pääkomponenttianalyysi

Pääkomponenttianalyysissa muodostetaan alkuperäisistä muuttujista uusia muuttujia. Uusia muuttujia kutsutaan pääkomponenteiksi, ja ne ovat lineaarisia lausekkeita alkuperäisistä muuttujista. Muuttujat ovat jatkuvia, eikä jakoa selittäviin ja selitettäviin tehdä, eli ne ovat kvantitiivisia. Mikäli aineistossa on vaihtelua, se pyritään selittämään keskenään korreloimattomia pääkomponentteja apuna käyttäen. Toisin sanoen ensimmäisellä pääkomponentilla selitetään mahdollisimman paljon alkuperäisessä aineistossa olevasta vaihtelusta, minkä jälkeen toinen pääkomponentti paneutuu jäljelle jäävään vaihteluun ja selittää siitä mahdollisimman paljon. Tätä prosessia jatketaan niin kauan kuin tarvitaan. Tavoitteena olisi saada selitettyä 70–90 % kokonaisvaihtelusta. (4, s. 247–248; 5, s. 57.)

Pääkomponenttien uusien muuttujien tuottamaa dataa käytetään usein seuraavien analyysien lähtöarvoina. Kyseisellä menetelmällä on helppo tapa saada muuttujien määrää huomattavasti pienemmäksi alkuperäiseen määrään verrattuna. Uudet pääkomponenteista saadut muuttujat eivät myöskään korreloi keskenään. Hankalaksi menetelmän tekee sen tulkinnan vaikeus, vaikka lopputulos vaikuttaisikin sopivalta laskennallisesti. Myös pienet aineistot tuottavat hankaluuksia, koska ne tuovat esiin lähinnä vain satunnaisvaihtelua. Tämän takia faktorianalyysilla usein täydennetään pääkomponenttianalyysia. (4, s. 247–248; 5, s. 57.)

3.3 Faktorianalyysi

Tavoitteet faktorianalyysissä ovat samantyyliset kuin pääkomponenttianalyysissä. Faktorianalyysinkin tarkoituksena on saada malli muuttujien kokonaisvaihtelusta mahdollisimman pienellä muuttujien määrällä. Faktorianalyysissä on myös tapana korostaa sitä, että se on kovarianssipainotteinen eli selittävänä muuttujana on jatkuva muuttuja ja mukana on myös ryhmittelymuuttuja. Pääkomponenttianalyysi puolestaan on varianssipainotteinen menetelmä eli tutkitaan ryhmittelevän muuttujan vaikutusta jonkin kvantitatiivisen muuttujan vaihteluun. Faktorianalyysimallit on yleensä tarkoitettu vähintään välimatka-asteikon tasolla oleville muuttujille, toisaalta mallia

käytetään myös usein järjestysasteikollisille muuttujille. Tulevat uudet muuttujat, faktorit, ovat lineaarikombinaatioita alkuperäisistä muuttujista, kuten pääkomponenttianalyysissä. Tärkein ero näiden kahden menetelmän välillä on se, että pääkomponenteillä, jotka ovat keskenään korreloimattomia, on tarkoituksena selittää mahdollisimman paljon alkuperäisten muuttujien sisältämästä kokonaisvaihtelusta. Faktorianalyysissä sen sijaan pyritään kartoittamaan mahdollisimman paljon muuttujien välistä vaihtelua. Muuttujien kokonaisvaihtelu jaetaankin faktorianalyysissä kahteen osaan, ominaisvaihteluun ja yhteisvaihteluun. (4, s. 248, 252-253; 5, s. 75.)

Faktorianalyysi perustuu tarkkaan malliin, jolla yritetään löytää havaittujen muuttujien taustalla piileviä tekijöitä, niin sanottuja piilomuuttujia. Kun faktorit on selvitetty, niiden tulkinta toteutetaan tutkimalla, mitkä alkuperäisistä muuttujista ovat kyseisen faktorin kanssa eniten korreloituneita. Jokaiselle havaintoyksikölle kyetään laskemaan faktoripisteet, jotka kuvaavat arvoja piilomuuttujille. Yleisesti havaitut muuttujat $X_1,X_2,...,Xp$ esitetään faktorianalyysissä seuraavanlaisen kaavan (1) avulla:

$$X_i = \mu_i + a_{i1}F_1 + a_{i2}F_2 + ... + a_{ir}F_r + U_i \text{ missä i = 1,2,...,p}$$

eli

$$X = \mu + AF + U \,, \tag{1}$$

Tässä pxr-matriisi A on faktorimatriisi. Yhteisfaktoreita ovat muuttujat $F = (F_1,F_2,...,F_r)$, ja yleisesti niitä on vähemmän kuin alkuperäisiä muuttujia r<p. Ominaisfaktoreita ovat puolestaan muuttujat $U = (U_1,U_2,...,U_p)$. Edellä mainittuun malliin tehdään yleensä seuraavanlaisia olettamuksia, jotka puolestaan antavat kovarianssimatriisille $\sum$ tietynlaisen rakenteen:

1. $X \sim N(\mu, \sum)$ missä $\sum > 0$.
2. $F \sim N(0,\Phi)$.
3. $U \sim N(0,\Psi^2)$ missä Ψ^2 on alkioiden $\Psi^2_1, \Psi^2_2,..., \Psi^2_p$ lävistäjämatriisi.
4. Ominaisfaktorit U ovat yhteisfaktoreista F riippumattomia.
5. r(A) = r.

Kyseiset olettamukset antavat siis rakenteen kovarianssimatriisille $\sum$. Laskemalla $\sum$ ulos saadaan esille faktorianalyysin perusyhtälö (2):

$$\sum = E\left[(X-\mu)(X-\mu)'\right] = E\left[(AF+U)(AF+U)'\right] = E\left(AFF'A'\right) + E\left(UU'\right)$$
$$\sum = A\Phi A' + \Psi^2 . \quad (2)$$

(4, s. 248; 5, s. 75–76.)

Toisinaan faktorianalyysin tulokset kyseenalaistetaan, koska analyysiin saattaa liittyä useita subjektiivisiä valintoja. Jos faktoreita on kaksi tai enemmän, mikä tahansa lineaarinen muunnos alkuperäisessä faktorissa tuottaa matemaattisesti yhtä hyvän tuloksen. Lineaarista muunnosta pidetään yleisesti koordinaatiston kiertämisenä. Kyseisen niin sanotun rotaation tarkoituksena on löytää tulkinnallisesti helpoin ratkaisu vaihtoehdoista. Mikäli näin ei tehdä, on vaarana, että ratkaisuun päästään muuttelemalla faktorien määrää ja kokeilemalla eri rotaatioita. Niin sanottu rotaatiomahdollisuus syntyy, koska faktorit F eivät aina määräydy yksikäsitteisesti olettamuksista ja perusyhtälöstä (2). Toisin sanoen jos $F^* = T'F$, missä T on säännöllinen rxr-matriisi, siitä seuraa, että $F^* \sim N(0,T'\Phi T)$. Jos oletetaan, että $\Phi^* = T'\Phi T$ ja $A^* = A(T^{-1})'$, voidaan muuttujat X esittää seuraavanlaisella kaavalla (3):

$$X = \mu + A^* F^* + U . \quad (3)$$

(4, s. 248; 5, s. 76, 80.)

Edellä olleiden määräysten perusteella muodostuu perusyhtälöstä (4) seuraavanlainen:

$$\sum = A\Phi A' + \Psi^2 = A^*\Phi^* A^{*'} + \Psi^2 . \quad (4)$$

Tästä huomataan, että jokainen $F^* = T'F$ ja sitä vastaava $A^* = A(T^{-1})'$ antaa myös ratkaisun mahdollisuuden. Tästä syntyvää ongelmaa kutsutaan rotaatio-ongelmaksi. Toisin sanoen ratkaisujen, jotka ovat teknisesti yhtä hyviä, joukosta on lupa valita helpoiten tulkittavat. Loppujen lopuksi on tehtävänä kuitenkin määrätä suureet r, Ψ^2, A ja T. Monesti tämän lisäksi halutaan vielä arvioida havainnoittain faktorien arvot, eli

suoritetaan niin sanottujen faktoripistemäärien laskeminen. Yleensä tätä varten tehdään vielä yksinkertaistava olettamus (jatkuu sivulta 22):

6. $\Phi = I$.

Faktorit oletetaan siis korreloimattomiksi keskenään. Tämän seurauksena perusyhtälö (5) saadaan muotoon:

$$\sum = AA' + \Psi^2 . \tag{5}$$

Lisäksi, koska vain ortogonaaliset kelpaavat rotaatiomatriiseiksi T, antaa matriisi A faktorien ja muuttujien väliset kovarianssit, koska

$$\operatorname{cov}(X, F) = E\left[(X - \mu)F'\right] = E\left[(AF + U)F'\right] = A . \tag{6}$$

Tämä pätee erityisesti, jos $\sum = P$, $\rho(X_i,F_j) = a_{ij}$ eli A sisältää faktorien ja muuttujien välillä olevat korrelaatiokertoimet. (4, s. ; 5, s. 76, 80.)

Faktorianalyysi antaakin usein hyvin karkean yleiskuvan tutkittavaan ongelmaan liittyvistä tekijöistä sekä niiden välisistä suhteista. Faktorianalyysi sopiikin paremmin esianalyysiksi. Sen käyttöä voidaan verrata esimerkiksi mielipidekyselyssä samoihin asioihin liittyvien kysymysten ryhmittelyssä. Samaan ryhmään kuuluvilla vastausten välillä huomataan usein korrelaatiota. Tästä saadun korrelaatiomatriisin avulla kyetään etsimään kysymyksiä samansuuntaisilla vastauksilla. Faktorianalyysillä kyetään kuitenkin ohjelman avulla itsenäisesti etsimään parhaiten korreloituvat muuttujat ja muodostamaan niistä faktoreita. (4, s. 248–249.)

3.4 Regressioanalyysi

Regressioanalyysillä pyritään hakemaan paras mahdollinen riippumattomien muuttujien yhdistelmä ennustettaessa yhtä riippuvaa muuttujaa. Lähtökohtana regressioanalyysissä on suhde- ja välimatka-asteikon tasoiset muuttujat. Myös nominaali- ja järjestysasteikolliset muuttujat ovat mahdollisia, mikäli niistä muodostetaan niin sanottuja dummy-muuttujia. Esimerkiksi nominaaliasteikon tasoisista muuttujista kyetään muodostamaan dummy-muuttujia siten, että koodataan se 0:lla ja 1:llä. Taulukko 1 esittää, kuinka dummy-muuttujat muodostetaan koodaamalla.

Taulukko 1. Dummy-muuttujien muodostus koodaamalla.

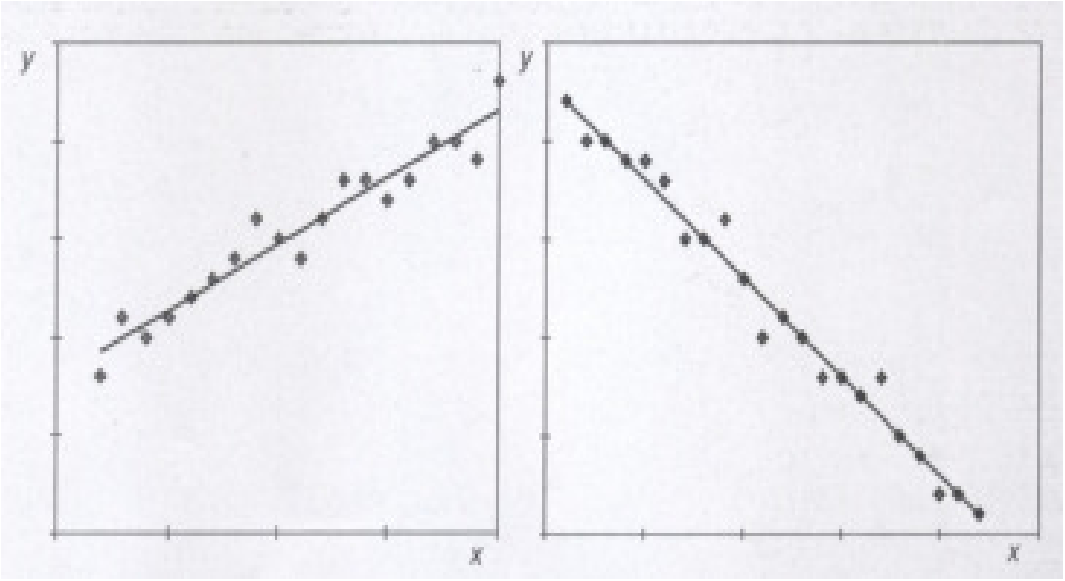
Dummy-muuttuja	Osasto 1	Osasto 2	Osasto 3
dum1	0	1	0
dum2	0	0	1

Kyseessä on siis muiden muuttujien lisäksi kaksi dummy-muuttujaa. Vertailuryhmänä taulukossa 1 toimii osasto 1, jonka regressioyhtälössä kumpikaan dummy-muuttuja ei ole mukana. Osaston 2 regressioyhtälössä dum2 saa arvon 0, ja dum1 arvon 1, joten mukana on vain dum1. Osaston 3 kohdalla dum1 saa arvon 0, ja dum2 puolestaan arvon 1, jolloin mukana on siis vain dum2. Näin pystytään siis esimerkiksi kuvaamaan muuttujien välisiä eroja. (4, s. 236–237.)

3.4.1 Regressiosuora

Mikäli kahden eri muuttujan välillä todetaan olevan selvästi lineaarista riippuvuutta ja muuttujan (y) käyttäytymistä kyetään selittämään muuttujan (x) avulla, pystytään riippuvuutta kuvaamaan niin sanotun regressiosuoran avulla. Tilannetta on helppo havannoillistaa piirtämällä ensin hajontakaavio, jossa muuttujat ovat x-ja y-akseleilla. Hajontakaaviosta valitun yksittäisen pisteen koordinaatit, toiselta nimeltään havaintopari (x, y), kertovat yhden tilastoyksikön vastausarvot kahteen tutkittavaan muuttujaan. Suoran yhtälö muodostetaan niin, että havaintoparien muodostama

pistejoukko sovitetaan siten, että y-akselin suuntaisten pisteiden ja suoran välisten etäisyyksien neliöiden summa on mahdollisimman pieni. (4, s. 92, 238.)



Kuva 7. Regressiosuora sovitettuna havaintopisteisiin (4, s. 91).

Kuten kuvasta 7 näkyy, on havaintopisteitä molemmin puolin suoraa. Regressiosuorasta käytetään myös nimitystä ”pienimmän neliösumman suora”. Pienimmän neliösumman suora on siis saanut nimensä siitä tavasta, jolla suoran yhtälö muodostetaan. Muodostaminen tapahtuu niin, että minimoidaan havaintopisteiden laskettujen etäisyyksien neliöiden summa suorasta. Näin ollen suora (7) saa muodon:

$$y = a + bx \tag{7}$$

Kaavassa (7) oleva kulmakerroin b ja vakio a voidaan laskea seuraavanlaisilla kaavoilla (8) ja (9) ulos:

$$b = \frac{n\sum x_i\ y_i - \left(\sum x_i\right)\left(\sum y_i\right)}{n\sum x_i^2 - \left(\sum x_i\right)^2} \tag{8}$$

$$a = \frac{\sum y_i - b\sum x_i}{n} \tag{9}$$

Kerrointa b kutsutaan myös regressiokertoimeksi, ja se kertoo kuinka paljon arvo y muuttuu keskimäärin, jos arvo x muuttuu yhden yksikön. Vakio a puolestaan ilmoittaa y-akselin ja suoran leikkauspisteen. (4, s. 92–93, 238.)

Saadun mallin hyvyyttä arvioidaan selitysasteen (korrelaatiokertoimen neliö) perusteella. Selitysaste kertoo siis, miten suuri osa voidaan selittää selitettävän muuttujan avulla muuttujan y vaihtelusta. Selitysasteen tulisi olla korkea, ainakin 0,6, jotta mallin avulla voitaisiin tehdä ennusteita selitettävälle muuttujalle. Tulee kuitenkin muistaa, että mallia voidaan käyttää ainoastaan lähellä niitä muuttujien arvoja, joita on käytetty sen muodostamiseen. On myös hyvä muistaa, että saatu malli kuvastaa ainoastaan keskimääräistä käyttäytymistä ilmiössä eikä tarkastelualueen ulkopuolella välttämättä päde hyväkään malli. Kun kyseessä on yhden selitettävän muuttujan tapaus, on kuvasta suhteellisen helppo lukea, onko suora sopiva pistejoukkoon vai tulisiko käyttää käyräviivaista mallia. (4, s. 238.)

3.4.2 Usean selittäjän regressioanalyysi

Regressioanalyysissä voi olla useita selitettäviä muuttujia, eikä malli ole aina välttämättä lineaarinen. Tämän takia usean selittäjän malliin lisätään loppuun virhetermi ε, jolloin lineaarinen usean selittäjän malli saa muodon:

$$y = a + b_1 x_1 + b_2 x_2 + ... + b_n x_n + \varepsilon\,. \tag{10}$$

Kuten yhden muuttujan selitettävässä mallissa, kyetään tässäkin tekemään ennusteita selitettävälle muuttujalle. Usean selitettävän muuttujan lineaarisiin regressiomalleihin liittyy kuitenkin monia edellytyksiä:

- Selittäjät, jotka ovat mukana, ovat järkeviä.
- X:n arvoista riippuvat lineaarisesti y:n arvot.
- Selittävät muuttujat eivät korreloi keskenään, eli multikollineaarisuutta ei esiinny.
- Peräkkäiset havainnot aikasarja-aineistossa eivät riipu toisistaan.
- Virhetermi ja selitettävä muuttuja y ovat normaalisti jakautuneita.
- Muuttujien arvot eivät korreloi virhetermin kanssa.

Usean selittäjän mallissa tulee selitysasteen olla myös vähintään 0,6. Tärkeätä on lisäksi löytää kaikki selittäjät, jotka ovat tarpeellisia, muutoin selitysaste jää liian alhaiseksi. Selityskertoimen lisäksi esimerkiksi monet tilasto-ohjelmat muodostavat niin sanotun suhteutetun selityskertoimen, joka huomioi selitettävien muuttujien lukumäärän. Suhteutettu selityskerroin muuttuu vain, jos uusi selittävä muuttuja onnistuu parantamaan mallia. (4, s. 238–239, 251–252.)

4 Mittausdatan analysointi

4.1 Tavoite

Työn tavoitteena oli kehittää nopeampi ja helpompi toteutustapa tuotantotestauksen mittausdatan analysointiin. Siihen pyrittiin perehtymällä ensin tietokantaan, johon mittausdata oli tallennettu. Tietokanta oli iso, noin 85 gigatavua tietoa. Tietokantaan perehdyttäessä selvitettiin, miten taulut liittyvät toisiinsa, ja mitkä taulut toimivat niin sanottuina haun pullonkauloina. Tietokanta oli rakennettu ja suunniteltu nykyaikaiseksi relaatiotietokannaksi. Aluksi suoritettiin SQL-kyselyt ilman parannuksia tietylle sarjanumerolle (serialnumber) sekä tietylle ryhmänumerolle (batchnumber), ja tämän jälkeen luotiin uudet indeksit ja optimoitiin tietokanta. Lopuksi suoritettiin samat SQL-kyselyt uudelleen, ja tutkittiin, saatiinko kyselyt nopeammiksi, ja otettiin vielä yhden kyselyn tulokset analysoitaviksi.

4.2 Työn suoritus

Työn suorituksessa käytettiin ohjelmaa Microsoft SQL server ja siihen kuuluvia SQL Query Analyzeria ja SQL Server Enterprise Manageria. Tehtävät kyselyt suoritettiin SQL-kielellä. Aluksi selvitettiin, miten tietokannan taulut liittyvät toisiinsa ja mitkä ovat niiden yhteysreitit. Tämä saatiin selville tutkimalla tietokannan rakennetta ja siinä olevia liitoksia. Se, miten taulujen tiedot liittyvät toisiinsa, oli astetta haastavampi tehtävä. Periaatteessa jokaisella taululla on jokin uniikki kenttä, jonka avulla taulu voidaan yhdistää seuraavaan tauluun, ja näin saadaan esille tieto tietylle sarja- tai ryhmänumerolle. Jotta oltaisiin saatu selville kahden taulun yhteiset tiedot, haettiin molempien taulujen tiedot jollain tietyllä sarjanumerolla. Tämän jälkeen vertailtiin tuloksia ja selvitettiin, mitkä tekijät liittyvät toisiinsa, ja minkä kautta. Tietokannan rakenne on havainnollistettu kuvassa 8.

POISTETTU

Samalla kun tutkittiin tietokannan rakennetta (kuva 8), selvitettiin, mitkä tauluista ovat isoimpia ja mistä tauluista olisi hitainta hakea tietoa. Myös taulun sisältöä tarkasteltiin, niin että saatiin selville, missä tauluissa oli oleellisinta tietoa. Tämä oli tärkeää, koska tämän avulla tehtiin ehtoja SQL-kyselyyn, ja näin saatiin karsittua pois ylimääräinen ja turha tieto.

Työssä huomattiin, että taulut STEP_RESULT ja UUT_RESULT ovat hankalimpia ja että näistä STEP_RESULT oli tiedonmäärältään kaikkein isoin. Oletuksena oli tästä syystä, että juuri tämän taulun kanssa ilmenee ongelmia. Todettiin myös, että kaikkia tauluja ei tarvitse käyttää tiedon hakemisessa. Aluksi haettiin tietoa ainoastaan UUT_RESULT-taulusta seuraavanlaisella koodilla:

```
SELECT    *
FROM      UUT_RESULT
```

SELECT-komennolla valittiin haettavat kentät UUT_RESULT-taulusta. Tässä tapauksessa haettiin kaikki kentät, joita kuvaa *-merkki. Saatua tietoa oli valtavasti, ja osa siitä olikin turhaa.

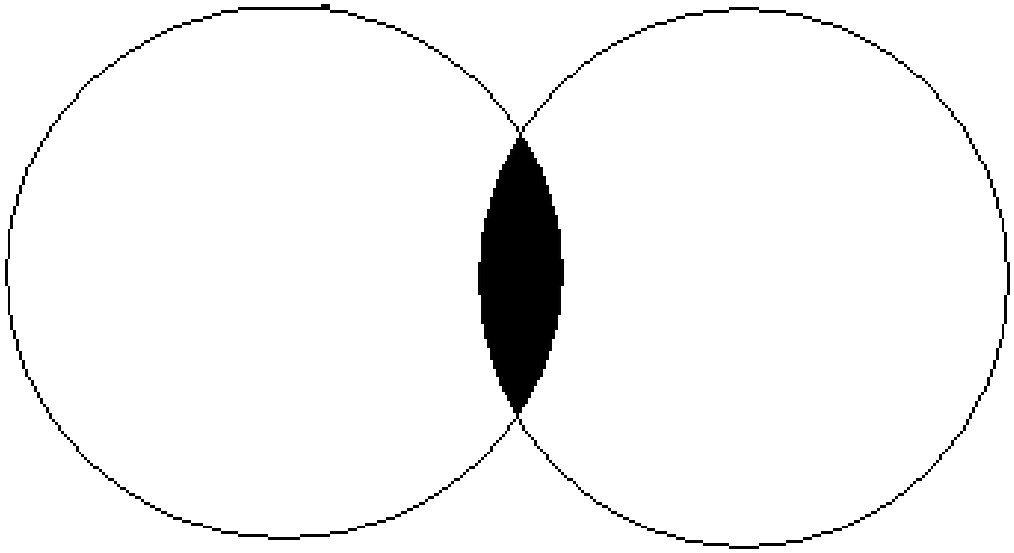
Jotta tiedon määrää saatiin pienemmäksi ja paremmin ymmärrettäväksi, lisättiin hakuun ehto, jolla haettiin vain tiettyä sarjanumeroa. Näin saatu tiedon määrä pieneni huomattavasti ja myös hakuaika nopeutui.

```
SELECT    *
FROM      UUT_RESULT
WHERE     (UUT_SERIAL_NUMBER = '633810105')
```

Edellä haettiin kaikki tieto taulusta UUT_RESULT sarjanumerolla 633810105. Saaduista tuloksista tehtiin toinen päätelmä eli se, että tarvitaan toinen ehto-lause. Sen avulla otettiin tuloksiin ainoastaan ne laitteet, jotka olivat läpäisseet testin. Nyt kysely oli muotoa:

```
SELECT      *
FROM        UUT_RESULT
WHERE       (UUT_SERIAL_NUMBER = ‘633810105’)
AND         (UUT_STATUS = ’passed’)
```

Tässä vaiheessa todettiin, että seuraavat tärkeät datat löytyvät taulusta, STEP_RESULT. Seuraavaksi selvitettiin kuinka saadaan liitettyä kyseiset kaksi taulua yhteen niin, että molempien yhteiset tiedot tulisivat esiin.



Kuva 9. Kaksi erillistä taulua, joiden leikkauspisteiden välissä on yhteinen data.

Kuvassa 9 olevat kaksi palloa kuvaavat kahta eri taulua tietokannassa, jotka molemmat sisältävät oman datansa. Näiden kahden pallon leikkauspisteiden väliin jäävä alue havainnollistaa kyseisten taulujen sisältämää yhteistä dataa, joka tässä työssä haluttiin saada esiin. Tiedon hakemista varten muodostettiin alikysely, sitä muodostettaessa tulee muistaa, että SQL-kielellä tehtyä koodia luetaan alhaalta ylöspäin eli päinvastoin kuin esimerkiksi C-kielellä kirjoitettua.

```
SELECT      *
FROM        STEP_RESULT
WHERE       (
UUT_RESULT
IN
(SELECT     ID
FROM        UUT_RESULT
WHERE       (UUT_SERIAL_NUMBER = ‘633810105’)
AND         (UUT_STATUS = ’passed’)))
```

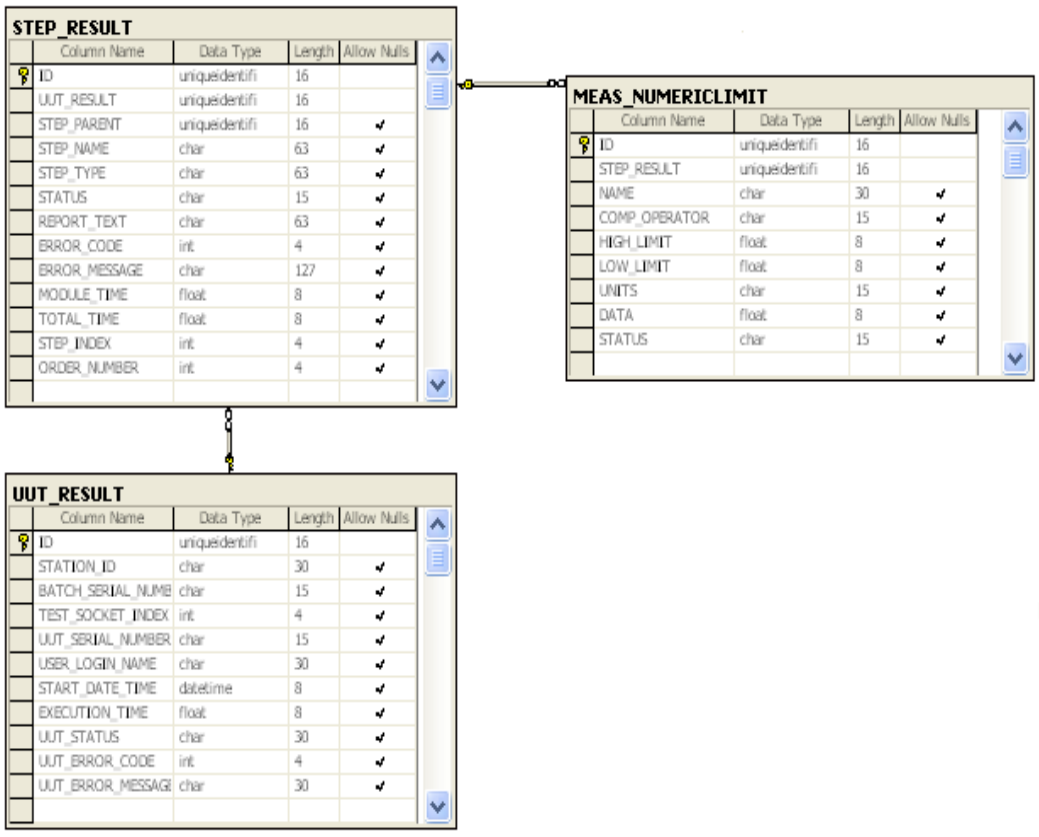
Edellä UUT_RESULT-taulusta valittiin kenttä ID, koska se sisältää uniikkia tietoa. Tämä huomattiin, kun tutkittiin tietokannan rakennetta. Kun suoritetaan alikyselyä, ei voida valita muuta kuin uniikkia tietoa sisältävä kenttä, koska muutoin tietoon saattaisi tulla päällekkäisyyksiä. IN-määreellä määriteltiin, mitkä joukot kelpuutetaan hakuun, eli tässä koodissa kaikki, mitä lukee sen alapuolella. WHERE-lause STEP_RESULT-taulun alapuolella aloittaa alikyselyn. Ylhäältäpäin luettuna ensimmäisten sulkeiden jälkeen oleva UUT_RESULT-kenttä löytyy STEP_RESULT-taulusta, ja se on linkki näiden kahden taulun välillä. Sen avulla voitiin etsiä yhteistä tietoa molemmista tauluista.

Seuraavaksi tarkoituksena oli laajentaa hakua kolmanteen tauluun. Seuraava taulu oli MEAS_NUMERICLIMIT, ja jälleen kerran piti selvittää, mikä on yhdistävä tekijä STEP_RESULT- ja MEAS_NUMERICLIMIT-tauluilla. Se onnistuikin tällä kertaa suhteellisen helposti, koska tietokannan rakenne oli suunniteltu hyvin. Uuden taulun lisääminen toteutettiin samalla kaavalla kuin edellisellä kerralla, ja näin se saatiin muotoon:

```
SELECT      *
FROM        MEAS_NUMERICLIMIT
WHERE
(STEP_RESULT IN (
SELECT
```

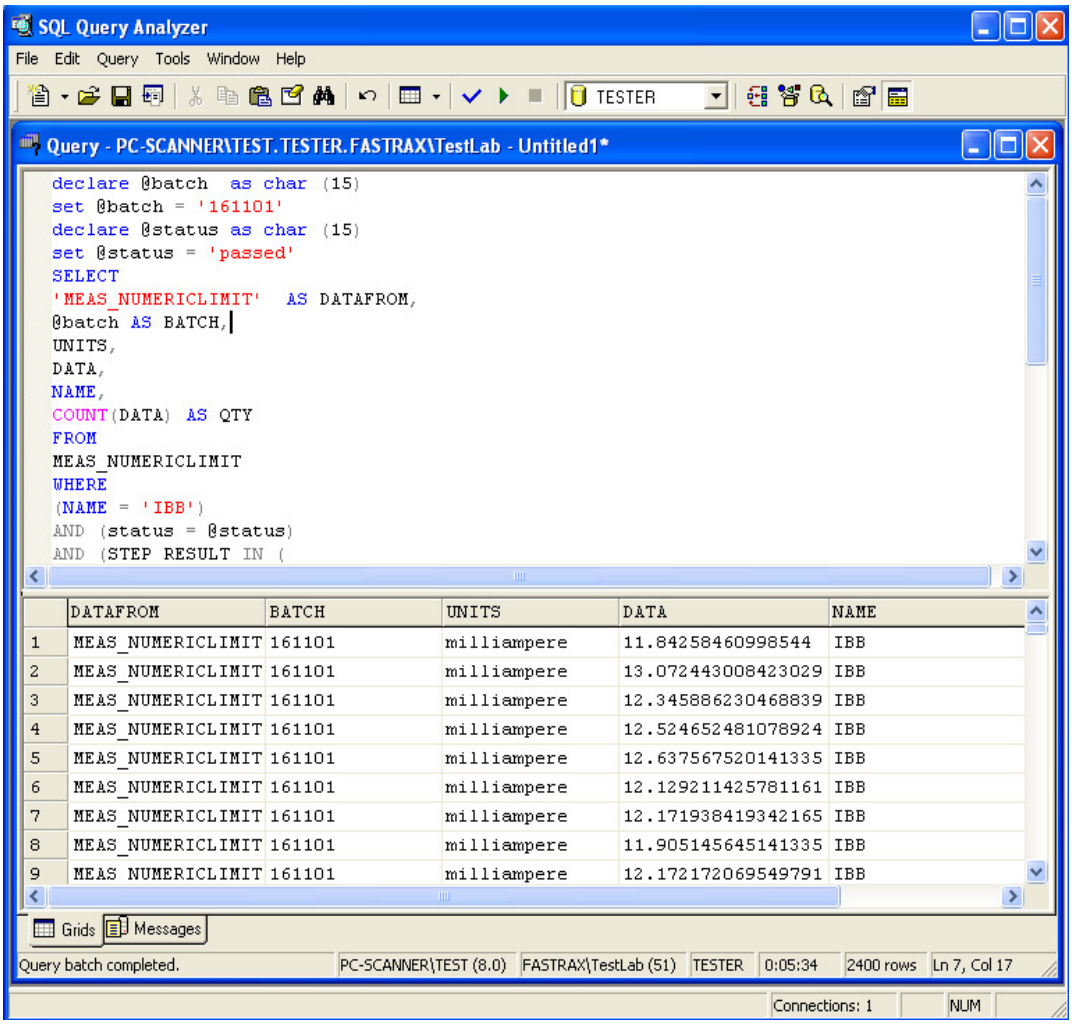
```
ID
FROM        STEP_RESULT
WHERE       (
UUT_RESULT
IN
(SELECT     ID
FROM        UUT_RESULT
WHERE
(UUT_STATUS = ‘passed’ )
AND (UUT_SERIAL_NUMBER = ‘633810105’)))
))
```

Kuvassa 10 on esitetty tiedonhaussa tarvitut taulut, kuvasta on myös luettavissa, millaisia kenttiä ne pitävät sisällään.



Kuva 10. Tiedonhaussa käytetyt taulut.

Tässä vaiheessa oli saatu suoritettua haku, jolla saatiin esille tuotteen tuotantotestauksessa tuottama mittausdata sarjanumerolla 633810105. Lopulliseen koodiin tehtiin vielä tarkennus siitä, mitä kenttiä haettiin, ja alkuun lisättiin optio, jolla voitiin helposti vaihtaa sarjanumeroa, ja sitä, otetaanko tuloksiin mukaan vain hyväksytyt vai myös hylätyt tuotteet (liite 1). Ryhmänumerolla suoritettiin sama haku, mutta siinä sarjanumeron tilalla käytettiin ehtona tiettyä ryhmänumeroa (liite 2).



Kuva 11. Koodin kirjoittamisessa käytetty SQL Query Analyzer.

Kuvassa 11 on esitetty SQL Query Analyzerin käyttöliittymä. Hakuja tehtäessä koodi kirjoitetaan valkoiselle pohjalle, ja kun kysely on valmis, painetaan yläreunasta löytyvää, vihreää, kyljellään olevaa kolmiota eli play-nappulaa. Tämän jälkeen ohjelma

aloittaa kyselyn, jos koodi on virheetön. Kun kysely on suoritettu, ohjelma tulostaa haetut tiedot kuvan alaosassa näkyvällä tavalla. Nämä tiedot pystytään jälkikäteen kopioimaan esimerkiksi Excel-tiedostoon, ja saadusta datasta voidaan toteuttaa kuvaaja.

4.3 Parannusten tekeminen tietokantaan

Kun oli saatu suoritettua haut normaalista tietokannasta, tarkoituksena oli aloittaa niin sanottu tietokannan huoltotyö. Huoltotyön tarkoituksena oli nopeuttaa tietokantaan tehtäviä hakuja ja näin säästää aikaa mahdollisia analyysejä tehtäessä. Tietokannan parantaminen aloitettiin optimoimalla se eli reindeksoimalla. Seuraavassa STEP_RESULT-taulun reindeksointikoodi (tarvittavat reindeksointikoodit liite 3):

```
Dbcc showcontig('STEP_RESULT')
Dbcc DBREINDEX('STEP_RESULT','',80)
Dbcc showcontig('STEP_RESULT')
```

Koodissa keskimmäinen komento suoritti itse reindeksoinnin taululle STEP_RESULT. Taulu, jolle reindeksointi tehtiin, on määritetty sulkeissa ensimmäisenä. Tämän jälkeen määriteltiin, mitkä indeksit reindeksoidaan. Koodissa on tässä kohtaa ainoastaan "-merkit. Se, että kyseinen kohta on jätetty tyhjäksi, tarkoittaa sitä, että kyseisen taulun kaikki indeksit reindeksoitiin. Lopussa on vielä luku 80, joka on niin sanottu fillfactor. Toisin sanoen sillä määriteltiin tietokannan täyttöaste. Tässä tapauksessa tietokannan täyttöaste oli 80/20, eli 20 prosenttia tietokannasta oli tyhjää tilaa, jonne ohjelma hakuja toteuttaessaan tallentaa tietoa hakutulosten nopeuttamiseksi.

Tietokannan optimointi nopeutti useita SQL-kyselyitä, varsinkin sarjanumeroa haettaessa tulokset olivat jo parempia. Sen sijaan ryhmänumerolla suoritettavat haut olivat edelleen liian pitkiä. Ratkaisuksi tähän luotiin uudet indeksit kahteen isoimpaan tauluun, UUT_RESULT ja STEP_RESULT. Pyrkimyksenä oli nopeuttaa näiden kahden taulun läpikäymistä, ja näin ollen saada kokonaishakuaika pienemmäksi.

Indeksit tehtiin tauluihin eri aikaan. Näin suljettiin pois mahdolliset sekaantumiset, ja samalla saatiin esille molempien indeksien luomiseen kulunut aika. Ensin tehtiin indeksi UUT_RESULT-taululle koodilla:

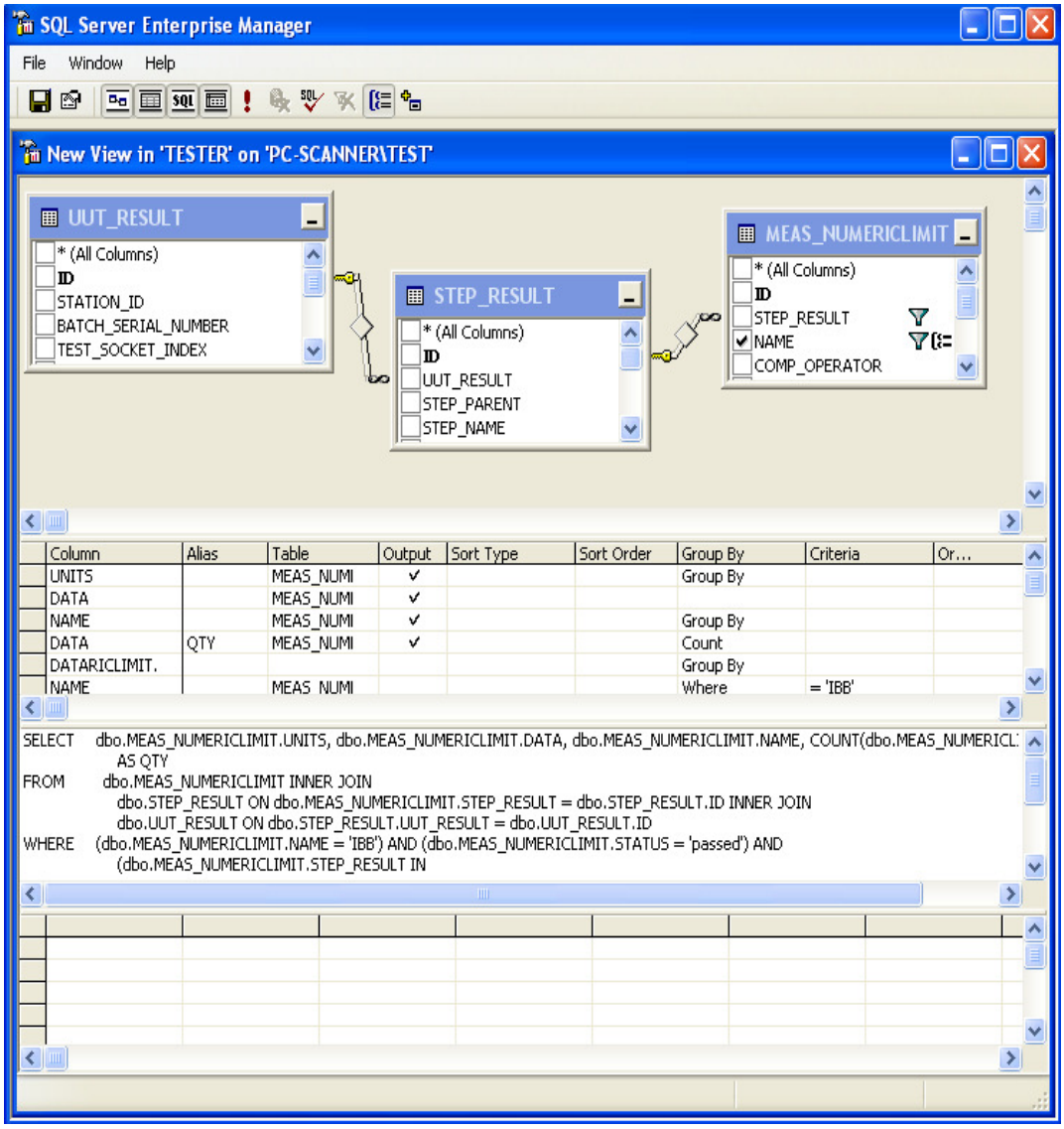
```
CREATE INDEX uut_result_col_batch_serial_number_1
ON UUT_RESULT (BATCH_SERIAL_NUMBER)
```

Luotu uusi indeksi oli nimeltään uut_result_col_batch_serial_number_1. Indeksi tehtiin siis UUT_RESULT-taululle, ja tarkasti ottaen taulusta löytyvään BATCH_SERIAL_NUMBER-nimiseen kenttään, juuri siihen missä kaikki ryhmänumerot sijaitsevat. Tämän toivottiin nopeuttavan ryhmänumeron löytymistä tietokannasta ja sitä kautta auttavan koko tietokannan parantamista.

Toisessa vaiheessa luotiin STEP_RESULT-taululle uusi indeksi. Tarkoituksena oli nopeuttaa sen ja UUT_RESULT-taulun yhteisen tiedon esille saamista. Koodi, jolla uusi indeksi luotiin, oli mallia:

```
CREATE INDEX step_result_uut_result_1
ON STEP_RESULT (UUT_RESULT)
```

Luotu uusi indeksi STEP_RESULT-taulussa oli nimeltään step_result_uut_result_1, indeksi oli luotu kenttään UUT_RESULT. Kenttä UUT_RESULT oli linkki tauluun UUT_RESULT, ja tämän linkin avulla voitiin tutkia näiden kahden taulun yhteisiä tietoja.



Kuva 12. SQL Server Enterprise Manageria käytettiin apuna hakukoodien luomisessa.

5 Analysoinnin tulokset

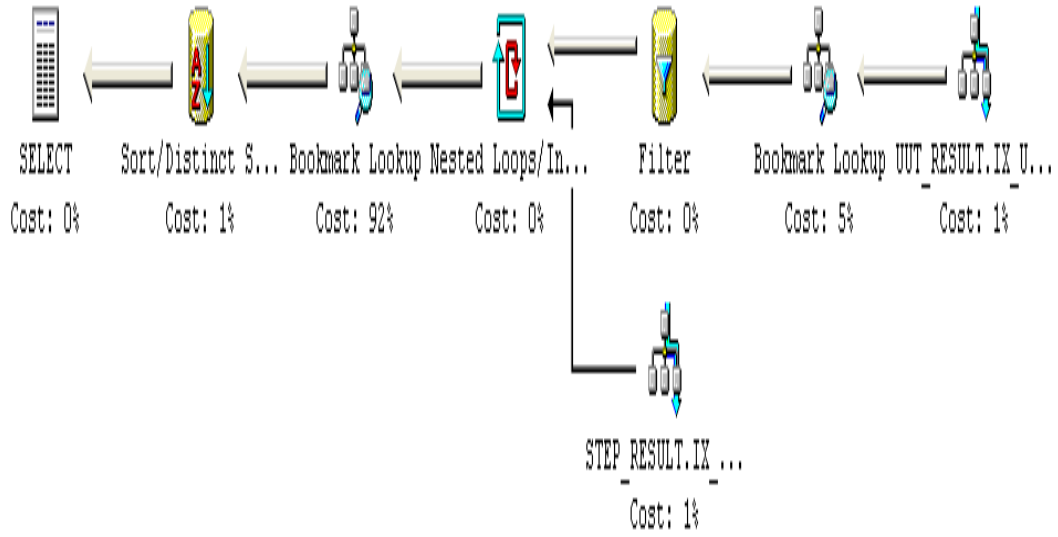
Työn tarkoituksena oli kehittää parempi ja nopeampi SQL-kysely hakemaan haluttua tietoa yrityksen omasta tietokannasta, johon tallentuu tuotantotestauksesta tuleva mittausdata. Tarkastellaan nyt, millaisia tuloksia saatiin aluksi ennen parannuksia ja miten tietokantaan ja kyselytapaan liittyvät muutokset paransivat asiaa. Tietokannan suuruus, noin 85 gigatavua, asetti omat haasteensa työn alussa, sillä hakuajat venyivät pahimmillaan useiksi tunneiksi. Toinen suuri haaste työn suorittamisessa oli selvittää, miten tietokannan taulut liittyvät toisiinsa ja miten saadaan valittua vain oleellinen tieto. Työtä helpotti tietokannan järkevä suunnittelu ja siinä oleva logiikka taulujen välillä.

Työn alussa suoritettiin kyselyt ilman parannuksia tietokantaan tai kyselytapaan, niin sanotusti tietokannan normaaliasetuksilla. Kyselyt suoritettiin kolmea taulua hyväksi käyttäen. Tuloksiksi saatiin, että hakuajat ovat todella pitkät, oikeastaan ei edes pystytty tekemään raskaita SQL-kyselyitä. Huomattiin myös, että kyselyissä pitää käyttää alikyselyä, kun edetään taulusta toiseen. Käytetyistä analysointiohjelmista SQL Server Enterprise Managerin käyttö voitiin tiedon haussa jättää pois kokonaan, koska se ei salli pitkiä hakuaikoja. Kyseistä ohjelmaa käytettiin tästä lähtien koodin oikeellisuuden ja hienosäädön tarkkailuun. Kaikki kyselyt suoritettiin SQL Query Analyzerilla, koska siinä ei ole rajoitettu hakuaikoja ja ohjelma kertoo myös kyselyn päätyttyä sen keston.

Taulukko 2. Hakuaika alussa, kun parannuksia ei ole tehty.

Kysely	Indeksointi	Optimointi	Hakuaika
Sarjanumero	ei	ei	3 tuntia
Ryhmänumero	ei	ei	5 tuntia

Taulukko 2 kuvaa hyvin, miten hankalaa oli suorittaa kysely ja koettaa löytää parempi kyselytapa, kun hakuaika oli todella pitkä. Tässä vaiheessa ei oltu tehty vielä optimointia eikä indeksointia. Syy hakuaikojen venymiseen oli myös suuri tiedon määrä tietokannassa, ja kun kyselytapa oli väärä, näkyi se heti pitkinä hakuaikoina.



Kuva 13. Hakuajan jakautuminen prosentteina kyselyn eri vaiheissa.

Kuvassa 13 on esitetty hakuajan jakautuminen prosentteina kyselyn eri vaiheissa. Suurin osa ajasta kuluu siis tiedon liittämisessä yhteen vasta kyselyn loppuosassa. Toisin sanoen kyselyssä käydään läpi kaikki mahdolliset rivit tietokannasta useaan kertaan Nested Loops -toiminnon avulla. Kun tietokanta sisältää miljoonia rivejä tietoa, hakuaika venyy. Kyselyn nopeuttamiseksi suurin osa tiedonkäsittelystä tulisi tapahtua ennen Nested Loops -toimintoa.

Tietokannan parannukseen ryhdyttiin optimoimalla aluksi tietokanta, pelkkä optimointi paransi sarjanumerokyselyä huomattavasti. Sarjanumerokyselyssä päästiinkin tavoitteeseen, toki sitä pystyisi varmasti vielä samaan nopeammaksikin. Syynä kyselyn nopeutumiseen oli, että tietokannan alkuperäiset indeksit oli suunniteltu sarjanumeroille. Alkuperäiset indeksit aktivoituivat nyt paremmin käyttöön, myös kyselytapaan kiinnitettiin enemmän huomiota. Tietokannan optimointiin kului aikaa 9 tuntia ja 30 minuuttia, samalla tietokannan koko kasvoi 85 gigatavusta 115 gigatavuun. Syynä tietokannan kasvuun oli fillfactor-asetus, joka luo tietokantaan tyhjiä paikkoja. Kyselyä suoritettaessa ohjelma hyödyntää näitä tyhjiä paikkoja tallettamalla niihin väliaikaisesti tietoa, jotta kysely nopeutuisi.

Taulukko 3. Hakuaika, kun optimointi tehty tietokantaan.

Kysely	Indeksointi	Optimointi	Hakuaika
Sarjanumero	kyllä	kyllä	alle minuutti
Ryhmänumero	ei	kyllä	5 tuntia

Taulukossa 3 on esitetty hakuajat sen jälkeen, kun tietokannan optimointi tehtiin ensimmäisen kerran. Sarjanumerokyselyn hakuaika parani huomattavasti, ja tähän aikaan voitiin olla tyytyväisiä. Kannan optimoinnin ja indeksien lisäksi nopeuteen vaikutti se, että sarjanumero kertoo vain yhden tuotteen tiedot, kun taas ryhmänumero sisältää useita tuotteita ja näin ollen myös enemmän tietoa.

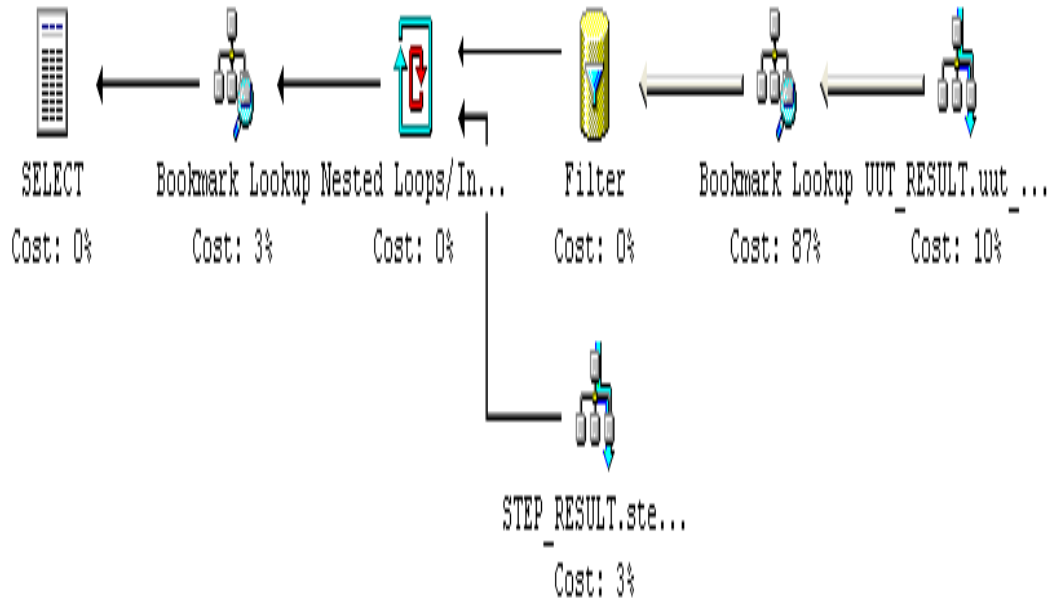
Ryhmänumeron hakuajan pienentämistä varten tehtiin uudet indeksit tauluille UUT_RESULT ja STEP_RESULT. Indeksien luominen ei kestänyt pitkään. UUT_RESULT-taulun uuden indeksin luomiseen meni vain kaksi minuuttia ja STEP_RESULT-taulun indeksin luomiseen meni noin 10 minuuttia. Kun indeksit oli luotu, optimoitiin koko tietokanta uudestaan. Nyt tietokannan optimointiin kului 8 tuntia ja 10 minuuttia. Tietokannan optimoinnin jälkeen hakuajat ryhmänumerolla paranivat huomattavasti.

Taulukko 4. Hakuajat, kun indeksointi ja optimointi tehty tietokantaan.

Kysely	Indeksointi	Optimointi	Hakuaika
Sarjanumero	kyllä	kyllä	alle minuutti
Ryhmänumero	kyllä	kyllä	~10 minuuttia

Kun vertaillaan taulukoita 2 ja 4 huomataan, että hakuajoilla on huomattava ero. Tietokannan optimoinnilla ja uusia indeksejä tekemällä voitiin parantaa huomattavasti suoritettujen kyselyiden nopeuksia. Vaikka kyselyt nopeutuivat, vei ryhmänumerokysely vieläkin turhan kauan. Ryhmänumerokyselyä koetettiin vielä nopeuttaa optimoimalla uudestaan ainoastaan uudet indeksit, mutta tällä ei havaittu olevan vaikutusta lopputulokseen.

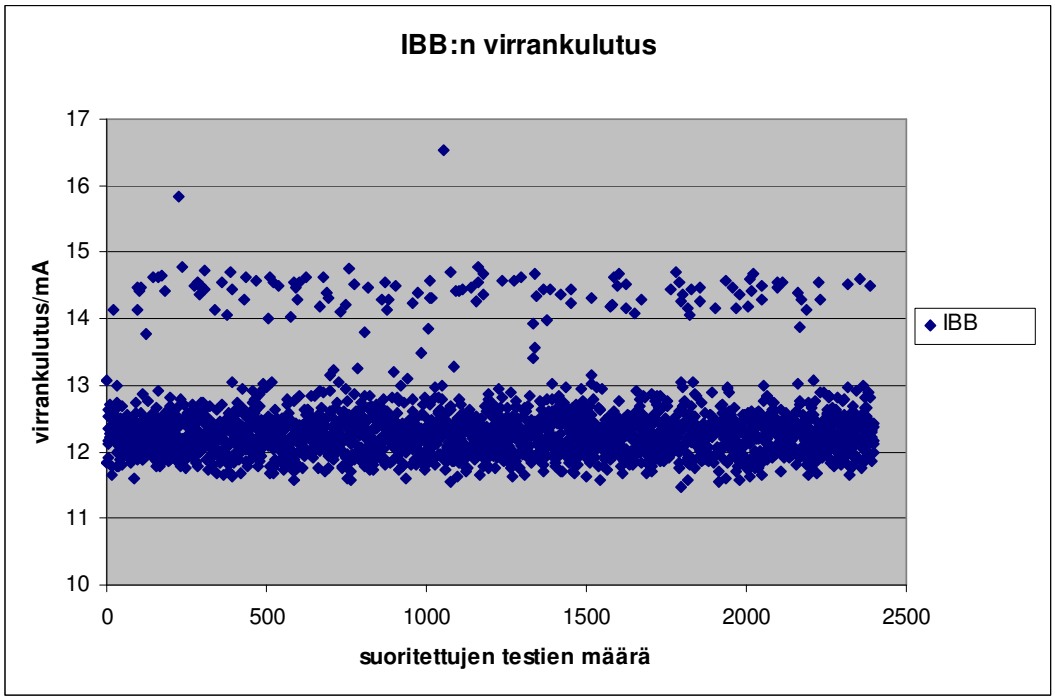
Käytössä oli vielä yksi keino, jolla voitiin saada ryhmänumerokysely nopeammaksi. Tiedettiin, että ohjelma tallentaa välimuistiin edellisiä kyselyitä, joten tätä hyväksi käyttäen suoritettiin ryhmänumerokysely kolmessa eri osassa (liite 4). Kyselyssä edettiin askel kerrallaan, kunnes saavutettiin haluttu tieto (liite 5). Eri osissa suoritettu kysely tuottikin hieman nopeamman hakuajan. Hakuaika saatiin putoamaan noin kymmenestä minuutista viiden minuutiin tuntumaan. Vaikka ryhmänumero sisältää paljon tietoa, uskon, että paremmalla indeksoinnilla ja kyselytapaan keskittymällä voidaan saavuttaa nopeampi kysely. Viiden minuutin kestoinen kysely on kuitenkin nykyään aika hidas.



Kuva 14. Hakuajan jakautuminen prosentteina indeksoinnin ja optimoinnin jälkee.

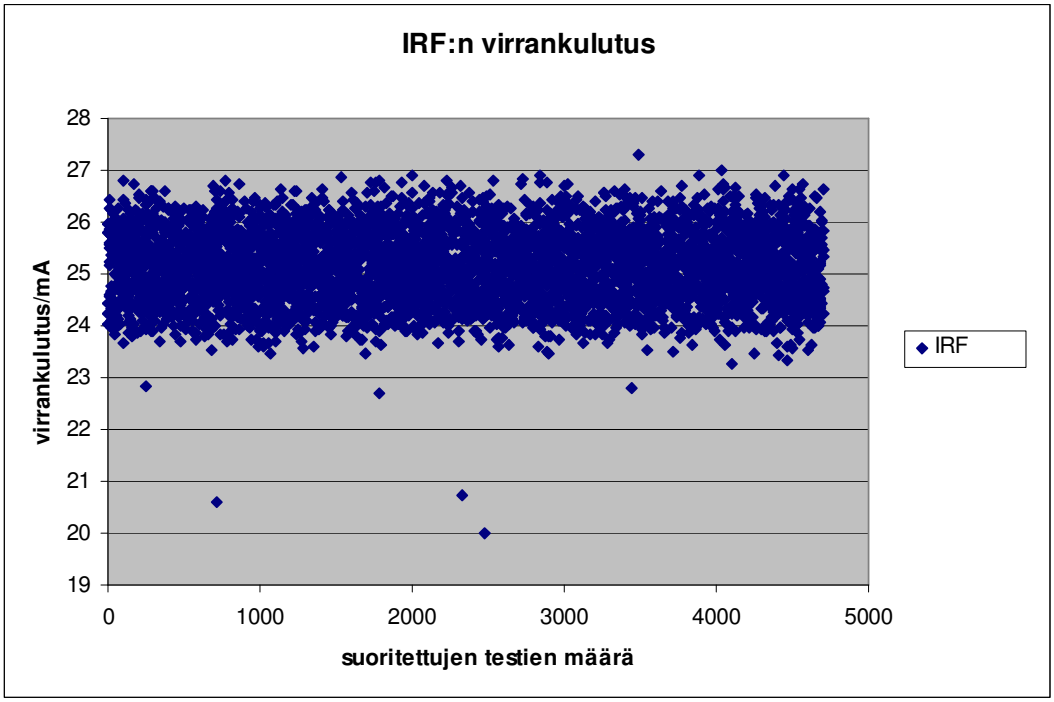
Kuvasta 14 havaitaan, että hakuajan käyttö on siirtynyt tauluun tapahtuvaksi, ja näin vain haluttu tieto liikkuu tauluista eteenpäin. Nyt Nested Loops -toiminto käy läpi paljon pienemmän tiedon määrän, ja näin hakuaika nopeutuu myös raskaissa SQL-kyselyissä.

Kyselyn ollessa kunnossa suoritettiin vielä saadun tiedon analysointi. Tiedot haettiin ryhmänumerokyselyllä (liite 2) kentästä NAME ja itse datan sisältö kahdesta eri tietueesta, IBB ja IRF. Ehtona oli myös aluksi, että tuote oli läpäissyt kyseisen testin. Vertailun vuoksi katsottiin vielä saman ryhmänumeron tuotteet, jotka eivät olleet läpäisseet testiä.



Kuva 15. IBB:n virrankulutus testattaessa.

Kuva 15 esittää tuotantotestauksesta saatua dataa IBB:n virrankulutukselle (liite 6), kun haetaan tiettyä ryhmänumeroa. Kyseisen ryhmänumeron tuotteille on suoritettu melkein 2500 hyväksyttyä testiä, ja voidaan sanoa, että virrankulutuksen hyväksyttävät rajat kulkevat 11 ja 17 milliampeerin välillä.



Kuva 16. IRF:n virrankulutus testattaessa.

Kuvassa 16 on kuvattuna IRF:n virrankulutusdata tuotantotestauksessa (liite 7). Tälle tietyn ryhmänumeron tuotteille on suoritettu lähemmäs 5000 hyväksyttyä testiä. IRF:n virrankulutuksen rajat testin läpäisemiseen näyttäisi olevan 20–28 milliampeeria. Kuvasta 16 kuitenkin huomataan, että suurin osa testattavana olleista tuotteista kuluttaa testissä 23–27 milliampeeria. Voidaan sanoa, että IRF:n virrankulutusrajat sallivat hiukan enemmän vaihtelua kuin IBB:n kuvassa 15.

Haettaessa IBB:n virrankulutusdataa (liite 8) hylättyjä tuotteita löytyi ainostaan yksi kappale, ja sen virrankulutus oli melkein 60 milliampeeria. IRF:n hylättyjä tuotteita ei löytynyt yhtään kappaletta. Tästä voitiin tehdä johtopäätös, että kyseinen ryhmänumero erä oli varsin laadukas.

6 Yhteenveto

Tuotantotestauksen mittausdatan analysoinnilla oli tarkoitus selvittää, miten tietokantaan tehtäviä kyselyitä saataisiin tehokkaammiksi ja nopeammiksi. Työssä käytettiin Fastrax Oy:n omasta tuotantotestauksesta tulevaa mittausdataa. Työn tavoitteet saavutettiin osittain. Sarjanumerokysely saatiin halutulle tasolle ja hakuaika, joka oli alussa 3 tuntia, on nyt alle 1 minuutin. Ryhmänumeroa haettaessa jäätiin tavoitteesta hiukan, vaikka senkin kysely parani ajallisesti 5 tunnista noin 5 minuuttiin. Jotta ryhmänumerokyselyyn oltaisiin tyytyväisiä, tulisi sen tapahtua minuutissa, aivan kuin sarjanumerokyselyn. Koska ryhmänumerokysely sisältää valtavan määrän dataa, voitaisiin sen kautta tapahtuvaa hakua tehostaa ainoastaan, jos hallittaisiin täydellisesti tietokanta sekä SQL-kieli. Näin pystyttäisiin luomaan uudet ja ehkä tarkemmat indeksit ja kyselyt.

Työn suorituksen kannalta ongelmallisinta oli tietokannan rakenteen ja taulujen välisten yhteyksien ymmärtäminen. Koska tietokanta oli iso, saatiin alussa tulokseksi paljon ylimääräistä tietoa. Oikean tiedon löytämistä ja seuraamista taulujen läpi helpotti tietokannan looginen suunnittelu. Tietokannan optimointi ja indeksoinnit tehtiin yöllä, koska ne olivat hitaita toteuttaa. Työn aikana tehtiin myös sellainen havainto, että kyselyitä suoritettaessa ja varsinkin raskaampia kyselyitä tehtäessä tietokannan serveri käytti lähes kaiken muistin tietokoneesta. Itse SQL-kielen kirjoittaminen ei tuottanut huomattavia vaikeuksia.

Jotta tietokannan eheys ja kyselyiden tehokkuus pysyisivät yllä tulevaisuudessa, tulee tietokannan huoltoa suorittaa tietyin väliajoin. Tietokannan optimointi tulisi tehdä ainakin kaksi kertaa vuodessa. Kun kyselyt pysyvät tehokkaina, saadaan nopeasti ajan tasalla olevaa tietoa esimerkiksi siitä, onko tuotteessa vikaa komponenteissä tai joissain muualla. Lisäksi pystytään tarkkailemaan, kuinka hyvin tuotteet pysyvät annetuissa raja-arvoissa. Työssä tehtyä SQL-koodia voidaan jatkossa käyttää pohjana erilaisille kyselyille ja sovelluksille. Jotkin tietyt kyselyt voidaan automatisoida ja niistä voidaan luoda esitys html- tai xml-kielellä verkkoon nähtäväksi halutuille tahoille.

Lähteet

1 Hovi, Ari & Huotari, Jouni & Lahdenmäki, Tapio. Tietokantojen suunnittelu & indeksointi. Porvoo: WS Bookwell, 2005.

2 Lahtonen, Tommi. SQL. Jyväskylä: Docendo Finland Oy, 2002.

3 Hernandez, Michael J. Tietokannat – suunnittelu ja toteutus. Jyväskylä: Edita, 2000.

4 Heikkilä, Tarja. Tilastollinen tutkimus. Helsinki: Edita, 2001.

5 Mustonen, Seppo. Tilastolliset monimuuttujamenetelmät. Helsinki: Helsingin yliopisto, tilastotieteen laitos, 1995.

6 Fastrax Oy. Yrityksen internetsivut. (WWW-dokumentti.) <http://www.fastraxgps.com/>. Luettu 26.5.2009.

Liite 1: Sarjanumerokysely

```
declare @status as char (15)
set @status = 'passed'
declare @serialnr as char (15)
set @serialnr = '633810105'


SELECT
'MEAS_NUMERICLIMIT'  AS DATAFROM,
@serialnr AS SERIAL,
SELECT
UNITS,
DATA,
NAME,
COUNT(DATA) AS QTY
FROM
MEAS_NUMERICLIMIT
WHERE
(NAME = 'IBB')
AND (status = @status)
AND (STEP_RESULT IN (
SELECT
ID
FROM        STEP_RESULT
WHERE     (
UUT_RESULT
IN
(SELECT     ID
FROM        UUT_RESULT
WHERE
(UUT_STATUS = @status )
AND (UUT_SERIAL_NUMBER = @serialnr)))
))
GROUP BY
NAME,
UNITS,
DATA
```

Liite 2: Ryhmänumerokysely

```
declare @batch  as char (15)
set @batch = '161101'
declare @status as char (15)
set @status = 'passed'


SELECT
'MEAS_NUMERICLIMIT'  AS DATAFROM,
@batch AS BATCH,
SELECT
UNITS,
DATA,
NAME,
COUNT(DATA) AS QTY
FROM
MEAS_NUMERICLIMIT
WHERE
(NAME = 'IBB')
AND (status = @status)
AND (STEP_RESULT IN (
SELECT
ID
FROM        STEP_RESULT
WHERE     (
UUT_RESULT
IN
(SELECT     ID
FROM        UUT_RESULT
WHERE     (BATCH_SERIAL_NUMBER = @batch)
AND (UUT_STATUS = @status )))
))
GROUP BY
NAME,
UNITS,
DATA
```

Liite 3: Reindeksoinnin koodi

```
Dbcc showcontig('MEAS_NUMERICLIMIT')
Dbcc DBREINDEX('MEAS_NUMERICLIMIT','',80)
Dbcc showcontig('MEAS_NUMERICLIMIT')

Dbcc showcontig('STEP_RESULT')
Dbcc DBREINDEX('STEP_RESULT','',80)
Dbcc showcontig('STEP_RESULT')

Dbcc showcontig('UUT_RESULT')
Dbcc DBREINDEX('UUT_RESULT','',80)
Dbcc showcontig('UUT_RESULT')
```

Liite 4: Ryhmänumerokysely askel kerrallaan

--Ensimmäinen kysely

```
declare @batch  as char (15)
set @batch = '161101'
declare @status as char (15)
set @status = 'passed'
SELECT     ID
FROM        UUT_RESULT
WHERE     (BATCH_SERIAL_NUMBER = @batch)
AND (UUT_STATUS = @status )
```

--Toinen kysely

```
declare @batch  as char (15)
set @batch = '161101'
declare @status as char (15)
set @status = 'passed'
SELECT
ID
FROM        STEP_RESULT
WHERE     (
UUT_RESULT
IN
(SELECT     ID
FROM        UUT_RESULT
WHERE     (BATCH_SERIAL_NUMBER = @batch)
AND (UUT_STATUS = @status )))
```

--Kolmas kysely

```
declare @batch  as char (15)
set @batch = '161101'
declare @status as char (15)
set @status = 'passed'
SELECT
'MEAS_NUMERICLIMIT'  AS DATAFROM,
@batch AS BATCH,
UNITS,
DATA,
NAME,
COUNT(DATA) AS QTY
FROM
```

```
MEAS_NUMERICLIMIT
WHERE
(NAME = 'IBB')
AND (status = @status)
AND (STEP_RESULT IN (
SELECT
ID
FROM        STEP_RESULT
WHERE    (
UUT_RESULT
IN
(SELECT    ID
FROM        UUT_RESULT
WHERE    (BATCH_SERIAL_NUMBER = @batch)
AND (UUT_STATUS = @status)))
))
GROUP BY
NAME,
UNITS,
DATA
```

Liite 5: Askel kerrallaan haun datat

--Ensimmäinen kysely

ID

C8230D3F-A2C4-4E0A-98D4-00132196FF83

B3621827-F6D1-44E9-BE17-0007DE9FF933
D1230EA3-3358-4D69-9244-001D944F4565
B2BA3A0B-0FD6-4F3C-B48E-001FB17CF6F2
97DBBFA1-24CB-4F55-A5C0-00C6312FC71E
0CF3C24E-ECCC-45E0-9612-00E52ADCC145
.
.
.6399 riviä dataa
.
488916EC-EDC8-48B7-B818-FF0E433B3B83
CAF47144-61C5-4E1D-8EDB-FF145E7DE15C
3CFCD3AC-E354-40F8-AB78-FF192DC91C88
827C52B4-60FF-4E3F-A93D-FF2C3B3A7CEE
99E38153-BDAB-47C1-86A4-FF309030F609

--Toinen kysely

ID

4ABE2141-02F3-426D-AFB0-0137FD19C35E

50755B5F-8757-4B3F-87D5-079217E04340

F4811239-225B-4195-ABC8-0A2D9C953E4B
A940FF20-A9D8-4961-892C-0A9E411A428A
B21F872B-D256-4556-BF67-0B2C060D4EB0
7EC92EBA-DB73-4D00-B817-0C4E90E43440
.
.
.517527 riviä dataa
.
EEFDD76B-7F99-4CFB-BCCA-B7A1C9E9C69F
076B70E9-03C4-467F-A75B-B8B7760C29B8
479E9D28-88C8-4002-85B6-B9C169E6F8C4

2C2D48AF-1C61-4D6D-9A96-C233FF471922
DE03D33B-567D-4E3D-A7E6-C46D840A346C

--Kolmas kysely

DATAFROM	BATCH	UNITS	DATA	NAME	QTY
MEAS_NUMERICLIMIT	161101	milliampere	11.84258460998544	IBB	2
MEAS_NUMERICLIMIT	161101	milliampere	13.072443008423029	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	12.345886230468839	IBB	2
MEAS_NUMERICLIMIT	161101	milliampere	12.524652481078924	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	12.637567520141335	IBB	2
MEAS_NUMERICLIMIT	161101	milliampere	12.129211425781161	IBB	7
	.				
	.				
	.2391 riviä dataa				
	.				
MEAS_NUMERICLIMIT	161101	milliampere	12.365722656249911	IBB	5
MEAS_NUMERICLIMIT	161101	milliampere	12.167358398437411	IBB	4
MEAS_NUMERICLIMIT	161101	milliampere	12.120056152344016	IBB	3
MEAS_NUMERICLIMIT	161101	milliampere	12.426807880401292	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	11.999511718749911	IBB	1

Liite 6: IBB:n virrankulutuksen datat

DATAFROM	BATCH	UNIT	DATA	NAME	QTY
MEAS_NUMERICLIMIT	161101	milliampere	11,84258	IBB	2
MEAS_NUMERICLIMIT	161101	milliampere	13,07244	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	12,34589	IBB	2
MEAS_NUMERICLIMIT	161101	milliampere	12,52465	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	12,63757	IBB	2
MEAS_NUMERICLIMIT	161101	milliampere	12,12921	IBB	7
MEAS_NUMERICLIMIT	161101	milliampere	12,17194	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	11,90515	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	12,17217	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	12,23297	IBB	2
MEAS_NUMERICLIMIT	161101	milliampere	12,71691	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	12,60858	IBB	2
MEAS_NUMERICLIMIT	161101	milliampere	12,28356	IBB	11
MEAS_NUMERICLIMIT	161101	milliampere	12,30035	IBB	3
MEAS_NUMERICLIMIT	161101	milliampere	12,41174	IBB	8
MEAS_NUMERICLIMIT	161101	milliampere	11,65161	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	11,79376	IBB	2
MEAS_NUMERICLIMIT	161101	milliampere	12,24977	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	14,13296	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	12,34589	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	12,24213	IBB	2
MEAS_NUMERICLIMIT	161101	milliampere	12,57501	IBB	1
	.				
	.				
	. 2366 riviä dataa				
	.				
MEAS_NUMERICLIMIT	161101	milliampere	12,82827	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	14,50222	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	12,40387	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	12,22229	IBB	6
MEAS_NUMERICLIMIT	161101	milliampere	12,33521	IBB	4
MEAS_NUMERICLIMIT	161101	milliampere	12,13684	IBB	5
MEAS_NUMERICLIMIT	161101	milliampere	11,85608	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	12,2879	IBB	3
MEAS_NUMERICLIMIT	161101	milliampere	12,36572	IBB	5
MEAS_NUMERICLIMIT	161101	milliampere	12,16736	IBB	4
MEAS_NUMERICLIMIT	161101	milliampere	12,12006	IBB	3
MEAS_NUMERICLIMIT	161101	milliampere	12,42681	IBB	1
MEAS_NUMERICLIMIT	161101	milliampere	11,99951	IBB	1

Liite 7: IRF:n virrankulutuksen datat

DATAFROM	BATCH	UNITS	DATA	NAME	QTY
MEAS_NUMERICLIMIT	161101	milliampere	25,80594	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	24,43178	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	24,04076	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	25,9511	IRF	3
MEAS_NUMERICLIMIT	161101	milliampere	24,2348	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	26,44701	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	25,48712	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	25,16527	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	25,59862	IRF	3
MEAS_NUMERICLIMIT	161101	milliampere	25,78311	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	25,24614	IRF	3
MEAS_NUMERICLIMIT	161101	milliampere	25,85497	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	24,56256	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	26,25933	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	25,57259	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	24,26389	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	24,59304	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	24,64374	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	25,37577	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	24,6508	IRF	1
	.				
	.				
	. 4669 riviä dataa				
	.				
MEAS_NUMERICLIMIT	161101	milliampere	26,01657	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	24,80821	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	25,68849	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	25,82421	IRF	2
MEAS_NUMERICLIMIT	161101	milliampere	6,657544	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	25,50989	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	24,43758	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	24,72734	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	26,63164	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	24,66599	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	24,23869	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	25,81966	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	24,74145	IRF	1
MEAS_NUMERICLIMIT	161101	milliampere	25,45977	IRF	6
MEAS_NUMERICLIMIT	161101	milliampere	25,33922	IRF	1

Liite 8: IBB:n hylätyt virrankulutuksen datat

DATAFROM	BATCH	UNITS	DATA	NAME	QTY
MEAS_NUMERICLIMIT	161101	milliampere	57,53417969	IBB	1