Petteri Laine

ASIAN- JA ASIAKIRJANHALLINTA-JÄRJESTELMÄN
TIETOKANTARAKENTEEN SELVITYS JA DOKUMENTOINTI

Opinnäytetyö
Sähköinen asiointi ja arkistointi

Toukokuu 2014

MAMK
University of Applied Sciences

KUVAILULEHTI

MAMK University of Applied Sciences	**Opinnäytetyön päivämäärä** 17.5.2014
Tekijä(t) Petteri Laine	**Koulutusohjelma ja suuntautuminen** **Sähköinen asiointi ja arkistointi**

Nimeke
Asian- ja asiakirjanhallintajärjestelmän tietokantarakenteen selvitys ja dokumentointi

Tiivistelmä

Opinnäytetyö tehtiin yksityisellä puolella toimivalle yritykselle, jonka toiminta perustuu ohjelmakehitykseen. Yritys suunnittelee ja toteuttaa mm. asian- ja asianhallintajärjestelmiä pääosin julkishallintoon.

Opinnäytetyön aihe sai alkunsa tarpeesta selvittää nykyisen asian- ja asiakirjanhallintajärjestelmän tietokannan rakenne. Tietokannan rakennetta ei ollut dokumentoitu alun suunnittelupalaverien jälkeen, joten 10 vuotta vanhan järjestelmän kantarakenteen selvittäminen jälkikäteen oli tehtävä.

Työn pääpaino oli teknisen asiakastuen tueksi tehtävä ohjeistus. Erityisesti ongelmien selvittelyssä avuksi oleva dokumentaatio tehtiin mahdollisimman kattavaksi, jotta sitä voitaisiin käyttää hyödyksi.

Raportissa kerrotaan teoriapuolta relaatiotietokannoista ja niiden optimoinnista. Tämän jälkeen selvitetään kantarakenteen dokumentoimisessa käytettyä aikaa ja työn varsinaista etenemistä. Varsinainen kantarakenteen selvitysraportti ei työhön kuulu, sillä kyseessä on yritykselle liiketoiminnan kannalta tärkeä yrityssalaisuus.

Lopuksi pohditaan vielä nykyisen tietokantarakenteen toimivuutta ja sen mahdollisia ongelmia. Tietokantarakennetta analysoidaan myös tietokannan optimoinnin näkökulmasta.

Asiasanat (avainsanat)

SQL, SQL Server, Tietokantaohjelmat, Tiedonhallinta, Tietojärjestelmät, Tietokannat

Sivumäärä	**Kieli**	**URN**
45	Suomi	

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi Jukka Selin	**Opinnäytetyön toimeksiantaja**

DESCRIPTION

MAMK University of Applied Sciences	**Date of the master's thesis** 17 May 2014
Author(s) Petteri Laine	**Degree programme and option** eServices and Digital Archiving

Name of the master's thesis

Documentation of the database structure in a case and document management system

Abstract

This master's thesis was made for a company whose main line of business was programming and software development with a focus on developing and programming case management systems. The aim of this master's thesis was to document the database structure of the current case management system. The last time the company had documented the database structure was almost 10 years ago when the company started to develop the software. A lot had happened after that. It was time to survey the whole database structure and to provide written material for future use.

The main focus was to make a guide for the database structure for the helpdesk personnel's use. The documentation was made as comprehensively as possible so it could be used as a tool in problem-solving cases.

First I went through the theory and optimization of relational databases. Next, I introduced the progress of the documentation project and how I was able to identify the meaning of all tables and rows in the database. At last, I analysed the database structure of the program and thought its possible problems. Considering the optimization of the database structure was part of the thesis. The actual documentation was not part of the thesis, because of the business secrets and non-disclosure agreements.

Subject headings, (keywords)

SQL, SQL Server, Databases, Database management, Information systems, Database systems

Pages	**Language**	**URN**
45	Finnish	

Remarks, notes on appendices

Tutor	**Master's thesis assigned by**
Jukka Selin	

SISÄLTÖ

1 JOHDANTO 1

2 MIKSI TIETOKANTARAKENNE TÄYTYY DOKUMENTOIDA? 2

3 OPINNÄYTETYÖN ETENEMINEN 4

4 ASIAN- JA ASIAKIRJANHALLINTAJÄRJESTELMÄ OSANA ORGANISAATION KOKONAISARKKITEHTUURIA 5

4.1 Metatiedot 7

4.2 Tietokanta osana kokonaisarkkitehtuuria 8

4.3 Integroinnit 11

5 RELAATIOTIETOKANNOISTA YLEENSÄ 12

5.1 Relaatiokannan rakenne 13

5.2 Tietokannan käsittely ja hallinta 13

5.3 Tietokannan eheys 15

5.4 Relaatiotietokantojen optimointi ja indeksit 16

5.5 Tietokannan suorituskyvyn parantaminen 18

5.6 Indeksointi käyttöönoton jälkeen 19

5.7 Tietoriippumattomuus 20

5.8 Näkymät ja triggerit 21

5.9 Tietokannan normalisointi 21

6 TIETOKANTOJEN KÄYTTÖTAVAT 22

6.1 Räätälöidyt tietojärjestelmät 24

6.2 Tietoturva 25

6.3 Tietokannan ylläpito 27

7 NYKYINEN TIETOKANNAN TIETOKANTARAKENNE 29

7.1 Nykyisen tietokantarakenteen perustelut 30

7.2 Kantarakenteen hyvät ja huonot puolet 31

7.3 Mitä parannettavaa kantarakenteessa on? 33

7.4 Miten tietokannan taulut on nimetty ja mihin tällä on pyritty? 34

7.5 Mitä tietokantaan tallennetaan? 35

7.6 Miksi dokumentointi on jäänyt tekemättä? 36

7.7 Tietokannoissa olisi mahdollista käyttää jo luontivaiheessa valmiita liitäntöjä toisiin tauluihin 37

7.8 Tuetut tietokantatuotteet ja onko niitä mahdollista laajentaa? 38
7.9 Miten järjestelmän käyttäjätunnukset hoidetaan 39
7.10 Triggerit, proseduurit ja näkymät 39
8 NYKYISEN TIETOKANNAN OPTIMOINTI JATKOSSA 40
9 TYÖN LOPPUTULOS 42
LÄHTEET 45
LYHENTEET 46

1 JOHDANTO

Valitsimme yhdessä työnantajan kanssa opinnäytetyön aiheeksi yrityksen toimittaman ja alusta alkaen itse tekemän asian- ja asiakirjanhallintajärjestelmän tietokantarakenteen selvittämisen. Ohjelmaa on kehitetty vuodesta 2002 alkaen, mutta alun suunnittelupalavereiden dokumentaatioiden jälkeen ei tietokantarakennetta ole aikaisemmin dokumentoitu. Omat haasteensa ohjelman hallinnalle tuo tietokantarakenteen kasvu alun kymmenestä taulusta yli 100 taulun tietokannaksi.

Yrityksellä ja sen henkilökunnalla on pitkä kokemus erilaisten diaarijärjestelmien ja asianhallintajärjestelmien toteutuksesta erityisesti julkishallinnossa. Ensimmäinen julkishallintoon toteutettu diaarijärjestelmä on ollut käytössä jo vuonna 1982, eli 20 vuotta ennen nykyisen järjestelmän kehityksen aloittamista. Vuosien kokemus julkishallinnon toimintatavoista näkyy vahvana myös asian- ja asiakirjanhallintajärjestelmän toteutuksessa.

Opinnäytetyön tavoitteena on saada tuotettua riittävän laaja tietokantarakenteen dokumentaatio, jota voidaan käyttää asiakastuen ongelmanratkaisuissa hyödyksi. Nykyinen ongelmatilanteiden ratkominen vaatii käytännössä aina sovelluskehityksen apua, jolloin kantarakennetta selvitetään suoraan sovelluksen lähdekoodista aina yksittäisten tapauksien takia. Tämä työllistää turhaan sovelluskehityksen henkilöstöä, joiden turhaa häiritsemistä pyritään vähentämään tämän työn myötä.

Työ vastaa kysymyksiin, miksi tietokanta täytyy dokumentoida. Mitä hyötyä tietokannan dokumentoinnista on yritykselle? Entä mitä hyötyä tietokannan dokumentoinnista on ohjelmaa käyttävälle organisaatiolle? Voiko asiakasorganisaatio vaikuttaa tietokannan rakenteeseen ja toimintaan?

Tietokanta on sovelluksen tärkein osa, joka toimii ohjelman tietojen tallennuspaikkana. Käytännössä kaikki metatieto tallennetaan tietokantaan ja itse asiakirjat tiedostoina levylle.

Syy miksi tietokantarakennetta ei koskaan dokumentoitu riittävän hyvin, perusteltiin käytännössä dokumentaation vanhenemisella. On totta, että kantarakenteen selvitys jää vanhaksi hyvin pian sen valmistuttua, mutta perustoiminnallisuus pysyy kuitenkin

jatkossakin samana. Ongelmien selvittelyn avuksi dokumentaatiosta on varmasti hyötyä ja uskon siitä olevan apua myös itse sovelluskehitykselle. Dokumentaatiosta tehtiin riittävän laaja ja yksityiskohtainen, jotta myös sovelluskehitys pystyy siihen luottamaan. Pyrkimyksenä on myös pitää dokumentaatio ajan tasalla tulevien versioiden myötä. Kantarakenne muuttuu aina uusien versioiden myötä, mutta muutokset ovat huomattavasti pienempiä, kuin koko kantarakenteen dokumentointi yhdellä kertaa.

Työssä selvitettiin kantarakenne, jonka myötä myös mahdolliset käyttämättömät tietokantataulut ja sarakkeet saatiin selville. Vanhoja tauluja, joiden sisältö oli siirretty toiminnallisuuden osalta muihin tauluihin tai parametrointitiedostoihin löydettiin muutamia, kuten myös useissa tauluissa oli turhia sarakkeita, joiden toiminnallisuutta ei enää tarvittu. Samalla selvitettiin onko samaa tietoa tallennettu normalisoinnin vastaisesti useampaan eri tauluun tai sarakkeeseen. Näiden syitä pohditaan työssä tarkemmin ja selvitetään voisiko turhia tauluja tai sarakkeita lähteä poistamaan.

Varsinaista tietokantarakennetta en työhön voi liittää, vaan työ koostuu käytännössä kantarakenteen selvittämisen raportoimisesta, työn etenemisestä, ratkaisujen pohtimisesta ja mahdollisista muutosajatuksista.

2 MIKSI TIETOKANTARAKENNE TÄYTYY DOKUMENTOIDA?

Tietokantarakenteen riittävä dokumentointi asiakastuen tueksi on kriittistä sujuvan asiakastuen takaamiseksi. Ongelmatilanteiden pitkät selvitysajat näkyvät heti asiakkaille. Tietokantarakenteen dokumentoinnin tärkein painopiste opinnäytetyössä on asiakastuen työn nopeuttaminen. Käytännössä opinnäytetyönä tehty tietokantarakenteen dokumentointi pyrkii nopeuttamaan asiakkaiden ongelmienselvitysprosessia.

Pelkän kantarakenteen dokumentointi ei tietysti itsessään riitä, vaan myös sovelluskoodi täytyisi olla riittävän laajasti kirjattuna ylös, jotta myös henkilöiden vaihtumisen myötä muut pystyisivät jatkamaan kesken jäänyttä työtä.

Laajojen, useita tuhansia rivejä sovelluskoodia sisältävien ohjelmien osalta on myös enemmän sääntö kuin poikkeus, että tekijöitä on enemmän kuin yksi. Tällöin ohjelmakoodin ja kantarakenteen dokumentointi tulee entistä tärkeämmäksi.

Myös useiden liitännäisohjelmien kannalta kantarakenteen tarkka selvittäminen on tärkeää. Asianhallintajärjestelmässä on SÄHKE2-määritysten mukaan oltava avoin rajapinta, jolloin rajapintaa varten on tiedettävä tietokantarakenne. (JHS 176 2014.)

Kantarakenteen selvittämisen pääpaino oli teknisen asiakastuen puolella, mutta selvitys helpottaa myös tuotekehitystä erityisesti liitännäisohjelmien kehityksen puolella. Kaikkea toiminnallisuutta ei tällöin tarvitse kysyä asianhallintajärjestelmän tuotekehityksen puolelta, vaan nyt riittää pelkkä varmistus. Usein kehitystyö keskeytyi aina tiettyjen kenttien ja sarakkeiden selvittelyn vuoksi, mutta nyt lähes täydellisen dokumentaation perusteella pystytään työtä jatkamaan ilman keskeytyksiä. Tämä vaatii sen, että dokumentaatio on luotettava ja ajantasainen.

Selvityksen pohjalta dokumentaatio on ajantasainen, mutta vaatii tietysti jatkuvaa ylläpitoa uusien versioiden osalta. Tietokantarakenteen selvitys on myös riittävän luotettava asiakastuen käyttöön, mutta tuotekehityksen osalta täytyy vielä tehdä tarvittavia varmistuksia.

Kantarakenteen dokumentoinnin myötä pyritään nopeuttamaan asiakastuessa tehtävien ongelmanselvittelyitä. Vielä alkuvaiheessa dokumentaatioon ei voida luottaa aivan täysin, vaan dokumentaatio on enemmän varmistuksena ja antamassa vinkkejä. Jatkossa dokumentti tulee olemaan luotettava.

Työssä ei oteta suoranaisesti kantaa varsinaiseen sovelluksen lähdekoodin dokumentointiin, vaan työn pääpaino on tietokannan dokumentoinnissa. Kuitenkin myös ohjelmakoodi on dokumentoitava riittävän hyvin, jotta myös muut ohjelmakehittäjät pystyisivät sen jatkokehitystä jatkamaan. Jos ohjelmakoodia ei ole dokumentoitu riittävän hyvin, se nähdään myös tietoturvaongelmana. Mahdollisten pääkehittäjien onnettomuudet voisivat olla suurikin takaisku yritykselle, jollei ohjelmakoodia olisi dokumentoitu riittävän laajasti.

Sovelluskoodi on kuitenkin dokumentoitu kohtuullisen laajasti, sillä siihen on kiinnitetty huomiota myös tietoturvan puolesta. Huomiotta on vain jäänyt tietokannan rakenne, jonka dokumentointi on jäänyt liian karkealle tasolle, jolloin siitä on ollut apua vain sovelluskehittäjille.

Tietokannan dokumentointityö myös painottuu asiakastuen avuksi, jolloin ohjelman lähdekoodin dokumentoinnista ei tässä tapauksessa ole hyötyä. Sovelluksen lähdekoodia ei jaeta tietoturvan takia yrityksessä kuin ohjelmakehittäjille. Asiakastuki ei kuulu ohjelmakehitykseen, joten heitä varten tarvitaan oma ohjeistus. Asiakastuessa työskenteleviltä ei myöskään voida vaatia niin tarkkaa ohjelmointitaustaa, jotta he pystyisivät sovelluksen lähdekoodin kautta selvittämään sovelluksen toimintalogiikkaa. Tietokannan dokumentointi ei ole ollut sillä tasolla, mistä olisi ollut hyötyä asiakastuelle ja niille henkilöille, jotka eivät kantarakennetta ennestään tunne.

Yritys on saanut sertifikaatin ISO/IEC 27001:2005, jolla tarkoitetaan tietoturvaan liittyvää osaamista. Käytännössä tämän myötä on ymmärretty myös dokumentoinnin tärkeys. Sertifikaatin myötä sovelluksen lähdekoodin dokumentointi laitettiin ajan tasalle, joten se on nyt kohtuullisen hyvällä mallilla.

3 OPINNÄYTETYÖN ETENEMINEN

Kantarakenteen selvittämistä varten asensin oman testiympäristön viimeisimmällä sovellusversiolla. Työtä varten tarvittiin siis yksi palvelin, joka tässä tapauksessa oli kannettava tietokone. Tietokantana toimi Microsoft SQL server 2008 R2 express edition. Alkuvalmistelut veivät oman aikansa, mutta niiden jälkeen päästiin aloittamaan varsinainen kantarakenteen selvittely. Tietokannan asennuksessa oli hyvä kiinnittää huomiota myös asennuksen parametreihin, sillä myös ne vaikuttavat tietokannan sujuvaan toimintaan.

Olin onneksi hieman kartalla sovelluksen kantarakenteesta jo ennen työn tekemistä, joten perustaulut olivat jollain tasolla hallussa. Ymmärsin mihin tauluun tallentuvat asioiden tiedot, mihin asiakirjojen tiedot ja mistä löytyy käyttöoikeudet. Näistä tiedoista ei kuitenkaan laajassa selvityksessä ollut kovin suurta apua.

Työ lähti liikkeelle lisäämällä testiympäristöön testimateriaalia. Tämän jälkeen tutkittiin suoraan tietokannasta mihin tauluun ja mihin sarakkeeseen mikäkin rivi tallentui. Työ oli huomattavasti työläämpi kuin olin aikaisemmin ajatellut, mutta motivaatio kuitenkin riitti työn tekemiseen.

Aluksi lisäsin yhden asian järjestelmään, josta osasin jo arvata mihin tauluun asian päätiedot tulevat. Tiedot tallentuivatkin hyvin odottamaani tauluun, mutta en osannut arvata kuinka moneen muuhun tauluun syntyi viittauksia ja merkintöjä yhden asian lisäämisestä. Käytännössä jouduin yhden asian lisäämisen myötä käymään läpi kaikki kantarakenteen tietokantataulut, jotta pystyin olemaan varma siitä, mihin kaikkialle tietoa tallennettiin.

Kun asiapuolen taulut oli selvitetty, siirryin asiakirjoihin, toimenpiteisiin, toimeksiantoihin ja käyttöoikeuksiin. Nämä kaikki käytiin läpi samalla periaatteella, eli lisättiin yksi kohde sovellukseen ja tutkittiin kaikki tietokannan taulut läpi. Työ vei oman aikansa, mutta ainakin voitiin olla kohtuullisen varmoja tiedon eheydestä, joka on keskeisessä osassa teknisen dokumentaation tekemisessä. Tiedon on oltava varmasti oikein, tai koko dokumenttiin ei voida luottaa.

Kun laaja syötettävän tiedon osuus tietokantarakenteesta oli käyty läpi, siirryttiin selvittämään kiinteitä kenttiä, joihin syötetään metatietoa. Näihin kenttiin ei voida sovelluksessa lisätä mitään, joten selvittely täytyi tehdä hieman toisella tavalla.

Nyt lisättiin tietokantaan rivi tai paremminkin muokattiin nykyistä riviä ja katsottiin sovelluksesta mihin kohtaan tehty muutos syntyi. Tällä tavalla saatiin taas varmuus tiedon eheydestä, kun jokainen muutos saatiin tutkittua tietokannasta ja sovelluksesta.

Selvittelyissä käytettiin myös tietokantaliikennettä tallentavaa ohjelmaa, joka kirjoitti kaiken JDBC -protokollan kautta tehdyn tietoliikenteen tekstitiedostoon. Myös MS sql -serverin omat lokitustyökalut olivat hyödyksi haastavissa kantarakenteen selvittelyissä. Jäljitystyökaluilla ei kuitenkaan nähdä triggerien tekemiä muutoksia, joten aivan täydellistä selvitystä ei saatu aikaiseksi.

4 ASIAN- JA ASIAKIRJANHALLINTAJÄRJESTELMÄ OSANA ORGANISAATION KOKONAISARKKITEHTUURIA

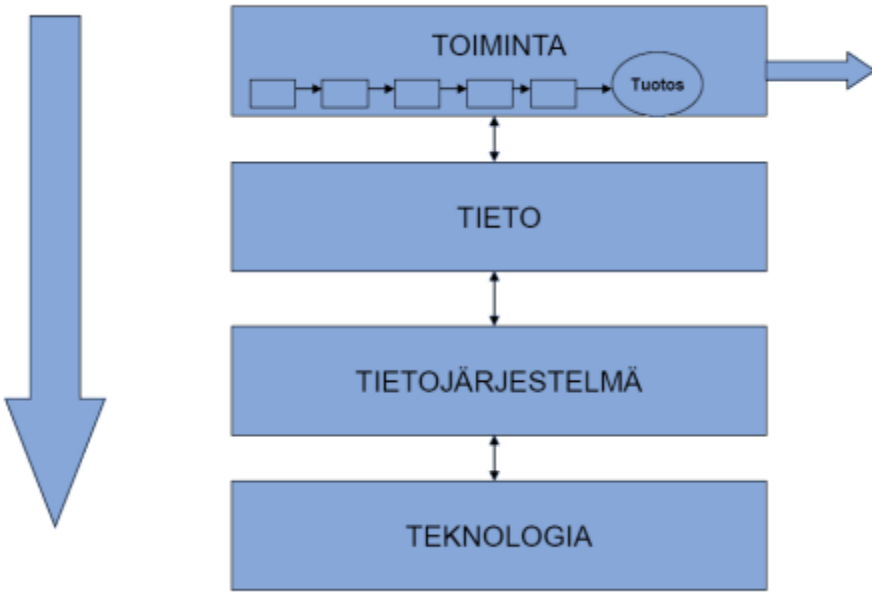
Asiakirjanhallinnasta on suomen standardoimisliitto tehnyt standardin SFS-ISO 15489-1, joka määrittelee asiakirjanhallinnaksi asiakirjojen elinkaaren hallinnan ja siihen kuuluvat prosessit riippumatta siitä, missä asiakirjan elinkaaren vaiheessa niitä suoritetaan ja kuka vastaa niiden suorittamisesta. Samassa standardissa määritellään

myös asiakirjajärjestelmäksi sopivan tietojärjestelmän. Järjestelmän täytyy pystyä ottamaan talteen ja käsittelemään asiakirjoja sekä mahdollistaa niihin pääsyn elinkaaren kaikissa vaiheissa.

Tietokanta on oleellinen osa asian- ja asiakirjanhallintajärjestelmää. Kaikki sovellukseen tallennetut tiedot tallentuvat tietokantaan. Tietokanta on sovelluksen ohella kahdessa alimmassa palkissa (kuva 1). Varsinainen organisaation toiminta tapahtuu ylimmässä palkissa.

Tietokanta toimii näkyvän sovelluskerroksen alla, eli käyttäjä ei missään vaiheessa tiedä minne tieto tallentuu. Käyttäjä käyttää häntä varten tehtyä graafista käyttöliittymää, jonne hän syöttää tietoja. Tiedot tallentuvat kuitenkin graafisen käyttöliittymän kautta tietokantaan.

Tietokannan sujuva toiminta on kuitenkin edellytyksenä ohjelman toiminnalle. Mahdolliset tietokannan ongelmat ja hitaudet näkyvät moninkertaisina ohjelman käytössä. Käyttäjän kannalta toimiva tietokanta parantaa ohjelman toimintaa, jolloin organisaation toimintaprosessi nopeutuu.



KUVA 1. Tietojärjestelmällä tarkoitetaan mm. Asian- ja asiakirjanhallintajärjestelmää (Nenonen, 2013)

Sovelluksella tehostetaan ensimmäisessä palkissa näkyvää (kuva 1) toimintaketjua. Esimerkkinä kunnan päätöksentekoprosessi, jonka tueksi asianhallintajärjestelmä on hankittu. Asianhallintajärjestelmällä mahdollistetaan SÄHKE-määritysten mukainen asioiden ja asiakirjojen hallinta koko niiden elinkaaren ajan.

4.1 Metatiedot

Metatiedot ovat oleellinen osa asian- ja asiakirjanhallintajärjestelmää. Dokumenttienhallinnan yksi tärkeimmistä ominaisuuksista on juuri tehokas metatietojen käyttö. Metatiedolla tarkoitetaan tietoa, joka kuvaa kohteen kontekstia, sisältöä ja rakennetta sekä hallintaa sen elinkaaren kaikissa vaiheissa. Käytännössä kyse on tavallaan pienestä hakemistosta, jonka perusteella on helppo hakea materiaalista haluttu kohta. (Seppä 2010; Arkistolaitos Sähke2 metatietomalli 2014.)

Metatietojen avulla pystytään tehokkaaseen tietojen hakuun eri järjestelmistä. Metatietoa on pyritty myös luokittelemaan eri ryhmiin ja tyyppeihin. Niistä yksi esimerkki on jakaminen viiteen osaan käyttötarkoituksen ja tiedon lähteen mukaan.

Metatieto jaetaan tässä mallissa sen mukaan liittyykö se tiedon hallintaan, onko kyseessä kuvailevaa metatietoa, liittyykö metatieto säilyttämiseen, onko metatieto tekniseen ympäristöön liittyvää vai tärkeimpänä käyttöä kuvailevaa ja käytöstä syntyvää metatietoa. (Seppä 2010; Salminen 2005.)

Metatieto on jaettavissa myös kahteen päätyyppiin. Näitä ovat vanhat staattiset metatiedot tai uudet dynaamiset metatiedot. Staattista metatietoa käytetään usein vanhojen paperimallisten arkistojen kanssa, joissa metatieto pysyy muuttumattomana koko asiakirjan elinkaaren ajan. Staattinen metatieto ei muutu käsittelyprosessissa ja usein staattinen metatieto luodaan asiakirjan luonnin yhteydessä. (Salminen 2005; Arkistolaitos Sähke2 metatietomalli 2014.)

Uudet asianhallintajärjestelmät ovat tuoneet mahdolliseksi dynaamiset metatiedot, joissa metatietoa voidaan helposti muuttaa koko asiakirjan elinkaaren ajan. Asianhallintajärjestelmä voi muodostaa dynaamista metatietoa asiakirjan käsittelyssä. Dynaamisella metatiedolla voidaan tarkoittaa esimerkiksi tiedolla, josta selviää asiakirjaa

viimeksi käsitellyt henkilö. (Seppä 2010; Salminen 2005; Arkistolaitos Sähke2 metatietomalli 2014.)

Metatiedot voidaan luokitella myös roolien perusteella. Tätä luokittelua on käytetty julkisen hallinnon tietohallinnon neuvottelukunnan suosituksessa JHS143. Rooliluokituksessa yksittäisen elementin tarkoitus määritellään sen kategorian kautta. Esimerkiksi asiakirjan elinkaarenhallintaan liittyvät elementit ovat aikamääre, tila, käsittelyhistoria ja säilytyshistoria. (JHS 143 2014)

Metatietoa tarvitaan asiakirjojen organisoimiseen, tuottamiseen, ylläpitämiseen, säilyttämiseen, jakeluun ja tuhoamiseen. Asianhallinnassa näiden lisäksi pyritään keräämään metatietoa koko asiankäsittelyprosessin ajan. Voidaan siis puhua dynaamisesta metatiedosta asiankäsittelyprosessissa. (JHS 143 2014; Arkistolaitos Sähke2 metatietomalli 2014.)

Metatietoa on pyritty standardisoimaan jo pitkään, mutta prosessi on yhä kesken. Erityisesti koneellisesti käsiteltävään metatietoon tarvitaan standardi, jolla voidaan rajoittaa jatkuvasti kasvavaa tietomäärää. (JHS 143 2014; Seppä 2010.)

Metatietojen standardoimattomuus on yksi suurimpia esteitä järjestelmien integroimiselle. Ohjelmat eivät voi käyttää toistensa tietoa hyväksi, jos metatiedot eivät ole vastaavat. Käytännössä esimerkiksi säilytysajat saattavat olla ristiriidassa keskenään.

4.2 Tietokanta osana kokonaisarkkitehtuuria

Tietokanta on vain yksi osa asianhallintajärjestelmää, joka on osa organisaation kokonaisarkkitehtuuria, mutta tietokannan vaikutus asianhallintajärjestelmään on kuitenkin suuri. Tietokanta on toimintaprosessin alimmassa kerroksessa, jolloin kaikki siihen tehdyt muutokset vaikuttavat suoraan muihin kerroksiin.

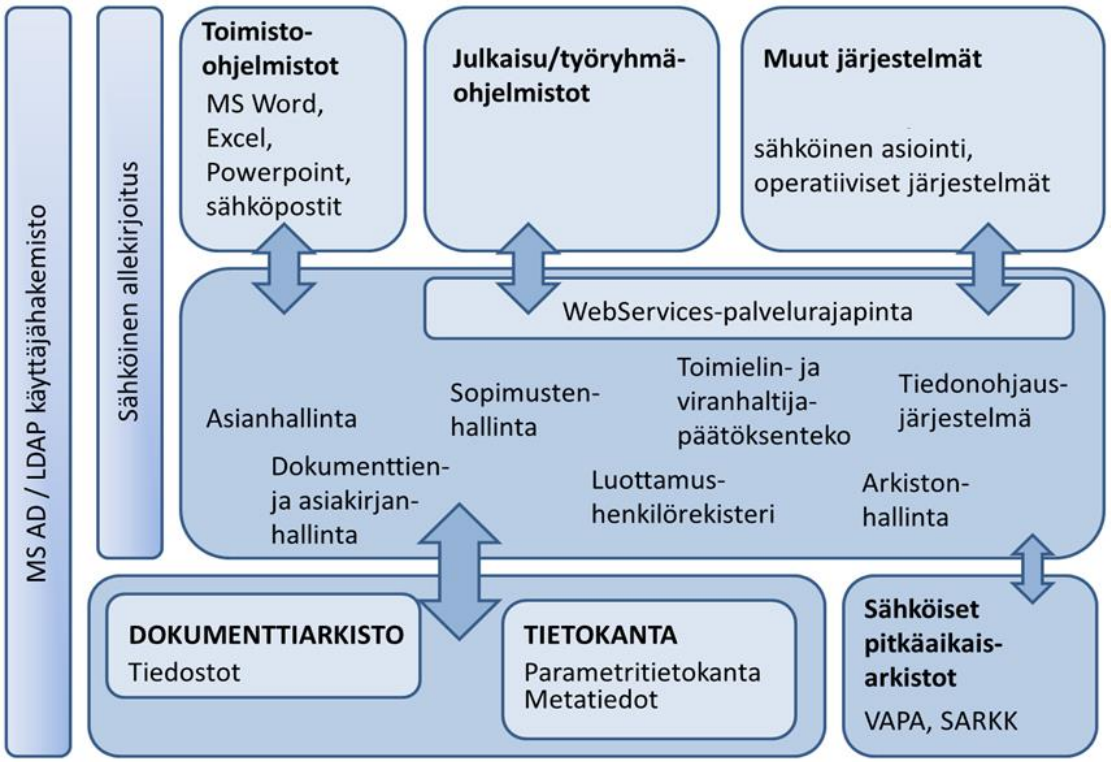
Varsinainen toimintaprosessi tapahtuu ylimmässä kerroksessa, jossa voidaan kuvata esimerkiksi kunnan päivittäinen toiminta. Tämä toiminta on yhteydessä tietoon, jota toimintaketjussa tarvitaan ja mitä sen tuloksena tuotetaan. Tietoa taas säilytetään ja käytetään tehokkaasti apuna usein tietojärjestelmän avulla. Tietojärjestelmällä kuvataan tässä tapauksessa asian- ja asiakirjanhallintajärjestelmää.

Asian- ja asiakirjanhallintajärjestelmällä on tärkeä rooli kunnan toimintaprosessissa, sillä se nopeuttaa ja helpottaa päivittäistä työtä. Järjestelmä ei ohjaa työtä, vaan se pyrkii olemaan apuna päivittäisessä käytössä. Asian- ja asiakirjanhallintajärjestelmä toimii kerroksena varsinaisen tietokannan päällä. Tietokanta on alimmassa kerroksessa, jonne tallentuu kaikki käytetty tieto. Tiedon saamiseksi käyttöön ja sen päivittäinen käyttö on toteutettu varsinaisen tietojärjestelmän avulla.

Jos tietokannan rakennetta muutetaan, vaikutukset näkyvät suoraan tietojärjestelmään, jonka kautta myös epäsuorasti vaikutukset näkyvät myös kunnan toimintaketjussa. Käytännössä muutoksia ei koskaan lähdetä tekemään tietokannasta alkaen, vaan muutostarve tulee kunnan toimintaketjun muutoksista. Tämän pohjalta lähdetään muokkaamaan tietojärjestelmää sellaiseksi, että se tukee muuttunutta toimintaketjua, joka taas aiheuttaa muutostarpeen tietokantaan.

Kun tietojärjestelmään tehdään muutoksia, täytyy tietokannan rakenne olla tiedossa. Tähän tietokannan dokumentointi tuo helpotusta, sillä jatkossa ei jokaista tietojärjestelmän muutosta varten tarvitse ensin selvittää sen osuuden tietokannan rakennetta ja vasta sitten lähteä tekemään muutoksia tietojärjestelmään. Nyt tietokannan rakenne on selvillä ja voidaan lähteä suoraan rakentamaan muutoksia tietojärjestelmään.

Dokumentoinnin jälkeen on myös huomattavasti helpompaa selvitellä mahdollisia tietojärjestelmän ongelmia, kun voidaan selvittää tietokannan osuus ongelmista ensin. Kun tietokannan rakenne tiedetään, voidaan sulkea sen osuus mahdollisesta ongelmasta pois ja nähdään heti, että ongelmaa on lähdettävä selvittämään sovelluksen puolelta.



KUVA 2: Kokonaiskuva järjestelmästä

Kuvasta 2 nähdään hyvin järjestelmän kokonaiskuva. Kuvassa on jaettu järjestelmä kolmeen osaan, joista tietokanta sijoittuu alimpaan laatikkoon. Tietokanta on pohjana koko järjestelmän toiminnalle.

Graafinen käyttöliittymä löytyy keskimmäisestä laatikosta, johon olen ryhmitellyt ohjelman eri toimintoja. WebServices-palvelurajapinta on tekninen toteutus erilaisille integraatioille muihin sovelluksiin. Sitä samaa rajapintaa käytämme myös itse omissa julkaisutuotteissamme. Ylimpään lokeroon on laitettu erilaiset työasemaohjelmistot, kuten tekstinkäsittely ja sähköpostit. Muita ylempiä lokeroita ovat aikaisemmin mainittu julkaisujärjestelmä, sähköinen asiointirajapinta ja muut asiakkaan omat tuotantojärjestelmät.

Kuvassa on mainittuna myös järjestelmään tunnistautumisen mahdollistavat lisätoiminnallisuudet, kuten ActiveDirectoryn tai LDAP-käyttäjähakemiston käyttö. Jos näitä lisätoimintoja halutaan käyttää, asianhallintajärjestelmän omaa käyttäjätunnistusta ei tarvita, vaan luotetaan AD/LDAPista tulevaan vahvaan tunnistukseen.

Sähköinen allekirjoitus on mainittuna, mutta sen tuomat haasteet ovat enemmänkin periaatetasolla. Teknisesti sähköinen allekirjoitus on toteutettu, mutta sen käyttöönotto

vaatii tutustumista finlexin lakiosuuteen, sillä eri organisaatiot pitävät sähköistä allekirjoitusta riittävänä allekirjoituksena, kun taas toiset vaativat edelleen käsin tehdyn allekirjoituksen.

4.3 Integroinnit

Nykytilanteessa monessa organisaatiossa on kymmeniä tietojärjestelmiä käytön tukena. Näiden ongelmana on se, etteivät järjestelmät keskustele keskenään. Ohjelmissa voi olla pahimmillaan samaa tietoa, jota kirjoitetaan käsin useampaan paikkaan.

Integrointien tarkoituksena on helpottaa tätä työtä. Jos kaikki ohjelmat keskustelisivat keskenään, voitaisiin esimerkiksi TOS eli tiedonohjaussuunnitelma määritellä yhteen sovellukseen, josta muut ohjelmat voisivat sitä lukea. Tällöin samaa tietoa ei tarvitsisi kirjoittaa käsin moneen eri järjestelmään, jolloin törmätään väistämättäkin kirjoitusvirheisiin. Kun tieto olisi vain yhdessä järjestelmässä, mutta kaikki muut ohjelmat voisivat käyttää samaa tietoa hyväkseen, päällekkäisyyksiltä säästyttäisiin.

Uudemmissa järjestelmissä on usein vaatimuksena avoimet rajapinnat, joilla mahdollistetaan toimivat integroinnit. Myös julkishallinnossa käytössä olevassa SÄHKE-määrityksessä on otettu tiukka kanta integrointeihin liittyen. SÄHKE:n vaatimusten mukaan järjestelmällä täytyy olla avoin rajapinta mahdollisia integrointeja varten. Tämä on saatettu aikaisemmin nähdä myös huonona puolena, sillä avoin rajapinta tarkoittaa aina ohjelman toimintalogiikan julkistamista tietyiltä osilta. Kuitenkin nykyään nähdään avoimet rajapinnat vain positiivisena asiana, joilla pystytään rakentamaan toimivia järjestelmiä. Vanhat järjestelmät eivät osaa käyttää integrointeja hyödykseen, eikä niihin välttämättä ole olemassa voimassa olevaa tukea. Usein myös useammassa eri järjestelmässä olevat tiedot ovat keskenään yhteensopimattomia ilman toimivaa avointa rajapintaa. (Arkistolaitos Sähke2 metatietomalli 2014)

Integraatiot ovat pääosin vasta kehitteillä, mutta jo toteutettuja integraatioita toisiin järjestelmiin on olemassa. Esimerkkinä näistä voidaan pitää eräänlaista paikkatietopalvelua. Asian- ja asiakirjanhallintajärjestelmään tallennetuissa karttakuvissa ja kauppasopimuksissa voidaan määritellä mihin sijaintiin ne liittyvät. Tällöin käyttäjä näkee suoraan järjestelmästä tarkan sijainnin, minne jokin lupahakemus liittyy. Sa-

moin käyttäjä voi hakea asian- ja asiakirjanhallintajärjestelmästä suoraan tiettyyn tonttiin liittyvät asiakirjat.

5 RELAATIOTIETOKANNOISTA YLEENSÄ

Relaatiotietokanta koostuu useista tauluista, joihin tieto tallennetaan. Tauluissa tiedot esitetään riveillä ja sarakkeilla. Relaatiotietokannan tarkoitus on pitää tieto tallennettuna vain yhteen paikkaan, jolloin tiedon eheys ei pääse kärsimään. Relaatiotietokantaan voidaan myös tallentaa tieto siitä, miten eri taulut liittyvät toisiinsa. (Hovi ym. 2005 6-8; Hernandez 2000 3-20; Hakkarainen 2012 57-59.)

Relaatiotietokannan etuna on tiedon helppo saatavuus. Kun tieto on tallennettu tietokantaan, se voidaan ottaa heti käyttöön ja juuri tallennettua tietoa voidaan lukea heti tallennuksen jälkeen. Nimi relaatiotietokanta syntyy siitä, että tallennettujen tietojen ja taulujen välille syntyy relaatio toisiinsa. Tämä relaatio nopeuttaa ja tarkentaa tietojen päivittämistä tietokantaan, sillä muutos tehdään vain yhteen paikkaan. (Hovi ym. 2005 68; Hernandez 2000 3-20; Hakkarainen 2012 57-59.)

Relaatiotietokantaa luodessa on hyvä varata aikaa suunnitteluun. Hyvin suunniteltu relaatiotietokanta on helppo ylläpitää, kun taas huonosti suunniteltu relaatiotietokanta aiheuttaa pahimmassa tapauksessa koko tietokannan uudelleenrakentamisen. Hyvin suunniteltua relaatiotietokantaa voidaan laajentaa ja päivittää eteenpäin.

Käyttäjätaulu

Käyttäjän ID	Sukunimi	Etunimi
1	Nieminen	Arja
2	Virtanen	Irja
3	Jokinen	Terhi

Oikeustaulu

Omistaja	Asiakirja	Oikeustaso
1	100	2
2	101	4
3	102	4

KUVA 3: Esimerkki relaatiotietokannan tauluista

Yhteen paikkaan tallennettu tieto päivittyy muihin tauluihin relaation kautta (kuva 3). Kun käyttäjän sukunimi muuttuu, riittää kun sukunimi vaihdetaan käyttäjätauluun. Muut taulut lukevat käyttäjän tiedot aina käyttäjätaulusta, eikä esimerkiksi oikeustauluun tarvitse muuttaa mitään. Käyttäjän ID -numero pysyy oikeustaulussa samana ja

tämän ID -numeron perusteella haetaan sukunimi käyttäjätaulusta. (Hovi ym. 2005 6-8; Hernandez 2000 3-20; Hakkarainen 2012 57-59.)

5.1 Relaatiokannan rakenne

Relaatiotietokanta perustuu tietokantatauluihin, joista koko rakenne koostuu. Taulut nimetään kuvaavasti ja ne usein ovatkin loogisesti yhteenkuuluvia asiakokonaisuuksia. Taulut sisältävät sarakkeita ja rivejä, joiden perusteella tieto syötetään tauluun. Sarakkeilla on aina saman taulun sisällä toisistaan poikkeavat nimet. Jokaiselle sarakkeelle annetaan aina tietty pituus ja tietotyyppi, kuten merkkijono tai päivämäärä. Tällä perusteella kaikki samaan sarakkeeseen lisätyt tiedot täyttävät tietotyypin vaatimukset. Liian pitkiä kenttiä ei kannata tehdä, vaan pituus olisi hyvä rajata tarvittavan pituiseksi. Liian pitkät kentät vievät turhaan tietokannan suoritustehoa ja kasvattavat sen kokoa.

Jokaiseen tauluun on aina luotava relaatiokannan sääntöjen mukaan primary key, eli perusavain. Primary key on aina uniikki yksilöivä tieto, jonka perusteella taulun sisältöä voidaan hakea. Primary key voidaan myös määritellä viittaamaan useampaan sarakkeeseen. (Hakkarainen 2012 65; Hernandez 2000 44, 211-235.)

Tauluihin voidaan määritellä myös foreign key, eli viiteavain. Foreign key viittaa aina toiseen tauluun, jota tässä tapauksessa kutsutaan isätauluksi. Tällöin puhutaan viiteeheydestä. Isätaululla voi olla monta lapsitaulua, mutta lapsitaululla voi olla vain yksi isätaulu. Viittaukset foreign keyllä tapahtuvat aina päätaulun primary keyhin. (Hakkarainen 2012 65; Hernandez 2000 44, 211-235.)

Relaatiomallin sääntöjen mukaan tauluissa ei voi olla päällekkäisiä rivejä, mutta tämä on myös estetty primary keyn avulla. Lisäksi voidaan luoda uniikkeja indeksejä, jotka estävät haluttujen tietojen päällekkäisen tallentamisen, vaikka primary key -arvo olisikin eri. (Hovi ym. 2005 8-11; Hakkarainen 2012 65; Hernandez 2000 44, 211-235)

5.2 Tietokannan käsittely ja hallinta

Yksi relaatiokannan tärkeimpiä ominaisuuksia on siihen tehtävien hakujen toiminta. Tietokannan täytyy olla rakennettu niin, että siihen voidaan tehdä hakuja käyttäen

SQL -lauseita. Huonosti rakennettuun relaatiokantaan ei saada toimivia hakulauseita aikaiseksi, vaikka ne olisi kuinka hienosti rakennettu. (Hovi ym. 2005 9-14; Hakkarainen 2012 88-100, 124-127.)

Relaatiokantaan tehdään kaikki toiminnot käyttäen SQL -kieltä. SQL -kieli käyttää pääosin kansainvälisiä standardeja, joten siinä ei ole suuria eroja eri tietokantatuotteiden välillä. Tämä tuo periaatteessa mahdollisuuden vaihtaa tietokantatuotetta sovelluksen alta, vaikka käytännössä tämä ei ihan suoraan onnistukaan. SQL:n tärkeimpiä komentoja ovat SELECT, INSERT, UPDATE, DELETE, DROP ja CREATE -lauseet. (Hovi ym. 2005 9-14; Hakkarainen 2012 88-100, 124-127.)

Tietokannan hallinnan ja ylläpidon osalta tärkeimpiä ovat SELECT ja UPDATE -lauseet, joilla voidaan hakea tietoa relaatiokannasta. UPDATE -lauseella voidaan päivittää tai muuttaa olemassa olevaa tietoa taulussa. Esimerkkinä yksinkertainen SELECT -lause: SELECT * from kayttaja where sukunimi='Nieminen';. Tällä hakulausekkeella haetaan tietokannasta kaikki kayttaja -taulun sisältämät henkilöt, joiden sukunimi on Nieminen. (Hovi ym. 2005 9-14; Hakkarainen 2012 88-100, 124-127.)

Tietokannan käsittelyssä käytettyä SQL -kieltä upotetaan suoraan ohjelmakoodiin, jolloin ohjelma pystyy itse syöttämään tietoa tietokantaan. Kun graafisen käyttöliittymän kautta lisätään esimerkiksi uusi käyttäjätunnus ohjelmaan, ohjelman sisällä tehdään INSERT -tyylinen lause. Tämä INSERT -lause lisää tietokantaan käyttäjän ohjelman kautta syöttämät tiedot. Jos käyttäjä muokkaa ohjelmassa nykyisen käyttäjän tietoja esimerkiksi vaihtamalla käyttäjän sukunimen toiseksi, tehdään ohjelmakoodissa pyyntö tietokannalle käyttäen UPDATE -tyylistä sql -lausetta. (Hovi ym. 2005 9-14; Hakkarainen 2012 88-100, 124-127.)

Tietokantaan suoraan tehtävien SQL -lauseiden kanssa on aina hyvä olla varovainen, sillä osaamattomissa käsissä monet SQL -lauseet saattavat saada paljon tuhoa aikaan. UPDATE -lauseesta where -ehdon unohtaminen muuttaa kaikki kyseiseen tauluun liittyvät rivit samoiksi, eikä SQL -kielessä ole mahdollisuutta palata takaisin edelliseen tilanteeseen kuin palauttamalla koko tietokanta varmuuskopiosta.

SQL -kielessä on mahdollisuus käyttää transactionia ennen SQL -lauseiden ajamista. Tällöin saadaan UNDO -toiminto joissakin tilanteissa käyttöön. Transactionin käyttö

lukitsee joissakin tilanteissa tietokannan rakennetta, joten kesken tuotantokäytön sen käyttö voi aiheuttaa lukkotilanteita. Transactionia ei myöskään voi käyttää kaikkien taulurakennetta muokkaavien SQL -lauseiden kanssa. (Hovi ym. 2005 9-14; Hakkarainen 2012 88-100, 124-127.)

5.3 Tietokannan eheys

Relaatiomallin mukaan tietokannan on oltava eheä. Tietokannan eheys on vaarantunut, jos samaan tauluun tallennetaan samaa tietoa kahteen kertaan. Sama asiakas ei voi olla tietokannassa tallennettuna kahteen kertaan ja molempiin merkittynä esimerkiksi osoitetieto erilaisiksi. Tällöin ei voida olla varmoja siitä, kumpi tieto on oikea. Tätä pyritään rajoittamaan primary key -tiedoilla, kuten myös uniikeilla indekseillä. (Hovi ym. 2005 11-15; Hernandez 2000 53-54, 289-298.)

Relaatiotietokannan määrityksissä rajataan myös, että primary key -kenttä ei saa olla tyhjä ja sen sisältämän tiedon on oltava uniikkia. (Hovi ym. 2005 11-15)

Isätaulun ja lapsitaulun eheyttä valvotaan primary key – foreign key -yhdistelmällä. Isätaulusta ei voida poistaa riviä, jos lapsitaulussa on viittaus siihen. Ilman foreign key -käytäntöä poisto olisi teknisesti mahdollista, mutta tällöin tietokannan viite-eheys olisi särkynyt. Lapsitauluun olisi jäänyt orpoja rivejä, joihin ei löydy riviä isätaulusta. (Hovi ym. 2005 11-15; Hernandez 2000 53-54, 289-298.)

Relaatiotietokannat ovat levinneet niiden helppokäyttöisyyden ja joustavuuden takia jo laajalle. Käytännössä kaikki uudet tietokannat tehdään käyttäen relaatiotietokantaa. Relaatiotietokannan rakentaminen vaatii kuitenkin kunnollisen suunnittelun, jotta tietokannan eheys pystytään varmistamaan myös jatkossa. Suunnittelussa on myös aina otettava huomioon tietokannan mahdollinen päivittäminen ja jatkokehittäminen. (Hovi ym. 2005 11-15; Hernandez 2000 53-54, 289-298.)

Tietokannan eheys on myös haastavaa pitää kunnossa suuremmissa tietokannoissa. Jo pienissäkin tietokannoissa syntyy nopeasti tilanteita, joissa relaatioiden määrä kasvaa suureksi. Jotta relaatioiden hallinta pysyy järkevällä tasolla, on tietokantaa hallittava jatkuvasti. (Hovi ym. 2005 11-15; Hernandez 2000 53-54, 289-298.)

5.4 Relaatiotietokantojen optimointi ja indeksit

Jotta tietokannan tiedot olisi helposti saatavilla, on tauluihin tehtävä indeksejä. Käytännössä indekseillä nopeutetaan hakuja relaatiotietokannan tauluista. Indeksi on kuin erikseen tehty sisällysluettelo tai aakkosellinen hakemisto. Hakemisto on järjestyksessä, joten sieltä voidaan hakea tietoa nopeammin, kuin käymällä aina koko materiaali läpi. (Hovi ym. 2005 140-160; Hakkarainen 2012 309-320.)

Tärkeintä relaatiotietokannan optimoinnissa onkin oikeiden avainten tai indeksien suunnittelu ja tekeminen. Jos ohjelman kautta haetaan usein asioihin liittyviä tietoja, täytyy asioiden tauluun tehdä indeksi. Jos asioista haetaan aina julkisuustietoa, on järkevää luoda indeksi perustumaan julkisuustietoon. (Hovi ym. 2005 140-160; Microsoft Coursebook 2011 5-25; Hakkarainen 2012 309-320.)

Samaan tauluun voidaan luoda useita indeksejä, jolloin tietokanta arvioi itse nopeimman tavan löytää tietoa. Tästä tavasta puhuttaessa käytetään termiä optimointiohjelma. Nämä ovat käytännössä tietokannan sisäisiä komentoja, joilla voidaan päivittää tietokannan statistiikat vastaamaan nopeinta toimintatapaa. Usein optimointiohjelmiin ei tarvitse kiinnittää huomiota, vaan normaalit tietokannan ylläpitorutiinit hoitavat ne automaattisesti. (Hovi ym. 2005 140-160; Microsoft Coursebook 2011 5-25; Hakkarainen 2012 309-320.)

Indeksit kuluttavat hieman levytilaa, mutta nykytilanteessa ollaan valmiita käyttämään levytilaa tietokannan vastausajan vähentämiseksi. Nykyisin palvelimiin annetaan niin paljon resursseja, ettei levytilan käyttö indeksien takia tule vastaan. Indeksien luonti kestää jonkun aikaa, mutta nykykoneilla puhutaan kuitenkin vain joistakin minuuteista. Indeksin luomisen jälkeen tietokantaan tehtävät haut nopeutuvat välittömästi. Indeksit voidaan organisoida uudelleen, mutta nykykoneiden osalta käytetään melkeinpä ennemmin koko indeksin uudelleen luomista. Nämä rutiinit kuuluvat tietokannan hallintaan ja ylläpitotyöhön, jotka ajastetaan tapahtuvaksi öiseen aikaan. (Hovi ym. 2005 140-160; Microsoft Coursebook 2011 5-25; Hakkarainen 2012 309-320.)

Jos taulun indeksointi ei tunnu riittävän, eli haut kyseiseen tauluun vievät edelleen liikaa aikaa, on yksi keino poiketa normalisoinnin säännöistä ja luoda aputaulu hakuja varten. Tämän aputaulun luominen tarkoittaa saman tiedon kopioimista toiseen tau-

luun, jota ei normalisoinnin ja relaatiokantojen määritysten mukaan suositella tehtäväksi. Kuitenkin laajaan materiaaliin tehtävä haku nopeutuu aputaulun avulla huomattavasti.

Indeksiä on mahdollista käyttää myös taulun rivien järjestyksen muuttamiseen. Tällöin puhutaan clustered indeksin tekemisestä. Clustered indeksit muokkaavat taulun rivejä indeksin mukaiseksi, jolloin haku nopeutuu hieman. (Hovi ym. 2005 140-160; Oracle DBA Handbook 2008 5-23; Hakkarainen 2012 309-320.)

Indeksien luominen voidaan tehdä heti taulun luomisen yhteydessä, tai koska vain jälkeenpäin. Indeksi on siis luotavissa jo valmiiksi täyteen tietokantatauluun. Tietokannan ei tarvitse olla tyhjä, jotta siihen voidaan luoda indeksi. (Oracle DBA Handbook 2008 5-12; Hakkarainen 2012 309-320.)

Usein tietokantaan tehdään indeksejä sitä mukaa, kun havaitaan tiettyjä hakuja olevan paljon ja niiden käyttämä aika on liian pitkä. Tuotannossa olevissa tietokannoissa tämä tulee vastaan usein asiakaspalautteena, jolloin ensimmäinen korjaustoimenpide on usein indeksien luominen jälkikäteen.

Esimerkki normaalista indeksistä:
Create index sukunimi on kayttaja(sukunimi);

Esimerkki uniikista indeksistä, jolloin estetään samojen tietojen kirjoitus tauluun:
Create unique index sotu on kayttaja(sotu);

Esimerkki clusteroidusta indeksistä:
Create clustered index tunnus on kayttaja(tunnus);

Uudemmat tietokantatuotteet myös perustavat aina indeksin taulua luodessa. Tällöin indeksi luodaan primary keyhin kohdennettuna.

Indeksien muuttaminen jälkikäteen on hyvä tehdä aina käyttökatkon aikaan. Suuriin tauluihin indeksien luominen kestää muutamia minuutteja, jolloin kyseinen tietokantataulu on lukossa muulta käytöltä. (Hovi ym. 2005 160-180; Hakkarainen 2012 309-320.)

Nopeasti muuttuviin sarakkeisiin tehtävien indeksien osalta on hyvä olla varovainen. Liian nopeasti muuttuviin sarakkeisiin tehtävät indeksit kuormittavat palvelimen fyysistä levyä ja aiheuttavat sitä kautta hidastumista. Tämä on kuitenkin Hovin, Huotarin ja Lähdenmäen kirjassa kumottu esimerkkien avulla. Nopeasti muuttuviin tauluihin tehtävät indeksit nopeuttavat käyttöä ja varsinainen fyysisen levyn rajoite tulee vastaan vasta, kun tauluun tehdään muutoksia 10 kertaa sekunnissa. Näin nopeasti muuttuvia tauluja ei juurikaan käytetä. (Hovi ym. 2005 160-180; Hakkarainen 2012 309-320.)

Indeksien määrällä ei ole varsinaista rajaa, vaikka joissain ohjeissa ilmoitetaankin käyttämään korkeintaan kuutta indeksiä tietokantatauluun. Tällä ei kuitenkaan ole varsinaista merkitystä ja samaan tauluun voidaan hyvin luoda useampiakin indeksejä. Indeksien määrä riippuu pitkälti kohdennetusta tietokantataulusta, sillä jotkin taulut eivät kestä edes kahta indeksiä samaan tauluun. Indeksien käyttö on erittäin suositeltavaa, mutta niiden käyttö on hyvä testata valmiissa tuotteessa riittävän laajasti. (Hovi ym. 2005 166)

5.5 Tietokannan suorituskyvyn parantaminen

Tietokannan suorituskykyä voidaan parantaa monella tavalla, joista nopein ja kätevin tapa on indeksoinnin rakentaminen. Usein indeksien parantaminen riittää, eikä muihin tapoihin tarvitse enää paneutua.

Tietokannan suorituskyvyn parantaminen näkyy suoraan varsinaisen ohjelman toimintanopeudessa. Tietokanta on toimintakerroksen alimmalla tasolla, jolloin kaikki sen hitaudet näkyvät moninkertaisena ylemmillä tasoilla, joista tärkein on tietysti itse sovelluskerros. Sovelluksen hitaudet taas haittaavat ohjelman käyttöä.

Seuraavana tapana ovat normaalit tietokannan hallintarutiinit, eli indeksien uudelleenjärjestely tai kokonaan uudelleen luominen. (Oracle DBA Handbook 2008 5-23; Hakkarainen 2012 309-320.)

Sovelluksen rakenne voi aiheuttaa tietokannan suorituskyvyssä suuriakin hitauksia. Sovelluksen tekemät turhat tietokantahaut hidastavat tietokantaa, jolloin sovelluslogiikka on hyvä käydä läpi. (Hovi ym. 2005 144-145; Hakkarainen 2012 309-320.)

Joissain SQL -kutsuissa tietyt toimintatavat ovat hitaampia kuin toiset. Näissä on tuotekohtaisia eroja, joten niiden selvittäminen on aina hieman hankalaa. Myös SQL -kutsuilla saatetaan kysellä liikaa tietoa, mitä ei kuitenkaan tarvita. Tällöin turhat SQL -rajoitteet olisi hyvä karsia pois. (Hovi ym. 2005 144-145; Fernandez, Beginning Oracle 2009 381-411; Hakkarainen 2012 309-320.)

Denormalisoinnilla voidaan vähentää liitoksia, joka nopeuttaa hakuja. Tämä tosin toistaa tietoa, joka aiheuttaa tietokannan ylläpidolle ja päivittämiselle haasteita. Usein tarkoituksellisella denormalisoinnilla tehdyt redundaaviset tiedot hoidetaan triggereiden avulla. Tällöin triggerit pitävät huolen siitä, että tieto kopioidaan myös toiseen tauluun. (Hovi ym. 2005 144-145; Fernandez, Beginning Oracle 2009 381-411; Hakkarainen 2012 309-320.)

Vasta tässä vaiheessa astuu kuvaan tietokantapalvelimen fyysiset resurssit. Nykykoneissa resurssit ovat usein riittävät suuret, joten niihin ei tarvitse juurikaan paneutua. Kuitenkin ne on syytä huomioida, jos mikään muu tietokannan suorituskyvyn parantamisista ei ole auttanut. Tähän luetaan mukaan myös SQL -prosessin priorisoiminen, jolla SQL -tuotteelle annetaan enemmän käyttötehoa palvelimelta. (Hovi ym. 2005 144-145; Fernandez, Beginning Oracle 2009 381-411; Hakkarainen 2012 309-320.)

5.6 Indeksointi käyttöönoton jälkeen

Usein indeksointia ei pystytä tekemään tietokannan luontivaiheessa riittävän tarkasti. Indeksointi astuu näkyvästi voimaan vasta riittävän laajoilla materiaaleilla ja testiympäristöissä eivät välttämättä materiaalit ole samaa luokkaa kuin tuotantokäytössä olevissa järjestelmissä. Indeksejä on kuitenkin helppo luoda operatiiviseen järjestelmään myös jälkikäteen, eikä niiden luominen vaikuta ohjelman toimintaan.

Usein ohjelmat ovat käyttöönoton jäljiltä liian hitaita, jolloin järkevin tapa korjata ongelma on luoda siihen oikein suunniteltuja indeksejä. Käyttäjillä voi myös olla toisenlaisia toimintatapoja tehdä hakuja tiettyihin tauluihin, kuin testiryhmällä. Tällöin in-

deksit täytyy luoda uudelleen niin, että niissä on mukana yleisimmin haetut hakuehdot. Vanhoja indeksejä ei välttämättä kannata poistaa, vaan luoda uusi indeksi uusilla hakurajauksilla. (Hovi ym. 2005 230-250; Hakkarainen 2012 353-367.)

Indeksoinnin parantamisessa voidaan käyttää erilaisia tietokantatyökaluja, joilla voidaan tutkia käytettyjä SQL lauseita ja niiden käyttämiä saantipolkuja. Saantipoluista nähdään mihin sarakkeisiin ja tauluihin haut kohdistuvat ja tätä kautta voidaan päätellä ovatko kyseiset sarakkeet indeksien piirissä. Ne voidaan joko lisätä olemassa olevaan indeksiin, tai luoda kokonaan uusi. Nykyaikana tietokannan hakulausekkeiden nopeuttaminen indeksien avulla on järkevin tapa, eikä oikein palvelinten resursseihin liittyviä ongelmia juurikaan ole. (Hovi ym. 2005 230-250; Hakkarainen 2012 353-367.)

Myös asianhallintajärjestelmän indeksejä on jouduttu parantamaan käyttöönoton jälkeen, kun on huomattu tiettyjen julkaisujärjestelmän hakujen tekevän liian raskaita toimenpiteitä. Korjaukset on tehty niin indekseihin kuin itse julkaisujärjestelmän hakuihinkin. (Hovi ym. 2005 230-250; Hakkarainen 2012 353-367.)

5.7 Tietoriippumattomuus

Relaatiotietokannan yksi suurimmista eduista muihin tietokantoihin on tiedon parantunut tietoriippumattomuus. Koska tietokanta on tietoriippumaton, voidaan sinne perustaa uusia tauluja tai sarakkeita ilman, että nykyinen toiminta häiriintyy. Tähän liittyy myös uusien indeksien luominen valmiiseen ohjelmaan ilman, että normaali käyttö häiriintyy.

SQL -lauseilla käsitellään vain niitä rivejä tauluista, joita sillä hetkellä tarvitaan. Tauluun tehtävä haku rajataan niin, että sillä hetkellä turhat rivit jätetään pois hakutuloksesta. Haku kohdistuu aina tauluihin, jolloin käyttäjä ei varsinaisesti tiedä mistä fyysisestä paikasta levyltä kyseinen tieto löytyy.

Osa tietoriippumattomuutta on myös se, ettei tietokannan taulussa olevien rivien järjestyksellä ole merkitystä tietokannan toimintaan. Rivit saadaan haluttuun järjestykseen käyttämällä ORDER BY -hakuehtoa. ORDER BY:n käyttö kuitenkin hidastaa hakulauseita, joten järkevämpää on luoda taulu oikeassa järjestyksessä, tai käyttää

clusteroitua indeksiä. (Hovi ym. 2005 11-13; Hakkarainen 2012 65, 124-127, 320-324.)

5.8 Näkymät ja triggerit

Näkymillä tarkoitetaan dynaamisia tauluja. Nämä näkymät eivät kuitenkaan ole varsinaisia tauluja, mutta niihin voidaan tehdä hakuja kuin normaaleihin tauluihin. Kyseessä on siis tavallaan virtuaalinen taulu, jonka tiedot on kasattu muista tietokannan tauluista. Näkymiä käytetään monien tietokantaoperaatioiden apuna ja luomaan käyttäjille tarvittaessa yksilöllisiä näkymiä tietoihin. (Hovi ym. 2005 14-15; Hakkarainen 2012 305; Hernandez 2000 359-383.)

Triggerit toteuttavat tietyn toiminnon aina kun triggeriin määritetty laukaisutoiminto tapahtuu. Triggerit seuraavat tiettyjä SQL -lauseita ja aina kun ehto täyttyy, tehdään tietty toimenpide. Esimerkkinä voidaan pitää käyttäjätauluun liittyvää triggeriä, joka seuraa kaikkia DELETE, INSERT, UPDATE -lauseita. (Hovi ym. 2005 14-15; Hakkarainen 2012 305; Hernandez 2000 359-383.)

5.9 Tietokannan normalisointi

Normalisointi on oleellinen osa relaatiotietokantojen toimintaa. Normalisoinnilla tarkoitetaan suositeltua tapaa, jolla tietokanta on tarkoitus toteuttaa. Normalisointi on menetelmä tietokannan järkevään suunnitteluun. Normalisoinnin seuraaminen minimoi tietojen toistamisen tietokannassa.

Normaalimuotoja on olemassa useita, joista kolme ensimmäistä ovat alkuperäisen tekijän E.F.Coddin suunnittelemia. Sama tekijä on määritellyt relaatiotietokantateorian. Ensimmäiset kolme normaalimuotoa ovat myös tärkeimpiä. Muut normaalimuodot ovat Boyce-Codd -normaalimuoto, neljäs normaalimuoto ja viides normaalimuoto. Normaalimuodot ovat asteittain tiukkenevia ehtoja, joista seuraavaan normaalimuotoon päästäkseen on täytettävä myös edellisten normaalimuotojen vaatimukset. (Myntinen 2009; Hernandez 2000 150-177)

Normalisointia käytetään, jotta tietorakenteesta saadaan toimivampi. Normalisoinnilla pyritään minimoimaan tietojen toistaminen, helpottamaan tietokantojen jatkokehitystä

ja päivittämistä, pitämään tietokanta yhdenmukaisena ja muutosjoustavana. (Mynttinen 2009; Hernandez 2000 150-177.)

Normalisoinnilla tavoitellaan lyhyesti sitä, että relaatiotietokannasta saadaan relaatiomallin mukainen. Normalisoinnilla pyritään minimoimaan käsitteellistä ja tietojen syöttämisestä johtuvaa redundanssia eli tietojen päällekkäisyyttä. Relaatioiden riveistä pyritään myös tekemään lisäysten ja muokkausten osalta itsenäisiä kokonaisuuksia. Käytännössä normalisointi auttaa myös tietokannan joustavuuteen, jolloin tulevat muutokset on mahdollista tehdä muokkaamatta tietokannan tauluja erikseen. (Mynttinen 2009; Hernandez 2000 150-177.)

Ensimmäisen normaalimuodon mukaan jokaisen taulun sarakkeen arvot ovat atomisia eli taulussa ei ole toistuvia ryhmiä tai moniarvoisia sarakkeita.

Toisessa normaalimuodossa määrätään, jos taulussa on moniosainen primary key, kaikkien sarakkeiden tulee olla funktionaalisesti riippuvia koko primary keystä, eikä vain osasta tätä avainta. (Mynttinen 2009; Hernandez 2000 150-177.)

Kolmannen normaalimuodon mukaan jokaisen sarakkeen on oltava funktionaalisesti riippuvainen vain primary keystä.

Usein tietokannan taulujen suunnitteluvaiheessa syntyy tilanteita, joissa osa tauluista ei täytä normalisoinnin sääntöjä. Tällöin on helpointa jakaa tietoa useampaan tauluun, jolloin normaalimuotojen säännöt täyttyvät. Tietokannan ollessa kolmannessa normaalimuodossa voidaan todeta kantarakenteen olevan normalisoitu. (Mynttinen 2009; Hernandez 2000 409-414.)

6 TIETOKANTOJEN KÄYTTÖTAVAT

Tietokantoja käytetään tiedon hallittuun säilyttämiseen. Nykyaikaisissa ohjelmissa tietojen tallennus tehdään tietokantaan, eikä vanhoja levylle tiedostona tallennettavia ratkaisuja enää ole käytössä. Tietokantojen parhaita ominaisuuksia on niistä tehtävien hakujen helppous. Tietokantaa voi käyttää useampi henkilö tai ohjelma samaan aikaan ja tietokanta on heti ajan tasalla myös muille käyttäjille. Tiedot on tallennettu tietokantaan vain yhteen kertaan, joten ristiriitaisia tuloksia ei synny.

Tietokannat mahdollistavat monien nykyaikaisten ohjelmien käytön. Tieto tallentuu reaaliajassa esimerkiksi pankkiliikenteessä, jolloin tililtä tehty nosto Rovaniemellä näkyy reaaliajassa myös Helsingistä tehdyssä tilitietojen kyselyssä.

Asian- ja asiakirjanhallintajärjestelmässä tietokantaan tallentuu kaikki käyttäjän kirjoittama metatieto. Asiakirjat tallentuvat levylle, eli niitä ei lähdetä lisäämään tietokantaan binääridatana, vaan ne pidetään tiedostoina palvelimen levyllä. Kaikki muu tieto käytännössä löytyy tietokannasta. Esimerkkinä voidaan pitää asiakirjan tallennuksessa lisättäviä metatietoja kuten nimi, tyyppi, säilytysaika, julkisuustieto ja muut organisaation toivomat tekstikentät. Usein jatkokäytön kannalta suositaan alasvetovalikkoja, jolloin ei pääse tapahtumaan inhimillisiä kirjoitusvirheitä.

Tietokantoja käytetään lähes kaikessa tietoteknisessä toiminnassa. Jopa verkkosivut pohjautuvat usein tietokantaan, joista yleisin käytössä oleva tuote on Mysql. Tietokantaa käytetään tässä tapauksessa ohjaustietokantana, jonne voidaan tallentaa tietoja.

Operatiivisessa tietokannassa tapahtuu pääosin kolmenlaisia toimintoja. On tapahtumankäsittelyä, kyselykäyttöä ja eräajoja. Tapahtumankäsittelyllä tarkoitetaan pieneen osaan tietokantaa kohdistuvasta toimenpiteestä, jolla tehdään esimerkiksi muutamaan tauluun liittyvää tietojen muokkausta. Tällöin monen taulun tietokannasta tapahtumankäsittely kohdistetaan vain pieneen osaan koko tietokannasta.

Kyselykäyttö tarkoittaa pääosin SELECT -lauseilla tapahtuvaa tietojen kyselyä. Tietokantana voi olla esimerkiksi väestörekisterikeskuksen henkilötietokanta, josta tehdään kyselyitä henkilön tietojen mukaan. Kaikki kyselyt tapahtuvat graafisen käyttöliittymän pinnan alla SELECT -lauseilla, joilla saadaan tulostettua ruudulle tarvittavat tiedot.

Viimeisenä eräajolla tarkoitetaan erilaisia automaattisia tapahtumia, joihin käyttäjä ei suoranaisesti osallistu ollenkaan. Käyttäjä voi laittaa eräajon käyntiin esimerkiksi ottamalla raportin tietokannassa olevista asioista, joita saattaa olla useita satoja. Tällöin eräajossa tulostetaan ruudulle tai suoraan paperille rajatut asiat tietokannasta, eli käydään läpi suuri määrä tietoa.

Usein tietokannoilla tarkoitetaan sähköisiä tietokantoja, mutta on hyvä huomioida, että myös paperisia tietokantoja on olemassa. Nämä ovat tietysti paljon harvinaisempia, mutta sanalla tietokanta voidaan tarkoittaa esimerkiksi ruutupaperille kirjoitettua tietoa sisältävää taulukkoa. Tietokanta on kokoelma tietoja, joilla on yhteys toisiinsa.

KUVA 4: Graafisen käyttöliittymän kautta ollaan yhteydessä sovellustasoon, joka toimii tietokantaa vasten.

6.1 Räätälöidyt tietojärjestelmät

Sovelluksia löytyy niin räätälöityjä kuin valmispakettejakin. Räätälöityjen ohjelmien etuna on yksittäiseen käyttöön tarkasti suunniteltu ja rajattu ohjelma. Tällöin ei jouduta tekemään kompromisseja erilaisten toimintojen osalta, vaan voidaan toteuttaa ohjelma juuri sellaisena kuin asiakas sen haluaa. Räätälöidyn ohjelman haittapuolena on sen suuri valmistus- ja ylläpitokustannus. Ohjelma on toteutettu ainoastaan yhdelle asiakkaalle, joten kyseinen asiakas maksaa kaupalliseen tapaan kaikki sen ohjelman valmistukseen ja ylläpitoon liittyvät kulut. Myös kaikki päivittämiseen ja uusien ominaisuuksien toimittamiseen liittyvät kulut jäävät asiakkaan maksettavaksi.

Toinen tapa on käyttää niin sanottua valmispakettia, jolloin ohjelma toteutetaan tukemaan useaa toimintatapaa. Tällöin samaa ohjelmaa voidaan myydä usealle asiakkaalle ja toimintatapojen muutos tehdään asiakaskohtaisesti parametreja muuttaen. Etuna tässä on kulurakenteen hillitseminen yksittäiseltä asiakkaalta ja koko järjestelmän tuotekehityksen kulut jakautuvat tasaisesti koko asiakaskunnan kesken.

Yrityksen suunnittelemaa asian- ja asiakirjanhallintajärjestelmää pidetään valmispakettina, jonka toimintaa voidaan ohjata asiakaskohtaisesti erilaisilla parametreilla. Käytännössä tämä on yrityksen kannalta ainoa järkevä tapa toimia, sillä tällöin voidaan pitää ohjelman hinta riittävän edullisena julkishallinnon käyttöön. Räätälöidyt

ohjelmat olisivat suurimmalle osalle liian arvokkaita. Tämän haittapuolena on kantarakenteen laajuus ja monimutkaisuus, joka tuli myös tietokannan dokumentointityön myötä ilmi. Kannasta löytyy edelleen paljon vanhoja tauluja ja sarakkeita, jotka on joskus toteutettu tiettyä asiakasta varten ja eivät ole enää käytössä yhdelläkään asiakkaalla.

Valmisohjelmien haittapuolena on rajoitettu räätälöitävyys asiakaskohtaisesti. Usein valmisohjelmien on tuettava montaa toimintatapaa, joka tuo haasteita tietokannan toiminnalle. Tietokannan rakenteen on oltava yleiskäyttöinen, jolloin kantarakenteen saaminen kuvaavaksi on haastavaa. Tarkoitus on pitää samaa tietokantaa kaikilla asiakkailla niin, ettei kantarakennetta tarvitse muuttaa erikseen yhtäkään asiakasta varten. Tämä tekee tietokantojen rakenteista usein monimutkaisia ja ei-kuvaavia. Tällöin tietokannan rakenteen tutkiminen, lukeminen ja ymmärtäminen ovat aina haastavaa.

6.2 Tietoturva

Tietokantoja käytettäessä on hyvä huomioida myös tietoturva. Vielä joitakin vuosia sitten ohjelmistojen tietoturvaan ei juurikaan kiinnitetty huomiota, vaan pääpaino oli ohjelman toiminnallisessa kehityksessä. Julkisen internetin tultua markkinoille ja ohjelmien rajapintojen avaaminen yleiseen käyttöön on tuonut myös huonot puolet esille. Toisaalta ohjelmien rajapintojen avaaminen on pistänyt ohjelmistotalot miettimään myös ohjelmien tietoturvaa.

Yleisimpiä tietoturva -aukkoja tietokantaa käyttävissä sovelluksissa ovat eräänlaiset SQL injectionin sallimiset. Ohjelman tekstikenttään syötetty tieto voidaan kirjoittaa suoraan SQL -lauseena ja poistaa turha tieto kommenttimerkeillä. Tällöin ohjelman syöttämä SQL -lause muuttuu ja käyttäjä pääsee syöttämään omia lauseita tietokantaan. (Turunen 2013; Hernandez 2000 551-642.)

Esimerkkinä voidaan ottaaa SELECT -lause.
Select id from kayttajat where tunnus='admin' and salasana='salasana';

Kyseisessä SQL -lauseessa tarkistetaan tietokannasta käyttäjän ID -numero etsimällä siihen sopiva käyttäjätunnus ja salasana. Jos käyttäjä syöttäisi käyttäjätunnuksen kyse-

lykenttään tiedon ”admin' --” ja jättäisi salasana -kentän tyhjäksi, ohjelma näkisi syötteen seuraavana:

Select id from kayttajat where tunnus='admin' - -' and salasana='';

Tästä lauseesta tietokanta lukisi vain select id from kayttajat where tunnus='admin' ja loput rivistä näkyisi kommenttina, jota ei huomioida. Tällöin joudutaan tilanteeseen, missä käyttäjä pääsisi kirjautumaan järjestelmään pääkäyttäjän oikeuksin tietämättä pääkäyttäjän salasanaa. (Turunen 2013; Hernandez 2000 551-642.)

SQL injectionit on kuitenkin huomioitu nykyaikana aika tehokkaasti ja niihin on tehty valmiita kirjastoja. Valmiissa kirjastoissa määritellään esimerkiksi yleisimpien katkaisumerkkien muuttaminen toisiksi merkeiksi, jolloin tieto ei mene SQL -lauseena tietokantaan asti. Esimerkiksi käyttäjätunnus -kenttään ”admin' - -” syöttäminen muutetaan kirjaston avulla ”admin' //” -näköiseksi, jolloin SQL lauseesta tulee toimimaton, eikä käyttäjä pääse kirjautumaan järjestelmään sisään. (Turunen 2013; Hernandez 2000 551-642.)

SQL injectionit eivät ole ainoita huomioitavia tietoturva -aukkoja, mutta ovat yksi yleisimmistä. Muita yleisiä tietoturvan kannalta huomioitavia asioita ovat XSS -syötteet, käyttäjän itse muokkaamat syötteet esimerkiksi selaimen URL -riviltä, salasanojen murtaminen, selkokielisten salasanojen käyttö ja sosiaalinen hakkerointi, jossa käyttäjä huijataan kertomaan tunnus ja salasana. (Turunen 2013; Hernandez 2000 551-642.)

Nykyisin tietoturvaan kiinnitetään paljon huomiota ja ohjelmien tietoturvaa myös tutkitaan. Monet uudet tietojärjestelmien hankinnat tehdään pitämällä tietoturva mielessä. Usein toimittajalta jo vaaditaan sertifikaatti, joka on kolmannen osapuolen tekemä arvio yrityksen tietoturvasta. Tällä pystytään aukottomasti todistamaan, että tietoturvaa on mietitty ja yrityksen tuotteet ovat riittävän tietoturvallisia.

Ilman sertifikaattia olevat yritykset joutuvat usein muilla keinoin selvittämään tuotteiden ja yrityksen tietoturvan tason. Usein tässä tilanteessa päädytään hankkimaan sertifikaatti kolmannelta osapuolelta. Suomessa on muutamia tietoturvan sertifiointeja tekeviä yrityksiä, joista suurin on Inspecta Oy. Inspectan tekemä ISO 27001 -sertifiointi

on kansainvälinen tietoturvaan liittyvä sertifikaatti, joka velvoittaa yrityksen ainakin miettimään tietoturvaan liittyviä asioita. Sertifikaatin saanut yritys on Inspectan mukaan panostanut riskienhallintaan ja on luotettava yhteistyökumppani. Organisaatio myös hallinnoi tietojaan pitääkseen ne virheettöminä, helposti käytettävissä ja hyvin suojattuina. (Inspecta 2014)

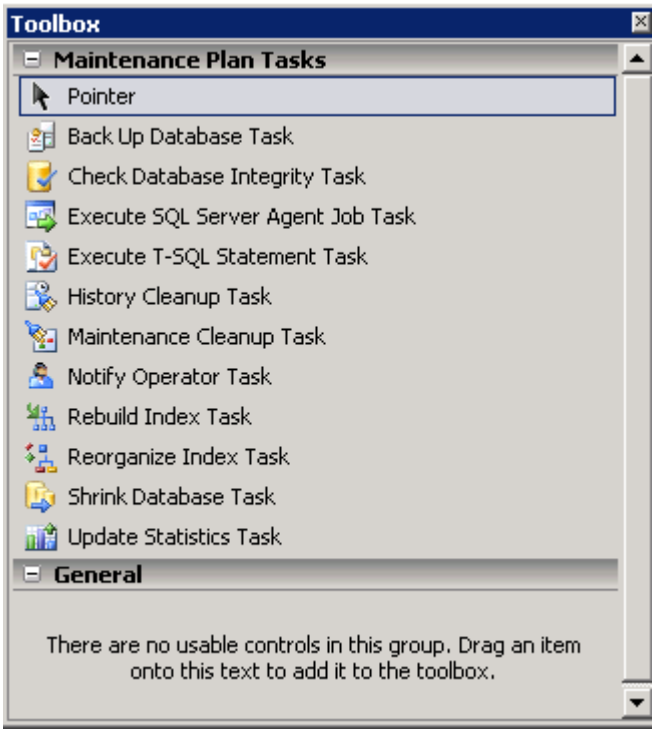
Myös julkishallinto on ottanut uusien tietojärjestelmien hankintaan vaatimuksiksi eräänlaisen tietoturvatasoihin perustuvan vaatimusmenettelyn. Nopein tapa määritellä tuotteen tietoturvataso julkishallintoa varten on virallisen sertifikaatin esittäminen. Tällä päästään tietylle julkishallinnon tietoturvatasolle.

6.3 Tietokannan ylläpito

Tietokannan luominen ei vielä riitä pysyvään käyttöön. Tietokannan optimoinnin ja tietoturvan kannalta on järkevää toteuttaa myös hallittu tietokannan ylläpito. Tietokantaan tallennetaan jatkuvasti uutta tietoa ja sitä luetaan sitäkin tehokkaammin. Tietokannassa oleva tieto säilyy tallessa, mutta sen optimoinnin kannalta täytyy indeksejä myös hoitaa. (Fernandez, Beginning Oracle 2009 381-411; Hernandez 2000 353-372.)

Indeksejä käytetään tiedon hakuun tietokannasta ja tietokannan omat optimointityökalut hallinnoivat tiedon hakua. Optimointityökalut priorisoivat reittejä, mitä kautta tietoa tietokannasta etsitään. Jotta optimointityökalujen tekemät priorisoinnit saadaan tallennettua indekseihin, täytyy indeksit organisoida uudelleen. Tähän hallintatyöhon on tietokantatuotteissa omat hallintasovellukset, jolla voidaan suorittaa tarvittavia ylläpitokomentoja. Indeksien osalta voidaan käyttää joko reorganize index -komentoja, tai luoda koko indeksi rebuild -komennolla uudestaan. (Fernandez, Beginning Oracle 2009 381-411; Hernandez 2000 353-372.)

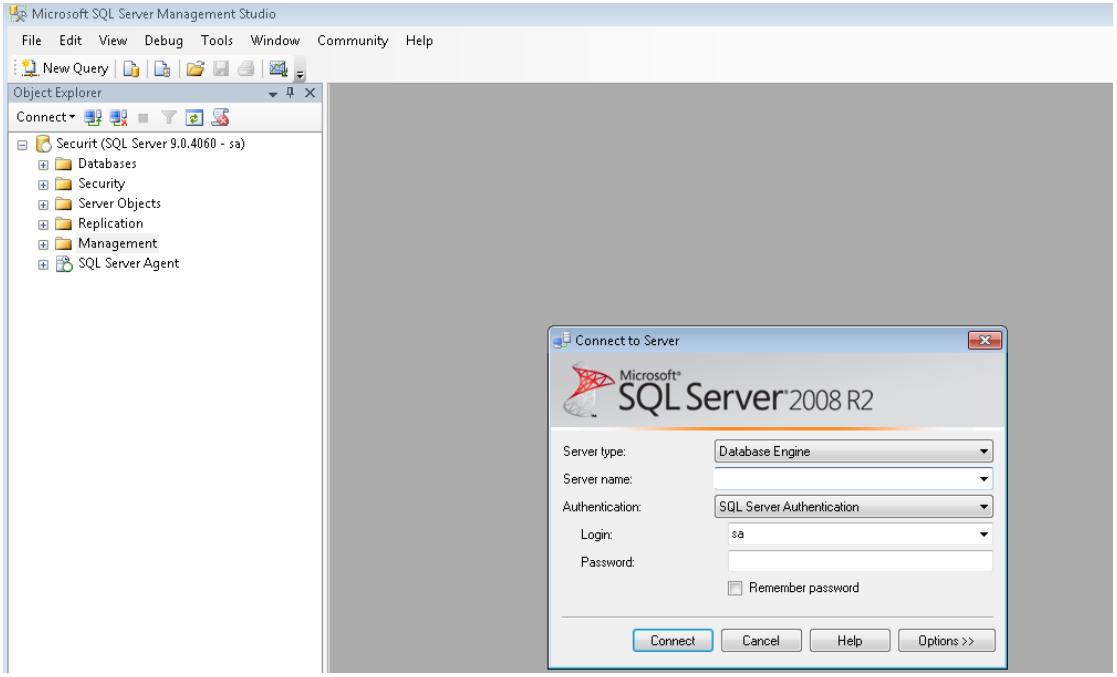
Komennot ovat hieman erilaisia riippuen tietokantatuotteesta, mutta niiden tekemät toiminnot ovat vastaavia kuin muissakin järjestelmissä. Usein yksinkertaiset ja kevyet indeksien uudelleenrakennukset toteutetaan kerran vuorokaudessa yöllisissä hallintarutiineissa. Näitä kevyempiä hallintatöitä on esimerkiksi indeksien uudelleen organisointi tai rakennus, tietokantojen muuttuneiden tiedostojen varmistus ja sen, että tieto on ehjää eikä rikkonaisia relaatioita ole. (Fernandez, Beginning Oracle 2009 381-411; Hernandez 2000 353-372.)



KUVA 5: Tietokannan hallintatyökalulla tehtävät ylläpitorutiinit (MS Sql server R2)

Raskaimmissa, usein kerran viikkoon tehtävissä hallintarutiineissa tehdään koko tietokannan varmistus, luodaan indeksit kokonaan uudestaan ja tyhjennetään vanhoja lokitietoja. Näiden suurempien töiden ohella siirretään levylle syntyneet varmistustiedostot myös nauhalle tai muulle varmistuslaitteelle. (Fernandez, Beginning Oracle 2009 381-411; Hernandez 2000 353-372.)

Tietokannan päivittäinen ylläpito on tärkeää, eikä sitä voi jättää tekemättä. Ilman varmistuksia ei pystytä palaamaan takaisin, jos palvelimen levy hajoaa. Jos varmistukset ovat kunnossa, voidaan luottaa järjestelmään tehdyn tiedon säilymiseen. (Fernandez, Beginning Oracle 2009 381-411; Hernandez 2000 353-372.)



KUVA 6: Tietokannan hallintatyökalu Microsoft SQL Server 2008R2

7 NYKYINEN TIETOKANNAN TIETOKANTARAKENNE

Nykyinen asianhallintajärjestelmän kantarakenne on alkujaan suunniteltu ja toteutettu vuosina 2002–2005. Tällöin ajateltiin vielä vahvasti ohjelmien toimivan niin, kuin ohjelmistotalot ne sattuvat toteuttamaan. Kuitenkin yrityksen silloin jo lähes 20 vuotuisen historian myötä oli opittu ajattelemaan hieman toisella tavalla, ja tuotteen kehittäminen lähti asiakkaan tarpeista. Kuitenkaan JHS -suositusten mukaista kokonaisarkkitehtuurin ajattelutapaa ei vielä ollut sisäistetty.

Asiakkaan yhteyshenkilöiden kanssa määriteltiin miten ohjelman pitäisi toimia ja miten se voitiin teknisesti toteuttaa. Tekniset toteutustavat olivat hieman erilaisia vielä vuonna 2002, mutta eivät kuitenkaan rajanneet toiminnallisuutta pois. Kuitenkin kompromisseja jouduttiin alkuvaiheessa tekemään.

Alkuperäinen suunnitelma ohjelman toiminnasta ja sitä myöten tehty tekninen suunnitelma, joka sisälsi myös kantarakenteen, on jäänyt vanhaksi jo vuosia sitten. Alkuperäinen suunnitelma kantarakenteesta sisälsi vain kymmenkunta tietokantataulua, joten nykyinen yli 100 -taulun järjestelmä on huomattavasti laajempi.

Usein tietokantarakenne suunnitellaan asiakkaan tarpeiden mukaan. Tietokantarakenne suunnitellaan käytännössä niin, että otetaan huomioon muuttuvat tilanteet ja tulevat

päivitykset. Yksikään sovellus ei ole valmis ensimmäiseen tuotantoversioon siirryttäessä, vaan päivityksiä tulee ja tietokannan on pysyttävä muutoksissa mukana. Tietokannan tuleva rakenne suunnitellaan selvittämällä organisaation toimintatavat, eli mitä tietoja asiakas tallentaa järjestelmään? Miten näitä tietoja käytetään? Ja miten tietoja käsitellään ja ylläpidetään? Kuitenkin kymmenessä vuodessa myös asiakkaan tarpeet ovat muuttuneet ja sovelluksen on täytynyt muuttua mukana. Asiakkaan tapa käsitellä tietoja on muuttunut, asiakas käyttää tietoja monella eri tavalla ja tietoja syötetään monesta eri lähteestä. (Hovi, Huotari, Lahdenmaki 2005 10-15; Hernandez 2000 21-31.)

Nykyinen kantarakenne on kasvanut kymmenessä vuodessa nykyiseen kokoonsa ja kasvu jatkuu edelleen. Kantarakenteeseen tulee muutoksia ja lisäyksiä sitä mukaa, kun uutta toiminnallisuutta kehitetään. Käytännössä vanhoja poistuneita tauluja ei ole koskaan lähdetty poistamaan, vaan ne ovat jääneet roikkumaan turhina tauluina järjestelmään. Jälkeenpäin sovelluskehityksen resurssit eivät enää ole riittäneet vanhojen turhien taulujen siivoamiseen, joten niiden poistaminen on jäänyt odottamaan hiljaisempaa hetkeä. Nyt työn myötä käytiin läpi myös vanhat käytöstä poistuneet taulut ja sarakkeet, joten siivoustyötä voidaan jatkaa.

Nykyinen kantarakenne sisältää yli 100 -tietokantataulua, useita triggereitä, näkymiä ja proceduureja. Näistä suurin osa on käytössä, mutta osa jäänyt myös vuosien kehityksen myötä turhiksi. Nykytilanteen selvityksen vuoksi tehtiin tietokantarakenteen dokumentointi, joka antoi tarkkaa tietoa mm. turhista tauluista, jolloin jatkokehityksenä voidaan nykyistä kantarakennetta lähteä siivoamaan.

Vastoin tietokannan normalisoinnin sääntöjä tallennetaan samaa tietoa myös useampaan tauluun vanhojen triggerien osalta, mutta osa näistä duplikaateista on perusteltuja.

7.1 Nykyisen tietokantarakenteen perustelut

Jos koko tietokantarakenne tehtäisiin nyt uudestaan, moni asia olisi varmasti tehty toisella tavalla. Alkuperäiset suunnitelmat ovat jääneet vanhaksi jo vuosia sitten. Ohjelmaa kehitetään jatkuvasti eteenpäin ja uusien toiminnallisuuksien myötä on kantarakennetta uudistettava.

Kuitenkin samalla on pidettävä mielessä vanhojen versioiden päivitysmahdollisuus. Asiakkailla tuotantokäytössä oleva järjestelmä on voitava päivittää uudempaan versioon ilman tietojen katoamista. Päivitys vanhemmasta versiosta uudempaan ei myöskään saa olla liian riskialtis, jolloin esimerkiksi taulujen sarakejärjestyksien muutokset jäävät tekemättä. Vaikka tietokannan sarakejärjestyksen muutos ei vaikuta ohjelman toimintaan, on selkeyden vuoksi hyvä pitää kantarakenne mahdollisimman samanlaisena vanhojen tietojen osalta. Tämä helpottaa ongelman selvittelyä, kun rakenne on pääosin sama myös vanhemmissa versioissa. Kantarakenteen on oltava helposti ylläpidettävä.

Kantarakenteen suunnittelussa ei pystytty ottamaan huomioon kaikkia niitä toimintatapoja, joita nykyinen versio tukee. Monet tietokantaan tehtävät haut vaativat erilaisten näkemysten luontia ja tietyissä tapauksissa myös aputaulujen käyttöä. Vaikka tietokantahaut pystytäänkin toteuttamaan normalisoinnin mukaisesti ilman aputauluja, suureen massaan kohdistuvat sisältöhaut hidastaisivat järjestelmän käyttöä tuntuvasti.

Taulujen määrä tietokannassa on kasvanut alun jälkeen hyvinkin paljon, mutta syynä tähän on uusien toiminnallisuuksien vaatimat tallennuspaikat. Tietokannan normalisoinnin mukaisesti on järkevää yrittää pitää tietokantarakennetta selkeänä, jolloin pyritään tallentamaan yhteen asiaan liittyvät tiedot yhteen tietokannan tauluun.

7.2 Kantarakenteen hyvät ja huonot puolet

Kantarakenteen hyviin puoliin voidaan lukea sen pääosin normalisoinnin mukainen toiminta. Tietokantarakenne on kohtuullisen uusi, vaikka onkin lähes 10 vuotta vanha. Samat sovelluskehittäjät ovat tehneet yrityksen muita ohjelmia jo vuosia ennen asianhallintajärjestelmän tuotekehityksen aloittamista, jolloin muiden järjestelmien virheistä pystyttiin ottamaan oppia uuden järjestelmän puolella.

Tarkoituksena on ollut pysyä tietokannan normalisoinnin säännöissä mukana, mutta pakottavista syistä tästä on jouduttu joissain tilanteissa poikkeamaan. Myös triggereiden käyttöä on pyritty rajoittamaan niiden hankalan hallittavuuden takia. Käytännössä järjestelmässä pidetään triggereitä vain käyttäjienhallinnan osalta. Triggereiden toi-

minta kun ei näy ohjelmalogissa ja ne voivat aiheuttaa yllättäviä muutoksia, jos tietokantaa operoidaan käsin.

Tietokanta on siis pyritty pitämään mahdollisimman yksinkertaisena, jossa jokaiselle toiminnolle on aina tehty oma taulu. Tämä selkeyttää toimintaa ja mahdollisten ongelmien etsintä helpottuu.

Kuitenkin taulurakenne on alkanut paisumaan ja tauluja on päässyt syntymään yli 100. Tämä ei sinänsä ole ongelma, mutta ilman ajantasaista dokumentaatiota ongelmien selvittäminen yli 100 -taulun järjestelmästä on haastavaa.

Huonona puolena voidaan myös pitää turhien taulujen ja sarakkeiden roikottamista tietokannassa, vaikka näistä ei varsinaisesti mitään haittaa olekaan.

Joissain kohdissa on jouduttu poikkeamaan normalisoinnin säännöistä, joilla on tarkoitus estää tiedon redundanssia, eli tietojen tallentamista useampaan kertaan. Näissä tapauksissa kuitenkin syy on ollut perusteltua.

Esimerkiksi logitauluja on kaksi. Toiseen tallennetaan kaikki määritelty toiminta tarkalla tasolla. Toiseen tauluun taas tallennetaan samaa tietoa vain päätasolla, josta selviää muutoksen tekijä, tehty toimenpide ja toimenpiteen kohde. Ensimmäiseen kaikki tiedot sisältävään logitauluun ei tehdä mitään muutoksia sovelluksen puolesta, mutta toista taulua seurataan ja sieltä otetaan raportteja.

On haluttu pitää varmuus siitä, ettei ohjelma pääse poistamaan logista mitään tärkeää tietoa. Toisaalta taas nopeampien hakujen takia tarvitaan taulu, josta ohjelma voi karsia turhaa ja vanhaa tietoa pois. Alkuperäiseen logitauluun ei tehdä muita kuin insert/lisäys -lauseita, jolloin asiakkaan pyytäessä tarkkaa tietoa mitä tietylle asialle tai asiakirjalle on tapahtunut, se voidaan myös varmasti antaa.

Ohjelman oma dynaaminen logitaulu taas tekee muutoksia tiettyihin kenttiin ja sitä voidaan käyttää vapaammin ohjelman omassa toiminnassa hyödyksi. Karsitumpi logitaulu myös nopeuttaa siihen tehtäviä hakuja huomattavasti, joten käyttäjälle näkyvä ohjelman toimintanopeus pysyy järkevänä.

Toisena esimerkkinä voidaan pitää asiakirjojen ja asioiden hakua. Tässä samaa tietoa tallennetaan kahteen eri tauluun, vaikka kyse onkin vain osasta tiedoista. Asiakirjojen kaikki tieto tallentuu yhteen tauluun, mutta kriittiset tiedot tallentuvat myös toiseen ns. hakutauluun.

Kriittisillä tiedoilla tarkoitetaan mm. asiakirjan nimikettä. Suureen massaan tehtäviä asioiden ja asiakirjojen hakua tutkittiin kehitysvaiheessa ja huomattiin, ettei pelkkä näkymä riittänyt takaamaan riittävän hyvää käyttönopeutta. Tultiin siihen tulokseen, että asiakirjojen ja asioiden haku saatiin käytettävyyden kannalta riittävän sujuvaksi ainoastaan luomalla toinen taulu, josta löytyy asiakirjan nimikkeiden lisäksi viittaus varsinaiseen asiakirjaan.

Tietojen tallentaminen useampaan tauluun tekee kantarakenteesta hankalasti hallittavan, mutta siihen pyritään pääsemään kiinni myös dokumentaation kautta. Aikaisemmin kyseinen tieto on ollut vain sovelluskehittäjillä ja pienen teknisen henkilöstön tiedossa. Nyt tietoa saadaan levitettyä, eikä tarvita sovelluskehitystä selvittämään tiettyjä erikoisia toimintatapoja.

7.3 Mitä parannettavaa kantarakenteessa on?

Suurin muutostarve löytyy oikeastaan tietokantatuotteista. Nykyinen kantarakenne on rakennettu MS sql server 2000 -standardin mukaan, jossa oli vielä mukana tietyt sql komennot. Uudemmissa MS sql servereissä näitä ei enää ole, joten kantarakennetta olisi uudistettava. Tavallaan tämä muutos liittyy tietokantarakenteeseen, mutta oikeastaan muutos on tehtävä sovelluskoodin puolelle. Uudemmat tietokantatuotteet eivät tue vanhoja SQL komentoja, joten komentoihin täytyy tehdä muutos sovelluskoodin puolelle.

Kosmeettisia muutoksia kantarakenteeseen tuntuisi olevan turhien taulujen ja sarakkeiden poisto, mutta tällä hetkellä kyseessä on aika pienen prioriteetin parannus. Vanhoista roikkuvista tauluista ja sarakkeista kun ei sinänsä ole mitään haittaa.

Laajoihin kantarakenteen muutoksiin ei kannata lähteä, sillä se vaatisi käytännössä koko tietokannan uudelleensuunnittelun. Tämä taas aiheuttaisi sovelluksen rakentamisen kantarakenteeseen liittyvien lauseiden osalta kokonaan uudelleen, joten kyseessä

on liian raskas ja kallis toimenpide. Tällä ei myöskään saavutettaisi riittävää hyötyä ohjelman toiminnan kannalta.

Johdannossa pohdittiin onko ohjelmaa käyttävällä organisaatiolla mahdollisuutta vaikuttaa tietokantarakenteeseen? Vaikka tietokantarakenne onkin tärkeä osa itse asianhallintajärjestelmää, ei asiakas voi tietokannan rakenteeseen käytännössä vaikuttaa. Tietokannan rakenne jää täysin sovelluskehittäjien tehtäväksi.

Asiakasorganisaatio pystyy kuitenkin käytön myötä näkemään tietokannan rakenteesta aiheutuvat ongelmat, joista helpoiten havaittavia ovat nopeuteen liittyvät virheet. Käytännössä huonosti rakennettu tietokanta tulee vastaan viimeistään järjestelmän kuormitustestauksissa, joissa tehdään tietojen tallennusta ja normaalia käyttöä vastaavaa simulaatiota suurella asia- ja asiakirjamassalla.

7.4 Miten tietokannan taulut on nimetty ja mihin tällä on pyritty?

Tietokannan taulut alkavat kaikki samalla Q -kirjaimella. Kirjaimella ei sinänsä ole merkitystä, mutta tällöin saman asianhallintajärjestelmän kaikki tietokantataulut listautuvat peräkkäin, kun niitä tutkitaan suoraan kantatyökaluilla. Tämän toimintatavan ainoa syy onkin ollut selkeyden saaminen taulurakenteeseen.

Joissain harvoissa tapauksissa samaan tietokantainstanssiin on lisätty useampien sovelluksien tietokantoja. Tällöin voidaan olla lähes varmoja, ettei samannimisiä tauluja ole muissa sovelluksissa.

Q -kirjaimen käyttö ohjelman taulujen nimessä tuo selkeyden, kun samaan tietokantaan lisätään muita oheisohjelmia. Esimerkiksi arkistonmuodostussuunnitelman hallintaan luotu ohjelma asennetaan usein samaan tietokantaan asianhallintajärjestelmän kanssa, mutta sen käyttämät taulut on nimetty eri tavalla. Tällöin nähdään heti kantarakennetta selatessa, että kumpaan järjestelmään tietty tietokantataulu kuuluu.

Lisäksi asianhallintajärjestelmässä on pyritty erottamaan tiettyihin toimintoihin kuuluvat taulut seuraavalla kirjaimella. Asioihin, asiakirjoihin, kansioihin, kokoustenhallintaan, oikeuksiin, käyttäjätietoihin, toimenpiteisiin ja toimeksiantoihin liittyvät taulut alkavat kaikki omalla alkukirjaimellaan.

Esimerkkinä voidaan pitää asiakirjoihin liittyvää päätaulua Qdoc, jossa ensimmäinen kirjain Q -tarkoittaa, että taulu on osa asianhallintajärjestelmää. Toinen kirjain D taas tarkoittaa kyseessä olevan asiakirjan.

Tähän liittyen seuraava taulu Qdocformat taas tarkoittaa alun osalta samaa, mutta loppu kertoo kyseessä olevan formaatteihin liittyvä taulu. Eli voidaan nimen perusteella päätellä varmaksi, että kyseinen taulu liittyy asiakirjoihin. Lopun format -nimen perusteella voidaan tehdä päätelmä, että taulu liittyy asiakirjan tiedostoformaatteihin.

Taulurakenteen osalta jokainen taulu on pääteltävissä samalla tavalla. Jokaisesta taulusta voidaan päätellä mihin kontekstiin se liittyy, eli onko kyseessä asioihin, asiakirjoihin, toimenpiteisiin, toimeksiantoihin, käyttäjiin, kansioihin, oikeuksiin vai kokoustenhallintaan liittyvä taulu. Tämä tuo selkeyttä kantarakenteeseen, mutta vei oman aikansa, ennen kuin taulujen lyhenteisiin pääsi kiinni.

7.5 Mitä tietokantaan tallennetaan?

Tietokantaan tallennetaan kaikki tieto, mitä asianhallintajärjestelmän kautta syötetään. Myös osa ohjaustiedoista on tallennettu tietokantaan, jolloin niiden hallinta on helppoa ja osa ohjaustiedoista voidaan avata myös asiakkaan pääkäyttäjälle. Tällöin asiakkaan pääkäyttäjällä on keino muokata hallintasovelluksen kautta asianhallintajärjestelmän ohjaustietoja. Osa ohjaustiedoista on tarkoituksella lukittuna, mutta osaa voidaan lähteä muokkaamaan myös pääkäyttäjän toimesta.

Oikeastaan ainoa tieto, mitä ei tallenneta tietokantaan, on asiakirjamassa. Valmiita asiakirjoja, eli fyysisiä tiedostoja, ei tallenneta tietokantaan. Tietokantaan tallennetaan ainoastaan viittaus ja metatiedot mistä kyseinen asiakirja löytyy. Varsinaisia tiedostoja ei kannata lähteä syöttämään binääridatana tietokantaan. Tiedostojen syöttäminen tietokantaan aiheuttaa tietokannan kasvamisen kohtuuttoman suureksi, jolloin syntyy ongelmia varmuuskopioiden kanssa. Suurin syy olla syöttämättä tietokantaan tiedostoja on varmasti suorituskyvyn laskeminen. Jos tietokantaan lisätään suuria tiedostoja, tietokannan nopeus laskee ja hakujen käyttämät ajat kasvavat.

Hyvänä puolena tiedostojen tallentamisella tietokantaan voisi pitää tiedostojen ja metatietojen olemista ajan tasalla. Jos tietokantaa jouduttaisiin palauttamaan muutama tunti taaksepäin, tiedosto palautuisi tässä tapauksessa myös samaan ajankohtaan. Jos tiedosto on erillään tietokannasta, näin ei voitaisi tehdä. Tässä toimintatavassa kuitenkin tiedostojen tallentaminen tiedostojärjestelmään levylle on järkevämpää, kuin niiden tallentaminen tietokantaan.

Nykytilanteessa ohjelmien pitäisi pystyä toimimaan nopeasti ja vaivattomasti, eikä tällöin voida ohjelman käyttömukavuudesta tinkiä tietyn teknisen toimintatavan takia.

7.6 Miksi dokumentointi on jäänyt tekemättä?

Jostain syystä ainoastaan sovelluksen lähdekoodi on dokumentoitu riittävän tarkalla tasolla, mutta tietokannan rakenne on jäänyt lähes kokonaan huomioimatta. Käytännössä kantarakenteen dokumentointi on jäänyt vain sovelluskehityksen omiin muistiinpanoihin ja sinnekin kohtuullisen suppealla tasolla.

Tietokantarakenteen dokumentoinnin puute on ollut tiedossa jo pitkään, mutta sovelluksen jatkuvan kehityksen myötä siihen ei ole pystytty varaamaan riittävästi resursseja. Tietokantarakenteen avaaminen vaatisi teknistä osaamista ja laajaa konsultointia sovelluskehityksen puolelta.

Myös sovelluskehityksen puolelta on perusteltu kantarakenteen kuvauksen puutetta sillä, että sovellus kehittyy jatkuvasti. Kun kantakuvaus saadaan valmiiksi, se on käytännössä heti vanhentunut. Tämä on tietysti otettava jatkossa huomioon. Kun dokumentointi saadaan tehtyä, sitä on myös päivitettävä riittävän usein. Vanhentunutkin kantakuvaus auttaa ongelmien selvittelyssä, sillä päätoiminnallisuus ei tule muuttumaan.

Täydellisen kantakuvauksen tekeminen valmiista sovelluksesta myös työllistää paljon, joten siihen ei ole pystytty sitomaan yhtä henkilöä riittävän pitkäksi aikaa. Sovelluksen jatkuvasti kehittyessä ja tietokantarakenteen laajentuessa on tietokantakuvauksen tekeminen ollut aina vain suurempi työ.

Yrityksen laajentuessa palkataan lisää väkeä niin sovelluskehitykseen kuin tekniseen tukeenkin. Uusilla henkilöillä ei ole tietämystä sovelluksesta, joten heitä varten täytyy jatkossa olla riittävä ohjeistus ja materiaali. Aikaisemmin yritykseen palkattiin uutta henkilöstöä niin harvoin, että ohjeistus pystyttiin hoitamaan henkilökohtaisella opastuksella.

SÄHKE2-määrityksen mukana tulevat avoimet rajapinnat velvoittavat asianhallintajärjestelmää tukemaan laajoja integraatioita toisiin järjestelmiin. Tämän tueksi tarvitaan sisäistä kantakuvausta, joka helpottaa oheisohjelmien ja integraatiokuvausten tekemistä.

7.7 Tietokannoissa olisi mahdollista käyttää jo luontivaiheessa valmiita liitäntöjä toisiin tauluihin

Tietokantatuotteet mahdollistavat primary key – foreign key -tyylisen viite-eheyden varmistamisen suoraan tietokantarakenteen kautta. Tätä ei kuitenkaan kaikissa tilanteissa käytetä. (Hovi, Huotari, Lahdenmaki 2005 12-15; Hernandez 2000 212, Hakkarainen 2012 116.)

Primary key – foreign key -viittaukset tuovat rajoitteita kantarakenteen käsittelylle. Tällöin tietokanta valvoo itse taulujen toimintaa, eli esimerkiksi päätaulua ei voi muuttaa tai poistaa ennen kuin siihen viittaavat alitaulut on poistettu, tai niiden viittaukset päätauluun katkaistu. Tämä on joissain tilanteissa hyvä asia, mutta se on nähty myös haittaavana tekijänä tietokantarakenteen päivityksissä. Etenkin, kun on käytetty vanhempia kantatuotteita (MS sql server 7, oracle 5.x).

Ilman tietokannan itse tekemää viite-eheyden tarkistusta on saatu varmistettua kantatuotteen mahdollinen vaihto toiseen, eli ohjelman kantarakenne on mahdollista siirtää kokonaan toiseen tietokantatuotteeseen.

Kuitenkin uudempien kantatuotteiden tekemät viite-eheyden tarkastukset tuntuvat olevan toimivampia ja edut menevät haittojen edelle. Kokonaan uusiin oheistuotteisiin ja niiden kantarakenteeseen on otettu käyttöön kiinteät viittaukset toisiin tauluihin. Tätä kautta kantarakenne uudistuu vähitellen, mutta se tietysti ottaa oman aikansa.

7.8 Tuetut tietokantatuotteet ja onko niitä mahdollista laajentaa?

Asianhallintajärjestelmä tukee MS sql serveriä ja Oraclen kantatuotetta. Näiden kahden yleisen kantatuotteen tukeminen on ollut tietoinen päätös. Kaupallisten tuotteiden osalta on uskottu mahdollisten tietoturva-aukkojen korjaamiseen nopealla aikataululla ja on voitu siirtää vastuu tietokannan ongelmista kantatuotteen toimittajalle. Ilmaisten open source -tyylisten tietokantatuotteiden ongelmana ovat aina mahdollisten korjauspäivitysten aikataulut ja mahdollisen tuen loppuminen joissain tapauksissa nopeastikin.

Tavallaan siis kaupallisia tuotteita on pidetty toimintavarmempina, kuin ilmaisia vapaaehtoistyövoimalla tehtyjä projekteja.

Ilmaista mysql -kantatuotetta ei ole otettu tuettujen tietokantatuotteiden piiriin oikeastaan sen takia, ettei yrityksellä ollut aikaisempaa kokemusta Mysql:sta. Kyseessä on myös open source -tyylinen tuote, jonka kehitys ja ylläpito on vapaaehtoisten kehittäjien vastuulla. Mysql:n osalta tämä on tosin muuttunut muutama vuosi sitten, kun Oracle osti Mysql:n, joten Oraclella on tavallaan kehitysvastuu Mysql -tuotteesta. Kuitenkin Mysql:n lähdekoodi on avointa, joten sitä pystyy kuka tahansa kehittämään eteenpäin.

Käytön laajentaminen tukemaan muita tietokantatuotteita työllistäisi myös jo nykyisinkin kiireistä sovelluskehitystä ja teknistä tukea lisää. Jouduttaisiin opiskelemaan kokonaan uuden kantatuotteen toiminnot, eikä tälle nähdä varsinaista tarvetta. Vaikka tietokannat tukevatkin pääosin samoja lauseita, tuotekohtaisia eroavuuksia on kuitenkin olemassa.

Etuna ilmaisten kantatuotteiden osalta voidaan pitää nykytilanteessa olevan erittäin tiukan kilpailutilanteen joustamista. Jos yritys pystyisi tarjoamaan tuotteensa hieman halvemmalla, osa menetetyistä tilauksista olisi saatettu saada. Kuitenkin uuden kantatuotteen opettelu ja tukeminen vaatisi työtunteja niin paljon, ettei tähän ole vielä lähdetty.

Kantatuotteita on olemassa huomattavasti enemmän, mutta tuetut kaupalliset MS sql server ja Oracle ovat varmasti tunnetuimmat. Kuten myös Mysql ilmaiselta puolelta.

7.9 Miten järjestelmän käyttäjätunnukset hoidetaan

Asianhallintajärjestelmän käyttäjätunnukset tallennetaan suoraan tietokantaan. Tietokantaan tallentuu käyttäjätunnus ja siihen liittyvä salasana. Salasana ei tallennu selkokielisenä, vaan kyseessä on salattu kenttä. Salasanan salaamiseen käytetään yleistä MD5 -algoritmia.

Tietokantaan tallentuvat käyttäjätiedot ylläpidetään sovelluksen kautta. Tunnuksia voidaan passivoida, jolloin käyttäjätunnus on olemassa, mutta sillä ei voida kirjautua sisälle järjestelmään.

Muita kirjautumistapoja on toteutettu, mutta sovelluksen oma kirjautumistapa on edelleen yleisin. Suuremmissa organisaatioissa käytetään kirjautumiseen AD -käyttäjätunnuksia ja sen tukena on myös SSO -kirjautumistapa. Tällöin käyttäjän ei missään vaiheessa tarvitse syöttää omaa käyttäjätunnusta tai salasanaa järjestelmään, vaan ohjelma luottaa käyttöjärjestelmään kirjautuneeseen tunnukseen.

AD -kirjautumisessa on käyttäjätunnuksien hallinta siirretty kokonaan AD:n puolelle, eikä asianhallintajärjestelmän tarvitse kuin tarkistaa tunnukset AD:sta.

7.10 Triggerit, proseduurit ja näkymät

Triggereistä on pyritty vähitellen pääsemään eroon, mutta niitä käytetään edelleen käyttäjätunnukseen liittyvissä tauluissa. Triggerit ovat muuten näppäriä, mutta niiden toimintaa on vaikea hallita. Triggerit toimivat aina piilossa käyttäjältä, eikä niiden toiminta jää logeihin kiinni.

Käyttäjätauluissa toiminta on kuitenkin perusteltua, sillä uutta käyttäjää luotaessa tai sitä muokattaessa, tehdään lisäyksiä moneen eri tauluun. Kun käyttäjätunnus luodaan, triggereillä toteutetaan käyttäjän omat kansiot ja henkilökohtaiset toimeksiannot. Käyttäjätunnusta poistettaessa ei kuitenkaan aina tule huomioitua, että samat triggerit toimivat myös poistovaiheessa. Tunnuksen poisto aiheuttaa myös samojen henkilökohtaisten kansioiden ja toimeksiantojen poistamisen järjestelmästä.

Proceduureja käytetään käytännössä vain järjestelmän versiopäivityksen yhteydessä. Näillä pystytään käymään laaja määrä tietoa läpi ja päivittämään siihen liittyvät muutokset tietokantaan. Proceduureja voitaisiin käyttää myös laajempia hakutoiminnallisuuksia käyttävissä hauissa.

Näkymät ovat tavallaan dynaamisia tietokantatauluja, joiden materiaali haetaan useammasta eri taulusta. Näitä käytetään pääosin erilaisten raporttien ottamiseen ja dynaamisten listojen päivittämiseen.

8 NYKYISEN TIETOKANNAN OPTIMOINTI JATKOSSA

Nykyinen tietokanta on kohtuullisen hyvin optimoitu, sillä perustoiminnallisuus on ollut käytössä jo useita vuosia. Tietokantaan tehtävät haut on optimoitu niin sovelluskoodin puolelta, kuin myös rakentamalla tehokkaat indeksit tietokantaan.

Jatkuvaa kehitystä myös optimoinnin puolelta tapahtuu, sillä ohjelmaan tehdään jatkuvaa sovelluskehitystä. Uudet ominaisuudet rakennetaan heti alusta lähtien toimimaan nopeasti. Kuitenkaan kaikkiin toimintatapoihin ei osata varautua, joten myös tietokannan optimointi täytyy huomioida tuotantoon siirron jälkeen. Ohjelmat testataan laajasti uusien versioiden jälkeen, mutta testauksessa ei osata aina huomioida käytetyn materiaalin laajuutta tai asiakkaiden käyttämiä toimintatapoja.

Asian- ja asiakirjanhallintajärjestelmä toimitetaan aina täytenä pakettina, mutta siitä rajataan pois ne osuudet, joita kyseinen asiakas ei käytä. Tällöin täysi paketti sisältää aina kaikkiin tauluihin luotavat indeksit, eikä indekseissä ole asiakaskohtaisia räätälöintejä. Jos havaitaan yhdellä asiakkaalla ongelmia jonkun haun hitauden kanssa, siihen toteutetaan nopeana korjauksena ensin indeksin uudelleenrakennus ja tarvittaessa rakennetaan erilaisilla ehdoilla rinnalle kokonaan uusi indeksi. Jos uusi indeksi havaitaan käytössä toimivaksi, se otetaan laajempien testausten jälkeen osaksi kokonaisjärjestelmää.

Jos indeksin luominen ei auta, lähdetään selvittämään asiakkaan käyttämiä hakuehtoja tarkemmin, joiden perusteella tutkitaan mihin hitaus perustuu. Sovelluskoodin puolelle tehtävät hakulauseiden optimoinnit ovat aina hitaita toteuttaa ja tulevat käytännössä

mukaan vasta seuraavaan sovellusversioon. Vanhaan versioon ei lähdetä toteuttamaan kuin kriittisiä korjauksia.

Usein suurin osa nykyversioon tehtävistä optimoinneista tapahtuu indeksien kautta. Osassa tapauksista tietokantatuote ei osaa optimoida oikean indeksin käyttöä, vaan sille on määriteltävä käsin halutun indeksin käyttö. Tämä on nykyään yhä harvinaisempaa, mutta joskus sitäkin esiintyy.

Erityisesti uusien ominaisuuksien osalla esiintyy hakujen hitautta. Ei esimerkiksi ole osattu huomioida, että jokin organisaatio käyttää toimeksiantoja niin suuressa määrin, että pelkkä otsikoiden päivittäminen sovellukseen kestää useita kymmeniä sekunteja. Tämä pystyttiin korjaamaan luomalla indeksi kyseiseen tauluun, jolloin otsikoiden päivittäminen nopeutui huomattavasti.

Tietokannan ylläpitorutiineihin tullaan jatkossa kiinnittämään enemmän huomiota. Erilaiset tietokantojen varmistuksiin liittyvät rutiinit, kuten statistiikkojen päivitykset, on hyvä käydä uudelleen läpi. Aikaisemmat ohjeet ovat SQL Server 2000 –aikaisia, joihin on tullut muutoksia uudempiin SQL Servereihin. Vanhat komennot toimivat edelleen ja ovat varmasti hyödyksi, mutta nykyisissä tietokantatuotteissa on enemmän ja parempia ylläpitorutiineja.

Usein tietokantojen hallintarutiinit jäävät sovellustoimittajan tehtäväksi, vaikka tietokantatuote olisikin asiakkaan itsensä hallinnoima. Jos asiakkaalla itsellään ei ole tietoa tai osaamista hallinnoida tietokannan yöllisiä rutiineja, usein sovellustoimittajilta kysytään neuvoa ja ohjeistusta miten sovellustoimittajan tietokantaa olisi hyvä hallinnoida.

Yöllisiin hallintarutiineihin olisi hyvä saada mukaan niin indeksien uudelleenrakennus, tietokannan yhtenäisyyden tarkistus, tietokannan statistiikkojen päivitys ja ehdottomasti myös tietokannan varmuuskopiointi. Näillä toiminnoilla saadaan tietokanta pysymään nopeana ja sen käyttö on sujuvaa myös vuosien kuluttua.

Usein nämä rutiinit tehdään myös manuaalisesti versiopäivitysten yhteydessä, jolloin tarkistetaan samalla myös tietokannan rakenne. Tällöin seurataan mahdollisia varmuuskopiointien epäonnistumisia, vaikka ne eivät sovellustoimittajan vastuulla ole-

kaan. On kuitenkin molempien osapuolten etu, että tieto pysyy tallessa ja ohjelmaa on mukava käyttää.

9 TYÖN LOPPUTULOS

Työn lopputuloksena saatiin yritykselle dokumentoitua asian ja asianhallintajärjestelmän tietokannan rakenne. Tämä tietokannan rakenne selvitettiin tuotekehitystä konsultoiden, joten kantakuvauksesta saatiin riittävän laaja ja luotettava. Tietokannan kuvauksesta selviää jokaisen tietokantataulun tehtävä ja jokaisen sarakkeen sisältö ja niiden mahdolliset arvot. Esimerkkinä voidaan pitää asiakirjan julkisuustilaa, joka merkitään arvoilla 0-3. Näistä yksi tarkoittaa julkista asiakirjaa, yksi salaista, osittain salaista ja sisäistä tietoa. Arvojoukkoa on myös mahdollista kasvattaa jälkeenpäin, jos asiakasorganisaatiossa olisi käytössä vielä jokin oma julkisuustilaa merkitsevä termi.

Tehdystä kantaselvityksestä on hyötyä asiakastuessa mahdollisten ongelmatapausten selvittelyssä. Nyt pystytään selvittämään tietokannan kautta mm. asiakirjan julkisuustila, eikä tätä tarvitse ensin selvittää sovelluskehityksen puolelta. Asiakastuella on ollut vastaava kantaselvitys muista yrityksen ohjelmista, mutta laajimmasta asianhallintajärjestelmästä tämä on puuttunut. Kantaselvityksiä on totuttu käyttämään muiden ohjelmien osalta, joten käyttöönotto uuden dokumentaation puolesta ei tule olemaan ongelma.

Työn hyöty sovelluskehitykselle on vielä epäselvää, mutta tietokannan dokumentointi on ainakin tueksi myös sovelluskehitykselle. Heillä on omat merkintänsä olemassa ja sovelluskehityksessä uuden toiminnon luominen vaatii joka tapauksessa tarkan selvityksen lähdekoodin puolelta. Sovelluskehitykselle tietokantadokumentointi näkyy ehkä parhaiten, kun tekninen asiakastuki pystyy selvittämään ongelmat ilman tuotekehityksen apua. Tällöin heidän työaikaa ei tarvita aivan kaikkien ongelmien selvittämisessä. Vaikka dokumentaatio ei poistakaan kaikkea sovelluskehitystiimiä vaativaa ongelman selvittelyä, huomattavasti se sitä kuitenkin vähentää.

Työn alussa mietittiin mitä hyötyä tietokannan dokumentoinnilla on yritykselle. Yrityksen hyöty tietokantadokumentaatiosta näkyy tietoturvan kannalta oleellisesti. Nyt yritykselle tärkeä tieto on saatu dokumentoitua, eli tieto on muuallakin kuin sovelluskehityksen arkistoissa. Kyseessä on myös rahallisesti kannattava työ, sillä käytetyt

työtunnit kalliilta ohjelmistokehittäjiltä voidaan nyt käyttää muuhun tuottavaan työhön.

Tietokannan dokumentointi on ollut kannattava työ. Dokumentointi olisi korkeintaan pitänyt tehdä jo vuosia aikaisemmin. Työ on saatu valmiiksi, mutta se vaatii jatkuvaa ylläpitoa ja hallintaa. Tietokanta muuttuu sovelluksen kehittyessä jatkuvasti, mutta jatkossa tulevat muutokset on helppo käydä läpi, kun työmäärä niiden lisäämisessä on kohtuullisen pieni. Työn saattaminen nykyiselle tasolle oli työläs ja aikaa vievä projekti, mutta hyvinkin kannattava jatkon kannalta.

Tietokannan dokumentointi antoi mahdollisuuden lähteä siivoamaan vanhaa tietoa pois kantarakenteesta. Käytännössä tällä tarkoitetaan käytöstä poistuneita tietokantatauluja ja sarakkeita, jotka eivät enää ole missään käytössä. Kun ylimääräistä aikaa löytyy, voidaan lähteä tutkimaan asiaa tarkemmin.

Tietokannan rakenteesta huomaa, että tietokannan tehnyt henkilö ymmärtää tietokannan toimintaperiaatteet ja on pyrkinyt tekemään tietokannan relaatiotietokantojen mallin mukaan. Tietokannoissa noudatetaan normaalimuotoa siltä osin kuin se on järkevää. Suurta korjattavaa ei kantarakenteesta löydy ja ainoa huomiota herättänyt asia ovat vanhat poistuneet taulut, joita edelleen roikotetaan mukana. Tietokanta on myös suunniteltu siten, että sen jatkuva päivittäminen ja jatkokehitys on otettu huomioon.

Asian- ja asiakirjanhallintajärjestelmän tietokantarakenne oli itselle päätaulujen osalta tullut tutuksi, mutta suurin osa tietokantarakenteesta oli vielä epäselvää. Opinnäytetyö oli haastava ja vei enemmän aikaa mihin olin varautunut, mutta työn edetessä oma mielenkiinto vain kasvoi ja tuntui mukavalta huomata, että tietokannan alun perin rakentanut henkilö on hyvinkin loogisesti ajatellut kantarakennetta luodessaan. Tämä ajatustapa vain piti ensin ymmärtää, jotta kokonaisuuden pystyi sisäistämään.

Työtä tehdessä kiinnostuin myös tietokannan optimoinnista. Indeksien ja clusteroitujen indeksien vaikutus tietokannan optimoinnissa oli jollain tasolla tuttua, mutta en tiennyt niiden toimintaa riittävän tarkalla tasolla. Kantarakennetta selvittäessä tuli tutkittua myös sovelluksen tietokannan indeksejä tarkemmin. Jatkon kannalta tulen varmasti osallistumaan tietokannan optimointiin ja ylläpitoon liittyville kursseille, sillä oma tietämys ei tältä osin ole sillä tasolla kuin se voisi olla. Tietokannan optimointi on

pääosin sovelluskehittäjien tehtävä, mutta oman työni kannalta joudun selvittelemään tietokannan hitauteen liittyviä ongelmia. Tämän tueksi olisi hyvä ymmärtää tarkemmin miten indeksit, clusteroidut indeksit ja muut hakujen nopeuteen liittyvät asiat vaikuttavat.

Tietokannan jatkokehitys jatkuu normaalilla rutiinilla, mutta vanhojen poistuneiden taulujen ja sarakkeiden siivoaminen on tuotu tuotekehityksen tietoon. Se ei varmasti kovin suurella prioriteetilla ole, mutta kuitenkin mukana työjonossa. Myös indekseihin ja tietokannan optimointiin tullaan kiinnittämään enemmän huomiota.

Opinnäytetyönä tehty kantarakenteen selvitys on jo nyt nopeuttanut tuotekehityksen jatkokehitystä, sillä tarvittavaa tietoa ei ole tarvinnut etsiä aina uudelleen, vaan on voitu katsoa suoraan dokumentaatiosta.

Työn myötä ymmärrettiin tietokannan dokumentoinnin arvo myös asiakasorganisaatioiden osalta. Vaikka asiakas ei itse näekään tietokannan dokumentaatiota, dokumentoinnin ajanmukaisuus nopeuttaa asiakkaan ongelmien ratkaisemista. Usein asian – ja asiakirjanhallintajärjestelmä on hyvin keskeinen järjestelmä kuntien ja julkishallinnon käytössä, joten siihen liittyvät ongelmat peilaavat myös muihin taustajärjestelmiin. Tästä johtuen on mahdolliset asianhallintajärjestelmän ongelmat saatava korjattua mahdollisimman nopeasti ja tältä osin tehty tietokantarakenteen dokumentaatio on iso apu.

Työn myötä saatiin vastaukset kysymyksiin, joita pohdittiin työn aikana. Opinnäytetyön tuloksena syntyi myös itselle tärkeä dokumentaatio päivittäisen työn tueksi.

LÄHTEET

Hovi Ari. 2004. SQL-opas. 1. Painos. Jyväskylä: Docendo Finland Oy.

Hovi Ari, Huotari Jouni & Lahdenmäki Tapio, 2005. Tietokantojen suunnittelu & indeksointi. 2. Painos. Jyväskylä: Docendo Finland Oy.

Turunen Janne, 2013. Tietoturva. Mikkelin ylemmän amk:n sähköisen asioinnin ja arkistoinnin luennot 2013.

Nenonen Markku, 2013. Opintomateriaali, kokonaisarkkitehtuuri

Mynttinen Timo, 2009. Tietokantaohjelmointi.

Inspecta 2014, www-dokumentti.
http://www.inspecta.com/fi/Palvelut/Sertifiointi/Jarjestelmasertifiointi/Tietoturvajarjestelman-sertifiointi-ISO-IEC-27001/. Luettu 30.5.2014.

JHS –suositukset 2014, www-dokumentti.
http://www.jhs-suositukset.fi/. Luettu 30.5.2014.

Bryla Bob, Loney Kevin, 2008. Oracle Database 11g, DBA Handbook.

Fernandez Iggy, 2009.Beginning Oracle Database 11g Administration.

Microsoft Coursebook, 2011. Implementing a Microsoft SQL Server 2008 R2 Database.

Elmasri Ramez, Navathe Shamkant, 2002. Fundamentals of Database Systems 4th edition.

Salminen Ari., Metatiedot organisaatioiden sisällönhankinnassa, www-dokumentti.
http://users.jyu.fi/~airi/papers/Metatietoartikkeli-2005.pdf. Luettu 30.5.2014.

Seppä Ilkka, 2010. Asianhallintajärjestelmän metadata. Pro gradu-tutkielma Aalto yliopisto.

Michael J. Hernandez, 2000. Tietokannat suunnittelu käytännössä

Hakkarainen Anssi, 2012. Oracle tietokannan tehokas hallinta

Keskikiikonen Mika, 2001. Tietokannat

Tietotekniikan liitto, 2002. Tietojärjestelmän hankinta

Sainio Arto, 2002. Tietovarastotekniikan perusteet

Arkistolaitos Sähke2 metatietomalli 2014, www-dokumentti.
http://www.arkisto.fi/uploads/normit/valtionhallinto/maarayksetjaohjeet/Liite2_Metatietomali.doc. Luettu 30.5.2014.

LYHENTEET

SÄHKE2

Arkistolaitoksen määräys niistä vaatimuksista ja ominaisuuksista, jotka ovat edellytyksenä asiakirjojen pysyvälle säilyttämiselle yksinomaan sähköisessä muodossa.

SQL

Structured query language, Tietokantakieli

Primary key

Perusavain. Käytetään tietokantarakenteessa viittaamaan yksilöivään avaimeen.

Foreign key

Viiteavain. Käytetään viittaamaan toiseen tauluun osoittavaan perusavaimeen.

Indeksi

Indeksillä tarkoitetaan tietokannoissa olevia sisällysluetteloita, joiden kautta tehty tietokantahaku on nopeampi kuin koko materiaalin käyminen läpi.

AD

Active Directory. Microsoftin tuote keskitettyyn työasemien hallintaan.

Asia

Asia on organisaatiolle määriteltyjen tehtävien mukainen yksi toimenpide tai käsittelyvaihekokonaisuus.

Asiakirja

Sisällön ja rakenteen muodostama kokonaisuus, joka täytyy toimintaympäristössä tuottaa.

Asiakirjanhallinta

Asiakirjojen elinkaaren hallinta ja siihen kuuluvat prosessit.