Erkki Mäkinen (toim.)

Pieniä tietojenkäsittely-tieteellisiä tutkimuksia (syksy 2003)

TIETOJENKÄSITTELYTIETEIDEN LAITOS
TAMPEREEN YLIOPISTO

B-2004-2

TAMPERE 2004

Lukijalle

Tähän julkaisuun on kerätty syyslukukaudella 2003 pitämälläni Tutkimuskurssilla tehdyt, määräaikaan mennessä valmistuneet suomenkieliset tutkielmat.

Toimittaja

Sisällysluettelo

Tekstidokumenttien klusterointi..... 1
Maritta Harjunpää
Semanttiset yhteydet tietokannoissa.....15
Janne Jämsen
Temporaalitietokannoista…….....41
Kimmo Kajas
Tietojärjestelmäprojektiin valmistautuminen yrityksen ollessa osa konsernia …….....53
Kari Kataja
Ravivedonlyönti internetissä …….....71
Tommi Kinnunen
Satelliittipaikannuksen tarkentaminen kaupunkiympäristössä.....83
Tommi Lehtinen
Tietotekniikan merkkipaaluja…….....95
Reko Linko
GPRS-verkkojen tietoturva…….....109
Jyrki Rantanen
NAT:n aiheuttamat ongelmat IPsec:lle ja niiden ratkaisu…….....119
Rauno Tamminen
Asiantuntijajärjestelmistä …….....141
Kirsi Varpa

Tekstidokumenttien klusterointi

Maritta Harjunpää

Tiivistelmä

Tutkielmassa tarkastellaan rakenteettomien tekstidokumenttikokoelmien automaattista ryvästämistä eli klusterointia, jonka tavoitteena on ryhmitellä johonkin kokoelmaan kuuluvat tekstidokumentit niiden sisällön perusteella. Käsittelen työssäni ensin klusterointiongelmaa yleisellä tasolla, jonka jälkeen käyn läpi klusterointiongelman ratkaisun vaiheita tekstidokumenttien ryhmittelyn yhteydessä.

Avainsanat ja -sanonnat: klusterointi, klusterointiongelma, tekstin louhinta, vektoriavaruusmalli, stemmaus, sulkulista
CR-luokat: I.5.3, I.2.7, H.3.3

1. Johdanto

Julkinen internet ja yritysten ja muiden organisaatioiden sisäisessä käytössä olevien intranetien sisältämien tekstidokumenttien lukumäärät ovat kasvaneet valtaviksi, joten ihmisen on vaikea enää hallita niiden sisältämää tietoa. Tämän vuoksi kaivataan apukeinoja järjestämään tätä tietoa. Eräs tähän tarkoitukseen käytetty menetelmä on tekstin louhinnan menetelmiin kuuluva klusterointi, joka tarkoittaa lyhyesti määriteltynä automaattista ryhmittelyä.

Tässä tutkielmassa tarkastellaan, miten klusterointia toteutetaan käsiteltäessä erityisesti tekstiä sisältäviä dokumentteja. Klusterointia voidaan soveltaa myös muun kuin tekstiaineiston käsittelyyn, mutta siihen ei tarkemmin paneuduta tässä työssä.

Tutkielman seuraavissa luvuissa käsitellään klusterointia yleisemmin, määritellään klusterointitermi yksityiskohtaisesti sekä esitellään klusteroinnin eri vaiheet. Tutkielman loppuosa käsittelee klusteroinnin vaiheita yksityiskohtaisemmin esitellen niihin liittyviä tekniikoita erityisesti tekstidokumentteja käsiteltäessä. Koska suuri osa klusterointia käsittelevistä julkaisuista on kirjoitettu englanniksi, olen useimpien tutkielmassani käytettyjen klusterointiin liittyvien suomenkielisten termien kohdalle sulkuihin lisännyt vastaavan englanninkielisen termin.

2. Yleiskatsaus klusterointiin

Tässä luvussa käydään lyhyesti läpi klusteroinnin tutkimusta, syitä aiheen kiinnostavuuteen sekä sen käyttökohteita. Luvun lopussa määritellään termi *klusterointi* (clustering) sekä esitellään klusterointiongelman ratkaisuvaiheet.

2.1. Klusteroinnin tutkimuksesta

Klusteroinnin tutkimusta on tehty jo vuosikymmeniä, vaikkakin kiinnostuksen määrä on vaihdellut aikakausittain. Esimerkiksi 1970-luvulla aihe herätti laajaa mielenkiintoa, kun taas 1980-luvulla kiinnostus aiheeseen oli vähäistä. Uusi intensiivinen kiinnostus aiheeseen heräsi 1990-luvulla sähköisten dokumenttien määrän alkaessa kasvaa ja niiden sisältämän tiedon hallittavuuden vaikeutuessa [Fasulo, 1999].

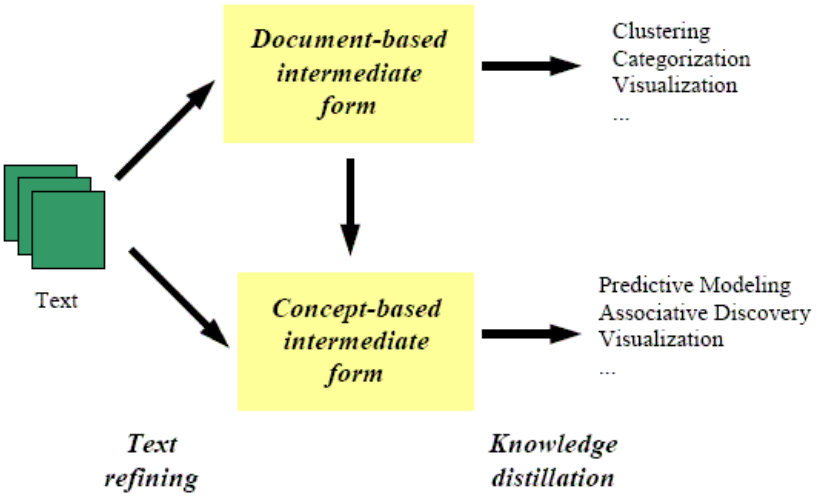
Syitä kiinnostuksen lisääntymiseen 1990-luvulla oli monia, mutta useimmiten mainitaan sähköisen tiedon määrän valtava kasvaminen. Suurten tietomäärien hallittavuus yleensä paranee, mikäli tietoa järjestetään jollain tavoin. Tämä tehtävä voidaan periaatteessa suorittaa manuaalisesti, mutta dokumenttien lukumäärät ovat usein niin isoja, että niiden manuaaliseen ryhmittelyyn kuluisi paljon aikaa ja näin ollen saattaisi olla hyvin kallista. Tekstidokumenttien sisällöstä riippuen saatettaisiin tarvita myös alan asiantuntemusta ryhmittelyn toteuttamiseksi. Tietokoneen suorittaman ryhmittelyn eduiksi lasketaankin yleensä se, että koneen suorittama ryhmittely on halpaa, se on nopeaa ja siihen ei vaadita tietyn erityisalueen asiantuntijan panostusta. [Beil et al., 2002; Rosell, 2002]

Automaattinen ryhmittely vaatii myös tietokoneelta resursseja, joten kiinnostusta klusterointiin on osaltaan lisännyt myös tietokoneiden laskentatehon ja tallennuskapasiteetin nopea lisääntyminen viime vuosikymmenen aikana, joka näin on tarjonnut paremmat edellytykset klusteroinnin soveltamiseen. [Zhao and Karypis, 2002]

2.2. Klusteroinnin käyttöalueita

Dokumenttien klusterointia voidaan hyödyntää tekstin louhinnassa, jossa taas hyödynnetään sekä tiedon louhinnan, kieliteknologian että tiedon haun yhteydessä käytettäviä menetelmiä [Neto et al., 2000]. Tiedonhaussa klusteroinnin tavoitteena on parantaa hakutulosten saantia ja tarkkuutta ryhmittelemällä dokumentit niin, että niitä voidaan selailla. Saanti tarkoittaa sitä, kuinka suuri osa kaikista hakijalle relevanteista eli hyödyllisistä dokumenteista sisältyy hakutulokseen ja tarkkuus sitä, kuinka suuri osa hakutuloksen kaikista löydetyistä dokumenteista on hakijalle relevantteja. [Carlberger et al., 2001]

Tekstin louhinta eroaa tiedon louhinnasta siinä, että tiedon louhinnassa käsitellään yleensä rakenteista tietoa, kun taas tekstin louhinnassa käsitellään rakenteetonta tietoa [Neto et al., 2000]. Tekstin louhinta voidaan jakaa kahteen vaiheeseen, tekstin jalostamiseen (text refining) ja tiedon tislaamiseen (knowledge distillation). Tekstin jalostamisessa luonnollisella kielellä kirjoitetut tekstidokumentit muutetaan valittuun välimuotoon (intermediate form). Välimuoto voi olla joko dokumenttipohjainen (document-based) tai käsitepohjanen (concept-based). Tiedon tislaaminen dokumenttipohjaisessa tekstinlouhinnassa tuottaa hahmoja tai tietämystä dokumenttikokoelmasta. Klusterointi sijoittuu tekstinlouhinnassa tähän. Tiedon tislaaminen käsitepohjaisessa tekstinlouhinnassa tuottaa hahmoja tai tietämystä, jota löytyy kokoelman dokumenttien sisältä. Kuvassa 1 näkyy tekstin louhinnan kehys [Tan, 1999].



Kuva 1. Tekstin louhinnan kehys.

Esimerkkinä tekstin louhinnan vaiheista voisi olla joukko uutisartikkeleita, jotka tekstin jalostamisen kautta ensin saatetaan haluttuun välimuotoon. Dokumenttipohjaiseen välimuotoon muutetun aineiston tiedontislausvaiheessa artikkelikokoelma järjestetään sisällön perusteella ryhmiin. Näin saadaan parempi kuva siitä, minkä aihealueen dokumentteja aineistoon kuuluu, ts. aineisto voidaan tehdä havainnollisemmaksi. Jos nyt haluttaisiin etsiä tietoa vaikkapa ryhmään ”yritys” kuuluvista teksteistä, voitaisiin se tehdä niin, että

kyseiseen ryhmään kuuluvista dokumenteista muodostetaan oma tietokanta, josta sitten voidaan hakea yrityksiin liittyvää tietoa.

Kieliteknologian piirissä Manning ja Schütze [1999] mainitsevat klusteroinnin käyttöalueeksi myös yleistämisen, jossa on kyse kielimallien tuottamisesta. Klusteroitujen havaintojen perusteella tehdään yleisiä päätelmiä sen perusteella, mitä tiedämme joistakin klusteriin kuuluvista havainnoista. Esimerkiksi yleistämisestä voidaan ottaa englannin kielen substantiivi *Friday* ja sen yhteydessä käytettävän preposition löytäminen. Oletetaan että käytössä on testimateriaali, joka sisältää fraasit *on Sunday*, *on Monday* ja *on Thursday* mutta ei fraasia *on Friday*. Klusteroinnin tuloksena viikonpäivät päätyisivät samaan klusteriin. Nyt voidaan päätellä, että jos prepositio *on* on oikea sanojen *Sunday*, *Monday* ja *Thursday* yhteydessä, sopii sitä käyttää myös sanan *Friday* yhteydessä. [Manning and Schütze, 1999].

2.3. Klusteroinnin määritelmä ja klusterointiongelman ratkaisemisen vaiheet

Edellä kävi ilmi, että klusterointi tarkoittaa automaattista ryhmittelyä. Tarkemmin määriteltynä *klusteroinnissa* (clustering) havainnoista muodostetaan klustereita, joiden sisällä havainnot ovat keskenään jollain lailla samankaltaisia ja klustereiden välillä erikaltaisia [Fasulo, 1999].

Englanninkielisissä tekstidokumenttien klusterointia käsittelevissä julkaisussa saattaa törmätä nimitykseen *automaattinen luokittelu* (automatic classifycation), jonka käyttö klusteroinnista puhuttaessa saattaa kuitenkin olla harhaanjohtavaa, sillä termillä *luokittelu* (classification) viitataan hieman eri asiaan. Luokittelu on *ohjattua oppimista* (supervised learning), sillä luokittelussa havainnot jaetaan ennalta määrättyihin luokkiin, kun taas klusteroinnissa luokitus syntyy klusteroinnin tuloksena ja näin ollen klusterointi on tyypiltään *ohjaamatonta oppimista* (unsupervised learning) [Maarek et al, 2000; Jain et al., 1999].

Klusterianalyysin toteuttaminen on hyvin monimutkainen tehtävä. Se ei ole mikään yksittäinen tekniikka, vaan koostuu ennemminkin useista vaiheista, joissa kussakin sovelletaan omia tekniikoitaan. Jokaisen vaiheen toteuttamiseksi käytettävien menetelmien valinta riippuu mm. siis siitä, mitä havaintoaineistoa on tarkoitus ryhmitellä. Kuvahavaintoaineistoa käsiteltäessä käytetään eri tapoja kuin tekstiaineistoa käsiteltäessä. Valintoihin vaikuttaa tietysti myös se, mihin tarkoitukseen klusterointia tarvitaan.

Havaintokokoelmaan kuuluvien havaintojen klusterointi voidaan jakaa seuraaviin vaiheisiin [Jain et al., 1999]:

- Syöteaineiston eli havaintojen kuvaaminen
- Havaintojen läheisyysmitan valinta
- Klusterointitekniikan valitseminen
- Tarvittaessa klusterointitulosten esittäminen
- Tarvittaessa tulosten arviointi.

Malli on hyvin yleisen tason kuvaus klusteroinnin vaiheista ja se soveltuu muunkin kuin tekstiaineiston ryhmittelyyn. Luonnollisen kielen sääntöjä noudattaen tuotetut tekstidokumentit, joihin tässä tutkielmassa keskitytään, sisältävät kuitenkin paljon aineksia, jotka täytyy huomioida ennen varsinaista klusterointia. Tämän vuoksi klusteroinnin vaiheisiin voidaan lisätä tekstin esikäsittely.

Seuraavissa luvuissa käydään klusteroinnin vaiheita läpi yksityiskohtaisemmin lukuun ottamatta kahta viimeistä vaihetta eli tulosten esittämistä ja tulosten arviointia.

3. Tekstidokumenttien esikäsittely ja kuvaaminen

Tässä luvussa kuvaillaan kaksi yleisesti käytössä olevaa tekstidokumenttien esikäsittelyn menetelmää, stemmaus ja sulkulistojen käyttö. Hyvin lyhyesti käydään läpi myös muuta tekstin esikäsittelyyn liittyvää. Luvun viimeisessä osassa esitellään vektoriavaruusmalli, joka on eräs tavallisimmin käytetty tekstidokumenttien kuvailutekniikka.

3.1. Stemmaus ja sulkulistat tekstidokumenttien esikäsittelyssä

Esikäsittelyn tavoitteena on vähentää käsiteltävän aineiston määrää havaintojen sisällä ja näin tehostaa itse klusterointia. Toisin sanoen tekstiä puhdistetaan niin, että vain sisällön kannalta tärkeät ainekset jäävät jäljelle ja vähemmän tärkeät poistetaan. Eräs keino tähän on *sulkulistojen* (stop words) käyttö. Sen idea perustuu olettamukseen, että tekstidokumentteihin sisältyvistä termeistä voidaan poistaa ne, joilla ei katsota olevan merkitystä dokumenttien sisällön kuvaajana. Tällaisina sanoina pidetään mm. artikkeleita, prepositioita ja konjunktioita (esim. *ja*). Myös sanat, jotka jostain muusta syystä esiintyvät kokoelman kaikissa tekstidokumenteissa tai sanat, jotka esiintyvät vain kerran dokumenttikokoelmassa, saatetaan kerätä sulkulistalle. [Hotho et al., 2003; Rosell, 2002]

Toinen piirredimensioiden vähentämiseen käytetty tekniikka on typistäminen eli *stemmaus* (stemming), joka tarkoittaa sanojen pääteainesten (sekä suffiksien että prefiksien) poistoa, jolloin jäljelle jää sanan vartalo. Jäljelle

jäänyt vartalo ei aina ole mikään sana vaan ennemminkin katkaistu sana. Etu- ja takaliitteet kielissä vaihtelevat, joten stemmaukseen liittyy aina kielikohtaista tietoa. Stemmauksen piirteisiin kuuluu myös se, että siinä eri sanaluokkiin kuuluvat sanat voivat saada saman muodon stemmauksen jälkeen. Esimerkiksi ruotsinkieliset sanat *cyklade* ja *cykel* saavat stemattuina muodon *cykl*. Kaksi yleisimmin käytettyä ovat Porterin ja Lovinin stemmerit, jotka ovat saaneet nimensä kehittäjiensä mukaan. [Carlberger et al., 2001; Manning and Schütze, 1999]

Stemmauksen sijasta voitaisiin myös käyttää kieliteknologian piirissä lemmaukseksi (lemmatization) kutsuttua tekniikkaa, jolla esiin saadaan sanan perusmuoto. Tämä on kuitenkin stemmaukseen verrattuna paljon työläämpää. Perusmuotoistaminen antaisi sanasta *cyklade* perusmuodoksi *cykla* ja sanasta *cykel* sanan *cykel*. [Carlberger et al., 2001]

Esikäsittelyvaiheeseen liittyy usein myös muiden kuin stemmauksen ja sulkulistojen käytön seurauksena poistettujen ainesten puhdistusta kuten esimerkiksi välimerkkien ja HTML-merkkausten poisto. Näiden lisäksi esikäsittelyyn saattaa sisältyä sanojen kirjoitusasun muuttaminen niin, että kaikki sanat kirjoitetaan pienillä kirjaimilla. [Beil et al., 2002]

3.2. Vektoriavaruusmalli tekstidokumenttien kuvaajana

Edellä on käynyt ilmi että klusteroinnissa verrataan tekstidokumenttien eli havaintojen sisällöllistä samanlaisuutta keskenään, joka toimii perusteena, sille mitkä havainnot kuuluvat samaan klusteriin. Klusterin sisällähän havainnot ovat keskenään samanlaisia ja klustereiden välillä erilaisia. Jotta tätä samanlaisuutta pystyttäisiin mittaamaan, täytyy tekstidokumentit ensin kuvata niin, että samanlaisuutta pystyttäisiin laskennallisesti mittaamaan. V*ektoriavaruusmalli* (vector space model) eli vektorimalliin on hyvin laajasti käytetty havaintojen kuvailutekniikka ja soveltuu myös tekstidokumenttien kuvailuun.

Perusajatus vektoriavaruusmallissa on se, että tekstidokumenteista poimitaan sanat, joita sen jälkeen kutsutaan dokumenttien piirteiksi. Dokumenttikokoelma voidaan esittää havaintomatriisina, jossa vaakariveillä ovat dokumenttikokoelman termit eli piirteet ja sarakkeissa ovat kokoelmaan kuuluvat tekstidokumentit. Termien painoarvot, jotka voidaan ilmoittaa esimerkiksi niiden esiintymistiheytenä dokumentissa, sijaitsevat matriisin soluissa. Kuva 2 on kaksiuloitteinen matriisi *A*, jossa kokoelmaan kuuluu *m* kappaletta dokumentteja *d* ja *n* kappaletta termejä *t*. Piirteiden painoarvot *w* kirjoitetaan matriisin soluihin [Manning and Schütze, 1999].

$$A = \begin{pmatrix} & d_1 & d_2 & \cdots & d_m \\ t_1 & w_{11} & w_{21} & \cdots & w_{m1} \\ t_2 & w_{12} & w_{22} & \cdots & w_{m2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_n & w_{1n} & w_{2n} & \cdots & w_{mn} \end{pmatrix}$$

Kuva 2. Matriisissa *A* kokoelman dokumentit (*m* kpl), kokoelman termit (*n* kpl) ja termien painoarvot.

Jokainen kokoelman dokumentti voidaan esittää piirrevektorina, joka kuvaa kaikkien siinä esiintyvien termiavaruuteen kuuluvien piirteiden frekvenssit. Tekstidokumentin *d* kuvaus *n* uloitteisessa termiavaruudessa voidaan näin esittää kaavalla:

Kaavassa *tf* on termin frekvenssi kyseisessä dokumentissa [Steinbachin et

$$d_{tf} = tf_1, tf_2, ..., tf_n.$$

al., 2000].

Termien painoarvot vektorimallissa voidaan yksinkertaisesti esittää niiden esiintymistiheytenä dokumentissa. Tämän takana on olettamus, että mitä useammin jokin termi dokumentissa esiintyy sitä paremmin se kuvaa dokumentin sisältöä [Manning and Schütze, 1999]. Pelkän termin esiintymistiheysmitan käytön heikkous on kuitenkin se, että se suosii pitkiä, unsastermisiä dokumentteja. Tämän vuoksi termien painojen laskeminen tehdäänkin usein käyttämällä kolmea elementtiä: termin frekvenssi dokumentissa, termin frekvenssi dokumenttikokoelmassa ja dokumenttien normalisointi, jonka tavoitteena on vähentää dokumenttien pituudesta johtuvia arvojen vääristymiä. Termien painottamisen tavoitteena on vähentää usein esiintyvien mutta sisällöllisesti merkityksettömien sanojen painoa dokumenttien kuvaajina. [Zhao and Karypis, 2002; Manning and Schütze, 1999; MOLE, 1999]

Termien esiintymistiheyteen perustuva vektoriavaruusmalli on yksinkertaisuutensa vuoksi suosittu dokumenttien kuvailutekniikka. Siihen kuitenkin liittyy omat rajoitteensa. Sitä on kritisoitu mm. sen vuoksi, että siinä sanojen kontekstia ei oteta huomioon. Tämä ongelma konkretisoituu esimerkiksi moniselitteisten sanojen kohdalla [Hotho et al., 2003]. Sanalla voi olla useita eri merkityksiä kuten esimerkiksi suomen kielen sanalla *lasku*, joka

kontekstista riippuen voi viitata laskutehtävään, mäenlaskuun tai maksettavaan laskuun.

4. Sisällöllisen samanlaisuuden mittaaminen

Edellisessä kappaleessa tarkasteltiin vektoriavaruusmallin käyttöä tekstidokumenttien sisällön kuvaamiseen. Tässä luvussa tarkastelen, miten dokumenttien välistä samanlaisuutta voidaan mitata laskennallisin menetelmin vektoriavaruusmallia käytettäessä.

Eräs usein käytetty samanlaisuuden mitta on kosinimitta [Steinbach et al., 2000]. Kosinimitalla mitataan kahden vektorin välisen kulman kosini vektoriavaruudessa. Voidaan sanoa, että mitä pienempi vektoreiden välinen kulma on, sitä suurempi on tekstidokumenttien sisällöllinen samanlaisuus [Manning and Schütze, 1999]. Laskentakaava on hieman erilainen normalisoiduille ja normalisoimattomille vektoreille. Normalisoimattomien vektoreiden kosinimitta lasketaan kaavalla

$$\cos ine(d_i, d_j) = (d_i \bullet d_j) / \|d_i\| \|d_j\|.$$

Kosinimitta saadaan, kun dokumenttien pistetulo jaetaan vektoreiden pituuksilla [Steinbach et al., 2000]. Jos vektorit ovat normalisoituja, kosinimitta saadaan dokumenttien vektoreiden pistetulona [Manning and Schütze, 1999]. Jos normalisoitujen vektoreiden kosinimitta on nolla, niillä ei ole lainkaan sisällöllistä samanlaisuutta [Zhao and Karypis, 2002].

Sisällöllistä samanlaisuutta mittamaan voidaan käyttää myös euklidista etäisyyttä. Siinä mitataan kahden vektorin välistä etäisyyttä toisistaan vektoriavaruudessa:

$$dis(d_i d_j) = \sqrt{(d_i - d_j)^t (d_i - d_j)} = \|d_i - d_j\|.$$

Jos etäisyydeksi saadaan nolla, vektorit ovat identtiset [Zhao and Karypis, 2002].

5. Klusterointitekniikan valinta

Klusterointialgoritmeja on monia ja tässä luvussa esitellään klusterointialgoritmien kaksi pääryhmää, hierarkkiset ja osittavat algoritmit sekä kummastakin ryhmästä yhden algoritmityypin pääperiaatteet.

5.1. Klusterointitekniikoiden ryhmiä

Klusterointialgoritmit voidaan karkeasti jakaa *hierarkkisiin* (hierarchical) ja *litteisiin* (flat) eli *osittaviin* (partitional). Näiden kahden ryhmän välinen ero on siinä, että hierarkkiset algoritmit jakavat havainnot klustereihin, joilla on

selkeä sisäkkäinen rakenne. Litteässä klusteroinnissa taas lopputuloksena on tietty määrä klustereita, mutta niiden välisiä suhteita toisiinsa ei ole määritelty. [Manning and Schütze, 1999]

Edellä esitetyn karkean jaon lisäksi algoritmeja voidaan kahden pääluokan sisällä jaotella vielä alaluokkiin, mutta en tässä työssä esittele tätä hienojakoisempaa jaottelua. Asiasta kiinnostunut lukija voi kuitenkin tutustua hienojakoisempaan luokitteluun esimerkiksi Jainin ja muiden [1999] kirjoittaman artikkelin avulla. Hierarkkisen ja osittavan luokittelun lisäksi klusterointia käsittelevissä julkaisuissa esitellään myös erikseen eksaktit (hard) ja sumeat (fuzzy) klusterointialgoritmit, joiden ero on se, että eksaktissa klusteroinnissa havainnot voivat kuulua vain yhteen klusteriin kun sumeassa klusteroinnissa taas havainnot voivat osittain kuulua moneenkin klusteriin [Jain et al., 1999].

5.2. Hierarkkiset klusterointitekniikat

Hierarkkiset algoritmit suorittavat klusteroinnin joko *kasaavasti* (agglomerative) tai *pilkkovasti* (divisive). Kasaava algoritmi lähtee liikkeelle tilanteesta, jossa jokainen havainto vastaa yhtä klusteria. Algoritmin edetessä klustereiden lukumäärä pienenee, kunnes kaikki havainnot kuuluvat yhteen klusteriin. Pilkkovat algoritmit toimivat päinvastoin. Aluksi kaikki havainnot kuuluvat yhteen klusteriin ja algoritmin edetessä klustereita jaetaan kahtia, kunnes jokaisessa klusterissa on tasan yksi havainto.

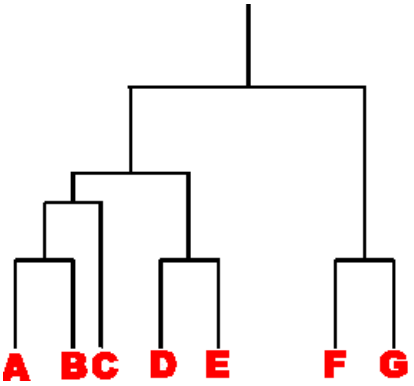
Hierarkkista kasaavaa klusterointia käytetään paljon tekstidokumenttien klusteroinnissa. Alkutilanteessa jokainen havainto eli yksittäinen tekstidokumentti muodostaa oman klusterin. Algoritmin periaate on seuraava [Steinbach et al., 2000]:

1. Lasketaan kaikkien klusteriparien välinen samanlaisuus esimerkiksi toteuttamalla samanlaisuusmatriisi, jonka solu *ij* kertoo klusterin *i* ja klusterin *j* välisen samanlaisuuden
2. Yhdistetään kaksi toisiaan lähinnä olevaa klusteria
3. Päivitetään samanlaisuusmatriisi heijastamaan pareittaista samanlaisuutta uuden klusterin ja muiden alkuperäisten klustereiden välillä
4. Toistetaan kohtia 2 ja 3 kunnes klustereita on vain yksi.

Hierarkkisessa klusteroinnissa algoritmeja jaotellaan myös sen mukaan, miten klusteroinnissa lasketaan samanlaisuus, jonka perusteella sitten päätetään, mitkä klusterit yhdistetään. Siihen käytetään yleensä yhtä seuraavista kolmesta menetelmästä: mitataan kahden toisiaan eniten muistuttavan kluste-

rin samanlaisuuden arvo (single-link clustering), mitataan kahden toisiaan vähiten muistuttavan klusterin samanlaisuuden arvo (complete-link clustering) tai kolmantena mitataan keskimääräinen samankaltaisuus klustereiden välillä (group-average) [Manning and Schütze, 1999].

Hierarkkisen klusteroinnin tulokset voidaan esittää nk. dendrogrammina, jossa samaan klusteriin kuuluvat havainnot on kytketty tietyllä korkeudella olevalla viivalla. Kuvassa 3 on dendrogrammi, jossa alkutilanteessa on seitsemän havaintoa, jotka kukin muodostavat oman klusterin. Kuvasta nähdään, että klusterien A ja B välinen samanlaisuus on suuri, samoin klusterien D ja E sekä klusterien F ja G, joten ne yhdistetään ensimmäiseksi omiksi klustereikseen. Tämän jälkeen klusteriin AB yhdistetään C ja sitten niiden muodostama klusteri yhdistetään klusteriin DE. Viimeksi klusteri ABCDE yhdistetään klusterin FG kanssa.



Kuva 3. Dendrogrammi seitsemän havainnon kokoelmasta

5.3. Osittavat klusterointitekniikat

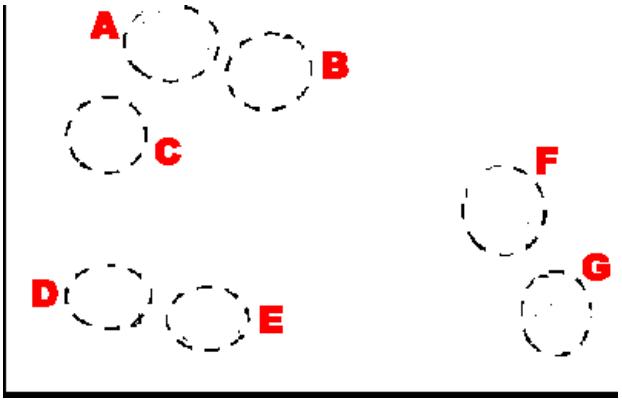
Useat osittavista algoritmeista iteroivat klusterointia, jota jatketaan kunnes siihen käytetyn mitan arvo, esimerkiksi klusterin hyvyys tai laatu, ei enää parane. Kun paranemista ei enää tapahdu tai jos arvo alkaa laskea, iterointi lopetetaan. [Manning and Schütze, 1999]

KM-klusterointi (K-means) on tunnetuimpia osittavia klusterointitekniikoita, jota käytetään laajasti tekstiä sisältävien dokumenttien klusteroinnissa. Se on eksakti klusterointi algoritmi. KM-algoritmejakin on monia, mutta perusidea niissä on se, että klusterit määritellään siihen kuuluvien havaintojen massakeskipisteen (centroid) avulla. Tämä massakeskipiste ei useinkaan vastaa yhtään todellista havaintoa tutkittavassa aineistossa [Manning and

Schütze, 1999]. KM-algoritmin perusidean kuvaavat Steinbach et al. [2000] seuraavasti:

1. Valitaan K-lukumäärä alustavia massakeskipisteitä eli klustereista.
2. Sijoitetaan kaikki havainnot siihen klusteriin, jonka etäisyys havainnosta on pienin.
3. Lasketaan uudestaan klusterin massakeskipiste.
4. Toistetaan kohtia 2 ja 3, kunnes massakeskipisteet eivät enää muutu.

Kohdassa 1 klustereiden lukumäärän valinta on vaikea. Se päättämisessä voidaan hyödyntää aikaisempaa tietämystä kokoelman rakenteesta, mikäli sitä on saatavilla. Näin ei useinkaan ole, joten voidaan myös kokeilla eri lukumäärillä ja verrata tulosten hyvyyttä erilaisia laskennallisia menetelmiä käyttäen [Manning and Schütze, 1999; Zhao and Karypis, 2002]. Alustavien massakeskipisteiden valinta voidaan useimmiten tehdä täysin satunnaisesti. Kohdassa kaksi etäisyys voidaan mitata esimerkiksi euklidista etäisyyttä käyttäen tai mittaus voidaan tehdä kosinimitalla, jolloin lasketaan samanlaisuutta. Kuva 4 havainnollistaa osittavaa klusterointia. Kuva on samasta kuvitteellisesta kokoelmasta kuin kuvassa 3 yllä ja siitä näkyy klustereiden välinen samanlaisuus kuten kuvan 3 dendrogrammissakin.



Kuva 4. Osittavan algoritmin muodostamat klusterit

KM-algoritmin toteutuksen ongelmakohta on se, että siinä ei oteta kantaa, miten käsitellään havainto, jonka etäisyys on sama useamman kuin yhden klusterin keskipisteestä eli se voisi kuulua moneen klusteriin [Manning and Schütze, 1999].

KM-klusteroinnin lisäksi osittavia algoritmeja ovat k-medoidalgoritmi ja EM-algoritmi. K-medoidalgoritmin ero KM-algoritmiin verrattuna on se, että massakeskipisteiden sijasta klusteria edustaa joku kokoelmaan kuuluva

havainto ja EM-algoritmi taas eroaa siinä, että se lasketaan sumeisiin algoritmeihin kuuluvaksi, jolloin havainto voi kuulua useampaankin kuin yhteen klusteriin. [Manning and Schütze, 1999; Zhao and Karypis, 2002]

Sekä osittavia että hierarkkisia algoritmeja käytetään tekstidokumenttien klusteroinnissa. Näistä osittava on joidenkin tutkimusten mukaan osoittautunut laadukkaammaksi varsinkin, kun klusterointi on kohdistettu suuriin tekstidokumenttiaineistoihin [Steinbachin et al., 2000].

6. Yhteenveto

Tekstidokumenttikokoelmien klusterointia on tutkittu paljon ja kattavan kuvan saaminen alueella tehdystä tutkimuksesta on haastava tehtävä. Haastavuutta lisää myös se, että klusterointia sovelletaan monilla tieteenaloilla. Kiinnostus tekstidokumenttien klusterointiin on lisääntynyt viime vuosikymmenen aikana. Siihen ovat vaikuttaneet useat seikat, mutta tärkeimpiä linee sähköisessä muodossa olevan tekstimuotoisen tietomäärän valtava kasvu ja toisaalta tietokoneiden laskentatehon ja tallennuskapasiteetin samaan aikaan tapahtunut nopea lisääntyminen.

Tekstidokumenttikokoelman klusterointiongelman ratkaisu on monivaiheinen ja siinä hyödynnetään eri tieteenalojen tekniikoita. Tekstidokumenttikokoelman klusteroinnin toteuttamista suunnitteleva kohtaakin monia valintatilanteita, sillä klusterointiongelman ratkaisun vaiheet voidaan toteuttaa useita vaihtoehtoisia tekniikoita käyttäen. Tässä tutkielmassa esiteltiin yleisimmin käytettyjä menetelmiä neljän klusterointivaiheen toteuttamisen yhteydessä, nimittäin tekstidokumenttien esikäsittelyn, niiden kuvaamisen, sisällöllisen samanlaisuuden mittaamisen ja klusterointitekniikan valinnan yhteydessä.

Viiteluettelo

[Beil et al, 2002] F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, 436 – 442.

[Carlberger et al., 2001] Johan Carlberger, Hercules Dalianis, Martin Hassel, Ola Knutsson, Improving precision in information retrieval for swedish using stemming, In *Proceedings of NODALIDA '01 - 13th Nordic Conference on Computational Linguistics.* Uppsala, Sweden, 2001. Available as *http://citeseer.nj.nec.com/carlberger01improving.html*

[Fasulo, 1999] Daniel Fasulo, An analysis of recent work on clustering algorithms, Technical Report 01-03-02, Department of Computer Science

and Engineering, University of Washington, Seattle, WA 98195, April 1999.

[Hotho et al., 2003] Hotho Andreas, Staab Steffen, Stumme Gerd, Text clustering based on background knowledge, University of Karlsruhe, Germany, Technical Report No. 425, 2003. Available as *http:// citeseer.nj.nec.com/585487.html.*

[Jain et al., 1999] A.K. Jain, M.N. Murty and P.J. Flynn, Data clustering: a review, *ACM Computing Surveys* **31**, 3, (1999), 264-323.

[Maarek et al., 2000] Maarek Yoëlle S., Fagin Ronald, Ben-Shaul Israel Z., Pelleg Dan, Ephemeral Document Clustering for Web Applications, IBM research report RJ 10186, 2000. Available as *http://citeseer.nj.nec.com/ maarek00ephemeral.html*

[Manning and Schutze, 1999] Christopher D. Manning and Hinrich Schütze, *Foundations of Statistical Natural Language Processing*, MIT Press, 1999.

[MOLE, 1999] MOLE-Text Analysis Group, THOR Center for Neuro-informatics, Section for Digital Signal Processing, IMM, DTU, 1999. Available as *http://isp.imm.dtu.dk/thor/projects/multimedia/textmining/ node5.html.*

[Neto et al, 2000] Neto, J. L., Santos, A. D., Kaestner, C. A. A., and Freitas, A. A., Document clustering and text summarization. In *Proceedings, 4th Int. Conference on Practical Applications of Knowledge Discovery and Data Mining*, 2000, 41-55. London: The Practical Application Company.

[Rosell, 2002] Magnus Rosell, Klustring av svenska tidningsartiklar, 2002, examensarbete i Kungliga Tekniska högskolan, Institutionen för numerisk analys och datalogi, 2002 Tillgänglig på *http://www.nada.kth.se/ ~rosell/publications/mastersThesis/rapport.pdf*

[Steinbach et al., 2000] Michael Steinbach, George Karypis, Vipin Kumar, A comparison of document clustering techniques, In *KDD Workshop on Text Mining*, 2000. Available as *http://citeseer.nj.nec.com/ steinbach00comparison. html*

[Tan, 1999] Tan, A.H., Text mining: The state of the art and the challenges. In *Proceedings of the PAKDD'99 workshop on Knowledge Disocovery from Advanced Databases*, Beijing, 1999, 65-70

[Zhao and Karypis, 2002] Ying Zhao and George Karypis, Criterion functions for document clustering, experiments and analysis, University of Minnesota, Department of Computer Science / Army HPC Research Center, Minneapolis, MN 55455, Technical Report #01-40, February 21, 2002.

Semanttiset yhteydet tietokannoissa

Janne Jämsen

Tiivistelmä

Tutkielmassa käsitellään keskeisiä tietokantaparadigmoja niihin liittyvine kyselykielineen ja luodaan katsaus seuraavan sukupolven tietojärjestelmille (Next Generation Information Systems - NGIS) ehdotetuista tyypillisistä piirteistä. Erityisesti esitellään semanttisen yhteyden käsite ja selvitetään, miten semanttisia yhteyksiä tukevat kyselykielet mahdollistaisivat uudenlaisten kyselyiden muodostamisen tietokannoissa.

Avainsanat ja -sanonnat: semanttinen yhteys, kyselykieli, tietokanta, tietojärjestelmä, tietomalli, tietokantaparadigma, seuraavan sukupolven tietojärjestelmät, NGIS.
CR-luokat: H.1.0

1. Johdanto

Relaatiomallin vajavaisuudet ovat synnyttäneet tarpeen kehittää uudenlaisia relaatiomallia rikkaampia ja ilmaisuvoimaisempia tietokantaparadigmoja. Osa näistä uusista tietokantaparadigmoista on jo aktiivisessa käytössä, kun taas osan merkitys on ainakin toistaiseksi ollut ennen kaikkea teoreettinen. Tässä tutkielmassa esitellään ensin keskeisimpiä näistä tietokantaparadigmoista. Tarkastelunäkokulmana on erityisesti kyseisiin paradigmoihin liittyvät kyselykielet: niiden ilmaisuvoima ja käytettävyys tavallisen loppukäyttäjän näkökulmasta. Toisen luvun loppuosiossa kurkistetaan myös seuraavan sukupolven tietojärjestelmille (Next Generation Information Systems - NGIS) ehdotettuihin uusiin piirteisiin. Aihealueeseen liittyvää tutkimustyötä on tehty runsaasti Tampereen yliopistossa ja valitsemmekin tämän luonnolliseksi tarkastelunäkökulmaksi aiheeseen.

Kesällä 2003 sain toteutettavakseni prototyypin semanttisten yhteyksien selvittämistä tukevasta kyselykielestä. Tutkielman kolmannessa luvussa esittelen tähän liittyen semanttisen yhteyden käsitettä ja sen soveltamismahdollisuuksia kyselykielissä. Erityisesti yritän yhdistää käsitettä aiempaan tutkimuskontekstiin, tarkastella niitä ilmaisuvoimallisia vaatimuksia, joita semanttisten yhteyksien muodostamista tukevilta kyselykieliltä edellytetään sekä pohtia niitä uusia mahdollisuuksia, joita näin muodostetut kyselykielet

tarjoavat käyttäjälle. Myös aiheeseen mahdollisesti liittyvät ongelmat tulevat tarkastelun kohteeksi. On huomattava, että tuki semanttisten yhteyksien muodostamiselle onkin eräs potentiaalisesti käyttökelpoinen piirre tulevissa NGIS-järjestelmissä.

2. Tietokantaparadigmat ja niihin pohjautuvat kyselykielet

Tietomalli on abstrakti kuvaus tietokannan sisältämästä tiedosta. Sen avulla käyttäjä voi tutkia ja muokata tietokannan tietosisältöä ilman että hänen täytyy tuntea sitä fyysistä tapaa, jolla tieto on organisoitu tietovälineillä. Tietomalliin kuuluvat paitsi tiedon kuvauksessa käytetyt loogiset tietoyksiköt ja niiden väliset suhteet myös ne operaatiot, joilla tietoa voidaan käsitellä. Jokainen tietokannan hallintajärjestelmä tarjoaa käyttäjälle yhden tai useamman tietomallin. [Ullman, 1988]

Käyttäjä kommunikoi tietokannan hallintajärjestelmän kanssa jonkin kysely-, tiedonmäärittely- tai tiedonmanipulointikielen välityksellä [Ullman, 1988]. Kyselykieli on erityisesti tietokannan tai tiedostojen sisältämän tiedon hakemiseen ja tarkasteluun tarkoitettu korkean tason tietokonekieli [Samet, 1981]. Tiedonmäärittelykielen avulla voidaan puolestaan luoda tietokantaan uusia skeemoja, joihin tallennettavia tietoja voidaan päivittää (lisätä, muokata, poistaa) tiedonmanipulointikielen avulla. Usein kyselykieleen yhdistyy myös tällaisia skeeman määrittely- ja päivitysominaisuuksia, minkä vuoksi rajanveto kyselykielten ja tiedonmäärittely- tai tiedonmanipulointikielten välillä voi olla käytännössä hankalaa [Samet, 1981; Date, 1989]. Tämän tutkielman puitteissa rajaudumme kuitenkin tarkastelemaan puhtaasti kyselykielille ominaisia piirteitä.

Tavallisesti kyselykieli toteuttaa saman tietomallin kuin itse tietokannan hallintajärjestelmä. Tietyissä tapauksissa kyselykieli voi kuitenkin perustua myös johonkin muuhun tietomalliin. Esimerkiksi funktionaalisia kyselykieliä käytetään tavallisimmin relaatio- tai olio-orientoituneisiin tietomalleihin perustuvien tietokantojen yhteydessä. Samassa kyselykielessä voi yhdistyä myös useita erilaisia tietomalleja. [Paton et al., 1996]

Selvyyden vuoksi erotamme tutkielmassa toisistaan tietokantaparadigman ja tietomallin. Tietokantaparadigmalla viittaamme tietomallia laajempaan käsitekokonaisuuteen. Esimerkiksi lukuisat erilaiset olio-orientoituneet tietomallit katsomme kuuluvan kaikki olio-orientoituneeseen tietokantaparadigmaan. Tietokantaparadigma nähdään tässä siis abstraktina piirrejoukkona, joka kuvailee paradigmaa edustaville tietomalleille tyypillisiä ominaisuuksia. Näin tulkittuna se muistuttaa läheisesti ohjelmointiparadigman käsitettä.

Yksittäiset tietomallit perustuvat käytännön toteutuksiin, minkä vuoksi samaan paradigmaan kuuluvien tietomallien välillä voi olla huomattaviakin keskinäisiä eroja.

Tässä luvussa käsiteltäviä tietokantaparadigmoja ovat relationaalinen, funktionaalinen, olio-orientoitunut, deduktiivinen, ja deduktiivinen olio-orientoitunut tietokantaparadigma. Lisäksi esitellään ehdotuksia seuraavan sukupolven tietojärjestelmille (NGIS) [Niemi et al., 2002b] tyypillisistä piirteistä. Tietokantaparadigmoja tarkastellaan niihin perustuvien kyselykielten näkökulmasta. Erityisesti kiinnitetään huomiota kielten ilmaisuvoimaan ja niiden helppokäyttöisyyteen käyttäjän näkökulmasta.

2.1. Relaatiomalli

Relaatiomalli on tutkielman kirjoittamisen hetkellä laajimmin käytössä oleva yksittäinen tietokantaparadigma ja esitettävistä tietokantaparadigmoista vanhin. Relaatiomallin suosiota selittävät osaltaan mallin yksinkertaisuus, korkea abstraktiotaso, vahva teoreettinen pohja ja siihen pohjautuvien kyselykielten deklaratiivisuus [Ullman, 1988; Codd, 1970].

Relaatiomallin yksinkertaisuus ja abstraktisuus perustuvat siihen, että mallin ainoita rakenteellisia konstruktioita ovat relaatiot, jotka kuvaavat puhtaasti tiedon loogisia ominaisuuksia. Ne eivät siis ole riippuvaisia itse tiedon fyysisestä tallennusmuodosta. Varhaisiin tietokantaparadigmoihin (esim. verkkomalli, hierarkkinen malli) perustuvat tietomallit edellyttivät tavallisesti, että käyttäjä tuntee yksityiskohtia tavasta, jolla tieto on fyysisesti organisoitu. [Codd, 1970] Tästä johtuen kyselyn formulointi relaatiomalliin perustuvalla kyselykielellä on peruskäyttäjän kannalta huomattavasti yksinkertaisempaa.

Relaatiomallin voidaan katsoa teoreettisesti perustuvat joukko-opilliseen relaation käsitteeseen. Matemaattisessa mielessä n-paikkainen relaatio R on siihen osallistuvien arvojoukkojen D_1, D_2, …, D_n muodostaman karteesisen tulon $D_1 \times D_2 \times \ldots \times D_n$ osajoukko. Yksittäistä relaatioon kuuluvaa arvojen järjestettyä joukkoa $(d_1, d_2, \ldots, d_n) \in R$ kutsutaan monikoksi. Käytännössä relaatio voidaan ymmärtää tauluksi, jonka rivit muodostuvat yksikkötapauksia kuvaavista monikoista ja sarakkeet vastaavat nimettyjä attribuutteja. [Ullman, 1988]

Kyselyt relaatiomallissa voidaan formalisoida E. Coddin vuonna 1970 julkaiseman relaatioalgebran merkinnöin. Vaihtoehtoisesti voidaan käyttää matemaattisen logiikan ilmaisutapaan perustuvaa relaatiokalkyylia. Relaatioalgebra sisältää operationaaliset vastineet relationaalisissa kyselykielissä

yleisesti esiintyville toiminnoille unioni, leikkaus, erotus, liitos, projektio ja valinta. Näiden lisäksi se tukee karteesista tuloa. [Zaniolo et al., 1997] Nykyaikaisten relationaalisten kyselykielten (kts. esim. SQL [Date, 1989]) ilmaisuvoima ylittää osin relaatioalgebran ilmaisuvoiman, sillä ne sisältävät relaatioalgebran operaatioiden lisäksi erilaisia aggregointifunktioita, joilla ei ole relaatioalgebrassa vastinetta [Järvelin and Niemi, 1999].

Relaatiomalliin pohjautuvat kyselykielet ovat luonteeltaan deklaratiivisia. Kyselykielen deklaratiivisuus merkitsee, että sillä tehtävä kysely on spesifikaatio kyselyn tuloksena saatavasta tiedosta, ei tavasta, jolla tämä tieto tuotetaan (vrt. algoritmi). Samalla vastuu kyselyn optimoinnista on siirretty käyttäjältä kyselykielen toteuttavalle ohjelmistolle. Kieliä, jotka eivät ole deklaratiivisia, kutsutaan proseduraalisiksi [Ullman, 1988].

Relaatiomallin ilmeisistä eduista huolimatta siihen on kohdistettu myös kritiikkiä. Keskeisimpiä ongelmia ovat relaatiomallin kykenemättömyys ilmaista kompleksisia rakenteita ja rekursiivisia suhteita sekä sen heikko tuki tiedon johdettavuudelle eli deduktiolle. Käsitteellisellä tasolla relaatiomallin formalismia on pidetty epäintuitiivisena verrattuna esimerkiksi ER-mallin esitysmuotoon, jota käytetään yleisesti relaatiotietokantojen skeemojen suunnittelussa. Myös ohjelmointityön näkökulmasta relaatiomalli on ongelmallinen. Ohjelmallisia kyselyitä ja päivityksiä ei voida toteuttaa suoraan yleisesti käytössä olevien olio-orientoituneiden ohjelmointikielten omin konstruktioin, vaan on suoritettava konversio kahden erilaisen tietomallin välillä. [Paton et al., 1996]

2.2. Funktionaalinen tietokantaparadigma

Funktionaalisia kieliä voidaan pitää proseduraalisten ja deklaratiivisten kielten välimuotona [Hillebrand and Kanellakis, 1994]. Niiden tärkein erottava piirre perinteisiin proseduraalisiin kieliin verrattuna on arvojen muuttumattomuuden periaate, joka antaa kielen ilmauksille yksinkertaisen semanttisen tulkinnan ja parantaa kielen johdettavuusominaisuuksia. Periaatteen nojalla proseduraalisissa kielissä tyypillisesti esiintyvän sijoitusoperaattorin käyttö kielletään. Muuttujan arvo ei voi muuttua kesken laskennan, vaan se on staattinen suhteessa käyttökontekstiinsa. Arvojen staattisuuden ansiosta funktiokutsujen keskinäinen suoritusjärjestys ei ole puhtaan proseduraalisten kielten tapaan sidottu, mitä seikkaa voidaan käyttää hyväksi kyselynoptimoinnissa. [Paton et al., 1996]

Funktionaalisissa kielissä ilmaukset samaistetaan niiden arvoihin, minkä vaikutuksesta kaksi saman arvon omaavaa ilmausta voidaan korvata keske-

nään. Periaatteen johdosta puhtaan funktionaalisessa kielessä ei voi esiintyä niin sanottuja sivuvaikutuksia, joiksi tulkitaan funktiokutsut, jotka palauttamansa arvon lisäksi suorittavat jonkin arvosemantiikan ulkopuolisen operaation. Pelkistetyssä funktionaalisessa kielessä ei literaaliarvojen ja muuttujien lisäksi ole muita konstruktioita kuin funktiot. Funktio voi saada mielivaltaisen määrän parametreja, joiden perusteella se palauttaa yhden arvon. Sekä palautettava arvo että parametrit voivat olla literaaliarvoja, muuttujia tai funktioita. Tietoalkioiden muodostamia kokoelmia voidaan käsitellä erityisten rakenninfunktioiden avulla. Tyypitetyissä funktionaalisissa kielissä on lisäksi mahdollisuus määritellä tietotyyppejä, jotka edustavat joko literaalityyppejä tai niistä johdettuja kompleksisia tyyppejä. [Paton et al., 1996; Knuutila, 2001]

Funktionaaliset kielet pohjautuvat lambda-kalkyyliin, jota voidaan pitää yksinkertaisimpana tapana määritellä matemaattinen funktion käsite. Lambda-kalkyylin esitti Alonzo Church 1930-luvulla. [Knuutila, 2001] Kalkyylista on olemassa myös tyyppimäärittelyt sisältävä versio, jota voidaan käyttää tyypitettyjen funktionaalisten kielten formalisointiin. [Hillebrand and Kanellakis, 1994] Lambda-kalkyylin ja samalla funktionaalisten kielten ilmaisuvoima on sama kuin Turingin koneen [Rosser, 1982].

Funktionaalisia kieliä voidaan käyttää erilaisiin tietokantaparadigmoihin perustuvien tietokantojen kyselykielinä [Paton et al., 1996]. Tässä tarkastelemme funktionaalisten kyselykielten yleisiä ominaisuuksia vertailemalla lyhyesti kolmea funktionaalista kyselykieltä: DAPLEX:ia, FQUERY:a ja FQL:ää. Näistä DAPLEX toimii aidosti funktionaalisen tietokannan kyselykielenä tarjoten lisäksi funktionaalisen tietokannan päivitykseen ja sen tietomallin ylläpitoon liittyviä toimintoja [Shipman, 1981]. FQUERY ja FQL puolestaan ovat puhtaita kyselykieliä, joista edellinen on tarkoitettu relaatiotietokantojen kyselykieleksi ja jälkimmäinen on periaatteessa riippumaton tietokantaparadigmasta. Käytettävältä tietokannalta edellytetään, että sille on määritelty rajapinta, jonka avulla tieto voidaan kuvata FQL:n ymmärtämään yksinkertaiseen funktionaaliseen muotoon [Buneman and Frankel, 1979; Warner and Odle, 1981].

Tiedon funktionaaliselle kuvaukselle on DAPLEX:in ja FQL:n tapauksissa ominaista, että tieto esitetään funktioiden, entiteettien ja literaaliarvojen avulla. Entiteettien ominaisuudet kuvataan funktioina, jotka saavat parametrikseen kohde-entiteetin ja palauttavat ominaisuuden arvoa vastaavan literaaliarvon. Vastaavasti entiteettien keskinäiset suhteet ilmaistaan niiden välisinä funktioina. DAPLEX:issa myös entiteettityypit kuvataan funktioina; ne ovat

funktioita, jotka eivät saa parametrejä, mutta palauttavat tyyppiin sisältyvien entiteettien joukon. [Shipman, 1981; Buneman and Frankel, 1979]

Funktionaalisen tiedon esittämisen näkyvin erityispiirre relaatiomalliin nähden se, että funktioiden käsittely tapahtuu lähtökohtaisesti vain yhdessä suunnassa. Haluttaessa käsitellä suhteita jossain muussa järjestyksessä täytyy tätä varten määritellä uusi johdettu funktio. Binaaristen suhteiden kohdalla tämä tapahtuu luontevasti käyttäen kielten tarjoamia valmiita primitiivejä, joilla voidaan johtaa funktion käänteisfunktio. Korkeampaa astelukua edustavien ja mahdollisesti omia attribuutteja sisältävien suhteiden kuvaaminen ja käsittely on sen sijaan ongelmallisempaa. Ilmeinen ratkaisu on kuvata tällaiset suhteet binaarisina funktioina, jotka liittyvät korkeampaa astelukua olevaa suhdetta vastaavaan entiteettiin.

FQUERY käsittelee relaatioihin tallennettua tietoa ja sisältää siten relaatioalgebralle tyypilliset liitos- ja valintaoperaatiot jo kielen primitiivien tasolla [Warner and Odle, 1981]. Myös muilta osin kielen syntaksi muistuttaa SQL:n rakennetta. Huomattava erottava piirre relaationaalisiin kyselykieliin nähden on se, että FQL:n ja DAPLEX:in tapaan kyselykieli sisältää primitiivit, joilla esitetään iterointi tietoalkioiden muodostamassa joukossa. Iterointia tarvitaan aina suoritettaessa funktiokutsu tai operaatio joukon jokaiselle alkiolle. Menettelyn vahvuutena on, että sen avulla voidaan muodostaa helposti esimerkiksi aggregointifunktiot, joiden esittäminen puhtaan relaatioalgebran keinoin on mahdotonta. Toisaalta se edellyttää käyttäjältä enemmän algoritmista ajattelua kuin puhdas relationaalinen kieli, mikä vähentää kielen deklaratiivisuutta. Vaihtoehtoinen funktionaalisissa kyselykielissä käytetty tapa operoida kokoelman alkioilla on logiikan joukonmuodostus operaatiota vastaava merkintä (engl. comprehension), jonka avulla kokoelmasta voidaan valikoida esitettävät valintakriteerit täyttävät alkiot [Paton et al., 1996]. Menettely on käyttäjän kannalta deklaratiivisempi, mutta toisaalta se ei itsessään edusta funktionaalista ilmaisutapaa.

Kolmesta kielestä vain FQL:ssä suoritettava kysely on itsessään funktiomäärittely. Kysely voi olla funktioiden tapaan parametrisoitu, jolloin samaa kyselyrunkoa voidaan periaatteessa käyttää useaan eri kyselyyn. DAPLEX sen sijaan muistuttaa ohjausrakenteiltaan enemmän imperatiivista kieltä ja FQUERY SQL:n tapaista deklaratiivista kyselykieltä. Mielenkiintoista on, että sekä DAPLEX.in että FQUERY:n suunnittelijat ilmoittavat ratkaisuillaan korostaneensa kielensä helppokäyttöisyyttä ja intuitiivisuutta loppukäyttäjän näkökulmasta [Shipman, 1981; Warner and Odle, 1981]. Myös FQL:n suunnittelijat esittävät varauksen oman kielensä soveltuvuudesta loppukäyttäjälle

[Buneman and Frankel, 1979]. Näyttää siis ilmeiseltä, että puhtaan funktionaalisille kyselykielille ominaisen syntaksin ja ajattelun oppiminen edellyttää käyttäjältä keskimääräistä enemmän ohjelmointimenetelmien tuntemusta, vaikkakin tarvittavan tiedon määrää on vaikeaa verrata esimerkiksi deduktiivisten kyselykielten käytölle asetettaviin vaatimuksiin.

2.3. Olio-orientoituneet tietokannat

Relaatiomalli on tyypillisesti arvo-orientoitunut tietokantaparadigma. Arvo-orientaatiolle on tyypillistä, että tietoyksiköt (relaatiomallin tapauksessa monikot) yksilöidään yksiselitteisesti jonkin niiden attribuuttien osajoukon eli avaimen arvojen perusteella. Relaatiomallin tapauksessa tämä merkitsee, että mikäli relaatiosta tehdään viittaus toiseen relaatioon, on viittaavaan relaatioon otettava kaikki viitattavan relaation avaimeen kuuluvat attribuutit. Tätä attribuuttien joukkoa kutsutaan viittaavan relaation vierasavaimeksi. [Ullman, 1988] Mikäli relaation avainattribuuttien arvoissa tapahtuu muutoksia, on muutokset tehtävä aina myös relaatioihin, joissa vierasavaimin viitataan kyseiseen relaatioon. Tämä viite-eheyden säilyttäminen tietokannan päivitysten yhteydessä on eräs arvo-orientaation huomattavista ongelmista [Bagui, 2003]. Relaatiomallin yhteydessä on lisäksi huomattava, että koska relaatio matemaattisessa mielessä on monikoiden muodostama joukko, se ei voi sisältää samaa monikkoa useampaan kertaan. Kahdella monikolla ei siis relaatiomallissa voi olla identtisiä attribuuttien arvoja. [Paton et al., 1996]

Suppeimmassa merkityksessään olio-orientoituneella tietokannalla tarkoitetaan tietokantaa, joka tukee olioidentiteettiä [Ullman, 1988]. Olioidentiteetti toteutetaan yleensä järjestelmän toimesta lisäämällä oliota vastaavalle tietoyksikölle ylimääräinen tunnisteattribuutti, jota käytetään yksilöimään kohdeolio ja jolla ei tämän tunnistamistehtävän lisäksi ole muuta semanttista merkitystä [Paton et al., 1996]. Käytännön toteutuksissa oliotunnisteena voidaan käyttää esimerkiksi tietoyksikön fyysisen muistipaikan osoitetta. Relaatiomallissa yksinkertaista olio-orientoitunutta mallia voidaan simuloida luomalla relaatiolle keinotekoinen avainattribuutti eli surrogaatti [Ullman, 1988]. Olio-orientoituneissa järjestelmissä attribuutin arvona voi olla paitsi literaaliarvo myös olioidentiteetin omaava tietoyksikkö eli olio. Tällainen olioarvoinen attribuutti [Niemi et al., 2002a] toteutetaan tallentamalla attribuutin arvoksi oliota vastaava oliotunniste. Puhtaissa olio-orientoituneissa järjestelmissä myös literaaliarvot kuvataan olioina [Koskimies, 2000].

On huomattava, että mitään yksiselitteistä määritelmää tai kriteeristöä olio-orientoituneelle tietomallille ei ole hyväksytty [Bertino et al., 1992], mikä selittää osaltaan olemassa olevien olio-orientoituneiden tietokantajärjestelmien suurehkot keskinäiset erot. Kuitenkin vakiintuneen käytännön mukaan olio-orientaatiolla viitataan nykyisin olio-identiteetin ohella myös laajempaan joukkoon piirteitä, jotka ovat peräisin olio-orientoituneista ohjelmointikielistä ja ohjelmointiparadigmoista [Kim, 1990]. Piirteet voidaan jakaa strukturaalisiin ja operationaalisiin (engl. behavioral) (vrt. esim. [Niemi et al., 2002b]). Strukturaalisesti olio-orientoitunut tietomalli voidaan johtaa relaatiomallista antamalla monikoille olio-identiteetti ja määrittelemällä attribuuttien arvoalueeksi jokin järjestelmässä määritelty tietotyyppi eli luokka. Luokka edustaa literaalityyppiä tai se voi olla literaalityypeistä johdettu kompleksinen tietotyyppi. Relaation skeema kuvautuu olio-orientoituneessa järjestelmässä luokaksi ja monikoita vastaavat oliot ovat sen ilmentymiä. Luokat järjestetään hierarkkisesti niin, että hierarkiassa alempana olevan luokan ilmentymät ovat myös hierarkiassa ylempänä olevan luokan ilmentymiä. Määrittelystä seuraa, että hierarkiassa alempana olevan luokan eli aliluokan tulee sisältää kaikki attribuutit, jotka sisältyvät sen yläluokkaan. (vrt. [Kim, 1990]) Tämän lisäksi aliluokka voi (ja hyvässä olio-orientoituneessa sen myös pitää) sisältää omia attribuutteja. Olion tilaksi kutsutaan sen kaikkien attribuuttien arvojen muodostamaa kokonaisuutta tietyllä ajanhetkellä [Koskimies, 2000].

Operationaalisesti luokkiin voidaan sisällyttää attribuuttien ohella ohjelmakoodilla toteutettuja metodeja [Kim, 1990]. Metodeita kutsutaan tavallisesti luokan ilmentymän kautta. Koska myös metodien palautusarvoille on attribuuttien tapaan on määritelty tyyppi, voidaan attribuutit ja metodit rinnastaa toisiinsa. Attribuutit vastaavat metodeita, joiden parametrien määrä on nolla (vrt. esim. [Niemi et al., 2002b]). Samoin kuin attribuuttien kohdalla aliluokka perii kaikki yläluokan metodit, mutta aliluokkamäärityksen yhteydessä on myös mahdollista kuormittaa metodi eli määritellä sille uusi toteutus. Olio-orientaatioon liittyy läheisesti myös niin sanottu tiedon kätkennän periaate eli kapselointi, jonka nojalla luokkaan sisältyvien olioiden attribuutteja ja metodeita voidaan kutsua vain määritellyn rajapinnan kautta. Yksittäinen olio voi rajapinnassa näkyvien attribuuttien ja metodien lisäksi sisältää myös suojattuja attribuutteja tai metodeita. Tämän ansioista luokat ja niitä vastaavat oliot toimivat eräänlaisina ”mustina laatikoina”, joiden palveluita voidaan käyttää ilman, että käyttäjä tuntee niiden sisäistä tilaa tai metodien toteutusta [Ullman, 1988; Koskimies, 2000]. Erityisesti tämä parantaa sovellusten modulaarisuutta ja helpottaa niiden suunnittelua ja päivitystä.

Varhaiset olio-orientoituneet kyselykielet olivat tyypillisesti proseduraalisia. Niiden olio-orientoituneet piirteet rajoittuivat usein olio-identiteetin tukemiseen. Esimerkkejä ovat CODASYL DBTG ja IMS, joista ensin mainittu edusti hierarkkista tietomallia ja jälkimmäinen verkkomallia. [Ullman, 1988] Nykyaikaisten olio-orientoituneiden tietokantojen yleistymisen yhteydessä ongelmana ovat pitkään olleet olio-orientoituneiden tietomallien keskinäiset erot ja relaatioalgebraan verrattavan yhtenäisformalismin puute [Bagui, 2003]. Tästä johtuen olio-orientoituneiden tietokantojen yhteydessä esiin ei ole noussut SQL:n tapaista yhtä laajalti hyväksyttyä kyselykieltä, vaan valmistajasta riippuen tietokantojen kyselyominaisuudet ovat vaihdelleet suuresti. Viime aikoina olio-orientoituneissa tietomalleissa suoritettaville kyselyille on esitetty runsaasti formaaleja malleja, muun muassa olioalgebra ja olio-orientoitunut predikaattikalkyyli [Alhajj and Polat, 2000; Bertino et al., 1992]. Nämä formalismit osoittavat, että kyselyt olio-orientoituneissa tietomalleissa voidaan suorittaa puhtaasti deklaratiivisesti, mutta niiden käytännön sovellettavuus on ollut rajoitettu, sillä ne eivät sinällään tarjoa esimerkiksi relaatiomallille tyypillisiä kyselynoptimointimenetelmiä [Bagui, 2003].

Olio-orientoituneet tietomallit tarjoavat ratkaisun moniin relaatiomallin ongelmiin, mutta eivät silti itsessäänkään ole täysin ongelmattomia. Olio-orientoitunut tietomalli mahdollistaa yksinkertaisen kompleksisten olioiden mallintamisen osa-kokonaisuus –suhteen avulla, jossa osa kuvataan kokonaisuutta esittävän olion olioarvoisena attribuuttina. Suhde voi sisältää mielivaltaisen määrän sisäkkäisyyden eri tasoja. Olioarvoiset attribuutit mahdollistavat myös useista mahdollisesti erityyppisistä olioista koostuvien kokoelmaolioiden muodostamisen. Niitä voidaan käyttää esimerkiksi moniarvoisten attribuuttien mallintamiseen, mitä tarkoitusta varten relaatiomallissa tarvittaisiin aina erillinen relaatio. Kokoelmaoliot voivat edustaa jotain tietorakennetta (pino, jono, keko jne.) tai loogista abstraktiota (joukko, monikko jne.). Olio-orientoituneiden kyselykielten selvä etu on niiden suora tuki vastaavaan paradigmaan perustuville ohjelmointikielille; kyselyjen vastaukset ovat kokoelmaolioita, joita voidaan käsitellä suoraan ohjelmointikielten omin rakentein. [Paton et al., 1996]

Olio-orientoituneessa tietomallissa myös olioiden väliset suhteet kuvataan tyypillisesti osa-kokonaisuus –suhdetta vastaavalla tavalla olioiden olioarvoisina attribuutteina. Tämä tekee suhteiden käsittelystä epäsymmetrisen, ja usein onkin epäselvää kummassa suhteeseen osallistuvasta oliosta suhdetta vastaava attribuutti tulee antaa [Paton et al., 1996]. Kompleksisiin olioihin kohdistuvien kyselyiden kohdalla ongelmana on, että käytössä olevissa olio-

orientoituneissa kyselykielissä käyttäjän täytyy etukäteen tuntea olion intensionaalinen rakenne. Ainoa tapa päästä käsiksi kokonaisuuden osiin on näissä kielissä yleensä niin sanottua polkunotaatio, jossa kokonaisuutta vastaavan muuttujan nimen liittäminen vastaavan attribuutin nimeen erotinoperaattorilla (esimerkiksi piste ilmaisussa kokonaisuus.attribuutti) palauttaa kyseistä attribuuttia vastaavan osaolion, jonka osa-olioon voidaan edelleen viitata liittämällä polkuun uusi erotinoperaattorilla erotettu attribuuttinimi jne. (kts. esim. [Niemi et al., 2002a]) Polkunotaation etuna esimerkiksi relaatiomalliin nähden on, että se vähentää relaatiomalliin nähden tarvittavien eksplisiittisesti annettujen liitosehtojen tarvetta [Bagui, 2003]. Toisaalta se edellyttää polun muodostavan attribuuttiketjun tuntemista, mikä rajoittaa olennaisesti näiden kielten käytettävyyttä kompleksisten olioiden rakenteeseen tehtävien kyselyiden osalta [Niemi et al., 2002a]. Polkunotaatio ei myöskään poista mahdollista tarvetta yhdistää olioita jollain muulla kuin skeemassa oletetulla tavalla. Tämän vuoksi olio-orientoituneessa kyselykielessä tulee olla myös relaatiomallin liitosoperaatiota vastaava operaatio. [Bertino et al., 1992]

Kyselyyn sisältyvien metodikutsujen osalta on huomattava, että yksinkertaisen semantiikan määrittely kyselylle edellyttää, että metodit eivät tee muutoksia olion tilaan, vaan ovat yksinomaan lukuoperaatioita [Bertino et al., 1992]. Olion tilaa muuttavien päivitysmetodien käyttö estää pahimmillaan tiedon deklaratiivisen käsittelyn edellyttäen proseduraalista kyselynmuodostusta.

Tunnetuimpiin käytössä oleviin olio-orientoituneihin kyselykieliin kuuluu O^2-tietokannan kyselykieleksi suunniteltu OQL. OQL noudattaa läheisesti SQL:lle tunnusomaista syntaksia. Kielen suurimpia etuja ovat sen deklaratiivisuus ja SQL:stä tuttu luonnollista kieltä muistuttava rakenne. Kielen ongelmallisia piirteitä ovat sen edellyttämä erilaisten rakentimien ja iteraattoreiden sisäkkäinen käyttö, mikä saattaa peruskäyttäjän kannalta osoittautua liian vaativaksi. Muiden olio-orientoituneiden kielten tapaan OQL edellyttää myös kyselyyn osallistuvien toisiinsa liittyvien olioiden välisen navigointipolun täydellistä tuntemista. [Niemi et al., 2000]

2.4. Deduktiiviset tietokannat

Relaatioalgebran ilmaisuvoima on rajallinen. Jos relaatiomallia täydennetään ensimmäisen kertaluvun predikaattilogiikkaan pohjautuvilla säännöillä, saadaan tietomalli, jossa tärkeimmät relaatiomallin laskennallisista rajoituksista poistuvat [Zaniolo et al., 1997]. Loogisen esitysformalismin etuina ovat

siihen liittyvä vahva malliteoreettinen semantiikka, joka laajentaa relaatiomallin vastaavaa, sekä todistusteoria, jonka avulla eksplisiittisesti annetusta tiedosta voidaan johtaa loogisesti päteviä päätelmiä. Lisäksi on huomattava, että looginen esitysmuoto on luonteeltaan deklaratiivinen, mikä tekee siitä käyttäjän kannalta intuitiivisen. [Paton et al., 1996]

Deduktiiviset tietomallit pohjautuvat tavallisesti logiikkaohjelmointikielille, joista tunnetuimpia ja vanhimpia on Prolog. Datalog vastaa Prologin puhtaaseen logiikkaohjelmointiin sisältyvää osajoukkoa, johon ei kuulu funktiosymboleita [Liu, 1999]. Primitiivisimmässä muodossaan se ei sisällä myöskään tue negaatiota. Deduktiivisten tietokantojen tutkimuksen kannalta Datalog on keskeinen teoreettinen väline, jonka asemaa voidaan verrata relaatioalgebran asemaan relaatiomallissa. [Zaniolo et al., 1997]

Datalogissa tieto mallinnetaan sääntöinä, jotka perustuvat predikaattilogiikan formalismiin. Proseduraalisen tulkinnan helpottamiseksi säännöt esitetään tavallisesti niin sanotussa Hornin muodossa, jossa säännöillä on positiivista loogista atomia vastaava pää, jota implikoi loogisten atomien konjunktiosta muodostuva säännön runko. Säännön rungossa ei esiinny disjunktiota, vaan vaihtoehtoiset tavat suorittaa johdos ilmaistaan erillisinä sääntöinä. Faktat ovat sääntöjen erikoismuoto, joilla ei ole runkoa ja jotka siten tulkitaan aina tosiksi. [Niemi, 2003] Yksinkertaisimmassa relaatiomallin ensimmäistä normaalimuotoa vastaavassa esitystavassa loogiset atomit ovat predikaatti-ilmauksia, jotka koostuvat literaalisymboleiden muodostamista merkkijonoista eli vakioista tai muuttujista, jotka samaistuvat mihin tahansa vakioon. Kielen ilmaisuvoimaa voidaan kasvattaa sallimalla sisäkkäiset predikaatti-ilmaukset eli funktiosymbolit. [Zaniolo et al., 1997]

Predikaattilogiikasta poiketen kvantifiointia ei esitetä eksplisiittisesti käyttämällä kvanttoreita, vaan eksistentiaalimuodossa olevat säännöt ilmaistaan vakioilla, kun taas universaalikvantifioinnin ilmaisemiseen käytetään muuttujia. Muuttujien arvotukset ovat funktionaalisten kielten tapaan staattisia käyttökontekstissaan. Logiikkaohjelmointikielet sisältävät automatisoidun teoreemantodistusmekanismin, jonka avulla niiden sääntökokoelmaan eli tietokantaan voidaan suorittaa kyselyjä. Loogisen deduktion onnistumiseen perustuvien totuusarvokyselyjen ohella voidaan muuttujien avulla esittää myös relationaalisille kielille tyypillisiä eksistentiaalikyselyjä, jotka palauttavat kyselyatomissa esiintyviin muuttujiin samaistettavissa olevat vakiot. [Niemi, 2003]

Logiikkaohjelmointiin perustuvat säännöt ovat luonnollinen tapa laajentaa relationaalista tietokantaa. Relaatiot kuvataan predikaateiksi, joissa ei

esiinny funktioilmauksia. Relaatioita vastaavat faktat yhdessä tietokantaan tallennettujen muiden sääntöjen kanssa muodostavat niin kutsutun suljetun maailman, jossa kyselyitä vastaavat arvotukset ja totuusarvot määräytyvät. Tämän suljetun maailman periaatteen mukaisesti väite on epätosi, mikäli sitä ei voida johtaa tietokantaan tallennetuista säännöistä, muussa tapauksessa tosi. [Paton et al., 1996]

Transitiiviset sulkeumat muodostetaan logiikkaohjelmissa rekursiivisten sääntöjen avulla. Sääntö on rekursiivinen, mikäli sen rungon oikeaksi todistaminen edellyttää säännön päähän samaistuvan atomin oikeaksi todistamista. Peruskäyttäjälle transitiivisten suhteiden laskemiseen soveltuvien kyselyiden muodostaminen Datalogia vastaavalla kielellä on kuitenkin liian vaativa tehtävä [Järvelin and Niemi, 1999]. Rekursion päättävän ehdon muotoilu ja esimerkiksi Prologin yhteydessä välittömän vasemman rekursion välttäminen edellyttävät, että käyttäjä tuntee myös yksityiskohtia tavasta, jolla logiikkaohjelmaa prosessoidaan. Näiden syiden vuoksi useissa yhteyksissä on esitetty erityisen transitiivisen operaattorin sisällyttämistä kyselykieleen [Järvelin and Niemi, 1999].

Kaikki relaatioalgebraan ja relaatiokalkyyliin sisältyvät kyselyt voidaan esittää Datalogilla. On kuitenkin olemassa muun muassa eräitä aggregointifunktioita, joita ei voida esittää puhtaalla Datalogilla. Datalogia voidaankin laajentaa sallimalla aritmeettiset ilmaukset ja negaatio. Erityisesti jos sallitaan funktiosymboleiden käyttö, saadaan kieli, jonka ilmaisuvoima vastaa Turingin koneen ilmaisuvoimaa. [Zaniolo et al., 1997]

2.5. Deduktiiviset ja olio-orientoituneet tietokannat

Puhdas Datalog ja ensimmäistä normaalimuotoa edustava relaatiomalli eivät sinällään sovellu kompleksisten entiteettien kuvailuun. Kuten todettiin Datalogia voidaan kuitenkin laajentaa sallimalla funktiosymboleiden käyttö. Myös ensimmäistä normaalimuotoa olevaa relaatiomallia voidaan yleistää niin sanottuun NF^2-muotoon sallimalla sisäkkäiset relaatiot [Järvelin and Niemi, 1999]. Toinen mahdollisuus, jota tarkastelemme tässä luvussa, on pyrkiä yhdistämään deduktiiviset ja olio-orientoituneet tietokantaparadigmat toisiinsa. Myös tämä lähestymistapa on kerännyt huomiota tieteellisessä keskustelussa ja useita erilaisia tietomalleja kyselykielineen on esitetty. [Niemi et al., 2002b]

Eräs suurimmista ongelmista, joka liittyy deduktiivisen ja olio-orientoituneen (DOOD) paradigman integrointiin, koskee olio- ja arvo-orientoituneiden tietomallien välistä eroa. Puhtaassa olio-orientoituneessa mallissa

myös literaalisymboleita vastaavat arvot, numerot, kirjaimet, merkkijonot, kuvataan olioina, joilla on olio-identiteetti [Koskimies, 2000]. Tämän vuoksi on mahdollista, että kahden samanikäisen henkilön ikäattribuuttien arvoja vastaavat oliot, joiden molempien lukuarvo on 45, ovat itse asiassa eri olioita ja niiden yhtäsuuruusvertailu johtaa negatiiviseen lopputulokseen. Yhden yhtäsuuruusoperaattorin sijasta tarvitaankin siis kaksi yhtäsuuruusoperaattoria, joista toinen viittaa olion olio-identiteettiin ja toinen sen arvoon [Bagui, 2003]. Loogisten sääntöjen muodostuksen kannalta tämä on hankalaa, sillä se vaikeuttaa bgiikkaohjelmointikielissä tyypillisesti käytettävien samaistamisalgoritmien käyttöä ja tekee kyselyistä huomattavasti kompleksisimpia. Vaihtoehtoinen tapa on käsitellä literaalisymboleita uniikkeina olioina, jolloin molempien henkilöiden ikäattribuutit viittaavat samaan olioon. Jos samaa menetelmää sovelletaan myös kompleksisiin olioihin, seurauksena on kuitenkin olio-identiteetin totaalinen menettäminen. Kompleksisen olion yhdessä arvossa tapahtuvat muutokset aiheuttavat sen, että koko olio tulkitaan toiseksi uniikiksi olioksi. Varsinaista teoreettista estettä näiden kahden paradigman yhdistämiselle ei kuitenkaan liene, vaikka myös vastaväitteitä on esitetty [Niemi et al., 2002b; Ullman, 1988].

Vanhin DOOD-kieli on O-Logic. O-Logic:in tuki olio-orientoituneille piirteille on rajallinen. Se ei tue metodeita tai luokkahierarkioita. O-Logic ei myöskään tue predikaatti-ilmauksia, vaan kaikki olioiden väliset suhteet esitetään attribuuttien avulla. Literaalit tulkitaan O-Logic:ssa uniikeiksi olioiksi. F-Logic on O-Logic:in johdos, joka sisältää muun muassa tuen periytymishierarkioille. [Liu, 1999] ROL on O-Logic:ia ja F-Logic:ia uudempi DOOD-tietokantojen kyselykieleksi ehdotettu ja myös implementoitu kieli. ROL:in mielenkiintoinen ominaispiirre on, että se antaa käyttäjälle mahdollisuuden tietoisesti valita, käyttääkö hän tiedon kuvaamiseen arvo- vai olio-orientoitunutta esitystä. Koska ROL sisältää Datalogin aitona osajoukkonaan, se mahdollistaa myös predikaatti-ilmausten käytön. Itse asiassa predikaatti-ilmaukset ovat ROL:ssa literaaliolioiden, oliotunnisteiden ja joukkojen tavoin olioita. ROL:n kehittynyt piirre on, että se mahdollistaa perinteisten ekstensionaalisen informaation lisäksi intensionaaliset eli skeemaan (ROL:in tapauksessa luokkarakenteeseen) kohdistuvat kyselyt. Olio-orientoituneista piirteistä ROL ei tue metodeita. [Liu, 1996]

Loppukäyttäjän kannalta deduktiivisten olio-orientoituneiden kyselykielten käyttö on vaativuustasoltaan verrattavissa deduktiivisiin kyselykieliin joskin käyttäjän on lisäksi tunnettava olio-orientoituneista ohjelmointiparadigmoista peräisin olevia käsitteitä. Näin kehittyvät tietomallit ovat oleelli-

sesti puhtaan deduktiivisia tietomalleja rikkaampia ja valmiiden konstruktioidensa avulla ne helpottavat itse tiedon kuvaamista. Samalla kyselyiden kompleksisuus kuitenkin kasvaa. [Järvelin and Niemi, 1999]

2.6. Seuraavan sukupolven tietojärjestelmät (NGIS)

Tässä luvussa kuvaamme piirteitä, joiden on esitetty olevan tyypillisiä seuraavan sukupolven tietojärjestelmille eli niin kutsutuille NGIS-järjestelmille [Niemi et al., 2002b]. Aiheeseen liittyvää tutkimusta on tehty runsaasti Tampereen yliopistossa ja valitsemmekin tämän kehitystyön luonnolliseksi lähtökohdaksi aihealueen tarkastelulle. On kuitenkin muistettava, että vasta käytäntö tulee näyttämään, mitkä tämän päivän kehitteillä olevista malleista ja teknologioista vastaisuudessa saavat merkittävän aseman.

RDOOM [Niemi et al., 2002b] on Tampereen yliopistossa kehitetty tietomalli, joka yhdistää toisiinsa piirteitä relationaalisista ja deduktiivisista olio-orientoituneista tietokantaparadigmoista. Tietomallin huomattavimmat erot edellä kuvattuun deduktiiviseen olio-orientoituneeseen tietomalliin nähden ovat olioiden välisten suhteiden kuvaaminen relaatioiden avulla sekä tuki deduktiivisille olio- ja assosiaatio- eli suhdetyypeille. Relaatioiden käyttö olioiden välisten suhteiden mallintamisessa on edullista, sillä se mahdollistaa relaatiomallille tyypilliset tehokkaat suhteiden käsittelyoperaatiot ja navigointimenetelmät, poistaa olioarvoisista attribuuteista aiheutuvan kuvauksen asymmetrisyyden sekä parantaa sen intuitiivista tulkintaa. Mielenkiintoista onkin, että näin valittuina RDOOM:in perusprimitiivit muistuttavat läheisesti ER-mallin (kts. esim. [Ullman, 1988]) primitiivejä. ER-mallia käytetään tavallisesti juuri tietokantaskeemojen suunnitteluun ja sitä voidaan siten pitää intuitiivisena tapana kuvata tietokannan intensionaalista rakennetta [Paton et al., 1996]. Erityisesti relaatio- ja olio-orientoituneiden tietomallien yhdistämistä on esitetty myös muissa yhteyksissä (kts. esim. [Premeriani et al., 1990; Cattell and Rogers, 1986]).

RDOOM tukee piirteitä, jotka helpottavat tavallisen loppukäyttäjän toimintaa sallimalla deduktiivisesti määriteltyjen olio- ja suhdetyyppien sekä metodien määrittelyn. Tiedon kätkentä -periaatteen mukaisesti käyttäjän ei tarvitse tuntea tapaa, jolla nämä johdetaan tietokannan eksplisiittisesti annetusta tiedosta. Deduktiivisten olio- ja suhdetyyppien käsittely kyselyissä tapahtuu samaan tapaan kuin varsinaisten olio- ja suhdetyyppien. Niiden avulla voidaan relaatiomallille tyypillisten näkymien ohella tarjota käyttäjälle korkeammin jalostettua, muun muassa transitiivisen sulkeumien avulla johdettavaa tietoa. [Niemi et al., 2002b]

Toinen NGIS-järjestelmien kannalta hyödyllinen piirre on tehokas tuki kompleksisten entiteettien mallintamiseen ja niihin liittyviin kyselyihin. Kuten totesimme, useissa olio-orientoituneissa tietomalleissa osa-kokonaisuussuhde esitetään tavallisesti yksisuuntaisesti olioarvoisten attribuuttien avulla, mikä tekee navigoinnista hankalaa haluttaessa viitata olioarvoisen attribuutin arvona olevasta oliosta viittaavaan olioon. Edelleen käyttäjän täytyy näihin malleihin perustuvissa kyselykielissä yleensä osoittaa polkuilmausten avulla eksakti navigointi kahden sisäkkäisyyden eri tasoilla olevan olion välillä, minkä vuoksi viittaaminen kokonaisuuden mielivaltaisella sisäkkäisyyden tasolla olevaan osaan on vaikeaa ellei mahdotonta. [Niemi et al., 2002a]

Tutkimuksessaan Niemi et al. [2002a] ehdottavat kompleksisten olioiden mallintamiseen NF^2-relaatiomallin ja olio-orientoituneiden tietomallien yhdistämiseen perustuvaa lähestymistapaa. Tällaisessa mallissa osia vastaavat tietorakenteet tallennetaan suoraan kokonaisuutta esittävään tietorakenteeseen. NF^2-mallista poiketen osarakenteilla on kuitenkin olioidentiteetti, mikä mahdollistaa esimerkiksi itsenäisen viittauksen tekemisen niihin kompleksisen olion ulkopuolelta. Suora sisäkkäisyys mahdollistaa osien ja kokonaisuuden välisten kyselyiden muodostamisen ilman tarvetta spesifioida erillisiä navigointipolkuja niiden välillä.

Kompleksisten entiteettien käsittelyssä erityisen hyödyllisiä ovat kyselyt, jotka mahdollistavat vastauksissa ekstensionaalisen eli ilmentymätason tiedon ohella tietokannan rakenteeseen eli skeemaan so. sen ekstensioon liittyvän informaation palauttamisen. Tämän ansiosta käyttäjän ei esimerkiksi kompleksisten olioiden tapauksessa tarvitse tuntea osia esittävien olioiden täsmällistä luokkaa tai rakennetta, vaan hän voi kyselykielen sisältämien primitiivien avulla sekä suorittaa tähän intensionaaliseen tietoon liittyviä kyselyitä, että asettaa ehtoja, jotka koskevat esimerkiksi osaolioiden attribuutteja tai niiden asemaa osa-kokonaisuus- ja periytymishierarkioissa. [Niemi et al., 2002a]

Kolmanneksi tietokannan kyselykieleen voidaan integroida dokumenttiorientoituneita piirteitä tukevia primitiivejä. (kts. esim. [Niemi et al., 2002a]) Dokumenttiorientoituneella tiedolla tarkoitetaan tietoa, jota ei tietokannan ekstensionaalisen tiedon tavoin ole organisoitu skeemaa eli intensiota noudattavaan semanttisesti yhdenmukaiseen tapaan. Oleellista on, että käytännön järjestelmissä on usein suuret määrät yleensä tekstuaalista tai XML-muotoista informaatiota, joka ei sinällään noudata tietokannalle tyypillistä tietomallia ja jota ei sen vuoksi ole voitu hyödyntää tietokanta-

kyselyissä. Tähän liittyviä NGIS-järjestelmien piirteitä tarkastellaan kuitenkin myöhemmin laadittavassa pro gradu –tutkielmassa, joten jätämme asian käsittelyn tältä osin tähän.

3. Semanttiset yhteydet ja niitä tukevat kyselykielet

Tässä luvussa esitämme deduktiivisiin ja olio-orientoituneisiin tietomalleihin sekä erityisesti niistä johdettuihin deduktiivisiin relationaalisiin ja olio-orientoituneisiin tietomalleihin perustuvien kyselykielten laajentamista uudella piirteellä: semanttisten yhteyksien muodostamiseen tarkoitetuilla primitiiveillä. Ehdotamme, että tämä uusi piirre voitaisiin integroida osaksi edellisessä luvussa kuvailtuja seuraavan sukupolven tietojärjestelmiä (NGIS).

Seuraavassa esittelemme ja johdamme semanttisen yhteyden käsitteen, motivoimme semanttisten yhteyksien muodostamisen tarpeellisuuden, pohdimme niitä ilmaisuvoimaan ja käytettävyyteen liittyviä aspekteja, jotka tulee huomioida semanttisia yhteyksiä muodostamaan kykenevien kyselykielten suunnittelussa sekä tarkastelemme lopuksi lyhyesti niitä potentiaalisia ongelmia, jotka liittyvät tällaisiin kyselyihin. Semanttisten yhteyksien käsitteen johtaminen aiemmasta tutkimuskontekstista on esitetty tässä puhtaasti subjektiivisen ajatuskulun perusteella, eikä se vastaa todennäköisesti tapaa, jolla ajatus todellisuudessa on kehittynyt. Tarkoitus onkin tarjota vain eräs näkökulma, joka ainakin jollain tasolla jäsentää kehiteltävää käsitteistöä aiempaan tietämykseen nähden.

3.1. Semanttisen yhteyden käsite ja sen soveltaminen

Tyypillisesti tietokannan suunnitteluvaiheessa sen skeema esitetään jonkin graafiesitystapaan perustuvan formalismin avulla. Tällainen graafi kuvaa tietokannan intensionaalista rakennetta. Graafi soveltuu kuitenkin myös tietokannan ekstensionaalisen tietosisällön kuvaamiseen. GOOD [Gyssens et al., 1990] on esimerkki graafiorientoituneesta tietomallista, jossa tietokanta on organisoitu graafin muotoon. GOOD tukee myös olio-orientoituneita piirteitä ja yleisesti funktionaaliset tai olio-orientoituneet tietokannat voidaankin esittää GOOD-tietomallia käyttäen. Käytettäessä RDOOM-järjestelmän tapaista relaationaalista ja olio-orientoitunutta tietomallia luonnollinen tapa esittää tietokannan ilmentymätason tietosisältö graafina on kuvata oliot graafin solmuiksi ja assosiaatiot niiden välisiksi kaariksi. Korkeampaa astelukua olevat assosiaatiot voidaan aina palauttaa binaarisiksi assosiaatioiksi funktionaalisen tietomallin esittelyn yhteydessä kuvatulla tavalla.

Koska RDOOM-tietomallia noudattavan tietokannan intensionaalisen tason täyttävä ilmentymätason tieto edellisen mukaisesti voidaan mieltää graafina, tietokantaan voidaan periaatteessa soveltaa kyselyitä, jotka perustuvat graafiteoreettisiin menetelmiin. Tyypillinen graafiteoreettisten menetelmien osajoukko ovat kahden solmun välillä olevien polkujen etsimiseen perustuvat algoritmit. Tällaisten menetelmien soveltaminen tietokantakyselyissä ei ole uusi ajatus. Erityisesti tieverkostojen ja prosessien suunnittelun yhteydessä on kirjallisuudessa kyselykieliä, jotka mahdollistavat esimerkiksi lyhyimmän kahden kaupungin välisen tiereitin etsimisen (kts. esim. [Zhao and Zaki, 1994], [Mannino and Shapiro, 1990]). Huomiotta on kuitenkin jätetty olennainen aspekti, joka laajentaa graafikyselyiden sovellettavuutta myös laajaan joukkoon muunlaisia tietokantoja: olioiden tai niitä edustavien solmujen välisen polun semanttinen tulkinta.

Perinteisissä graafisovelluksissa solmujen välinen polku muodostetaan yleensä semanttisesti homogeenisten suhteiden yli. Ne kuvaavat esimerkiksi tieverkon osia tai prosessin vaiheita. Tällaisissa sovelluksissa polun etsiminen kahden solmun välillä vastaa transitiivisen sulkeuman muodostamista kyseiselle suhdetyypille [Mannino and Shapiro, 1990]. Transitiivisen sulkeuman muodostaminen voidaan esittää Datalog-tyylisen pseudokoodiesityksen avulla kuten ohjelmakatkelmassa 1. Oletamme tässä yksinkertaisuuden vuoksi, että predikaateissa esiintyvät atomit vastaavat yksittäisten järjestelmään tallennettujen olioiden oliotunnisteita ja, että predikaateilla kuvataan olioiden välisiä suhteita eli assosiaatioita.

```
transitiivinen_sulkeuma(A,C):-suhde(A,C).
transitiivinen_sulkeuma(A,C):-
    suhde(A,B),transitiivinen_sulkema(C,B).
```

Ohjelmakatkelma 1. Transitiivisen sulkeuman muodostaminen Datalog-tyylisenä pseudokoodiesityksenä.

On ilmeistä, että relaatioalgebraan perustuvassa kyselykielessä, esimerkiksi SQL:ssä, transitiivista sulkeumaa, ei voida muodostaa kuin kyselyssä ennalta määrättyyn syvyyteen asti [Mannino and Shapiro, 1990]. Esimerkiksi ohjelmakatkelmassa 2 esitetty SQL-kysely palauttaa vastaavan transitiivisen sulkeuman syvyystasolle kolme.

```
SELECT C1:attr1,C3:attr2 FROM suhde C1, suhde C2, suhde C3
WHERE (C1:attr2 = C2:attr1 AND C2:attr2 = C3:attr1) OR
(C1:attr2 = C3:attr1) OR (C1:attr1 = C3:attr1 AND C1:attr2 =
C3:attr2)
```

Ohjelmakatkelma 2. SQL-kysely, joka palauttaa transitiivisen sulkeuman syvyystasolle 3.

Laajennamme nyt transitiivisen sulkeuman käsitettä sallimalla sulkeuman muodostamisen mielivaltaisen suhdetyypin ylitse. Lisäksi on huomattava, että yleisessä tapauksessa Datalog-tyylisessä esityksessä tällaisen laajemman sulkeuman kannalta mielenkiintoisessa predikaatti-ilmauksessa voi olla n atomia, missä n on jokin yhtä suurempi luonnollinen luku. Edelleen sulkeuma voidaan muodostaa minkä tahansa kahden samaan predikaattiin osallistuvan atomin kautta. (Negaatiolla täydennetyn) puhtaan Datalogin ilmaisuvoima riittää periaatteessa tällaisen sulkeuman muodostamiseen, jos korvaamme suhteiden perinteisen Datalog-esitystavan uudella esitystavalla seuraavassa esitettävällä tavalla.

Jokainen predikaatti-ilmaus PredName(X_1,X_2,...,X_n), missä PredName on vastaavan suhdetta esittävän predikaatin nimi ja X_1, X_2, ..., X_n ovat n kappaletta suhteeseen osallistuvia atomeja, korvataan n:llä kappaleella predikaatti-ilmauksia, jotka ovat muotoa pred(PredName,PredID,i,X_i). PredID on tunniste, joka yksilöi yksiselitteisesti samasta alkuperäisestä Datalog-tyylisestä predikaatti-ilmauksesta generoidut uudet predikaatti-ilmaukset. Parametri i on atomin X_i asemaa alkuperäisessä predikaatissa osoittava järjestysluku, joka saa arvot yhdestä n:ään. Olkoon alkuperäinen predikaatti-ilmaus f(a,b,c). Tämä voidaan edellisen mukaisesti muuntaa kolmeksi uutta muotoa olevaksi faktaksi, jotka ovat pred(f,id_1,1,a), pred(f,id_1,2,b) ja pred(f,id_1,3,c), missä id_1 on mielivaltaisesti valittu yksilöivä tunniste. Sulkeuma voidaan nyt karkeasti ottaen muodostaa ohjelmakat-kelman 3 pseudokoodiesityksellä.

```
sulkeuma(A,C):-
    sulkeuma(A,C,ei_kaytossa_oleva_pred_id).

sulkeuma(A,C,_):-
    pred(_,PredID,_,A),
    pred(_,PredID,_,C),
    not (A = C).

sulkeuma(A,C,VanhaPredID):-
    pred(_,PredID,_,A),
    pred(_,PredID,_,B),
    not (A = B),
    not (PredId = VanhaPredId),
    sulkeuma(B,C,PredID).
```

Ohjelmakatkelma 3. Pseudokoodiesitys sulkeumalle.

Jos olemme tarkkoja, voi edellä esitetty pseudokoodi tietyissä tilanteissa johtaa syklien muodostumiseen sekä sulkeuman muodostavien atomien, että käytettyjen alkuperäisten predikaatti-ilmausten suhteen, mikä johtaa vääjäämättä päättymättömään prosessointiin. Syklien eliminoinniksi tarvitsemme tietorakenteita, joihin keräämme tiedot atomeista tai predikaatti-ilmauksista, joita on jo käytetty polun muodostamisessa. Koska lisäksi olemme kiinnostuneita nimenomaan löydetyn polun semanttisesta tulkinnasta, on kertyneen polun tallentaminen johonkin rekursiiviseen tietorakenteeseen ainoa mielekäs vaihtoehto. Käytännössä tarvitsemme edellisten ehtojen tarkistamiseen ja polun tallentamiseen kieltä, joka ilmaisuvoimaltaan vastaa Datalogia, jossa sallitaan negaation lisäksi funktiosymboleiden käyttö.

Edellä olemme käyttäneet olioiden välillä vallitsevista suhteista suhteen ohella hieman epätäsmällisesti nimityksiä relaatio, assosiaatio tai predikaatti. Jatkossa korvaamme nämä enemmän tai vähemmän rinnakkaiset ilmaukset termillä semanttinen yhteys. Välittömäksi semanttiseksi yhteydeksi kutsumme yhteyttä, joka on muodostettu ainoastaan yhden relaatiota, assosiaatiota tai predikaattia vastaavan faktan yli. Välittömän semanttisen yhteyden pituus on yksi. Edellä kuvatun sulkeuman avulla voimme muodostaa lisäksi semanttisia yhteyksiä, joiden pituus on mielivaltainen. Termin semanttinen yhteys käyttö on tässä yhteydessä mielekästä, koska haluamme korostaa

näiden yhteyksien semanttista tulkittavuutta. Käytännössä tulkinta voidaan antaa esimerkiksi luonnollisen kielen ilmausten avulla.

Olettakaamme RDOOM-tietomallia [Niemi et al., 2002b] mukaileva tietokanta, jossa on sekä oliota että niiden välillä vallitsevia semanttisia yhteyksiä. Oliot voivat edustaa esimerkiksi luokkia henkilö, auto tai yritys. Semanttiset yhteydet voivat puolestaan pitää sisällään perhe-, omistus- ja työskentelysuhteita. Eräs tällaisessa tietokannassa mahdollinen semanttinen yhteys Matti-nimisen henkilön ja auton, jonka rekisterinumero on GRP-454, välillä on seuraava: ”Matin isä työskentelee yrityksessä, jossa työskentelee myös hekilö X, joka omistaa kyseisen auton”. Olemassa olevat kyselykielet eivät suoraan tue tällaisen informaation johtamista tietokannasta. Kuitenkin voidaan kuvitella useita erilaisia tilanteita, joissa kyselykielen kyky vastata muotoa: ”Miten kaksi oliota liittyvät semanttisesti toisiinsa?” tai ”Mitkä oliot liittyvät annettuun olioon semanttisesti?” oleviin kysymyksiin voisi selvästi olla hyödyllinen. Käytännön sovelluksia voisivat olla esimerkiksi potentiaalisten intressiristiriitojen selvittäminen kahden henkilön välillä, onnettomuuden uhrin tuntevien henkilöiden löytäminen tai erilaiset rikostutkinnalliset tarkoitukset.

Ohjelmakatkelman 3 pseudokoodi kaikkine puutteineenkin antaa selvän kuvan siitä, että semanttisia yhteyksiä tuottavan kyselyn muodostaminen olemassa olevilla kyselykielillä on huomattavasti transitiivisen sulkeuman muodostamista vaativampi tehtävä ja siten tavallisen loppukäyttäjän ulottumattomissa. Tarvitaan siis kyselykieltä, joka mahdollistaa semanttisten yhteyksien etsinnän loppukäyttäjän kannalta helpolla ja deklaratiivisella tavalla.

3.2. Semanttisten yhteyksien selvittämistä tukevien kyselykielten erityispiirteet ja ongelmat

Miten semanttisen yhteyden selvittämiseen liittyvät kyselyt eroavat muista kyselykielien tukemista kyselytyypeistä? On muistettava, että perinteiset tietokantojen kyselykielet ja tietomallit mahdollistavat vain kyselyt, jotka palauttavat tietokannan ekstensioon eli sen ilmentymätasoon liittyvää informaatiota. Seuraavan sukupolven tietojärjestelmien käsittelyn yhteydessä esittelimme kyselytyyppejä, jotka kykenivät lisäksi palauttamaan tietokannan intensioon eli skeemaan liittyvää informaatiota. Semanttisten yhteyksien käsittelyn yhteydessä nämä kaksi vastaustyyppiä sekoittuvat toisiinsa. Toisaalta semanttisia yhteyksiä vastaavat polut muodostuvat ekstensionaalisen tason olioista ja yhteyksistä, toisaalta vastaus sisältää intensionaalista

informaatiota, erityisesti yhteystyypin semanttisen tulkinnan, joka auttaa selittämään, miten kyseiset oliot ja yhteydet liittyvät toisiinsa.

Myös kyselyn muodostajalta vaadittavan skeemaan liittyvän tietämyksen kannalta semanttisia yhteyksiä selvittävät kyselyt ovat mielenkiintoisia. Perinteisissä kyselykielissä, esimerkiksi SQL:ssä, kysely voidaan ymmärtää intensionaalisena spesifikaationa kyselyn tuloksena haluttavasta tiedosta [Motro, 1994]. Voidakseen muodostaa kyselyjä tällaisella kielellä käyttäjän on siis aina tunnettava täysin kyselyn kannalta relevantit skeeman osat. Semanttisia yhteyksiä muodostettaessa käyttäjän ei tarvitse periaatteessa tuntea skeemaa lainkaan. Tällainen ääritapausta edustava kysely vastaa tietokannan kaikkien semanttisten yhteyksien palauttamista, mikä ei kuitenkaan käytännössä ole relevantti kyselytyyppi kuten jäljempänä tulemme osoittamaan. Yleisessä tapauksessa käyttäjä voi tuntea joko yksilöivästi tai vain joiltain piirteiltään oliot, joiden välisistä yhteyksistä hän on kiinnostunut. Lisäksi mahdollista on, että käyttäjä antaa rajoituksia, jotka koskevat tapaa, jolla yhteys muodostetaan, esimerkiksi sen maksimipituuden, yhteyteen osallistuvat suhdetyypit jne. Yhteenvetona voidaan siis todeta, että semanttisten yhteyksien selvittämiseen tähtäävien kyselyjen muodostamisessa, toisin kuin tavanomaisissa kyselyissä, käyttäjän ei tarvitse antaa eksplisiittistä navigointipolkua kahden olion välillä, vaan tämä navigointipolku siihen liitettävine tulkintoineen on itsessään kyselyn tulos.

On huomattava, että RDOOM:lle [Niemi et al., 2002b] tyypillinen relationaalinen ja olio-orientoitunut tietomalli tarjoaa semanttisten yhteyksien muodostamiselle luonnollisen lähtökohdan. Tavallisestihan olemme kiinnostuneita juuri olioiksi mallinnettavien entiteettien välisistä suhteista; emme esimerkiksi kahden assosiaation ilmentymän välisestä yhteysketjusta. Esimerkiksi relaatiomallissa semanttisten yhteyksien muodostaminen ei ole edes mielekästä, koska siinä relaatioiden välistä navigaatioita ei ole eksplisiittisesti määritelty, vaan yhteys voitaisiin periaatteessa muodostaa minkä tahansa kahden eri monikkoon kuuluvan samanarvoisen attribuutin kautta. Periaatteessa myös relationaalisessa olio-orientoituneessa tietojärjestelmässä voi esiintyä tilanteita, joissa yhteys voitaisiin muodostaa jonkin muun kuin eksplisiittisesti navigointiin käytettävien attribuuttien tietojen perusteella. Tällainen tilanne syntyy esimerkiksi, jos asuinkunta on kuvattu henkilö-olion merkkijonoarvoiseksi attribuutiksi. Tällöin kahden henkilön välillä voitaisiin esittää semanttinen yhteys ”asuvat samalla paikkakunnalla”, mikä kahdella henkilö-oliolla kyseisten attribuuttien arvot ovat samat (vrt. esim. [Bertino et al., 1992]). RDOOM:ssa tämä voitaisiin periaatteessa määritellä deduktiivisen

suhdetyypin avulla, mutta suoraviivaisin vaihtoehto on jo tietokannan skeeman suunnittelun yhteydessä kuvata tällaiset vaihtoehtoiset navigointipolut olioiden välisinä semanttisina yhteyksinä.

Keskeisin ongelma semanttisten yhteyksien muodostamisessa on niiden potentiaalinen suuri määrä. Jos oletamme, että tietokannassa on 3 oliota, joista jokainen on yhdistetty toisiinsa välittömällä yhteydellä, on mielivaltaisen kahden olion välisten erilaisten syklittömien polkujen määrä 2. Vastaavasti olioiden määrän ollessa 5, erilaisia polkuja on vastaavassa tapauksessa 16. Kuitenkin kahdeksan tällaisen olion tapauksessa erilaisten polkujen määrä lähestyy jo kahta tuhatta. On siis selvää, että tällaisessa tilanteessa käyttäjän ei kannata tutkia koko vastausjoukkoa. Käytännössä voitaneen kuitenkin olettaa, että useissa käytännön sovelluksissa yhteystiheys ei ole näin suurin ja että käyttäjän semanttisten yhteyksien muodostamiselle antamat ehdot karsivat ainakin osan mahdollisista yhteyksistä. Todennäköistä on, että käyttäjä ei ole edes kiinnostunut kaikista mahdollisista kahden olion välillä vallitsevista semanttisista yhteyksistä. Selattuaan muutaman joukon alkupään tuloksen käyttäjä voi antaa lisäehtoja, jotka parantavat hakutuloksen tarkkuutta ja karsivat epärelevanttien yhteyksien määrää.

Ideaalista olisi tietenkin, että järjestelmä itsessään osaisi tehdä päätelmiä yhteyden relevanttiudesta. Yksinkertainen päättelysääntö voisi olla esimerkiksi, että lyhyempi semanttinen yhteys kuvastaa yleensä läheisempää ja siten vastauksen kannalta kiinnostavampaa suhdetta kuin vastaava pidempi semanttinen yhteys. Valtaosassa tapauksia tämä ehkä pitääkin paikkaansa, mutta toisaalta vastaesimerkkien keksiminen on yksinkertaista. Kahden sisaruksen välinen suhde voi muodostua esimerkiksi välillisesti yhden vanhemman kautta kahden vanhemmuusyhteyden kautta, kun taas tosille sisarista tammikuussa 2001 autonrenkaita myynyt henkilö voi liittyä tähän välittömästi yhden semanttisen yhteyden kautta. Toisaalta myöskään kaksi henkilöä, joilla on vakuutuksia samassa vakuutusyhtiössä, eivät välttämättä liity toisiinsa siinä määrin läheisesti, että kolmansien olioiden välisiä semanttisia yhteyksiä kannattaisi muodostaa tämän osayhteyden ylitse. Potentiaalisen sovellusalueen monialaisuudesta johtuen tällaisten heurististen menetelmien kehittäminen lienee varsin haasteellinen tehtävä. Niinpä jätämme kyseisen ongelma-alueen tämän tutkielman ulkopuolelle ja tyydymme korostamaan, että toteutettavan kyselykielen on tarjottava käyttäjälle riittävät välineet, joilla hän voi karsia vastausjoukosta omalta kannaltaan epärelevantteja yhteyksiä.

4. Yhteenveto

Seuraavan sukupolven tietojärjestelmiksi (NGIS) on ehdotettu deduktiivisista olio-orientoituneista tietomalleista johdettuja, mahdollisesti relationaalisia ja olio-orientoituneita piirteitä yhdistäviä, järjestelmiä. Nämä järjestelmät tukevat kyselykielten deklaratiivisuutta, tiedon loogista johdettavuutta eli deduktiota, kompleksisten entiteettien mallintamista, intensionaalisen ja ekstensionaalisen informaation käsittelyä kyselyissä sekä kyselykielten laajentamista dokumenttiorientoituneilla tekstihakuominaisuuksilla. Tampereen yliopistossa kehitetyssä mahdollisen NGIS-järjestelmän prototyyppiä edustavassa RDOOM-tietomallissa oliot kuvataan olio-orientoituneesti, kun taas niiden väliset assosiaatiot esitetään relationaalisesti. [Niemi et al., 2002a, 2002b]

Tässä tutkielmassa olemme esittäneet NGIS-järjestelmien kyselykielten laajentamista semanttisten yhteyksien muodostamista tukevilla primitiiveillä. Semanttinen yhteys on olioista ja niitä yhdistävistä assosiaatioista muodostuva polku, joka yhdistää toisiinsa kahta tietokannan ekstensioon kuuluvaa oliota. Tälle polulle annetaan jokin semanttinen tulkinta joka voidaan esittää esimerkiksi luonnollisella kielellä. Välitön semanttinen yhteys vastaa yhtä tällaista assosiaatiotyypin ilmentymää ja siihen osallistuvia olioita, kun taas välillinen semanttinen yhteys voi muodostua mielivaltaisesta määrästä olioita ja assosiaatioita.

Semanttisten yhteyksien muodostamista tukevilla kyselykielillä voimme vastata esimerkiksi kysymyksiin ”Miten kaksi tunnettua oliota liittyvät toisiinsa?” tai ”Mitkä oliot liittyvät tunnettuun olioon?”. Lisäksi näiden kielten tulee mahdollistaa sekä semanttisessa yhteydessä esiintyviin yhteystyyppeihin että olioihin kohdistettavien ehtojen tai rajoitusten muodostaminen. Totesimme, että olemassa olevien deduktiivisten kyselykielten ilmaisuvoima riittää ainakin teoriassa tällaisten kyselyiden muodostamiseen. Kyselynmuodostus on kuitenkin dramaattisesti vaikeampi ja laajempi tehtävä kuin esimerkiksi transitiivisen sulkeuman laskeminen. Tämän vuoksi semanttisten yhteyksien selvittäminen onkin olemassa olevilla kyselyvälineillä tavallisen loppukäyttäjän ulottumattomissa.

Viiteluettelo

[Alhajj and Polat, 2000] Reda Alhajj and Faruk Polat, Closure maintenance in an object-oriented query model. In: *Proc. of the Third Iternational Conference on Information and Knowledge Management*, 72 – 79.

[Bagui, 2003] Sikha Bagui, Achievements and weaknesses of object-oriented Databases. *Journal of Object Technology* **2**, 4 (Jul. – Aug. 2003), 29 – 41.

[Bertino et al., 1992] Elisa Bertino, Mauro Negri, Giuseppe Pelagatti and Licia Sbattella, Object-oriented query languages: the notion and the issues. *IEEE Transactions on Knowledge and Data Engineering* **4**, 3 (Jun. 1992), 223 – 237.

[Buneman and Frankel, 1979] Peter Buneman and Robert E. Frankel, FQL – User Interfaces: A functional query language. In: *Proc. of the 1979 ACM SIGMODI International Conference on Management of Data*, 52 – 57.

[Cattell and Rogers, 1986] G. G. Cattell and T. R. Rogers, Combining object-oriented and relational models of data. In: *Proc. on the 1986 International Workshop on Object-Oriented Database Systems*, 212 – 213.

[Codd, 1970] E. F. Codd, A relational model of data for large shared data banks. *Communications of the ACM* **13**, 6 (Jun. 1970), 377-387.

[Date, 1989] C. J. Date, *A Guide to the SQL Standard.* Addison-Wesley, 1989.

[Gyssens et al., 1990] Marc Gyssens, Jan Paredaens and Dirk Van Gucht, A graph-oriented object database model. In: *Proc. of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 417 – 424.

[Hillebrand and Kanellakis, 1994] Gerd G. Hillebrand and Paris C. Kanellakis, Functional database query languages as typed lambda calculi of fixed order. In: *Proc. of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 222 – 231.

[Järvelin and Niemi, 1999] Kalervo Järvelin and Timo Niemi, Integration of complex objects and transitive relationships for information retrieval. *Information Processing & Management* **35**, (1999), 655 – 678.

[Kim, 1990] Won Kim, Research directions in object-oriented database systems. In: *Proc. of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1 – 15.

[Knuutila, 2001] Timo Knuutila, Ohjelmointikielten periaatteet –luentomateriaali. Turun yliopisto, Informaatioteknologian laitos, 2001. Saatavilla: http://www.cs.utu.fi/knuutila/Courses/okp/default.html [14.12.2003].

[Koskimies, 2000] Kai Koskimies, *Oliokirja*. Satku – Kauppakaari, Jyväskylä, 2000.

[Liu, 1996] Mengchi Liu, The ROL deductive and object-oriented database system. 1 – 15. University of Regina, Dept. of Computer Science, Report **TR 96-02**, 1996. Also available as http://www.cs.uregina.ca/research/Techreports/9602.ps [14.12.2003].

[Liu, 1999] Mengchi Liu, Deductive database languages: problems and solutions. *ACM Computing Surveys* **31**, 1 (Mar. 1999), 27 – 62.

[Mannino and Shapiro, 1990] Michael V. Mannino and Leonard D. Shapiro, Extensions to query languages for graph traversal problems. *IEEE Transactions on Knowledge and Data Engineering* **2**, 3 (Sep. 1990), 352 – 363.

[Motro, 1994] Amihai Motro, Intensional answers to database queries. *IEEE Transactions on Knowledge and Data Engineering* **6**, 3 (Jun. 1994), 444 – 454.

[Niemi, 2003] Timo Niemi, Logiikkaohjelmointi –luentomateriaali. Tampereen yliopisto, Tietojenkäsittelytieteiden laitos, 2003.

[Niemi et al., 2000] Timo Niemi, Maria Christensen, Kalervo Järvelin, Query language approach on the deductive object-oriented database paradigm. *Information and Software Technology* **42**, (2000), 777 – 792.

[Niemi et al., 2002a] Timo Niemi, Marko Junkkari, Kalervo Järvelin and Samu Viita, Advanced query language for manipulating complex entities. University of Tampere, Department of Computer and Information Sciences, Report **A-2002-17**, 2002.

[Niemi et al., 2002b] Timo Niemi, Marko Junkkari and Kalervo Järvelin, Relational object-oriented modeling (RDOOM) approach for finding, representing and integrating application-specific concepts. *International Journal of Software Engineering and Knowledge Engineering* **12**, 4 (2002), 415 – 451.

[Rosser, 1982] J. Barkley Rosser, Highlights of the history of the lambda-calculus. In: *Proc. of the 1982 ACM Symposium on LISP and Functional Programming*, 216 – 225.

[Paton et al., 1996] Norman Paton, Richard Cooper, Howard Williams and Philip Trinder, *Database Programming Languages.* Prenctice Hall, 1996.

[Premeriani et al., 1990] William J. Premeriani, Michael R. Blaha, James E. Rumbaugh and Thomas A. Varwig, An object-oriented relational database. *Communications of the ACM* **33**, 11 (Nov. 1990), 99 – 109.

[Samet, 1981] P. A. Samet ed., *Query Languages – a Unified Approach.* Heyden & Son Ltd., The British Computer Society, Cambridge, 1981.

[Shipman, 1981] David W. Shipman, The functional data model and the data language DAPLEX. *ACM Transactions on Database Systems* **6**, 1 (Mar. 1981), 140-173.

[Ullman, 1988] Jeffrey D. Ullman, *Principles of Database and Knowledge-Base Systems, Volume I: Classical Database Systems*. Computer Science Press, Rockville, 1988.

[Warner and Odle, 1981] Hoyt D. Warner and David Odle, A high-level functional query language for a small relational system. In: *Proc. of the 1981 ACM SIGSMALL Symposium on Small Systems and SIGMOD Workshop on Small Database Systems*, 90 – 95.

[Zaniolo et al., 1997] Carlo Zaniolo, Stefano Ceri, Christos Faloutsos, Richard T. Snodgrass, V. S. Subrahmanian and Roberto Zicari, *Advanced Database Systems.* Morgan Kaufmann Publishers, Inc., San Fransisco, 1997.

[Zhao and Zaki, 1994] J. Leon Zhao and Ahmed Zaki, Spatial data traversal in road map database: a graph indexing approach. In: *Proc. of the Third International Conference on Information and Knowledge Management*, 355 – 362.

Temporaalitietokannoista

Kimmo Kajas

Tiivistelmä

Tutkimuksessa esitellään aikasidonnaisen tiedon olennaisia käsitteitä ja tallennusmahdollisuuksia. Työssä esitellään temporaalitietokantojen alan tärkeimpiä saavutuksia sekä samalla alan merkittävimmät vaikuttajat.

Avainsanat ja -sanonnat: temporaali, historiatieto, tietokanta, aikajaksot, aikasarjat, SQL.
CR-luokat: H.1.1

1. Johdanto

Lähtökohtana tutkimukselleni on ajallisten ilmiöiden huomioon ottaminen tallennettaessa tietoja tietokantoihin. Tarkastelen aikakäsitteen yleisimpiä ongelmia tuottavia ominaisuuksia. Esittelen muutamia ongelmia ja niiden ratkaisuja sekä sovelluksia, joissa ratkaisuja on käytetty jo hyväksi.

Huomattavan laajalti tietokantajärjestelmissä käytetään edelleenkin SQL-92-kieleen perustuvia tietokantoja, jotka eivät itsessään tue historiatietojen tallentamista tai jotka eivät sovi luontevasti ajallisten ilmiöiden tallentamiseen tai varsinkaan niiden hakuihin tietokannoista. Aikaisemmin tiedot on tallennettu tavalliseen tietokantaan siten, että ajallinen dimensio on tallennettu aikaleimoina. Tällainen toteutus toimii yksinkertaisessa tietokannassa, mutta ongelmat tulevat esiin silloin, kun tapahtumia joudutaan vertailemaan niiden ajallisten ominaisuuksien kesken. Sovelluksen tarvitsema historiatieto tai hieman vaativampi ajallinen malli vaatisi tavanomaisen tietokannan rinnalle SQL-92-standardista poikkeavaa toteutusta.

Temporaalisen tiedon tallentamiseen soveltuvia ohjelmia on alettu kehittää vasta viime vuosina. Aikasarjojen toteutuksen alallakin alettiin ratkaista ongelmia vasta 1990-luvun alkupuolella, vaikka ongelmat tiedostettiin jo yli kymmenen vuotta tätä aiemmin. Ennen 1990-lukua teoreettinen tutkimus ja sovellusten kehitteleminen keskittyi lähes ainoastaan ajallisten ilmiöiden tutkimiseen aikajaksojen näkökulmasta.

Nykyisellä SQL-92-standardin mukaisilla tietokannoilla on mahdollista toteuttaa ajallisiin ilmiöihin kohdistuvia monimutkaisempia hakuja, mutta ongelmaksi muodostuu tietokannan ajallisten operaatioiden puute. Puut-

teesta johtuen ilmiöihin kohdistuvien ajallisten hakujen käsittely joudutaan siirtämään tietokantaohjelmien ulkopuolelle. Ajallisen logiikan toteutus jää näin ollen sovelluskehittäjän tehtäväksi.

Toteutuksessa ensimmäisiä ongelmia ovat erilaisten kalenterien mallintamisen ongelmat, joiden ratkominen on yllättävän vaivalloista, kun otetaan huomioon, kuinka paljon erilaisia kalentereita käytössämme saattaa olla. Lisäksi puhtaasti ajan keskinäisten relaatioiden käsittelystä muodostuu seuraava ongelma: tavanomaisella SQL-92-lauseella on vaikeaa etsiä jokin tapahtuma, joka on voimassa tai tapahtumaa, joka on seuraava jonkin toisen tapahtuman jälkeen.

Ajasta riippuvien tietojen tallentamiselle on ollut tarvetta aivan tietojenkäsittelyn historian alusta alkaen. Snodgrass ja Ahn [1985] ovat todenneet, että jo 1976 J. A. Bubenko teki tutkimusta IBM:lle ajallisten ilmiöiden kuvaamisesta tietojärjestelmiin. Ajallisten ilmiöiden tallennuksesta tehdään edelleenkin paljon tutkimusta.

2. Ajan käsitteitä

Tavallisesti aika on sitä, mitä rannekellosta näkyy tällä hetkellä. Aika merkitsee ihmisille tätä hetkeä tai etäisyyttä tästä hetkestä eteenpäin tai taaksepäin. Ajoittain käsittelemme tapahtumia verraten niitä toisiinsa ja samalla vertailemme tapahtumahetkiä toisiinsa. Harvemmin vertailemme suurta joukkoa tapahtumia ja järjestämme niitä kronologiseen järjestykseen.

Aikaa käsitteenä voidaan tutkia monesta eri näkökulmasta. Hayes [1995] esittelee englannin kielen sanan *time* merkityksiä: ajan fysikaalinen ulottuvuus (time-dimension), aikajana l. aika-avaruus (time-line), ajanjakso (time-interval), aikapiste (timepoint), kesto (duration) ja ajankohta (time-position). Näistä termeistä ajanjaksoa, aikapistettä, kestoa ja ajankohtaa käytetään yleensä temporaalisen tiedon tallentamisen yhteydessä.

Aikapisteet muodostavat ajanjakson osoittamalla alku- ja päätepisteet jaksolle. Aikapiste ja ajanjakso ovatkin käsitteinä yhteensopivia keskenään. Sen sijaan tapahtuman kesto ei ole täysin yhteensopiva käsite aikapisteen ja ajanjakson kanssa. Tästä huolimatta keston määrittelyyn tarvitaan kuitenkin jonkinlaista ajanjaksoa, kuten Hayesin [1995] mainitsemaa standardiajanjaksoa, esimerkiksi silmänräpäys tai päivä.

Yhteensopivuus syntyykin käsitteiden abstraktiotasoeroista. Kesto on omalla abstraktiotasollaan ja ajankohta on kestoa konkreettisemmalla tasolla. Ajankohdalla tarkoitetaan tiettyä nimettyä hetkeä aikajanalla, kuten esimerkiksi 12.00 tai 24.12.2003. Ajankohdat voivat määrätä keston samoin kuin

aikapisteetkin voivat. Kestoa voidaan käyttää abstraktina käsitteenä puhuttaessa esimerkiksi ajanjakson kestosta. Kahden ajankohdan välistä eroa voidaan myös kutsua kestoksi.

Tietojärjestelmien suunnittelussa on huomattu, että ajalla on erittäin ratkaiseva merkitys, kun tallennetaan suurta määrää esimerkiksi mittaustietoa. Enää ei riitä pelkästään tapahtuman tapahtumahetki, vaan tarvitaan myös tallennushetki, voimaantuloaika ja voimassaolon päättymishetki.

2.1. Ajan rakeisuus, ankkuroitu aika ja kalenteri

Aikajana jaetaan tavallisesti ennalta sovittuihin aikayksiköihin. Normaalisti aika jaetaan paikallisen kalenterin käyttämiin yksiköihin, joita ovat sekunnit, minuutit, tunnit, päivät, kuukaudet ja vuodet. Näitä yksiköitä käytetään myös tietojärjestelmissä, mutta käsiteltäessä yleisesti aikayksiköitä käytetään käsitteitä *chronon* ja *granule*. *Chronon* tarkoittaa fysiikassa hypoteettista ajan yksikköä, joka on valon käyttämä aika matkatessaan elektronin halkaisijan pituisen matkan. *Chronon*in oletetaan olevan jakamaton yksikkö. Tietojärjestelmissä samaa termiä käytetään järjestelmään toteutetusta pienimmästä ajan yksiköstä. Jakamattomuus on käytetyn termin tärkein ominaisuus. *Granule* on karkearakeinen joukko *chronon*eita, jotka voidaan yhdistää päiviksi, kuukausiksi tai vuosiksi [Dyreson and Snodgrass, 1998]. *Granularity* eli ajan rakeisuus onkin yksi temporaalitietokantojen peruskäsitteistä. Näillä termeillä käsitellään aikamallien rakeisuutta, aikayksiköiden muunnon ongelmia sekä käytössä olevien aikayksiköiden jakamisen ongelmia [Goralwalla *et al*., 2001].

Tapahtumat voidaan ilmaista aikajanaan ankkuroituna (anchored) tai ankkuroimattomina (unanchored). Ankkuroimattomista käytetään usein myös termiä *interval* ja ankkuroiduista termiä *period* [Dyreson and Snodgrass, 1998]. Ajan ankkuroitu esitys ilmaistaan esimerkiksi 1. helmikuuta 2001, jossa ilmoitetaan aikajanan tarkka paikka. Ankkuroimaton esitys ilmaistaan matkana aikajanalla, esimerkiksi kaksi päivää ja kolme tuntia. Ajan ankkuroiminen liitetään yleensä kalentereihin.

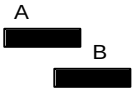
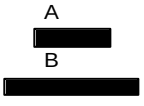
Goralwalla *et al*. [2001] ovat tutkineet muun muassa kalentereiden esittämistä. Kalenterit ovat ihmisen tapa esittää fyysinen aika helpommin ymmärrettävässä muodossa. Edelleenkään maailmassa ei käytetä ainoastaan yhtä kalenterijärjestelmää, vaan erilaisia kalentereita on useita. Useinkaan ei tulla ajatelleeksi, että Suomessakin käytetään useampaa kalenterijärjestelmää. Gregoriaanisen kalenterin lisäksi meillä on käytössä organisaatioiden asettamia ajanlaskumalleja. Yrityksissä ja koulutuslaitoksissa ovat omat kalenterinsa, joilla aikaa jaksotetaan. Ajallisten ilmiöiden tallentaminen

järjestelmään tuottaakin ongelmia, koska kalentereiden seuranta pakottaa toteuttamaan järjestelmään useita päällekkäisiä kalentereita. Käytössä olevien kalentereiden ajan rakeisuutta onkin kyettävä muuntamaan keskenään, niin että kalentereita voidaan käyttää rinnakkain. Yhteiskunnan käyttämien kalentereiden vertaaminen keskenään on kuitenkin melko helppoa, koska pienimmät yksiköt ovat saman mittaisia. [Goralwalla *et al.*, 2001]

2.2. Aikajaksot, aikasarjat

Ajallisten ilmiöiden tallentamisessa tutkijat ovat kiinnittäneet huomiota suurelta osin aikajaksoihin. Perimmäisenä oletuksena on se, että kaikilla tapahtumilla on alku- ja loppupiste ajassa. Jakson päätepisteet eivät välttämättä ole tiettyjä aikaleimoja, jotka voidaan osoittaa tiettyinä nimettyinä aikaalkioina. Aikajaksoja tarkasteltaessa ei keskitytä tapahtuman tapahtumiseen tietyssä hetkessä, vaan painopisteenä on lähinnä jakson suhde muihin jaksoihin.

Allen [1983] esitti, kuinka kaksi ajanjaksoa voivat olla suhteessa toisiinsa. Hän esitteli 13 mahdollista suhdetta, miten kaksi jaksoa voivat olla järjestyneenä keskenään. Kuvasta 1 ilmi, että suhteiden ajan kulkusuunnassa käänteiset tapaukset on otettu huomioon.

Suhde	Symboli	Käänteinen symboli	
A ennen B:tä	<	>	
A yhtä kuin B	=	=	
A kohtaa B:n	m	mi	
A leikkaa B:n	o	oi	
A B:n aikana	d	di	
A aloittaa B:n	s	si	
A lopettaa B:n	l	li	

Kuva 1. Temporaalisuhteet Allenin [1983] mukaan

Nämä suhteet ovat aikajaksojen käsittelyssä tärkein perusta, jolla selviää useimmat temporaalisen päättelyn ongelmat. Allenin päättelysääntöjä on käytetty monissa eri yhteyksissä, ja hänen tutkimustuloksiaan on hyödynnetty ahkerasti.

Allen esittelee artikkelissaan erään toisenkin huomionarvoisen seikan aikajaksojen ominaisuuksissa. Aikajaksojen päätepisteet eivät ole niinkään pisteitä, joilla ei ole leveyttä, vaan ne ovat äärimmäisen pieniä aikajaksoja. Allen mainitsee tästä ominaisuudesta esimerkkinä valojen sammuttamisen. Valojen sammuttaminen kuvataan kahdella aikajaksolla, joista toinen kuvaa tilannetta, kun valot ovat päällä ja toinen tilannetta, kun valot ovat pois. Allen esittelee tilanteen, jossa nämä kaksi jaksoa kuvaavat valojen sammuttamisen. Jos jaksot kohtaavat, syntyy aikapiste, jossa valot ovat yhtä aikaa päällä ja pois. Kun taas jaksot ovat erillään, jaksojen väliin muodostuu ajankohta, jossa valot eivät ole pois eivätkä päällä. [Allen, 1983]

Aikasarjat ovat käytössä teknisellä alalla, jossa tapahtumien arvot ovat yleensä numeerisia ja arvoista halutaan tietää voimaanastumishetki ja voimassaoloaika. Kuten aikajaksojen käyttö myös aikasarjojen käyttö edellyttää erikoistunutta hakukieltä. Hakukielellä onkin suuri merkitys tietokannan onnistumiselle ja yleistymiselle.

Aikasarjoista on tehty tutkimusta jo useita vuosia [Dreyer *et al.*, 1994]. Kuitenkin yleiskäytännöllisiä toteutuksia on ilmaantunut markkinoille vasta 1990-luvulla. [Wolski *et al.*, 1999]. Aikasarjat ovat käyttökelpoisia sellaisissa sovelluksissa, joissa tietoalkioita tallennetaan huomattavia määriä sekunnissa. Aikasarjatoteutuksissa tarkastellaan tapahtumia niiden mittaushetkinä ja mittaustulosten voimassaoloaikoina.

Tallennushetki (transaction time) tarkoittaa hetkeä, jolloin mittaus tai kuvailtava tapahtuma tallennetaan. Esimerkiksi tarkka hetki, jolloin prosessin lämpötilan mittaus tallennetaan, on tallennushetki. Voimassaoloaika (valid time) merkitsee hetkeä, jolloin tallennettu tapahtuma on tai oli voimassa. Varsinkin teollisessa prosessitiedon tallentamisessa on tärkeää tietää, milloin mittaustiedot ovat voimassa. Tällöin voimassaoloaika voidaan tallentaa kahdella erillisellä aikaleimalla. Toinen aikaleima osoittaa mittaustuloksen voimaanastumishetken ja toinen voimassaolon päättymishetken. Tällä menettelyllä voidaan kuvata myös nykyisen mittaustuloksen voimassaolo jättämällä voimassaolon päättymishetki tyhjäksi.

Mittaussarjan ominaisuudet voivat vaihdella. Säännöllisessä sarjassa mittausajat ovat säännöllisin aikavälein. Epäsäännöllisessä sarjassa

mittausajat voivat olla mielivaltaisin väliajoin tapahtuvia. Tämän lisäksi mittaustuloksien historiatiedot voidaan tallentaa esimerkiksi koostearvoina, jolloin kaikkia historiatietoja ei tarvitse säilyttää tietokannassa, vaan tieto voidaan tallentaa juuri sellaisella tarkkuudella, kuin vaaditaan. [Wolski *et al.*, 1999]

2.3. Nykyhetki

Temporaalitietokantojen yhteydessä tulee keskustelun aiheeksi hyvin usein myös nykyhetken käsite. Nykyhetken (current-time, NOW) käsitteleminen temporaalitietokannoissa tuo esiin ongelmia, joissa esimerkiksi nykyhetken määritteleminen tuottaa vaikeuksia tai tietokannan toteutuksessa lukitukset aiheuttavat sen, että nykyhetki näyttää liikkuvan hetkellisesti takaperin. Toisaalta joudumme myös tarkkailemaan tilannetta, jossa nykyhetki aikajaksojen joukossa liikkuu eteenpäin ja osoittaa jaksot, jotka ovat nykyhetkessä päteviä. Tavallisissa tietokannoissa tällaista ominaisuutta ei ole käytössä, vaan se toteutetaan toteuttajan oman mielikuvituksen pohjalta.

Nykyhetkeen liittyy läheisesti tiedonpätevyyden tarkastelu. SQL-92-tietokannat tarjoavat transaktioajan (transaction-time), jolla voidaan todeta, milloin tieto tallentui pysyvästi tietokantaan. Temporaalitietokannoissa tarjotaan voimassaoloaika (valid-time), jolla voidaan osoittaa tiedon voimassaolo jonkin ajanjakson aikana. [Elmasri *et al.*, 1993.] Tavallisessa tietokannassa sitouttamishetki kertoo tiedon olevan voimassa siihen asti, kun tieto vanhentuessaan korvataan uudella tiedolla. Temporaalitietokannassa tieto on voimassa silloin, kun nykyhetki kuuluu voimassaoloaikajaksoon. Allenin mukaan voimassaolohetki voi sijoittua myös tulevaisuuteen [1983].

Finger ja McBrien esittelevät kolme mahdollista määritelmää, joilla nykyhetki voidaan määritellä tietokannassa. Ensimmäinen keino on määritellä nykyhetki tallennettujen tietojen mukana tulleesta tiedosta. Esimerkkinä tälle vaihtoehdolle on tietokantaan tallennettava tieto "Mikon ja Matin palkka on *tällä hetkellä* 2000 e". Tätä vaihtoehtoa ei kannata käyttää laajemmin sen epämääräisyyden vuoksi. Toinen vaihtoehto todeta nykyhetki on käyttää todellista aikaa. Nykyhetki selvitettäisiin tämän vaihtoehdon perusteella tietokoneen kellosta välillisesti SQL-92-standardin CURRENT_TIMESTAMP-muuttujalla. Fingerin ja McBrienin mukaan tämäkään ei ole hyvä vaihtoehto, koska nykyhetki voisi muuttua tapahtuman aikana. Ongelmaksi muodostuvat tilanteet, joissa yksi päivitystapahtuma kestää kauemmin kuin tietokannassa tarkasteltavan lyhimmän aikayksikön kesto. Esimerkkinä tästä on tilanne, jossa rahaa siirretään tililtä toiselle ja nykyhetki ehtii siirron aikana

vaihtua. Tällöin siirrettävä rahamäärä katoaa hetkeksi ja tästä johtuen rahan siirtotapahtuman jäljessä seuraava tapahtuma saattaa epäonnistua kadonneen rahasumman takia. Kolmas vaihtoehto tarkastelee nykyhetkeä tietokannan tapahtumien näkökulmasta. Tapahtuman käsittelyn vaiheiden *submit-time* ja *begin-time* käyttö aiheuttaa ajassa taaksepäin liikkumisen ilmiön. Finger ja McBrien [1996] toteavat lopuksi, ettei SQL-92-standardin mukaisella tietokannalla saada täydellisesti toimivaa toteutusta aikaiseksi. Temporaalitietokannasta on kehitettävä sellainen, joka ohjaa nykyhetken tarkastelua itsenäisesti tarkoitukseen suunniteltujen tapahtuman käsittelysääntöjen mukaan.

Aikasarjojen toteutuksissa nykyhetken ongelmat ovat hieman erilaisia. Tapahtumasta saatavan tiedon mittaushetkestä kestää mahdollisesti useampi aikayksikkö tiedon pysyvään tietokantaan tallentamiseen eli sitouttamiseen. Tällöin tietokannan nykyhetki on hieman jäljessä todellisesta nykyhetkestä.

2.4. Historiatiedot tietokannoissa

Historiatietokannoilla tarkoitetaan järjestelmiä, joissa tietokanta hoitaa automaattisesti vanhentuneiden tietojen tallentamisen. Tällaista järjestelmää kuvaillaan usein kolmiulotteisena mallina, jossa normaali tietokanta on kaksiulotteinen taulukko ja historiatiedot muodostavat kolmannen ulottuvuuden. Historiatietokantoja voidaan käyttää tavalliseen tapaan tavanomaisella SQL-kielellä, jolloin tietokannasta näkyy vain voimassaolevat tiedot. Historiatietoja voidaan toteutuksesta riippuen muokata ja tarkastella SQL-kielen laajennettuja komentoja käyttäen. Historiatietoja on tietenkin mahdollista tallentaa SQL-92-standardin mukaisella tietokannalla, mutta tällöin tietokannan käyttäjä joutuu huolehtimaan monimutkaisesta toteutuksesta itse, eikä kaikkia historiatietokannan ominaisuuksia ole mahdollista toteuttaa tavanomaisin keinoin.

3. SQL-kielen laajennukset

Ensimmäiset tietokantaohjelmat tai ohjelmien prototyypit, joissa oli jonkinlainen tuki ajallisten ilmiöiden tallentamiseen tai hakuun ilmestyivät jo vuonna 1971. Findler ja Chen kehittivät vuonna 1971 tietokantaohjelmaa AMPPL-II, jossa otettiin huomioon *valid-time* eli hetki, jolloin tieto on paikkansa pitävä todellisuudessa. [Snodgrass, 2003]

Ensimmäinen kaupallinen versio SQL-kielestä valmistui 1970-luvun lopussa. SQL-standardia on päivitetty ahkerasti tähän päivään asti ja viimeisin päivitys on SQL:1999. Työnimenä projektilla on ollut SQL3, jota tänäkin päivänä käytetään epätietoisessa keskustelussa valmiista SQL:1999 –stan-

dardista. SQL3-standardiin lisättiin objektimalli ja muita ominaisuuksia, jotka puuttuivat aiemmasta SQL-92-standardista. SQL3-standardiin oli tarkoitus lisätä myös SQL/Temporal–temporaalimalli, mutta se otettiin SQL3-standardista kuitenkin pois vuonna 2001 pari vuotta standardin valmistumisen jälkeen. Syy temporaalimallin poistoon standardista oli komitean sisäinen erimielisyys temporaalimallin paikasta SQL3-standardissa; temporaalimallia ei haluttu jättää jälleen kerran vain ulkoiseksi laajennukseksi. [Snodgrass, 2003]

3.1. TSQL2 ja TQuel

Richard Snodgrass kehitteli jo 1985 Tquel-laajennusta Ingres-nimisen yrityksen Quel-tietokantaan, joka otti huomioon ajan tietojen käsittelyssä. Valitettavasti Quel hävisi markkinoilta 1980-luvun loppupuolella SQL:n noustessa käytetyimmäksi hakukieleksi tietokannoissa. Quel kehitettiin yliopistoissa ja sen uskottiin olevan paremmin suunniteltu kuin SQL. Tästä huolimatta IBM:n vaikutuksesta SQL sai valta-aseman tietokantojen hakukielten joukossa. [SQL]

TQuel hallitsi historiatietojen haut ja temporaalisten suhteiden algebran. Edellä esitetyistä Allenin kahden aikajakson välisistä suhteista TQuel hallitsi suhteet ’A kohtaa B:n’ (precede), ’A aloittaa B:n’ (start of) ja ’A B:n aikana’ (overlap). [Snodgrass and Ahn, 1985]

Snodgrass oli puheenjohtajana TSQL2-kielen suunnittelukomiteassa, jossa oli mukana myös lähes kaikki alan vaikuttavat henkilöt. Komitean perustamisesta kesästä 1993 asti TSQL2:ta kehitettiin syyskuuhun 1994, jolloin julkaistiin kielen spesifikaatio. TSQL2:n rakennetta ja oivalluksia ehdotettiin liitettäväksi SQL3-kieleen. ATSQL on TSQL2:n pohjalta muodotettu hieman laajempi kieli, mutta ei kuitenkaan kopio TSQL:stä. [Snodgrass, 2003]

3.2. SQL3 ja SQL/Temporal

Richard Snodgrass ja muutamat muut TSQL2-kielen suunnittelukomiteasta ovat olleet mukana hankkeessa saada aikaominaisuudet TSQL2:sta SQL3:een. Vuonna 1995 saatiinkin SQL/Temporal-niminen osio mukaan SQL3:n spesifikaatioon, kuten jo aiemmin on mainittu. Vuoteen 1998 mennessä hyväksyttiin muutama uusi ehdotus SQL3:een ja hyväksyttyjä ominaisuuksia alettiin toteuttaa. Toteutuksen lopputulos oli TimeDB-niminen prototyyppi, josta tuli kaupallinen tuote TimeConsult-yrityksen omien www-sivujen mukaan [TimeConsult, 1999]. TimeDB on kuitenkin jäänyt vain kaupalliseksi prototyypiksi, eikä TimeConsult-yrityksen TimeDB-www-sivuja ole päivitetty

vuoden 1999 jälkeen. Vaikka työtä oli tehty paljon temporaalituen mahdollistamiseksi, vuoden 2001 lopulla ISO-komitea hylkäsi temporaalituen SQL3:n spesifikaatiosta. [Snodgrass, 2003]

3.3. Muut laajennukset

Ajallisten ilmiöiden käsittelemiseen tarkoitettuja laajennuksia on esitelty huomattava määrä, mutta yksikään ei ole levinnyt yleiseksi käytännöksi. Maantieteellisissä ongelmissa tarvitaan myös spatiaalinen eli sijaintiin liittyvä tieto. Tällaisten ongelminen mallintamiseen on kehitelty spatio-temporaalista SQL:n laajennusta nimeltään STSQL tai STQL. Kieli perustuu SQL3-, TSQL2- ja Spatial SQL-kieliin. [Kim *et al.*, 2002]

TIP on Jun Yangin ja hänen kumppaneidensa tekemä temporaali-laajennus Informixin tietokantaan. Heidän kuvailemansa laajennus on melko yksinkertainen ja siihen on sisällytetty vain muutamia temporaaliominaisuuksia. Monimutkaisempien kyselyiden optimointi voi olla vaikeaa, koska TIP on vain tietokannan laajennusrajapintaan liitetty laajennus. [Yang *et al.*, 2000]

4. Sovelluksia

Temporaalitietokantoja on toteutettu useita. Toteutukset ovat olleet yleisimmin joko aitoja temporaalitietokantoja tai laajennuksia tavallisille SQL-92-tietokannoille. Aidot temporaalitietokannat ovat tarkasti tiettyyn tarpeeseen varta vasten tehtyjä ohjelmia, joita ei ole voinut käyttää yleisesti ohjelmistojen tietokantoina. Tavallisiin SQL-92-tietokantoihin tehdyt laajennukset on tehty tietokantojen laajennettaville versioille, joissa on oma rajapinnan laajennusrajapinta.

GIS (Geographic Information System) on maantieteellisen tiedon mallinnusjärjestelmä, johon on suunniteltu temporaaliominaisuuksia. GIS-järjestelmään suunniteltu STQP (Spatiotemporal database query prosessor) pohjautuu SQL3- ja TSQL2-kieliin. Järjestelmässä hyödynnetään varsinkin historiatietojen käsittelyominaisuuksia. Ominaisuuksia tarvitaan esimerkiksi merivirtauksien liikkeiden muutosten tutkimiseen. [Kim *et al.*, 2002]

Toisena esimerkkinä on suomalainen RapidBase-tietokanta, joka on suunniteltu teollisuuden tarpeet huomioon ottaen. VTT Tietotekniikka on kehittänyt RQL-kielen ja sitä vastaavan tietomallin, joissa on otettu huomioon teollisuusprosesseissa syntyvän tiedon suuren määrän hallinnan vaikeudet ja ajallisten ilmiöiden kuvaamisen tarve. Antoni Wolski on kollegoineen todennut, ettei yleisimmin käytetty SQL-92-relaatiomalli sovellu kovin hyvin

ajallisten ilmiöiden mallintamiseen. He antavat esimerkkinä tilanteen, jossa tallennettujen mittaustietojen vanhentuneita eli historiatietoja pitäisi tutkia voimassaololiitoksella. Tietokannan tauluista pitäisi etsiä arvot, joka olivat tietyllä hetkellä voimassa, mutta tällaista ei voida toteuttaa pelkällä SQL-kielellä. Sen sijaan RapidBase-tietokannan tietomallissa aikasarjat ja historiatietojen hallinta ovat olleet toteutuksessa etusijalla. [Wolski *et al*., 1999]

Ajallisten ilmiöiden mallinnuksen tarve korostuu mallinnettaessa erilaisia multimediaesityksen lajeja. TIEMPO (Temporal Modeling and Authoring of Interactive Multimedia) on esimerkki tavanomaisista tietokannoista poikkeavasta ajallisten ilmiöiden mallinnussovelluksesta. Se poikkeaa toteutukseltaan ja aika-algebraltaan huomattavasti muista temporaalitietokantoihin liittyvistä sovelluksista: multimediatiedon mallintaminen vaatii multimedian ominaisuuksien takia esimerkiksi ajan nopeuden käsitteitä. TIEMPOn kehittäjät ovat käyttäneet hyväkseen esimerkiksi Allenin esittelemiä aikajaksojen suhteita koskevia havaintoja. [Wahl *et al*., 1995]

5. Yhteenveto

Lähes kaikki mahdolliset tapahtumat muuttuvat ajassa. On vaikea kuvitella tapahtumia, joilla ei olisi kestoa tai menneisyyttä ja tulevaisuutta. Näin ollen onkin huomionarvoista, ettei tietojärjestelmissä ole huomioitu ajallisten ilmiöiden mallintamista nykyistä paremmin. Temporaalitietokantojen alalla on kuitenkin tehty paljon tutkimusta, ja prototyyppitoteutuksia on olemassa muutamia. Tästä huolimatta laaja-alaisiin, käytetyimpiin yleisiin tietojärjestelmiin ei ole sisällytetty temporaaliominaisuuksia.

Aikaan liittyviä ominaisuuksia ja historiatietoja on tarvittu ja käytetty tietojenkäsittelyn historian alkuajoista asti ilman temporaalitietokantoja. Tutkimuksen edetessä en itsekään täysin vakuuttunut, ovatko temporaalitietokannat välttämättömiä, sillä SQL-92-tietokantojen puuttuvista ominaisuuksista huolimatta aikaominaisuuksia on mallinnettu onnistuneesti, vaikka ongelmiakin on esiintynyt. Ongelmaksi on koettu useimmiten puutteista johtuva toteutuksen monimutkaisuus, mikä on aiheuttanut sen, että puutteellisia ominaisuuksia on jouduttu kiertämään sovelluksen kehittäjän omilla keksinnöillä.

Mikäli tulevaan SQL4-standardiin saadaan temporaaliominaisuudet, olisi mielenkiintoista tarkastella, mihin temporaaliominaisuuksia käytettäisiin ja miten kaupalliset sovellukset muuttuisivat näiden ominaisuuksien myötä.

Lähdeluettelo

[Allen, 1983] James F. Allen, Maintaining knowledge about temporal intervals, *Communications of the ACM* **26,** 11 (1983) 832-843.

[Dreyer *et al.* 1994] Werner Dreyer, Angelika Kotz Dittrich and Duri Schmidt, Research perspectives for time series management systems, *SIGMOD Record* **23**, 1 (1994) 10-15.

[Dyreson and Snodgrass, 1998] Curtis E. Dyreson and Richard T. Snodgrass, Supporting valid-time indeterminacy, *ACM Transactions on Database Systems* **23**, 1 (1998) 1-57.

[Elmasri *et al.*, 1993] Ramez Elmasri, Vram Kouramajian and Shian Fernando, Temporal database modeling: an object-oriented approach, *Conference on Information and Knowledge Management, Proceedings of the Second International Conference on Information and Knowledge Management* (1993) 574-585.

[Finger and McBrien, 1996] Marcelo Finger and Peter McBrien, On the semantics of 'current-time' in temporal databases, *In Proceedings of the 11th Brazilian Symposium on Databases* (1996) 324-337.

[Goralwalla *et al.*, 2001] Iqbal A. Goralwalla, Yuri Leontiev, M. Tamer Özsu and Duane Szafron, Temporal granularity: Completing the puzzle, *Journal of Intelligent Information Systems* **16**, 1 (2001) 41-63.

[Hayes, 1995] P. Hayes, A catalog of temporal theories. University of Illinois, Technical Report **UIUC-BI-AI-96-01,** 1995 1-60. Also available as *http://www.ihmc.us/users/phayes/TimeCatalog1.ps, http://www.ihmc.us/users/phayes/TimeCatalog2.ps*

[Kim et al., 2002] Dong Ho Kim, Keun Ho Ryu and Chee Hang Park, Design and implementation of spatiotemporal database query processing system, *The Journal of Systems and Software* **60** (2002), 37-49.

[Yang *et al.*, 2000] Jun Yang, Huacheng C. Ying and Jennifer Widom, TIP: a temporal extension to Informix, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (2000), 596-601.

[Snodgrass, 2003] Richard Snodgrass, Biography of Richard T. Snodgrass, *http://www.cs.arizona.edu/people/rts/* [14.12.2003]

[Snodgrass and Ahn, 1985] Richard Snodgrass and Ilsoo Ahn, A taxonomy of time databases, *Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data* (1985), 236-246.

[SQL] Stephan Ziemer, An Essay about SQL, *http://www.sts.tu-harburg.de/~st.ziemer/sql.pdf* [14.12.2003]

[TimeConsult, 1999] TimeConsult, *http://www.timeconsult.com/* [14.12.2003]

[Wahl et al., 1995] Thomas Wahl, Stefan Wirag and Kurt Rothermel, TIEMPO:

temporal modeling and authoring of interactive multimedia, *International Conference on Multimedia Computing and Systems* (1995), 274-277.

[Wolski *et al.* 1999] Antoni Wolski, Johannes Arminen ja Antti Pesonen, Aikasarjatietomalli mittaustietojen hallintaan, *Systeemityö* **2** (1999), 10-11.

Tietojärjestelmäprojektiin valmistautuminen yrityksen ollessa osa konsernia

Kari Kataja

Tiivistelmä

Tietojärjestelmäprojekti on aina suuri haaste yritykselle. Niinpä yrityksen kannattaakin valmistautua tietojärjestelmäprojektiin huolella. Tässä työssä tarkastellaan tietojärjestelmäprojektiin valmistautumista yrityksen ollessa osa konsernia. Työssä pohdittaan mm. konsernirakenteen, johtamisen, organisaation, tavoitteiden ja strategian merkitystä projektin onnistumisen ja riskien kannalta.

Avainsanat ja -sanonnat: Tietojärjestelmäprojekti, tietojärjestelmä
CR-luokat: H.0

1. Johdanto

Yhä useamman yrityksen liiketoiminta on maailmanlaajuista. Raaka-aineita ostetaan eripuolilta maailmaa. Vastaavasti myös valmiita lopputuotteita kuljetetaan asiakkaille ympäri maapallon.

Pärjätäkseen kovassa kansainvälisessä kilpailussa, yritykset yhdistyvät yhä suuremmiksi kansainvälisiksi konserneiksi. Konsernit ostavat uusia yrityksiä ja toisaalta myyvät ydinliiketoimintaansa sopimattomia osia. Joitakin toimintoja keskitetään, toisia taas saatetaan ulkoistaa.

Kilpailun koventuessa myös tietojärjestelmille asetetut vaatimukset kasvavat. Isossa konsernissa keskushallinnolla saattaa olla varsin merkittävä rooli tietojärjestelmistä päätettäessä. Esimerkiksi kaikki konsernin toiminnanohjausjärjestelmäasennukset saatetaan keskittää yhteen tietokonekeskukseen.

Tässä tutkielmassa keskitytään tutkimaan niitä seikkoja, joita kansainväliseen konserniin kuuluvan yrityksen tulisi ottaa huomioon laajaan tietojärjestelmäprojektiin (kuten toiminnaohjausjärjestelmäprojekti) valmistautuessa. Tutkimusmenetelmänä työssä käytetään kirjallisuusanalyysiä.

2. Tietojärjestelmäprojektin lähtökohdat

2.1. Konsernirakenne

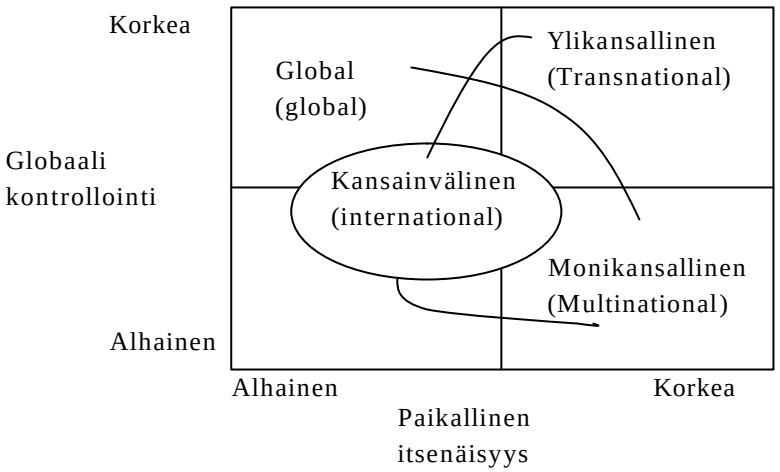
Globaalin kontrolloinnin ja paikallisen itsenäisyyden perusteella kansainvälinen yritys voidaan luokitella neljään ryhmään. Yritys voi olla globaali (global), monikansallinen (multinational), kansainvälinen (international) tai ylikansallinen (kansallisuuksien välinen, transnational). Taulukossa 1 on koottuna näiden organisointivaihtoehtojen ominaisuuksia.

Taulukko 1. Monikansallisen, globaalin, kansainvälisen ja ylikansallisen organisaation vertailua. [Bartlett and Ghoshal 1989]

	Monikansallinen	Globaali	Kansainvälinen	Ylikansallinen
Varat ja kyvykkyys	Hajautettu ja maakohtainen omavaraisuus	Keskitetty ja globaalisti mitoitettu	Ydinosaaminen keskitetty, muut hajautettu	Hajallaan, vastavuoroinen, erikoistuneet
Mantereiden välisten operaation rooli	Paikallisien mahdollisuuksien tunnustelu ja hyödyntäminen	Emoyhtiön strategioiden implementointi	Emo yhtiön kompetenssien sovitus ja vaikutus	Maayksiköiden erilaistunut vaikutus maailmanlaajuisiin integroituihin operaatioihin
Osaamisen kehittäminen ja levittäminen	Jokainen yksikkö kehittää ja säilyttää osaamisensa	Osaaminen kehitetään ja säilytetään keskitetysti	Osaaminen kehitetään keskitetysti ja siirretään muihin yksiköihin	Osaaminen kehitetään yhdessä ja jaetaan maailmanlaajuisesti

Monikansallisessa yrityksessä paikallisilla yksiköillä on siis paljon päätäntävaltaa ja globaali kontrollointi on vähäistä. Globaali konserni vastaavasti tarkoittaa sitä, että konsernissa keskitettyä ohjausta käytetään paljon ja paikallinen itsenäisyys on vähäistä. Kansainvälisessä (international) konsernissa yritys on väliaikaisessa tilassa kun pyritään paikalliseen ja globaalin tasapainoon [Lehmann 2000].

Ylikansallisissa yrityksissä on tiukka globaali kontrolli samalla kun voimallisesti edistetään paikallista itsenäisyyttä. Kysessä siis on eräänlainen “think global and act local” –tapaus. Ylikansallista rakennetta pidetään monissa tapauksissa optimaalisena mallina. Kuvassa 1 hahmotellaan vielä tilannetta. [Lehmann 2000]



Kuva 2. Globaalin kontrollointi ja paikallisen itsenäisyys. [Lehmann 2000, s. 372]

Kuvassa 1 on mukana yksi varsin merkittävä viesti: konsemi saattaa tänään olla monikansallinen, mutta kenties tulevaisuudessa globaali kontrollointi kasvaa konsernin muuttuessa ylikansalliseksi. Jos tarkastellaan esimerkiksi nopeasti kasvaneita konserneita, konserni ei välttämättä ole ”ehtinyt” rakentamaan globaalia kontrollointia. Siis vasta kasvuvaiheen tasaantuessa pystytään keskittymään globaalin kontrolloinnin vaatimien järjestelmien ja organisaation luomiseen.

Tietojärjestelmäprojektin onnistumisen kannalta on tärkeätä olla selkeä käsitys siitä “mikä yritys on”. Erityisesti kansainvälisissä konserneissa tulee yrityksen johdolla olla selkeä näkemys siitä, miten integroitua yritystoiminta on nyt ja tulevaisuudessa. [Davenport 2000]

2.2. Federalistinen lähestymistapa

Federalistisessa lähestymistavassa ideana on, että yrityksen keskushallinto kontrolloi osaa prosesseista ja toimintaperiaatteista. Paikalliset yksiköt puolestaan vastaavat lopusta toiminnasta. Jos keskushallinto määrittelee suurimman osan asioista, kyseessä on eräänlainen yksinvaltius, johon on lisätty federalistisia piirteitä. [Davenport 2000]

Tietojärjestelmien kohdalla federalismi tapauksessa kenties haastavinta on aluksi päättää, mitkä asiat kuuluvat keskushallinnon vastuulle. Tässä kohtaa on syytä ottaa huomioon, että kaikki tietojärjestelmätoimittajat eivät suinkaan tue federalistista lähtökohtaa. Osa tietojärjestelmistä mahdollistaa kuitenkin

automaattisen konsernitason tietojen aggregoinnin, toisissa järjestelmissä tämä joudutaan tekemään manuaalisesti. [Davenport 2000]

Käytännössä konsernin pääkonttoriin asennetaan yksi toiminnanohjausjärjestelmä, jossa konsernin yhteiset asiat (kuten talousasiat) hoidetaan. Jokaisella yrityksen yksiköllä on tällöin oma tietojärjestelmä, josta pitää löytyä konsernin pääkonttorin tietojärjestelmän vaatimat tiedot. Välttämättä yksiköillä ei tarvitse olla sama tietojärjestelmä, mutta tarvittavien tietojen kerääminen muista järjestelmistä on yleensä vaikeampaa. [Davenport 2000]

Konsernin eri yksiköillä voi siis federalistisessa lähestymistavassa olla paljon omaa yksikkökohtaista tietoa tietojärjestelmässään. Näin asian laita onkin usein isoissa konserneissa, sillä eri yksiköillä voi olla eri asiakkaita, toimintaprosesseja ja jopa johtamistapoja. Tämä onkin federalistisen lähestymistavan hyvä puoli. [Davenport 2000]

Federalistisessa johtamisessa on myös huonot puolensa. Yhteisten tietojen ylläpito voi olla vaikeata ja aikaa vievää. Lisäksi tämän prosessin pitäisi olla jatkuva. Organisaatiosta saattaa löytyä myös vastustajia, sillä konsernin vaatimien tietojen ylläpito voidaan nähdä ylimääräisenä työnä. Myös tietojärjestelmän implementointi federalistiseen ympäristöön on haastava operaatio. [Davenport 2000]

Federalismi lyhyesti sanottuna tarkoittaa suurempia liiketoiminnan joustavuutta teknisen monimutkaisuuden kustannuksella. Federalistista tietojärjestelmän käyttöä tulisi kuitenkin harkita vain silloin, kun on ehdottoman tärkeätä, että eri puolilla konsernia asiat on voitavat tehdä eritavoilla. Useimpien yritysten kohdalla asian laita ei kuitenkaan ole näin. [Davenport 2000]

Tietojärjestelmissä on usein vain rajoitetut mahdollisuudet muokata toimintatapoja. Jotta tietojärjestelmä toimisi järkevästi, tulee käyttäjien kurinalaisesti noudattaa sovittuja toimintatapoja. Tietojärjestelmäprojektin alussa on tärkeätä tehdä organisaatiossa selväksi, etteivät osastot tai yksittäiset henkilöt voi suunnitella omia järjestelmiä. Mikäli näin ei toimita, saattaa organisaatio pettyä pahasti, kun kaikkia toivottuja ominaisuuksia uudesta järjestelmästä ei löydykään. [Davenport 2000]

Toiminnanohjausjärjestelmä mahdollistaa käyttäjille hyvin laajan tiedonsaannin. Tämän lisäksi tiedot ovat paremmin ajantasalla ja tarkempia. Tietoihin myös pääsee käsiksi laajempi joukko työntekijöitä, mikäli niin halutaan. [Davenport 2000]

Kehittynyt tietojärjestelmä helpottaa yrityksen sisäisen tehokkuuden arviointia. Integroidussa järjestelmässä eri toimintojen vastuuhenkilöt eivät enää pääse manipuloimaan oman osastonsa tehokkuuslukuja. Tietojärjestelmä

myös yhdenmukaistaa tunnuslukujen laskennan. Tiedot tulevat siis paremmin vertailukelpoisiksi. [Davenport 2000]

2.3. Strategia

Ennen tietojärjestelmäprojektin aloittamista tulee sekä konsernin että yrityksen strategia olla selvä. Siis yrityksen johdon tulee tiedostaa, mikä on yrityksen ydinliiketoiminta-alue, miten asiakkaalle luodaan lisäarvoa ja miten yritys erottuu kilpailijoista yms. [Davenport 2000]

Yrityksen strategian tulee olla selvä, jotta tietojärjestelmäprojektilla pystytään tukemaan sitä. Jos yrityksen strategiana esimerkiksi on asiakasläheisyys, tietojärjestelmäprojektissa voidaan panostaa enemmän markkinointiin, myyntiin ja asiakaspalveluun. [Davenport 2000]

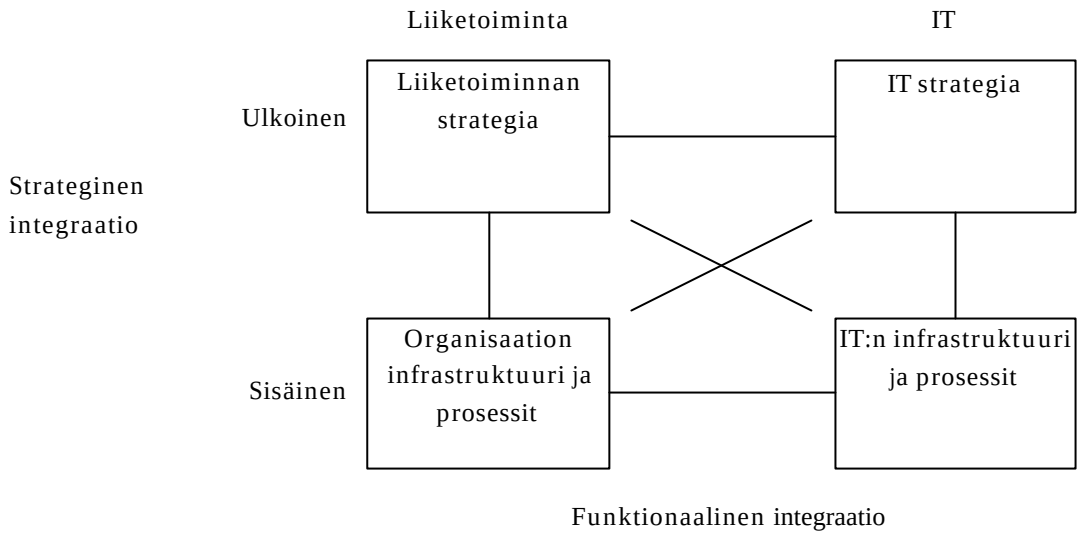
Tietojärjestelmäprojektin perustana tulisi olla tietohallinnon tietohallintostrategia. Tietohallintostrategiasta tulisi löytyä ainakin seuraavat osa-alueet [Kettunen 2002, s. 49 - 50]:

- Tietohallinnon nykytilanne
- Yrityksen tavoitteet ja tulevaisuuden muutokset
- Tietohallinnon tarpeet ja visiot
- Riskianalyysi
- Toimintasuunnitelma tarpeiden kattamisesta
- Toimenpiteiden resurssit
- Varautuminen poikkeustapauksiin.

Tietohallintostrategia ei sinällään ole “oma” strategia vaan osa yrityksen kokonaistrategiaa. Jatkossa tarkastellaankin vielä tietohallinnon ja organisaation strategian yhteensovittamista.

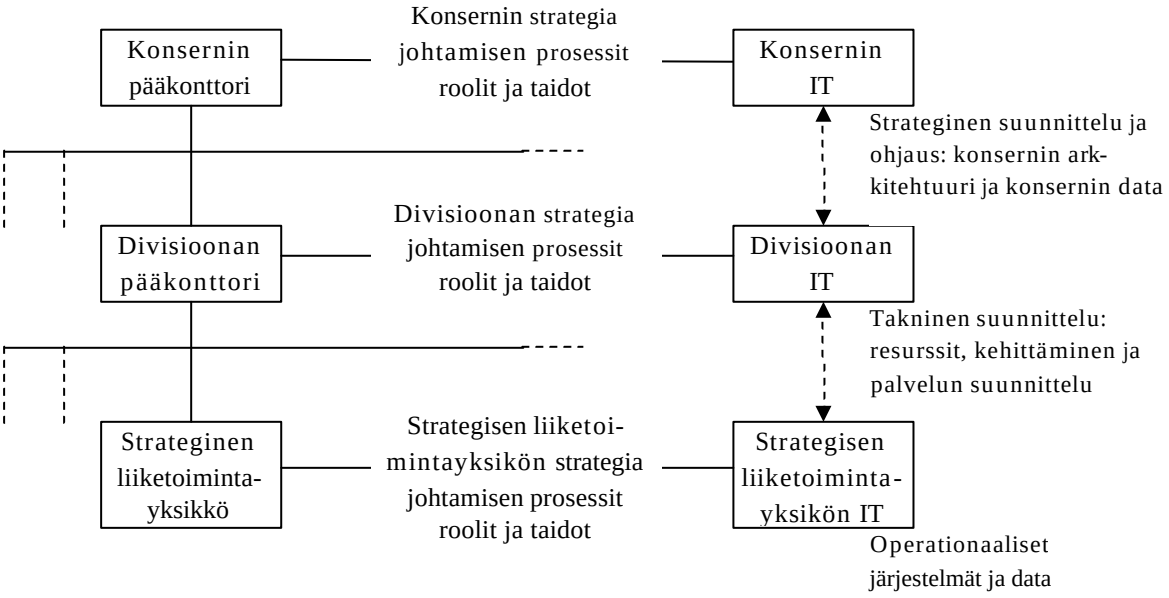
2.4. Tietotekniikan ja organisaation strateginen niveltäminen

Tietotekniikan ja organisaation strateginen niveltäminen (Strategic Alignment) tarkoittaa mallia, jonka tarkoituksena on varmistaa, että liiketoiminnan strategia, IT-strategia, organisationaalinen infrastruktuuri ja prosessit sekä IT-infrastruktuuri ja prosessit tukevat toinen toistaan. Tämä tilanne on esitettynä kuvassa 2. [Sauer et al. 1997]



Kuva 3. Strategisen niveltämisen malli. [Sauer et al. 1997, s. 59]

Strateginen niveltäminen on sovellettavissa myös konsernitapauksessa. Kuvassa 3 on strateginen niveltäminen sovitettu konsernitapaukseen. [Sauer et al. 1997, s. 8]



Kuva 4. Tietotekniikan ja organisaation strateginen niveltäminen konsernirakenteessa [Sauer et al. 1997, s. 8]

Tietotekniikan ja organisaation yhteen niveltämisen mallissa on omat heikkoutensa. Erityisesti konfikteja saattaa ilmetä, kun yrityksen kansainvälistymisstrategia perustuu yritysostoihin. [Sauer et al. 1997]

Strategiseen niveltämiseen liittyy myös sekaantumisongelma. Tämä esiintyy erityisesti silloin, kun konsernista löytyy eri kasvuvaiheessa olevia osia. Eri puolilla maailmaa esimerkiksi tietoteknisessä kehityksessä ollaan eri vaiheissa. [Sauer et al. 1997]

Yrityksen strategian toteuttaminen saattaa kuitenkin vaikeuttaa tietojärjestelmäprojektia. Järjestelmää saatetaan joutua muuttamaan tai siihen voidaan joutua liittämään kolmannen osapuolen ratkaisuja. Tällöin muutosten ja liittymien hallinta lisää työmäärää ja monimutkaistaa järjestelmän hallintaa. [Davenport 2000]

2.5. Tietojärjestelmän implementoinnin vaikutukset muihin strategioihin

Kuten kuvassa 2 esitettiin, IT-strategian pitää tukea yrityksen muita strategioita ja toimintaa ylipäätään. Usein yritykset kuitenkin jättävät liian pienelle huomiolle tietojärjestelmän vaikutukset liiketoimintastrategiaan, organisaation rakenteeseen ja toimintatapoihin. [Davenport 2000]

Kalliilla tietojärjestelmäprojektilla on vaikutuksensa myös rahoitusstrategiaan. Erityisesti alhaisen kustannustrategian yrityksissä saattaa olla vaikeata hyväksyä tietojärjestelmän kustannuksia, vaikka pidemmällä aikavälillä tarkasteltaessa projektin tulos olisikin positiivinen. Tietojärjestelmä saattaa tehostaa esimerkiksi varastojen hallintaa, jolloin sitoutunut pääoma pienenee ja yrityksen rahoitustilanne paranee. [Davenport 2000]

Esimerkiksin uudella toiminnanohjausjärjestelmällä (ERP-järjestelmä) saattaa olla merkittävä vaikutus yrityksen operatiiviseen strategiaan, koska järjestelmäprojektilla on vaikutuksia avainprosesseihin. Jotkut strategiat ovat vaikeita (tai suorastaan mahdottomia) toteuttaa ilman soveltuvaa tietojärjestelmää. Esimerkikkinä tästä on Lean–valmistus. [Davenport 2000]

Uuden tietojärjestelmän käyttöönotto muuttaa organisaatiota. Usein tietojärjestelmän käyttöönoton myötä tarvitaan vähemmän työntekijöitä, mutta työnte-kijöiden pitää olla koulutetumpia. [Davenport 2000]

Tietojärjestelmän implementoinnissa voidaan käyttää apuna konsultteja. Kuitenkin lopulta järjestelmän käyttämiseen tarvittava tietotaito pitää löytyä omasta organisaatiosta. Koulutuksen tarve usein aliarvioidaan. [Davenport 2000]

Yksi tyypillinen ERP-järjestelmäprojektin tavoite on tiettyjen toimintojen (kuten taloushallinto, osto yms.) keskittäminen. Näin meneteltäessä tavoitel-

laan suuruuden ekonomian perusteella tehokkuutta ja sitä kautta kustannussäästöjä. [Davenport 2000]

Tietojärjestelmän implementointi ei välttämättä kuitenkaan tuo organisaation toimintaan tehokkuutta: itse asiassa organisaatio saattaa usein jopa kasvaa tietojärjestelmän implementointiongelmien johdosta. Tämä voi johtua esimerkiksi siitä, että organisaation merkitystä ei ymmärretä tai muutoksia ei viedä organisaation osalta loppuun [Davenport 2000].

2.6. Kilpailuedun saavuttaminen tietojärjestelmän avulla

Jos kaikilla yrityksillä jo on tietojärjestelmä käytössä, pelkällä järjestelmän implementoinnilla ei vielä saada kilpailuetua. Tärkeätä on se, miten järjestelmä implementoidaan ja miten hyvin se saadaan sovitettua yrityksen toimintaan. Tässä avain asemassa on järjestelmän huolellinen konfigurointi ja mahdollisesti tarvittavien soveltuvien lisäosien käyttöönottaminen. [Davenport 2000]

Kilpailuetu ei tule itse järjestelmästä vaan siitä, että asiat tehdään paremmin kuin kilpailijat. Ko. toimialalla järjestelmän ensimmäisellä implementoijalla saattaa olla mahdollisuus saavuttaa huomattavaa kilpailuetua. [Davenport 2000]

Tulevaisuuden kilpailuetujen hahmottaminen helpottaa tietojärjestelmäprojektin suunnittelua. Seuraavassa on esitettynä joukko kysymyksiä, joilla voidaan tarkastella tietojärjestelmäprojektin vaikutuksia kilpailuetuun [Davenport 2000]:

- Mikä on yrityksen tämänhetkinen kilpailuetu? Miten tietojärjestelmän käyttöönotto vaikuttaa niihin?
- Mahdollistaako tietojärjestelmä uusia strategisia mahdollisuuksia?
- Miten tietojärjestelmäprojektin kustannukset vaikuttavat yrityksen kustannustasoon?
- Mitä tietojärjestelmiä kilpailijat implementoivat? Muuttuvatko kilpailijoiden vahvuudet ja heikkoudet tästä johtuen? Ketkä todennäköisesti implementoivat hyvin / huonosti järjestelmät?
- Onko tarpeen sähköisesti liittyä toimialan yrityksiin?
- Voiko joku osa yrityksen kilpailuedusta vahingoittua, mikäli käyttöön otetaan jokin ”standardi” tapa käsitellä tietoa / prosesseja?
- Häiritseekö tietojärjestemäprojekti yritystä keskittymästä liiketoimintaan?

Edellisien kysymyksien perusteella saatetaan päätyä jopa ratkaisuun, jossa koko tietojärjestelmäprojekti päätetään jättää tekemättä. Esimerkiksi kustannustehokkuuteen pyrittäessä saatetaan päätyä tähän ratkaisuun tai sitten vaihtoehtoisesti tietojärjestelmä implementoidaan halvalla.

Konsernista saattaa löytyä apua järjestelmän konfiguroinnissa. Iso konserni on jopa saattanut kehittää oman esikonfiguroidun ympäristön. Näin pyritään nopeampaan (ja halvempaan) tietojärjestelmän implementointiin. Esikonfiguroinnin avulla myös pyritään yhtenäistämään konsernin toimintamalleja. Kyseessä on siis samalla eräänlainen ”Best Practise” –tiedon levittäminen organisaation.

3. Projektin tavoitteet

3.1. Tuloksen merkitys

Tietojärjestelmäprojekteissa lähtökohtana pitää olla vahva tulossuuntautuneisuus. Tuloksilla voidaan tässä yhteydessä tarkoittaa esimerkiksi rahoituksellisia parannuksia, prosessin parantumista, varastojen pienentymistä tai virheiden vähenemistä yms. Tuloksella pitää olla arvoa joko liiketoiminnalle tai asiakkaille. [Davenport 2000]

Pääsääntöisesti tietojärjestelmäprojektin lähtökohtana ei siis pidä olla pelkkä tietojärjestelmän tekninen käyttöönotto. Ainoastaan silloin, jos uusi tietojärjestelmä on edellytys yrityksen toiminnan jatkumiselle, voidaan lähtökohdaksi ottaa pelkkä järjestelmän vaihtaminen. Esimerkkeinä tällaisista tilanteista ovat olleet vuosituhannen vaihde (y2k-ongelmat) ja euron käyttöönotto. [Davenport 2000]

Myös silloin kun osa yrityksestä myydään toiseen konserniin saattaa olla tarkoituksenmukaista keskittyä vain järjestelmän käyttöönottoon. Tilannehan saattaa olla sellainen, että ostettu yksikkö joutuu luopumaan siirtymäajan jälkeen vanhojen järjestelmiensä käyttämisestä, koska ne jäävät edelleen palvelemaan entistä omistajaa.

Mikäli tietojärjestelmäprojekti on yrityksen toiminnan kannalta välttämätön, tulee itse implementointi tehdä niin nopeasti ja halvalla kuin mahdollista. Tällöin ei siis keskitytä järjestelmän muokkaamiseen eikä optimointiin vaan pyrkimyksenä on mahdollisimman yksinkertainen implementointi. [Davenport 2000]

Usein suuressa osassa organisaatiota tietojärjestelmäprojekti ei kuitenkaan ole yrityksen toiminnan jatkumisen kannalta välttämätöntä. Tällöin pitääkin liiketoiminnasta vastaavien henkilöiden etukäteen määritellä, mitä tuloksia tietojärjestelmäprojektilla tulee saavuttaa. Tietojärjestelmäprojektia voidaankin käsitellä kuten mitä tahansa yrityksen merkittävää investointia. [Davenport 2000]

Tietojärjestelmäprojektin arvioinnissa voidaan käyttää hyväksi business casea. Hyötyjen ja kustannusten määrittämisen lisäksi oikein tehty business case helpottaa organisaatiota ymmärtämään tietojärjestelmäprojektin tavoitteita. Businessa casea voidaan myös kehittää sitä mukaa kun tietämys implementoitavasta järjestelmästä kasvaa. [Davenport 2000]

Kenties yleisin korkean tason organisaatio-ongelma tietojärjestelmäprojekteissa on järjestelmien korkean tason integrointi. Usein ajatellaan, että saman järjestelmän käyttöönotto organisaation eri osissa tuo automaattisesti mukanaan myös korkean tason integroinnin. Kuitenkin todellisuudessa eri organisaation osissa järjestelmä voidaan ottaa käyttöön hyvin eri tavalla. Mikäli korkean tason prosessi ja informaation integrointi on järjestelmäprojektin tavoitteena, on korkean tason organisaatiomuutos on onnistumisen edellytys. [Davenport 2000]

Konserneissa kannattaa hyödyntää mahdollisuutta perehtyä siihen, miten konsernin muut yksiköt ovat tietojärjestelmäimplementoinnin hoitaneet. Tätä kautta on mahdollista saada arvokasta tietoa projektin kustannuksien, hyötyjen, riskien, potentiaalisten ongelmakohtien yms. ennakoimiseksi.

3.2. Tavoitteiden määritys

Liiketoiminnasta vastaavat henkilöt voivat tarvita koulutusta, jotta tietojärjestelmäprojektin tavoitteet saataisiin asetettua kohdalleen. Johtavassa asemassa olevat henkilöt voivat myös asettaa kriittisiä menestystekijöitä, joiden avulla tietojärjestelmän implementoinnissa pysytään keskittymään olennaisiin asioihin. [Davenport 2000]

Vaihtoehtoisena lähestymistapana voidaan myös lähteä liikkeelle esimerkiksi asiakkaan liiketoiminnan vaatimuksista. Myös kilpailijoiden toimien analysoinnista saadaan tarvittavaa lisätietoa tavoitteiden asettamiseksi. [Davenport 2000]

Tavoitteiden saavuttamisen valvominen pitäisi kuulua jokaiselle yrityksen ylimmän johdon henkilölle. Kuitenkin tavoitteiden saavuttamisessa vastuuta ei pidä liiaksi jakaa, koska tällöin tietojärjestelmäprojektin aiheuttamissa busineksen muutoksien hallinnassa tulee helposti ongelmia. Tilanteesta riippuen tietojärjestelmäprojektin tuloksista vastuussa voi olla esimerkiksi yrityksen hallintojohtaja tai jopa toimitusjohtaja. [Davenport 2000]

Tietojärjestelmäprojektin tavoitteet ja hyödyt tulee kertoa yrityksen työntekijöiden lisäksi esimerkiksi osakkeenomistajille ja asiakkaille. Mikäli tietojärjestelmäprojektista ei ole hyötyä osakkeen omistajille, tulisikin harkita, tehdäänkö projektia laisinkaan. [Davenport 2000]

Myös konsernijohdolla saattaa olla tavoitteita tietojärjestelmäprojektille. Esimerkiksi konsernijohto saattaa viestittää, että jokin tietty tietojärjestelmäprojekti pitäisi saada valmiiksi vaikkapa puolessa vuodessa.

4. Konsernistandardit

Konsernin keskushallinto on yleensä tehnyt linjaukset siitä, millaisia tietojärjestelmäratkaisuita konsernin yritykset käyttävät. Seuraavassa tarkastellaan syitä siihen, miksi näin toimitaan.

4.1. Lisenssin hinta ja konsernin neuvotteluvoima

Jos tarkastellaan lisenssien kappalehintaa, usein hinnoittelu on sellainen, että lisenssin yksikköhinta laskee, jos lisenssejä ostetaan suurempi määrä. Tällöin iso konserni saa lisenssinsä halvemmalla, kun ostot keskitetään.

Toisaalta, sopimusneuvotteluissa isolla konsernilla saattaa olla enemmän mahdollisuuksia sopia niin hinnoista kuin muistakin ehdosta. Myös neuvotteluihin kuluvan työajan kannalta linsenssineuvottelut kannattaisi keskittää. Konserni saattaa siis esimerkiksi hankkia koko konsernia koskevan konsernikohtaisen lisenssin. Lisenssien laskemisessa on löydettävissä esimerkiksi kolme seuraavaa päätapausta [Takki 2002, s. 164, 174]:

- Laitekohtainen lisenssi (designated equipment)
- Henkilökohtainen (named users)
- Käyttäjien kokonaismäärä (users).

Käyttäjien kokonaismäärä voidaan laskea joko absoluuttisien tai yhtäaikaisien käyttäjien (concurrent users) perusteella. Yhtäaikaiset käyttäjien laskeminen voi puolestaan perustua tietyssä paikassa sijaitsevien nimettyjen käyttäjien (named concurrent users) laskemiseen tai missä tahansa sijaitsevien käyttäjien (floating concurrent users) laskemiseen. [Takki 2002]

Konserni saattaa saada merkittäviä säästöjä, mikäli tietojärjestelmän lisenssipolitiikkana on missä tahansa sijaitseviin yhtäaikaisiin käyttäjiin perustuva yhtäaikaisten käyttäjien laskeminen. Tällöin voidaan hyödyntää aikavyöhykkeistä johtuvia aika-eroja: kun Euroopassa nukutaan, lisenssi voi olla käytössä vaikkapa Kiinassa tai Amerikassa. Toisin sanoen toisella yksikkö voi kenties ottaa tietojärjetelmän käyttöön ilman että konsernin lisenssikustannukset nousisivat. Tämä ei kuitenkaan tarkoita sitä, että yksikkö saisi lisenssit konsernilta ilmaiseksi, sillä konsernissa todennäköisesti on sovittuna joku sisäinen kustannuksien laskutustapa.

Lisenssien hinta saattaa olla myös tietojärjestelmän ylläpitomaksun perusteen. Konserni siis saattaa myös säästää vuotuisissa ylläpitomaksuissa käyttämällä konsernisopimuksia.

Tämän lisäksi useiden toisistaan erillisten järjestelmien ylläpito on raskasta. Mitä enemmän erityyppisiä järjestelmiä on integroitu yhteen, sitä haastavampaa on kokonaisuuden hallinta. Mahdollisten virheiden etsiminen ja korjaaminen vaikeutuu huomattavasti, jos järjestelmien lukumäärä kasvaa. [Davenport 2000]

4.2. Globaali numerointi

Käyttämällä yhtä yhteistä tietokantaa voidaan varmistua siitä, että sama tieto tarkoittaa samaa asiaa eripuolilla organisaatiota riippumatta käytetystä järjestelmästä. Tällöin esimerkiksi asiakasnumerot ovat koko konsernissa samat. [Davenport 2000].

Materiaalinimikkeiden tunnisteiden yhtenäistämisessä voidaan käyttää aluksi rinnakkain konserninlaajuisia tunnisteita ja paikallisia tunnisteita. Näin vältytään esimerkiksi työohjeiden välittömältä päivitystarpeelta: lähtökohtana voidaan pitää esimerkiksi sitä, että uudet tuotteet ohjeistettaisiin käyttämällä uusia tunnisteita. Yhteisten tunnisteiden käyttäminen mahdollistaa ostojen keskittämisen ja helpottaa eri yksiköiden välistä kommunikointia. [Peltonen et al. 2002]

Globaaleissa numeroinneissa on myös omat heikkoutensa. Aina, kun paikallisesti tarvitaan esimerkiksi uudelle asiakkaalle asiakasnumero, pitää kyseinen numero pyytää keskusorganisaatiolta. Järjestelmä voi tietenkin toimia automaattisesti esimerkiksi intranetissä. Mikäli numeroinnissa on jotakin logiikkaa mukana, esimerkkinä olevan uuden asiakkaan asiakasnumero valinnan todennäköisesti suorittaa erikseen nimetty vastuuhenkilö. Globaalissa ympäristössä tulee ottaa huomioon aikavyöhykkeiden vaikutus. Ei liene sopivaa, että esimerkiksi materiaalinimikkeen koodia joutuisi odottamaan aikaerosta johtuen esimerkiksi vuorokauden.

Globaalin numeroinnin käyttäminen sinällään ei välttämättä pakota käyttämään koko konsernissa samoja tietojärjestelmiä. Kuitenkin työmäärä todennäisesti helpottuu, jos koko konsernissa on sama järjestelmä käytössä.

5. Tietojärjestelmän toimintaan vaikuttavia tekijöitä

Tietojärjestelmiltä toivotaan joustavuutta, sillä kaikkia yrityksen muutoksia on mahdotonta ennakoida. Laaja-alaisen toiminnanohjausjärjestelmän (ERP) implementoinnin jälkeen yrityksen toimintaa ja organisaatiota saattaa olla

vaikeampaa muuttaa. Tämä johtuu siitä, että tietojärjestelmä linkkaa yhteen organisaation ja prosessit. [Davenport 2000]

Toiminnanohjausjärjestelmiä onkin usein moitittu joustamattomiksi. Toisaalta, jos yhdellä ERP-järjestelmällä korvataan useita erillisiä järjestelmiä, muutoksien tekeminen saattaa helpottua, kun muutokset tehdään vain yhteen järjestelmään. ERP-järjestelmissä monia asioita voidaan muuttaa konfigurointia muuttamalla. Tällaisten muutoksien tekeminen on yleensä suhteellisen helppoa. [Davenport 2000]

ERP-järjestelmän mukanaan tuoma toiminnan standardointi voi johtaa myös joustavuuden lisääntymiseen. Liiketoiminnan muutosten käyttöönotto voi olla helpompaa ympäristössä, jossa on yksi maailmanlaajuisesti käytössä oleva IT-alusta. [Davenport 2000]

Jos kyseessä on valmistustoimintaa tekevä konserni, mahdollisten tuotekohtaisten ohjelmistojen siirtäminen tehtaasta toiseen helpottuu, mikäli molemmissa tehtaissa on käytössä samat perusjärjestelmät. Tuotantoa siirrettäessä eri järjestelmiin ei tarvitse räätälöidä samoja liityntöjä uudelleen. Joustavuus siis itse asiassa lisääntyy ERP:n tuoman standardoinnin myötä.

5.1. Raportit

Monikansallisessa konsernissa saattaa ainoa eri yksiköitä yhdistävä tekijä olla talousosaston käyttämä raportointivaluutta. Mikäli konsernin eri yksiköt käyttävät erilaisia järjestelmiä, tietojen konsolidointi saattaa muodostua todella suureksi ja paljon käsityötä vaativaksi tehtäväksi. [Davenport 2000]

Talousosaston vaatimat raportit saadaan aikaan helposti, mikäli kaikki yrityksen yksiköt käyttävät samaa järjestelmää. Tällaisessa integroidussa ratkaisussa joudutaan kuitenkin paljon asioita koordinoimaan eri tehtaiden välillä. Lisähaasteita tähän luo vielä eri maiden erilaiset lainsäädännöt. [Davenport 2000]

5.2. Brändi ja sähköinen kauppapaikka

Yksi ERP- järjestelmien perusteluista on se, että järjestelmät luovat perustan elektronisen kaupankäynnin käyttöönotolle [Davenport 2000]. Jos tarkastellaan maailmanlaajuisesti operoivaa yritystä, on loogista, että yrityksellä on sama sähköisen kauppapaikan konsepti käytössä eripuolilla maailmaa. Mahdollista myös on, että yrityksellä ylipäätään on vain yksi maailmanlaajuinen sähköinen kauppapaikka.

Yksi yhteinen sähköinen kauppapaikka helpottaa yrityksen brändin hallintaa, sillä tällöin asiakas asioi aina samanlaisella verkkosivustolla. Myös

asiakkaan kannalta tämä on helpompaa, koska kaiken asioinnin voi tehdä yhdessä pisteessä.

Sähköisen kauppapaikkan luotettava toiminta on tärkeätä, sillä mahdolliset ongelmat vaikuttavat helposti konsernin maineeseen ja brändiin. Ei riitä, että käytettävät järjestelmät ovat luotettavia, vaan myös operoinnin ja IT-infrastruktuurin pitää olla hallinnassa. [Robertson and Sribar 2002]

5.3. Konsernin resurssien käyttö

Isolla konsernilla saattaa olla käytössä omia konsultteja, joita voidaan hyödyntää tietojärjestelmäprojekteissa. Joissakin tilanteissa sisäisien konsulttien käyttäminen saattaa olla jopa välttämätöntä. Näin voi olla esimerkiksi silloin, jos konserni on kehittänyt tietojärjestelmään omia lisäpiirteitä.

Tietojärjestelmäprojektiin valmistautuessa tuleekin selvittää konsernin sisäisien konsulttien rooli implementoinnissa. Samoin näiden sisäisien resurssien saatavuus kannattaa tarkistaa ajoissa.

5.4. Asiakkaan vaatimukset

Asiakassuuntautuneessa yritystoiminnassa pitää myös tietojärjestelmäprojekteissa muistaa asiakkaan toiveet ja vaatimukset. Siirryttäessä käyttämään konsernin standardijärjestelmää, pitää erityisesti korkean tietoturvan ympäristössä asiasta keskustella myös asiakkaan kanssa.

Isoissa konserneissa esimerkiksi toiminnanohjausjärjestelmät saatetaan keskittää johonkin tiettyyn paikkaan. Asiakas taas voi vaatia saada auditoida yrityksen toiminnan, mukaan lukien tietojärjestelmät. Tällaisessa tapauksessa asiakas ei välttämättä ole tyytyväinen, mikäli tietojärjestelmä pyöriikin esimerkiksi toisessa maanosassa. Jo tietojärjestelmäprojektia suunniteltaessa tämä tulee ottaa huomioon.

6. Riskienhallinta

Tietojärjestelmäprojekteihin liittyy aina huomattava määrä riskejä. Projektiin liittyvällä riskianalyysillä on tarkoituksena riskien tunnistaminen ja analysointi. Tämän jälkeen valitaan jokaiselle riskille sopivin menettelytapa: riski voidaan yrittää välttää, riskiä voidaan pienentää, riski voidaan siirtää tai riski voidaan ottaa sellaisenaan. [Harju 2003]

6.1. Konserniriski

Mikäli yritys on osa konsernia, tulee riskitarkasteluun sisällyttää myös konserniriskin arviointi. Isossa konsernissa yrityksen sisäisten tapahtumien ennakointi saattaa olla haastava tehtävä.

Konsernin pääkonttori saattaa esimerkiksi havaita, että jokin konsernistandardin mukainen ohjelmaratkaisu ei enää ole kilpailukykyinen. Tällöin voidaan jopa keskeneräiset projektit keskeyttää. Nopeasti muuttuvissa tilanteissa myös tehdyt lupaukset voidaan perua. Konsernin sisäisissä asioissahan ei mitään "sopimussakkoja" voida käyttää.

Joissakin tilanteissa konsernin tekemien päätösten ymmärtäminen paikallisesti saattaa olla hankalaa. Asioita pitäisi pystyä ajattelemaan koko konsernin näkökulmasta: välttämättä se, mikä näyttäisi parhaalta ratkaisulta yksittäisen yksikön kohdalta ei ole optimaalinen ratkaisu tarkasteltaessa pidemmällä aikavälillä koko konsernin tilannetta.

Fuusioitumisissa saattaa tapahtua hyvin dramaattisia muutoksia. Fuusion tuloksena saatetaan esimerkiksi päätyä ratkaisuun, jossa toinen osapuoli ottaa käyttöön toisen osapuolen ERP-järjestelmän. [Davenport 2000].

6.2. Standardoinnin vaikutus tietoturvaan

Mielenkiintoinen kysymys on, kuinka pitkälle tietoturva-asiat pitäisi konsernissa standardoida. Kasvaako riski liian suureksi, mikäli esimerkiksi kaikki yrityksen palomuurit ja virustorjuntaohjelmat ovat samanlaisia? Mikäli virustorjuntaohjelma ei tunnista jotakin virusta, tällöin koko konsemi ehtii mahdollisesti saastua ennenkuin asia havaitaan.

Konsernin keskushallinnon kannalta on erittäin vaikeata varmistua vaikkapa kaikkien mahdollisien virustorjuntaohjelmien luotettavuudesta. Ongelmana on määritellä "hyväksyttävän tehokas" virusten torjuntaohjelma muuten kuin määrittelemällä mitä ohjelmia käytetään.

Tilannetta voi verrataan esimerkiksi sijoitustoimintaan, jossa hajauttamisella pyritään vähentämään riskiä. Tällöin ei siis menetetä kaikkea, mikäli riski realisoituu. Toisaalta, vaihtoehtona olevaan yhden järjestelmän käyttämiseen voisi siteerata Mark Twainin kuuluisaa lausetta "Put all your eggs in the one basket and -WATCH THAT BASKET". Mikäli siis päädytään standardoimaan tietoturvaan liittyviä asioita, tulee käytettävät standardiratkaisut valita huolella ja seurata koko ajan osa-alueen kehittymistä tarkasti.

7. Yhteenveto

Konsernilla on varsin huomattavia vaikutuksia yksikön tietojärjestelmäprojekteja suunniteltaessa. Erityisesti globaalissa ja ylikansallisessa konsernissa konsernin keskushallinto kontrolloi yksiköiden toimintaa tarkasti.

Konsernin eri yksiköissä saattaa olla vaikeata ymmärtää tai hyväksyä konsernin keskushallinnon päätöksiä ja ohjeita. Välttämättä koko konsernin kan-

nalta paras tilanne ei olekaan optimaalinen yhden yksikön kannalta. Paikallisesti ei välttämättä myöskään ymmärretä tai tiedetä konsernin johdon strategisia linjauksia.

Konsernijohdon päätöksiä ja ohjeita kannattaa toisaalta tulkita kriittisesti. Tarvittaessa konsernin sisäiset päätökset voidaan perua tai muuttaa. Konsernin mukanaolo lisääkin tietojärjestelmäprojektiin yhden merkittävän riskin, konserniriskin.

Konsernin ohjeiden noudattamisella on mahdollista saavuttaa myös merkittäviä etuja tietojärjestelmäprojekteissa. Konsernin ansiosta tietojärjestelmän lisenssikustannukset voivat olla alhaisemmat ja implementoinnissa voidaan kenties käyttää sisäisiä resursseja. Ennen kaikkea konsernin standardiratkaisuja käyttämällä toteutetaan konsernijohdon strategisia suuntauksia. Tällöin myös kommunikointi konsernin keskushallinnon suuntaan on helpompaa.

Viiteluettelo

[Bartlett and Ghoshal 1989] Christopher A. Bartlett and Sumantra Ghostal, *Managing Across Borders, the Transnational Solution*. Harvard Business School Press, 1989.

[Davenport 2000] Thomas H. Davenport, *Mission Critical, realizing the promise of Enterprise Systems.* Harvard Business School Press, 2000.

[Harju 2003] Ansa Harju, *Tietohallinnon kustannus- ja hyötyanalyysi.* Helsingin ammattiikorkeakoulu Stadian julkaisuja. Sarja A: Tutkimukset ja raportit 1. Yliopistopaino, Helsinki, 2003.

[Kettunen 2002] Sami Kettunen, *Tietojärjestelmän ostaminen – käytännön opas yrityksille.* WS Bookwell Oy, Porvoo, 2002.

[Lehmann 2000] Hans Lehmann, The Design of Information Systems for the International Firm: Grounded Theory of Some Critical Issues. In: Prashant C. Plavian, Shailendra C. Jain Plavian and Edward M Roche (eds.), *Global Information Techonology and Electronics Commerse, Issues for the New Millennium*. Ivy League Publishing, Marietta 2002, 370 – 392.

[Peltonen et al. 2002] Hannu Peltonen, Asko Martio ja Reijo Sulonen, *PDM – Tuotetiedonhallinta.* Edita Prima Oy, Helsinki, 2002.

[Robertson and Sribar 2002] Bruce Robertson and Valentin Sribar, *The Adaptive Enterprise, IT infrastructure Strategies to Manage Change and Enable Growth.* Intel Press, 2002.

[Sauer et al. 1997] Christopher Sauer, Philips W. Yetton and Associates, *Setps to the Future, Fresh Thinking on the Management of IT-Based Organiszational Transformation.* Jossey-Bass Publishers, San Francisco, 1997.

[Takki 2002] Pekka Takki, *IT-sopimukset: käytännön käsikirja.* Gummerus, Helsinki, 2002.

Ravivedonlyönti internetissä

Tommi Kinnunen

Tiivistelmä

Tässä tutkimuksessa tutkitaan, kuinka Fintoto oy:n vuonna 2002 avaaama Nettitoto-palvelu on vaikuttanut ravien kokonaisvaihtoihin. Lisäksi tutkitaan, minkälaisia käytettävyysongelmia palvelua käyttäneillä pelaajilla on ollut ja minkälaisia teknisiä ongelmia itse palvelussa on ollut.

CR-luokat: C.3

1. Johdanto

Vuonna 1996 Suomen Hippos ry aloitti etäpeliprojektin, jonka tarkoituksena oli helpottaa ravipelaamisen saavutettavuutta. Eri puolelle Suomea avattiin paikallisten yhdistyksien hallinnoimia etäpelipisteitä, joissa asiakkailla oli mahdollisuus pelata aina kyseisen päivän raveja. Pelipisteisiin annettiin käyttöön totomyyntiä varten kassapäätteet. Esimerkiksi Tampereelle avattiin tällaisia pisteitä kolme. Seuraavassa vaiheessa (vuodesta 1999 eteenpäin) kassapäätteitä levitettiin erityyppisiin kioskeihin (esim. R-kioskit) ja huoltoasemille.

Vuonna 2001 aloitettiin projekti, jonka tarkoituksena oli mahdollistaa ravien pelaaminen internetin välityksellä. Järjestelmä saatiin koekäyttökuntoon keväällä 2002. Koekäyttövaiheessa käyttäjiksi hyväksyttiin lähinnä ns. ravien sisäpiiriläisiä - henkilöitä, jotka työskentelevät ravien parissa. Koekäyttäjiä oli noin 600 henkilöä.

Valmis järjestelmä lanseerattiin 19. syyskuuta 2002 ja sille annettiin nimeksi Nettitoto. Käyttäminen vaatii rekisteröitymistä eli pelitilin avaamista. Voidakseen pelata on pelaajan siirrettävä pelitililleen rahaa joko online-maksuna (esim. Nordean Solo) tai offline-maksuna eli käyttämällä pankkien maksupalvelupäätteitä. Nettitoton ja muiden internet-pelipalveluiden välinen suurin ero on Nettitoton reaaliaikaisuus pelitilin saldon suhteen. Jos Nettitotossa pelattu peli on oikein, ovat voitetut rahat välittömästi käytössä uusia pelejä varten, kun taas muissa palveluissa voitot tulevat pelitilille vasta seuraavana päivänä. Nopea rahaliikenne aiheuttaa tietenkin omat riskinsä.

Osoituksena internet-pelaamisen helppoudesta ravipelien kokonaisvaihto kasvoi syyskuussa 2002 11,8 prosenttia verrattuna vastaavaan ajankohtaan

vuonna 2001 [Raviurheilun kuukausitiedote 2002]. Kun järjestelmä oli ollut käytössä vajaan kuukauden, oli rekisteröityneiden käyttäjien määrä noussut jo 5000 henkilöön.

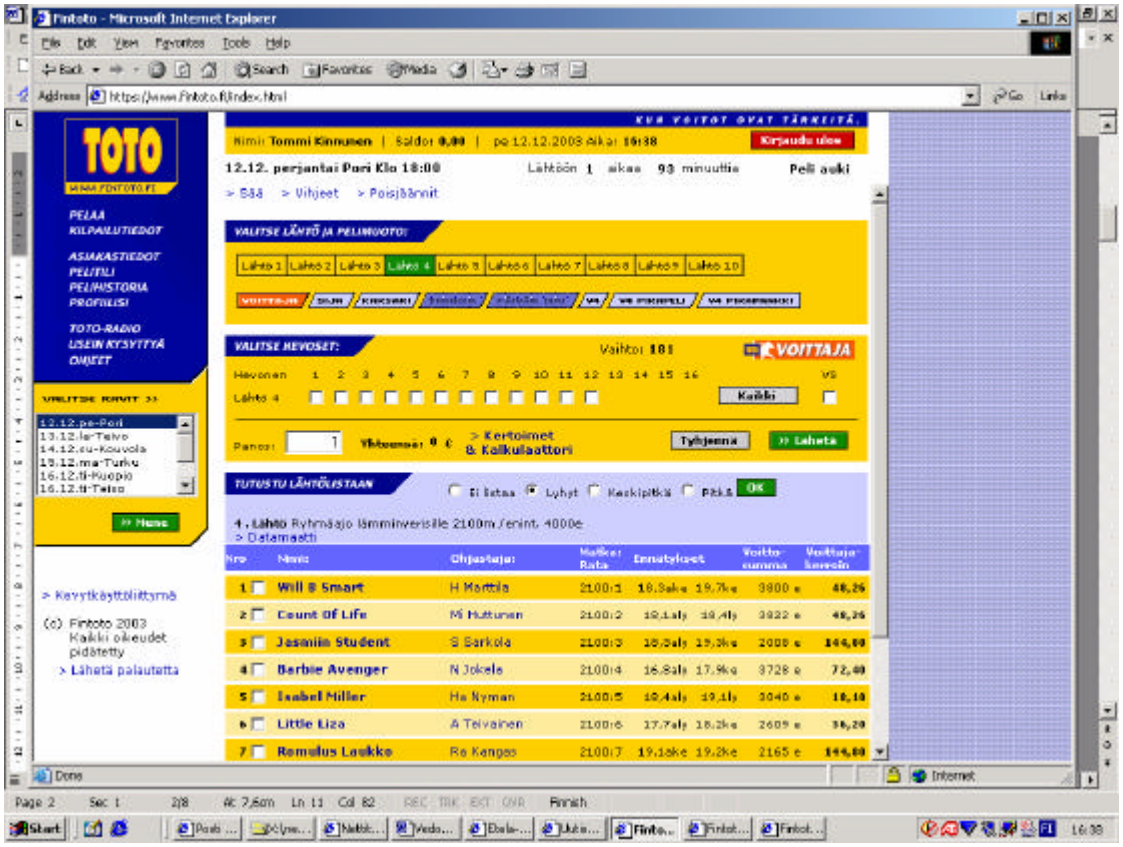
Tutkimuksessani keskityn ongelmiin, joita Nettitoton käytön aikana on ilmennyt. Käyttäjien käytettävyys ongelmat ovat oma mielenkiintoinen aihepiirinsä. Mielenkiintoinen aihepiiri on myös tutkia miten pelivaihdot tulevat kehittymään. Onko pelivaihtojen nousu vain alkuhuumaa vai jatkuuko kasvu edelleen? Vaikuttaako suosioon nimenomaan se, että järjestelmän rahaliikenne on nopeaa? Miten pelaajien rahat riittävät, kun pelaaminen on nyt tehty helpoksi? Tutkimusta luonnollisesti vaikeuttaa se seikka, että saatavilla oleva materiaali on rajallinen, koska järjestelmä on ollut käytössä melko vähän aikaa.

2. Järjestelmä

Nettitoton alusta www-pohjainen käyttöliittymä, josta on myös yhteydet hevos- ja ohjastajatietokantoihin. Järjestelmä käyttää tiedonsiirrossa SSL 3.0 standardia ja RC4-jonosalausta, jossa on 128 bittinen salausavain. Rekisteröityneen käyttäjän on mahdollista pelata kaikkia Fintoton hallinnoimia pelimuotoja, joita ovat voittaja-, sijoitus- ja kaksoisvedonlyönti, troikka, päivän duo, V4, V4-pikapeli ja V4-pikapankki. Pelaamisen edellytyksenä on, että käyttäjällä on rahaa pelitilillään. Ravien lähtölistat ovat selattavissa monessa eri muodossa. Pelattavien hevosten kertoimet ovat helposti saatavilla. Etuna normaaliin rata- tai etäpelipelaamiseen on, että kerroinlaskimen avulla pelaaja voi helposti laskea, paljonko sijoitetulla panoksella on mahdollista voittaa.

2.1. Pelaaminen

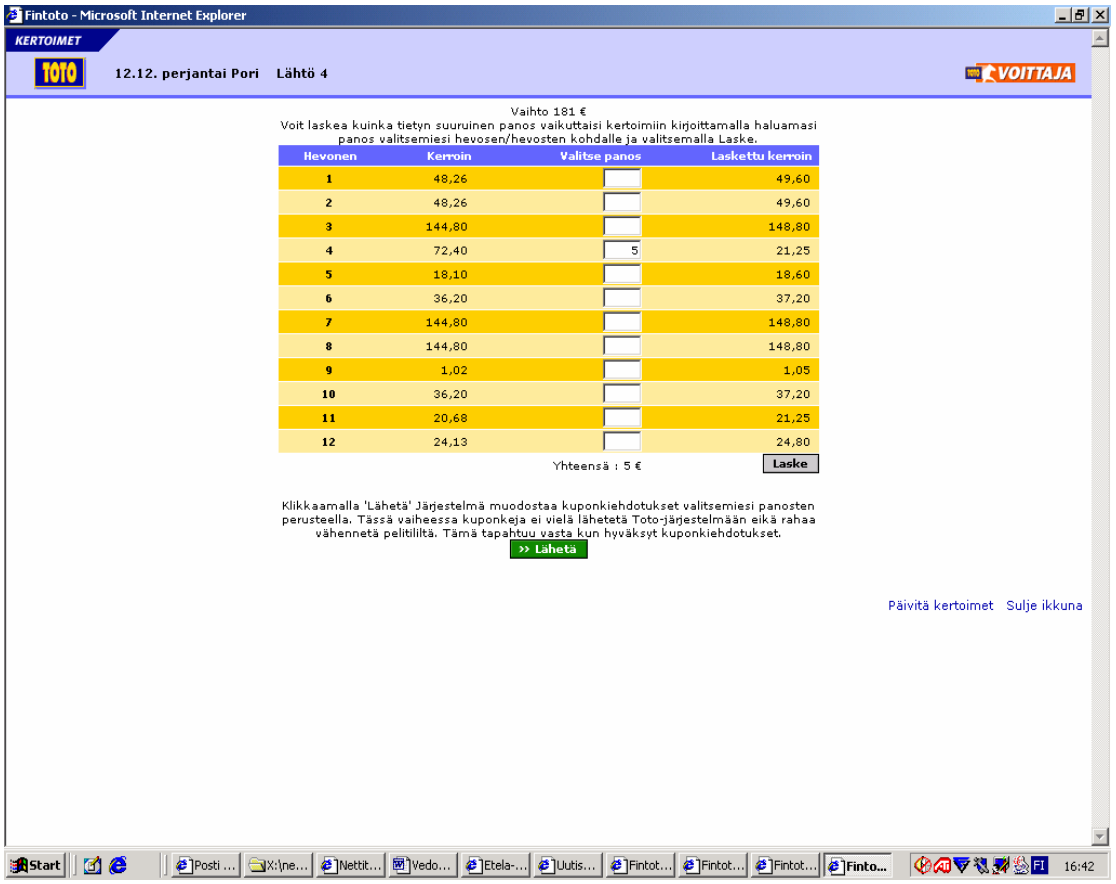
Kuvassa 1 on näkyvissä nettitoton pelisivu, jossa on valittuna Porin ravien neljäs lähtö ja pelimuotona voittaja peli. Kuvan alaosassa on lista lähdössä juoksevista hevosista. Sivulta löytyy myös pelaamisen kannalta oleellista lisätietoa kuten paikkakunnan sää, vihjeitä päivän lähtöihin ja poisjäännit. Vasemmassa reunassa on lista kaikista raveista, joita sillä hetkellä on mahdollista pelata.



Kuva 1. Nettitoton pelisivu.

2.2. Lisäominaisuudet

Kuvassa 2 näkyy sivujen kerroinlaskuriosuus. Vasemmalta ensimmäisenä on hevosten kilpailunumerot. Seuraavana on kunkin hevosen senhetkinen voittajakerroin. Valittu panos -sarakkeeseen voi syöttää haluamansa panoksen. Tässä numerolle neljä on sijoitettu panos 5 euroa, jolloin voittajakerroin on laskenut 72,40:stä 21,25:een. Näin pelaaja näkee välittömästi oikealla olevasta sarakkeesta, kuinka paljon hänen on mahdollista voittaa. Tämä toiminto on selvä etu verrattuna normaaliin pelaamiseen, jossa kertoimen näkee vasta kun peli on jätetty totojärjestelmään.



Kuva 2. Nettitoton kerroinlaskuri.

2.3. Edut

Perinteisestihän sähköisiltä kauppapaikoilta odotetaan edullisempia hintoja kuin normaaleilta kaupoilta. Nettitoton kohdalla tämä ei kuitenkaan ole mahdollista vaan pelaaminen netin kautta on jopa kalliimpaa. Näin siinä tapauksessa, että laskelmissa huomioidaan yhteyskustannukset. Tässä konseptissa edut ovat toisenlaisia. Nettitoto tarjoaa käyttäjälleen parempaa palvelua ja pelaamista ajasta ja paikasta riippumatta. Käyttäjällä on myös mahdollisuus kuunnella ilmaiseksi ravien selostusta. Toiminto vaatii RealOne player -ohjelman.

Kettusen [1998] mukaan verkkoliiketoiminnan tavoitteet määritellään seuraavasti:

1. yrityksen nykyisen liiketoiminnan tehostaminen
2. kilpailuedun ja kustannussäästöjen hankkiminen
3. uusien asiakkaiden ja liiketoimintamallien etsiminen tietoverkkojen avulla.

Nettitoton osalta voidaan sanoa, että kaikki nämä ovat toteutuneet. Internetin avulla pelivaihtoja on saatu kasvatettua. Kilpailuetua internet ei

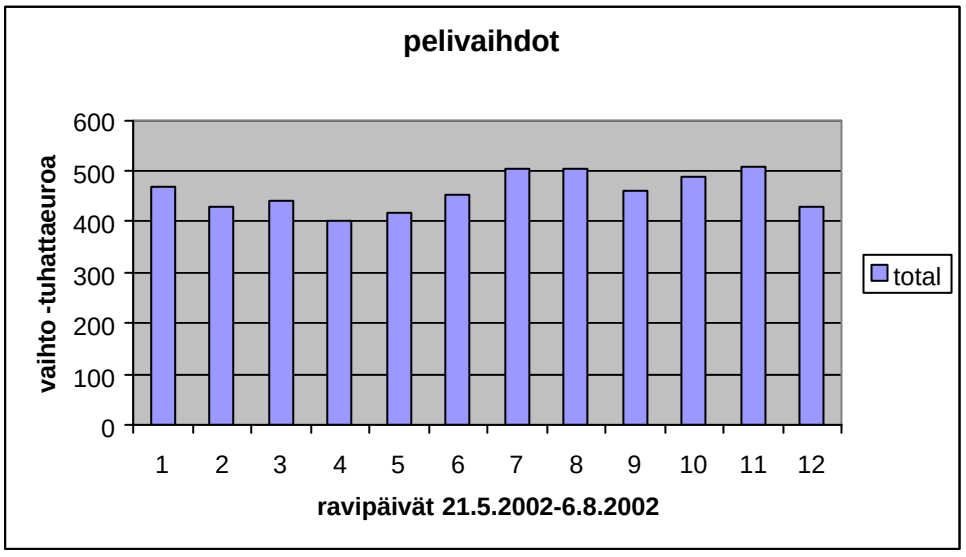
varsinaisesti tarjoa, koska kilpailua ei Suomen lainsäädännön puitteissa ole. Tosin jotkut ulkomaalaiset peliyhtiöt tarjoavat pelattavia kohteita myös Suomen raveista, mutta tarjonta on kuitenkin vähäistä ja pelimuodot erilaisia. Oy Veikkaus Ab voidaan toki luokitella kilpailijaksi olettaen, että pelaajille on yhdentekevää, mitä he pelaavat kunhan pelaavat. Ravipelaaminen on kuitenkin aivan oma lajinsa, joten Veikkauksen pelitarjonta ei sinänsä kilpaile Fintoton tuotteiden kanssa. Luonnollisesti pelaajilla on rajallinen määrä pelirahaa, jonka sijoituskohteista sinänsä tietenkin kilpaillaan. Onko uusia asiakkaita sitten pystytty hankkimaan? Todennäköisesti jonkin verran, mutta varmaa tietoa ei asiasta ole. Asian tutkiminen on mahdollista ainoastaan Fintotolle.

3. Vaihtojen kehitys

Tässä luvussa luodaan katsaus pelivaihtojen kehitykseen ennen Nettitotoa, koekäyttövaiheen ja lanseerausvaiheen aikana ja lopuksi tämänhetkinen tilanne. Tiedot perustuvat Teivon raviradan ravipäivien vaihtoihin [Kujansuu 2003]. Teivon ravirata on Suomen toiseksi suurin ravirata, mutta siellä pelattava V4-ratapeli on pelivaihdoiltaan Suomen suurin.

3.1. Vaihdot ennen Nettitotoa

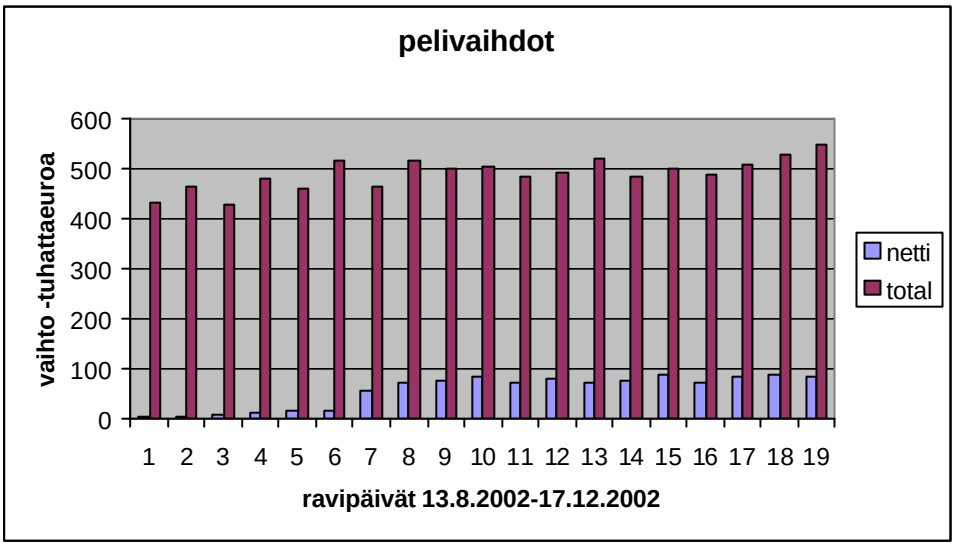
Pelivaihdot on otettu ajalta 21.5.2002-6.8.2002, jolloin Nettitoto ei vielä ollut käytössä. Vaihdoissa on normaalia vaihtelua. Pienin vaihto on noin 400 000 euroa ja suurin hieman yli 500 000 euroa.



Kuva 3. Pelivaihdot ennen Nettitotoa.

3.2. Koekäyttövaihe ja Nettitoton lanseeraus

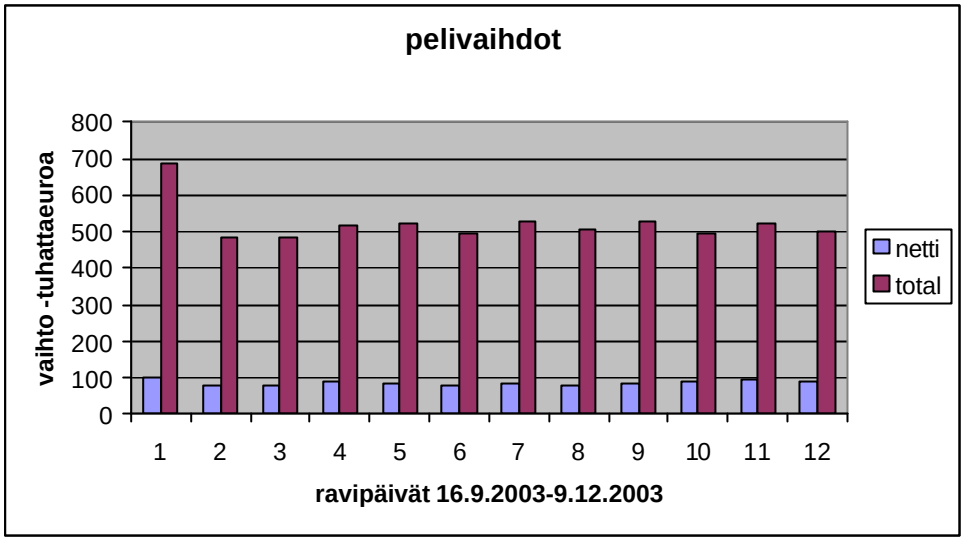
Seuraava otos ajoittuu aikaan, jolloin nettitoton koekäyttö on alkanut ja ajalle jolloin varsinainen lanseeraus (19.9.2002) on tapahtunut. Kuvassa 4 lanseeraus on päivä numero 7. Tämän päivän kohdalta huomataan, että Nettitoton vaihto on selvästi suurempi kuin edellisenä päivänä (päivä numero 6), mutta kokonaisvaihtoon sillä ei ole vaikutusta. Kokonaisvaihto on jopa pienempi kuin edellisenä päivänä. On kuitenkin huomattavissa, että jatkossa kokonaisvaihdot lähentelevät tasaisesti 500 000 euroa ja jopa ylittävät sen, kun ne aikaisemmin vaihtelivat välillä 400 000 – 500 000 euroa. Nettitoton osuus on jonkin verran alle 100 000 euroa suurimman vaihdon ollessa noin 86 000 euroa. Nettitoton käyttöönotolla on siis ollut selvä positiivinen vaikutus kokonaisvaihtoihin.



Kuva 4. Koekäyttö- ja lanseerausvaihe.

3.3. Nykyinen tilanne

Tällä hetkellä näyttää siltä, että pelivaihdot ovat todellakin asettuneet pysyvästi 500 000 euron tasolle. Nettitoton osalta pelivaihdot ovat asettuneet hieman alle 100 000 euroon. Tämä voidaan havaita kuvasta 5, joka ajoittuu ajalle 16.9.2003-9.12.2003. Kuvassa 5 näkyvä poikkeuksellisen suuri vaihto selittyy sillä, että Teivossa järjestettiin kyseisenä päivänä Euroopan mestaruusosakilpailu UET-Grand Prix, mikä automaattisesti nosti pelivaihtoja normaalista. Vuonna 2003 Fintoton liikevaihto internetissä on yli 26 miljoonaa euroa [Fintoto 2003], joka vastaa noin 16 prosenttia kokonaismyynnistä.



Kuva 5. Nykyinen tilanne

3.4. Tulevaisuus

Nettitoto kehittyy tällä hetkellä edelleen. Marraskuussa 2003 esiteltiin uusi versio, joka mahdollistaa pelaamisen matkapuhelimella. Toton pelaamiseen kännykällä vaaditaan matkapuhelin, joka on varustettu XHTML-selaimella. Soveltuvia puhelimia ovat tällä hetkellä Nokian mallit 3650, N-Gage, 3660, 6600, 7650, 9210i ja SonyEricssonin malli P800. Nykyaikaisen mobiilipelisovelluksen rakentamisesta on vastannut European Game and Entertainment Technology Ltd Ab (EGET). Tulevaisuudessa totopelit ovat mahdollisia myös digi-TV:n avulla. [Fintoto 2003]

4. Teknisiä ongelmia

Lanseerauksen jälkeen Nettitoton käytössä on ilmennyt joitakin pieniä ongelmia. Lokakuun ensimmäisellä viikolla 2002, jolloin järjestelmään rekisteröityneiden pelaajien määrä oli ylittänyt 6000 käyttäjän rajan, havaittiin pieniä katkoksia. Katkokset ilmenivät, kun käyttäjiä oli saamaan aikaan paljon. Hätäisimmät käyttäjät virittelivät jopa boikotteja. Fintotolta kuitenkin vakuutettiin, että katkokset eivät johtuneet kapasiteetistä vaan ongelmat johtuivat ohjelmistovirheistä. Katkoksia ilmeni yhteensä kolmena päivänä ja ne olivat noin puolen tunnin mittaisia kerrallaan.

Suurin yksittäinen pelimuoto on V4-ratapeli, jonka jättöaikana myös käyttäjiä järjestelmässä on kaikkein eniten. Kuitenkin keskiviikkona 2.10.2002, jolloin V4-ratapeliä ei pelata, ilmeni myös katkoksia. Tämä todistaa sen, että

myös pieni kuorma saattaa aiheuttaa virheitä eivätkä katkokset siis johdu liian pienestä palvelin kapasiteetista [Hevosurheilu 2002].

Tällä hetkellä (tilanne marraskuu 2003) rekisteröityneiden käyttäjien määrä on noin 16500. Viime aikoina on ilmaantunut uusia ongelmia. Ne eivät kuitenkaan ole olleet samanlaisia kuin aikaisemmin, vaan niiden taustalla on ollut teleoperaattorien vaikeudet. Teleoperaattoreilla on ollut suuria vaikeuksia yhteyksiensä kanssa, mikä on vaikuttanut myös nettitoton toimivuuteen [Jääskeläinen 2003]. Fintotossa toivotaankin, että teleoperaattorit saavat ongelmansa korjattua pikaisesti.

Veikkauksen keskustelupalstalla on käyty keskustelua nettitoton toimivuudesta. Mielipiteet jakautuvat suurinpiirtein puoliksi. Toiset väittävät, että järjestelmä jumiutuu usein. Toiset taas ovat sitä mieltä, että mitään ongelmia ei ole ollut vaan kaikki toimii normaalisti. Monien mielestä joidenkin käyttäjien ongelmat johtuvat lähinnä vajaatehoisesta kotikoneesta ja vanhoista selainohjelmista. Yhteenvetona keskustelupalstan kommenteista voi sanoa, että käyttöongelmia on varmasti ollut, mutta sitä mikä ne on aiheuttanut, ei voi tarkasti tietää.

Nettitoton palvelun ylläpitäjä on ElisaCom oy, joka on tehnyt sähköisen toteutuksen yhdessä tytäryhtiö Elisa Internet oy:n kanssa. Pelisovellukset pyörivät Elisan palvelimella ympärivuorokautisesti. Fintotolla ja ElisaComilla on viiden vuoden sopimus tietoliikenne ratkaisuista. ElisaCom vakuuttaa, että sillä on kyky vastata kasvun haasteisiin ja tarpeen tullen muuttaa laitteisto kokoonpanoa.

Nettitoto poikkeaa muista sähköisistä kauppapaikoista siten, että sille on luonteenomaista pelaamisen raju lisääntyminen juuri ennen kohteiden sulkeutumista. Kymmenen pelattavaa kohdetta kolmen tunnin sisällä vaatii sovelluksia pyörittäviltä palvelimilta ja tietoliikenneyhteyksiltä erityisesti skaalautuvuutta sekä toimintavarmuutta.

Tehtyjen kuormitustestien mukaan järjestelmä on selviytynyt hyvin kuormituspiikeistä ja myös viime hetken pelitapahtumat ovat onnistuneet. Totopelaamista varten pelaaja tarvitsee internetyhteyden, mutta muiden pelijärjestelmän tarjoamien palveluiden täysimääräinen hyödyntäminen onnistuu parhaiten laajakaistayhteyksien avulla. [Kolumbus 2002]

5. Käytettävyys

Hyvä käytettävyys on varsinkin tämän kaltaisen www-palvelun elinehto. Käyttäjien kommenttien perusteella voidaan sanoa, että Nettitoton www-

sivujen käytettävyys on kunnossa. Kukaan haastattelemistani henkilöistä ei ole löytänyt sivuilta mitään merkittävää käytettävyysongelmaa.

Nettitoton www-sivut löytyvät osoitteesta www.fintoto.fi. Ennen kuin pelaamisen voi aloittaa, on käyttäjän täytettävä rekisteröintilomake. Saatuaan käyttäjätunnuksen ja salasanan käyttäjä voi siirtyä haluamallensa sivulle. Valittavana on koko viikon ravitarjonta. Pääsääntöisesti on mahdollista pelata kaikkia listassa olevia raveja, mutta jos valitun ravien lähtölista ei ole valmistunut, pelaaminen ei luonnollisestikaan ole mahdollista.

Käytettävyyden kyselytutkimukseen osallistui neljä koekäyttäjää. He olivat noin 30-vuotiaita miehiä. Kaikki ovat pelanneet ravipelejä jo useiden vuosien ajan ja näin ollen kaikki pelaamiseen liittyvä oli jo ennalta tuttua. Kaikille esitettiin seuraavat kysymykset:

1. Kuinka kokenut tietokoneen ja internetin käyttäjä olet?
2. Minkälainen internetyhteys sinulla on?
3. Kuinka usein pelaat?
4. Miten pelaamisesi on muuttunut, kun aloit käyttää nettitotoa?
5. Minkälaisia käytettävyysongelmia olet havainnut käytön aikana?
6. Oletko törmännyt teknisiin ongelmiin – järjestelmä kaatunut yms.

Vastaukset kolmeen ensimmäiseen kysymykseen on kerätty taulukkoon 1.

	Kysymys 1	**Kysymys 2**	**Kysymys 3**
Pelaaja 1	Käyttänyt tietokonetta sekä opiskelussa että työssä. Käyttää tietokonetta päivittäin.	ISDN-liittymä kotona	2-3 kertaa viikossa
Pelaaja 2	Tutustunut tietokoneeseen opiskeluaikoina. Tuntee internetin ja käyttää sitä päivittäin työpaikallaan.	Kotona kiinteä internetyhteys.	2-3 kertaa viikossa, joskus useamminkin
Pelaaja 3	Käyttänyt tietokonetta opiskeluaikoina. Internetin käyttäminen on jonkin verran outoa.	Kotona modeemiyhteys	Päivittäin
Pelaaja 4	Käyttänyt tietokonetta opiskeluaikoina ja töissä. Internetin käyttäminen on erittäin tuttua.	Kotona kiinteä internetyhteys	Lähes päivittäin

Taulukko 1. Vastaukset kysymyksiin 1-3.

Vastaajat ovat siis yhtä lukuun ottamatta varsin kokeneita tietokoneen ja internetin käyttäjiä, ja myös vähiten käyttökokemusta omaava Pelaaja 3 on käyttänyt tietokonetta opiskellessaan.

Kysymykseen 4 Pelaaja 1 vastasi seuraavasti: ”Pelaaminen on muuttunut holtittomaksi. Rahaa tulee helpommin sijoitettua isompiakin summia ja pelattua useammin, koska se ei kirpaise samalla tavalla, kun rahasta ei konkreettisesti luovu. Lisäksi tulee tosiaan pelattua viikoittain useampiinkin raveihin, kun ennen tuli pelattua käytännössä vain Teivon tiistairaveihin. Pelit on helpompi kohdentaa juuri haluamiinsa lähtöihin.” Muiden vastauksissa mainittiin lisäksi mahdollisuus pelata työn lomassa, pelata vain kiinnostavia lähtöjä ja mahdollisuus pelata yhtä paljon kuin ennenkin, vaikka käy raviradalla entistä harvemmin.

Modeemiyhteyttä käyttävä Pelaaja 3 pitää sivujen latautumisen hitautta tuskastuttavana. Pelaajan 4 mielestä järjestelmä ei riittävän hyvin tiedota hevosten poisjäänneistä: jos pelaaja on valinnut hevosen suosikkihevosekseen ja se jää pois, niin viesteihin tulee ilmoitus, että hevonen starttaa, vaikka näin asia ei ole. Pelaaja 1 puolestaan valittaa sitä, ettei voi Fintoton sivujen kautta toteuttaa viime hetken ideoitaan haluamistaan riveistä Veikkauksen peleissä V5 ja V75, joita ei voi pelata Nettitoton kautta. Pelaaja 1 siis toivoo, ettei tarvitsi käyttää kahta erillisestä pelitiliä (Fintoton ja Veikkauksen).

Teknisiä ongelmia eivät Pelaajat 2 ja 3 ole kohdanneet. Sen sijaan Pelaajalla 1 ovat omalta pankkitililtä siirretyt rahat kerran kuittautuneet pelitilille vasta muutaman päivän viipeellä. Pelaajalla 4 järjestelmä on ajoittain ”takkuillut”. Hän on myös havainnut virheellisen toiminnon kertoimien laskussa ja pelitilin saldon automaattisessa päivityksessä.

6. Yhteenveto

Teivon raviradan vaihtotietojen perusteella voidaan sanoa, että Nettitoto on lisännyt vaihtoja ja vakiinnuttanut ne korkeammalle tasolle kuin aikaisemmin. Pelivaihtotiedot olivat noin puolentoista vuoden ajanjaksolta, joten niiden perusteella voidaan varmuudella sanoa, että Nettitotolla on ollut vaikutusta. On mielenkiintoista seurata jatkossa pelivaihtojen kehitystä ja tarkastella, kuinka mobiilipelaaminen tulee niihin vaikuttamaan.

Käytettävyyskyselyjen ja omien käyttökokemuksien perusteella voidaan sanoa, että Nettitoton käytössä ei ole merkittäviä puutteita. Sivut toimivat ja käytettävyys on hyvä. Ravipelaamisen luonteeseen kuuluva viimehetken pelaaminen onnistuu luontevasti ja nopeasti myös Nettitoton kautta ainakin silloin, kun käyttäjällä on käytössään modeemia nopeampi internet-yhteys.

Vastauksista nousi selvästi esille nettitoton aikana muuttunut pelitoimintamalli, jossa pelit on helppo kohdentaa juuri haluamiinsa lähtöihin, jolloin ns. turhat pelit jäävät vähemmälle. Näin on luonnollisesti mahdollista toimia ilman Nettitotoakin, mutta kokemus on osoittanut, että esimerkiksi paikanpäällä ollessaan pelaaja pelaa helposti jokaiseen, lähtöön vaikka varsinaista näkemystä hevosista ei olisikaan.

Rahojen siirrossa esiintynyt viive ei todennäköisesti ole mikään järjestelmä virhe vaan on aiheutunut mahdollisesti käyttäjän huolimattomuudesta. Näin voi päätellä www-sivuilta löytyvästä faq-osiosta. Mikäli siirrossa on käytetty verkkomaksua, nettipelijärjestelmä ei ole jostain syystä saanut kuittausta suoritetusta siirrosta. Tämä voi johtua esimerkiksi siitä, että pankin järjestelmästä ei ole palattu takaisin nettipelisivuille tai internet-yhteys on katkennut. Tällöin rahat tulevat pelitilille 1-2 pankkipäivän viiveellä.

On epätodennäköistä, että Fintoto ja Veikkaus toteuttaisivat järjestelmän, jonka avulla pelaaja voisi pelata V5- tai V75-raviveikkaukset suoraan Nettitoton kautta, vaikka se järkevää olisikin. Molemmat kehittävät omia internet- palveluitaan omalla tahollaan.

Järjestelmän toimivuudessa on selvitysten mukaan ollut ongelmia, mutta esimerkiksi itse en ole niihin koskaan törmännyt. Viimeaikaiset ongelmat ovat aiheutuneet teleoperaattoreista, joten niille Fintoto ei voi mitään. Jos käyttäjä

löytää järjestelmästä moitittavaa tai selviä virheitä, hänen kannattaa ottaa yhteyttä suoraan Fintotoon sähköpostilla. Kokemuksien mukaan vastaus tulee nopeasti, oli sitten kyse ongelmasta tai parannusehdotuksesta.

Viiteluettelo

[Elisa 2002] Netttoto pyörii Elisan palvelimilla ja yhteyksillä. http://www.kolumbus.com/Pressi/Tiedotearkisto/Nettitoto/nettitoto.html (marraskuu 2003).

[Fintoto 2003] http://yritys.fintoto.fi/indexpocket.html. (joulukuu 2003).

[Hevosurheilu, 2002] Nettitotossa viime viikolla katkoksia. Hevosurheilu 9.10.2002

[Jääskeläinen, 2003] Lauri Jääskeläinen, Mobiilia ravipeliä. IS Veikkaaja **46** (11.11.2003).

[Kettunen, 1987] Sami Kettunen, Elektroninen kaupankäynti Liiketoiminta tietoverkoissa. Teknolit, Jyväskylä, 1998.

[Kujansuu, 2003] Jukka Kujansuu, Henkilökohtainen tiedonanto, joulukuu 2003.

[Suomen Hippos 2002] Suomen Hippos, Syyskuun kuukausitiedote. http://www.hippos.fi/raviurheilu/syyskuu.pdf (marraskuu 2003).

Satelliittipaikannuksen tarkentaminen kaupunkiympäristössä

Tommi Lehtinen

Tiivistelmä

Tämä tutkielma käsittelee joitakin GPS-paikannuksen erityispiirteitä ja ongelmia urbaanissa ympäristössä sekä mahdollisia menetelmiä, joilla voidaan analysoida ja tarkentaa digitaalisella kartalla esitettävää sijaintia.

Avainsanat ja -sanonnat: GPS, kartta, paikannus, tarkentaminen.
CR-luokat: G.3, I.2

1. Johdanto

Erilaiset paikannusmenetelmät ovat jo arkipäivää myös siviilikäytössä. Yleisimmin hyödynnetty on yhdysvaltalainen GPS eli Global Positioning System. Euroopalla ja Kiinalla on kehitteillä oma vastaava hankkeensa nimellä Galileo. Järjestelmän suunniteltu valmistumisvuosi on 2008. Lisäksi on olemassa ainakin entisen Neuvostoliiton sotilaskäyttöön perustettu GLONASS, joka on rakenteeltaan samankaltainen kuin GPS ja edelleen käytössä. Muitakin paikannusmenetelmiä on olemassa, esimerkiksi GSM-matkapuhelinverkon perusominaisuus paikantaa matkapuhelimia. Myös WLAN-verkkoihin on kehitetty joitakin paikannusmenetelmiä. Muiden paikannustekniikoiden kuin GPS:n hyödyntäminen yleisesti on vielä melko vähäistä.

Erilaiseen ammattikäyttöön GPS on leviämässä tällä hetkellä kovaa vauhtia. Erityisesti viranomaiset, taksit sekä logistiikkayritykset saavat paikannustekniikoista merkittävää hyötyä. Tyypillisesti nämä ryhmät käyttävät tai ovat siirtymässä käyttämään järjestelmiä, jossa mm. käyttäjän sijainti esitetään reaaliaikaisena karttanäytöllä. Tämä tutkielma keskittyy siihen, kuinka GPS-satelliittipaikannusjärjestelmän esittämää sijaintia digitaalisella kartalla voidaan tarkentaa ja optimoida erityisesti kaupunkiympäristössä, kun otetaan huomioon sekä analysoidaan myös muita saatavilla olevia tietoja. Lisäksi tarkastellaan mahdollisuuksia hyödyntää paikannusjärjestelmää tehokkaammin liikenteeseen liittyvässä tutkimuksessa. Muiden kuin GPS-järjestelmän ominaisuuksia ei ole otettu tässä tutkielmassa huomioon. Kuvatut menetelmät voivat kuitenkin olla jossain määrin hyödynnettäviä myös muiden paikannusmenetelmien kanssa.

2. GPS-paikannus

Global Positioning System (GPS) on satelliitteihin perustuva maailmanlaajuinen paikannusjärjestelmä, jota ylläpitää USA:n puolustusministeriö. Järjestelmä suunniteltiin alun perin sotilaskäyttöön, mutta sitä käyttävät myös miljoonat siviilit ympäri maailmaa. Siviilikäyttäjille on tarjolla vähemmän tarkka Standard Positioning Service (SPS). USA:n armeijalla sekä sen liittolaisilla on käytössä tarkempi Precise Positioning Service (PPS). [Kaplan, 1996]

GPS-paikannuksen hyödyntäminen on lisääntynyt siviilikäytössä erityisesti vuoden 2000 toukokuun jälkeen, kun GPS-satelliittien sotilaallinen häirintäsignaali poistettiin käytöstä. Tällöin GPS-paikannus tarkentui 100-200 metristä keskimäärin 10-20 metriin.

Satelliittipaikannuksen tarkkuutta voidaan tarkentaa kiinteiden jatkuvasti satelliittien kulkua seuraavien maa-asemien avulla. Suhteellisen satelliittipaikannuksen (Differential GPS) avulla päästään jo muutaman metrin paikannustarkkuuteen. Sisätiloissa ja tiheästi rakennetuilla kaupunkialueilla satelliittipaikannusta voidaan tehostaa A-GPS (Assisted GPS) menetelmällä. [Aittola, 2003]

2.1. GPS-paikannuksen tekniikka

GPS-järjestelmä kokonaisuudessaan koostuu satelliiteista, maan päällä sijaitsevista tarkkailuasemista sekä järjestelmän käyttäjien GPS-vastaanottimista. Järjestelmä antaa käyttäjälleen tietoa sijainnista, keskimääräisestä nopeudesta sekä liikkeen suunnasta.

GPS-satelliitteja on yhteensä 24 kappaletta, joita uusitaan säännöllisesti. Satelliitit on jaettu kuuteen neljän satelliitin ryhmään, jotka kiertävät tasaisin välein omia reittejään niin että koko maapallolla on mahdollisimman tasainen satelliittipeitto. Yhden satelliitin kierros maapallon ympäri kestää noin 24 tuntia, jolloin ne ovat aina tiettyyn vuorokauden aikaan tietyssä kohdassa. Tällä tavalla saadaan yhteys 5-8 satelliittiin mistä päin maapalloa tahansa.

Tarkkailuasemia on useita ympäri maapalloa. Nämä tarkkailevat GPS-satelliittien signaaleja sekä todellisia sijainteja ja laskevat korjaukset näiden ennalta määrättyihin ratoihin. Näiden tietojen perusteella päätarkkailuasema ylläpitää ja päivittää satelliittien sijainnit ja kellot.

GPS-vastaanottimet tarkkailevat satelliittien lähettämiä signaaleja ja määrittävät niiden perusteella oman sijaintinsa. Sijainti saadaan määritettyä, kun signaalien perusteella lasketaan etäisyys vastaanottimen ja näkyvien satelliittien välillä. Sekä huomioidaan etukäteen tunnetut satelliittien sijainnit tiettynä aikana. Tarkkaan kolmiulotteiseen paikanmääritykseen vaaditaan

yhteys vähintään neljään satelliittiin. Järjestelmä toimii jossain määrin myös 2-3 satelliitilla, mutta paikannus ei ole silloin kovin luotettava. Satelliittiyhteyksien laatuun vaikuttavat monet asiat. [Kaplan, 1996]

Globaalit koordinaattijärjestelmät esitetään yleensä CTS:n (Conventional Terrestial System) mukaisesti. GPS:n käyttämä koordinaattijärjestelmä on WGS84 (World Geodetic System 1984), joka on CTS-järjestelmän mukainen. GPS antaa käyttäjälleen fysikaaliset koordinaatit maapallon pinnalla suhteessa planeetan ekvaattoriin ja sovittuun meridiaaniin. Nämä maantieteelliset koordinaatit voidaan muuttaa tietyillä kaavoilla suorakulmaisen kolmiulotteisen koordinaatiston koordinaateiksi. Sekä tietenkin sovittaa kaikenlaisten kaksiulotteisten karttojen koordinaattijärjestelmiin. [Kaplan, 1996]

Havaitakseen satelliitin signaalin vastaanottimen ja satelliitin välillä tulee olla melko esteetön yhteys. Suurin este on tietenkin maapallo, horisontin takana olevia satelliitteja ei havaita. Myös rakennukset sekä tiheämmät metsät estävät signaalien kulun. Ikkunat, kuten esimerkiksi auton tuulilasi, eivät häiritse yhteyttä. Satelliittien signaalit voivat myös heijastua seinistä tai maan pinnasta, jolloin vastaanottimeen voi tulee signaaleja useita reittejä pitkin. Tätä kutsutaan monitie-etenemiseksi (MTE). Kaupunkiympäristössä tämä on merkittävä virhelähde. [Tiainen, 2001] Esimerkiksi kerrostalojen välissä signaalit voivat kadota ja paikannus vääristyy tai jopa keskeytyy. Henkilökohtaiset kokemukseni uusilla ammattikäyttöön kehitetyillä järjestelmillä keväällä 2003 tukivat näitä havaintoja. Esitetyn sijainnin todellisuudesta poikkeava harhailu sekä satelliittiyhteyksien pätkiminen olivat hyvin yleinen ilmiö kaupunkialueilla.

2.2. Paikannusjärjestelmät ajoneuvoissa

Pohjois-Amerikassa ja Keski-Euroopassa on jo useita vuosia käytetty yleisesti järjestelmiä, joissa autossa sijaitseva GPS-vastaanotin sekä siihen liitetty sovellus, opastavat kuljettajan haluttuun paikkaan, ilmoittamalla esimerkiksi maanteillä ennen risteyksiä suunnan mihin pitää kääntyä. Nämä perustuvat pitkälti vektoripohjaisiin karttoihin ja ohjelmistoihin, jotka tarjoavat mm. nopeimman tai lyhimmän reitin laskemisen paikasta A paikkaan B.

Ajoneuvoon asennettavan laitteiston perusosat koostuvat GPS-paikantimesta, tietokoneesta sekä karttanäytöstä. Tällaisia mahdollisia kokoonpanoja on lukematon määrä kehittyneimmistä kännyköistä kämmentietokoneisiin, kannettaviin tietokoneisiin tai muihin ajoneuvoihin soveltuviin tietokonelaitteistoihin.

Kuva 1: Genimap Navigator –karttanäyttö. [www.genimap.fi]

Taksiautot ovat maailmanlaajuisesti yksi keskeinen suuri ryhmä joka on ryhtynyt hyödyntämään satelliittipaikannusta. Esimerkiksi Pariisin alueella on n. 14900 taksiautoa, Tampereella taksiautoja on n. 260.

Tällaista yhtenäistä käyttäjämäärää on erittäin kannattavaa hyödyntää myös liikennetutkimuksessa, varsinkin kun tarvittava laitteisto on usein jo valmiina. Taksit voivat muodostaa tehokkaasti yhteisen reittitietokannan hyödyntäen koko toiminta-alueen ja järjestelmän käyttäjien potentiaalia. Tällä tavalla tietoa ajetuista reiteistä kertyy jatkuvasti ja tilanne toisaalta myös elää reaaliajassa. Esimerkiksi tietöiden aiheuttamat katkokset päivittyvät järjestelmään nopeasti kun jonkun reitin käyttäminen loppuu yllättäen kokonaan. Ja mitä enemmän tietoa kertyy, sitä enemmän ja tarkemmin sitä voidaan analysoida, jopa yksittäisten päivien tai tiettyjen kellonaikojen tarkkuudella.

2.3. Paikannusjärjestelmien kartat

Erilaisissa laitteistoissa toimivia paikannusohjelmistoja on useita, niin kuin valmistajiakin. Tyypillisesti ne koostuvat jonkun tietyn alueen karttapohjasta tai useammista kartoista, jotka voivat olla joko vektorityyppisiä vapaasti skaalattavia tai bittikarttatyyppisiä kuvia eri mittakaavoissa. Näistä kartoista esitetään käyttäjälle tietty alue jolla käyttäjä sijaitsee sekä GPS-laitteen antama sijainti kartalla. Lisäksi niissä on yleensä jonkinlaista yksinkertaista ennakointia korjaamassa GPS-signaalin viivettä autolla ajaessa.

Kun GPS-paikannusta käytetään autossa ja kaupunkiympäristössä, on erittäin todennäköistä että auto myös kulkee pitkin kaupungin katuja eikä esimerkiksi kerrostalojen katoilla tai vesialueilla. Tarkimmat nykyiset siviilijärjestelmät tyytyvät useimmiten näyttämään vain sijainnin kartalla, eivätkä sen erityisemmin analysoi karttapohjaa suhteessa auton liikkeisiin tai sitä, onko esitetty sijainti kartalla järkevä. Syitä esitetyn sijainnin poikkeamiseen todellisuudesta on monia. Useimmissa tapauksissa pelkkä GPS-järjestelmän esittämä tarkkuus ilman erityisiä korjauksia onkin täysin riittävä.

2.4. Tieinformaatio digitaalisissa kartoissa

Digitaalisen karttapohjan analysoinnin sekä paikannuksen tarkentamisen kannalta informaatio tien keskilinjasta on hyvin oleellinen tieto. Vektoripohjaisissa kartoissa tieto tien sijainnista ja sen keskilinjasta on tallennettu sarjana koordinaattipisteitä, jotka ovat yhteydessä toisiinsa, yleensä vain jonkinlaisen koordinaatteja tallentavan tietorakenteen perusteella. Nykyiset vektorigrafiikkaan perustuvat kartat ovat tarkimmillaankin melko suuripiirteisiä ja eroa todellisuuteen voi tulla jopa kymmeniä metrejä. Tällaisia karttoja ei ole juuri saatavilla sillä tarkkuudella, joka esittäisi kaikki kaupunkien kadut ja kujat, kaistoista puhumattakaan. Nykyiset tämän mittakaavan kartat, joita tarkimmissa paikannusjärjestelmissä käytetään, ovat bittikarttapohjaisia kuvia, joista helposti saatavilla oleva ja tarkka teiden keskiviivatieto puuttuu. Bittikarttapohjaisen kartan analysointi tien keskiviivan löytämiseksi on melko vaativa suoritus. Varsinkin jos se pitäisi tehdä tarkasti lähes reaaliajassa nykyisillä mobiililaitteilla sekä kuvantunnistusmenetelmillä.

Monet kehitteillä olevat navigointi- ja turvallisuusjärjestelmät edellyttävät kuitenkin hyvin tarkkoja tiekarttoja, joissa kuvataan jopa yksittäiset kaistat ja niiden sijainti. Tällaisia sovelluksia ovat mm. erilaiset varoitusjärjestelmät jotka tarkkailevat kaistalla pysymistä sekä viereisten kaistojen liikennettä, erilaiset kaistatasolla toimivat navigointioppaat sekä reaaliaikaisen tietyö- ja ruuhkatilanteen kertovat järjestelmät. [Rogers et al., 1999]

Suomen Tiehallinnolla on meneillään hanke nimeltä Digiroad. Tämän hankkeen yhtenä tarkoituksena on tarjota eri käyttäjäryhmille kattava ja yhtenäinen tietojärjestelmä koko Suomen tieverkosta. Tämä järjestelmä tarjoaisi mm. kaikkien yleisten teiden, katujen, yksityisteiden sekä metsäautoteiden sijainnin n. 1-3 metrin tarkkuudella, tien leveyden, kaistojen lukumäärän, nopeusrajoitukset, sillat ja tunnelit ym. Tämän tason kattavat tiedot mahdollistaisivat hyvinkin hienostuneiden paikannuksen tarkkuutta lisäävien tekoälyjärjestelmien kehittämisen. Digiroad-hankkeen aineisto on suunniteltu julkaistavaksi vuonna 2004. [Tiehallinto, 2003]

2.5. Paikka- ja liiketiedon kerääminen

Ajoneuvon liikkeistä on mahdollista kerätä tietoa hyvin monella tapaa, erilaisia videokameroihin, etäisyysmittareihin ja tunnistimiin perustuvia järjestelmiä on jo käytössäkin. Taksimittaristakin voisi tarvittaessa saada tarkan nopeustiedon digitaalisena. Tässä tapauksessa keskityn kuitenkin pelkästään GPS-laitteen antamiin tietoihin.

GPS-järjestelmä päivittää tiedon sijainnista muutaman sekunnin välein. Tämä viive riippuu pitkälti myös GPS-paikantimen laadusta ja mallista. Näiden tietojen perusteella saadaan laskettua myös keskimääräinen nopeus sekä suunta, mikäli paikannin liikkuu. Ajoneuvokäytössä GPS-järjestelmän viive täytyy ottaa huomioon. Esimerkiksi nopeudella 80 km/h liikkuva auto siirtyy sekunnissa n. 22 metriä. Tällöin karttanäytöllä esitettävä sijainti näyttää liikkuvan jonkin matkaa todellisen sijainnin perässä. Tätä voidaan jossain määrin korjata yksinkertaisella ennakoinnilla.

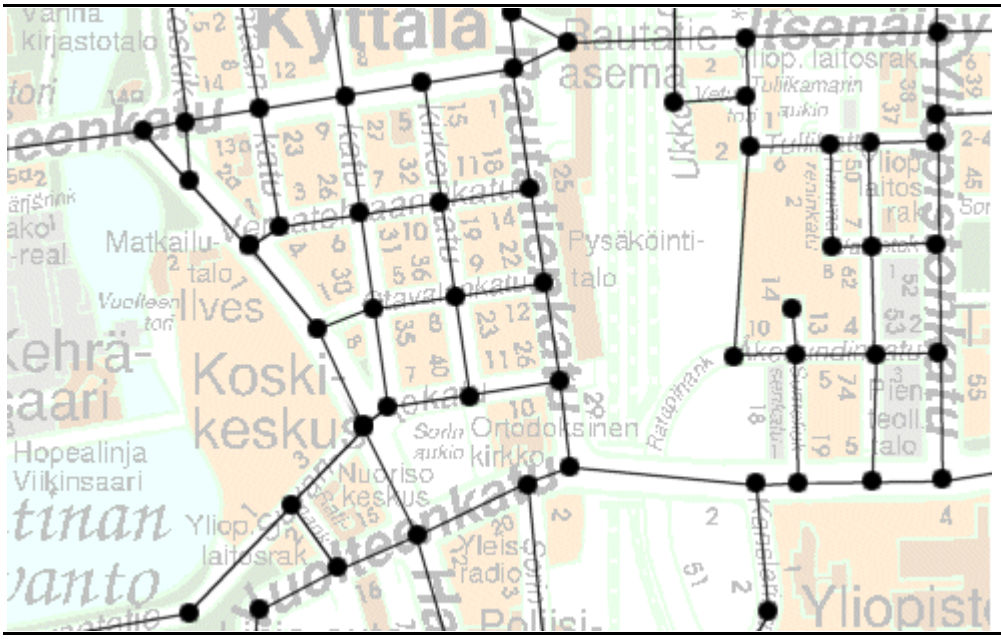
3. Reittitiedon esittäminen graafin avulla

Tässä tutkielmassa graafi tarkoittaa eräänlaista verkkoa, joka muodostetaan tieverkon pohjalta. Graafi koostuu solmuista ja poluista. Solmut kuvaavat risteyksiä ja polut solmujen välisiä yhteyksiä eli käytännössä teitä. Graafin tarkoituksena on mahdollistaa tehokas katujen keskilinjojen etsintä sekä ajoreittien tallennus tietokantaan. Suomen Tiehallinnon Digiroad–hanke olisi myös todennäköisesti jossain määrin tähän tarkoitukseen soveltuva.

3.1. Graafin muodostaminen

Teoriassa karttapohjasta voisi muodostaa katuverkkoa kuvaavan graafin automaattisesti erilaisia kuvantunnistusalgoritmeja hyödyntäen. Käytännössä tuollainen olisi kuitenkin hankalaa, ja saatu tulos olisi vähintäänkin vajaa, ellei jopa virheellinen. Todellisuudessa kaupungista löytyy paljon enemmän laillisia ajoreittejä kuin tarkimpaankaan karttaan on merkitty. Esimerkkeinä mainittakoon vaikkapa parkkihallit, erilaiset parkkialueet tai joutomaat, talojen takapihat ja erilaiset tunnelit. Käsin muodostetun graafin tarkkuuteen on lähes mahdotonta, tai ainakin järjettömän työlästä, päästä automaattisin kuvantunnistusmenetelmin. Paras tulos saadaan siis toistaiseksi käsin piirtämällä ja muokkaamalla, mikä edellyttää myös omakohtaista kokemusta kyseiseltä alueelta.

Olennaista tällä tavalla muodostetussa graafissa on se, että kaikki erilliset ajoreitit eli ainakin kadut, kujat sekä muut kulkukelpoiset ja lain sallimat ajoreitit saadaan eroteltua. Solmujen pitää siis sijaita risteysten keskipisteissä, paikoissa joista on mahdollista kääntyä eri suuntiin, umpikujien päissä tai mutkaisilla teillä niin, että solmujen väliset polut kulkevat riittävän tarkasti keskellä tietä. Liialliseen tarkkuuteen ei kuitenkaan kannata pyrkiä, kaikenlaisia epämääräisesti ajettavia alueita löytyy kaupungeista joka puolelta.



Kuva 2: Katuverkkoa kuvaava graafi

Kartta kuvassa 2 esittää Tampereen rautatieaseman ympäristöä n. 1 km x 0.6 km alueella. Tämä on käytännössä tarkin mittakaava, jolla kaupunkikartat nykyisissä GPS-järjestelmissä esitetään. Seuraava olennainen mittakaavan tarkennus olisi kaistojen sekä rakennusten rajojen tarkka esitys.

Kuvan 2 alueella epämääräisiä alueita ovat mm. Koskikeskuksen, yliopiston sekä rautatieaseman seudut. Tällaisilta alueilta tarkkojen graafien tekeminen ei ole tarkoituksenmukaista. Myös suuremmat risteysalueet voivat olla turhan monimutkaisia esitettäväksi useammilla solmuilla. Tarvittaessa tarkkuutta, jonka perusteella hyväksytään se että GPS-koordinaatti on solmun alueella, voidaan muuttaa. Suurissa ja monimutkaisissa risteyksissä tai epäselvillä alueilla tämä tarkkuus eli suurin hyväksyttävä etäisyys solmun keskipisteestä voi olla suurikin. Nämä tarkkuudet ovat pitkälti tapauskohtaisia ja niiden selvittäminen vaatii myös jossain määrin käytännön kokeiluja. Pääasia on kuitenkin saada selville katujen riittävän tarkat keskilinjat sekä erottaa mahdolliset reittivaihtoehdot.

Jokaisella solmulla on tiedossa ainakin järjestysnumero tai vastaava yksilöllinen tunniste, oma sijaintinsa eli koordinaattinsa sekä kartalla että todellisuudessa, sekä tietenkin yhteydet muihin solmuihin. Yksisuuntaisten katujen tapauksessa siis solmusta A on yhteys solmuun B muttei toisinpäin. Joskin yksisuuntaiset muodostuisivat myös tietyssä määrin itsestään kun dataa ajoreiteistä alkaa kertyä. Erikoistapauksissa kuten umpikujissa, tunneleissa tai suurille parkkialueille päättyvissä solmuissa voidaan käyttää mahdollisesti muitakin merkintöjä auttamaan järjestelmän optimointityötä.

3.2. Ajoreittien tallennus muistiin

Ajoreitit tallennetaan tietokantaan solmujen yhdistelminä. Vähimmäismäärä on kolme solmua, sillä siitä saadaan ainakin selville suunta, josta risteystä lähestyttiin, sekä suunta johon jatkettiin. Kolmen solmun tarkkuus mahdollistaa tarvittaessa myös potentiaalisten pidempien reittien koostamisen. Mikäli reittitiedot tallennetaan neljän tai useamman solmun tarkkuudella, analysoitava tietomäärä voi kasvaa huomattavan korkeaksi, kun mahdollisten reittiyhdistelmien määrä kasvaa eksponentiaalisesti.

Otetaan esimerkiksi teoreettinen katuverkko, joka muistuttaa ruutupaperia. Eli jokaisesta risteyksestä lähtee tie neljään suuntaan eli neljään uuteen risteykseen. Oletetaan myös ettei u-käännöksiä tehdä ja jokainen tie on kaksisuuntainen. Tällöin risteykseen saavuttaessa, on kolme mahdollisuutta jatkaa kolmeen uuteen risteykseen. Kolmen solmun tarkkuudella mahdollisia reittivaihtoehtoja tulee 3 * 3 = 9. Neljällä solmulla 3 * 3 * 3 = 27 ja kuudella solmulla jo $3^5 = 243$.

Se, millä tarkkuudella reitit kannattaa tallentaa, riippuu siis käyttötarkoituksesta ja käytettävän laitteiston rajoituksista. Risteyksiä toki on todellisuudessa monenlaisia, joista useimmista ei pääse kolmeen uuteen suuntaan. Useamman kuin kolmen solmun tarkkuus voi myös tarjota merkittävääkin hyötyä jo pelkästään todennäköisen kääntymissuunnan ennustamisessa. Ainakin taksikäytössä tietyt reitit toistuvat säännöllisesti, työ- ja arkimatkoja ajavista kansalaisista puhumattakaan. Mikäli käytössä on lisätietoa esimerkiksi siitä, mistä matka alkoi ja mihin madollisesti ollaan menossa, saadaan useamman solmun tarkkuudesta suurempaa hyötyä. Tallennettavien solmujen määrä on siis pitkälle tapauskohtainen. Käytännön kokeilla asiasta saisi tietenkin enemmän tietoa.

Jokaisen solmun eli risteyksen keskipiste on siis tiedossa. Tiedossa on myös GPS-järjestelmän antama koordinaatti ja sitä vastaava piste kartalla. Kahden pisteen välinen etäisyys suorakulmaisessa koordinaatistossa määritetään euklidisena etäisyytenä

$$d = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$$

jossa Δx, Δy ja Δz ovat pisteiden koordinaattiarvojen erotukset koordinaatiston akseleilla. Kaksiulotteisen kartan tapauksessa z-koordinaatin arvo on siis 0. Sopiva hyväksyttävä etäisyys solmusta määritellään tapaus- ja sovelluskohtaisesti; keskimäärin se lienee 10-30 metrin luokkaa, kuitenkin niin että vältytään epäselvyyksiltä muiden lähellä sijaitsevien solmujen kanssa.

4. Paikkatiedon tarkentaminen

Seuraavaksi käyn läpi joitakin menetelmiä, joilla esitettävä paikkatieto saadaan vastaamaan paremmin todellisuutta. Mahdollisista häiriötilanteista, kuten pitkien tunneleiden tai korkeiden talojen aiheuttamissa katkoksissa paikannuksessa, voidaan myös jossain määrin selvitä ainakin näennäisesti jatkamalla sijainnin esittämistä hetken aikaa vanhojen tietojen sekä ennakoinnin perusteella.

4.1. Paikkatiedon tarkentaminen liiketiedon perusteella

Kun tiedetään järjestelmän viive, ajoneuvon nopeus sekä suunta, voidaan käyttäjälle esitettävä sijainti piirtää karttanäytöllä tietyn matkaa eteenpäin. Ennakoidun sijainnin etäisyys GPS-järjestelmän sijainnista saadaan, kun lasketaan kyseisen hetken nopeudesta matka joka liikutaan viiveen aikana.
Jos nopeus on a km/h ja GPS-paikantimen viive b s saadaan viipeen aikana kuljettu matka m (metreissä) kaavalla

$$m = (a \cdot 10)/(b \cdot 36).$$

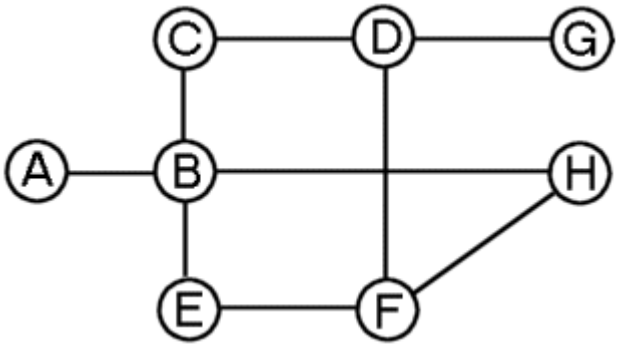
Ennakoidun sijainnin esitettävä paikka karttanäytöllä saadaan siis kun siirrytään matkan m verran GPS-paikantimen sijainnista siihen suuntaan, jonka paikannin ilmoittaa.

Tämän menetelmän käytössä on joitakin ongelmia. GPS-laitteen antama nopeus ja suunta ovat summittaisia arvioita, jotka lasketaan lähes reaaliajassa aikaisempien koordinaattien perusteella. Kun kaupunkiympäristössä ajetaan risteykseen, tapahtuu vauhdin hiljentäminen usein vasta muutamia sekunteja ennen risteystä. Tällöin GPS-paikannin ei pysy tapahtumissa mukana ja pelkästään yllä kuvattua ennakointimenetelmää käyttämällä ennakoitu sijainti näyttää hetken jatkavan matkaansa ohi risteyksen riippumatta siitä, pysähtyikö auto vai kääntyikö se johonkin suuntaan. Kun GPS-paikannin saa viiveen jälkeen taas lukemat vastaamaan paremmin todellisuutta, on auto ehtinyt, mikäli siis kääntyi eikä pysähtynyt, jo ajaa joitakin kymmeniä metrejä johonkin suuntaan. Tällä välin järjestelmän käyttäjälle esitetty sijainti on ollut virheellinen.

4.2. Reittitietokannan hyödyntäminen ennakoinnissa

Reittitietokantaan muodostuu yksinkertaisimmillaan suuri taulukko vähintään kolmen solmun yhdistelmistä. Aina kun jokin auto ajaa hyväksyttävästi tietyn kolmen solmun välin, ajettu reitti tallennetaan tietokantaan. Ei kuitenkaan välttämättä välittömästi vaan sitten, kun se on kannattavinta ottaen huomioon ajoneuvon käyttämän langattoman tiedonsiirtotekniikan. Tällä tavalla

saadaan nopeasti katettua kaikki kaupungin kadut, mikäli tietoa keräämiä autoja on useita. Kun sitten reittitietokantaa ylläpitävään palvelimeen kertyy tarpeeksi tietoa, voidaan sitä ryhtyä analysoimaan.



Kuva 3: Esimerkkigraafi

Kuvassa 3 on esimerkki mahdollisesta katugraafin osasta. Solmujen B-H välillä olisi siis todennäköisesti silta tai tunneli, ja solmu G olisi siis ehkä umpikuja. Oletetaan, että erään kaupungin n. 200 taksiautoa olisivat normaalin työnsä ohella keränneet tietoa ajamistaan reiteistä. Todennäköisesti siis jo kuukaudessa olisi esimerkkigraafinkin jokainen solmu ajettu läpi satoja kertoja. Näin olisi tietokantaan kolmen solmun tarkkuudella muodostunut esimerkiksi seuraavanlaista tietoa:

Reitti	**lukumäärä**
A-B-C	200
A-B-H	1900
A-B-E	400.

Tietokannasta voidaan siis helposti ja keskitetysti laskea ajomäärien perusteella eri reittivaihtoehtojen klassiset todennäköisyydet. Esimerkiksi kun ajetaan suuntaan A-B, todennäköisyys jatkaa suuntaan B-C olisi 200 / (200+1900+400) = 0.08 eli n. 8%. Ja vastaavasti todennäköisyys suuntaan B-H on 1900 / (200+1900+400) = 0.76 eli n. 76%.

Koska tällä tavalla erilaisten reittivaihtoehtojen laskentateho- ja muistivaatimukset siirtyvät ajoneuvolaitteista palvelimille, voidaan tietokantaan tallentaa suuria määriä myös neljän tai useamman solmun sisältäviä polkuja. Tällä tavalla ennustamiseen saadaan entisestäänkin lisää tarkkuutta. Samoin tarkkuutta voidaan teoriassa lisätä ottamalla huomioon myös kellonaika sekä viikonpäivä, sekä laskea todennäköisyydet vain kyseisinä aikoina ajetuilta reiteiltä. Yhteiskunta ja kaupungin liikenne noudattavat omaa rytmiänsä. Päivällä kello 16 aikaan liikenne on usein ruuhkaista ja kaikki liikennevalot todennäköisesti palavat. Tilanne voi olla erilainen reitinvalintojen suhteen vaikkapa arkiyönä kello neljä. Kun tällä tavalla saadaan risteystä lähestyttäessä

todennäköinen kääntymissuunta, voidaan GPS-koordinaatin esitettävää sijaintia optimoida sen mukaan.

4.3. Esitettävän koordinaatin optimointi tien keskipisteeseen

Perusoletuksena on se, että mikäli auto liikkuu, se kulkee tiellä. Kahden solmun välisen suoran viivan pitäisi esittää tien keskiviivaa. Solmujen koordinaattien (x_1,y_1) ja (x_2,y_2) perusteella saadaan kaavalla

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

tämän suoran yhtälö.

Kun auto liikkuu, se kulkee risteysten eli solmujen läpi. Tällöin siis saadaan selville aikaisemmin läpi ajettujen solmujen perusteella myös todennäköiset reitit, johon matkaa jatketaan. GPS-laitteen antamaa koordinaattipistettä verrataan kulloinkin siihen tai niihin polkuihin joissa on mahdollista ja jopa todennäköisintä kulkea. Tällöin yksinkertaisimmillaan karttaan piirrettävän koordinaatin paikka on se piste, joka on lähinnä GPS-koordinaattia ja sijaitsee todennäköisimmällä polulla. Kaavalla

$$d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

saadaan pisteen (x_0, y_0) etäisyys d suorasta $ax + by + c = 0$.

Joissain tapauksissa, varsinkin leveillä monikaistaisilla moottoriteillä voi olla tarkoituksenmukaista piirtää sijainti lähemmäksi omaa kaistaansa. Tällöin esitettävän sijainnin paikasta voidaan laskea tietty keskimääräinen etäisyys tien keskilinjasta. Luonnollisesti jostain pitää saada informaatio, että kyseistä polkua vastaavalla tiellä on useampia kaistoja. Esimerkiksi yleisten liikennesääntöjen perusteella voisi päätellä, että kun liikutaan yli 60 km/h nopeutta, tie on todennäköisesti leveämpi. Myös mahdollisia kääntymissuuntia on monesti useampia ja koska aina ei käännytä todennäköisimpään suuntaan, voi sijainti tietenkin esiintyä väärässä paikassa. Tätä tilannetta pitää tarkkailla ajettaessa risteykseen. Heti kun GPS-laitteesta saadaan uutta suunta- sekä sijaintitietoa, piirrettävää sijaintia tarkistetaan ja se siirretään mahdollisesti uudelle polulle. Edelleen tällaiset menetelmät vaativat tarkempaa tutkimista sekä myös käytännön testejä.

Kokonaisuutena esitetty tarkennustekniikka ei ole virheetön, mutta sen pitäisi olla keskimäärin merkittävästi tarkempi verrattuna puhtaaseen GPS-koordinaatin esittämiseen pelkän yksinkertaisen nopeusennakoinnin kanssa.

5. Yhteenveto

Digitaaliseen karttaan ja satelliittipaikannukseen perustuvan järjestelmän esittämää sijaintia voidaan tarkentaa analysoimalla paremmin kaikkia saatavilla olevia tietoja erilaisilla menetelmillä jotka mahtuvat myös tekoälyn määritelmiin. Tämä tutkielma kartoitti alustavasti tällaisia menetelmiä ja erilaisia huomioon otettavia asioita. Jotta tutkielmassa kuvattujen menetelmien, tai mahdollisten muiden tehokkaampien menetelmien käyttökelpoisuudesta saadaan enemmän tietoa, pitäisi asiaa tutkia enemmän sekä myös käytännön kokein ja sovelluksin.

Kaupunkiliikenteen tuottaman reittidatan tietojen tallettamisesta ja niiden analysoinnista on myös varmasti suurta hyötyä kaupunkien liikennesuunnittelussa sekä katujen kuormitusten arvioinnissa. Tässä tutkielmassa kuvattu graafitekniikka on yksi mahdollinen tapa saada tarkkaa sekä tehokkaasti analysoitavaa tietoa autojen liikkeistä kaupunkien kaduilla.

Viiteluettelo

[Aittola, 2003] Markus Aittola, Location-aware services in mobile devices. University of Oulu, Department of Electrical and Information Engineering. Master of Science Thesis 2003.

[Heimes and Nagel, 2002] F. Heimes, H-H. Nagel, Towards active machine-vision-based driver assistance for urban areas. *International Journal of Computer Vision* **50**, 1, (Oct. 2002), 5-34.

[Kaplan, 1996] Elliott D. Kaplan, *Understanding GPS: Principles and Applications*, Artech House Publishers, 1996

[Rogers et al., 1999] Seth Rogers, Pat Langley and Christopher Wilson, Mining GPS data to augment road models In: *Proc of the fifth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1999, 104 – 113.

[Russell and Norvig, 1995] Stuart J. Russell and Peter Norvig, *Artificial Intelligence : A Modern Approach*, Prentice Hall, 1995.

[Seul et al., 2000] Michael Seul, Lawrence O'Gorman and Michael J. Sammon *Practical Algorithms For Image Analysis*. Cambridge University Press, 2000.

[Tiainen, 2001] Sami Tiainen, Multipath detection and mitigation in GPS. Tampere University of Technology. Master of Science Thesis, 2001.

[Tiehallinto, 2003] Tiehallinto, Digiroad. http://www.tiehallinto.fi/digiroad/. [12.12.2003]

Tietotekniikan merkkipaaluja

Reko Linko

Tiivistelmä

Tämä tutkimus esittelee merkittävimmät tapahtumat tietokoneiden, erilaisten laitteiden ja yleisimpien ohjelmointikielien kehityksessä. Esitys etenee kronologisessa järjestyksessä, alkaen ajasta ennen tietokoneita ja päättyen supertietokoneisiin, joita alkoi ilmestyä 1970-luvulla. Tutkimus osoittaa, että tekninen kehitys on ollut huimaa ja edennyt harppauksin. Lisäksi huomataan, että monet tietojenkäsittelyopin perusajatuksista ovat melko vanhaa perua.

Avainsanat ja -sanonnat: Tietokoneiden historia, FORTRAN, COBOL, BASIC, Babbage, Byron, Hollerith, Zuse, ABC, MARK I, ENIAC, UNIVAC, IBM650, mikrotietokone, Mooren laki, VLSI, supertietokoneet, Cray.
CR-luokat: K.2

1. Johdanto

Vielä viisikymmentä vuotta sitten tietokoneita ei ollut olemassa. Nykyään tietokone löytyy jo joka taloudesta. Jos ei muuten niin mikroprosessorin muodossa. Nykyään monet mahdottomilta tuntuneet tehtävät ovat arkipäivää, ja on huomattava, että enää ei ole edes mahdollista suoriutua tietokoneiden tehtäväksi annetuista laskuista käsin. Ainakaan järjellisessä ajassa. Suurin osa yrityksistä on jo niin riippuvaisia tietokoneista ja tietoteknisistä sovelluksista, että ne eivät selviytyisi päivääkään ilman tietokoneiden apua. Tietotekniikka-alan merkitys lisäksi kasvaa koko ajan yhä suuremmaksi. Tämä tutkimus valottaa toivottavasti hieman miten ja keiden ansiosta nykyiseen tilanteeseen on päästy, tai jouduttu. Kritiikkiä ja pelkoa koneita ja kehitystä vastaan on esitetty jo luddiittien[1] ajoilta. Kritiikki ja pelot eivät aina ole aiheettomia, mutta kehityskulku on mielestäni väistämätöntä.

[1]Nimitys tulee englantilaisesta Ned Luddista, joka apureineen tuhosi työttömyyden pelossa sukkatehtaan koneita 1700-luvulla.

2. Yksittäisten koneiden aika

Tässä luvussa esitellään tärkeimmät henkilöt, joiden voidaan katsoa vaikuttaneen tietojenkäsittelyyn ja sen kehitykseen merkittävästi. Suurin osa esitellyistä henkilöistä ja keksinnöistä ovat luonnollisestikin ulkomaalaisia, johtuen luullakseni suomalaisen yhteiskunnan agraaripainotteisesta yhteiskuntarakenteesta. Muutos alkoi Suomessa toden teolla vasta 1950-luvulla, kun sotakorvauksien maksamiseksi jouduttiin kehittelemään uudenlaista metalli- ja paperiteollisuutta. Suomeen ensimmäinen varsinainen tietokone saatiin vasta vuonna 1957.

2.1. Charles Babbage

Vuonna 1822 englantilainen matemaatikko Charles Babbage esitteli idean automaattisesta laskulaitteesta, differentiaalikoneesta. Babbagen differentiaalikone ei kuitenkaan toiminut, koska sen aikainen mekaniikka ei ollut kyllin kehittynyttä. Toisin sanoen konetta ei voitu valmistaa.

Babbagen piirustusten perusteella rakennettiin vuonna 1991 kone, joka toimi täysin Babbagen suunnittelemalla tavalla. Laskutarkkuudeksi saatiin 31 desimaalia. Kone oli luonnollisesti täysin mekaaninen ja laskutoimitusten suorittamiseksi käyttäjän täytyi väännellä erinäisiä vipuja ja rattaita.

Toinen Babbagen oivallus oli analyyttinen kone. Vuonna 1834 esitellyssä ideassa laskutoimitusten suorittamiseen käytettiin reikäkortteja. Tulokset saatiin iteratiivisesti aiempien laskutoimitusten perusteella. [wikipedia]

2.2. Augusta Ada Byron

Vuonna 1841 Charles Babbage esitteli Torinossa pidetyssä seminaarissa analyyttisen koneensa piirustukset. Ranskalainen Luigi Federico Menabrea kirjoitti ranskaksi tiivistelmän Babbagen aikaansaannoksista.

Tämän tiivistelmän Byron käänsi englanniksi ja ehdotti Babbagelle, että tämä lisäisi käännökseen omia muistiinpanojaan. Muistiinpanot olivat huomattavasti laajemmat kuin Menabrean alkuperäinen teksti. Muistiinpanoissaan Byron esitti, että analyyttistä konetta voitaisiin käyttää minkä tahansa tehtävän suorittamiseen."The Analytical Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it perform." [Toole, 1996] Tämä Byronin lause osoittaa, että Byron loi pohjan käsitteellisen mallintamisen käytöstä tiedon käsittelyssä.

2.3 Herman Hollerith

Vuonna 1884 Hollerith patentoi menetelmän, jonka avulla rei’itetyistä korteista saatu mekaaninen informaatio muutettiin sähköiseksi impulssiksi. Nämä sähköimpulssit taasen muuttivat mekaanisten laskureiden arvoa. [wikipedia]

Todellisen tehonsa Hollerithin reikäkorttikone näytti ensimmäisen kerran vuoden 1890 väestötietojen laskennassa Yhdysvalloissa. Aiemmissa väestönlaskennoissa kerätyt tiedot oli jouduttu laskemaan käsin, ja aikaa kului tolkuttoman paljon. Koneellinen laskenta nopeutti prosessia huomattavasti. Vuoden 1890 väestölaskennan tulokset saatiin laskettua kahdessa ja puolessa vuodessa. Reikäkortteja käytettiin Yhdyvalloissa vielä vuoden 2000 presidentinvaaleissa. Ongelmaksi muodostui tuolloin kuitenkin huonosti rei'itettyjen korttien tulkinta ja laskenta käsin.

Ensimäiset keksijänsä mukaan holleriittikoneiksi nimetyt laiteet tulivat Suomeen 1920-luvulla. Sijoituspaikkana oli Tilastollinen päätoimisto, nykyinen Tilastokeskus. Ensimmäisiä tietojenkäsittelyn parissa työskennelleitä ihmisiä ryhdyttiin kutsumaan "reikäkorttimiehiksi". [Suominen, 2000]

2.4. Konrad Zuse

Sota synnyttää usein uutta teknologiaa ja toimii kimmokkeena uudenlaisille teknologioille. Zuse, joka rakensi Z1 konettaan vuosina 1936-1938, joutui keskeyttämään työnsä jouduttuaan kutsuntoihin ja armeijaan. Varsinaisiin sotatoimiin Zuse ei joutunut, koska hänet napattiin valmistamaan ohjusten ohjausjärjestelmiä.

Z1:n seuraajaa Z3:a pidetään ensimmäisenä ohjelmoitavana tietokoneena maailmassa. Z3 valmistui vuonna 1941. Se sisälsi kaksi mullistavaa ideaa. Ensiksikin aikaisempiin heksadesimaaliesitykseen perustuneisiin koneisiin verrattuna kone toimi binääriluvuilla. Toiseksi koneen laskenta- ja kontrolliyksiköt olivat erillään muistiyksiköstä. Tästä yksiköstä Zuse käytti termiä Speicher, eikä von Neumannin ehdottamaa termiä memory. [Rojas, 1996] Z3 koneessaan Zuse korvasi mekaaniset laitteet sähkömagneettisella releillä. Zusea pidetään myös myös ensimmäisenä, joka kehitti ns. korkean tason ohjelmointikielen, Plankalkülin. Ohjelmointikieli ei tullut kovin tunnetuksi, johtuen varmaankin Kolmannen Valtakunnan tuhosta ja sen jälkeisestä epäsuosiosta kaikkea saksalaista kohtaan.

2.5. Atanasoffin ABC Computer

Jotkut pitävät John Vincent Atanasoffin ja Clifford Berryn tekemää Atanasoff Berry Computeria (ABC) ensimmäisenä varsinaisena tietokoneena. ABC-

koneen perusideoita olivat Zusen tavoin binäärijärjestelmän käyttäminen laskutoimituksissa kymmenjärjestelmän sijaan. Toinen pääkohta oli aikaisemmista laskulaitteista poiketen sähköinen, ei mekaaninen, tiedonsiirto.

Atanasoffin ja Berryn hahmottelemassa ja tekemässä koneessa oli Zusen koneen tavoin ulkoinen muistiyksikkö ja laskutoimitukset perustuivat loogiseen päättelyyn mekaanisten laskutoimitusten sijaan. Vallankumouksellista ABC-koneessa oli kuitenkin elektroniputkien käyttö. Ideat olivat kuitenkin hyvin samankaltaisia Zusen koneen kanssa, joten nostaisin Zusen Z1-koneen ABC:n edelle.

2.6. MARK I

Samaan aikaan kun Zuse rakensi Z3-konettaan, tekivät amerikkalaiset omia kokeilujaan, jotka osoittautuivat myöhemmin hyvinkin samankaltaisiksi Zusen ratkaisujen kanssa.

Harvardin yliopiston professori Mark Aiken kehitti Mark I -nimisen koneen, joka valmistui vuonna 1944. [wikipedia] Kuten muissakin relekoneissa, niin myös Mark I:ssä laskentaan tarvittava informaatio syötettiin reikäkorttien avulla. Laskutoimitukset suoritettiin yksitellen sähkömekaanisilla laitteilla, releillä, ja tulokset annettiin lävistettyinä reikäkortteina. Koneen toiminta-ohjeet taasen annettiin nauhapaperilla, josta kone luki yhden ohjeen kerrallaan.

2.7. ENIAC ja seuraajat

ENIAC (Electronic Numerical Integrator And Computer) oli maailman ensimmäinen varsinainen elektroninen tietokone. Ongelmana ENIAC:issa oli kuitenkin sen käytön hitaus. Koodi kuitenkin syötettiin ohjaustaulun kautta ja ohjelma suoritettiin yksi toiminto kerrallaan. Kun kone oli suorittanut toiminnon, jouduttiin se ohjelmoimaan mekaanisesti uudelleen. Tämä vei luonnollisestikin kohtuuttomasti aikaa ja vaivaa.

ENIACIN seuraajat EDVAC (Electronic Discrete Variable Automatic Computer) [wikipedia] ja EDSAC (Electronic Delay Storage Automatic Computer) [wikipedia] erosivat ENIAC:ista siten, että niille voitiin antaa useampi toimintakäsky kerrallaan. Toisin sanoen voitiin ohjelmoida kone suorittamaan useita toimintoja peräkkäin. Teknisesti ohjelmointi toteutettiin siten, että kaikki tarvittavat käskyt sijoitettiin niin sanottuun akustiseen viivemuistiin. Näin syntyi varastoidun ohjelman käsite.

3. Sarjavalmisteiset koneet

Edellisessä luvussa esitellyt koneet olivat kaikki niin sanottuja laboratoriokoneita, eli niitä valmistettiin kutakin tyyppiä yksi tai korkeintaan muutama kappale. Käyttöalueena olivat pääasiassa yliopistot ja sovellusalueet olivat matemaattisvoittoisia. Myös sotateollisuus oli tärkeä ensimmäisten tietokoneiden käyttäjä. Tietokoneita käytettiin etenkin ammusten lentoratojen laskentaan, muun muassa Manhattan-projektissa, kun kehitettiin ensimmäinen ydinpommi.

John von Neumann esitti kuitenkin, että tietokoneilla voisi olla muitakin käyttöalueita kuin matemaattisten ongelmien laskeminen. Tästä seurasi, että muun muassa liike-elämä, pankit, tavaratalot ja teollisuuslaitokset alkoivat kiinnostua tietokoneista ja niiden luomista mahdollisuuksista. Seurauksena oli tietokoneiden sarjavalmistuksen alkaminen 1950-luvun alkupuolella.

3.1. Ensimmäinen sukupolvi

Sarjavalmisteisten koneiden ensimmäisen sukupolven tietokoneille tyypillisiä piirteitä olivat muun muassa muistin lisääntyminen ja tallennetun ohjelman käsitteen syntyminen. Koneiden teho ja useiden käskyjen antaminen kerralla syrjäytti reikäkorttiteknologian nopeasti.

Ensimmäisen sukupolven koneilla oli kuitenkin nykyaikaa ajatellen monia rajoitteita. Ensinnäkin koneet olivat erittäin isokokoisia ja koneissa oli valtava määrä elektroniputkia. Käytössä putket kuumenivat ja näin ollen tarvittiin jatkuvaa valvontaa ja tehokasta jäähdytystä. Putket rikkoituivat helposti. Tämä johti siihen, että koneiden luotettavuus oli melko kehno. Rikkoutumisen sattuessa ei ehkä saatu oikeita tuloksia, jos saatiin tuloksia lainkaan. Kone jouduttiin ohjelmoimaan jokaista tehtävää varten uudelleen ja ohjelmointi oli konekohtaista ja aikaa vievää. Sen aikaisilta ohjelmoijilta vaadittiin siis valtava määrä konekohtaista nippelitietoa ja pitkien konekohtaisten koodirivien muistamista. Tämä johtui siitä, että koneet ohjelmoitiin konekielellä. Virheiden löytäminen koneelle annetuista ohjeista jouduttiin tekemään käsky kerrallaan. Jos ja kun virhe löytyi, jouduttiin jokaista muutakin käskyä muuttamaan, jotta kone olisi toiminut halutulla tavalla.

3.1.1. UNIVAC I

ENIACin tekijät John Presper Eckert ja John Mauchly huomasivat ehkäpä ensimmäisinä tietokoneiden kaupallisen potentiaalin. Yhdessä he perustivat Eckert-Mauchly Computer Corporation -nimisen yrityksen ja aloittivat UNIVAC I (UNIVersal Automatic Computer) nimisen koneen suunnittelun.

Rahoitusta oli kuitenkin vaikea löytää ja lopulta Eckert-Mauchly Computer Corporation myytiin Remington Randille. Remington Rand vei UNIVACI:n kehittelyn loppuun. Ensimmäinen Univac tietokone (Univac I) valmistui kesällä 1951. Ensimmäinen osoitus tietokoneen mahdollisuuksista oli vuoden 1952 Yhdysvaltain presidentinvaalien lopputuloksen ennustaminen oikein hyvin pienestä otoksesta ääniä. Ennustetta ei kuitenkaan julkaistu ennen varsinaisia tuloksia, koska ei uskottu, että kone voisi olla oikeassa. [wikipedia]

3.1.2. IBM 650

International Business Machines, IBM, hallitsi reikäkorttikoneiden markkinoita jo 1950-luvun alussa. Yrityksen perustaja Thomas Watson Sr. oli aluksi vastaan ajatusta siirtymisestä tietokoneiden valmistukseen.

Kun IBM:n johtoon siirtyi hänen poikansa Thomas Watson Jr., asenne yrityksessä muuttui. [wikipedia] Vuonna 1952 IBM toimitti Yhdysvaltain hallitukselle tietokoneen, jonka tyyppimerkintä oli 701. Vuotta myöhemmin esiteltiin IBM 650. Koska IBM oli markkinajohtaja reikäorttikoneiden alalla, oli IBM 650 -tietokoneiden myynti toimivan markkinointikoneiston ja hyvän maineen ansiosta valtava. IBM 650 -koneita myytiin alkuperäisesti suunnitellun 50 sijasta yli 2000. Myös Suomen ensimmäinen varsinainen tietokone, Postisäästöpankin vuonna 1958 hankkima "Ensi", oli IBM 650 -sarjaa. [Tienari, 1993]

3.2. Toinen sukupolvi

Elektroniputket osoittautuivat melko kalliiksi vaihtoedoksi tietokoneissa. Niitä oli myös hankala käyttää. Toisen sukupolven tietokoneiden ajan voidaan katsoa alkaneen transistorien käyttöönoton jälkeen tietokoneteollisuudessa.

Transistori keksittiin vuonna 1947 Bell yhtiön laboratorioissa. Keksijät saivat vuoden 1956 Nobelin fysiikan palkinnon. [wikipedia] Puolijohdeteknologian myötä saavutettiin huomattavia parannuksia tietokoneiden suorituskykyyn ja yleisiin ominaisuuksiin. Elektroniputkiin verrattuna transistorit olivat huomattavasti pienempiä ja nopeampia. Lisäksi ne olivat luotettavampia ja ennen kaikkea halvempia kuin elektroniputket.

3.2.1. Toisen sukupolven erityispiirteitä

Eräs toisen sukupolven koneisiin liittyvä merkittävä asia oli eräajo. Tämä tarkoitti sitä, että yksittäinen ohjelma tai yksittäinen tehtävä suoritettiin tietokoneella loppuun asti ennen seuraavan tehtävän tai ohjelman suorittamista.

Toinen huomioitava seikka toisen sukupolven koneissa oli uudenlaisten ohjelmointikielien ilmaantuminen.

Koska konekielinen ohjelmointi oli työlästä, haluttiin aikaansaada muutos. Ryhdyttiin käyttämään symbolista notaatiota. Tämä tarkoitti sitä, että konekielinen käsky korvattiin symbolilla. Koska symboliset käskyt haluttiin saada koneen ymmärtämään muotoon, piti käskyt koostaa. Tähän tarkoitukseen tehtiin kokoojaohjelmia, assemblereita.

3.2.2. FORTRAN

Ensimmäinen korkean tason ohjelmointikieli oli FORTRAN, FORmula TRANslation. Kieli kehitettiin tieteellisten ja matemaattisten tutkimuskohteiden ohjelmointikieleksi. Kieli julkaistiin 1957 IBM:n työryhmän johdolla. Ryhmän vetäjänä oli John Bachus.

FORTRAN suunniteltiin riippumattomaksi koneympäristöstä, ettei ohjelmoijan tarvitsisi tietää kunkin koneen erityispiirteitä. Riitti, että ohjelmoija osasi esittää ohjeet matemaattisessa esitysmuodossa. Tämän jälkeen kääntäjäohjelma käänsi koodin konekielelle. Ohjelmakatkelma 1 on esimerkki FORTRAN-kielisestä ohjelmasta. [wikipedia]

```
      PROGRAM HELLO
      WRITE(*,10)
10 FORMAT('Hello, world!')
      STOP
      END
```

Ohjelmakatkelma 1. FORTRAN-kielinen Hello World -ohjelma.

3.2.3. COBOL

Toinen merkittävä ohjelmointikieli tietokoneiden toisen sukupolven ajalta oli COBOL, (COmmon Business Oriented Language). [wikipedia]

Kuten nimestäkin jo voi päätellä, kieli kehitettiin kaupallisia sovelluksia ajatellen. Merkittävin uudistus oli englantia muistuttavien sanojen käyttö ohjelmoinnissa. Lisäksi koodi voitiin tulkita ja ajaa monen eri laitevalmistajan koneilla. COBOL generoi myös itse dokumentaationsa. COBOLissa oli myös varattuja sanoja, kuten esimerkiksi Javassa nykyään. COBOL -ohjelmien rakenne oli myös hyvin hierarkkinen. Tasot olivat jakso (division), lohko (section), kappale (paragraph) ja kohta (entry). Pääjakso koostui lohkoista, lohkot kappaleista ja kappaleet kohdista. Pääjaksoja oli neljää eri tyyppiä: tunnistus-, käyttöympäristö-, aineisto- ja prosessijakso. Tunnistuslohkossa esi-

tettiin tunnistetiedot, kuten ohjelman nimi, päiväys ja ohjelmoijan nimi. Käyttöympäristöjaksossa esiteltiin kone, jolla ohjelma suoritettiin ja esimerkiksi se, miten ohjelma tulostettiin. Aineistojaksossa kuvattiin tietotyypit ja niiden käyttäytyminen. Prosessijaksossa kuvattiin se, miten ohjelma eteni. Ohjelmakatkelma 2 on esimerkki COBOL-kielisestä ohjelmasta. [wikipedia]

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.          HELLOWORLD.
000300 DATE-WRITTEN.        02/05/96    21:04.
000400*                    AUTHOR BRIAN COLLINS
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100100
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400         DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100500         DISPLAY "HELLO, WORLD!" LINE 15
POSITION 10.
100600         STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800         EXIT.
```

Ohjelmakatkelma 2. COBOL-kielinen Hello World! -ohjelma.

3.3. Kolmas ja neljäs sukupolvi

Kolmannen sukupolven merkittävin uutuus oli mikroprosessorin tulo markkinoille. Uusi tekniikka oli halvempaa ja luotettavampaa kuin vanha. Toinen merkittävä seikka kolmannen sukupolven aikana oli päätetyöskentelyn lisääntyminen.

Päätetyöskentely oli mahdollista, koska koneilla pystyttiin käsittelemään yhtäaikaisesti monia ohjelmia ja käyttöaikaa voitiin jakaa käyttäjien kesken. Lisäksi monia eri tietokonemerkkejä ja -malleja voitiin standardoinnin avulla

sovittaa yhteen. Kolmannen sukupolven aikana kehitettiin myös käyttöjärjestelmän käsite. Korkean tason ohjelmointikielien lisääntyminen ja kehittyminen olivat myös tämän sukupolven tunnusmerkkejä.

3.3.1. Mikrotietokoneet

Ensimmäinen kaupallinen mikroprosessori oli marraskuussa 1971 julkistettu 4004. [wikipedia] Suurin mikroprosesssorien valmistaja oli Intel. Mikroprosessorin julkistamisen jälkeen alkoi markkinoille ilmestyä mikrotietokoneita.

Mikrotietokone näytti tuolloin hyvin paljon kirjoituskoneelta, joka oli yhdistetty johdoilla laatikkoon. Näppäimistö oli pääte, jonka avulla ohjeet syötettiin koneelle. Laatikko taas oli itse tietokone, jossa data ja ohjelma sijaitsivat. Syötteet olivat melko ymmärrettävässä muodossa annettuja käskyjä, kuten RUN, GO TO, NEXT ja niin edelleen. Käyttäjälle tietokoneen tulokset esitettiin yksinkertaisessa ja ymmärrettävässä muodossa. Tästä seurasi se, että käyttääkseen tietokonetta ei enää tarvinnut olla teknisesti lahjakas tai näppärä.

Mikrotietokoneiden markkinoita hallitsi aluksi Apple. Muita merkkejä olivat muun muassa Tandy, Radio Shack, Sinclair ja Commodore. Vuonna 1981 IBM julkaisi IBM PC:n, ja sille VisiCalc ohjelman. [wikipedia] VisiCalc oli ensimmäinen kaupallinen taulukkolaskentaohjelma. Kotiin hankittujen tietokoneiden määrä kasvoi räjähdysmäisesti 1980-luvun alussa. Koneiden hankintaa perusteltiin yleensä järkisyilllä. Mielestäni suurin osa sen aikaisista tietokoneista päätyi kuitenkin pelikoneiksi ja sittemmin pölyyntymään komeroon. IBM-kloonien ja graafisen käyttöliittymän yleistyttyä löytyi tietokoneille muutakin käyttöä kuin pelaaminen. Taulukkolaskenta, tekstinkäsittely, erilaiset piirto-ohjelmat ja kortisto-ohjelmat auttoivat ihmisiä perustelemaan tietokoneen hankintaa kotiin. Nykyään tietokone kotona on jo melkein itsestäänselvyys.

3.3.2. BASIC

Merkittävin kolmannen sukupolven ohjelmointikielistä oli BASIC (Beginner's All-purpose Symbolic Instruction Code). Kieli oli erityisen suosittu mikro- ja pientietokoneissa. Kieli oli helppo oppia ja virheiden etsiminen koodista ja niiden korjaaminen oli myös vaivatonta.

BASIC oli melko yksinkertainen korkeamman tason ohjelmointikieli, sillä siinä oli melko vähän muistettavia sääntöjä. Koodi oli sekoitus englantia muistuttavia sanoja ja aritmeettisia symboleja. Sen avulla voitiin antaa käskyt koneelle reaaliajassa. Jos tuli virhe, niin kone ilmoitti, missä kohtaa koodia virhe sijaitsi. Ohjelmakatkelma 3 on esimerkki BASIC -kielisestä ohjelmasta.

```
10 LET A=5
20 LET B=6
30 LET C=A*B
40 PRINT A,B,C
50 END
```

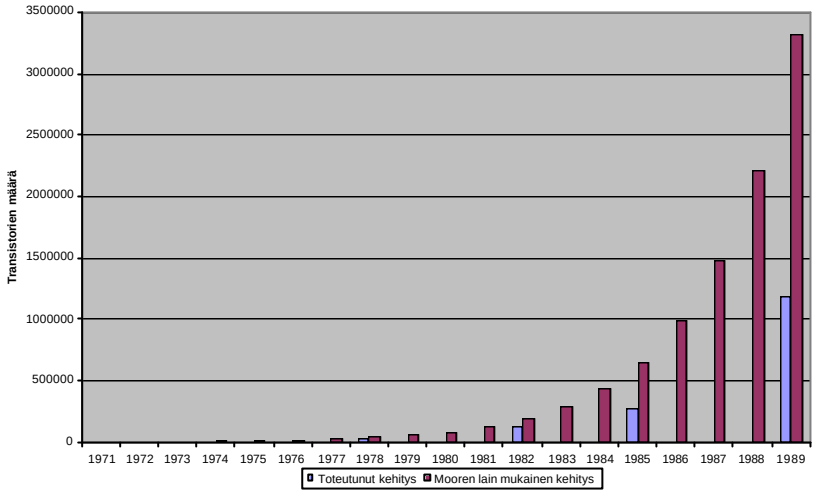
Ohjelmakatkelma 3 esimerkki BASIC -kielisestä ohjelmasta

Ohjelmakatkelman 3 ohjelma esittelee muuttujan A rivillä 10, muuttujan arvoksi asetetaan 5. Muuttuja B esitellään rivillä 20 ja sen arvoksi asetetaan 6. Rivillä 30 esitellään muuttuja C ja se saa arvon A*B, eli 30. Rivilla 40 tulostetaan luvut 5, 6, 30. Rivillä 50 lopetetaan ohjelma.

3.3.3. VLSI

VLSI (Very Large-Scale Intergration) tarkoitti sitä, että yhä pienemmälle pinta-alalle pystyttiin juottamaan yhä enemmän komponentteja. Tämä merkitsi koneiden laskentatehon kasvua ja massatuotannon kautta halvempia hintoja. Ensimmäisiä VLSI-tekniikkaa olevia piirejä oli Intel 4004, jossa transistoreita oli 2250. [Intel]

Koneiden tehon kasvua kuvaa ns. Mooren laki. Lain perusteella transistorien määrä kaksinkertaistuu 18 kuukauden välein. Lain perusteella voidaan siis päätellä, että koneiden teho kasvaa eksponentiaalisesti. Lain perusteella rajaton kasvu on mahdollista. Kuva 1 osoittaa, että ainakin lyhyellä ajanjaksolla tarkasteltuna laki ei pidä paikkansa. Kuvasta 1 huomataan, että Mooren hypoteesi ei pitänyt paikkansa ainakaan vuosina 1971 - 1989. Käänne tapahtui jo 1980-luvun alussa.



Kuva 1. Mooren lain mukainen kehitys vs. toteutunut kehitys.[Intel]

3.3.4. DBMS

DBMS (Data Base Management System) oli seuraava uudistus tällä aikakaudella. Tietokantaohjelmistot mahdollistivat informaation keskitetyn hallinnan. Dataa voitiin varastoida tietokantaan ja tehdä kyselyitä samasta kannasta.

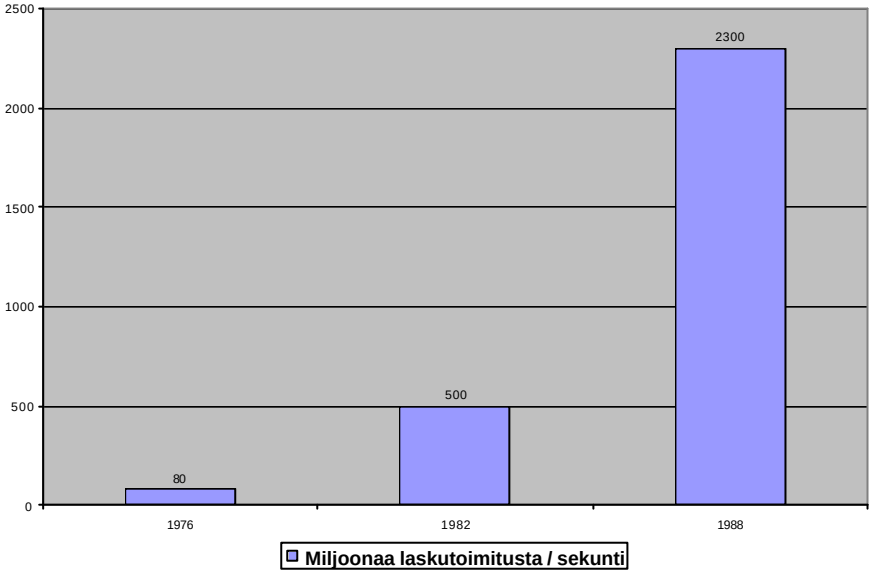
Ennen tietokanta-arkkitehtuurien olemassaoloa jouduttiin miettimään hyvinkin tarkkaan, mitä halutaan tehdä. Tietokanta-arkkitehtuurit mahdollistivat datan dynaamisen hallinnan. Enää ei tarvinnut miettiä, missä ja millaisessa muodossa data sijaitsi. Kyselykielien avulla tietoihin pääsi helposti käsiksi, vaikkei olisi osannut ohjelmoida. Itse ohjelmakoodi ja data olivat siis riippumattomia toisistaan

Päällekkäisen, redundantin, datan määrä väheni myös huomattavasti. Data varastoitiin ainoastaan, jos se muuttui. Tästä seurasi kustannusten lasku, koska dataa ei tarvinnut enää tallentaa joka kerta uuteen paikkaan ja näin ollen kuluttaa arvokasta muistitilaa. Dataa ei myöskään voitu vahingossa muuttaa, koska tietokantoihin ei päässyt käsiksi suoraan, vaan ainoastaan sovelluksen avulla. Sovelluksessa oli, ja on yleensäkin käyttäjän autentikointi, tunnistus. Tästä seurasi myös se, että data oli paremmin suojattua kuin aikaisemmin. Jos data kuitenkin jostain syystä vaurioitui, oli käytössä yleensä varmistusjärjestelmä, kuten esimerkiksi nauhalle tallennetut tiedot. Ensimmäisiä varsinaisia kyselykieliä olivat IQF (Interactive Query Facility) ja GIS/2 (Generalized Information System/2). [wikipedia]

3.3.5. Supertietokoneet

1970-luvun alussa ilmestyivät ensimmäiset niin sanotut supertietokoneet. Supertietokoneita käytettiin etenkin sääennusteiden tekoon, ydinräjähdys-simulaatioihin ja maanjäristysten mallintamiseen.

Supertietokoneista puhuttaessa nousee nimi Cray yli muiden. Seymour Cray perusti vuonna 1972 Cray Research, Inc. -yhtiö. Yhtiön ensimmäinen supertietokone Cray 1 toimitettiin asiakkaalle vuonna 1976. [wikipedia] Nykyisiin kotitietokoneisiin verrattuna Cray 1, sen aikainen supertietokone, oli laskentakyvyltään melko vaatimaton. Sen laskentateho oli 80 miljoonaa laskutoimitusta sekunnissa. Esimerkiksi Pentium 4 -prosessorin laskentateho on noin 1700 miljoonaa laskutoimitusta sekunnissa. Kuvassa 2 on havainnollistettu supertietokoneiden laskentatehon kehitystä. [Cray]



Kuva 2. Supertietokoneiden laskentatehon kehitys. [Cray]

4. Yhteenveto

Ensimmäiset mekaaniset laskulaitteet olivat helmitaulut ja muut samantyyppiset konstruktiot. 1800-luvulla Charles Babbage konstruoi differentiaalikoneensa, joka ei kuitenkaan koskaan nähnyt päivänvaloa. Syynä oli tekniset rajoitteet. Ajatus tietokoneesta oli kuitenkin syntynyt. Babbagen ideoiden pohjalta Augusta Ada Byron esitti ajatuksen tietokoneesta ja sen kaikkivoipaisuudesta.

1800-luvun lopulla Herman Hollerith aloitti reikäkorttikoneiden aikakauden, joka kesti pitkälle 1950-luvulle asti. Ensimmäiset elektroniset koneet ideoitiin 1940-luvulla. Asialla olivat Atanasoff, Aiken ja Zuse. Ensimmäinen

täysin elektroninen kone, ENIAC, valmistui 1946. Tällä aikakaudella tietokoneita käyttivät lähinnä yliopistot ja armeija.

1950-luvulla kauppa ja teollisuus otti tietokoneet käyttöönsä. ENIACin kehittäjät perustivat yhtiön, joka valmisti UNIVAC I -koneen. Pian tämän jälkeen IBM ryhtyi valmistamaan tietokoneita. IBM saavutti nopeasti markkinajohtajan aseman. Seuraava suuri edistysaskel oli transistorin keksiminen ja korkean tason ohjelmointikielien ilmaantuminen.

Mikropiiri ilmestyi tietokoneisiin IBM 360 System -koneen myötä 1964. 1970-luvulla VLSI-tekniikan tulo merkitsi yhä nopeampaa ja varmempaa tietojenkäsittelyä. Samaan aikaan muistien hinta laski huomattavasti. Myös tietokantaohjelmistot näkivät päivänvalon 1970-luvulla. 1970-luvulla ilmaantuivat myös ensimmäiset supertietokoneet ja tekstinkäsittelyohjelmat. 1970-luvun lopulla ilmestyivät mikrotietokoneet. Markkinoita hallitsi aluksi Apple. 1981 IBM julkisti ensimmäisen IBM PC:n ja taulukkolaskentaohjelman ja osin siitä johtuen alkoi IBM-koneiden ja kloonien voittokulku.

Viiteluettelo

[Cray, 2003] History of Cray Inc. http://www.cray.com/company/history.html

[Intel, 2003] Intel Research - Silicon - Moore's Law http://www.intel.com/research/silicon/mooreslaw.htm

[Rojas, 1997] Raúl Rojas, Konrad Zuse's legacy: the architecture of the Z1 and Z3 . *IEEE Ann. Hist. Comput.* **19**, 2 (1997) 5-16.

[Suominen, 2000] Jaakko Suominen, *Sähköaivo sinuiksi, tietokone tutuksi tietotekniikan kulttuurihistoriaa.* Jyväskylän yliopisto, 2000.

[Tienari, 1993] Martti Tienari, *Tietotekniikan alkuvuodet Suomessa.* Suomen atk-kustannus, Espoo, 1993.

[Toole, 1996] Betty Alexandra Toole, Ada Byron, Lady Lovelace, An analyst and metaphycican. *IEEE Ann. Hist. Comput.* **18**, 3 (1996) 4-12.

[wikipedia, 2003] Main Page - Wikipedia. http://en2.wikipedia.org/wiki

GPRS-verkkojen tietoturva

Jyrki Rantanen

Tiivistelmä

GPRS-verkko on kehitetty GSM-verkon pohjalta, jotta pystyttäisiin tarjoamaan langattomille päätelaitteille nopeampi internet-yhteys. Vanha GSM-data ei pysty tarjoamaan riittävää nopeutta nykyajan multimediasovelluksille. Täysin uudenlainen teknologia olisi vaatinut uuden verkon sekä uuden päätelaitekannan. Tähän ei ollut vielä resursseja, mutta nopean internet-yhteyden tarpeen kasvaessa kehitettiin vanhaa järjestelmää pohjanaan käyttävä GPRS. Tutkielmassani tuon ilmi tietoturvaan liittyviä ongelmia, jotka johtuvat siitä, että GSM-verkko on piirikytkentäinen, kun taas internet, johon sillä ollaan yhteydessä, on pakettikytkentäinen. Tästä johtuen GPRS-verkossa liikkuvalle datalle tarvitsee tehdä tyyppimuunnos verkkojen välissä. Pyrin havainnollistamaan siitä johtuvia tietoturvaan liittyviä ongelmia sekä esittelemään joihinkin niistä olemassa olevia ratkaisuja.

Avainsanat ja -sanonnat: GPRS, Tietoturva
CR-luokat: C.2.1, C.2.2, C.4

1. Johdanto

1.1. GPRS-teknologian taustaa

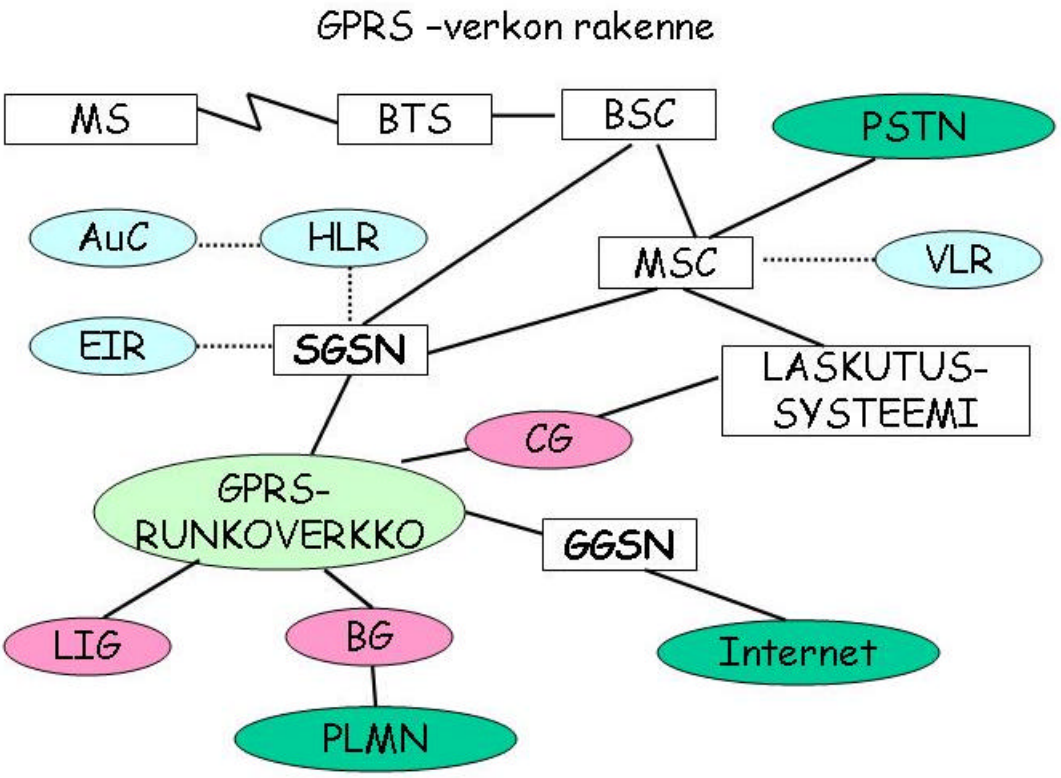
Länsimaissa lähes kaikilla ihmisillä on pääsy internetiin. Päivittäisiä asioita hoidetaan kiireisessä yhteiskunnassa yhä useammin sähköisessä muodossa. Tämä on saanut aikaan sen, että mm. sähköposti on muodostunut tärkeäksi osaksi työtä ja vapaa-aikaa. Samalla raja työn ja vapaa-ajan välillä on hämärtynyt. Työasioita hoidetaan paikasta ja ajankohdasta riippumatta. Sähköisen tiedon määrän kasvaessa on ollut pakko kehittää nopeampi tapoja siirtää dataa langattomasti paikasta toiseen. Tätä varten on kehitetty langattomia teknologioita, kuten GSM-data ja GPRS. Toisaalta tiedon määrän kasvaessa tietoturvan rooli korostuu entisestään.

2. GPRS-verkon rakenne

2.1. GPRS-verkon esittely

GPRS-verkon perusrakenne on sama kuin GSM-verkossa, mutta rakenne on kuitenkin mutkistunut huomattavasti. Keskeisimpiä uusia komponentteja ovat SGSN ja GGSN. Puhe kulkee edelleen samaa reittiä kuin GSM-verkossa, mutta GPRS-data lähetetään SGSN:lle (Serving GPRS Support Node), joka reitittää liikenteen runkoverkon (GPRS Backbone) kautta GGSN:ään. GGSN on liitetty ulkoisiin pakettiverkkoihin vaikkapa internetiin. GPRS tukee IP-ja X.25 verkkoja, jolloin se voi toimia yhdyskäytävänä myös muihin verkkoihin. [Vesanen, 2003]

GPRS (General Packet Radio Service) on pakettikytkentäinen verkkoteknologia, joka toimii GSM-verkon päällä. GSM-datan maksimisiirtonopeus on 14,4kb/s, kun GPRS-verkossa siirtonopeus voi olla teoriassa jopa 170kb/s. Verkko tarjoaa käyttäjälle niin sanotun aina auki olevan yhteyden internetiin ja intranetiin. Suurimmat hyödyt GSM-verkkoon verrattuna ovatkin yhteyden muodostamisen nopeus sekä suurempi datan siirtonopeus. Datan siirto tapahtuu fyysisesti kuten GSM:ssä, mutta GPRS voi käyttää samasta TDMA-kehyksestä useita aikavälejä.



Kuva 1. GPRS-verkon rakenne [Vesanen, 2003]

2.2. GPRS-verkon osat

Mobile station tarkoittaa päätelaitetta, jolla verkkoa käytetään. Se voi olla esimerkiksi kämmentietokone. Päätelaiteen ja verkon ominaisuuksista riippuen päätelaite voi olla kolmessa eri tilassa:

Luokka A sallii päätelaitteen suorittaa pakettikytkentäistä liikennettä samaan aikaa, kuin se on piirikytkentäisessä CS (Circuit Switched) yhteydessä puhelua varten.

Luokka B sallii päätelaitteen kytkeytyä molempiin verkkoihin, niin CS kuin PS (Packet Switched), mutta se ei voi käyttää niitä yhtä aikaa. Kuitenkin päätelaitteen ollessa pakettikytkentäisessä tilassa se voi vastaanottaa kutsun piirikytkentäisesti liikenteestä ja keskeyttää pakettikytkentäisen liikenteen sekä jatkaa sitä piirikytkentäisen liikenteen päätyttyä.

Luokka C sallii päätelaitteen kytkeytyä vain toiseen verkkoon. Käytännössä tämä tarkoittaa laitetta, joka on tarkoitettu vain pakettikytkentäiseen liikenteeseen.

BSS (Base Station System) pitää sisällään kaksi verkon osaa BSC (Base Station Controller) ja BTS (Base Transceiver Station).

Base Transceiver Station on radio lähetin-vastaanotin , joka mahdollistaa BSC:n kommunikoinnin MSC:N ja SSGN:n kanssa riippuen siitä millaisesta liikenteestä on kyse. Yksi BSC kontrolloi ryhmää BTS:ta.

MSC (Mobile Services Switching Centre), huolehtii kaikesta piirikytkentäisestä liikenteestä. Se vastaa puheluiden käsittelystä sekä ohjaa muuta mahdollista piirikytkentäistä liikennettä.

SGSN (Serving GPRS Support Node) on yksi GPRS-verkon tärkeimmistä komponenteista. Sen päätehtävänä on reitittää pakettikytkentäistä liikennettä päätelaitteilta oikealle GGSN:lle (Gateway GPRS Support Node) ja päinvastoin palvelualueellaan. Lisäksi sen vastuulla on suorittaa käyttäjän tunnistaminen AuC:lta (Authentication Center) saatujen tietojen avulla, pitää yllä sijaintitietoja, huolehti loogisesta linkityksestä ja luoda laskutukseen tarvittavat tiedot.

EIR (Equipment Identify Register) on tietokanta, joka pitää kirjaa päätelaitteiden identifiointiin käytettävistä tiedoista.

HLR (Home Location Register) on tietokanta, jossa säilytetään verkon käyttöön oikeutettujen käyttäjien verkon toiminnan kannalta oleelliset tiedot, kuten tunnistamiseen käytettävät tiedot. Siellä on myös tiedot palvelun tarjoajan nimeämistä yhdyskäytävistä sekä mahdollinen tieto staattisesta IP-osoitteesta. Lisäksi HLR pitää kirjaa SGSN:n lähettämistä päätelaitteen sijaintitiedoista.

AuC (Authentication Center) toimittaa HLR:lle tarvittavat tiedot verkossa liikkuvan datan salaukseen ja purkamiseen. AuC on yleensä osa HLR:ä.

VLR (Visiting Location Register) pitää kirjaa verkossa vierailevien muiden operaattorien päätelaitteista. VLR on usein osa MSC piirikytkentäisessä verkossa ja osa SGSN pakettikytkentäisessä verkossa.

GPRS-runkoverkon tehtävä on yhdistää SGSN ja GGSN. Runkoverkossa käytetään GTP-protokollaa (GPRS Tunneling Protocol) pakettien reitittämiseen. Operaattori on vastuussa runkoverkon turvallisuudesta.

LIG (Lawful Interception Node) tehtävä on kerätä tieto ennalta määrätyistä käyttäjistä. Tämä tarkoittaa myös verkossa liikkuvaa dataa sekä laitteen sijaintia kullakin hetkellä. LIG rakentaminen osaksi GPRS-verkkoa on määrätty laissa.

BG (Border Gateway) tehtävä on yhdistää eri operaattoreiden verkkoja, jotta liikenne SGSN ja GGSN välillä sujuisi vaikka ne on eri operaattoreiden verkoissa. BG on osa GGSN:a.

CG (Charging Gateway) kerää tietoa verkossa liikkuvassa liikenteestä laskutusta varten. Varsinaisen keruun hoitaa SGSN ja GGSN, jotka toimittavat tiedon CG:lle.

GGSN (Gateway GPRS Support Node) on toinen GPRS-verkon tärkeistä komponenteista. Sen tehtävä on toimia siltana GPRS-verkon ja ulkoisen pakettiverkon(PDN) välillä. GGSN konvertoi SGSN:ltä saadut paketit ulkoiseen pakettiverkkoon sopiviksi ja päinvastoin. GGSN ja SGSN välillä vallitsee niin sanottu ”moni-moneen”-yhteys. Yksi GGSN palvelee monia SGSN:a ja päinvastoin. Tämä siksi, että GPRS:n laskutus perustuu siirrettyyn dataan. Jotta asiakasta osataan laskuttaa oikein, vaikka hän olisi toisen operaattorin alueella, dataliikenne reititetään SGSN ja BG avulla oman operaattorin GGSN:lle, joka kerää laskutustiedot CG:lle.

2.3. **Toiminta GPRS-verkkojen välillä**

Toiminta GPRS-verkkojen välillä on välttämätöntä, jotta pystyttäisiin tarjoamaan GPRS-palveluja maantieteellisestä sijainnista riippumatta. Päätelaitteen ollessa vieraan verkon alueella sen dataliikenne käyttää edelleen oman verkon GGSN:ä. Näin ollen data reititetään SGSN:ltä BG:lle, joka on yhteydessä oman operaattorin BG:hen ja sieltä edelleen yhden tai useamman SGSN:n kautta GGSN:n. Border Gateway:den välissä oleva verkko voi olla jokin erikseen kyseistä tarkoitusta palveleva verkkoratkaisu tai kuten useimmissa tapauksissa yhdyskäytävänä käytetään julkisia IP-verkkoja. Oletusarvoisesti operaattorit käyttävät IPSec:ä, autentikointia ja salausta taatakseen liikenteen perusturvallisuuden. Muista tietoturva ratkaisuista operaattorit voivat sopia keskenään.

2.4. **Toiminta IP-verkon kanssa**

Jotta voitaisiin toimia ulkoisten pakettiverkkojen(PDN) kanssa, päätelaitteen on saatava jokin ulkoisen verkon hyväksymä osoite, esimerkiksi IP-osoite IP-verkon tapauksessa. Tätä osoitetta kutsutaan PDP-osoitteeksi (Packet Data Protocol address). Osoite voidaan luoda istunnon yhteydessä, joten GPRS toimii sekä Ipv4, että Ipv6-protokollien kanssa. Haluttaessa voidaan käyttää myös staattisia PDP-osoitteita.

Istunnon luomisen yhteydessä luodaan PDP-konteksti, joka määrittelee PDP-tyypin, PDP-osoitteen, palvelun laadun ja välityspalvelimena toimivan GGSN:n osoitteen. Tämä konteksti talletetaan päätelaitteeseen, SGSN:ään ja GGSN:ään. Näin saadaan kuvaus IMSI ja PDP-osoitteen välille, jotta GGSN voi reitittää paketit ulkoisen verkon ja päätelaitteen välillä.

Ulkoisesta pakettiverkosta katsottuna GPRS-runkoverkko näyttää normaalilta IP aliverkolta ja GGSN IP-reitittimeltä. Jokaiselle verkon käyttäjälle on valittu GPRS-operaattorin osoiteavaruudesta IP-osoite. Jotta jokaiselle päätelaitteelle riittäisi osoite, sovelletaan dynaamista osoitteiden luontia, verkkoon voidaan asentaa DHCP-palvelin (Dynamic Host Control Protocol). IP-osoitteen linkittämisen IMSI:iin hoitaa GGSN PDP-konseptin perusteella.

GPRS-verkkoa voidaan pitää langattomana IP-verkon jatkeena. Näin ollen sitä koskevat samat tietoturva haasteet, kuin mitä tahansa IP-verkkoa. [Vesanen, 2003]

3. GPRS-verkon turvallisuusmalli

Ilmatien salaus ja autentikaatio toimivat GPRS-verkossa pieniä eroja lukuun ottamatta kuten GSM-verkossa. Ilmatien salaukseen on valittu uusi algoritmi, joka on GSM-verkon A5:n kaltainen jonosalausalgoritmi. Algoritmin yksityiskohdat eivät ole vuotaneet vielä julkisuuteen, kuten on käynyt A5:n kanssa. Algoritmin oletetaan kuitenkin olevan vahvuudeltaan A5:n luokkaa. Autentikaatio tapahtuu muuten samoin kuin GSM-verkossa, paitsi että piirikytkentäistä liikennettä ja pakettikytkentäistä liikennettä varten tehdään erilliset autentikaatiot. Tämä tietysti vain tapauksessa, että päätelaite on edellä mainittuja luokkia A tai B.

Ilmatien salaus ulottuu pidemmälle kuin GSM-verkossa, koska tukiasemat eivät osaa yhdistää useampaa aikaväliä käyttävää liikennettä samalta laitteelta tulevaksi. Näin ollen ilmatien salaus avataan vasta SGSN:llä. Tämä koskee kuitenkin vain pakettikytkentäistä liikennettä, koska piirikytkentäinen liikenne toimii perinteisen mallin mukaan.

Verkon uusi osuus verrattuna GSM-verkkoon on GPRS-runkoverkko. Runkoverkko käyttää sisäiseen paketin reititykseen GTP-protokollaa (GPRS Tunneling Protocol). Tämän avulla keskustelevat SGSN ja GGSN, muiden verkon toimijoiden ei tarvitse tuntea protokollaa. GTP-protokolla voi kapseloida erilaisia pakettiverkkojen protokollia. Oletusarvoisesti GTP -liikenne ei ole salattua, joten runkoverkkoon tunkeutuja voi seurata liikennettä. [Vesanen, 2003]

Samoin kuin GSM:ssa operaattori on vastuussa runkoverkkonsa turvallisuudesta. Erityisesti sen pitää estää ulkoapäin tulevat tunkeutumiset, näin ollen kaikki kytkennät ulkopuolisiin verkkoihin tulee varustaa palomuureilla.

4. Verkkoon kohdistuvat uhat

Lähes kaikki GSM-verkon turvallisuusuhat koskevat myös GPRS-verkkoa, koska GPRS on GSM:n päällä toimiva ratkaisu. Näin ollen en mainitse tässä yhteydessä GSM-verkkoon liittyviä turvaongelmia vaan totean, että samat uhat ovat edelleen olemassa koskien erityisesti salausalgoritmeja ja protokollia.

Yleisesti ajatellen GPRS:n runkoverkon osat ovat melko turvassa. Verkon sisällä käytettävä GTP-protokolla ei ole yhtä tunnettu, kuin IP-protokolla. Lisäksi verkon fyysinen turvallisuus on operaattoreiden vastuulla, koska runkoverkon liikenne ei ole oletusarvoisesti salattua. Tietoturva paranisi olennaisesti jos liikenne olisi salattua, mutta tämä ei liene mahdollista, jotta LIG:n toiminta voidaan taata.

Verkon varsinaiset uhat kohdistuvat osiin, jotka ovat yhteydessä ulkoisiin pakettiverkkoihin. Näitä ovat BG ja GGSN. Lisäksi GGSN:n kautta on mahdollista vaikuttaa ainakin päätelaitteiden ja CG:n toimintaan. Näistä BG on helposti suojattavissa, koska se on yhdyskäytävä operaattoreiden verkkojen välillä ja sitä tarkoitusta varten voidaan tehdä oma verkko. Jolloin vain fyysiset hyökkäykset muodostavat todellisen uhan. Suurimpana uhkana voidaankin pitää GGSN:ä heikkouksia. Lähestulkoon kaikki toteutetut hyökkäykset kohdistuvat nimenomaan GGSN:n, joita käsittelen seuraavassa luvussa.

5. Hyökkäykset GPRS-verkkoa vastaan internetistä

Koska GPRS käyttää dynaamista IP-osoitteiden hallintaa, joudutaan käyttämään NAT:a (Network Address Translation). NAT on metodi, jonka avulla laitteen IP-osoite muutetaan toisen IP-laitteen osoitteeksi pyrittäessä läpinäkyvään reititykseen. Tavallisesti NAT:a käytetään yksityisissä lähiverkoissa, jotta yksi osoite voidaan jakaa useamman laitteen kesken ja näin taata verkon kaikille laitteille pääsy ulkoiseen verkkoon.

Hyökättäessä Internetistä GPRS-verkkoa vastaan hyökkäys kohdistuu GGSN:ä suojelevaan NAT:vaan palomuuriin. Järjestelmän heikkous piilee siinä, että NAT-osoite muodostuu laitteen IP-osoitteesta sekä porttinumerosta, jonka avulla tunnistetaan sisäverkossa oleva päätelaite. Porttinumero on 16-bittinen luku, joten porttinumeroita ei voi olla kuin 65536. Aina, kun päätelaite on jonkin tukiaseman lähettyvillä, sille on varattuna PDP-konsepti, jossa sille on annettu myös IP-osoite. Yhteys on käytännössä aina auki, kun päätelaite on on-line tilassa ja verkon alueella. Näin ollen on oletettavaa, että GPRS-päätelaitteiden yleistyessä yhä useampi portti on

jatkuvasti varattuna, jolloin satunnainenkin hyökkäys todennäköisesti onnistuu.

Tyypillinen hyökkäys onkin roskahyökkäys. Verkossa olevalle päätelaitteelle lähetetään hyödyttömiä data paketteja, jolloin ylikuormituksen seurauksena päätelaitteen käyttö estyy sekä asiakkaalle kertyy laskua. Tällaista hyökkäystä vastaan on asiakkaan mahdotonta suojautua. Periaatteessa operaattori voisi havaita hyökkäyksen tarkkailemalla verkossa liikkuvaa dataa. Lisäksi IPSec suodattaisi ulkoapäin tulevat keinotekoiset paketit pois. IpSec:n käyttö runkoverkon sisällä voi kuitenkin olla mahdotonta LIG:n takia. Voidaan olettaa, että LIG:n olemassaolo perustuu juurikin siihen, että runkoverkossa liikkuva data on salaamatonta. [Vesanen, 2003]

6. Turvallisen GPRS-yhteyden luominen

Yleisin käyttötarkoitus GPRS:lle on luoda organisaation työntekijöille langaton yhteys Intranetiin. Voidaan sanoa, että GPRS-verkko ei tarjoa riittävää tietoturvatasoa, joten yhteys pitää saada turvalliseksi muilla tavoin. Yhteys voidaan salata päästä päähän tai turvata vain suojattu reitti julkisen verkon yli. Yhteys voidaan turvata ainakin kolmella tavalla [Rautpalo, 2000].

6.1. Päästä-päähän-VPN

Käyttäjän kannalta päästä-päähän-vpn on paras ratkaisu. Liikenne on salattu koko matkan päätelaitteelta organisaation VPN-palvelimelle asti, josta se reitietään edelleen intranetissä haluttuun palveluun.

Ongelmana on, että yleisimmät VPN-tekniikat eivät toimi verkossa, jossa on käytössä dynaaminen osoitteiden hallinta. Perinteiset IPSec ratkaisut eivät salli NAT:n käyttöä monestakaan syystä. Esimerkiksi tarkistettaessa paketin ehettyä sitä luullaan korruptoituneeksi, koska tarkistussumma ei täsmää ja näin ollen se hylätään. Ratkaisu tähän ongelmaan olisi staattisten IP-osoitteiden käyttö, mutta niitä on valitettavasti rajallisesti käytössä. Tulevaisuudessa ongelman tulee ratkaisemaan IPv6:n käyttöönotto, jolloin osoitteita on riittävästi.

On olemassa myös NAT:sen sallivia VPN-yhteyksiä, jotka käyttävät UDP:tä jolloin tarkistussumma ei lasketa. Haittapuolena on se, että yhteyden muodostaminen sekä datan siirtonopeus hidastuu sekä päätelaitteelta vaaditaan enemmän laskentatehoa, joten ratkaisua ei voi pitää hyvänä mikäli päätelaitteena toimii puhelin.

6.2. VPN-yhteys GGSN ja organisaation intranetin välillä

VPN-yhteys GGSN ja organisaation intranetin välillä toimii muuten samoin kuin edellä osoitettiin, mutta nyt yhteys ulottuu vain GGSN:lle asti, jolloin NAT:sta johtuva pakettien hylkäämisongelma poistuu. Uutena haittana on kuitenkin se, että GGSN pitää olla varustettuna laitteilla, jotka pystyvät hoitamaan yhteyden. Tämä tietää lisää kustannuksia sekä palveluntarjoajan pitää olla luotettava, koska liikenne GPRS-runkoverkossa on edelleen salaamatonta.

Yksi vaihtoehto on kytkeä organisaation intranet suoraan GGSN:ään, jolloin välissä ei ole julkista verkkoa. Ratkaisua voi suositella vain suurille organisaatioille, koska se on kallis toteuttaa.

Ongelmaksi jää edelleen käyttää jonkinlaista salausta, koska GPRS runkoverkossa liikenne on salaamatonta.

7. Yhteenveto

GPRS on ollut markkinoilla jo melko pitkään. Vaikkakin tarve nopealle ja globaalisti toimivalle langattomalle tiedonsiirrolle on suuri, niin GPRS ei tunnu saaneen tuulta alleen. Langalliset internetyhteydet ovat olleet jo pitkään kuluttajien saatavilla ja lähes poikkeuksetta niiden laskutus perustuu yhteysaikaan tai kiinteään periodimaksuun. GPRS:n siirrettyyn dataan perustuva laskutus ei näytä miellyttävän kuluttajia.

Verrattaessa GPRS:ä muihin langattomiin teknologioihin se tuntuu vanhentuneelta sekä turhan monimutkaiselta. Näyttäisi siltä, että palvelun tarjonta on mennyt palvelun laadun edelle. Lisäksi GPRS:n oma turvallisuusmalli ei näytä riittävältä, jotta palvelua voisi käyttää turvallisesti. Organisaation tarvitsee sijoittaa kohtuuttomasti varoja taatakseen riittävän turvallisuuden saavutettuun hyötyyn nähden.

Viiteluettelo

[Bjåen and Kaasin, 2001] Geir Stian Bjåen and Erling Kaasin, Security in GPRS. Agder University College, Information and Commynication Technology, 2001.
http://siving.hia.no/ikt01/ikt6400/ekaasin/Master%20Thesis%20Web.htm [15.12.2003]

[Candolin and Lundberg, 2001] Catharina Candolin and Janne Lundberg, Attacks on GPRS.
http://www.tml.hut.fi/~candolin/ studies/hakkeri/. [15.12.2003]

[Nichols and Lekkas 2002] R. Nicholas and P. Lekkas, *Wireless Security*: *Models, Threats and Solutions.* McGraw-Hill, 2002.

[Rautpalo 2000] Jussi Rautpalo, GPRS Security - Secure Remote Connections over GPRS. In: Helger Lipmaa and Heidi Pehu-Lehtonen (eds.) *Proceedings of the Seminar on Network Security*. Helsinki University of Technology, 2000. Available at http://www.hut.fi/~jrautpal/gprs/gprs_sec.html. [15.12.2003]

[Vesanen, 2003] Ari Vesanen, Langattomien verkkojen tietoturva, kurssimateriaali ja muistiinpanot GPRS:n osalta. Oulun yliopisto, kevät 2003.

NAT:n aiheuttamat ongelmat IPsec:lle ja niiden ratkaisu

Rauno Tamminen

Tiivistelmä

Tutkielmassa tarkastellaan NAT:n ja IPsec:n yhteiskäytön ongelmia sekä kuvataan IETF:n ehdottama ratkaisu niihin NAT-Traversal-protokollaa käyttäen.

Avainsanat ja -sanonnat: IPsec, NAT, NAT-T, NAT Traversal.
CR-luokat: D.4.4

1. Johdanto

Ehkä yleisin käyttötarkoitus IPsec-arkkitehtuurille on virtuaalisten yksityisverkkojen (VPN) rakentaminen. Yksi suosituimmista VPN:n käyttötavoista on tarjota etätyöntekijöille yhteys työpaikan sisäverkkoon. NAT (Network Address Translation) on tekniikka, jolla mahdollistetaan reititys kahden verkon välillä, joiden osoiteavaruudet eivät ole yhteensopivat. NAT on yleisesti käytössä yhdyskäytävissä, jotka tarjoavat Internet-yhteyksiä koteihin ja muihin etätyöskentelypaikkoihin, kuten esimerkiksi hotelleihin. Tästä on seurannut se, että IPsec:n ja NAT:n yhteensopivuusongelmat ovat merkittävä este IPsec:n käytön yleistymiseen IPv4:n turvaratkaisuna. [draft-ietf-ipsec-nat-reqts-06;2]

Tässä tutkielmassa esitellään IETF:n turvallisuusalan työryhmän laatima luonnos yhteensopivuusongelmien ratkaisemiseksi. Ongelmien ja ratkaisun kuvauksessa keskitytään Internet-protokollan versioon 4, koska versiolla 6 tarve NAT:n käyttöön on vähäinen tai olematon.

2. IPsec ja IKE

IPsec

IPsec on arkkitehtuuri, joka tarjoaa IP-tason tietoturvapalveluja. IPsec:iä voidaan käyttää suojaamaan siirtoyhteyksiä kahden päätelaitteen välillä, kahden tietoturvayhdyskäytävän välillä tai päätelaitteen ja tietoturvayhdyskäytävän välillä. Tietoturvayhdyskäytävällä tarkoitetaan siirtotiellä olevaa laitetta, joka toteuttaa IPsec-protokollia. [RFC 2401;5]

IPsec tarjoaa pääsynvalvonnan, yhteydettömän eheyden tarkistuksen, lähettäjän tunnistuksen, suojan toistohyökkäyksille, siirrettävän tiedon luottamuksellisuuden (salauksen) ja rajoitetusti tietovuon luottamuksellisuuden. Koska nämä tarjotaan IP-tasolla, on ne myös ylemmän tason protokollien kuten TCP:n, UDP:n tai ICMP:n käytettävissä. [RFC 2401;5]

IPsec määrittelee kaksi protokollaa tietoliikenteen turvaamiseen: AH (Authentication Header) ja ESP (Encapsulating Security Payload). AH tarjoaa eheyden tarkistuksen, lähettäjän tunnistuksen ja suojan toistohyökkäyksiltä. ESP tarjoaa luottamuksellisuuden, eheyden tarkistuksen, lähettäjän tunnistuksen ja suojan toistohyökkäyksiltä. Molempia voidaan käyttää pääsynvalvontaan käytettyjen salausavainten jakelun ja suojattujen tietovoiden hallinnan avulla. [RFC 2401;5-6]

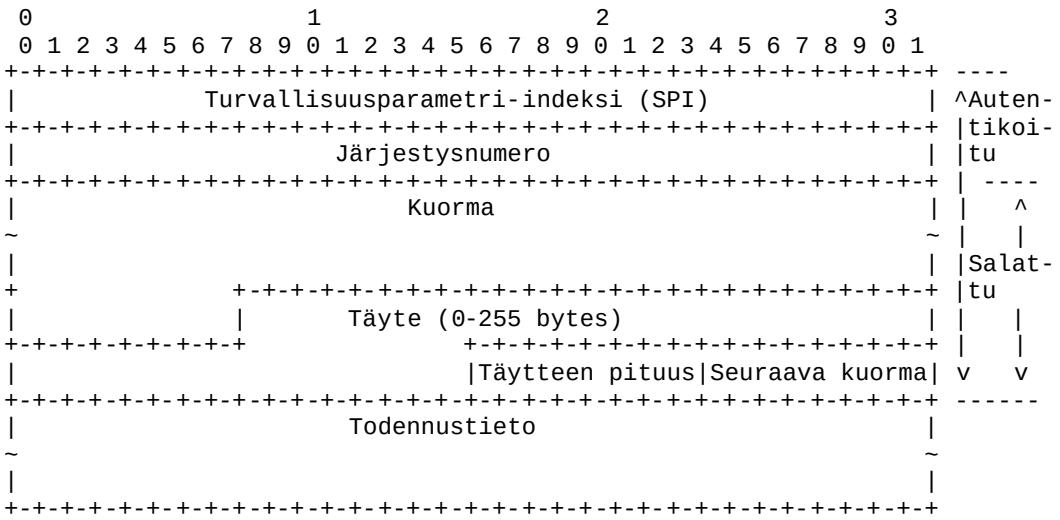
Molempia protokollia voidaan käyttää yksinään tai yhdessä, sekä IPv4:n että IPv6:n kanssa. Kumpikin protokolla tarjoaa kaksi menettelytapaa: kuljetustavan ja tunnelitavan. Kuljetustavassa suoja tarjotaan ylemmän tason protokollille, tunnelitavassa tunneloiduille IP-paketeille. [RFC 2401;6]

Tässä tutkielmassa keskitytään ESP-protokollaan, koska sillä voidaan toteuttaa kaikki mitä AH:llakin, eikä AH:lle ole juurikaan olemassa käytännön sovellutuksia. Siksi jatkossa esitellään vain ESP.

Turvallisuusassosiaatio (SA, Security Association) määrittelee tietoturvamenetelmän yhdelle yksisuuntaiselle yhteydelle, eli mitä liikennettä suojataan ja miten sitä suojataan. Yksi SA määrittelee joko AH:n tai ESP:n käytön, mutta ei molempia. Kaksisuuntaisen yhteyden turvaamiseen kahden päätelaitteen, kahden tietoturvayhdyskäytävän tai päätelaitteen ja tietoturvayhdyskäytävän välillä tarvitaan kaksi SA:ta, yksi molempiin suuntiin. [RFC 2401;7] Turvallisuusassosiaatio voidaan luoda manuaalisesti, mutta yleensä sen neuvottelemiseen käytetään myöhemmin esiteltävää IKE (Internet Key Exchange) -protokollaa.

Turvallisuusassosiaatio yksilöidään turvallisuusparametri-indeksillä (SPI, Security Parameter Index), kohde-IP-osoitteella ja turvaprotokollatunnisteella (AH tai ESP). [RFC 2401;7]

Kuvassa 1 on esitetty ESP-paketti. ESP-paketilla ei ole erillistä otsikkoa vaan kuorma on upotettu ESP-pakettiin. ESP-paketin tunniste sitä edeltäneessä IPv4-otsikkokentässä on 50. [RFC 2406;2]



Kuva 1. ESP-paketti

Kuvassa 1 turvallisuusparametri-indeksi (SPI) on 32-bittinen luku, joka yhdessä kohde-IP-osoitteen ja turvaprotokollan kanssa identifioi SA:n jota sovelletaan kyseiselle paketille. [RFC 2406;3]

Järjestysnumero on 32-bittinen luku, joka toimii laskurina kasvaen yhdellä joka paketissa. SA:n luonnin yhteydessä laskuri nollataan. Ensimmäinen paketti saa järjestysnumerokseen 1. Jos toistohyökkäysten tunnistus on käytössä, ei laskuri saa pyörähtää ympäri vaan SA on tuhottava kun laskuri tulee täyteen. [RFC 2406;3-4]

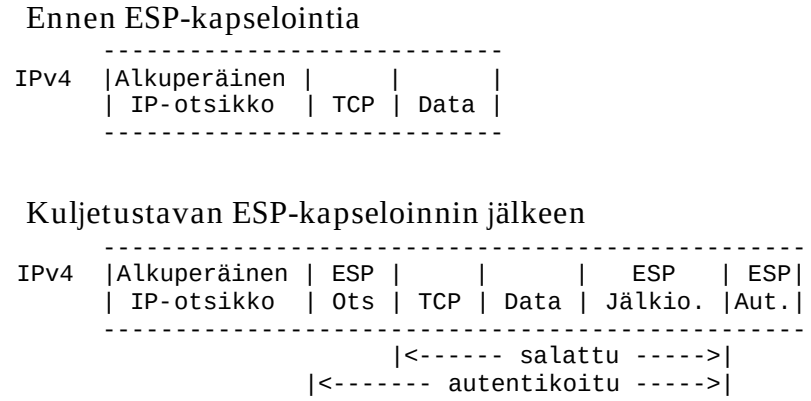
Kuorma-kenttä on pituudeltaan vaihteleva, ja sen sisältö on kuvattu *seuraava kuorma* -kentässä. Mikäli käytetty salausalgoritmi edellyttää alustusvektorin käyttöä, on alustusvektori sisällytetty kuormaan. [RFC 2406;4]

Täyte on 0-255 tavun pituinen kenttä. Täytettä käytetään, jos salausalgoritmi edellyttää, että salattava viesti on salausalgoritmin lohkokoon monikerta. Täytettä käytetään myös siihen, että *täytteen pituus* ja *seuraava kuorma* -kentät asemoituvat oikein, kuten kuvassa 1. Täytettä voi myös käyttää osittaiseen tietovuon salaamiseen, kätkemään varsinaisen kuorman pituus. [RFC 2406;5-6]

Täytteen pituus on tavun pituinen kenttä, joka ilmoittaa täytteen pituuden (0-255) ja seuraava kuorma on 1 tavun pituinen kenttä, joka ilmaisee kuorman sisältämän IP-protokollan tyypin. [RFC 2406;6]

Todennustieto on pituudeltaan vaihteleva kenttä, joka sisältää eheyden tarkistukseen käytetyn arvon, joka on laskettu ESP-paketista poislukien

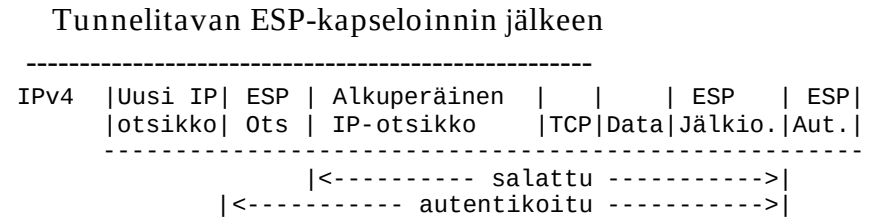
Todennustieto-kenttä. Kentän pituuden määrää käytetty eheydentarkistusalgoritmi. [RFC 2406;6]



Kuva 2. ESP-paketin sijainti kuljetustavalla

Kuvissa 2 ja 3 on esitelty ESP-paketin sijainti kuljetus- ja tunnelitavalla. Kuljetustavalla ESP-otsikko tulee IP-otsikon ja ylemmän tason protokollan, kuten TCP, UDP tai ICMP, otsikkokentän väliin. Kuormaa seuraava ESP:n jälkiosa sisältää täytteen, täytteen pituus ja seuraava kuorma -kentät. [RFC 2406;6-7]

Kuljetustapaa voidaan käyttää vain kahden päätelaitteen välillä, eli päästä päähän suojatuilla yhteyksillä.



Kuva 3. ESP-paketin sijainti tunnelitavalla

Tunnelitavalla koko suojattava IP-paketti kapseloidaan ESP-pakettiin ja ESP-otsikon eteen luodaan uusi IP-otsikko. Sisempi IP-otsikkokenttä sisältää alkuperäiset lähde- ja kohde-IP-osoitteet kun taas ulompi IP-otsikkokenttä sisältää IPsec:llä suojatun siirtotien (eli tietoturvayhdyskäytävien) lähde- ja kohde-IP-osoitteet. Tunnelitavalla voidaan toteuttaa yhteyksiä kahden päätelaitteen, kahden tietoturvayhdyskäytävän tai päätelaitteen ja tietoturvayhdyskäytävän välillä. [RFC 2406;7-8]

IPsec:n todellinen suoja määräytyy vasta käytettyjen salaus- ja eheydentarkistusalgoritmien mukaan. Tyypillisesti nykyään käytetään AES:ää ja SHA-1:stä.

IKE

Jotta kaksi tahoa voisivat turvallisesti kommunikoida käyttäen IPsec:n tarjoamia turvapalveluita, on näiden ensin neuvoteltava turvallisuus-assosiaatiot (SA:t) välilleen. IKE on menetelmä turvallisuusassosiaatioiden neuvotteluun turvallisesti. Se ei ole ainoa mahdollisuus turvallisuusassosiaatioiden muodostamiseen, mutta se on käytetyin. IKE on hybridi-protokolla, joka on yhdistelmä useampaa muuta protokollaa. Se perustuu ISAKMP-protokollaan, joka on määritelty RFC 2408:ssa. ISAKMP:n lisäksi IKE lainaa osia Oakley-avaintenhallintaprotokollasta. [RFC 2409; 5] IKE toimii UDP:n päällä ja se käyttää porttia 500. [RFC 2408; 20]

IKE määrittelee kolme toimintatapaa (mode) ja kaksi vaihetta (phase). Vaiheessa 1 (phase 1) osapuolet neuvottelevat turvallisen, autentikoidun viestintäkanavan, jota kutsutaan IKE SA:ksi. Vaihe 1 voidaan suorittaa käyttäen kahta eri toimintatapaa, Main Modea tai Aggressive Modea. Vaiheessa 2 (phase 2) käytetään tätä viestintäkanavaa IPsec SA:iden neuvotteluun. Vaihe 2 voidaan käydä vain yhdellä toimintatavalla, joka on nimeltään Quick Mode. [RFC 2409; 5]

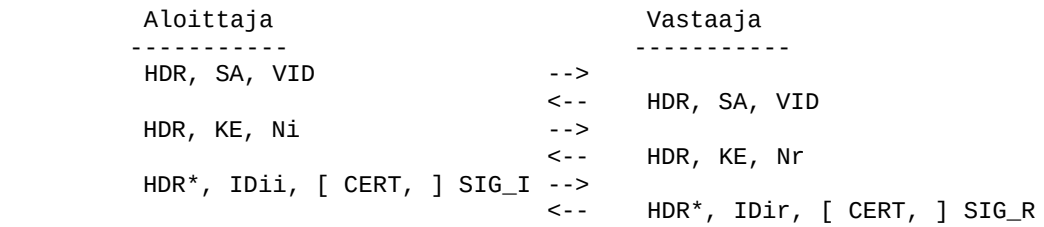
Toisin kuin IPsec SA, IKE SA on kaksisuuntainen, ja kun se on muodostettu, sitä voidaan käyttää useiden Quick Mode -neuvottelujen käymiseen sekä tiedonantopakettien (Informational) lähettämiseen. Tiedonantopaketit ovat yksisuuntaisia, eikä niihin odoteta vastausta. [RFC 2409; 5]

Main Mode -toimintatavassa siirretään yhteensä kuusi pakettia. Kaksi ensimmäistä sopivat parametrit, joilla neuvottelu käydään, kuten salausalgoritmin, tiivistealgoritmin ja tunnistustavan. Seuraavat kaksi vaihtavat Diffie-Hellman-vaihdon julkiset avaimet ja satunnaisluvut (nonces), joilla muodostetaan salausavain. Kaksi viimeistä autentikoivat Diffie-Hellman-vaihdon, eli salattu yhteys on muodostettu oikean tahon kanssa, eikä välissä ole ketään kolmatta osapuolta. [RFC 2409; 7]

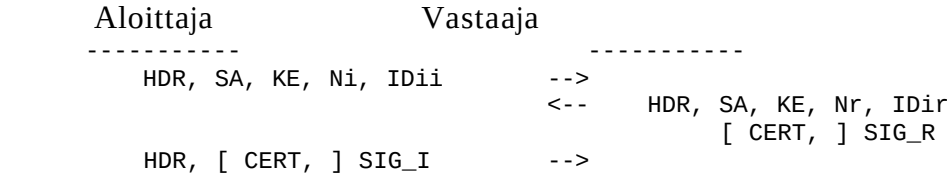
Aggressive Mode -toimintatavassa käytetään puolta vähemmän paketteja. Kahdessa ensimmäisessä paketissa vaihdetaan neuvottelun parametrit, julkiset avaimet, satunnaisluvut ja identiteetit. Lisäksi toinen paketti autentikoi vastaajan. Kolmas paketti autentikoi aloittajan. Tämä toimintatapa ei suojaa neuvottelevien osapuolten identiteettejä neuvottelua kuuntelevilta kolmansilta osapuolilta. [RFC 2409; 7-8]

Sekä Main Mode että Aggressive Mode -toimintatavoissa mahdollistetaan neljä erilaista tunnistustapaa: digitaalinen allekirjoitus, kaksi julkisen avaimen

salakirjoituksella toteutettavaa tapaa ja jaetulla salaisuudella tunnistaminen. Esimerkkinä (ks. kuvat 4 ja 5) käsitellään vain IKE-neuvottelu digitaalisella allekirjoituksella tunnistamisen kanssa. Muut eivät eroa esitellystä NAT-ongelman suhteen. [RFC 2409; 9-11]



Kuva 4. Main Mode -toimintatapa



Kuva 5. Aggressive Mode -toimintatapa

Kuvien 4 ja 5 esimerkeissä on käytetty seuraavia lyhenteitä:

HDR on ISAKMP-otsikko ja HDR* ilmaisee että kuorman olevan salattu.

SA on SA-neuvottelukuorma, joka sisältää yhden tai useampia ehdotuksia. Aloittaja voi sisällyttää useamman ehdotuksen, vastaajan pitää vastata yhdellä.

VID on valmistajatieto (Vendor ID).

KE on avaintenvaihtokuorma joita käyttämällä toteutetaan Diffie-Hellman-vaihto.

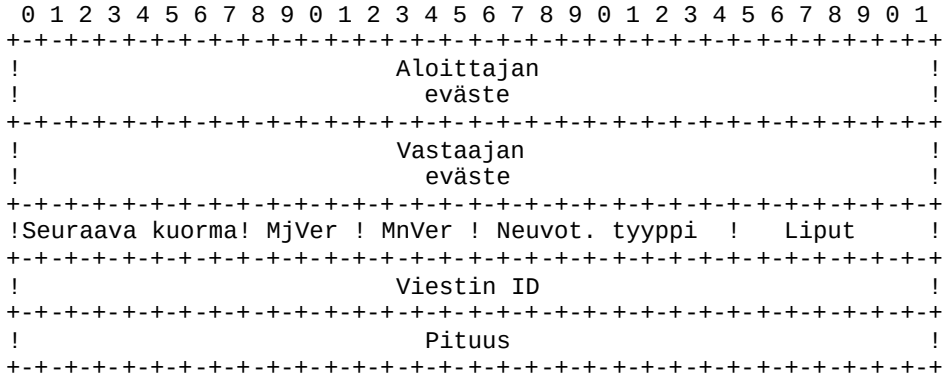
Ni ja Nr ovat satunnaislukuja (nonce).

IDii ja IDir ovat IKE-identifiointikuormia, joita käytetään vastapuolien tunnistamiseen ja oikean IPsec-säännön valitsemiseen.

CERT on sertifikaattikuorma. Sertifikaattien lähetys voi olla tarpeen allekirjoituksen tarkastamista varten.

SIG_T ja SIG_R ovat allekirjoituskuormia, jotka ovat SA-kuormissa neuvotellulla allekirjoitusalgoritmilla allekirjoitettuja tiivisteitä IKE-neuvottelussa siihen mennessä vaihdetusta tiedosta. Allekirjoitus samalla todentaa neuvottelun eheyden.

Kuvassa 6 on ISAKMP-otsikon rakenne.



Kuva 6. ISAKMP-otsikko. [RFC 2408;21-25]

Kuvassa 6 on seuraavat kentät:

Aloittajan eväste (8 tavua): Eväste jolla yksilöidään aloittaja.

Vastaajan eväste (8 tavua): Eväste jolla yksilöidään vastaaja.

Seuraava kuorma (1 tavu): Ilmaisee seuraavan kuorman tyypin.

Major versio (4 bittiä): Ilmaisee ISAKMP-protokollan major version.

Minor version (4 bittiä): Ilmaisee ISAKMP-protokollan minor version.

Neuvottelun tyyppi (1 tavu): Esimerkiksi Main Mode tai Aggressive Mode-toimintatapa.

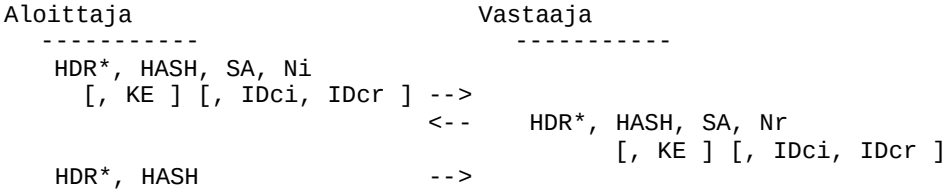
Liput (1 tavu).

Viestin ID (4 tavua). Käytetään erottamaan vaihe 2:n neuvottelut toisistaan.

Pituus (4 tavua): Koko paketin pituus (otsikko + kuormat).

Kun vaiheessa 1 on ensin muodostettu turvallinen IKE-yhteys osapuolten välille, vaiheessa 2 neuvotellaan IPsec-yhteyden vaatimat parametrit, kuten salausalgoritmi, eheydentarkistus-algoritmi ja avaimet, eli muodostetaan IPsec SA:t. Neuvottelu on salattu IKE SA:n määräämällä tavalla. [RFC 2409;16]

Kuvassa 7 on vaiheen 2 neuvottelu.



Kuva 7. Quick Mode

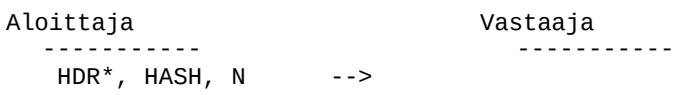
Kuvassa 7 on käytetty seuraavia lyhenteitä:

HASH on avaimellinen tiiviste neuvottelussa vaihdetusta tiedosta.

KE on avaintenvaihtokuorma, jos vaiheessa 2 halutaan tehdä uusi Diffie-Hellman-vaihto.

IDci ja IDcr ovat IKE-identifiointikuormia.

Lisäksi IKE määrittelee tiedonannot, joilla voidaan esimerkiksi keskeyttää IKE-neuvottelu (abort notification) tai ilmoittaa vastapuolelle tuhotuista SA:ista (delete notification). Kumpi tahansa osapuoli voi lähettää tiedonantoja. Tämä on esitetty kuvassa 8.



Kuva 8. Tiedonanto

Kuvassa 8 N on tiedonantokuorma, esimerkiksi ilmoitus SA:n tuhoamisesta.

3. NAT

NAT on metodi, jolla IP osoitteet muunnetaan osoiteavaruudesta toiseen ja tarjotaan läpinäkyvä reititys näiden osoiteavaruuksien välillä. Yleisesti NAT-laittetta käytetään yhdistämään eristetty, yksityistä osoiteavaruutta käyttävä verkko ulkoiseen verkkoon (kuten Internet) käyttäen yhtä tai useampaa ulkoisessa verkossa rekisteröityä IP-osoitetta. [RFC 2663;1]

Tarve NAT:n käytölle tulee, kun verkon sisäisiä IP-osoitteita ei voida käyttää verkon ulkopuolella, joko siksi, että ne ovat epäkelpoja (ei-reititettäviä) tai siksi, että ne halutaan salata. NAT mahdollistaa päätelaitteiden yksityisessä verkossa läpinäkyvästi kommunikoida ulkoisessa verkossa olevien kohteiden kanssa ja toisinpäin. Tämä toteutetaan muuttamalla IP-osoitteita ja säilyttämällä näiden muutosten tila niin, että paluupaketeille osataan tehdä vastakkainen muutos. [RFC 2663;2]

NAT toimii istuntotasolla (session) eli samaan istuntoon kuuluvilla paketeille tehdään sama muunnos. NAT ei voi olla täysin varma, mitkä paketit kuuluvat mihinkin istuntoon, joten yleensä päätellään että ensimmäinen paketti, jolle ei vielä ole NAT-muunnosta, aloittaa uuden istunnon. Istunnon päättymisen toteamiseksi ehkä toimivin tapa on aikakatkaisu silloin, kun yhteys on ollut käyttämättä määriteltävissä olevan ajan. [RFC 2663;3-5]

Ulkoisessa/julkisessa verkossa (yleensä Internet) osoiteavaruus muodostuu yksilöllisistä osoitteista, jotka on myöntänyt IANA (Internet Assigned Numbers Authority) tai vastaava osoitteita hallinnoiva taho kuten APNIC,

RIPE tai ARIN. [RFC 2663;5] Sisäverkko/yksityisverkko on osoiteavaruus, jonka osoitteet ovat riippumattomia ulkoisen verkon osoitteista ja osoitehallinasta. Reititystä sisäverkon ja ulkoverkon välillä hoitaa NAT-reititin. [RFC 2663;5]

Mikäli sisäverkosta on tarkoitus liikennöidä myös ulkoverkkoon, on sisäverkossa syytä käyttää osoitteita, jotka IANA on varannut tähän tarkoitukseen, jotta sisäverkon osoitteet eivät törmää ulkoverkon kanssa.

IANA:n varaamat osoiteavaruudet yksityisille verkoille ovat seuraavat [RFC 1918]:

10.0.0.0 - 10.255.255.255 (10/8)
172.16.0.0 - 172.31.255.255 (172.16/12)
192.168.0.0 - 192.168.255.255 (192.168/16).

Kaikki protokollat eivät tue NAT-muunnosta. Jotkut liittävät IP-osoitteita ja TCP/UDP-porttinumeroita hyötykuormaansa. Sovellustason yhdyskäytävät (Application Level Gateway, ALG) ovat sovelluskohtaisia muunnosagentteja, jotka mahdollistavat osoitemuunnokset tällaisille sovelluksille muuttamalla IP-osoitteita ja TCP/UDP-porttinumeroita myös hyötykuormassa ja varaamalla valmiiksi uusia NAT-muunnoksia samaan istuntoon liittyville uusille yhteyksille tai porteille. [RFC 2663 ;6]

Kaikille ICMP-viesteille lukuun ottamatta Re-direct-viestiä pitää myös tehdä NAT-muunnos, jotta yhteydet (istunnot) toimisivat oikein. NAT:n pitää muokata myös ICMP-viestin hyötykuormaan upotettua IP-pakettia ja ICMP-otsikon tarkistussumma pitää laskea tällöin uudelleen. Myös normaali IP-otsikko ICMP-viestissä pitää muokata. [RFC 2663;10]

Perinteinen NAT sallii päätelaitteiden muodostaa yksityisestä verkosta yhteyksiä ulkoiseen verkkoon, mutta ei toisinpäin. Tällöin vaatimuksena on, että julkisen verkon osoitteet ovat reititettäviä yksityisessäkin verkossa eikä niissä ole päällekkäisyyksiä. [RFC 2663;11]

Perus-NAT:ssa joukko ulkoisen verkon osoitteita on varattu NAT-muunnoksessa käytettäviksi. Muunnos tehdään vain IP-osoitteille. Kyseessä on siis 1:1 -muunnos sisä- ja ulkoverkon osoitteiden välillä. [RFC 2663;11]

NAPT:ssa (Network Address and Port Translation) muunnos tehdään myös kuljetuskerroksen tunnisteille, kuten TCP ja UDP-porttinumeroille ja ICMP-viestien tunnisteille. Näin useampi yksityisen verkon päätelaite voidaan kanavoida käyttämään yhtä julkisen verkon osoitetta. NAPT on n:1 -muunnos sisä- ja ulkoverkon osoitteiden välillä. NAPT:ia voidaan käyttää myös yhdessä perus-NAT:n kanssa. [RFC 2663;11] Jatkossa lyhenne NAT yleistetään tarkoittamaan myös NAPT:ia.

Molemmat NAT:t mahdollistavat myös rajoitetusti sääntöjen luonnit yhteyksille ulkoverkosta sisäverkkoon, eli esimerkiksi NAT-laitteen TCP-porttiin 80 avattu yhteys voidaan edelleenohjata www-palvelimelle sisäverkossa.

4. IPsec:n ja NAT:n yhteensopivuusongelmat

IPsec:n ja NAT:n tunnetut yhteensopivuusongelmat voidaan jakaa kolmeen kategoriaan [draft-ietf-ipsec-nat-reqts-06;3-8]:

1. Varsinaiset NAT-toiminnallisuuteen liittyvät ongelmat. Nämä ongelmat esiintyvät kaikissa NAT-laitteissa.
   1.1. AH:n ja NAT:n yhteensopimattomuus. AH varmentaa myös IP-otsikon lähde- ja kohdeosoitteiden eheyden (muuttumattomuuden). ESP ei varmenna lähde- ja kohdeosoitteiden eheyttä, joten tämä ongelma ei esiinny ESP:n kanssa. [draft-ietf-ipsec-nat-reqts-06;4]
   1.2. NAT:n rikkomat tarkistussummat. TCP:n ja UDP:n tarkistussummiin lasketaan mukaan myös IP-otsikon lähde- ja kohdeosoitteet. Vastaanottajan tarkastaessa nämä tarkistussummat eivät enää pidä paikkaansa, kun NAT on muuttanut lähde- tai kohdeosoitteita. Tämän seurauksena ESP toimii vain, jos kuorma on jotain muuta kuin TCP tai UDP (kuten esimerkiksi IPIP tunnelointitavan ESP:n yhteydessä) tai tarkistussummia ei lasketa. NAT-laite ei voi muuttaa TCP:n tai UDP:n tarkistussummia, koska ne ovat kuljetustavan ESP-kapseloinnin suojaamia. [draft-ietf-ipsec-nat-reqts-06;4-5]
   1.3. Yhteensopimattomuus IKE-osoitetunnisteiden ja NAT:n välillä. IP-osoitteita voidaan käyttää tunnisteina IKE:n vaiheissa 1 tai 2. NAT:n muokkaamina nämä osoitteet eivät enää ehkä täsmää haluttuun sääntöön. [draft-ietf-ipsec-nat-reqts-06;5]
   1.4. Kiinteiden IKE-porttien ja NAPT:n välinen ongelma. Kun monta tahoa aloittaa IKE- neuvottelun saman päätepisteen kanssa saman NAPT:n takaa, kaikki eivät saa NAPT:lta oikeaa lähdeporttia. Tällöin vastaajan pitää huomioida, että lähdeportti voi olla muukin kuin 500. Myös uudelleen avaintaminen (Re-keying) pitää tehdä käyttäen tätä porttia. [draft-ietf-ipsec-nat-reqts-06;5]
   1.5. Saman NAT:n takaa tulevilla voi olla lomittaisia (overlapping) IPsec-sääntöjä ja sama kohdeosoite. Tällöin paketeille voidaan käyttää väärää IPsec SA:ta. [draft-ietf-ipsec-nat-reqts-06;5]
   1.6. NAT ei näe ESP:n sisälle, joten se ei voi tehdä muunnosta lähde/kohdeporttien perusteella. [draft-ietf-ipsec-nat-reqts-06;5-6]

1.7. Protokolliin upotetut IP-osoitteet. Protokollat kuten FTP, IRC, SNMP ja LDAP käyttävät IP-osoitteita omaan toimintaansa. IPsec estää sovellustason yhdyskäytävää NAT:n yhteydessä muuttamasta näitä osoitteita. [draft-ietf-ipsec-nat-reqts-06;6]

2. NAT:n toteutukseen liittyvät heikkoudet. Vaikka nämä ongelmat eivät esiinny kaikkien NAT laitteiden kanssa, ne saattavat olla kuitenkin sen verran yleisiä, että ne on syytä huomioida NAT:n ja IPsec:n yhteensopivuusongelmia ratkottaessa.
   2.1. Tuen puute muilta kuin UDP/TCP-protokollilta. Tällaisten NAT:ien läpi ei pääse ESP tai AH.
   2.2. NAT-muunnostaulujen liian lyhyt elinikä [draft-ietf-ipsec-nat-reqts-06 ;7]
   2.3. NAT:ien kyvyttömyys fragmenttien kokoamiseen (defragmentointiin) ja uudelleen fragmentointiin (refragmentointiin) tarvittaessa. [draft-ietf-ipsec-nat-reqts-06 ;7]
3. Apusovellusten aiheuttamat ongelmat. Näitä esiintyy NAT-laitteissa, jotka yrittävät tarjota yhteensopivuutta IPsec:n kanssa, yleensä siinä ainakin osittain epäonnistuen. Nämä ominaisuudet aiheuttavat lisäongelmia NAT:n ja IPsec:n yhteensopivuusongelmien ratkaisemiseen.
   3.1. Jotkut NAT:t eivät tee muunnosta portille 500 vaan yrittävät soveltaa muuta kanavointia, kuten IKE:n kanavointi IKE-evästeiden perusteella. Uudelleenavainnuksessa IKE-evästeet muuttuvat ja on mahdotonta yhdistää niitä edellisiin IKE SA:ihin. [draft-ietf-ipsec-nat-reqts-06;8]
   3.2. IPsec-tietoiset NAT:t yrittävät kanavoida ESP-paketteja IPsec SPI:en avulla, mutta koska eri suuntiin menevät SPI:t valitaan toisistaan riippumatta, ei ole mahdollista varmasti tietää, mitkä SPI:t liittyvät toisiinsa. Jos useampi kohde saman NAT:n takaa neuvottelee samaan päätepisteeseen, voivat paluupaketit päätyä väärälle kohteelle. [draft-ietf-ipsec-nat-reqts-06;5-6]

5. NAT-Traversal:n vaikutus IKE-neuvotteluun

IKE-neuvottelun vaiheessa 1 selvitetään tuki NAT-Traversal:lle ja havaitaan, tehdäänkö siirtotien varrella NAT-muunnosta. [draft-ietf-ipsec-nat-t-ike-07;2]

NAT voi vaihtaa IKE:n UDP-lähdeporttia ja vastaanottajien pitää kyetä käsittelemään IKE-paketteja, joiden lähdeportti ei ole 500. Vastaanottajan pitää käyttää pakettien lähdeporttia kohdeporttina vastauksille. Mikäli alkuperäinen vastaaja suorittaa uudelleenavainnuksen tai lähettää huomautuksia, ne pitää lähettää tähän kohdeporttiin. [draft-ietf-ipsec-nat-t-ike-07;3]

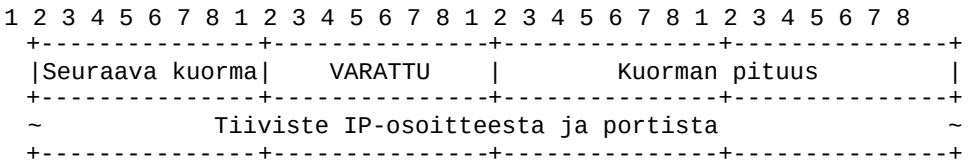
Osapuolet ilmaisevat tukensa NAT-Traversal:lle lähettämällä luonnoksen määrittelemä valmistaja ID (MD5 tiiviste "RFC XXXX") vaiheen 1 kahdessa ensimmäisessä viestissä. Molempien osapuolten pitää lähettää ja vastaanottaa tämä valmistajatunnistekuorma (Vendor Id payload), jotta NAT-Traversal-luotaus voi jatkua. [draft-ietf-ipsec-nat-t-ike-07;3]

Täällä hetkellä draft-ietf-ipsec-nat-t-ike on vasta Internet Draft -luonnos, joten IETF ei ole vielä määritellyt RFC-numeroa ja sitä tukevan toteutuksen tekeminen on hankalaa. Aikaisemmat luonnokset määrittelivät referenssi-toteutuksia varten valmistaja id:n, kuten draft-internet-ike-nat-t-ike-04.

IKE:n vaiheen 1 (Main Mode -toimintatavassa) kolmannessa ja neljännessä viestissä (tai Aggressive Mode -toimintatavassa toinen ja kolmas viesti) lähetetään NAT-D (NAT-Detection) -kuormat, joilla havaitaan sekä NAT:n olemassaolo siirtotiellä että NAT:n sijainti, eli kumman osapuolen osoitteelle NAT-muunnos tehdään. Tämä on tärkeää siksi, että NAT-muunnoksen kohteena olevan osapuolen pitää lähettää NAT-elossapitoviestejä. [draft-ietf-ipsec-nat-t-ike-07;3-4]

Jotta NAT:n olemassaolo voidaan havaita, pitää tarkkailla, muuttuvatko pakettien lähde- tai kohde-IP-osoitteet tai -portit siirtotien varrella. Molemmat osapuolet lähettävät toisilleen tiivisteet sekä lähde- että kohde-IP-osoitteista ja -porteista. Vastaanottajat laskevat tiivisteen myös molemmista, ja jos tiivisteet ovat samat, ei siirtotiellä ole NAT:ia. Jos jompi kumpi tiivisteistä eroaa, tehdään välillä NAT-muunnosta ja tarve NAT-Traversal:lle on olemassa. Sen mukaan poikkeaako lähde- vai kohdetiiviste, voidaan päätellä, kumman osapuolen osoitteelle NAT-muunnos tehdään, tai harvinaisessa tapauksessa, tehdäänkö molempien. Jos ensimmäinen vastaanotettu NAT-D -kuorma ei vastaa minkään paikallisen verkkoliitynnän tiivistettä, on osapuoli NAT-muunnoksen kohteena ja sen on syytä lähettää NAT-elossapitoviestejä. [draft-ietf-ipsec-nat-t-ike-07;3-4]

Jos osapuolella on useampi IP-osoite eikä hän tiedä, minkä kautta paketit tullaan reitittämään, se voi lähettää useamman tiivisteen omina NAT-D -kuorminaan. Tässä tapauksessa NAT on havaittu vain, jos yksikään tiivisteistä ei vastaa vastaanottajan laskemaa tiivistettä. [draft-ietf-ipsec-nat-t-ike-07;3-4] NAT-D –kuorma on esitetty kuvassa 9.



Kuva 9. NAT-D-kuorma

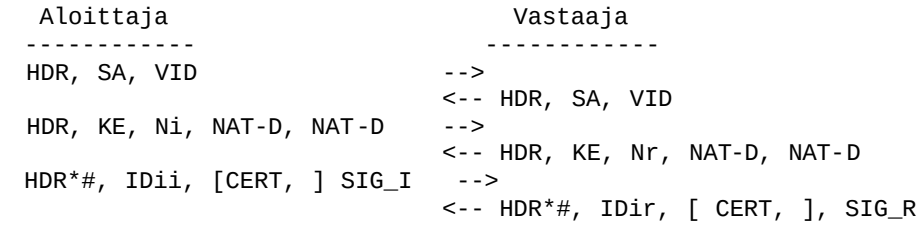
Kuorma-tunniste (payload id) NAT discovery kuormalle on 15. [draft-ietf-ipsec-nat-t-ike-07;4]

Tiiviste lasketaan kaavalla

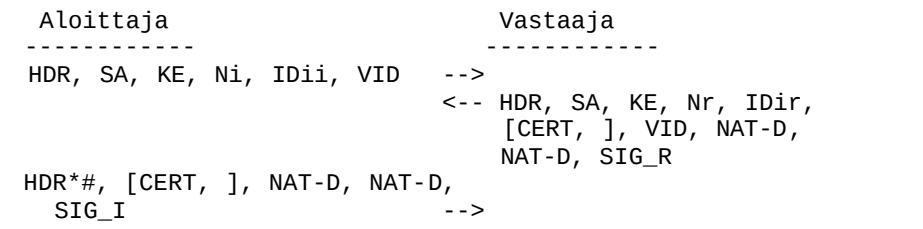
HASH = HASH(CKY-I | CKY-R | IP | Port)

käyttäen neuvoteltua tiivistealgoritmia. [draft-ietf-ipsec-nat-t-ike-07 ;4]

Ensimmäinen NAT-D-kuorma sisältää vastapuolen IP-osoitteen ja portin ja seuraavat paikalliset IP-osoitteet ja portit. CKY-I ja CKY-R ovat aloittajan ja vastaajan IKE-evästeet ja ne on lisätty, jotta “precomputation attack” IP-osoitetta tai porttia vastaan olisi mahdoton. [draft-ietf-ipsec-nat-t-ike-07;4] Main Mode ja Aggressive Mode –toimintatavat on esitetty kuvissa 10 ja 11.



Kuva 10. Main Mode -toimintatapa, kun tunnistaminen suoritetaan digitaalisilla allekirjoituksilla.



Kuva 11. Aggressive Mode -toimintatapa, kun tunnistaminen suoritetaan digitaalisilla allekirjoituksilla.

Kuvissa 10 ja 11 merkintä '#' tarkoittaa että kyseiset viestit on lähetetty siirtyen uusiin portteihin kun NAT on havaittu.

IPsec-tietoiset NAT:t voivat aiheuttaa ongelmia kuten luvussa 4 kuvattiin. Jotkut NAT:t eivät vaihda IKE-lähdeporttia 500, vaikka NAT-muunnos tehtäisiin useammalle päätelaitteelle. Ne saattavat myös käyttää IKE-evästeitä kanavoimaan liikennettä lähdeporttien sijasta. IKE:n on vaikea havaita NAT:ien ominaisuuksia, ja siksi onkin parasta siirtää neuvottelu pois porteista 500 niin pian kuin mahdollista. [draft-ietf-ipsec-nat-t-ike-07;5]

Main Mode -toimintatavassa aloittajan pitää siirtyä käyttämään porttia 4500 alkaen vaiheen 1 viestistä 5. Sen jälkeen kaikki paketit, mukaan lukien huomautukset, tulee lähettää portista 4500. Lisäksi IKE otsikkoa pitää edeltää ei-ESP-merkki, joka määritellään myöhemmin. [draft-ietf-ipsec-nat-t-ike-07;5]

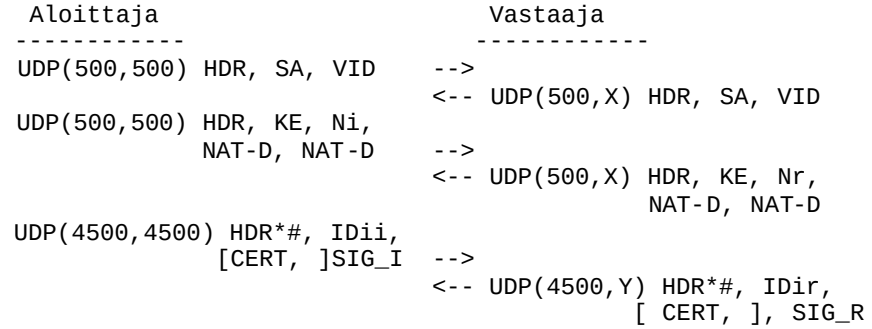
Kun tunnistamiseen käytetään allekirjoituksia, IKE vaiheen 1 viesti 5 näyttää seuraavalta:

IP UDP(4500,4500) <ei-ESP-merkki> HDR*, IDii, [CERT,] SIG_I

IKE-implementaatio, joka tukee tätä luonnosta, ei saa käyttää ESP:n SPI-kenttää arvolla nolla ESP-paketeille. Tällä varmistetaan, että IKE-paketit ja UDP-koteloidut ESP-paketit erotetaan toisistaan. [draft-ietf-ipsec-udp-encaps-06]

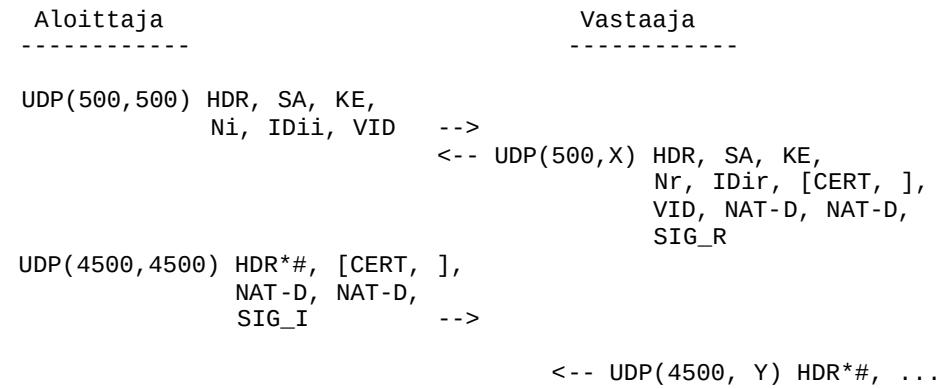
Kun vastaaja saa tämän viestin, hänen on myös siirryttävä käyttämään seuraaville viesteille paikallista porttia 4500 ja lähettävä ne kohdeporttiin josta viides paketti tuli. Jos vastaajan pitää neuvotella uudet avaimet IKE SA:lle, hänen täyttää käyttää näitä samoja portteja. Mikäli porttiin 500 tulee vielä kyseiseen neuvotteluun liittyviä paketteja porttiin 4500 siirtymisen jälkeen, ne ovat vanhentuneita ja ne tulee hylätä. NAT-Traversal:a tukevien IPsec-toteutusten pitää tukea myös IKE neuvottelun aloittamista porteissa 4500, jolloin tarvetta siirtyä toisiin portteihin ei enää ole. [draft-ietf-ipsec-nat-t-ike-07;6]

Kuvassa 12 on esitetty vaiheen 1 neuvottelu NAT-Traversal:lla käyttäen Main Mode -toimintatapaa ja tunnistamista allekirjoituksilla:



Kuva 12. Main Mode -toimintatapa NAT-Traversal-tuella.

Aggressive Mode -toimintatavassa (kuva 13) toiminta on melko samanlaista. Aloittaja siirtyy porttiin 4500 kolmannessa viestissä.



Kuva 13. Aggressive Mode -toimintatapa NAT-Traversal-tuella.

Myös elossapitoviestien ajastettu lähetys käynnistyy sen jälkeen, kun ollaan siirrytty uusiin portteihin, joten elossapitoviestejä ei lähetetä porttiin 500. [draft-ietf-ipsec-nat-t-ike-07 ;7]

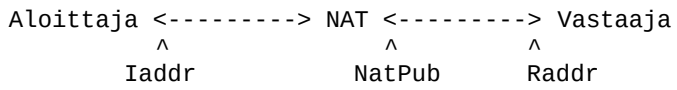
Vaiheen 1 jälkeen molemmat osapuolet tietävät, tehdäänkö siirtotiellä NAT-muunnosta. Päätös NAT-Traversal:n käytöstä tehdään vaiheessa 2. Quick Modessa molemmat osapuolet voivat myös lähettää alkuperäiset IP-osoitteensa NAT-OA (Original Address) -kuormina. Kuljetustapaa käytettäessä vastapuoli voi käyttää alkuperäistä IP-osoitetta TCP-tarkistussumman korjaamiseen NAT:n jäljiltä. [draft-ietf-ipsec-nat-t-ike-07;7]

NAT-Traversal:n neuvottelua varten tarvitaan kaksi uutta kapselointitapaa, UDP-Encapsulated-Tunnel ja UDP-Encapsulated-Transport. Jos siirtotiellä tehdään NAT:a, ei normaali kuljetus- tai tunnelitapa välttämättä toimi, ja siksi on syytä käyttää UDP-kapselointitapaa. Jos siirtotiellä ei ole NAT:a, UDP-kapselointi olisi turhaa. [draft-ietf-ipsec-nat-t-ike-07;7]

Jotta TCP:n tarkistussumma voidaan korjata vähentämällä vanha ja lisäämällä uusi osoite, pitää molempien osapuolien tietää alkuperäiset IP-osoitteet, joita toinen osapuoli on käyttänyt tarkistussumman laskemiseen. Aloittajan IP-osoite on aloittajan alkuperäinen IP-osoite, ja vastaajan osoite on aloittajalle näkyvä vastaajan osoite. Vastaavasti vastaajalla alkuperäinen aloittajan IP-osoite on vastaajalle näkyvä IP-osoite ja vastaajan osoite on vastaajan alkuperäinen IP-osoite. [draft-ietf-ipsec-nat-t-ike-07;8]

Alkuperäiset osoitteet lähetetään NAT-OA-kuormina. [draft-ietf-ipsec-nat-t-ike-07;8] Vaiheen 1 NAT-D-kuormia ei voi käyttää, koska ne olivat vain tiivisteitä alkuperäisistä osoitteista eikä niitä siten voi muuttaa takaisin alkuperäisiksi osoitteiksi.

Aloittajan NAT-OA-kuorma tulee ensin, sitten vastaajan NAT-OA- kuorma. Kuvassa 14 aloittajan osoitteelle tehdään NAT-muunnos. (NatPub on NAT:n julkinen osoite).



Kuva 14. NAT-muunnos aloittajan osoitteelle.

Kuvan 14 tilanteessa aloittajalle pätee

NAT-OAi = Iaddr

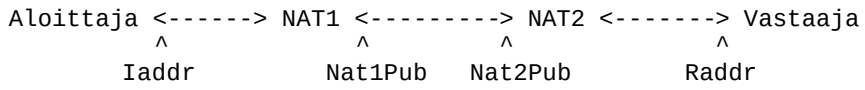
NAT-OAr = Raddr

ja vastaavasti vastaajalle

NAT-OAi = NATPub

NAT-OAr = Raddr.

Kuvassa 15 on esitetty, miten NAT2 edelleenlähettää kaiken liikenteen vastaajalle.



Kuva 15. NAT2 edelleenlähettää kaiken liikenteen vastaajalle.

Kuvassa 15 myös vastaajan osoitteelle tehdään NAT-muunnos. Kuvan 15 tilanteessa aloittajalle pätee

NAT-OAi = Iaddr

NAT-OAr = Nat2Pub

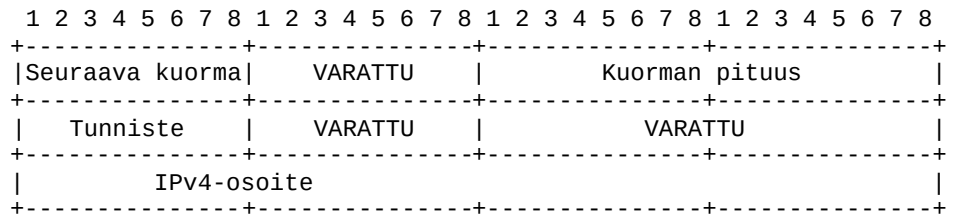
ja vastaajalle

NAT-OAi = Nat1Pub

NAT-OAr = Raddr.

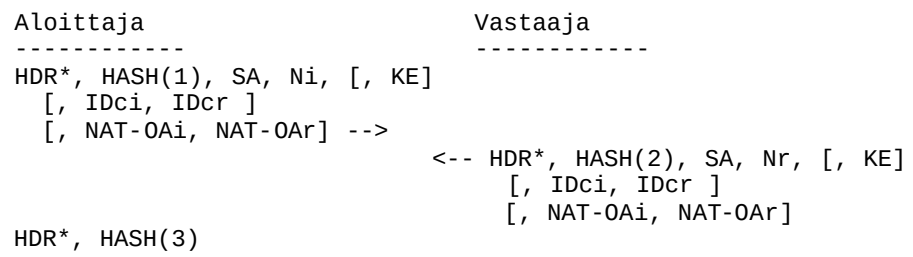
Kuljetustavassa molempien osapuolten pitää lähettää molemmat alkuperäiset osoitteet vastapuolelle. Tunnelitavassa tämä ei ole tarpeellista, eikä NAT-OA-kuormia tulisi lähettää. [draft-ietf-ipsec-nat-t-ike-07;8-9]

NAT-OA-kuormat lähetetään vaiheen 2 ensimmäisessä ja toisessa viestissä. Aloittajan pitää lähettää kuorma, jos se ehdottaa UDP-kapseloitua kuljetustapaa, ja vastaajan pitää lähettää kuorma, jos se valitsee UDP-kapseloidun kuljetustavan. [draft-ietf-ipsec-nat-t-ike-07;9] NAT-OA-kuorman rakenne on esitelty kuvassa 16.



Kuva 16. NAT-OA-kuorman rakenne.

NAT-OA-kuorman tunniste on 16. [draft-ietf-ipsec-nat-t-ike-07;9]
Kuva 17 esittää vaihetta 2 NAT-OA-kuormilla.

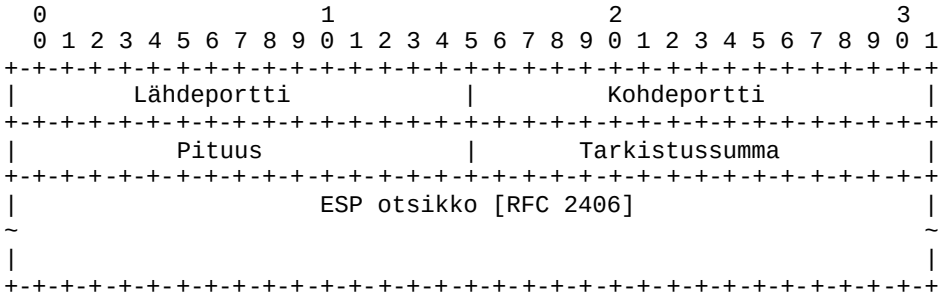


Kuva 17. Vaihe 2 NAT-OA-kuormilla.

Ensimmäinen yhteys -tiedonantotyypin IKE-viestissä olevat lähde-IP-osoite ja portti ovat merkityksettömiä, mikäli vastapuolen osoitteelle tehdään NAT-muunnos, joten niitä ei tule käyttää perusteena valittaessa mitkä IKE tai IPsec SA:t poistetaan. Mutta ID-kuormaa vastapuolelta pitäisi sen sijaan käyttää, eli ensimmäinen yhteys -tiedonannon vastaanottaneen osapuolen tulisi hävittää kaikki SA:t, jotka liittyvät samaan ID-kuormaan. [draft-ietf-ipsec-nat-t-ike-07;9]

6. NAT-Traversal

Kuvassa 18 on esitelty UDP-kapseloitu ESP-otsikkokenttä [draft-ietf-ipsec-udp-encaps-06].



Kuva 18. UDP-kapseloitu ESP-otsikkokenttä

Kuvassa 18 on käytetty seuraavia merkintöjä:

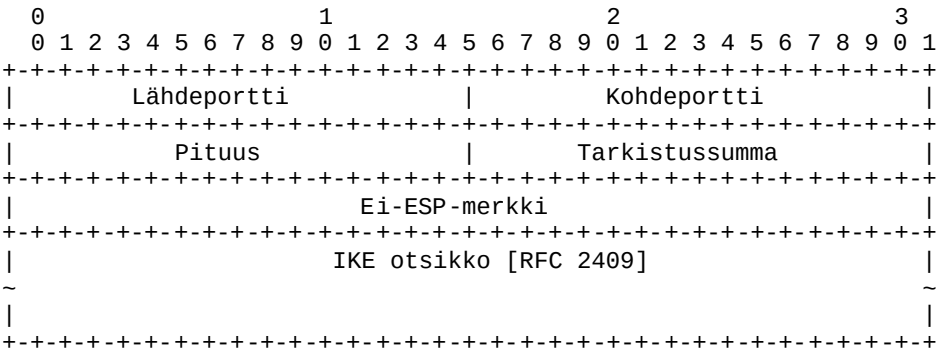
Lähdeportti on 16-bittinen luku, joka ilmoittaa UDP-protokollan lähdeportti-numeron.

Kohdeportti on 16-bittinen luku, joka ilmoittaa UDP-protokollan kohdeportti-numeron.

Pituus on 16-bittinen luku, joka ilmoittaa tavuina otsikkokenttä mukaan lukien kuorman pituuden.

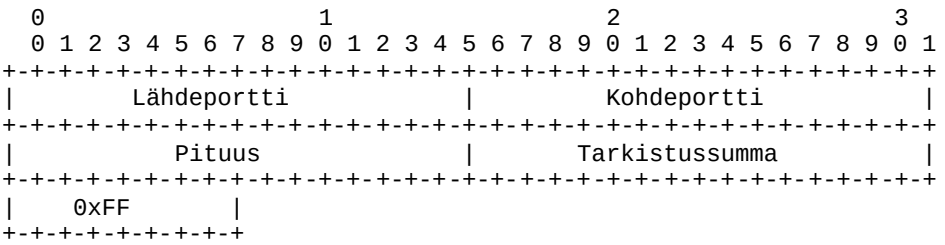
Tarkistussumma on 16-bittinen luku, joka pitää asettaa nollaksi.

Kuvan 18 otsikko on tavallinen RFC 768:n määrittelemä UDP-otsikko, jossa lähde- ja kohdeportit ovat samat kuin kellutetussa IKE-liikenteessä. Tarkistussumma on nolla. ESP-otsikon SPI-numero ei saa olla nolla, jotta UDP-kapseloidut ESP-paketit erotetaan IKE-paketeista. [draft-ietf-ipsec-udp-encaps-06] Kuvissa 19 ja 20 on esitetty kellutettu IKE-otsikko ja NAT-elossapitopaketti.



Kuva 19. Kellutettu IKE otsikko [draft-ietf-ipsec-udp-encaps-06].

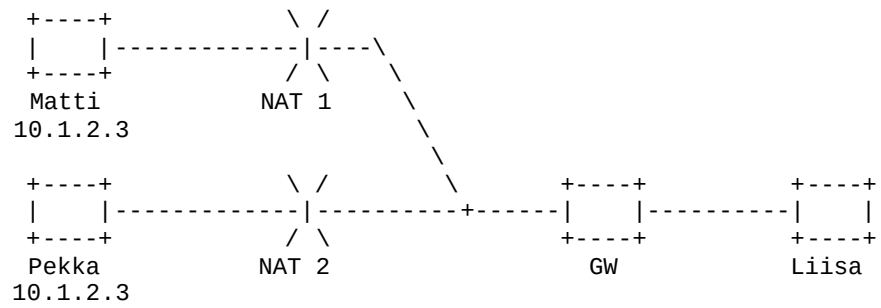
Kuvassa 19 Ei-ESP-merkki on 32 bittinen luku, joka on asetettu nollaksi, jotta IKE-paketit erotetaan UDP-kapseloiduista ESP-paketeista.



Kuva 20. NAT-elossapitopaketti [draft-ietf-ipsec-udp-encaps-06].

Kuvassa 20 0xFF on 8-bittinen luku, jonka arvoksi asetetaan heksadesimaalina 0xFF.

Tunnelitapaa käytettäessä alkuperäinen IP-otsikko voi sisältää osoitteita, jotka eivät ole käyttökelpoisia kohdeverkossa. Paketeille on tehtävä NAT-muunnos. Se tarvitaan myös tapauksessa, jossa kahdella päätelaitteella on sama IP-osoite. (ks. kuva 21).



Kuva 21. Kahdella päätelaitteella on sama IP-osoite.

Kuvan 21 esimerkissä Liisan on kyettävä erottamaan Matilta ja Pekalta tulevat paketit toisistaan. Tämä mahdollistetaan NAT:lla GW:ssä.

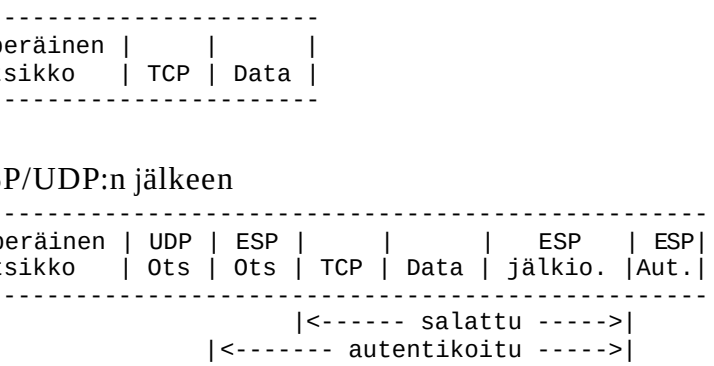
Kuljetustapaa käytettäessä TCP- ja UDP-otsikot sisältävät väärän tarkistussumman muuttuneen IP-osoitteen osalta. Ongelma pitää korjata seuraavasti [draft-ietf-ipsec-udp-encaps-06]:

a) Vähennä IP-otsikossa olleet IP-osoitteet tarkistussummasta ja lisää tarkistussummaan oikeat osoitteet (NAT-OA)

b) Laske tarkistussumma uudelleen (hitaampaa kuin kohta a)

c) Kytke tarkistussumman tarkistus pois.

Siirtotavan ESP:n kapselointi tehdään kuvassa 22 esitetyllä tavalla.



Kuva 22. Kuljetustavan UDP-kapselointi. [draft-ietf-ipsec-udp-encaps-06]

UDP-kapselointia varten:

1) Suoritetaan normaali ESP-kapselointi

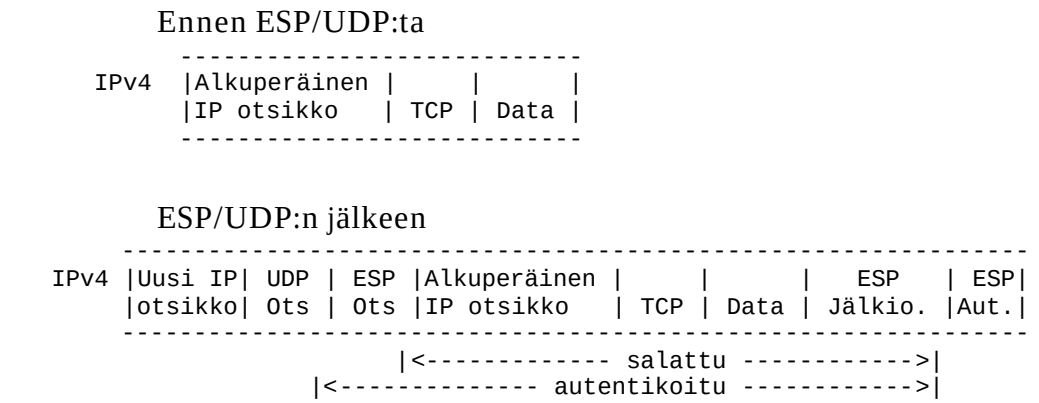
2) Lisätään UDP-otsikko alkuperäisen IP-otsikon ja ESP-otsikon väliin

3) Päivitetään IP-otsikossa kokonaispituus-, protokolla- ja otsikon tarkistus-

summakentät vastaamaan muutosten jäkeistä IP-pakettia.

Kuljetustavan ESP:n kapselointi puolestaan poistetaan seuraavasti [draft-ietf-ipsec-udp-encaps-06]:
1) UDP-otsikko poistetaan paketista
2) Päivitetään IP-otsikon kokonaispituus-, protokolla- ja tarkistussummakentät vastaamaan muutoksen jälkeistä pakettia
3) Suoritetaan normaali ESP-kapseloinnin poisto
4) Tehdään tarvittaessa NAT.

Tunnelitavassa ESP-kapselointi tehdään kuvassa 23 esitetyllä tavalla.



Kuva 23. Tunnelitavan UDP-kapselointi. [draft-ietf-ipsec-udp-encaps-06]

UDP-kapselointia varten:
1) Suoritetaan normaali ESP-kapselointi
2) Lisätään UDP-otsikko uuden IP-otsikon ja ESP-otsikon väliin
3) Päivitetään uudessa IP-otsikossa kokonaispituus-, protokolla- ja otsikon tarkistussummakentät vastaamaan muutosten jäkeistä IP-pakettia

Tunnelitavan ESP-kapseloinnin poisto [draft-ietf-ipsec-udp-encaps-06]:
1) UDP-otsikko poistetaan paketista
2) Päivitetään IP-otsikon kokonaispituus-, protokolla- ja tarkistussummakentät vastaamaan muutoksen jälkeistä pakettia
3) Suoritetaan normaali ESP-kapseloinnin poisto
4) Tehdään tarvittaessa NAT-muunnos

NAT-elossapitopakettien tarkoituksena on estää NAT osoitemuunnosten vanheneminen niin kauan kuin päätepisteiden välillä on olemassa yksi tai useampi IKE tai IPsec SA, tai sellainen on ollut olemassa enintään *N*

minuuttia aikaisemmin. *N* on paikallisesti määriteltävissä oleva muuttuja, jonka oletusarvo on 5. [draft-ietf-ipsec-udp-encaps-06]

Viestivät osapuolen pitää lähettää NAT-elossapitopaketti, mikäli sellaiselle on tarve havaittu ja mitään muuta pakettia ei toiselle osapuolelle ole lähetetty *M* sekuntiin. *M* on paikallisesti määriteltävissä oleva muuttuja, jonka oletusarvo on 20. [draft-ietf-ipsec-udp-encaps-06]

7. NAT-Traversal:n arviointia

Ratkaisuluonnoksessa NAT-Traversal:lla ratkaistaan kaikki tärkeimmät NAT:n ja IPsec:n yhteensopivuusongelmat. Toisaalta se myös monimutkaistaa entisestään IKE-neuvottelua, jonka virheetön ja kattava toteutus on jo muutenkin vaikeaa.

Yleisemmin voisi vielä pohtia, onko yhteensopivuusongelman ratkaisu mielekästä IKE- ja IPsec-protokollia muuttamalla. Samanlaisia yhteensopivuusongelmia NAT:n kanssa on myös muilla protokollilla, kuten MIP:llä (Mobile IP). Voisiko NAT-T:n toteuttaa yleisempänä ratkaisuna niin että se voisi UDP-kapseloida muutakin kuin ESP:tä ja voisiko tarpeen NAT-T:lle havainnoida muualla kuin osana IKE-neuvottelua?

Viiteluettelo

[RFC 1918]. Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot and E. Lear, Address Allocation for Private Internets. February 1996.

[RFC 2401]. S. Kent and R. Atkinson, Security Architecture for the Internet Protocol. November 1998.

[RFC 2402]. S. Kent and R. Atkinson, IP Authentication Header. November 1998.

[RFC 2406]. S. Kent and R. Atkinson, IP Encapsulating Security Payload (ESP). November 1998.

[RFC 2408]. D. Maughan, M. Schertler, M. Schneider and J. Turner, Internet Security Association and Key Management Protocol (ISAKMP). November 1998.

[RFC 2409]. D. Harkins and D. Carrel, The Internet Key Exchange (IKE). November 1998.

[RFC 2663]. P. Srisuresh and M. Holdrege, IP Network Address Translator (NAT) Terminology and Considerations. August 1999.

[draft-ietf-ipsec-nat-reqts-05]. Bernard Aboba and William Dixon, IPsec-NAT Compatibility Requirements. Internet-Draft. August 2003.

[draft-ietf-ipsec-nat-t-ike-07]. Tero Kivinen, Negotiation of NAT-Traversal in the IKE. Internet-Draft. Septemper 2003.

[draft-ietf-ipsec-udp-encaps-06]. Ari Huttunen, UDP Encapsulation of IPsec Packets. Internet-Draft. January 2003.

Asiantuntijajärjestelmistä

Kirsi Varpa

Tiivistelmä

Tutkielmassani käsittelen asiantuntijajärjestelmiä ja esittelen lyhyesti niiden käyttötarkoituksia sekä aloja, joilla niitä hyödynnetään. Lisäksi kuvaan niiden yleisiä rakenneosia ja kerron suppeasti asiantuntijajärjestelmissä käytetyistä päättelymekanismeista. Esimerkkitapauksena esittelen huimaustautien diagnosoinnin tukena käytettävän ONE (Otoneurological expert system) -asiantuntijajärjestelmän.

Avainsanat ja -sanonnat: asiantuntijajärjestelmä, asiantuntijajärjestelmän rakenne, asiantuntijajärjestelmän tarkoitus, ONE.
CR-luokat: I.2.1, I.2.3

1. Johdanto

Tietokoneavusteisia (computer-aided) järjestelmiä käytetään useimmilla aloilla päätöksenteon tukemiseen ja yleensä erilaisten työtehtävien helpottamiseen. Tällaisiksi järjestelmiksi voidaan luokitella myös *asiantuntijajärjestelmät* (expert systems), jotka pyrkivät ratkaisemaan tekoälyn (artificial intelligence) avulla tarkasti rajatun sovellusalueen asiantuntijatietämystä vaativia ongelmia [Viikki, 2002; Waterman, 1986]. Niiden ongelmanratkaisussa, toisin sanoen *päättelyssä* (reasoning, inference), hyödynnetään kohdealueen asiantuntijoilta kerättyä tietämystä, joka tallennetaan sekä tietokoneohjelman että asiantuntijoiden ymmärtämään muotoon [Metaxiotis and Samouilidis, 2000]. Useimmiten tietämys kuvataan symbolein, jotka vastaavat sovellusalueen käsitteitä [Waterman, 1986].

Tutkielmani tarkoitus on selventää asiantuntijajärjestelmien käyttötarkoitusta ja toimintaa sekä kuvata niiden rakenneosia. Lisäksi yhtenä tarkoituksena on esitellä järjestelmien ongelmanratkaisussa eli päättelyssä käytettyjä päättelymekanismeja. Aihetta olen lähestynyt kirjallisuuskartoituksen kautta sekä osin oman kokemuksen pohjalta. Esimerkki-tapauksena asiantuntijajärjestelmistä esittelen huimaustauteihin erikoistuneen ONE (Otoneurological expert system) -järjestelmän. Tarkastelen ONE:n rakenneosien ja päättelyme-

netelmän toteutusta sekä esitän tutkimustuloksia ONE:n ja lääkäreiden päättelyn vertailusta.

2. Asiantuntijajärjestelmien käyttötarkoitus

Asiantuntijajärjestelmät ovat tietokoneohjelmia, joita käytetään ongelman-ratkaisun ja päätöksenteon tukena [Auramo and Juhola, 1996], kohdealueen datan tulkinnassa, tapahtumien seurausten ennustamisessa, tautien tai vikojen määrityksessä, kokoonpanojen ja tapahtumien suunnittelussa, kohteiden tilan seurannassa, virhetoimintojen parannuskeinojen etsinnässä ja korjauksessa, kouluttamisen ja hallinnan apuna [Waterman, 1986] ja monissa muissa tehtävissä. Ne eroavat perinteisistä tietokoneohjelmista muun muassa siinä, että ne kykenevät oppimaan virheistään, käsittelemään tietämystä pelkän datan sijasta ja perustelemaan, miten ne ovat päätyneet ehdottamaansa lopputulokseen [Waterman, 1986]. Asiantuntijajärjestelmät voivat päätöksenteon yhteydessä muistuttaa päätöksentekijää asioista, jotka tulisi ottaa huomioon päätöstä tehdessä, ehdottaa, mitä tietoja pitäisi vielä saada lisää, jotta ratkaisu olisi luotettavampi, tai esittää erilaisia vaihtoehtoja sille, miten kyseisessä tilanteessa tulisi toimia.

Inhimillisten asiantuntijoiden tapaan asiantuntijajärjestelmät voivat tehdä virheitä [Waterman, 1986], koska niiden päättelyssä pyritään imitoimaan asiantuntijoiden päättelyä ja siinä käytetään ihmisiltä kerättyä tietämystä. Tietämyksen mallintaminen koneen ymmärtämään muotoon hävittää aina osan kohteen tiedoista, sillä mallit ovat vain pelkistyksiä todellisuudesta. Näin ollen päättelyn tuloksiin tulee suhtautua objektiivisesti ja huomioida se, että tulokset ovat vain järjestelmän esittämiä suosituksia. Etenkin lääke-tieteellisissä sovelluksissa on tärkeää muistaa, että kun asiantuntijajärjestelmiä käytetään potilaiden diagnosoinnissa, on järjestelmä useimmiten oikeassa, mutta toisinaan sekin voi erehtyä.

Asiantuntijajärjestelmät luodaan yleensä vain tietylle sovellusalueelle, minkä vuoksi niiden toiminta-ala on toisinaan hyvinkin kapea. Sovellusalueiden suppeuden takia saadaan järjestelmien tietämyskantojen tietämys kohdealueesta syvällisemmäksi ja tarkemmaksi ja samalla myös niiden toimintamenetelmät asiantuntijamaisemmiksi [Waterman, 1986].

Kuten asiantuntijajärjestelmien käyttötoiminnoista voi päätellä, ne soveltuvat monenlaisiin aloihin. Järjestelmiä hyödynnetäänkin monipuolisesti niin fysiikassa, kemiassa, matematiikassa, lääketieteessä, maataloudessa, oikeustieteessä, teollisuuden tuotannossa kuin informaation hallinnassa ja prosessien kontrolloinnissa [Waterman, 1986]. Etenkin lääketieteellisiä asian-tuntija-

järjestelmiä on kehitetty runsaasti 30 viime vuoden aikana [Chae, 1998], mutta myös muilla aloilla niiden määrä on kasvanut räjähdysmäisesti.

Teollisuudessa asiantuntijajärjestelmiä käytetään suhteellisesti enemmän varsinaisten päätösten tekoon kuin lääketieteessä, jossa ne toimivat enimmäkseen potilaiden diagnosoinnin tukena sekä hoitotoimenpiteiden neuvojina [Chae, 1998]. Lääketieteessä asiantuntijajärjestelmiä hyödynnetään tämän lisäksi muun muassa datan tulkinnassa, elintoimintomonitorien yhteydessä tarkkailemassa potilaan tilaa ja hälyttämässä äkillisistä tilan muutoksista, huomauttamassa potilaiden hoitosuunnitelmien epäjohdonmukaisuuksista ja lääketieteen kandidaattien kouluttamisessa [Metaxiotis and Samouilidis, 2000; Waterman, 1986]. Järjestelmät voivat myös ehdottaa lääkärille, mitä jatkotutkimuksia potilaalle olisi hyvä tehdä diagnoosin tarkentamiseksi [Viikki and Juhola, 2001]. Yksi ensimmäisistä ja tunnetuimmista lääketieteen asiantuntijajärjestelmistä on MYCIN, joka kehitettiin 1970-luvun puolivälissä diagnosoimaan ja hoitamaan veritartunta-tauteja [Chae, 1998]. Se on kuitenkin ollut vain tutkimuskäytössä.

Muilla sovellusalueilla käytetään asiantuntijajärjestelmän tietämystä apuna muun muassa teollisuuden yritysten tuotannon automaatiovaiheiden aikataulutuksessa ja valvonnassa, taloussuunnitelmien laadinnassa ja jopa vielä tuntemattomien yhdisteiden kemiallisten ominaisuuksien selvittämisessä [Metaxiotis and Samouilidis, 2000]. Ensimmäinen kemiallinen asiantuntijajärjestelmä DENDRAL kehitettiin jo 1960-luvun puolivälissä, ja se olikin ensimmäisiä asiantuntijajärjestelmiä maailmassa [Waterman, 1986].

3. Asiantuntijajärjestelmien rakenne

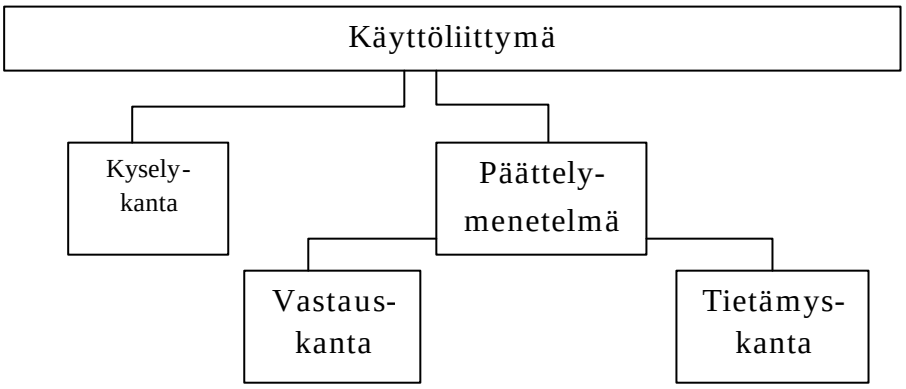
Jokaisella asiantuntijajärjestelmällä on joitakin niille kaikille yhteisiä piirteitä. Tällaisiksi piirteiksi tai osiksi voidaan laskea ainakin *tietämyskanta* (knowledge base), *päättelymenetelmä* (inference engine), käyttäjän ja ohjelman välinen käyttöliittymä [Metaxiotis and Samouilidis, 2000], *kyselykanta* (query base) ja vastauskanta, joka on yleensä tiedosto tai tietokanta, jonne tallennetaan asiantuntijajärjestelmällä kerätty data. Kuva 1 näyttää näiden rakenteiden väliset yhteydet. Tässä luvussa esittelen yleisellä tasolla kysely- ja tietämyskantojen, päättelymenetelmien ja käyttöliittymien tarkoitusta.

3.1. Kyselykanta

Asiantuntijajärjestelmissä kyselykannat sisältävät tiedon siitä, kuinka käyttäjän kanssa kommunikoidaan eli mitä kysymyksiä ja miten ne esitetään käyttä-

jälle [Waterman, 1986]. Kyselykannan voidaankin sanoa sisältävän ohjelman käyttöliittymän rakennusohjeet.

Kyselykannassa voidaan ilmaista esimerkiksi se, miten ohjelman eri elementit liittyvät toisiinsa eli miten sen eri osioista voidaan liikkua eteenpäin. Lisäksi kyselykanta voi sisältää tiedon siitä, miten käyttäjän antamaan syötteeseen tulee reagoida. Toisin sanoen käyttäjälle esitettyihin kysymyksiin on voitu asettaa vastausvaihtoehtoja, joiden mukaan järjestelmä osaa jatkaa toimintaansa. Tarvittaessa järjestelmä vaatii tarkennusta saamiinsa vastauksiin, jollei sen kyselykannasta löydy sellaista toimintoa, jota käyttäjä tarjoaa. Näin toimitaan etenkin tekstipohjaista käyttöliittymää käytettäessä. Käyttäjän antaman syötteen perusteella asiantuntijajärjestelmä voi muun muassa pyytää käyttäjältä lisätietoja käsiteltävästä aiheesta, muuntaa saamaansa dataa toisenlaiseksi tai aloittaa päättelyprosessin.



Kuva 1. Asiantuntijajärjestelmän rakenneosat.

3.2. Tietämyskanta

Tietämyskantaan kerätään asiantuntijoiden tietämystä sovellusalueesta kyselemällä heiltä heidän ongelmanratkaisussaan käyttämiään menetelmiä, strategioita ja sääntöjä [Waterman, 1986], jotka muunnetaan sellaiseen muotoon, jota ymmärtävät niin asiantuntijat kuin asiantuntijajärjestelmäkin. Tietämystä kerätään asiantuntijoiden lisäksi sovellusalueen kirjallisuudesta, tietokannoista ja tapaustutkimuksista [Waterman, 1986]. Tietämyskannan sisältämät kuvaukset, ns. tietämysrakenteet, voivat olla erilaisia sääntöjä, semanttisia verkkoja, kehyksiä [Waterman, 1986] tai predikaattilogiikan lauseita [Metaxiotis and Samouilidis, 2000]. Ne tallennetaan tietämyskantaan, joka on asiantuntijajärjestelmän älykkään toiminnan perusta. Tietämyskannan sisäl-

tämä tietämys voi vaikuttaa myös asiantuntijajärjestelmän suorituskykyyn [Viikki, 2002; Viikki and Juhola, 2001].

Tietämyskanta sisältää faktoja (dataa) ja kuvauksia. Kuvaus voi olla sääntö, jossa luonnollisella kielellä kuvataan muuttuvan järjestelmän prosesseja [Chae, 1998; Waterman, 1986]. Kuvaus voi olla myös kehys tai semanttinen verkko, jotka ovat molemmat kehyspohjaisia kuvauksia. Niissä perustana ovat hierarkkisesti järjestetyt, toisiinsa yhteydessä olevien solmujen verkot (network of nodes). Solmut kuvaavat käsitteitä tai sovellusalueen olioita; verkkolähestymistavassa ne kuvaavat myös tapahtumia. Kehyskuvauksessa pääosassa ovat solmuihin liittyvät ominaisuudet ja niiden arvot toisin kuin semanttisissa verkoissa, joissa painopiste on solmujen välisissä suhteissa. [Waterman, 1986]

3.3. Päättelymenetelmä

Asiantuntijajärjestelmän päättelymenetelmä sisältää tiedon siitä, kuinka järjestelmä toimii ratkaistessaan ongelmia [Waterman, 1986]. Tavallaan se vastaa kohteesta annettujen tietojen yhteydestä tietämyskantaan, koska päättelymenetelmä ratkaisee, miten ja missä järjestyksessä tietämyskannan kuvauksia käytetään päättelyssä.

Asiantuntijajärjestelmän päättelymekanismin rakenteeseen vaikuttaa se, mikä on asiantuntijajärjestelmän sovellusalue ja miten sen tietämys on kuvattu ja järjestetty [Waterman, 1986]. Esimerkiksi *sääntöpohjaisessa menetelmässä* (rule-based reasoning) tietämyskannan kuvaukset esitetään sääntöinä, jolloin päättely tapahtuu IF-lauseiden toiston, ns. päättelyketjun, perusteella [Chae, 1998]. Säännöt ovat muodoltaan sellaisia, että niissä on ensin annettu jokin ehto (IF), jonka täyttyessä (THEN) suoritetaan tiettyjä toimenpiteitä kohdistuen joko käyttäjään, ohjelman suoritukseen tai päättelyn lopputulokseen [Waterman, 1986]. Sääntöpohjaisen menetelmän hyvänä puolena on se, että säännöt ovat käyttäjälle luonnollinen tapa ilmaista päättelyn etenemistä. Tällöin päättely on helposti ymmärrettävissä ja arvioitavissa. Huonoa menetelmässä on puolestaan se, että sääntöjen ehtojen on täysin vastattava käyttäjän antamaa syötettä, jotta toimenpiteet voidaan toteuttaa. [Chae, 1998]

Useimmiten asiantuntijajärjestelmissä käytetty päättelymenetelmä on edellä mainittu sääntöpohjainen päättely. Sen lisäksi päättelymenetelminä käytetään muun muassa *Bayesin menetelmää* (Bayesian learning), *lähimmän naapurin menetelmää* (nearest neighbor learning) ja *neuroverkkoja* (neural networks) [Chae, 1998]. Bayesin ja lähimmän naapurin menetelmät ovat hahmontunnistusmenetelmiä, jotka luokittelevat sovellusalueen kohteita niiden

ominaisuuksien perusteella. Bayesin menetelmässä hahmojen luokittelu perustuu todennäköisyyksiin [Mitchell, 1997], jotka on edeltä käsin asetettu tulosluokille ja niiden havainnoille. Menetelmässä oletetaan, että päättelyn tulokseksi voidaan saada vain yksi sovellusalueen luokista, ts. tulokset ovat toisensa poissulkevia. Esimerkiksi potilaita diagnosoitaessa oletetaan, että potilaalla voi esiintyä korkeintaan yksi taudeista kerrallaan. Huono puoli Bayesin menetelmässä on se, ettei se salli hahmojen päivittämistä. [Chae, 1998]

Lähimmän naapurin menetelmä perustuu esimerkkipohjaiseen oppimiseen (instance-based learning), jossa oppiminen tapahtuu opetustapausten tallentamisen kautta. Menetelmä luokittelee uuden kohteen siten, että se etsii opetustapauksistaan lähellä kohdetta olevia hahmoja. Jos kyseessä on k:n lähimmän naapurin menetelmä, haetaan kohteen naapureita k kappaletta. Jos k on 1, haetaan vain kohteen lähin naapuri. Naapureiden löytämisen jälkeen menetelmä tutkii lähimpiä naapureita ja selvittää luokat, joihin ne kuuluvat. Uusi kohde kuuluu siihen luokkaan, jota esiintyy eniten lähimpien naapureiden keskuudessa. [Mitchell, 1997]

Neuroverkot jäljittelevät biologisia hermoverkkoja [Chae, 1998]. Verkon perustana on joukko toisiinsa yhteydessä olevia yksikköjä (processing elements) [Mitchell, 1997], jotka ottavat vastaan syötettä. Jos yksikön virittymisehto täyttyy, muodostaa se saadusta syötteestä laskusäännöillä tulostetta muille yksiköille. Neuroverkon tieto on varastoituna yksiköiden välisiin yhteyksiin, joille on asetettu painoarvot. Tieto syötetään neuroverkolle kaksiosaisessa prosessissa: ensin käyttäjä määrittelee, mitkä yksiköt ovat toisiinsa yhteydessä ja miten niiden vuorovaikutus etenee, ja sen jälkeen neuroverkolle opetetaan yhteyksien arvot opetusjoukon avulla. [Callan, 2003] Neuroverkkojen hyvä puoli on siinä, että ne kykenevät analysoimaan nopeasti ja virheettömästi suuria määriä dataa [Chae, 1998]. Huonona puolena on kuitenkin se, että koska neuroverkon tietämys sijaitsee yksiköiden yhteyksissä ja se on hajautettuna ympäri järjestelmää, eivät neuroverkot kykene esittämään perusteluja päättelytulokselleen [Chen and Rada, 1998]. Tämän takia niiden tuottamien tulosten luotettavuuden arviointi on hankalaa.

3.4. Käyttöliittymä

Asiantuntijajärjestelmissä käyttöliittymä on keskeisessä osassa käyttäjän ja järjestelmän välisessä vuorovaikutuksessa, sillä sen kautta järjestelmä asettaa toimintonsa käyttäjän ulottuville [Chen and Rada, 1998]. Käyttöliittymän avulla asiantuntijajärjestelmä kerää sovellusalueensa kohteesta tietoja, joita se

käyttää päättelyssään. Lisäksi sen kautta järjestelmä esittää käyttäjälle päättelyn tulokset sekä niiden perustelut [Metaxiotis and Samouilidis, 2000]. Tulosten perustelemiskyky on hyvin tärkeä ominaisuus asiantuntijajärjestelmissä [Chen and Rada, 1998], sillä käyttäjien on kyettävä arvioimaan järjestelmän ehdotuksien luotettavuutta ja tässä sekä päättelyprosessit että perustelut lopputulokseen päätymiselle toimivat hyvänä apuna.

Käyttöliittymät voivat olla graafisia, jolloin on mahdollista kuvata taulukoilla ja muilla visuaalisilla elementeillä päättelyn tuloksia. Ne voivat olla myös tekstipohjaisia, jolloin kommunikointi tapahtuu vuorovaikutussuhteen kautta. Järjestelmä reagoi tekstipohjaisessa kommunikoinnissa käyttäjän syötteeseen ja etenee sen pohjalta toiminnassaan eteenpäin.

4. Huimaustautiasiantuntijajärjestelmä

Esimerkkinä asiantuntijajärjestelmistä esittelen ONE:n [Auramo et al., 1993], jonka käyttötarkoituksena on toimia lääkärin tukena huimaussairauksien diagnosoinnissa [Auramo, 1999; Viikki, 2002] sekä perehdyttää lääketieteen opiskelijoita kohdealueeseen [Kentala et al., 1995; Viikki and Juhola, 2001]. ONE rakentuu pääasiassa viidestä oleellisesta komponentista, jotka ovat samoja kuin kuvassa 1 esitetyt asiantuntijajärjestelmän rakenneosat. Nämä komponentit ovat kyselykanta, tietämyskanta, päättelymekanismi, käyttöliittymä ja tietokanta [Auramo and Juhola, 1995; Auramo et al., 1993]. Tässä luvussa esittelen näitä rakenneosia ONE:n kannalta. Tämän jälkeen esitän lyhyesti vertailutuloksia ONE:n ja lääkäreiden diagnosointikyvystä.

4.1. Kyselykanta

Kyselykannalla on merkittävä asema huimaustautiasiantuntijajärjestelmän toiminnassa, koska se sisältää järjestelmässä esitettävät kysymykset ja niiden vastausvaihtoehdot. Näiden perusteella generoidaan myös asiantuntijajärjestelmän kyselylomakkeet, toisin sanoen muodostetaan järjestelmän graafinen käyttöliittymä. Ote 1 sisältää pienen osan ONE:n kyselykannasta, joka esittelee Tinnitus-näkymän alkua.

```
IF @SYM_TINNITUS
BEGIN SCREEN
      "Tinnitus"
      HELPI4.ONE
@AGE_TIN_SYM : REALBAR
      "When did tinnitus first occur?"
       "The time elapsed: 0 = no tinnitus, 1=days, 2=1-4 weeks, 3=1-4 months, 4=<1 year, 5=1-4 years, 6=4-10 years, 7=10-15 years,
8=15-20 years, 9=more"
      0
      9
      1
      3
.
@TINNITUS : REALBAR
      "Severity of the tinnitus"
      "Tinnitus / ringing ears 0 = no handicap, 1 = slight handicap, 2 = moderate handicap, 3 = severe handicap"
      0
      3
      1
      0
.
```

Ote 1. Kyselykanta sisältää käyttöliittymän rakennusohjeen.

Kysymysten lisäksi kyselykanta sisältää vastauksista aritmeettisilla, relationaalisilla ja loogisilla operaatioilla johdettuja muuttujia [Auramo et al., 1993]. ONE:n kysymykset ovat jotakin seuraavista neljästä tyypistä: loogisia (kyllä-ei-tyyppisiä, yesno), valintalistasta valittavia (luokittelevia, realbar), numeerisia (prosentti- tai reaalilukuja, bar) tai merkkijonoja (textbar) [Auramo and Juhola, 1995; Auramo et al., 1993; Kentala et al., 1998].

Molemmat otteen 1 kysymykset ovat valintalistatyyppisiä. Tämän näkee siitä, että muuttujan nimen, esimerkiksi @TINNITUS, perässä lukee REALBAR ja lisäksi siitä, että aputekstin kohdalla on selitetty, mitä mikäkin annettu arvo tarkoittaa. Tämän jälkeen on esitetty, mikä on vastauksen minimiarvo ja maksimiarvo, kuinka suurin hyppäyksin vastausten arvot etenevät sekä mikä on vastauksen oletusarvo. Viimeksi mainitut määrittelyt esiintyvät vain numeeristen ja valintalistatyyppisten kysymysten yhteydessä. Kyselykanta sisältää myös viitteen näkymän aputekstiin (HELPI4.ONE) sekä kysymykseen liittyvät vihjetekstit.

4.2. Tietämyskanta

Tietämyskanta sisältää huimaustautien kuvaukset [Auramo and Juhola, 1995; Auramo et al., 1993], joita asiantuntijajärjestelmä tarvitsee potilaan taudinmäärityksessä. Tietämyskannassa on tällä hetkellä 18 huimaussairauden kuvaukset, jotka on laadittu lääketieteellisten asiantuntijoiden tietämyksen ja kokemusten perusteella, satojen potilaiden datan sekä alan kirjallisuuden avulla [Auramo and Juhola, 1996; Kentala et al., 1998; Viikki, 2002].

Käytännössä ONE:n tietämyskanta rakentuu siten, että eri huimaustaudeille on määritelty niiden kannalta *välttämättömät* (necessary) ja *tukevat* (supportive) kysymykset [Auramo and Juhola, 1996; Auramo et al., 1993; Kentala et al., 1998], jotka huomioidaan päättelyssä. Välttämättömät kysy-

mykset ovat taudin kannalta keskeisiä, joten niiden vastausten puuttumisesta huomautetaan käyttäjälle diagnoosin tuloksen esityksen yhteydessä. Samoin ilmoitetaan, jos annettu vastaus ei sovi taudinkuvaan. Tukevat kysymykset sen sijaan voivat jäädä ilman vastausta [Kentala et al., 1998]. Koska ne eivät ole merkittävässä asemassa diagnosoinnin kannalta, vastausten puuttumisesta ei ilmoiteta käyttäjälle. Lisäksi ONE:ssa on nyt määritelty *hylättävissä olevat* (rejectable) kysymykset niille taudeille, jotka eivät voi jostakin syystä olla kyseessä. Kysymys vastaa yhtä tietokannan taulun attribuuttia, ominaisuutta, joten tässä yhteydessä on mahdollista puhua myös ominaisuudesta kysymyksen sijaan.

Taudinmäärityksessä tarvittavien ominaisuuksien määrä vaihtelee yhdestä useisiin kymmeniin, ja jotta tauti voisi olla kunnolla diagnosoitavissa, tulee kaikkiin sen kannalta välttämättömiin kysymyksiin vastata taudin kannalta sopivasti [Viikki, 2002]. Jos potilaalla ei ole huimaustaudin kannalta välttämättömäksi luokiteltua ominaisuutta, ts. oiretta, on tällöin todennäköistä, ettei hänellä myöskään ole kyseistä tautia [Auramo et al., 1993]. Sen sijaan se, että potilaalla esiintyy näitä oireita, ei automaattisesti tarkoita sitä, että potilaalla olisi tauti, jonka kannalta oire on välttämätön.

Nykyisen ONE:n taudinmäärityksessä ei yleensä hylätä huimaustautia sen takia, että sen kannalta välttämättömiin kysymyksiin ei olisi vastattu tai vastaus ei ole taudin kannalta sopiva; ONE vain huomauttaa, ettei annettu vastaus sovi kyseiseen tautiin. Poikkeuksena on kuitenkin lapsuusiän kohtauksittainen hyvänlaatuinen huimaus (benign paroxysmal vertigo of childhood), jota voi esiintyä vain alle 10-vuotiailla lapsilla, joten se on hylättävissä potilaan liian korkean iän perusteella. Tämän taudin kuvauksessa onkin AGE_FIRST-muuttuja määritelty tyypiltään hylättävissä olevaksi, jolloin on mahdollista hylätä tauti myös päättelyn tuloksissa.

```
;----------------------------------
# Benign positional vertigo
;----------------------------------
AGE_FIRST      2 V
     INTERP
      0.000000 120.000000
      0.235756 0.0
      6.129666 0.8
      15.324165 1.6
      21.925344 51.5
      34.420432 96.1
      55.166994 89.8
      79.921415 0.3
      102.789784 0.5
      119.528487 0.3
      END
.
AGE_SYMPTOMS    0 T
     INTERP
      1 6
      1 10
      2 10
      3 20
      4 70
      5 90
      6 100
      7 100
      8 100
      9 100
      END
.
VER_ROTA  4 T
.
```

Ote 2. Tietämyskanta on asiantuntijajärjestelmän asiantuntijuuden perusta.

Jokaiselle ominaisuudelle on annettu *painoarvo* (weight value*)* ja tämän lisäksi ominaisuuden arvoille on asetettu *sopivuusarvot* (fitness values) kyseiseen huimaussairauteen. Painoarvo kertoo, kuinka merkittävä kyseinen ominaisuus on taudin kannalta. Sen arvo vaihtelee välillä -5–5, jossa -5 tarkoittaa, ettei ominaisuus liity ollenkaan kyseiseen tautiin ja 5 puolestaan ilmaisee ominaisuuden olevan hyvin merkittävä taudin kannalta [Auramo et al., 1993]. Voidaan siis sanoa, että mitä tärkeämpi ominaisuus on taudin kannalta, sitä suurempi on myös sen painoarvo. Sopivuusarvo puolestaan ilmaisee ominaisuuden arvon sopivuuden tautiin ja sen arvot ovat väliltä 0-1. [Auramo and Juhola, 1995; Auramo et al., 1993; Viikki, 2002; Viikki and Juhola, 2001] Tietämyskannan ominaisuuksien arvoille annetuista luvuista lasketaan interpoloimalla niiden sopivuusarvot.

Otteessa 2 on ote hyvänlaatuisen asentohuimauksen (benign positional vertigo) kuvauksesta. Sillä on tietämyskannan kuvauksessaan yhteensä 98 ominaisuutta, joista 16 on välttämätöntä ja 82 tukevaa ominaisuutta. Tämän huimaustaudin kannalta välttämätön kysymys on mm. AGE_FIRST, joka kertoo, minkä ikäisenä huimausoireet alkoivat potilaalla. Lisäksi on nähtävissä, että tautiin sopii se, että oireet alkavat 34–79 vuoden iässä. Esimerkkiotteessa näkyy myös kaksi tukevaa kysymystä, joista AGE_SYMPTOMS on valintalistatyyppinen ja VER_ROTA puolestaan looginen kysymys.

4.3. Päättelymenetelmä

Huimaustautien asiantuntijajärjestelmälle on toteutettuna oma päättelymekanisminsa, joka muistuttaa lähimmän naapurin hahmontunnistusta [Auramo and Juhola, 1995; Auramo and Juhola, 1996; Viikki and Juhola, 2001; Viikki, 2002]. Sen päättelymekanismi pyrkii hakemaan tietämyskannasta taudinkuvauksen, joka vastaa parhaiten potilaasta annettuja tietoja. Potilaasta syötetyistä tiedoista voi puuttua osia, sillä ONE:n päättelymekanismi huomioi myös datan puuttumisen [Auramo and Juhola, 1996]. Asiantuntijajärjestelmä ehdottaa lääkärille diagnoosiksi sitä taudinkuvausta, joka täsmää parhaiten potilaasta annettuun dataan.

Päättelyssään ONE huomioi potilaan oireet, lääketieteellisen historian ja kliiniset testit [Auramo and Juhola, 1995; Auramo et al., 1993]. Sen päättelymenetelmä on tehty asiantuntijalääkäreiden diagnosointiprosessia jäljitteleväksi. Päättelytulosten esityksen yhteydessä ONE pyrkii selittämään, minkä vuoksi tauti ei ole annettujen tietojen perusteella mahdollinen ja mitä tietoja olisi vielä tarpeen antaa varmemman diagnoosin luomiseksi [Kentala et al.,

1993]. Lisäksi se kertoo, mikäli taudin kannalta välttämättömien ominaisuuksien vastaukset eivät sovi taudin kuvaukseen.

ONE:n oman päättelymekanismin perustana ovat tietämyskannassa esitettyjen huimaustautien kuvauksissa määritellyt välttämättömät ja tukevat ominaisuudet sekä niiden painoarvot ja arvojen sopivuusarvot [Auramo et al., 1993; Kentala et al., 1998; Viikki, 2002; Viikki and Juhola, 2001]. Näiden perusteella se laskee tietämyskannan sisältämille taudeille pistemäärät, jotka kuvaavat, kuinka hyvin potilaan tiedot vastaavat taudin kuvausta [Auramo and Juhola, 1996]. Taudin pistemäärän *P(t)* laskemisessa käytetään kaavaa

$$P(t) = \frac{\sum_{a=1}^{m(t)} x(a)\, p(t,a) s(t,a,k)}{\sum_{a=1}^{m(t)} x(a)\, p(t,a)},$$

jossa

t tarkoittaa kyseessä olevaa tautia,

a ilmaisee, kuinka mones kysymys (ominaisuus) on kyseessä,

m(t) kertoo taudin kuvauksessa olevien kysymyksien määrän,

x(a) on looginen muuttuja, joka ilmaisee onko kysymykseen vastattu vai ei; sen arvo on 1, jos a:nnen ominaisuuden arvo tunnetaan kyseessä olevasta taudissa, ja 0, jos vastausta ei ole annettu,

p(t, a) ilmaisee taudin *t* attribuutin *a* painoarvon ja

s(t, a, k) puolestaan on taudin *t a*:nnen ominaisuuden arvon *k* sopivuusarvo kyseiseen tautiin; se voi saada arvoja nollasta yhteen. [Auramo and Juhola, 1996; Viikki and Juhola, 2001]

Syötettyjen tietojen perusteella lasketaan siis taudille edelle esitetyn kaavan mukaan pistemäärä (score) sekä pistemäärien ala- ja ylärajat, joilla pyritään kontrolloimaan puuttuvien tietojen aiheuttamaa epävarmuutta [Auramo and Juhola, 1996; Kentala et al., 1998; Viikki and Juhola, 2001]. Alaraja lasketaan puuttuvien ominaisuuksien arvojen pienimmistä, ts. vähiten sopivista, sopivuusarvoista ja vastaavasti yläraja niiden suurimpien, ts. täsmäävimpien, sopivuusarvojen perusteella [Auramo and Juhola, 1995; Auramo and Juhola, 1996; Auramo et al., 1993]. Auramo ja Juhola [1996] esittävät pistemäärien laskennan tarkemmin.

Mitä pienempi erotus on taudinkuvauksen saamalla pistemäärällä sekä sen ala- ja ylärajoilla, sitä suuremmalla todennäköisyydellä tauti on kyseessä [Auramo et al., 1993; Kentala et al., 1993]. Ala- ja ylärajan pistemäärän suuri erotus ilmaisee sen, että taudin kannalta välttämättömistä kysymyksistä

puuttuu vastauksia [Auramo and Juhola, 1996]. Tällä hetkellä tautien pistemäärien ala- ja ylärajat eivät vaikuta asiantuntijajärjestelmän päättelyn tulokseen [Auramo and Juhola, 1995], mutta on harkittu, että tautien järjestämisessä huomioitaisiin myös pisterajojen läheisyys tai vaihtoehtoisesti laskettaisiin kaikkien kolmen pistemäärän keskiarvo.

4.4. Käyttöliittymä

Huimaustautiasiantuntijajärjestelmän käyttöliittymä on toteutettu graafisesti Javalla. Käyttöliittymän muodostamisen kannalta kriittisessä asemassa on kyselykanta, sillä sen perusteella luodaan käyttöliittymän kyselyosion eri näkymät, ts. kysymykset ja vastausvaihtoehdot haetaan kyselykannasta.

Käyttöliittymän kyselyosion näkymät on jaoteltu kolmeen pääryhmään: yksi osio keskittyy potilaan oireisiin, toinen lääketieteelliseen historiaan ja kolmas löydöksiin. Oireosiossa esitetään kysymyksiä koskien muun muassa huimausta, kuulonalenemaa, tinnitusta ja päänsärkyä, lääketieteellisessä historiassa puolestaan kysytään esimerkiksi lääkkeiden käytöstä, pään ja korvan vammoista, ja löydöksissä puolestaan näytetään tutkimusten tuloksia, mm. kliinisten tutkimusten tulokset, kuulokäyrät ja erilaisten otoneurologisten tutkimusten dataa, kuten sakkadit, jotka kuvaavat silmän liikkeitä, ja posturografian, tasapainolevyn, tulokset [Kentala et al., 1995]. Posturografialla mitataan potilaiden horjuntaliikettä.

4.5. Tietokanta

Käyttöliittymän kautta kerätty data potilaasta ja hänen oireistaan, lääketieteellisestä historiastaan ja löydöksistään tallennetaan tietokantaan, josta data on mahdollista hakea esiin myöhempää tarkastelua varten joko ohjelman kautta yhden potilaan tiedot kerrallaan tai hakemalla kaikkien potilaiden data tiedostoon tilastollista analyysia varten. Potilastietoja voidaan täydentää myös jälkeenpäin [Kentala et al., 1998] hakemalla potilaan tiedot järjestelmään käsiteltäviksi. Data on pilkottu käyttöliittymän kyselyjen mukaisesti eli lähes jokaisella näkymällä on tietokannassa oma taulunsa, jonne data tallennetaan. Tällä pyritään välttämään puuttuvien tietojen esiintymistä tietokannan tauluissa.

ONE:n tietokantana käytetään MySQL-relaatiotietokantaa, jossa potilaan tiedot tallennetaan 29 tauluun, jotka on luotu näkymien mukaisesti. Päättelyssään ONE käyttää myös johdettuja muuttujia, mutta niitä ei ole tallennettu tietokantaan, koska ne saadaan haluttaessa johdettua varsinaisista muuttujista uudelleen [Auramo et al., 1993].

4.6. Asiantuntijoiden vertailua

ONE:n päättelyn oikeellisuuden ja myös toimivuuden testaamiseksi on verrattu inhimillisten asiantuntijoiden eli lääkäreiden ja asiantuntija-järjestelmän tekemiä diagnooseja potilaista [Juhola et al., 1995; Kentala et al., 1998]. Ensimmäiseen vertailuun osallistui neljä nuorta lääkäriä ja tällöin testitapauksia oli kaiken kaikkiaan 29. Diagnooseista ONE sai oikein 83 % (24 kpl) ja lääkärit diagnosoivat oikein keskimäärin 51 % (13, 19, 15 ja 12 kpl). Tuloksista havaittiin, että ONE teki enemmän oikeita diagnooseja kuin nuoret lääkärit, mutta se ei kuitenkaan vielä pärjännyt kokeneelle asiantuntijalle, joka diagnosoi kaikki tapaukset oikein. [Juhola et al., 1995]

Jonkin ajan kuluttua vertailu suoritettiin uudelleen ONE:n tietämyskannan muokkauksen jälkeen. Tällöin Kentala et al. [1998] vertasivat ONE:n ja kuuden lääkärin, joista yksi oli otoneurologian erikoislääkäri, diagnosointikykyä. Testijoukossa oli 23 potilasta ja niistä ONE diagnosoi oikein 65 % (15 kpl). Lääkärit saivat oikean diagnoosin keskimäärin 56 % (12, 15, 12, 11, 12 ja 15 kpl) potilaista, kun heillä oli käytettävissään samat lähtötiedot kuin ONE:lla. Kun lääkärit saivat täydelliset potilaskertomukset diagnosoinnin avuksi, saivat he selkeästi enemmän oikeita diagnooseja potilaille (15, 19, 15, 17, 13 ja 18 kpl). Tässä testissä ONE menestyi yhtä hyvin kuin erikoislääkärikin, kun diagnosoinnissa käytettiin samanlaisia pohjatietoja.

5. Yhteenveto

Asiantuntijajärjestelmien käyttö asiantuntijatietämystä vaativissa ongelmanratkaisutehtävissä on lisääntynyt huomattavasti viime aikoina. Samoin ovat laajentuneet sovellusalueet, joilla niitä hyödynnetään. Osasyynä kasvuun voi olla käyttäjien lisääntynyt luottamus sekä teknologiaan että asiantuntija-järjestelmien asiantuntevuuteen.

Jonkin verran on tutkittu ja vertailtu inhimillisten asiantuntijoiden ja asiantuntijajärjestelmien ongelmanratkaisua [Juhola et al., 1995; Kentala et al., 1998]. Vertailut osoittavat, että asiantuntijajärjestelmät pärjäävät tarkasti rajattujen sovellusalueiden ongelmanratkaisussa lähes yhtä hyvin kuin inhimilliset asiantuntijat, toisinaan jopa paremminkin.

Viitteet

[Auramo, 1999] Yrjö Auramo, *Construction of an Expert System to Support Otoneurological Vertigo Diagnosis*. Ph.D. Thesis. Report A-1999-2, Dept. of Computer Science, University of Tampere, 1999.

[Auramo and Juhola, 1995] Yrjö Auramo and Martti Juhola, Comparison of inference results of two otoneurological expert systems. *International Journal of Bio-Medical Computing* **39** (1995), 327–335.

[Auramo and Juhola, 1996] Yrjö Auramo and Martti Juhola, Modifying an expert system construction to pattern recognition solution. *Artificial Intelligence in Medicine* **8**, 1 (1996), 15–21.

[Auramo et al., 1993] Yrjö Auramo, Martti Juhola and Ilmari Pyykkö, An expert system for the computer-aided diagnosis of dizziness and vertigo. *Med. Inform.* **18**, 4 (1993), 293–305.

[Callan, 2003] Rob Callan, *Artificial Intelligence.* Palgrave MacMillan, 2003.

[Chae, 1998] Young Moon Chae, Expert system in medicine. In: Jay Liebowitz (ed.), *The Handbook of Applied Expert Systems.* CRC Press LLC, Boca Raton, 1998, 32-1–32-50.

[Chen and Rada, 1998] Chaomei Chen and Roy Rada, Expert system technology: expert system interface. In: Jay Liebowitz (ed.), *The Handbook of Applied Expert Systems.* CRC Press LLC, Boca Raton, 1998, 6-1–6-12.

[Juhola et al., 1995] Martti Juhola, Yrjö Auramo, Erna Kentala and Ilmari Pyykkö, An essay on power of expert systems versus human expertise. *Med. Inform.* **20**, 2 (1995), 133-138.

[Kentala et al., 1995] Erna Kentala, Ilmari Pyykkö, Yrjö Auramo and Martti Juhola, Database for vertigo. *Otolaryngology – Head and Neck Surgery* **112** (1995), 383–390.

[Kentala et al., 1998] Erna Kentala, Yrjö Auramo, Martti Juhola and Ilmari Pyykkö, Comparison between diagnoses of human experts and a neurotologic expert system. *Ann. Otol. Rhinol. Laryngol.* **107**, 2 (1998), 135–140.

[Metaxiotis and Samouilidis, 2000] K.S. Metaxiotis and J.-E. Samouilidis, Expert systems in medicine: academic illusion or real power? *Information Management & Computer Security* **8**, 2 (2000), 75–79.

[Mitchell, 1997] Tom M. Mitchell, *Machine Learning.* McGraw-Hill, New York, 1997.

[Viikki, 2002] Kati Viikki, *Machine Learning on Otoneurological Data: Decision Trees for Vertigo Diseases.* Ph.D. Thesis. Report A-2002-8, Dept. of Computer and Information Sciences, University of Tampere, 2002. Also available as http://acta.uta.fi/pdf/951-44-5390-5.pdf 17.10.2003.

[Viikki and Juhola, 2001] Kati Viikki and Martti Juhola, Refining the knowledge base of an otoneurological expert system. In: J. Crespo , V.

Maojo and F. Marting (eds.), *Medical Data Analysis*, *Lecture Notes in Computer Science* **2199**. Springer, Berlin, 2001, 276–281.

[Waterman, 1986] Donald A. Waterman, *A Guide to Expert Systems*. Addison-Wesley, Reading, 1986.