Assosiaatiosääntöjen johtaminen epäyhtenäisellä minimituella

Tina Willner

19.12.2003

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

TIIVISTELMÄ

Tietämyksen etsimisellä tarkoitetaan monivaiheista prosessia, jolla etsitään tietoa tietokannoista. Tietämyksen etsimisen yksi vaihe on tiedonrikastus, jonka eräs tekniikka on assosiaatiosääntöjen johtaminen. Assosiaatiosäännöillä saadaan tietokannasta esille mielenkiintoisia mielleyhtymiä.

Assosiaatiosääntöjä etsivissä algoritmeissa käytetään usein esiintyvien alkiojoukkojen etsimisessä minimitukea, joka voi olla yhtenäinen tai epäyhtenäinen. Yhtenäistä minimitukea käytetään esimerkiksi Apriori-algoritmissa, johon useat assosiaatiosääntöjen johtamiseen kehitetyt algoritmit perustuvat. Tässä tutkielmassa perehdymme kirjallisuuden perusteella etenkin epäyhtenäisen minimituen käyttöön. Epäyhtenäistä minimitukea käyttämällä voidaan ratkaista harvinaisten alkioiden ongelma sekä saadaan assosiaatiosääntöihin mukaan myös merkittävää, harvinaista dataa.

Avainsanat: Apriori-algoritmi, assosiaatiosääntö, epäyhtenäinen minimituki, harvinaisten alkioiden ongelma

SISÄLTÖ

1. JOHDANTO

Nykyään kerätään valtavasti tietoja erilaisiin tiedostoihin ja tietokantoihin. Tallennettuja tietoja tulisi käyttää myös hyväksi, jotta tallentaminen ei olisi turhaa työtä. Tietokantoihin tallentuu suuri määrä erilaista tärkeätä tietoa, joka pitää vain etsiä sieltä. Tätä prosessia kutsutaan tietämyksen etsimiseksi. Tietämyksen etsimisen yksi vaihe on tiedonrikastus, johon sisältyy erilaisia tekniikoita. Yksi näistä tekniikoista on assosiaatiosääntöjen johtaminen, johon perehdymme tässä työssä. Assosiaatiosäännöt ovat tietokantaan tallentuneita tärkeitä tietoja. Tietokantaan tallentuneilla tiedoilla on erilaisia suhteita ja yhteyksiä toisiinsa. Assosiaatiosäännöt paljastavat näitä yhteyksiä.

Assosiaatiosääntöjen johtamiseen on kehitetty useita erilaisia algoritmeja. Yilmaz & al. (2003) nostavat esiin viisi algoritmia, joilla on olennainen osa assosiaatiosääntöjen etsimisen historiassa. Ensimmäinen lähtökohta assosiaatiosääntöjen etsimiseen oli Agrawalin, Imielinskin ja Swamin vuonna 1993 kehittämä AIS-algoritmi. Heidän kehittämänsä algoritmi oli suunniteltu käyttämään hyväksi asiakkaiden ostoskoreista tallennettuja tietoja (Agrawal & al., 1993). Löydettyjä assosiaatiosääntöjä sovellettiin käytäntöön, kun analysoitiin asiakkaiden kulutustottumuksia. Jo tässä algoritmissa käytettiin minimituki-parametria, johon paneudumme tarkasti tässä työssä. AIS-algoritmilla löydettiin kaikki assosiaatiosäännöt, mutta ne olivat kuitenkin hyvin suppeita (Agrawal & Srikant, 1994).

Houtsma ja Swami (1995) kehittivät oman ratkaisunsa assosiaatiosääntöjen johtamiseen esittämällä, että tiedonrikastusta voidaan tehdä käyttämällä SQL-kyselykieltä. He kehittivät relaatiotietokantoja varten SETM-algoritmin, jossa käytetään relaatio-operaatioita (Savasere & al., 1995). SETM on joukkopohjainen algoritmi (set-oriented algorithm) ja muunnelma AIS:stä (Toivonen, 1996). Myös SETM-algoritmissa käytettiin AIS-algoritmin tapaan minimituki-parametria.

Agrawal ja Srikant (1994) esittelivät kolme uutta algoritmia, jotka saivat paljon tunnustusta. Nämä algoritmit olivat Apriori, AprioriTid ja AprioriHybrid. Näillä algoritmeilla pystytään etsimään hyvin laajoja assosiaatiosääntöjä, toisin kuin vuonna 1993 kehitetyllä AIS-algoritmilla. AprioriHybrid on yhdistelmä Apriorista ja AprioriTidis-

tä. Suorituskykyä testaavat asiantuntijat ovat havainneet useilla eri tietojoukoilla, että Apriori-algoritmi on nopeampi ensimmäisissä läpikäynneissä, mutta hitaampi myöhemmissä läpikäynneissä verrattuna AprioriTid-algoritmiin (Rantzau, 1997). Park & al. (1995) huomauttavat, että AprioriHybrid-algoritmi voi vaihtaa suorituksensa Apriori-algoritmista AprioriTid-algoritmiin ensimmäisten läpikäyntien jälkeen. Yilmaz & al. (2003) korostavat, että Apriori-algoritmi on ydin tämän päivän assosiaatiosääntöjä etsiville algoritmeille. Apriori-algoritmissa käytetään aikaisempien algoritmien tapaan minimituki-parametria, joka on *yhtenäinen* (uniform) kaikille johdetuille säännöille, kuten aiemmissakin algoritmeissa.

Assosiaatiosääntöjen johtamisessa merkittävä tutkiskelu kohdistuu Yilmazin & al. (2003) mukaan *osiointiin* (partition). Savaseren & al. (1995) pitivät edellä mainittujen algoritmien ongelmana sitä, että ne vaativat useita tietokannan läpikäyntejä. Osioinnissa löydetään assosiaatiosääntöihin johtavat alkiojoukot ainoastaan kahdella tietokannan läpikäynnillä (Yang & al., 2001). Yilmaz & al. (2003) mainitsevat huomattavana tutkimuksen kohteena myös *otannan* (sampling). Toivonen (1996) on kehittänyt algoritmeja, joissa tietokannasta haetaan satunnaisesti otantoja, joiden avulla löydetään assosiaatiosääntöjä. Algoritmit käyvät tietokannan läpi yhdesti, jolloin haetaan satunnainen otanta. Otannan avulla määritellään assosiaatiosäännöt, joiden ajatellaan olevan voimassa koko tietokannassa. Tulokset tarkistetaan jäljelle jääneestä tietokannasta. Tällä menetelmällä menetetään virheettömyyttä, mutta voitetaan tehokkuudessa (Han & Kamber, 2001). Sekä osioinnin että otannan tarkoitus on parantaa siis algoritmien suorituskykyä suuria tietojoukkoja käsiteltäessä.

Edellä mainituilla algoritmeilla ja tekniikoilla ei kuitenkaan saada selville kaikkia mahdollisia assosiaatiosääntöjä. Viime aikoina assosiaatiosääntöjen johtamisessa onkin tutkittu, miten assosiaatiosääntöihin saadaan mukaan myös tietokannassa harvoin esiintyviä alkioita. Wang & al. (2003) toteavat, että kaikilla tietokannan alkioilla ei ole samanlainen esiintymistodennäköisyys tietokannassa. Harvoin esiintyvä data on kuitenkin merkittävää ja näitä alkioita tulisi saada mukaan assosiaatiosääntöihin. Ratkaisuksi on osoittautunut assosiaatiosääntöjä etsivien algoritmien minimitukiparametrin käyttö siten, että se on *epäyhtenäinen* (non-uniform) algoritmin läpikäynnin eri vaiheissa.

Tämän tutkielman tarkoituksena on tutustua assosiaatiosääntöihin sekä assosiaatiosääntöjä etsivien algoritmien minimituki-parametrin käyttöön. Käyttämällä epäyhtenäistä minimituki-parametria algoritmeissa, saadaan assosiaatiosääntöihin mukaan myös harvoin esiintyviä alkioita. Pääpaino työssäni onkin tarkastella epäyhtenäisen minimituki-parametrin käyttöä ottamatta kuitenkaan erityisesti kantaa eri algoritmien suorituskykyyn. Tutkielmassa esiintyvissä algoritmeissa sanasto ja muuttujat on pidetty englanninkielisenä, jotta yhteys kirjallisuuteen säilyisi paremmin. Samoin luettavuuden kannalta algoritmit on pyritty kuvaamaan yhtenäisessä muodossa muuttamatta kuitenkaan niiden alkuperäistä rakennetta. Seuraavassa luvussa perehdymme assosiaatiosääntöjen peruskäsitteisiin ja luokitteluun. Kolmannessa luvussa käsittelemme tapoja, joilla minimituki-parametria voidaan käyttää algoritmeissa, jotta löydetään harvoin esiintyviä alkioita assosiaatiosääntöihin. Lopuksi teemme yhteenvedon tutkielmassa käsitellyistä asioista.

2. ASSOSIAATIOSÄÄNTÖJEN LUOKITTELU JA ARVIOINTI

Nykyään kerätään valtavasti tietoa tiedostoihin ja tietokantoihin. Tallennettuja tietoja tulisi käyttää myös hyväksi, jotta aikaa vievä tallentaminen ei menisi hukkaan. Tietokantoihin tallentuu suuri määrä erilaista tärkeätä tietoa, joka pitää vain etsiä sieltä. Tätä prosessia kutsutaan tietämyksen etsimiseksi, jonka periaatteet käymme läpi tämän luvun ensimmäisessä kohdassa. Tämän jälkeen perehdymme tarkemmin tiedonrikastukseen, joka on tietämyksen etsimisen osa-alue. Assosiaatiosäännöt ovat yksi tiedonrikastuksen tekniikoista, jonka peruskäsitteet käsittelemme luvun toisessa kohdassa. Kolmannessa kohdassa käymme läpi assosiaatiosääntöjen käytännön sovelluksia. Assosiaatiosääntöjä on olemassa erilaisia liittyen niiden monimutkaisuuteen ja käyttötarkoitukseen. Luvun neljännessä kohdassa tutustumme assosiaatiosääntöihin vaikuttaviin tekijöihin. Luvun lopussa tarkastelemme, miten löydettyjä assosiaatiosääntöjä voidaan arvioida.

2.1 Tietämyksen etsiminen

Tietämyksen etsimisellä (knowledge discovery in databases) tarkoitetaan prosessia, jolla etsitään tietoa tietokannoista. Kandel & al. (2001) määrittelevät tietämyksen etsimisen siten, että se on monimutkainen prosessi, jolla tunnistetaan kelvollisia, uusia, käyttökelpoisia ja ymmärrettäviä malleja datasta.

Elmasri ja Navathe (2000) jakavat tietämyksen etsimisen kuuteen vaiheeseen. Nämä vaiheet ovat tiedon valinta, tiedon puhdistus, tiedon integrointi[1], tiedon muuntaminen, tiedonrikastus ja tulosten ilmoittaminen. Han ja Kamber (2001) lisäävät omassa tietämyksen etsimisen jaottelussaan mallien arviointivaiheen, ennen tulosten ilmoittamista. Elmasri ja Navathe (2000) sisällyttävät sen sijaan mallien arvioinnin tiedonrikastusvaiheeseen.

[1] Elmasri ja Navathe (2000) käyttävät jaottelussaan termiä data enrichment. Tällä he tarkoittavat kuitenkin tiedon integrointia, joka on Hanin ja Kamberin (2001) käyttämä termi.

Tietämyksen etsimisen ensimmäisessä vaiheessa, *tiedon valinnassa* (data selection), valitaan kohde, jota tutkitaan. Tutkittava kohde valitaan halutulta osa-alueelta, esimerkiksi kaupan alalta. Kauppaketjun tietokannoista voidaan edelleen poimia tarkemmin tutkittava kohde, kuten tietyn alueen kaupan, tietyn ikäisten asiakkaiden kulutustottumukset. Ikähaarukaksi voidaan ottaa asiakkaat kolmenkymmenen ja kolmenkymmenenkuuden ikävuoden väliltä ja alueeksi valita Keski-Pohjanmaa. Kun on valittu tutkittava kohde, siirrytään seuraavassa vaiheessa *tiedon puhdistukseen* (data cleansing). Tiedon puhdistuksessa korjataan virheellisiä tietoja tai jätetään pois epätäydellisiä monikkoja. Virheellisiä tietoja voidaan löytää suunnattomasti, kun tietokannat ovat tarpeeksi suuria. Tiedon puhdistuksessa voidaan esimerkiksi muuntaa tietyllä funktiolla kaikki tallennetut kahdeksannumeroiset syntymäajat yksi- tai kaksinumeroiseksi iäksi. Esimerkiksi syntymäaikaa 01011978 vastaava ikä vuonna 2003 olisi 25. Mikäli ikä on kuitenkin tallennettu eksplisiittisesti, se voidaan tallennuspäivämäärän perusteella päivittää funktion avulla. Jos ikää ei ole tallennettu tietokantaan lainkaan, koko monikko voidaan jättää pois. Tiedon puhdistaminen on tärkeä vaihe, sillä sen johdosta tietämyksen etsimisen seuraavat vaiheet onnistuvat helpommin.

Tietämyksen etsimisen kolmannessa vaiheessa, *tiedon integroinnissa* (data integration), parannetaan ja lisätään tietoa muista lähteistä tulleilla tiedoilla. Muita lähteitä voivat olla erilliset tietokannat, joiden tiedot liittyvät alkuperäisen tietokannan tietoihin. Esimerkiksi kanta-asiakastietokannan tiedot liittyvät kaupan tietokannan tietoihin. Lisäyskohteeksi voidaan ottaa kanta-asiakastietokannasta asiakkaiden kuukausitulot, jotka liitetään aikaisempaan tietueeseen. Tiedot voidaan yhdistää näistä kahdesta tietokannasta kokonaisuudeksi asiakasnumeron avulla, joka esiintyy molemmissa tietokannoissa. Kuvassa 2.1 on asiakastaulut kaupan tietokannasta sekä kanta-asiakastietokannasta, joista saadaan kattavat tiedot SQL-kyselykielellä käyttämällä asiakasnumeroa perusavaimena. Perusavain yhdistää näiden taulujen tiedot toisiinsa.

Kahdesta taulusta tulleet tiedot eivät välttämättä kuitenkaan ole yhteneväiset toistensa kanssa esimerkiksi kirjoitusvirheiden tai osoitteenmuutosten takia. Jin & al. (2002) ovat kehittäneet StringMap-algoritmin, joka on menetelmä tekstitiedon yhdistämiselle. Algoritmilla voidaan korjata kahden taulun eroavaisuuksia, kuten kirjoitusvirheitä.

KAUPAN_ASIAKASTAULU			
ASIAKASNUMERO	**SUKUNIMI**	**ETUNIMI**	**SYNTYMA_AIKA**
100	WILLNER	TINA	051278

KANTA_ASIAKASTAULU			
ASIAKASNUMERO	**SUKUNIMI**	**ETUNIMI**	**KUUKAUSITULOT**
100	WILLNER	TINA	100

SQL-KYSELYN PERIAATE
SELECT K.SUKUNIMI, K.ETUNIMI, K.SYNTYMA_AIKA, P.KUUKAUSITULOT **FROM** KAUPAN_ASIAKASTAULU K, KANTA_ASIAKASTAULU P **WHERE** K.ASIAKASNUMERO = P.ASIAKASNUMERO

Kuva 2.1 Asiakastietojen integrointi.

Toisessa taulussa voi olla esimerkiksi etunimen kohdalla pelkkä "T." ja toisessa se voi olla "TINA", algoritmin avulla saadaan yhdistettyä oikeat tiedot.

Tietämyksen etsimisen neljäs vaihe on *tiedon muuntaminen* (data transformation). Tiedon muuntamista voidaan tehdä tietomäärän vähentämiseksi ja yksinkertaistamiseksi. Kun kohteeksi on valittu 30-36 vuoden ikäiset asiakkaat, voidaan nämä asiakkaat ryhmitellä esimerkiksi kahden tai kolmen vuoden ikähaarukoihin. Tällaisen esityön jälkeen päästään tietämyksen etsimisessä viidenteen vaiheeseen, joka on *tiedonrikastus* (data mining). Tiedonrikastukseksi sanotaan uuden tiedon löytämistä tietokannoista. Tiedonrikastuksella löydetään uusia malleja ja sääntöjä suuresta määrästä tietoa (Wur & Leu, 1999). Hand & al. (2001) määrittelevät tiedonrikastuksen siten, että se on analyysiä, jolla löydetään ennalta arvaamattomia yhteyksiä ja tämä tieto kootaan yhteen siten, että se on ymmärrettävää sekä hyödyllistä datan omistajalle.

Elmasri ja Navathe (2000) huomauttavat, että tietokannan hallintajärjestelmät eivät nykyään ole hyvin yhdistetty tiedonrikastukseen. Tietokantojen hallintaan ja tietojen etsimiseen on perinteisesti käytetty SQL-kyselykieltä. Sitä käyttääkseen pitää kuitenkin tietää tietokannan rakenne. Tiedonrikastus pitäisikin ottaa huomioon jo siinä vaiheessa, kun tallennustapaa ja tietokannan rakennetta suunnitellaan. Hyvin rakennetuista tietokannoista on helppo hakea tietoa, eikä tällöin tarvitse tehdä ylimääräistä

työtä tietokantojen muuttamisessa käytettävämmiksi. Hyvin rakennetuissa tietokannoissa tiedon puhdistus ja tiedon muuntaminen onnistuvat helposti tai niitä ei tarvitse tehdä lainkaan.

Tiedonrikastustekniikoita ovat assosiaatiosääntöjen johtaminen, luokittelu, regressio ja ryhmittelyanalyysi (Kantardzic, 2003; Han & Kamber, 2001). Tämä työ kertoo yhdestä tiedonrikastuksen tekniikasta, assosiaatiosääntöjen johtamisesta. Han ja Kamber (2001) mainitsevat tiedonrikastuksen jälkeen tietämyksen etsimisen yhtenä vaiheena *mallien arvioinnin* (pattern evaluation), jota Elmasri ja Navathe (2000) pitävät tiedonrikastuksen osana. Mallien arvioinnissa etsitään mielenkiintoisia assosiaatiosääntöjä ja jätetään huomioimatta säännöt, jotka eivät ole mielenkiintoisia. Seulonta tapahtuu mittareilla, joita tarkastelemme kohdassa 2.5.

Tietämyksen etsimisen viimeinen vaihe on *tulosten ilmoittaminen* (data reporting). Tulokset voidaan esittää eri ulkoasuissa riippuen siitä kenelle tulokset on luotu ja mihin käyttötarkoitukseen. Ulkoasuja ovat erilaiset listat, selventävät graafiset esitykset tai yhteenvetotaulut (Elmasri & Navathe, 2000).

2.2 Peruskäsitteet

Kohdassa 2.1 kävimme läpi tietämyksen etsimisen seitsemän vaihetta. Tässä kohdassa perehdymme assosiaatiosääntöjen johtamiseen, joka on tietämyksen etsimisen viidennen vaiheen eli tiedonrikastuksen tekniikka.

Tietokantaan tallennetuilla tiedoilla on erilaisia suhteita ja yhteyksiä toisiinsa. Tietokannan taulut ovat yhteydessä toisiinsa, mutta myös eri tietokannoilla on usein suhteita toisiinsa. Esimerkiksi relaatiotietokannassa tietokantataulut liittyvät toisiinsa perus- ja viiteavaimilla. Tietokantataulujen tiedot ja erillisten tietokantojen tiedot saadaan yhdistettyä toisiinsa näillä avaimilla. Esimerkiksi kuvassa 2.1 erillisten tietokantojen tiedot yhdistetään asiakasnumeron avulla. Tietokantaan tallennetuilla tiedoilla voi olla myös toisenlaisia assosiaatioita eli mielleyhtymiä.

Assosiaatioita voidaan löytää esimerkiksi kaupan *tapahtumaperäisestä tietokannasta* (transactional database), johon on tallennettu jokaisen asiakkaan ostoskorin sisältö yhtenä tapahtumana viivakoodien avulla. Jokaisesta ostoskerrasta tallentuu uusi tapahtuma tietokantaan. Kaupan tapahtumaperäisestä tietokannasta voidaan etsiä usein yhdessä ostettuja tuotteita, jolloin näillä löydetyillä tuotteilla on assosiaatio toisiinsa. Park & al. (1997) huomauttavat, että dataa on kerättävä ainakin kolmenkymmenen päivän ajalta, jotta voidaan tehdä mielekkäitä johtopäätöksiä. Löydetty assosiaatio voi olla esimerkiksi ketsupin ja pastan välillä. Ketsuppi ja pasta muodostavat siten 2-*alkiojoukon* (2-itemset), {ketsuppi, pasta}. Tällaisia *usein esiintyviä alkiojoukkoja* (frequent itemset) etsitään tietokannasta ja tuloksena saadaan *assosiaatiosääntöjä* (association rule). Assosiaatiosääntö voi olla muodossa ketsuppi $\Rightarrow$ pasta, jolloin nuolen vasemmalla ja oikealla puolella olevilla alkioilla tiedetään olevan mielleyhtymä toisiinsa. Edellisestä assosiaatiosäännöstä voidaan todeta, että asiakkaat, jotka ostavat ketsuppia, ostavat samalla myös pastaa. Kun tutkitaan, mitä tuotteita asiakkaat keräävät ostoskoriinsa käydessään kaupassa, puhutaan *ostoskorianalyysistä*. Ostoskorianalyysi on varhaisin ja alkeellisin sovellus assosiaatiosäännöistä (Han & Kamber, 2001).

Asiakkaiden ostostietoja voidaan kerätä tapahtumaperäiseen tietokantaan D (kuva 2.2). Han ja Kamber (2001) esittävät assosiaatiosääntöjen muodostamisen siten, että kaupassa olevat tuotteet muodostavat ensin alkiojoukon $I = \{i_1, i_{2,}..., i_m\}$. Tietokannan jokainen rivi muodostaa ostoskerran eli tapahtuman T, jolloin T muodostaa osajoukon *I*:stä, $T \subseteq I$. Jokainen tapahtuma yksilöidään omalla tunnuksella TID. Olkoon X ja Y alkiojoukkoja. Tapahtuman T sanotaan sisältävän alkiojoukon X, jos ja vain jos X on osajoukko T:stä eli $X \subseteq T$. Assosiaatiosääntö on implikaatio $X \Rightarrow Y$, jossa X on *edeltäjä* (antecedent) ja Y on *seurauksena oleva* (consequent) alkiojoukko (Rantzau, 1997). Assosiaatiosäännössä $X \Rightarrow Y$ alkiojoukot X ja Y ovat aitoja osajoukkoja *I*:stä, $X \subset I$, $Y \subset I$. Näiden kahden alkiojoukon X ja Y leikkaus muodostaa tyhjän joukon $X \cap Y = \phi$ (Han & Kamber, 2001).

Kuvan 2.2 tapahtumaperäinen tietokanta sisältää kolme tapahtumaa eli kolme riviä. Ensimmäinen tapahtuma on yksilöity numerolla 1, TID = 1. Tapahtuma sisältää kolme alkiota $\{i_1, i_{10}, i_{20}\}$ joukosta *I*, jolloin se muodostaa 3-alkiojoukon ja aidon osa-

joukon I:stä. Assosiaatiosäännössä $i_{10} \Rightarrow i_{20}$, nuolen molemmilla puolilla on 1-alkiojoukot ja ne kuuluvat alkiojoukkoon I. Alkioiden i_{10} ja i_{20} leikkaus muodostaa tyhjän joukon. Alkio i_{10} voisi merkitä muroja ja alkio i_{20} voisi merkitä maitoa, jolloin assosiaatiosääntö tulisi muotoon murot $\Rightarrow$ maito. Murojen ja maidon välillä on siis mielleyhtymä.

TID	ALKIOT
1	$\{i_1, i_{10}$(muroja)$, i_{20}$(maitoa)$\}$
2	$\{i_5, i_{15}, i_{20}$(maitoa)$\}$
3	$\{i_3, i_{10}$(muroja)$, i_{20}$(maitoa)$\}$

Kuva 2.2 Tapahtumaperäinen tietokanta D.

2.3 Assosiaatiosääntöjen soveltaminen

Kohdassa 2.2 käsittelemämme ostoskorianalyysi on yksinkertaisin assosiaatiosääntöjen johtamisen sovellus. Tässä kohdassa tarkastelemme laajemmin, mihin ostoskorianalyysiä ja assosiaatiosääntöjä voidaan käytännössä soveltaa.

Ostoskorianalyysissä tutkitaan, mitä tuotteita asiakkaat keräävät ostoskoriinsa käydessään ostoksilla. Tätä tietoa kauppiaat voivat käyttää monella tavalla hyödykseen. Ganti & al. (1999) ehdottavat, että kauppiaat voivat ostoskorianalyysin perusteella asetella assosioivat tuotteet, esimerkiksi murot ja maidon, hyllyihin joko lähekkäin tai sitten aivan eri puolille kauppaa. Jos tuotteet ovat hyllyssä vierekkäin, asiakkaat huomaavat ottaa ostoskoriinsa molemmat tuotteet. Kauppiaat asettavatkin usein yhdessä ostetut tuotteet, kuten suklaakastikkeen ja jäätelön, lähelle toisiaan. Mikäli kauppias päättää asettaa assosioivat tuotteet eri puolelle kauppaa, on suurempi todennäköisyys, että asiakas kerää ostoskoriinsa matkan varrelta muitakin tavaroita. Usein maitohylly onkin asetettu kaupan kauimmaiseen seinään sisääntulopaikasta katsottuna. Pelkkää maitoa kauppaan hakemaan tulleet asiakkaat joutuvat kiertämään ostoksiaan varten koko kaupan. Tällöin ostoskoriin saattaa tarttua monenlaisia heräteostoksia, joita asiakas ei etukäteen tiennyt tai muistanut tarvitsevansa.

Toinen sovellus ostoskorianalyysistä ja assosiaatiosäännöistä on Chiun & al. (2002) mainitsema tuotteiden ristiinmarkkinointi. Ristiinmarkkinointia voidaan tehdä kahden tuotteen välillä laittamalla alennukseen toinen assosioivista tuotteista. Assosioivat tuotteet voivat olla esimerkiksi kahvi ja suodatinpussit, jolloin assosiaatiosääntö on kahvi $\Rightarrow$ suodatinpussit. Kauppias voi laittaa kahvin alennukseen, jolloin asiakkaat suurella todennäköisyydellä ostavat myös suodatinpusseja samasta kaupasta. Näiden tuotteiden asettaminen kaupassa lähekkäin kasvattaa edelleen todennäköisyyttä parempaan myyntiin. Assosioivia tuotteita mainostetaan usein mainoslehtisissä vierekkäin, jolloin toinen tuote saattaa olla tarjouksessa ja toinen tuote esiintyy normaalihintaisena. Näin ollen asiakas saattaa valita ostospaikakseen juuri tämän kaupan, koska kuvittelee saavansa halvalla molemmat tarvitsemansa tuotteet.

Assosiaatiosäännöistä on hyötyä myös tuotteita markkinoiville yrityksille. He voivat mahdollisesti laajentaa tarjontaansa, jos osoittautuu, että joku heidän tuotteistaan assosioi voimakkaasti jonkun toisen tuotteen kanssa. Tällainen assosiaatio voi olla esimerkiksi perunalastujen ja dippikastikkeiden välillä. Yritys hyötyy tilanteesta, kun se voi tarjota molemmat assosioivat tuotteet asiakkailleen.

Assosiaatiosäännöillä voidaan tutkia myös asiakkaiden kulutustottumuksia. Säännöissä pystytään yhdistelemään asiakkaan eri tietoja. Esimerkiksi iän, osoitteen ja tulojen perusteella voidaan markkinoida tuotteita tietyille kohderyhmille. Yhteys asiakkaan ja tuotteen välillä syntyy esimerkiksi kanta-asiakaskortin välityksellä, kun asiakasnumeron ja tuotenumeron tiedot yhdistetään. Suoramarkkinoinnissa voidaan etsiä esimerkiksi hyvätuloiset keski-ikäiset naiset, jotka ovat potentiaalisia ostajia esimerkiksi kylpylälomapaketille. Witten ja Frank (2000) kehottavat kuitenkin varovaisuuteen, sillä tämänkaltainen erottelu esimerkiksi sukupuolen perusteella on eettisesti arveluttavaa ja joillakin alueilla myös laitonta. Suoramarkkinoinnilla säästetään rahaa, sillä mainokset tai kirjeet lähetetään vain osalle alkuperäisestä kohderyhmästä.

Cooley & al. (1999) mainitsevat että assosiaatiosääntöjä käytetään myös *web-käytön rikastuksessa* (Web Usage Mining). Tällä tekniikalla tutkitaan ihmisten käyttäytymistä Internetissä. Tuloksena voidaan saada seuraavanlainen assosiaatiosääntö (Cooley & al., 1999): Lentopallo-sivut $\Rightarrow$ Käsipallo-sivut. Edellä mainittu sääntö on löydetty

Olympialaisten 1996-websivuilta. Sääntö tulkitaan siten, että lentopallo-sivuilla vierailleet ovat samalla käyneet virallisilla käsipallo-sivuilla. Assosiaatiosääntö on saatu selville käyttämällä analyysissä hyväksi *sivuviittausfrekvenssiä* (page access frequency) sekä kerättyjä *lokitiedostoja* (log file). Löydettyjen sääntöjen avulla voidaan Internet-sivuja kehitellä käytettävämmiksi. Srivastavan & al. (2000) mukaan assosiaatiosääntöjä käytetään hyväksi myös silloin, kun halutaan ennakoida käyttäjän seuraavia toimenpiteitä. Esimerkiksi navigointia voidaan kehittää siten, että tietyn usein käytetyn polun seuraaminen on helppoa.

2.4 Assosiaatiosääntöihin vaikuttavat tekijät

Kohdassa 2.3 kävimme läpi assosiaatiosääntöjen soveltamista käytäntöön. Käytettävät assosiaatiosäännöt voivat olla yksinkertaisia tai sitten hyvin monimutkaisia. Tässä kohdassa tarkastelemme, minkälaiset tekijät vaikuttavat assosiaatiosääntöihin. Han ja Kamber (2001) jakavat assosiaatiosäännöissä olevan tiedon olemuksen kolmeen eri luokkaan. Näihin luokkiin kuuluvat tiedon tyyppi, tiedon ulottuvuus ja tiedon hierarkia. Näitä käsittelemme kolmessa ensimmäisessä alakohdassa. Nykyisen tutkimuksen perusteella tiedon olemuksen jaotteluun voidaan lisätä kohta tiedon harvinaisuudesta. Tiedon harvinaisuutta tarkastelemme neljännessä alakohdassa.

2.4.1 Tiedon tyyppi

Assosiaatiosääntöjä muodostavat alkiot koostuvat eri tietotyypeistä. Yksinkertaisin tyyppi on boolen-tyyppi, jota käytetään *boolen-assosiaatiosäännöissä*. Esimerkiksi ostoskorianalyysillä saadaan esille boolen-assosiaatiosääntöjä. Kaupan tuotteet muodostavat alkiojoukon ja ostoskorin sisältö muodostuu tietyistä alkioista. Kaikki tuotteet voidaan käsittää boolen-vektorina, jolloin ostoskorin sisällön mukaan tuotteet saavat arvoksi joko "paikalla" tai "poissa". Assosiaatiosääntöjä löydetään usein esiintyvien alkiojoukkojen perusteella. Säännössä (2.1) on löydetty assosiaatio dippikastikkeen ja kermaviilin välillä. Asiakkaat jotka ostavat dippikastiketta ostavat usein myös kermaviiliä. Säännössä on lisäksi tuki- ja uskottavuus-arvot. Näitä mittareita

käsittelemme alakohdassa 2.5.1. Boolen-assosiaatiosääntö on yksiulotteinen ja yksitasoinen assosiaatiosääntö.

dippikastike $\Rightarrow$ kermaviili tuki 5 % uskottavuus 75 % (2.1)

Toinen assosiaatiosäännöissä käytetty tiedon tyyppi on *kvantitatiivinen tietotyyppi*. Assosiaatiosäännöissä käytetään kvantitatiivisia arvoja kuvaamaan alkioiden numeerista tietoa. Määrälliset arvot voidaan jaotella eri väleihin, jotka erottelevat säännöt toisistaan. Arvojen jakaminen tiettyihin väleihin helpottaa myös sääntöjen tulkintaa. Kvantitatiivisilla arvoilla voidaan määritellä esimerkiksi ikää, kuten säännössä (2.2). Säännön mukaan 15-17 vuoden ikäiset miehet ostavat kevytmoottoripyöriä. Kvantitatiiviset arvot jaotellaan tilanteeseen sopiviin väleihin, esimerkiksi ikä voidaan ilmoittaa kolmen vuoden välein. Arvojen jaottelu voi tapahtua joko ennalta määrättyihin väleihin tai jaottelu suoritetaan dynaamisesti. Sääntö (2.2) käsittää myös assosiaatiosäännöissä käytetyn *kategorisen tietotyypin*, "sukupuolen". Kategoriset arvot ovat ennalta määrättyjä ja niitä on äärellinen määrä. Ne ovat samanarvoisia, eivätkä sen vuoksi ole missään järjestyksessä.

ikä(X,"15…17") $\wedge$ sukupuoli(X,"mies") $\Rightarrow$ ostaa(X,"kevytmoottoripyörä") (2.2)

2.4.2 Tiedon ulottuvuudet

Assosiaatiosäännössä voi olla useampi tutkittava ulottuvuus. Yksinkertaisimmillaan säännössä tutkitaan kuitenkin vain yhtä ulottuvuutta, jolloin sitä sanotaan *yksiulotteiseksi assosiaatiosäännöksi*. Säännössä (2.3) tutkittava ulottuvuus on ostaminen. Tällöin ollaan kiinnostuneita vain siitä, mitä tuotteita asiakas ostaa.

ostaa(dippikastike) $\Rightarrow$ ostaa(kermaviili) (2.3)

Moniulotteisessa assosiaatiosäännössä tutkitaan useampaa ulottuvuutta samanaikaisesti. Moniulotteisissa assosiaatiosäännöissä arvoja yhdistellään tiedon eri ulottuvuuksilta, jolloin säännöt ovat informatiivisempia käyttäjälle (Pinto, 2001). Sääntö (2.2) on esimerkki kolmiulotteisesta assosiaatiosäännöstä. Tutkittavat ulottuvuudet

ovat ikä, sukupuoli ja ostaminen. Sääntö (2.4) on esimerkki neliulotteisesta assosiaatiosäännöstä, jossa esiintyvät ulottuvuudet ikä, siviilisääty, sukupuoli ja ostaminen. Säännön mukaan 20-25 vuoden ikäiset naimattomat miehet ostavat moottoripyöriä. Moniulotteisia assosiaatiosääntöjä saadaan esimerkiksi relaatiotietokannasta, joka on rakenteeltaan moniulotteinen. Verrattuna tapahtumaperäiseen tietokantaan, moniulotteisista tietokannoista saadaan esille esimerkiksi ostostilanteeseen liittyviä attribuutteja, kuten ostajan ikä.

$$\text{ikä(X,"20...25")} \wedge \text{siviilisääty(X,"naimaton")} \wedge \text{sukupuoli(X,"mies")} \Rightarrow \text{ostaa(X,"moottoripyörä")} \quad (2.4)$$

Au ja Chan (2003) kehittivät *FARM II*-tekniikan (Fuzzy Association Rule Mining) löytääkseen moniulotteisia assosiaatiosääntöjä pankin tietokannasta. Pankin työntekijät olivat kiinnostuneita assosiaatiosäännöistä, jotta he voisivat palvella asiakkaitaan tehokkaammin ja pitää asiakkaat tyytyväisinä. Pankin tietokanta sisälsi tapahtumadataa kuuden kuukauden ajalta yli 32000 asiakkaasta. Tietokannasta löytyi muun muassa neliulotteinen assosiaatiosääntö (2.5). Assosiaatiosääntö kertoo 35-45 vuotiaista naimattomista asiakkaista, joiden tilin saldo on 1000$ ja 2500$ välillä. Nämä asiakkaat hakevat FARM II:sen mukaan todennäköisesti lainaa, jonka suuruus on 10000$ ja 15000$ välillä. Säännössä kategorinen tietotyyppi on siviilisääty ja kvantitatiiviset tietotyypit ovat ikä, tilisaldo ja laina.

$$\text{siviilisääty(X,"naimaton")} \wedge \text{ikä(X,"35...45")} \wedge \text{tilisaldo(X,"1000...2500")} \Rightarrow \text{laina(X,"10000...15000")} \quad (2.5)$$

Sääntöjä (2.4) ja (2.5) voidaan sanoa myös *profiili-assosiaatiosäännöiksi* (profile association rule). Profiili-assosiaatiosäännössä asiakkaan profiiliin liittyvät tiedot ovat assosiaatiosäännön implikaation vasemmalla puolella ja oikealla puolella näkyy asiakkaan oletettu käyttäytyminen (Dunham & al., 2000). Säännössä (2.5) asiakkaan profiili koostuu siviilisäädystä, iästä ja tilin saldosta. Asiakkaan käyttäytymiseksi oletetaan lainahakemus, joka on 10000$ ja 15000$ väliltä.

Han ja Kamber (2001) jakavat yksi- ja moniulotteiset assosiaatiosäännöt edelleen kolmeen luokkaan. *Ulottuvuuden sisäisissä assosiaatiosäännöissä* (intradimension association rule) käsitellään ainoastaan yhtä ulottuvuutta. Näin ollen kaikki yksiulot-

teiset assosiaatiosäännöt ovat ulottuvuuden sisäisiä assosiaatiosääntöjä. Koska ulottuvuuden sisäisissä assosiaatiosäännöissä käsitellään vain yhtä ulottuvuutta, se toistuu assosiaatiosäännön implikaation vasemmalla ja oikealla puolella, kuten säännössä (2.3). *Ulottuvuuksien välisissä assosiaatiosäännöissä* (interdimension association rule) sen sijaan käytetään jokaista ulottuvuutta ainoastaan yhdesti. Nämä säännöt ovat siis aina moniulotteisia assosiaatiosääntöjä. Sääntö (2.4) on ulottuvuuksien välinen assosiaatiosääntö, sillä siinä esiintyy neljä ulottuvuutta, ikä, siviilisääty, sukupuoli ja ostaa, joita jokaista käytetään vain yhdesti. Mikäli moniulotteisessa assosiaatiosäännössä ulottuvuus toistuu useamman kerran, sitä kutsutaan *ulottuvuuksiltaan hybridiksi assosiaatiosäännöksi* (hybrid-dimension association rule). Sääntö (2.6) on ulottuvuuksiltaan hybridi assosiaatiosääntö, sillä se on moniulotteinen ja ostaa-ulottuvuus esiintyy useammin kuin kerran. Säännön mukaan 20-25 vuoden ikäiset naimattomat miehet ostaessaan moottoripyörän ostavat myös moottoripyöräkypärän.

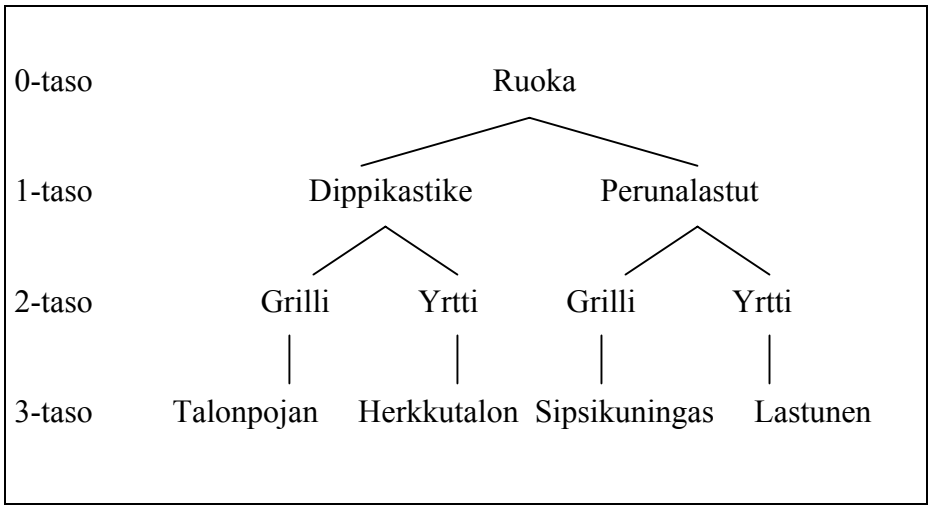
$$\text{ikä(X,"20...25")} \wedge \text{siviilisääty(X,"naimaton")} \wedge \text{sukupuoli(X,"mies")} \wedge \text{ostaa(X,"moottoripyörä")} \Rightarrow \text{ostaa(X,"moottoripyöräkypärä")} \quad (2.6)$$

2.4.3 Tiedon hierarkia

Assosiaatiosäännöissä voidaan ottaa huomioon alkioiden määrittelyjen tasot. Tämä edellyttää sitä, että alkiot on luokiteltu *monitasoiseen käsitehierarkiaan* (concept hierarchy). Han ja Kamber (2001) esittelevät käsitehierarkian puumaisena rakenteena. On sovittu, että käsitehierarkian tasot numeroidaan ylhäältä alaspäin. Käsitehierarkian ylimpään tasoon, nolla-tasoon, kuuluvat kaikki alkiot. Nolla-taso on yleisluontoisin taso. Kun tullaan hierarkiassa seuraavaan tasoon, 1-tasoon, alkiot jakautuvat moneen eri lokeroon. Jokainen aleneva taso on tarkempi kuin sitä edeltävä ja alin taso on kaikista tarkin. Alkiot ovat usein valmiiksi luokiteltu tietokantaan, jolloin ne voidaan helposti siirtää käsitehierarkian eri tasoille (Han & Fu, 1995).

Kuvassa 2.3 on esimerkki käsitehierarkiasta. Ruokatuotteet voidaan kuvata kategorian, ominaisuuksien ja tuotemerkin mukaan (Yen & Chen, 2001). Käsitehierarkian ylimmällä tasolla on kaikkien tuotteiden yhteinen nimitys, ruoka, ja tason numero on nolla. Seuraavalla tasolla, 1-tasolla, on eri ruokatuotteita, kuten dippikastike ja peru-

nalastut. Toisella tasolla on tarkennettu edellisen tason tuotteita ominaisuuksien mukaan uuteen tasoon. Esimerkiksi dippikastike ja perunalastut voivat olla grillimaustettuja tai yrttimaustettuja. Kuvan 2.3 tarkimmalla tasolla eli 3-tasolla on ruokien tuotemerkit.



Kuva 2.3 Käsitehierarkia.

Yksitasoisessa assosiaatiosäännössä ei alkioiden määrittelyjen tasoja kuitenkaan huomioida, sillä alkiot käsitetään yleisellä tasolla eli yksitasoisena. Yksitasoisten assosiaatiosääntöjen alkiot esiintyvät käsitehierarkiassa ensimmäisellä tasolla. Sääntö (2.7) on yksitasoinen ja sen mukaan dippikastiketta ostavat asiakkaat ostavat myös perunalastuja.

$$\text{ostaa(dippikastike)} \Rightarrow \text{ostaa(perunalastut)} \tag{2.7}$$

Joihinkin sovelluksiin tarvitaan tarkempia tietoja, kuin mitä yksitasoisista assosiaatiosäännöistä saadaan selville (Han & Fu, 1999). Srikant ja Agrawal (1995) toteavat, että käyttäjät ovat usein kiinnostuneita säännöistä, jotka koostuvat useista käsitehierarkian tasoista. Yksitasoisella assosiaatiosäännöllä saadaan esimerkiksi selville, että 75 % niistä ihmisistä, jotka ostavat dippikastiketta, ostavat myös perunalastuja samalla ostosmatkalla. *Monitasoisella assosiaatiosäännöllä* saadaan tarkempaa ja konkreettisempaa tietoa asiakkaiden ostoskäyttäytymisestä, sillä alkiot ovat eri hierarkiatasoilla. Esimerkiksi 75 % asiakkaista, jotka ostavat Talonpojan grillimaustettua dippikastiket-

ta, ostavat rinnalle Sipsikuninkaan grillimaustettuja perunalastuja. Monitasoinen assosiaatiosääntö, kuten sääntö (2.8), edellyttää siis dataa, joka on monitasoisessa käsitehierarkiassa. Säännössä (2.8) on käytetty käsitehierarkian kolmatta tasoa, jolloin saadaan selville tarkkaa tietoa kuluttajan ostoskäyttäytymisestä.

ostaa(Talonpojan grillimaustettu dippikastike) ⇒
ostaa(Sipsikuninkaan grillimaustetut perunalastut) (2.8)

2.4.4 Tiedon harvinaisuus

Yun & al. (2003) korostavat, että tietokannan muodostava data voi esiintyä siinä suhteellisen usein tai sitten ei, riippuen tietokannan ominaisuuksista. Wang & al. (2003) toteavat, että koska kaikilla tietokannan alkioilla ei ole samanlainen esiintymistodennäköisyys tietokannassa, ei näillä alkioilla myöskään ole samanlainen todennäköisyys yltää assosiaatiosääntöihin. Esiintymistiheydet vaihtelevat alkiosta riippuen. Esimerkiksi tarkastellessamme tavaratalon tuotteita huomamme, että maito ja leipä esiintyvät ostoskoreissa huomattavasti useammin kuin sinappi tai ketsuppi. Myös harvoin esiintyvät alkiot on syytä ottaa mukaan assosiaatiosääntöihin. Esimerkiksi assosiaatiosäännössä ketsuppi ⇒ pasta on mukana harvoin tietokannassa esiintyvä alkio, ketsuppi.

Cohen & al. (2001) toteavat, että merkittävä, mutta harvoin esiintyvä data tuo esille uusia näkemyksiä. Tiedon harvinaisuudella ei välttämättä ole vaikutusta säännön muotoon, kuten muilla tiedon olemukseen liittyvillä ominaisuuksilla. Tiedon harvinaisuus näkyy etenkin säännön esiintymisenä ja oikeamman informaation antamisena käyttäjälle.

2.5 Mallien arviointi

Kohdassa 2.4 tarkastelimme assosiaatiosääntöihin vaikuttavia tekijöitä. Assosiaatiosääntöjä etsittäessä voidaan algoritmeilla löytää lukuisia assosiaatiosääntöjä. Suuri osa näistä löydetyistä säännöistä ei kuitenkaan ole mielenkiintoisia. Tässä kohdassa

käymme läpi mallien arviointia, jota Elmasri ja Navathe (2000) pitävät tiedonrikastuksen osana, mutta jonka Han ja Kamber (2001) lukevat tietämyksen etsimisen erilliseksi vaiheeksi.

Sääntöjen joukossa voi olla itsestään selviä assosiaatioita tai sääntöjen alkiot eivät välttämättä esiinny niin usein aineistossa, jotta ne kannattaisi ottaa tarkasteluun. Mielenkiintoiset säännöt saattavat piiloutua näiden turhien sääntöjen sekaan ja olennaisten sääntöjen poimiminen on lopulta mahdotonta. Loppujen lopuksi voi käydä niin, että tietämyksen etsimisestä ei ole mitään konkreettista hyötyä. Tätä ongelmaa kutsutaan *mielenkiintoisuusongelmaksi* (interestingness problem) (Liu & al., 1999a). Tässä kohdassa tarkastelemme mittareita, joilla voidaan suodattaa ylimääräisiä sääntöjä pois. Mittareita tarvitaan, sillä assosiaatiosääntöjen etsimisessä haetaan nimenomaan mielenkiintoisia sääntöjä.

Liu & al. (2000) jakavat mittarit *objektiivisiin* ja *subjektiivisiin mittareihin*. Padmanabhan ja Tuzhilin (1999) pitävät molempia mittareita yhtä tärkeinä. Sahar (1999) korostaa, että vaikka objektiiviset mittarit ovat tärkeitä, subjektiivisia mittareita tarvitaan kuitenkin viime kädessä, jotta löydetään tietylle henkilölle mielenkiintoisia sääntöjä. Ensimmäisessä alakohdassa käsittelemme objektiivisia mittareita, jotka perustuvat käytettävien algoritmien rakenteeseen, sekä tilastotietoihin. Toisessa alakohdassa käymme läpi subjektiivisia mittareita, joihin liittyy käyttäjän subjektiivisuus assosiaatiosääntöjä tulkittaessa.

2.5.1 Objektiiviset mittarit

Objektiivisilla mittareilla on tietty kynnysarvo, jota käyttäjä kontrolloi. Han ja Kamber (2001) tarkastelevat seuraavia objektiivisia mittareita: yksinkertaisuus (simplicity), hyödyllisyys (utility) ja varmuus (certainty).

Yksinkertaisuus saadaan aikaan tekemällä assosiaatiosäännöistä yksinkertaisempia, rajoittamalla esimerkiksi säännön pituutta käyttäjän asettamalla kynnys-arvolla. Määräys yksinkertaisuudesta voidaan lisätä algoritmiin funktiona, jolloin se on osana algoritmin rakennetta objektiivisten mittareiden tapaan. Funktiolla voidaan pienentää

alkiojoukkoja säännön implikaation vasemmalta tai oikealta puolelta. Sääntöjen tulkinta on yksinkertaisempaa, kun alkioita on säännöissä vähemmän. On helpompaa tulkita ja käyttää hyväksi esimerkiksi sääntöä suklaakastike $\Rightarrow$ jäätelö kuin sääntöä suklaakastike $\Rightarrow$ {jäätelö, maito, peruna}. Assosiaatiosääntöjen lukumäärä kasvaa kuitenkin huomattavasti, kun sääntöjä yksinkertaistetaan alkiojoukkoja pienentämällä. Toinen keino tehdä säännöistä yksinkertaisempia on määrätä staattisesti kvantitatiivinen arvo, esimerkiksi ikä, viiden vuoden väleihin, kolmen vuoden välien sijaan. Kvantitatiivisen arvon ryhmittely suurempaan väliin vähentää samalla assosiaatiosääntöjen lukumäärää.

Hanin ja Kamberin (2001) esittämä toinen objektiivinen mittari on *hyödyllisyys*. Jotkut löydetyt assosiaatiosäännöt eivät ole tilastollisesti hyödyllisiä eivätkä mielenkiintoisia, sillä ne eivät ylitä käyttäjän vaatimaa *minimitukea* (minimum support). Minimituki-parametrilla saadaan selville miten monessa tapauksessa säännössä implikaation molemmilla puolilla olevat alkiojoukot esiintyvät käsiteltävässä tietokannassa. Minimituki-parametrin avulla jätetään etsimättä säännöt, jotka esiintyvät vain harvoin tietokannassa ja ovat sen takia hyödyttömiä. Kun esimerkiksi säännön (2.1) tuki on 5 %, silloin sekä dippikastiketta että kermaviiliä esiintyy viidessä prosentissa kaikista ostoskoreista. Minimituki-parametrin toteuttavat alkiojoukot ovat usein esiintyviä alkiojoukkoja (Agrawal & Srikant, 1995). Alkiojoukon tuki *sup* saadaan selville kaavalla (2.9) (Rantzau, 1997). Kaavassa tietokanta D koostuu tapahtumista $D = \{T_1, T_2, \ldots, T_m\}$. Alkiojoukko X on osajoukko tapahtumasta T, $X \subseteq T$. Kaavassa verrataan alkiojoukon X esiintymistiheyttä koko tietokantaan.

$$\sup(X) = \frac{|\{T \in D \mid X \subseteq T\}|}{|D|} \tag{2.9}$$

Kolmas objektiivisiin mittareihin liittyvä kohta on *varmuus*. Varmuudella testataan assosiaatiosääntöjen *uskottavuutta* (confidence). Assosiaatiosääntöjä johdettaessa voi käyttäjä määrätä tietyn uskottavuus-kynnyksen, jonka assosiaatiosääntöjen tulee toteuttaa. Uskottavuus-mittarilla tarkoitetaan sitä, kuinka monessa tapauksessa säännön implikaation vasemmalla puolella olevasta alkiojoukosta seuraa oikealla puolella oleva alkiojoukko. Uskottavuus-mittarin avulla saadaan suodatettua ne säännöt, jotka ei-

vät ole tilastollisesti merkittäviä. Kun uskottavuus on säännössä (2.1) 75 %, silloin 75 %:a dippikastiketta ostaneista asiakkaista, ostaa samalla myös kermaviiliä. Uskottavuus ilmoitetaan mieluummin prosentteina, kuin välillä 0...1 (Han & Kamber, 2001). Uskottavuus *conf* saadaan selville kaavalla (2.10) (Rantzau, 1997). Kaavassa X ja Y ovat alkiojoukkoja, jotka ovat aitoja osajoukkoja alkiojoukosta *I*, $X \subset I$ ja $Y \subset I$. Näiden joukkojen leikkaus muodostaa tyhjän joukon $X \cap Y = \phi$. Kaavassa verrataan alkiojoukkojen X ja Y esiintymistä pelkän X:n esiintymistiheyteen tietokannassa. Sääntöä, jonka uskottavuus-arvo on 100 %, sanotaan *tarkaksi* (exact) (Han & Kamber, 2001). Mikäli assosiaatiosäännön suklaakastike $\Rightarrow$ jäätelö uskottavuus-arvo on 100 %, niin suklaakastikkeen kanssa ostetaan aina jäätelöä.

$$\mathrm{conf}(X, Y) = \frac{\sup(X \cup Y)}{\sup(X)} \tag{2.10}$$

Garcia-Molina & al. (2002) huomauttavat, että vaikka säännöllä olisi korkea uskottavuus, se ei ole hyödyllinen ellei alkioilla ole korkea tuki. Mikäli tuki on alhainen, säännön esiintymisaste ei ole tarpeeksi suuri, jotta sääntöä kannattaisi hyödyntää. Assosiaatiosääntöjä, joilla on korkea uskottavuus ja vahva tuki sanotaan *vahvoiksi säännöiksi* (Chen & al., 1996). Tämän tutkielman luvussa 3 tutkimme, miten tarvittaessa alhaisenkin tuen omaavat assosiaatiosäännöt on mahdollista saada mukaan sääntökokoelmaan.

2.5.2 Subjektiiviset mittarit

Subjektiiviset mittarit ovat yhteydessä käyttäjän kokemuksiin ja uskomuksiin. Assosiaatiosäännön aiheuttama reaktio vaihtelee riippuen käyttäjän tiedoista, taidoista sekä kokemuksista, joten kaikki eivät koe samoja sääntöjä yhtä mielenkiintoisina. Liu & al. (2000) jakavat subjektiiviset mittarit kahteen luokkaan: yllätyksellisyys (unexpectedness) ja toiminnan sallittavuus (actionability).

Liu & al. (2000) määrittelevät *yllätyksellisyyden* siten, että sääntö on mielenkiintoinen, mikäli se on ennestään tuntematon käyttäjälle ja ristiriidassa käyttäjän aikaisemman tiedon kanssa. Esimerkiksi assosiaatiosäännöt kantarelli $\Rightarrow$ kermavaahto ja

piparkakku ⇒ sinihomejuusto voivat olla hyvin mielenkiintoisia sellaiselle ihmiselle, joka on kuvitellut, että näitä tuotteita ei voida yhdistää keskenään. Sen sijaan kokille edellä mainitut assosiaatiosäännöt eivät ole mielenkiintoisia, sillä hän on varmasti kokeillut näitä tuotteita yhdessä ja tietää ne kokemuksesta hyviksi yhdistelmiksi. Käyttäjä kokee siis säännöt mielenkiintoisina kokemustensa ja uskomustensa perusteella. Kun joku assosiaatiosääntö horjuttaa hänen uskomuksiaan, se tietenkin yllättää hänet.

Silberscatz ja Tuzhilin (1995) jakavat ihmisten uskomukset kahteen luokkaan. Toiset uskomukset ovat lujia ja toiset löyhiä uskomuksia. *Löyhät uskomukset* ovat niitä, jotka voivat muuttua, kun saamme uutta todistusaineistoa jostakin asiasta. Joistakin asioista meillä on vain tietty aavistus ja saadessamme uutta tietoa, uskomuksemme muuttuvat. Löyhä uskomus voi olla esimerkiksi uskomus siitä, että golf ei ole kunnon urheiluharrastus. Tällaisen uskomuksen saa helposti muutetuksi jollakin todistusaineistolla, esimerkiksi tilastotiedoilla. Tilastotieto voi olla esimerkiksi tieto siitä, että täyden golfkierroksen aikana kävellään noin 8-12 km. *Lujat uskomukset* ovat sen sijaan niitä, jotka eivät muutu, vaikka saisimme asiasta uutta todistusaineistoa. Muuttumattomien uskomuksien takia todistusaineiston ajatellaan olevan virheellistä. Luja uskomus voi olla esimerkiksi siitä, että ydinvoimalat ovat vaarallisia. Vaikka saisimme todistusaineistoa päinvastaisesta, esimerkiksi että ydinvoimaloiden jätteiden sijoituspaikat ovat turvallisia, luja uskomus ei silti muutu. Todistusaineiston ajateltaisiin olevan virheellistä, sillä esimerkiksi luonnon ilmiöitä ei voida ennustaa. Assosiaatiosäännöt, jotka ovat ristiriidassa omien lujien uskomusten kanssa, koetaan toki mielenkiintoisiksi, vaikka niihin ei uskottaisikaan.

Toiminnan sallittavuus tarkoittaa, että jotkut säännöt ovat mielenkiintoisia käyttäjälle, sillä ne aiheuttavat hänessä toimenpiteitä (Silberscatz & Tuzhilin, 1995). Säännön aiheuttama reaktio voi olla esimerkiksi lisätiedon hankkiminen asiasta tai syiden selvittäminen ja edelleen tilanteen parantaminen. Käyttäjä kokee mielenkiintoisiksi säännöt, joita hän voi käyttää hyödyksi työssään ja perustaa ratkaisujaan näille uusille tiedoille (Liu & al., 2000).

Silberscatz ja Tuzhilin (1995) korostavat, että jotkut säännöt voivat olla käyttäjälle samanaikaisesti sekä yllättäviä että toimenpiteitä aiheuttavia. Tällainen tilanne voi olla, jos saadaan selville, että kaakaojauhetta ja maitoa ostetaan samanaikaisesti: kaakaojauhe $\Rightarrow$ maito. Käyttäjä voi yllättyä saamastaan informaatiosta ja vaihtaa välittömästi hyllyjärjestystä omistamassaan kaupassa. Aina yllättyminen ei kuitenkaan aiheuta toimenpiteitä. Kauppias voisi edellisessä tilanteessa yllättyä tiedosta, mutta ei kuitenkaan toimisi mitenkään, mikäli kylmiö on eristetty muusta kaupasta liukuovilla. Silberscatz ja Tuzhilin (1995) mainitsevat myös, että jotkin toimenpiteitä aiheuttavat säännöt voivat olla myös odotettuja. Tällöin aineistoa voidaan käyttää tukemaan aikaisempia uskomuksia tai hypoteeseja ja ne aiheuttavat viimein toimenpiteitä. Esimerkiksi odotettu sääntö suklaakastike $\Rightarrow$ jäätelö voi aiheuttaa lopulta toimenpiteen, kun suklaakastike sijoitetaan jäätelöaltaan päälle.

Koska ihmiset reagoivat sääntöihin eri tavalla, on luotu järjestelmiä, joissa otetaan huomioon käyttäjän aikaisemmat kokemukset ja tiedot. Näin saadaan etsittyä mielenkiintoisia sääntöjä juuri tälle tietylle käyttäjälle. Liu & al. (2000) ovat kehittäneet *IAS-järjestelmän* (Interestingness Analysis System), jolla etsitään tietylle käyttäjälle mielenkiintoisia assosiaatiosääntöjä. IAS-järjestelmän jokaisella iteraatiokerralla käytetään hyväksi käyttäjän antamaa tietoa. IAS jakaa säännöt käyttäjän antamien tietojen mukaan neljään luokkaan. Ensimmäisessä luokassa ovat ne säännöt, jotka toteuttavat käyttäjän odotukset. Esimerkiksi sääntö suklaakastike $\Rightarrow$ jäätelö voi olla käyttäjän mielestä odotettu sääntö. Kolmessa muussa luokassa olevat säännöt ovat yllätyksellisiä. Käyttäjä voi merkata löydetyt säännöt mielenkiintoisiksi tai poistaa niitä. Mielenkiintoiset säännöt voi tallentaa myöhempää tarkastelua varten. IAS on saanut vaikutteita Apriori-algoritmista, jonka käymme läpi luvun 3 alussa.

3. EPÄYHTENÄISEN MINIMITUEN SOVELTAMINEN

Edellisessä luvussa perehdyimme tietämyksen etsimisen vaiheisiin, sekä tarkemmin assosiaatiosääntöihin. Tarkastelimme myös käytännön sovelluksia, assosiaatiosääntöihin vaikuttavia tekijöitä sekä sääntöihin liittyviä mittareita.

Assosiaatiosääntöjen johtamisessa on kaksi vaihetta (Han & Kamber, 2001). Ensin etsitään jollakin algoritmilla tietokannassa usein esiintyvät alkiojoukot. Toinen vaihe on johtaa näistä löydetyistä alkiojoukoista assosiaatiosääntöjä. Assosiaatiosääntöjä etsivissä algoritmeissa käytetään minimituki-parametria, jonka avulla saadaan esille tietokannassa usein esiintyvät alkiojoukot. Minimituki-parametri voi olla käytetyissä algoritmeissa yhtenäinen tai epäyhtenäinen. Tyypillisesti minimituki on yhtenäinen, sillä useat assosiaatiosääntöjen johtamiseen kehitetyt algoritmit perustuvat vuonna 1994 esitettyyn Apriori-algoritmiin, jossa käytetään yhtenäistä minimituki-parametria.

Wangin & al. (2003) mukaan yhtenäistä minimitukea käyttämällä ei kuitenkaan aina löydetä kaikkia usein esiintyviä alkiojoukkoja, esimerkiksi mielenkiintoisia poikkeustapauksia. Yun & al. (2003) huomauttavat, että tietokannan muodostama data voi esiintyä siinä suhteellisen usein tai sitten ei, riippuen tietokannan ominaisuuksista. Liu & al. (1999b) määrittelevät *harvinaisten alkioiden ongelman* (rare item problem): mikäli algoritmin minimituki-parametri asetetaan liian korkeaksi, sääntöihin ei saada mukaan tietokannassa harvoin esiintyviä alkioita. Harvoin esiintyvät alkiot löydetään, kun minimituki-parametria lasketaan. Tällöin seurauksena on sääntöjen räjähdysmäinen kasvu, sillä usein esiintyvät alkiojoukot yhdistyvät assosiaatiosäännöiksi kaikilla mahdollisilla tavoilla (Liu & al., 1999b). Harvoin esiintyvä data voi kuitenkin olla merkittävää. Yun & al. (2003) määrittelevät *merkittävän*, *harvinaisen datan* (significant rare data) siten, että sen frekvenssi tietokannassa ei ylitä minimitukea, mutta data assosioi kuitenkin usein tietyn datan kanssa suhteutettuna frekvenssiinsä. Merkittävällä harvinaisella datalla on alhainen tuki, mutta korkea uskottavuus.

Tässä luvussa tarkastelemme miten objektiivisiin mittareihin kuuluvan minimituki-parametrin käytön vaihtelulla saadaan esiin erilaisia alkioita assosiaatiosääntöihin.

Kohdassa 3.1 käymme läpi Apriori-algoritmin, jossa käytetään yhtenäistä minimitukea. Apriori-algoritmi on lähtökohta useille myöhemmin kehitetyille assosiaatiosääntöjä etsiville algoritmeille. Viime aikoina tutkimuksen kohteena on ollut epäyhtenäisen minimituki-parametrin käyttö assosiaatiosääntöjä etsivissä algoritmeissa. Kohdassa 3.2 perehdymme minimituki-parametrin käyttöön monitasoisia assosiaatiosääntöjä etsittäessä hierarkkisesta tiedosta. Monitasoisten assosiaatiosääntöjen etsimisessä voidaan käyttää epäyhtenäistä tai yhtenäistä minimitukea. Kun käytetään epäyhtenäistä minimitukea, on minimituki hierarkian jokaisella tasolla kuitenkin yhtenäinen. Tämän luvun kohdassa 3.3 käsittelemme menetelmän, jossa harvinaisten alkioiden ongelma ratkaistaan käyttämällä useaa alkiokohtaista minimitukea, yhtenäisen minimituki-parametrin sijaan. Kohdassa 3.4 saamme merkittävän, harvinaisen datan esille käyttäen suhteellista tukea. Tässä menetelmässä käytetään kahta minimituki-parametria, sekä suhteellista tukea. Kohdassa 3.5 käsittelemme tukirajoitteisiin perustuvan menetelmän periaatteen, jossa epäyhtenäinen minimituki-parametri annetaan kokonaiselle alkiojoukolle.

3.1 Lähtökohtana yhtenäisen minimituen käyttö Apriori-algoritmissa

Useat assosiaatiosääntöjen etsimiseen kehitetyt algoritmit perustuvat Apriori-algoritmiin. Se on Agrawalin ja Srikantin (1994) luoma algoritmi usein esiintyvien alkiojoukkojen etsimistä varten. Toisin kuin vuonna 1993 kehitetyllä AIS-algoritmilla, saadaan Apriori-algoritmilla aikaan assosiaatiosääntöjä, joissa implikaation seurauksena oleva alkiojoukko koostuu useista alkioista (Agrawal & al., 1993; Agrawal & Srikant, 1994). Sääntö lahjapaperi $\Rightarrow$ {lahjanaru, teippi} on esimerkki assosiaatiosäännöstä, jossa säännön implikaation seurauksena oleva alkiojoukko muodostuu kahdesta alkiosta.

Apriori-algoritmissa käytetään yhtenäistä minimituki-parametria. Tällöin minimituki-parametri on sama kaikille k-alkiojoukoille. Käytännössä esimerkiksi k-alkiojoukon tuki tietokannassa on suurempi kuin esimerkiksi (k+1)-alkiojoukon. Tästä johtuen Apriori-algoritmi ei välttämättä löydä kaikkia usein esiintyviä (k+1)-alkiojoukkoja, sillä niiden tuki ei ylitä minimituki-parametria. Kun algoritmi käyttää yhtenäistä minimitukea, käytetään samaa minimituki-parametria, huolimatta siitä, onko käsittelyssä

oleva alkio tai alkiojoukko tietokannassa usein esiintyvä vai harvoin esiintyvä. Tietokannassa esiintyvien alkioiden tuki kuitenkin vaihtelee riippuen alkioiden ominaisuuksista. Tämän seurauksena assosiaatiosääntöihin ei saada mukaan merkittävää, harvinaista dataa.

Boolen-assosiaatiosääntö (2.1) on yksinkertaisin muoto assosiaatiosäännöistä, sillä se on yksitasoinen ja yksiulotteinen. Boolen-assosiaatiosääntöjä saadaan esille Apriori-algoritmilla. Algoritmille annetaan lähtötietoina tapahtumaperäinen tietokanta D, sekä käyttäjän ilmoittama numeerinen minimituki-parametri *min_sup*. Apriori-algoritmin suorituksessa käytetään hyväksi aikaisempaa tietoa, josta sen nimi johtuukin (Han & Kamber, 2001). Algoritmin suorituksessa käydään iteratiivisella leveyshaulla läpi tietokantaa, jolloin saadaan aikaisempaa tietoa hyväksi käyttäen selville *kandidaattialkiojoukot* C_k. Kandidaattialkiojoukot ovat mahdollisia usein esiintyviä alkiojoukkoja L_k. Usein esiintyvät alkiojoukot tietokannassa saadaan kandidaattialkiojoukoista vertaamalla niiden frekvenssiä annettuun minimituki-parametriin. Näistä usein esiintyvistä alkiojoukoista johdetaan lopuksi assosiaatiosäännöt.

Apriori-algoritmissa (kuvat 3.1-3.3) on kaksi vaihetta. Ensimmäisessä vaiheessa luodaan kandidaattialkiojoukot C_k. Toisessa vaiheessa käydään läpi tietokanta ja lasketaan kandidaattialkiojoukkojen esiintymislukumäärät. Kandidaattialkiojoukoista saadaan usein esiintyvät alkiojoukot L_k karsimalla ne, joiden esiintymislukumäärä ei yllä annettuun minimituki-parametriin. Näitä kahta vaihetta jatketaan niin kauan, että usein esiintyviä alkiojoukkoja ei enää löydy (Han & Kamber, 2001).

Apriori-algoritmin tehostamiseksi käytetään *Apriori-ominaisuutta*. Apriori-ominaisuuden ideana on, että alkiojoukko, jonka alijoukko ei ole usein esiintyvä, ei voi itsekään olla usein esiintyvä alkiojoukko. Apriori-ominaisuutta kutsutaan *alaspäin suunnatuksi sulkeumaksi* (downward closure). Kun kandidaattialkiojoukko muodostetaan, hylätään Apriori-algoritmin funktiolla *has_infrequent_subset* (kuva 3.2) ne alkiojoukot, jotka eivät toteuta Apriori-ominaisuutta. Funktio saa parametreina kandidaattialkiojoukon sekä edellisen kierroksen usein esiintyvän alkiojoukon. Kandidaattialkiojoukon alijoukkoja verrataan edellisen kierroksen usein esiintyvään alkiojoukkoon. Esimerkiksi alkiojoukon {1, 2, 3} alijoukkojen {2, 3}, {1, 2}, {1, 3} täytyy

esiintyä edellisen kierroksen usein esiintyvässä alkiojoukossa L_2, jotta alkiojoukko hyväksytään kandidaattialkiojoukoksi. Mikäli alkiojoukon {1, 2, 3} alijoukot esiintyvät alkiojoukossa L_2, se toteuttaa Apriori-ominaisuuden ja funktio *has_infrequent_subset* palauttaa arvon FALSE.

Input: Database, D, of transactions; minimum support threshold, *min_sup*.
Output: L, frequent itemsets in D.
Method:

(1) L_1 = find_frequent_1-itemsets(D);
(2) **for** (k = 2; $L_{k-1} \neq \phi$; k++) {
(3) C_k = apriori-gen(L_{k-1});
(4) **for each** transaction $t \in D$ { // scan D for counts
(5) C_t = subset(C_k, t); // get the subsets of t that are candidates
(6) **for each** candidate $c \in C_t$
(7) $c.count$++;
(8) }
(9) $L_k = \{c \in C_k \mid c.count \geq min\_sup\}$
(10) }
(11) **return** $L = \cup_k L_k$;

Kuva 3.1 Apriori-algoritmi (Han & Kamber, 2001).

function has_infrequent_subset(c: candidate k-itemset; L_{k-1}: frequent (k-1)-itemsets);

(1) **for each** (k-1)-subset s of c
(2) **if** $s \notin L_{k-1}$ **then** //use prior knowledge
(3) **return** TRUE;
(4) **return** FALSE;

Kuva 3.2 Apriori-ominaisuus (Han & Kamber, 2001).

Apriori-algoritmin ensimmäisen vaiheen mukaisesti kandidaattialkiojoukkoja muodostetaan edellisen iteraatiokierroksen usein esiintyvistä alkiojoukoista. Kandidaattialkiojoukot muodostetaan luonnollisella liitosoperaatiolla Apriori-algoritmin funkti-

olla *apriori-gen* (kuva 3.3). Alkiojoukot ovat ensimmäisellä iteraatiokierroksella 1-alkiojoukon kokoisia. Tällöin on haettu tietokannasta kaikki usein esiintyvät alkiot algoritmin (kuva 3.1) funktiolla *find_frequent_1-itemsets*. Usein esiintyvistä 1-alkiojoukoista tehdään liitosoperaatio $L_1 \bowtie L_1$ (tarkoittaa karteesista tuloa $L_1 \times L_1$, koska k-1 = 0), jolloin saadaan kandidaatti-2-alkiojoukot ilman duplikaatteja. Kandidaattialkiojoukot kasvavat jokaisella iteraatiokierroksella k. Leveyshaussa käytetään aikaisempaa tietoa, kun k-alkiojoukkoa käytetään (k+1)-alkiojoukon etsimiseen.

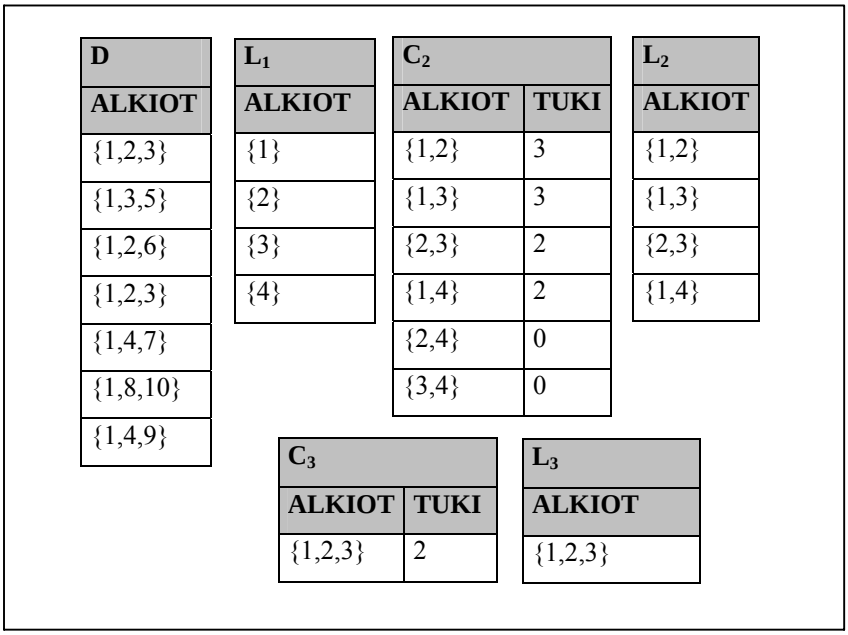
```
function apriori-gen(L_{k-1}: frequent (k–1)-itemsets);
(1)  for each itemset l_1 ∈ L_{k-1}
(2)      for each itemset l_2 ∈ L_{k-1}
(3)          if (l_1[1] = l_2[1]) ∧ ( l_1[2] = l_2[2]) ∧ … ∧ ( l_1[k-2] = l_2[k-2]) ∧ ( l_1[k-1] < l_2[k-1]) then {
(4)              c = l_1 ⋈ l_2 ; // join step: generate candidates
(5)              if has_infrequent_subset(c, L_{k-1}) then
(6)                  delete c; // prune step: remove unfruitful candidate
(7)              else add c to C_k;
(8)          }
(9)  return C_k;
```

Kuva 3.3 Kandidaattialkiojoukkojen luominen (Han & Kamber, 2001).

Apriori-algoritmin toisen vaiheen mukaisesti tietokannasta haetaan kandidaatti-k-alkiojoukkojen esiintymislukumäärät. L_k-alkiojoukkoon lisätään C_k-alkiojoukosta kaikki ne alkiojoukot, joiden frekvenssi toteuttaa minimituki-parametrin. Sekä kandidaattialkiojoukkojen muodostamista että niiden suodattamista jatketaan niin kauan, että usein esiintyviä alkiojoukkoja ei enää löydy. Apriori-algoritmilla saadaan siis tuloksena usein esiintyviä alkiojoukkoja tapahtumaperäisestä tietokannasta.

Tarkastelemme seuraavaksi Apriori-algoritmin suoritusta kuvan 3.4 esimerkin mukaisesti. Annamme algoritmille minimituki-parametriksi *min_sup* = 2. Usein esiintyvien alkiojoukkojen tulee tällöin esiintyä tietokannassa kahdesti tai useammin. Apriori-algoritmin ensimmäinen tehtävä on hakea tietokannasta usein esiintyvät 1-

alkiojoukot, jotka muodostavat ensimmäisen alkiojoukon L_1. Alkiojoukko L_1 saadaan algoritmin (kuva 3.1) funktiolla *find_frequent_1-itemsets*. Alkiojoukossa on tällöin kaikki tapahtumaperäisen tietokannan minimituen toteuttavat alkiot 1-alkiojoukkoina. Kuvan 3.4 tapahtumaperäisestä tietokannasta saadaan L_1-alkiojoukkoon neljä 1-alkiojoukkoa, {1}, {2}, {3} ja {4}. Seuraavaksi muodostetaan usein esiintyvistä 1-alkiojoukoista kandidaatti-2-alkiojoukot karteesisena tulona $L_1 \times L_1$. Kandidaattialkiojoukkoon C_2 saadaan 2-alkiojoukot {1, 2}, {1, 3}, {2, 3}, {1, 4}, {2, 4} ja {3, 4}. Apriori-algoritmin toisen vaiheen mukaisesti käydään tietokanta läpi ja lasketaan kandidaattialkiojoukkojen esiintymislukumäärät tietokannassa. Alkiojoukkoon L_2 saadaan mukaan osa kandidaattialkiojoukoista, kun verrataan kandidaattialkiojoukkojen esiintymislukumääriä minimituki-parametriin. Kandidaattialkiojoukot {2, 4} ja {3, 4} karsitaan, sillä niiden frekvenssi ei ylitä minimitukea. Seuraavalla kierroksella luodaan kandidaattialkiojoukko C_3 kuvan 3.3 liitosoperaatiolla, funktion *apriori-gen* rivien kolme ja neljä mukaisesti. Apriori-ominaisuuden johdosta C_3-alkiojoukkoon ei lisätä esimerkiksi 3-alkiojoukkoa {1, 2, 4}, sillä sen alijoukko {2, 4} ei ole usein esiintyvä. Tuloksena saamme L_3-alkiojoukon {1, 2, 3}, joka on tapahtumaperäisessä tietokannassa usein esiintyvä alkiojoukko. Näin olemme toteuttaneet assosiaatiosääntöjen johtamisen ensimmäisen vaiheen.



D

ALKIOT
{1,2,3}
{1,3,5}
{1,2,6}
{1,2,3}
{1,4,7}
{1,8,10}
{1,4,9}

L_1

ALKIOT
{1}
{2}
{3}
{4}

C_2

ALKIOT	TUKI
{1,2}	3
{1,3}	3
{2,3}	2
{1,4}	2
{2,4}	0
{3,4}	0

L_2

ALKIOT
{1,2}
{1,3}
{2,3}
{1,4}

C_3

ALKIOT	TUKI
{1,2,3}	2

L_3

ALKIOT
{1,2,3}

Kuva 3.4 Apriori-algoritmin esimerkki.

Assosiaatiosääntöjen johtamisen toisen vaiheen mukaisesti, usein esiintyvistä alkiojoukoista muodostetaan assosiaatiosääntöjä. Edellisestä esimerkistä saamme alkiojoukon {1, 2, 3} usein esiintyväksi alkiojoukoksi, josta voimme tuottaa assosiaatiosääntöjä. Vahvat säännöt ylittävät uskottavuus- ja minimituki-mittarit, jotka käsittelimme alakohdassa 2.5.1. Apriori-algoritmilla löydetyt usein esiintyvät alkiojoukot ylittävät automaattisesti minimituki-parametrin (Han & Kamber, 2001). Assosiaatiosääntöjen täytyy ylittää myös uskottavuus-parametri.

Assosiaatiosääntöjä muodostettaessa jaetaan usein esiintyvä alkiojoukko ensin epätyhjiksi alijoukoiksi. Kuvan 3.4 3-alkiojoukosta {1, 2, 3}, saamme epätyhjät alijoukot {1}, {2}, {3}, {1, 2}, {1, 3} ja {2, 3}. Nämä joukot yhdistetään säännöiksi ja niille lasketaan uskottavuus kaavalla (2.10). Kuvassa 3.5 on laskettu uskottavuus-arvot usein esiintyvästä alkiojoukosta {1, 2, 3} johdetuille assosiaatiosäännöille. Säännön 2 $\wedge$ 3 $\Rightarrow$ 1 alkiojoukko {1, 2, 3} esiintyy tietokannassa kaksi kertaa, kuten myös alkiojoukko {2, 3}. Assosiaatiosäännön (3.1) uskottavuudeksi saamme 100 %, jolloin sääntöä sanotaan tarkaksi. Alkiojoukosta {2, 3} seuraa siis aina alkiojoukko {1}. Mikäli asetamme uskottavuus-parametriksi 35 %, ensimmäinen assosiaatiosääntö ylittää annetun parametrin. Kuvan 3.5 assosiaatiosäännöistä ainoastaan sääntö (3.4) alittaa uskottavuus-parametrin.

$2 \wedge 3 \Rightarrow 1$ Uskottavuus 2/2 = 100 %	(3.1)
$1 \wedge 2 \Rightarrow 3$ Uskottavuus 2/3 = 67 %	(3.2)
$1 \wedge 3 \Rightarrow 2$ Uskottavuus 2/3 = 67 %	(3.3)
$1 \Rightarrow 2 \wedge 3$ Uskottavuus 2/7 = 29 %	(3.4)
$2 \Rightarrow 1 \wedge 3$ Uskottavuus 2/3 = 67 %	(3.5)
$3 \Rightarrow 1 \wedge 2$ Uskottavuus 2/3 = 67 %	(3.6)

Kuva 3.5 Assosiaatiosääntöjen uskottavuus-arvot.

Kuvan 3.4 tapahtumaperäisestä tietokannasta huomaamme, että alkio 4 esiintyy tietokannassa kaksi kertaa ja aina esiintyessään, tapahtumassa on mukana myös alkio 1. Tämä tieto voidaan lukea merkittäväksi, harvinaiseksi dataksi. Apriori-algoritmin yh-

tenäisen minimituki-parametrin johdosta, tätä assosiaatiota ei kuitenkaan huomioida. Luvun myöhemmissä kohdissa perehdymme menetelmiin, joilla ratkaistaan harvinaisten alkioiden ongelma sekä löydetään myös merkittävää, harvinaista dataa.

Tietokannat, joista assosiaatiosääntöjä etsitään, ovat usein valtavia. Läpikäymästämme esimerkistä huomaamme, että algoritmin suoritus vaatii useita tietokannan läpikäyntejä sekä useita kandidaattialkiojoukkoja. Apriori-algoritmilla saadaan suunnaton määrä kandidaattialkiojoukkoja, kun alkiojoukot kasvavat kierros kierrokselta. Kantardzicin (2003) esittämästä kaavasta (3.7) huomaamme, miten alkiojoukon suuruus vaikuttaa kandidaattien määrään. Yhden usein esiintyvän 100-alkiojoukon muodostamiseen tarvitaan huomattava määrä kandidaattialkiojoukkoja.

$$\sum_{i=1}^{100} \left(\frac{100}{i} \right) = 2^{100} - 1 \approx 10^{30} \tag{3.7}$$

Monimutkaisuus näkyy hyvin myös liitosten lukumäärän lisääntyessä *k*:n kasvaessa Houtsman ja Swamin (1995) esittämässä SQL-tulkinnassa L_k-alkiojoukkojen etsimiseksi (kuva 3.6).

```
insert into Lk
select r1.alkio,…, rk.alkio, count(*)
from Lk-1 l, D r1,…, D rk
where r1.TID =…= rk.TID and
        r1.alkio = l.alkio1 and
        …
        rk-1.alkio = l.alkiok-1 and
        rk.alkio > rk-1.alkio
group by r1.alkio,…, rk.alkio
having count(*) >= :min_sup
```

Kuva 3.6 L_k-alkiojoukkojen etsiminen (Houtsma & Swami, 1995).

Kuvan 3.6 SQL-tulkinnassa tapahtumatiedot ovat tietokantataulussa D (kuva 3.7), joka on muodostettu kuvan 3.4 esittämästä tietokannasta D.

TID	ALKIO	TID	ALKIO
1	1	4	3
1	2	5	1
1	3	5	4
2	1	5	7
2	3	6	1
2	5	6	8
3	1	6	10
3	2	7	1
3	6	7	4
4	1	7	9
4	2		

Kuva 3.7 Tietokantataulu D.

Apriori-algoritmista on tehty useita tehokkaampia vaihtoehtoja. Yang & al. (2001) jakavat suorituskykyä parantavat keinot kolmeen kategoriaan. Ensimmäisessä kategoriassa vähennetään kandidaattien määrää hajautustekniikalla. Toisessa kategoriassa vähennetään tietokannan läpikäyntien määrää osioimalla. Kolmannessa kategoriassa käytetään *kokoavan* (bottom-up) *haun* ja *osittavan* (top-down) *haun* yhdistelmää L_k-alkiojoukkoja muodostettaessa. Han ja Kamber (2001) esittävät edellä mainittujen tehostusmenetelmien lisäksi otannan ja *dynaamisen alkiojoukkojen laskennan*. Tässä tutkielmassa ei kuitenkaan keskitytä algoritmien tehokkuuteen assosiaatiosääntöjä etsittäessä, vaan assosiaatiosääntöjen luotettavuuteen.

3.2 Hierarkkinen tieto

Kohdassa 3.1 kävimme läpi Apriori-algoritmin, jossa käytetään yhtenäistä minimituki-parametria. Tässä kohdassa perehdymme minimituen käyttöön, kun etsitään monitasoisia assosiaatiosääntöjä hierarkkisesta tiedosta. Monitasoinen assosiaatiosääntö on esimerkiksi sääntö (2.8) ja esimerkki hierarkkisesta tiedosta on kuvan 2.3 monitasoi-

nen käsitehierarkia. Monitasoisessa käsitehierarkiassa alkiot on luokiteltu hierarkian eri tasoille, jolloin alkioiden tuki on todennäköisesti suurempi hierarkian ylemmillä tasoilla kuin alemmilla, tarkemmilla tasoilla (Han & Fu, 1995). Ylimpien tasojen alkioilla on suurempi tuki, sillä ne sisältävät yleisempää tietoa alkioista. Monitasoisten assosiaatiosääntöjen etsimisalgoritmeissa törmätään siis harvinaisten alkioiden ongelmaan. Mikäli algoritmin yhtenäinen minimituki-parametri asetetaan liian korkeaksi, menetetään alempien tasojen harvinaisia assosiaatioita. Jos sen sijaan minimituki-parametri asetetaan alhaiseksi, saadaan ylemmiltä tasoilta useita turhia sääntöjä. Esimerkiksi kuvan 2.3 käsitehierarkiaa läpikäyvälle algoritmille voimme antaa yhtenäiseksi minimituki-parametriksi 5 %. Tällöin ei kuitenkaan välttämättä löydetä tarkimpien tasojen assosiaatiosääntöjä, sillä ne eivät ylitä annettua tukea. Hierarkkisen tiedon "harvinaisten alkioiden ongelmasta" huolimatta on kehitetty algoritmeja, joissa minimituki eri tasoilla on yhtenäinen (Han & Kamber, 2001; Srikant & Agrawal, 1995).

Han ja Kamber (2001) esittävät epäyhtenäiseen minimitukeen pohjautuvan periaatteen, jota kutsutaan *tasoittain vähenevästi minimitueksi*. Tasoittain vähenevää minimitukea käytettäessä annetaan käsitehierarkian eri tasoille yksilölliset minimituet, kun hierarkiaa käydään läpi jollakin algoritmilla. Minimituet pienenevät, kun siirrytään käsitehierarkiassa alaspäin, tarkemmille tasoille. Kuvan 2.3 tapauksessa 1-tasolle voimme antaa minimituki-parametriksi 13 %, 2-tasolle 5 % ja 3-tasolle 2 %. Tällöin otetaan huomioon se, että alemmilla tasoilla tuki ei ole niin suuri kuin ylemmillä tasoilla. Lisäksi ylemmillä tasoilla ei tarvitse olla liioitellun pieni minimituki-parametri, kun alempia tasoja ei oteta huomioon. Jos jonkun jälkeläisen vanhempi ylemmältä tasolta ei ylitä annettua minimitukea, optimointimahdollisuus on olla käymättä läpi tämän jälkeläiset alemmalla tasolla (Han & Kamber, 2001). Huomattavaa on se, että vaikka tasoittain vähenevässä minimituki-menetelmässä käytetään useita minimituki-parametreja, jokaisella tasolla minimituki-parametri on kuitenkin yhtenäinen.

Tasoittain vähenevän minimituen menetelmä perustuu Hanin ja Fun (1995) käsitehierarkiaan pohjautuvaan menetelmään monitasoisten assosiaatiosääntöjen etsimiseksi. Heidän menetelmässään tapahtumaperäisen tietokannan alkiot muunnetaan käsitehierarkian avulla numerosarjoiksi. Esimerkiksi kuvan 2.3 käsitehierarkian mukaisesti

voidaan Talonpojan grillimaustettu dippikastike muuntaa numerosarjaksi 111. Numerosarjan ensimmäinen luku viittaa käsitehierarkiassa ensimmäisellä tasolla olevaan dippikastikkeeseen. Keskimmäinen luku osoittaa toisen tason grillimaustettuun vaihtoehtoon ja viimeinen luku tarkoittaa kolmannen tason Talonpoikaa. Kuvassa 3.8 on tapahtumaperäinen tietokanta T[1], jossa alkiot on muunnettu käsitehierarkian mukaisesti numerosarjoiksi.

TID	ALKIOT
1	{111,121,211,221}
2	{111,211,222,323}
3	{112,122,221,411}
4	{111,121}
5	{111,122,211,221,413}
6	{211,323,524}
7	{323,411,524,713}

Kuva 3.8 Muunnettu tapahtumaperäinen tietokanta T[1] (Han & Fu, 1995).

Han ja Fu (1995) ovat kehittäneet ML-T2-algoritmin (kuva 3.9) monitasoisten assosiaatiosääntöjen johtamiseksi käsitehierarkian avulla. Algoritmissa käytetään tasoittain vähenevää minimitukea. Algoritmille annetaan lähtötietoina yhtenäinen minimitukiparametri *min_sup*[*l*], jokaista tasoa *l* varten, jolloin ylemmille tasoille voidaan antaa suurempi minimituki kuin alemmille tasoille. Toinen algoritmin vaatima lähtötieto on käsitehierarkian mukaan numerosarjoiksi muunnettu tapahtumaperäinen tietokanta, kuten kuvan 3.8 tietokanta T[1]. Algoritmille annetaan myös maksimitaso *max_level*, joka ilmoittaa käsitehierarkian tasojen lukumäärän. Algoritmin tuloksena saadaan jokaiselta tasolta usein esiintyvät alkiojoukot monitasoisten assosiaatiosääntöjen johtamista varten (Han & Fu, 1995).

Hanin ja Fun (1995) algoritmissa ML-T2 etsitään usein esiintyviä alkiojoukkoja iteratiivisesti. Algoritmi lähtee liikkeelle käsitehierarkian ensimmäiseltä tasolta ja jatkaa siitä alemmille tasoille käyttäen Apriori-algoritmin tapaan hyväkseen sekä aikaisempaa tietoa että Apriori-ominaisuutta. Algoritmi ML-T2 etsii tasolta ensin usein esiintyvät 1-alkiojoukot. Tämän jälkeen etsitään tältä tasolta muut usein esiintyvät *k*-

alkiojoukot ($k \geq 2$). Usein esiintyvien k-alkiojoukkojen etsimistä varten käytetään kandidaattialkiojoukkoja, jotka saadaan tuotettua aiemmista usein esiintyvistä alkiojoukoista. Kandidaattialkiojoukkoja muodostettaessa tarkistetaan, että niiden alijoukot ovat myös usein esiintyviä alkiojoukkoja. Algoritmin läpikäyntiä jatketaan, kunnes saadaan jollakin tasolla tyhjä 1-alkiojoukko tai saavutetaan maksimitaso (Han & Fu, 1995).

Input: A hierarchy-information encoded database transaction, $T[1]$;
minimum support threshold, $min\_sup[l]$ for each concept level l;
maximum level in concept hierarchy, max_level.

Output: LL, frequent itemsets in $T[l]$.

Method:

```
(1)  for (l = 1; L[l,1] ≠ ϕ and l < max_level; l++) {
(2)    if l = 1 then {
(3)      L[l,1] = get_large_1_itemsets(T[1], l);
(4)      T[2] = get_filtered_transaction_table(T[1], L[1,1]);
(5)    }
(6)    else L[l,1] = get_large_1_itemsets(T[2], l);
(7)    for (k = 2; L[l, k-1] ≠ ϕ; k++) {
(8)      C_k = get_candidate_set(L[l, k-1]);
(9)      for each transaction t ∈ T[2] {
(10)       C_t = get_subsets(C_k, t); // Candidates contained in t
(11)       for each candidate c ∈ C_t
(12)         c.count++;
(13)     }
(14)     L[l,k] = {c ∈ C_k | c.count ≥ min_sup[l]}
(15)   }
(16)   LL[l] = ∪_k L[l, k]
(17) }
(18) return LL;
```

Kuva 3.9 ML-T2-algoritmi (Han & Fu, 1995).

Tarkastelemme seuraavaksi monitasoisten assosiaatiosääntöjen etsimistä ML-T2-algoritmilla Hanin ja Fun (1995) esimerkin mukaisesti. Käytämme kuvan 3.8 tapahtumaperäistä tietokantaa, jossa alkiot on muunnettu käsitehierarkian mukaisesti numerosarjoiksi. Tapahtumaperäisessä tietokannassa yksittäisen tapahtuman toistuvat alkiot käsitellään, kuin ne esiintyisivät tapahtumassa vain kerran.

Hanin ja Fun (1995) menetelmässä käytetään tasoittain vähenevää minimitukea, jolloin jokaisella tasolla on oma yhtenäinen minimituki-parametri. Annamme ensimmäiselle, yleiselle tasolle, minimituki-parametriksi *min_sup*[1] = 4. Algoritmin maksimitasoksi annamme *max_level* = 3, jolloin algoritmi käy korkeintaan kolme käsitehierarkian tasoa läpi. Tietokannan ensimmäisessä läpikäynnissä haetaan usein esiintyvät 1-alkiojoukot ensimmäiseltä tasolta alkiojoukkoon *L*[1,1]. Ensimmäisellä tasolla käytetään usein esiintyvien 1-alkiojoukkojen etsimiseen algoritmille annettua tapahtumaperäistä tietokantaa T[1], jossa alkiot on muunnettu numerosarjoiksi. Usein esiintyvät 1-alkiojoukot saadaan alkiojoukkoon *L*[1,1] (kuva 3.10) algoritmin funktiolla *get_large_1_itemsets* rivin 3 mukaisesti. Usein esiintyvät 1-tason 1-alkiojoukot löydetään, kun otetaan numerosarjan ensimmäinen numero huomioon ja verrataan sen frekvenssiä tietokannassa minimitukeen. Numerosarjan ensimmäinen numero viittaa käsitehierarkian ensimmäiseen tasoon. Tällöin etsitään alkioita numerosarjoilla 1**,…, 7**, sillä numerosarjojen kaksi viimeistä lukua eivät ole merkitseviä ensimmäisellä tasolla. Kuvan 3.10 alkiojoukon *L*[1,1] ensimmäinen alkiojoukko {1**}, esiintyy tietokannassa viidesti, sillä se esiintyy kuvan 3.8 viidessä ensimmäisessä tapahtumassa.

Ensimmäisen tason usein esiintyvien 1-alkiojoukkojen avulla saadaan tapahtumaperäisestä tietokannasta (kuva 3.8) suodatettua tietokanta T[2] (kuva 3.10), jota algoritmissa siirrytään käyttämään ensimmäisen tason usein esiintyvien 1-alkiojoukkojen löydyttyä. Suodatettu tietokanta saadaan algoritmin funktiolla *get_filtered_transaction_table* rivin 4 mukaisesti suodattamalla alkuperäisestä tietokannasta alkiot, jotka eivät esiinny alkiojoukossa *L*[1,1].

Kun usein esiintyvät 1-alkiojoukot on löydetty, etsitään tältä tasolta usein esiintyvät *k*-alkiojoukot (kuvan 3.9 rivit 7-15). Ensimmäiseltä tasolta löytyy vain yksi usein

esiintyvä 2-alkiojoukko $L[1,2]$ (kuva 3.10). Se on muodostettu käyttäen ensimmäisen tason usein esiintyviä 1-alkiojoukkoja kandidaattialkiojoukkoina. Kandidaattialkiojoukko C_k saadaan algoritmin funktiolla *get_candidate_set* rivin 8 mukaisesti. Apriori-ominaisuuden johdosta kandidaattialkiojoukkoihin ei oteta mukaan alkiojoukkoja, joiden alijoukot eivät ole usein esiintyviä alkiojoukkoja. Alkiojoukon $L[1,2]$ tuen löytämiseksi käytetään suodatettua tietokantaa T[2] (kuvan 3.9 rivit 9-13). Tietokannasta T[2] löydetään neljä tapahtumaa, joissa esiintyy ensimmäisen tason 2-alkiojoukko $L[1,2]$. Alkiojoukko {1**, 2**} toteuttaa siis tasolle annetun minimitukiparametrin, algoritmin rivin 14 mukaisesti.

ALKIOT	TUKI
{1**}	5
{2**}	5

TID	ALKIOT
1	{111,121,211,221}
2	{111,211,222}
3	{112,122,221}
4	{111,121}
5	{111,122,211,221}
6	{211}

ALKIOT	TUKI
{1**, 2**}	4

Kuva 3.10 $L[1,1]$, T[2] ja $L[1,2]$.

Annetaan Hanin ja Fun (1995) esimerkin mukaisesti tasolle kaksi minimitukiparametriksi *min_sup*[2] = 3. Toisen tason usein esiintyvät 1-alkiojoukot saadaan alkiojoukkoon $L[2,1]$ (kuva 3.11) algoritmin funktiolla *get_large_1_itemset* rivin 6 mukaisesti. Toisella tasolla käytetään tietokannan T[2] alkioiden numerosarjojen kahta ensimmäistä lukua, jotka osoittavat alkion sijainnin kahdella ensimmäisellä tasolla. Numerosarjan kolmas numero ei tällöin ole merkitsevä. Toisen tason usein esiintyvissä 1-alkiojoukoissa ovat mukana vain ne alkiojoukot, joiden tuki toteuttaa minimituki-parametrin suodatetussa tietokannassa T[2]. Toisen tason 2-alkiojoukot alkiojoukossa $L[2,2]$ (kuva 3.11) saadaan selville käyttämällä saman tason usein esiintyviä 1-alkiojoukkoja kandidaattialkiojoukkojen muodostamisessa. Alkiojoukkojen tuet saadaan tietokannasta T[2]. Toisen tason 3-alkiojoukon $L[2,3]$ (kuva 3.11) etsimiseen käytetään saman tason usein esiintyviä 2-alkiojoukkoja kandidaattialkiojoukkojen

muodostamisessa. Kandidaattialkiojoukoksi ei hyväksytä esimerkiksi alkiojoukkoa {11*, 12*, 21*}, sillä sen alijoukko {12*, 21*} ei ole usein esiintyvä.

ALKIOT	TUKI
{11*}	5
{12*}	4
{21*}	4
{22*}	4

ALKIOT	TUKI
{11*, 12*}	4
{11*,21*}	3
{11*,22*}	4
{12*,22*}	3
{21*,22*}	3

ALKIOT	TUKI
{11*,12*,22*}	3
{11*,21*,22*}	3

Kuva 3.11 $L[2,1]$, $L[2,2]$ ja $L[2,3]$.

Tasolla kolme löydetään usein esiintyviä alkiojoukkoja samalla menetelmällä kuin kahdella edellisellä tasolla. Kolmannelle tasolle annamme minimituki-parametriksi *min_sup*[3] = 3. Kolmannella tasolla saadaan usein esiintyväksi 1-alkiojoukoksi $L[3,1]$ ja 2-alkiojoukoksi $L[3,2]$ (kuva 3.12). Usein esiintyvien alkiojoukkojen etsiminen päättyy, sillä maksimitaso hierarkiassa on saavutettu.

ALKIOT	TUKI
{111}	4
{211}	4
{221}	3

ALKIOT	TUKI
{111,211}	3

Kuva 3.12 $L[3,1]$ ja $L[3,2]$.

Assosiaatiosääntöjen johtamisen toisen vaiheen mukaisesti voidaan tasolta l löytyneistä usein esiintyvistä alkiojoukoista muodostaa monitasoisia assosiaatiosääntöjä. Han ja Fu (1999) huomauttavat, että jokaiselle tasolle l annetaan oma uskottavuus-parametri *min_conf*[l], joka assosiaatiosääntöjen tulee toteuttaa. Asetamme tasolle kolme uskottavuus-parametriksi *min_conf*[3] = 25 %. Voimme nyt muodostaa kuvan

3.12 alkiojoukosta L[3,2] assosiaatiosäännön 111 $\Rightarrow$ 211. Säännön uskottavuus-arvoksi saamme 75 %, kun käytämme kuvan 3.8 tietokantaa T[1]. Tällöin monitasoinen assosiaatiosääntö ylittää uskottavuus-parametrin. Kuvan 2.3 mukaisesti alkio 111 voisi merkitä Talonpojan grillimaustettua dippikastiketta ja alkio 211 voisi merkitä Sipsikuninkaan grillimaustettuja perunalastuja, jolloin monitasoinen assosiaatiosääntö tulisi muotoon Talonpojan grillimaustettu dippikastike $\Rightarrow$ Sipsikuninkaan grillimaustetut perunalastut. Tämä assosiaatiosääntö antaa tarkkaa tietoa kuluttajan ostoskäyttäytymisestä.

3.3 Alkiokohtainen minimituki

Kohdassa 3.1 käsittelimme Apriori-algoritmin, jossa käytetään yhtä yhtenäistä minimituki-parametria. Kohdassa 3.2 tutustuimme epäyhtenäisen minimituen käyttöön, monitasoisia assosiaatiosääntöjä etsittäessä hierarkkisesta tiedosta tasoittain vähenevän minimituen avulla. Myös Liun & al. (1999b) kehittämä menetelmä perustuu epäyhtenäisen minimituen käyttöön. Heidän menetelmässään assosiaatiosääntöjä etsitään tietokannasta käyttämällä *useaa minimitukea* (multiple minimum support). Menetelmän ideana on, että käyttäjä voi määritellä jokaiselle alkiolle oman minimituen, jolloin otetaan huomioon alkioiden erilainen luonne niiden esiintyessä tietokannassa. Usean minimituen määrittelyn avulla voidaan harvoin esiintyvälle alkiolle antaa sellainen minimituki, että se saadaan mukaan assosiaatiosääntöihin. Tällöin ratkaistaan harvinaisten alkioiden ongelma, joka esiintyy yhtenäisen minimituen käytössä.

Liu & al. (1999b) antavat jokaiselle alkiolle oman *minimialkiotuen* (minimum item support, MIS), johon alkion frekvenssin pitää yltää tietokannassa, jotta se toteuttaa minimialkiotukensa. Esimerkiksi harvoin esiintyvälle tuotteelle, kuten paistinpannulle, voimme antaa pienen minimialkiotuen, jolloin se saadaan mukaan assosiaatiosääntöihin. Paistinpannun minimialkiotueksi voimme antaa 1 % ja usein ostetulle tuotteelle, kuten maidolle, minimialkiotueksi voimme antaa 10 %. Näin otetaan huomioon näiden kahden tuotteen erilaiset ominaisuudet ja saadaan aikaan mielenkiintoisia assosiaatiosääntöjä. Minimialkiotuen määrittäminen jokaiselle alkiolle on tärkeää esimerkiksi kaupalle sen takia, että näillä harvemmin ostetuilla tuotteilla, joilla on usein suurempi kate, voidaan saada enemmän voittoa (Liu & al. 1999b).

Liu & al. (1999b) esittävät minimialkiotuen *MIS*(*i*) määräytyvän alkiolle *i* kaavan (3.8) mukaisesti. Kaavassa käytetään kahta parametria. Toinen niistä on käyttäjän antama *pienin tuki* (least support) *LS*. Pienin tuki on alin arvo, joka sallitaan alkion minimialkiotueksi. Toinen parametri β kontrolloi *MIS*-arvon suhdetta alkion frekvenssiin. Parametrin β arvo on nollan ja yhden väliltä ($0 \leq \beta \leq 1$). Alkion frekvenssi *sup*(*i*) kerrotaan parametrilla β ja tulosta verrataan pienimpään tukeen. Mikäli tulos ei ylitä parametria *LS*, annetaan alkion *MIS*-arvoksi pienin tuki. Jos parametrin β arvo on nolla, otetaan käyttöön yhtenäinen minimituki (Liu & al., 1999b).

$$MIS(i) = \begin{cases} M(i), M(i) > LS \\ LS, \text{muulloin} \end{cases} \qquad (3.8)$$

$$M(i) \quad = \beta\, sup(i)$$

Esimerkiksi alkiot 1, 2 ja 3, joiden frekvenssit ovat *sup*(1) = 2 %, *sup*(2) = 5 % ja *sup*(3) = 10 %, määräytyvät kaavan (3.8) mukaan seuraavalla tavalla. Annamme pienimmän tuen *LS* arvoksi 5 % ja parametrin β arvoksi 0,6. Kaavan mukaan *MIS*(1) = 5 % ja *MIS*(2) = 5 %, sillä alkioiden frekvenssit kerrottuna parametrin β arvolla alittavat pienimmän tuen arvon. Tämän johdosta näille alkioille annetaan minimialkiotueksi 5 %, sillä se on pienimmän tuen arvo. Sen sijaan kolmannen alkion *MIS*(3) = 6 %, sillä alkion frekvenssi kerrottuna parametrin β arvolla ylittää pienimmän tuen arvon.

Liu & al. (1999b) määrittelevät menetelmässään *säännön minimituen* uudelleen. Heidän menetelmässään säännön minimituki on säännössä esiintyvien alkioiden pienin minimialkiotuki. Tällöin sääntö $\{i_1, i_2, \ldots, i_k\} \Rightarrow \{i_{k+1}, \ldots, i_m\}$, jossa $i_j \in I$, ylittää säännön minimituen, mikäli säännön varsinainen tuki on suurempi tai yhtä suuri kuin $\min(MIS(i_1), MIS(i_2), \ldots, MIS(i_m))$.

Esimerkiksi tarkasteltaessa ruokakaupan tuotteita sinappi, pasta, ketsuppi ja salaatti voimme antaa niille minimialkiotuet siten, että harvoin esiintyvä tuote, kuten sinappi saadaan sääntöihin mukaan. Annamme tässä esimerkissä sinapille *MIS*-arvoksi 5 %,

MIS(sinappi) = 5 %, pastalle arvon *MIS*(pasta) = 20 %, ketsupille arvon *MIS*(ketsuppi) = 6 % ja salaatille arvon *MIS*(salaatti) = 10 %. Sääntö (3.9) ylittää nyt minimituen, sillä min(*MIS*(sinappi), *MIS*(pasta)) arvo on 5 % ja säännön varsinaiseksi tueksi oletetaan lasketuksi 6 %.

sinappi $\Rightarrow$ pasta tuki = 6 % uskottavuus = 70 % (3.9)

Liu & al. (1999b) kehittivät MSApriori-algoritmin (kuvat 3.13-3.15), joka käyttää alkioille annettuja minimialkiotukia. MSApriori on paranneltu versio Apriori-algoritmista, sillä MSApriori ottaa huomioon harvinaisten alkioiden ongelman. MSApriori-algoritmilla saadaan harvoin esiintyviä alkioita assosiaatiosääntöihin ilman, että sääntöjen lukumäärä kasvaa liikaa. Algoritmi perustuu Apriori-algoritmin tapaan leveyshakuun ja se käyttää aikaisempaa tietoa hyväksi kandidaattialkiojoukkoja muodostettaessa edellisen kierroksen usein esiintyvistä alkiojoukoista. Usein esiintyvät alkiojoukot löydetään karsimalla kandidaattialkiojoukkoja ja käymällä tietokantaa läpi useita kertoja. Algoritmin läpikäyntiä jatketaan niin kauan kuin löydetään usein toistuvia alkiojoukkoja. Algoritmille annetaan lähtötietoina alkiot *I*, alkioiden minimialkiotuet *MS* sekä tapahtumaperäinen tietokanta D.

MSApriori-algoritmissa kiinnitetään huomiota Apriori-algoritmin tapaan alaspäin suunnattuun sulkeumaan: usein esiintyvän alkiojoukon alijoukot ovat myös usein esiintyviä. Mikäli esimerkiksi alkiojoukon {salaatti, pasta} varsinainen tuki on 9 %, se ei ylitä säännön minimitukea eli kummankaan alkion *MIS*-arvoa olettaen, että *MIS*(salaatti) = 10 % ja *MIS*(pasta) = 20 %. Tällöin alaspäin suunnatun sulkeuman mukaan alkiojoukko {salaatti, pasta} poistetaan. Usein esiintyväksi alkiojoukoksi ei näin ollen saataisi myöskään alkiojoukkoa {salaatti, pasta, ketsuppi}, jonka alkioiden minimialkiotuki on vain 6 % olettaen, että *MIS*(ketsuppi) = 6 %. Ongelman ratkaisuun käytetään MSApriori-algoritmissa *järjestetyn sulkeuman ominaisuutta* (sorted closure property) (Liu & al., 1999b). Järjestetyn sulkeuman ominaisuuden mukaan alkiot järjestetään *MIS*-arvon mukaisesti nousevaan järjestykseen. Esimerkiksi sinapin, ketsupin, salaatin ja pastan tapauksessa sinapilla on pienin *MIS*-arvo ja pastalla suurin, jolloin järjestetyksi alkiolistaksi tulee <sinappi, ketsuppi, salaatti, pasta>.

MSApriori-algoritmin ensimmäisellä rivillä (kuva 3.13) järjestetään alkiot *I* *MIS*-arvonsa mukaan nousevaan järjestykseen listaan *M*. Tietokannan ensimmäisessä läpikäynnissä rivillä kaksi, lasketaan jokaisen listassa olevan alkion tuki. Järjestettyä listaa *M* käydään läpi ja pysähdytään, jos käsittelyssä olevan alkion tuki alittaa kohdalla olevan alkion minimitukiarvon. Käsittelyssä oleva alkio lisätään listaan *F*, jos alkion tuki toteuttaa alkion minimituen. Jos jokin listaan *F* lisättyä alkiota *i* seuraavan alkion *j* tuki listassa *M*, on suurempi tai yhtä suuri kuin *MIS*(*i*), myös alkio *j* lisätään listaan *F*. Usein esiintyvät 1-alkiojoukot, saadaan L_1-alkiojoukkoon listasta *F*, vertaamalla alkion tukea alkion minimialkiotukeen algoritmin rivin kolme mukaisesti. Ensimmäiset kandidaattialkiojoukot muodostetaan kierroksella $k = 2$ listasta *F* funktiolla *level2-candidate-gen* (kuva 3.14). Muilla kierroksilla, kun $k > 2$, kandidaattialkiojoukot tuotetaan funktiolla *candidate-gen* (kuva 3.15).

Input: Database, *D*, of transactions; minimum item supports, *MIS*(*i*), stored in *MS*; items, *I*.

Output: *L*, frequent itemsets in *D*.

Method:

```
(1)   M = sort(I, MS); // according to MIS(i)'s stored in MS
(2)   F = init-pass(M, D); // make the first pass over D
(3)   L1 = {<f> | f ∈ F, f.count ≥ MIS(f)};
(4)   for (k = 2; Lk-1 ≠ ϕ; k++) {
(5)       if k = 2 then
(6)           C2 = level2-candidate-gen(F);
(7)       else Ck = candidate-gen(Lk-1);
(8)       for each transaction t ∈ D {
(9)           Ct = subset(Ck, t);
(10)          for each candidate c ∈ Ct
(11)              c.count++;
(12)      }
(13)      Lk = {c ∈ Ck | c.count ≥ MIS(c[1])}
(14)  }
(15)  return L = ∪k Lk;
```

Kuva 3.13 MSApriori (Liu & al., 1999b).

Kierroksella $k = 2$ käytetään MSApriori-algoritmissa kandidaatti-2-alkiojoukkojen muodostamiseen funktiota *level2-candidate-gen* (kuva 3.14). Funktio käyttää parametrinaan funktion *init-pass* tuottamaa listaa F. Listaa F käytetään usein esiintyvän alkiojoukon L_1 sijasta sen takia, että listassa F esiintyy mahdollisesti alkioita, joiden tuki on suurempi tai yhtä suuri kuin järjestetyssä listassa M edellä olevan alkion *MIS*-arvo. Sen sijaan joukosta L_1 on karsittu jo alkioita, jotka alittavat minimialkiotukensa algoritmin rivin 3 mukaisesti (kuva 3.13). Kandidaatti-2-alkiojoukot muodostetaan C_2-kandidaattialkiojoukkoon listasta F siten, että liitos tehdään alkion f ja jokaisen sen jälkeen esiintyvän alkion h kanssa, mikäli alkioiden f ja h tuet tietokannassa ovat suurempia tai yhtä suuria kuin minimialkiotuki $MIS(f)$.

```
function level2-candidate-gen(F: items from M);
(1)   for each item f in F in the same order
(2)     if f.count ≥ MIS(f) then
(3)       for each item h in F that is after f
(4)         if h.count ≥ MIS(f) then
(5)           insert <f, h> into C2;
(6)   return C2;
```

Kuva 3.14 Kandidaattialkiojoukkojen luominen, kun $k = 2$ (Liu & al., 1999b).

Tarkastelemme seuraavaksi kandidaatti-2-alkiojoukkojen muodostusta MSApriori-algoritmilla Liun & al. (1999b) esimerkin mukaisesti. Käsittelemme alkioita 1, 2, 3 ja 4, joiden minimialkiotuet ovat $MIS(1) = 10$ %, $MIS(2) = 20$ %, $MIS(3) = 5$ % ja $MIS(4) = 6$ %. MSApriori-algoritmin ensimmäisen rivin mukaan järjestämme alkiot *MIS*-arvon perusteella nousevaan järjestykseen listaan <3, 4, 1, 2>. Tietokannassa on 100 tapahtumaa ja niistä 25 kohdistuu alkioon 2, 9 kohdistuu alkioon 1, 6 kohdistuu alkioon 3 ja 3 kohdistuu alkioon 4. Tietokannan ensimmäisessä läpikäynnissä saadaan selville lista F, johon kuuluu <3, 1, 2>. Alkio 4 ei yllä listaan F, sillä sen tuki tietokannassa on pienempi kuin järjestetyssä listassa sitä edellä olevan alkion 3 *MIS*-arvo, $3 < MIS(3) = 5$ %. Usein esiintyvässä alkiojoukossa L_1 on alkiojoukko {3, 2}. Alkiojoukkoon L_1 ei tule mukaan listan F alkiota 1, sillä sen tuki ei toteuta $MIS(1)$-

arvoa, joka on 10 %. Kandidaatti-2-alkiojoukon muodostamisessa käytetään parametrina listaa *F*. Saamme C_2-kandidaattialkiojoukkoon alkiojoukot {3, 1} ja {3, 2} listasta *F*, koska funktiossa tehdään liitos listassa *F* olevan alkion *f* ja jokaisen sitä seuraavan alkion *h* kanssa. Lisäksi alkioiden *f* ja *h* pitää toteuttaa alkion *f* *MIS*-arvo. Kandidaattialkiojoukkoon C_2 ei tule alkiojoukkoa {1, 2}, sillä alkion 1 tuki ei toteuta *MIS*(1)-arvoa.

MSApriori-algoritmin funktio c*andidate-gen* (kuva 3.15) muistuttaa Apriori-algoritmin funktiota *apriori-gen* (kuva 3.3). Funktion *candidate-gen* liitosoperaatio on samanlainen kuin Apriori-algoritmissa, mutta kandidaattialkiojoukkojen karsiminen on erilainen (Liu & al., 1999b). Liitosoperaation jälkeen voi C_k-kandidaattialkiojoukossa olla mukana alkiojoukkoja, jotka eivät voi olla usein esiintyviä. Mikäli kandidaattialkiojoukon *c* (*k*-1)-alijoukko *s* ei esiinny L_{k-1}:ssä, voidaan kandidaattialkiojoukko *c* poistaa (kuva 3.15, rivit 6-9). Kandidaattialkiojoukon poistamisessa otetaan kuitenkin huomioon tilanne, jolloin *s* ei sisällä alkiota *c*[1], jolla on pienin *MIS*-arvo. Tällöin kandidaattialkiojoukkoa *c* ei voida poistaa, koska ei olla varmoja, että *s* ei tyydytä ominaisuutta *MIS*(*c*[1]), paitsi jos *MIS*(*c*[1]) = MIS(*c*[2]). Oletuksena L_k sisältää *k*-alkiojoukot *c*: *c.count* ≥ *MIS*(*c*[1]) ≤ *MIS*(*c*[2]), kuten kuvan 3.13 algoritmin riviltä 13 käy ilmi.

```
function candidate-gen(L_{k-1}: frequent (k-1)-itemsets);
(1)   insert into C_k
(2)       select p.item_1, p.item_2,..., p.item_{k-1}, q.item_{k-1}
(3)       from L_{k-1} p, L_{k-1} q
(4)       where p.item_1 = q.item_1,..., p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}
(5)   for each itemset c ∈ C_k // Prune
(6)       for each (k-1)-subset s of c
(7)           if (c[1] ∈ s) or (MIS(c[2]) = MIS(c[1])) then
(8)               if (s ∉ L_{k-1}) then
(9)                   delete c from C_k;
(10)  return C_k;
```

Kuva 3.15. Kandidaattialkiojoukkojen luominen, kun *k* > 2 (Liu & al., 1999b).

Tarkastelemme seuraavaksi Liun & al. (1999b) esimerkkiä järjestetystä sulkeumasta. Mikäli L_3-alkiojoukossa on usein esiintyvät alkiojoukot {1, 2, 3}, {1, 2, 5}, {1, 3, 4}, {1, 3, 5}, {1, 4, 5}, {1, 4, 6}, {2, 3, 5} saadaan MSApriori-algoritmin *candidate-gen*-funktiolla (kuva 3.15) liitos, jossa ovat alkiojoukot {1, 2, 3, 5}, {1, 3, 4, 5} ja {1, 4, 5, 6}. Alkiojoukko {1, 4, 5, 6} karsiutuu C_4-kandidaattialkiojoukosta, sillä sen alijoukko {1, 5, 6} ei esiinny alkiojoukossa L_3. Sen sijaan kandidaattialkiojoukkoa c = {1, 3, 4, 5} ei karsita, vaikka sen alijoukko s = {3, 4, 5} ei esiinny L_3-alkiojoukossa. Kuvan 3.15 rivillä 7 tarkastetaan sisältääkö s alkiota c[1], jolla on pienin *MIS*-arvo. Koska alkiojoukko s ei sisällä alkiota c[1] eikä alkioilla c[1] ja c[2] ole samaa minimitukiarvoa, niin kandidaattialkiojoukkoa c ei poisteta.

Kandidaattialkiojoukkojen muodostuksen jälkeen selvitetään, ovatko kandidaattialkiojoukot usein esiintyviä alkiojoukkoja, päivittämällä kandidaattialkiojoukkojen tuet tapahtumittain (kuvan 3.13 rivit 8-12). Usein esiintyvät alkiojoukot tietokannassa saadaan kandidaattialkiojoukoista vertaamalla niiden tukea arvoon $MIS[c(1)]$ (kuvan 3.13 rivi 13), joka on säännön minimituki. Näistä usein esiintyvistä alkiojoukoista voidaan assosiaatiosääntöjen johtamisen toisen vaiheen mukaisesti muodostaa assosiaatiosääntöjä.

Yun & al. (2003) huomauttavat, että Liun & al. (1999b) *MIS*-mallissa on kaksi ongelmaa. Ensimmäinen ongelma on se, että MSApriori vaatii tapahtumasarjan, jolla *MIS*-arvot asetetaan alkioille. Toinen ongelma on siinä, että löydetyt säännöt riippuvat parametrin β arvosta, jota käytetään minimialkiotuen määrittämisessä kaavassa (3.8). Vaikka MSApriori-algoritmissa otetaan huomioon alkion frekvenssi tietokannassa, suurempi merkitys on kuitenkin parametrin β arvolla.

Wang & al. (2003) luonnehtivat minimialkiotuki-menetelmää luonnottomaksi kolmesta syystä. Ensimmäiseksi epäkohdaksi he mainitsevat sen, että yksittäisten alkioiden minimialkiotuet kuvaavat määrittelyhetkellä tuntemattomien alkiojoukkojen minimitukia. Wang & al. (2003) antavat kritiikkiä myös siitä, että joissakin sovelluksissa minimituki voi olla yksittäisenä käsitteenä tietylle alkiojoukolle eikä alkiojoukon yksittäiselle alkiolle. Yksittäinen käsite Suomen oloissa voi olla esimerkik-

si alkiojoukko {suomalainen, mies}, jolloin *minimialkiojoukkotuki* (minimum itemset support) on tavallista pienempi kuin alkiojoukon alkioiden minimialkiotuet. Viimeiseksi he kritisoivat sitä, että kahdelle eri alkiojoukolle ei voida antaa erillisiä minimitukia, mikäli alkiojoukossa esiintyy pienen minimialkiotuen omaava yleinen alkio. Yleinen alkio Suomen oloissa voi olla esimerkiksi {lukutaitoinen}. Tällöin alkiojoukot {lukutaitoinen, mies} ja {lukutaitoinen, mies, isä} ovat riippuvaisia tästä pienen minimialkiotuen omaavasta alkiosta.

3.4 Suhteellinen tuki

Kun kohdissa 3.2 ja 3.3 kiinnitettiin assosiaatiosääntöjen johtamisessa huomiota harvinaisten alkioiden ongelmaan, pyritään Yunin & al. (2003) *suhteelliseen tukeen* (relative support) perustuvassa menetelmässä löytämään assosiaatiosääntöihin etenkin merkittävää, harvinaista dataa.

Yunin & al. (2003) menetelmässä etsitään tietokannasta merkittävää, harvinaista dataa epäyhtenäisellä minimituella. Merkittävän, harvinaisen datan alkiojoukoista voidaan assosiaatiosääntöjen johtamisen toisen vaiheen mukaisesti muodostaa assosiaatiosääntöjä. Alkiojoukkoja etsitään epäyhtenäisellä minimituella, sillä käytössä on kaksi käyttäjän määrittämää erillistä minimituki-parametria, $support_1$ ja $support_2$ sekä tietokannan alkioiden frekvenssiin perustuva suhteellinen tuki *Rsup* (kuva 3.16).

Yunin & al. (2003) menetelmässä käytetään kahta minimituki-parametria, joiden avulla saadaan tietokannasta esille kaksi erilaista kandidaattialkiojoukkoa. Minimituki-parametrin $support_1$ arvon tulee olla suurempi kuin minimituki-parametrin $support_2$ arvon, koska parametria $support_1$ käytetään tietokannassa usein esiintyvien alkiojoukkojen etsimisessä ja parametria $support_2$ käytetään tietokannassa harvoin esiintyvän datan löytämiseksi. Minimituki-parametrille $support_1$ voidaan antaa arvoksi esimerkiksi 40 %. Minimituki-parametrin $support_2$ arvoksi voidaan antaa esimerkiksi 15 %.

Suhteellisen tuen *Rsup* avulla saadaan harvoin esiintyvästä datasta esille merkittävää,

TUKI	SELITYS
$support_1$	Minimituki usein esiintyville alkiojoukoille. Tuki on arvoltaan suurin verrattuna tukiin *Rsup* ja $support_2$.
Rsup	Suhteellinen tuki merkittävälle, harvinaiselle datalle. Tuki on arvoltaan tukien $support_1$ ja $support_2$ välissä.
$support_2$	Minimituki harvoin esiintyvälle datalle.

Kuva 3.16 Käytetyt tuet.

harvinaista dataa. Suhteellinen tuki saadaan lasketuksi kandidaattialkiojoukosta $\{i_1, i_{2,}..., i_m\}$ kaavalla (3.10), jolloin ($0 \leq Rsup \leq 1$). Kaavassa (3.10) $sup(i)$ merkitsee alkion i tukea tietokannassa. Tulokseksi saadaan kandidaattialkiojoukon suhteellinen tuki, kun verrataan alkiojoukon tukea joukon jokaisen alkion tukeen ja valitaan näistä suurin. Suhteellinen tuki mittaa datan suhteellisen frekvenssin lisäksi alkiojoukon uskottavuutta. Suhteellisen tuen tulee toteuttaa käyttäjän määrittelemä *minimi suhteellinen tuki* (the minimum relative support, *minRsup*).

$$Rsup\{i_1, i_{2,}..., i_m\} = \max(sup\{i_1, i_{2,}..., i_m\} / sup\{i_1\}, sup\{i_1, i_{2,}..., i_m\} / sup\{i_2\}, \ldots, sup\{i_1, i_{2,}..., i_m\} / sup\{i_m\}) \quad (3.10)$$

Tarkastelemme seuraavaksi suhteellisen tuen määrittämistä kandidaattialkiojoukolle $\{1, 2\}$. Oletetaan, että minimi suhteellinen tuki *minRsup* = 0,6 ja kandidaattialkiojoukon tuki on 6 % tietokannassa D. Kandidaattialkiojoukon suhteelliseksi tueksi saadaan *Rsup* = 1, kun tietokannassa on 100 tapahtumaa ja kuusi niistä kohdistuu alkioon 1 ja 25 kohdistuu alkioon 2. Suhteellinen tuki saadaan siis säännöllä (3.10), kun kandidaattialkiojoukon tuki jaetaan molempien alkioiden tuella ja valitaan näistä vaihtoehdoista (6/6 ja 6/25) suurempi. Kandidaattialkiojoukon suhteellinen tuki ylittää vaaditun minimi suhteellisen tuen *minRsup*.

Yun & al. (2003) ovat kehittäneet suhteellisen tuen Apriori-algoritmin (Relative Support Apriori Algorithm, RSAA) (kuvat 3.17-3.19) assosiaatiosääntöjen johtamista varten. Algoritmille RSAA annetaan lähtötietoina tapahtumaperäinen tietokanta D, tietokannan alkiot *I*, kaksi minimituki-parametria, $support_1$ ja $support_2$ sekä minimi suhteellinen tuki *minRsup*. Algoritmin suorituksessa etsitään kolmea alkiojoukkoa, L_k, NLL_k ja NL_k kuvan 3.20 mukaisesti. Algoritmin riveillä 1-5 (kuva 3.17) muodostetaan

alkiojoukot L_1 ja NL_1, joista alkiojoukkojen NLL_k ja NL_k muodostaminen aloitetaan. Alkiojoukko L_1 koostuu ensimmäisen minimituki-parametrin $support_1$ toteuttavista alkioista ja NL_1 koostuu toisen minimituki-parametrin $support_2$ toteuttavista alkioista. Alkiojoukon L_k muodostamisessa (kuvan 3.17 rivi 6) käytetään Apriori-algoritmia (kuva 3.1). Alkiojoukko L_k sisältää tietokannassa usein esiintyviä alkiojoukkoja ja ne toteuttavat algoritmille annetun ensimmäisen minimituki-parametrin $support_1$.

Input: Database, D, of transactions; items, I; minimum support threshold, $support_1$;
minimum support threshold, $support_2$; minimum relative support, $minRsup$.
Output: LL, quasi-frequent itemset in D.
Method:

(1) **for** (all $i_k \in I$)
(2) **if** (i_k.support $\geq support_1$) **then**
(3) $i_k \in L_1$ **where** $L_1 = C_1$
(4) **else if** (i_k.support $\geq support_2$) **then**
(5) $i_k \in NL_1$ **where** $NL_1 = NC_1$
(6) L_k = Apriori(D, $support_1$);
(7) **if** ($k = 2$) {
(8) NC_2 = rsaa-gen(NL_1, NL_1);
(9) NLC_2 = rsaa-gen(NL_1, L_1);
(10) NL_2 = significant_rare_data(D, $support_2$, NC_2, $minRsup$);
(11) NLL_2 = significant_rare_data(D, $support_2$,NLC_2, $minRsup$);
(12) }
(13) **for** ($k = 3$; $NL_{k-1} \neq \phi$ **or** $NLL_{k-1} \neq \phi$; k++) {
(14) NC_k = rsaa-gen(NL_{k-1}, NL_{k-1});
(15) NLC_k = rsaa-gen(NLL_{k-1}, NLL_{k-1});
(16) NL_k = significant_rare_data(D, $support_2$, NC_k, $minRsup$);
(17) NLL_k = significant_rare_data(D, $support_2$, NLC_k, $minRsup$);
(18) }
(19) **return** $LL = \cup_k L_k + \cup_k NL_k + \cup_k NLL_k$;

Kuva 3.17 RSAA-algoritmi.

RSAA-algoritmilla löydetään *näennäisesti usein esiintyvät alkiojoukot* (quasi-frequent itemset) yhdistämällä alkiojoukot NL_k ja NLL_k. Nämä alkiojoukot muodostetaan kandidaattialkiojoukoista NC_k ja NLC_k. Kandidaattialkiojoukot muodostetaan kuvan 3.18 pelkistetyllä funktiolla *rsaa-gen*. RSAA-algoritmin ensimmäisellä kierroksella (kuvan 3.17 rivit 1-5) etsitään usein esiintyvä alkiojoukko L_1 ja harvoin esiintyvä alkiojoukko NL_1. Näitä alkiojoukkoja L_1 ja NL_1 käytetään *rsaa-gen*-funktiossa, kun $k = 2$ (kuvan 3.18 rivit 1-5). Tällöin tehdään NLC_2-kandidaattialkiojoukko usein esiintyvien 1-alkiojoukkojen L_1 ja NL_1 liitoksella sekä NC_2-kandidaattialkiojoukko alkiojoukkojen NL_1 liitoksella.

```
function  rsaa-gen();
 (1)    if (k = 2) then
 (2)        insert into NC2
 (3)            select p.item1, q.item1 from NL1p, NL1q
 (4)        insert into NLC2
 (5)            select p.item1, q.item1 from NL1p, L1q
 (6)    else
 (7)        insert into NCk
 (8)            select p.item1, p.item2,…, p.itemk-1, q.itemk-1 from NLk-1p, NLk-1q
 (9)            where p.item1 = q.item1, p.item2 = q.item2,…, p.itemk-2 = q.itemk-2,
 (10)           p.itemk-1 < q.itemk-1
 (12)       insert into NLCk
 (13)           select p.item1, p.item2,…, p.itemk-1, q.item k-1 from NLLk-1p, NLLk-1q
 (14)           where p.item1 = q.item1, p.item2 = q.item2,…, p.itemk-2 = q.itemk-2,
 (15)           p.item k-1 < q.itemk-1
 (16)    return NCk or NLCk;
```

Kuva 3.18 Kandidaattialkiojoukkojen luominen.

Kun $k > 2$, muodostetaan kandidaattialkiojoukot NC_k ja NLC_k käyttämällä hyväksi aikaisempaa tietoa, kun tehdään liitos edellisen kierroksen (k-1)-alkiojoukosta funktion *rsaa-gen* rivien 7-15 mukaisesti (kuva 3.18). Kandidaattialkiojoukko NC_k sisältää (k-1)-alkiojoukkoja, jotka toteuttavat toisen minimituki-parametrin sekä suhteellisen tu-

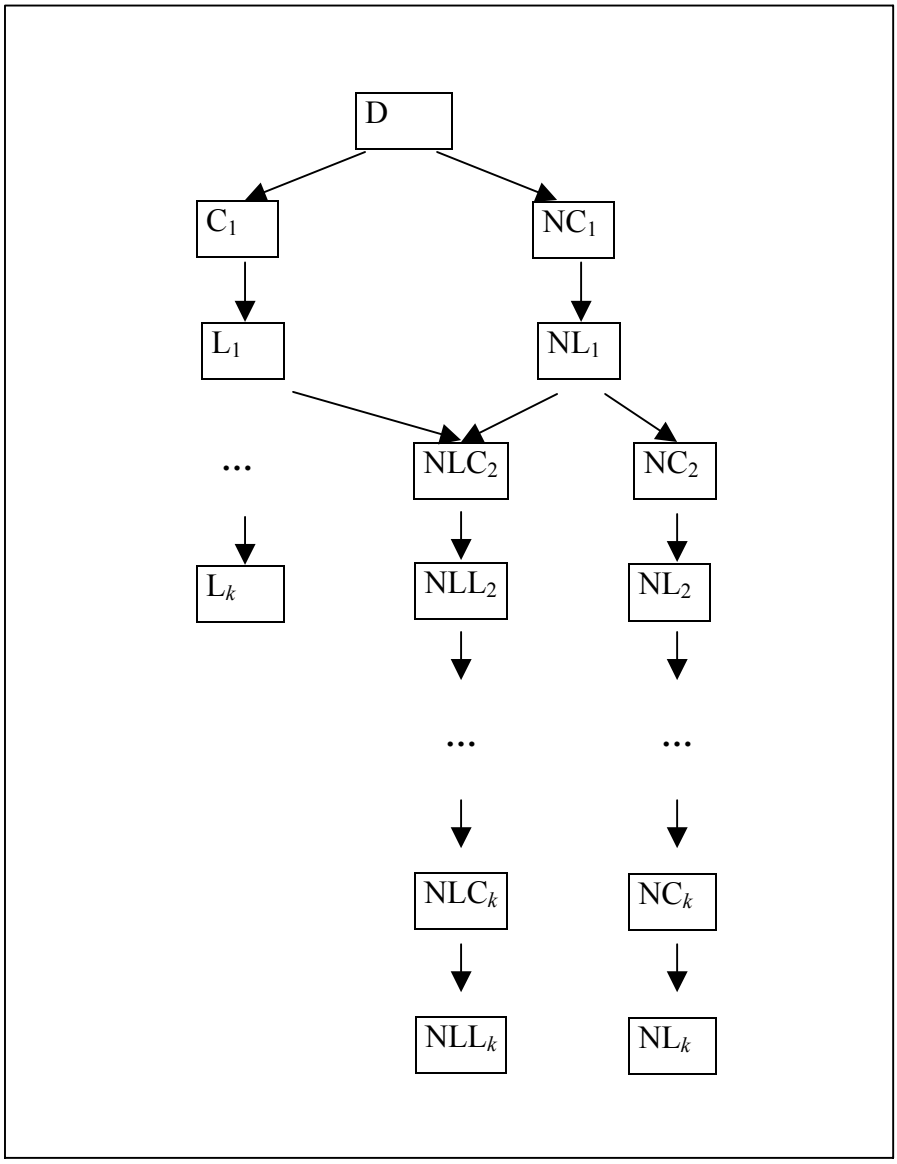
en. Kandidaattialkiojoukko NLC_k sisältää alkiojoukkoja, jotka koostuvat minimi suhteellisen tuen toteuttavista (k-1)-alkiojoukoista, ensimmäisen minimituki-parametrin ylittävistä alijoukoista sekä toisen minimituki-parametrin ylittävistä alijoukoista.

Kuvan 3.18 *rsaa-gen*-funktiolla muodostetuista kandidaattialkiojoukoista NC_k ja NLC_k tuotetaan alkiojoukot NL_k ja NLL_k funktiolla *significant_rare_data* (kuva 3.19).

```
function significant_rare_data(D: Database, D of transactions; support2:
         second minimum support threshold; c_itemset: candidate itemset;
         minRsup: minimum relative support);
(1)    SRD_temp = {};
(2)    for all transactions t ∈ D {
(3)       SRD_t = {}; // SRD_t = {nc|nc ∈ c_itemset and (all data items nc accessed by t)}
(4)       for all nc in c_itemset {
(5)          if (all data items in nc are accessed by t)
(6)             SRD_t = SRD_t ∪ nc;
(7)          for all candidates nc ∈ SRD_t
(8)             nc.count++;
(9)       }
(10)      SRD_temp = SRD_temp ∪ SRD_t;
(11)   }
(12)   for all nc in SRD_temp {
(13)      if nc.count ≥ support2 then
(14)         for each item i_k in nc // count relative support
(15)            nc.Rsup = max(sup(i1, i2,…, im) / sup(i1), sup(i1, i2,…, im) / sup(i2),
(16)            …
(17)            sup(i1, i2,…, im) / sup(im));
(18)      if nc.Rsup ≥ minRsup then
(19)         SL_k = {nc|nc ∈ SRD_temp and nc.Rsup ≥ minRsup}
(20)   }
(21)   return SL_k;
```

Kuva 3.19 Merkittävän harvinaisen datan muodostuminen.

Funktiolle annetaan parametreina tapahtumaperäinen tietokanta, toinen minimituki-parametri, kandidaattialkiojoukko sekä minimi suhteellinen tuki. Funktiossa käydään tietokanta läpi tapahtumittain (rivit 1-11) ja lasketaan kandidaattialkiojoukoille *nc* niiden tuet (rivit 7-8). Tukia verrataan toiseen minimituki-parametriin (rivi 13) ja tämän jälkeen kandidaattialkiojoukoille lasketaan suhteelliset tuet (rivit 14-17). Funktio palauttaa alkiojoukon SL_k, joka toteuttaa toisen minimituen sekä minimi suhteellisen tuen (rivit 12-20). Alkiojoukko SL_k koostuu näin ollen merkittävästä, harvinaisesta datasta. Algoritmin läpikäyntiä jatketaan niin kauan kuin löydetään alkiojoukkoja NL_k ja NLL_k.



Kuva 3.20 RSAA-algoritmin suoritus.

Tarkastelemme seuraavaksi RSAA–algoritmin suoritusta Yunin & al. (2003) kuvaaman esimerkin avulla. Kuvassa 3.21 on tapahtumaperäinen tietokanta sekä siinä esiintyvien alkioiden tuet. Annamme RSAA-algoritmille ensimmäiseksi minimitueksi 40 % ja toiseksi minimitueksi 20 % ja minimi suhteelliseksi tueksi 0,7. Kuvasta 3.21 huomaamme, että alkiot 5 ja 7 eivät toteuta ensimmäistä tukea 40 %, mutta ne ylittävät toisen tuen 20 %.

TID	ALKIOT
1	{1,2,3}
2	{3,4}
3	{1,2,3,4,5,6,7}
4	{6,7}
5	{3,4,5,6}
6	{1,2}
7	{3,4,6,7}
8	{1,2}
9	{1,3,4,5,6}
10	{1,2}

ALKIO	TUKI
1	6
2	5
3	6
4	5
5	**3**
6	5
7	**3**

Kuva 3.21 Tapahtumaperäinen tietokanta ja alkioiden tuet.

Kuvan 3.21 alkioista saamme RSAA-algoritmilla (kuva 3.17) usein esiintyväksi 1-alkiojoukoksi L_1 alkiojoukot {1}, {2}, {3}, {4} ja {6} sekä harvoin esiintyviksi alkiojoukoiksi {5} ja {7} alkiojoukkoon NL_1. Näistä joukoista saamme muodostettua kandidaatti-2-alkiojoukot NLC_2 ja NC_2. Kandidaattialkiojoukko NC_2 muodostetaan toisen tuen ylittävien alkiojoukkojen liitoksella $NL_1 \bowtie NL_1$, jolloin saadaan kandidaattialkiojoukoksi {5, 7}. Kandidaattialkiojoukko NLC_2 muodostetaan liittämällä toisen tuen toteuttavat alkiot ja ensimmäisten tuen ylittävät alkiot toisiinsa $NL_1 \bowtie L_1$, jolloin saadaan NLC_2-alkiojoukkoon alkiojoukot {5, 1}, {5, 2}, {5, 3}, {5, 4}, {5, 6}, {7, 1}, {7, 2}, {7, 3}, {7, 4}, {7, 6}. Jotta saadaan selville alkiojoukot NL_2 ja NLL_2, RSAA-algoritmin suorituksessa lasketaan kandidaattialkiojoukoille frekvenssit ja suhteelliset

tuet. Minimituki-parametrin $support_2$ ja minimi suhteellisen tuen toteuttavat kandidaattialkiojoukot muodostavat alkiojoukot NL_2 ja NLL_2. Suhteellinen tuki on esimerkiksi alkiojoukolle {5, 7} 0,33, kun alkiojoukon tuki 1 jaetaan molempien alkioiden tuella ja valitaan näistä vaihtoehdoista (1/3 ja 1/3) suurempi. Alkiojoukon {5, 7} suhteellinen tuki ei ylitä minimi suhteellista tukea eikä sen frekvenssi ylitä toista minimitukea. Alkiojoukko NL_2 on siis tyhjä. Sen sijaan kandidaattialkiojoukon NLC_2 alkiojoukkojen {3, 5},{4, 5}, {5, 6} ja {6, 7} suhteellinen tuki on 1 ja ne ylittävät minimi suhteellisen tuen. Kandidaattialkiojoukon NLL_2 muodostavat alkiojoukot {3, 5}, {4, 5}, {5, 6}, {6, 7}, koska ne ylittävät suhteellisen tuen lisäksi myös toisen minimituen.

Kun $k = 3$, muodostuu NLC_3-kandidaattialkiojoukko. Kandidaattialkiojoukko saadaan muodostettua edellisen kierroksen alkiojoukosta NLL_2, kun alkiojoukkojen muodostamisessa käytetään liitosta $NLL_2 \bowtie NLL_2$. Kandidaattialkiojoukoksi NLC_3 saadaan tällöin alkiojoukot {3, 4, 5}, {3, 5, 6}, {5, 6, 7} ja {4, 5, 6}. Vertaamalla kandidaattialkiojoukkojen tukia toiseen minimitukeen ja suhteellisia tukia minimi suhteelliseen tukeen saamme NLL_3-alkiojoukon. Alkiojoukko koostuu alkiojoukoista {3, 4, 5}, {3, 5, 6} ja {4, 5, 6}. Algoritmin suoritusta jatketaan niin kauan kuin alkiojoukkoja löydetään.

Yun & al. (2003) päättelevät kuvan 3.21 esimerkistä, että alkiojoukot {3, 4, 5}, {3, 5, 6} ja {4, 5, 6} ovat merkittävää harvinaista dataa, sillä alkio 5 ei toteuta ensimmäistä tukea, mutta esiintyy tietokannassa usein samojen alkioiden kanssa. Alkio 5 toteuttaa toisen tuen, jolloin se saadaan mukaan harvinaiseen dataan ja edelleen RSAA-algoritmin suoritukseen. Apriori-algoritmilla tätä alkiota ei huomioitaisi lainkaan, mikäli tuki olisi esimerkiksi 40 %. Sen sijaan Liun & al. (1999b) kehittämällä minimialkiotuen menetelmällä saataisiin tämä alkio mukaan assosiaatiosääntöihin, jos sille annettaisiin pieni minimialkiotuki.

3.5 Tukirajoitteiden käyttö assosiaatiosääntöjä muodostettaessa

Wang & al. (2003) ovat kehittäneet epäyhtenäiseen minimitukeen perustuvan menetelmän assosiaatiosääntöjen etsimiseksi. Heidän menetelmässään annetaan epäyhteinen minimituki-parametri kokonaiselle alkiojoukolle. Annettu minimituki riippuu siitä, mitä alkioita alkiojoukko sisältää. *Tukirajoitteet* (support constraint, SC) määräävät, mitkä alkiojoukot vaativat kulloisenkin minimituen. Minimituki siis vaihtelee alkiojoukosta riippuen, koska jokaiselle alkiojoukolle annetaan sen vaatima minimituki. Tukirajoitteiden käyttäminen alkiojoukkojen tuottamisessa karsii kandidaattialkiojoukkoja.

Tukirajoitteita ei anneta jokaiselle alkiojoukolle erikseen. Tukirajoite esitetään muodossa $SC_i(B_1,\ldots, B_m) \geq \theta_i$, jossa $m \geq 0$ ja θ_i on minimituki nollan ja yhden väliltä tai funktio, joka tuottaa minimituen. Jokainen B_j on *lokero* (bin), joka sisältää alkioita $\{i_1, i_2,\ldots, i_m\}$. Näillä alkioilla $i_j \in B_j$ on yhtenäinen minimituki. Joukkoa $\gamma = \{B_1, B_2,\ldots, B_m\}$ sanotaan alkiojoukkojen $\{i_1, i_2,\ldots, i_m\}$ *skeemaksi* (schema).

Wang & al. (2003) jakavat tukirajoitteiden tulkinnan kahteen luokkaan, avoimeen tulkintaan (open interprepation) ja suljettuun tulkintaan (closed interprepation). *Avoimessa tulkinnassa* kandidaattialkiojoukko C_k vastaa tukirajoitetta SC_i, kun alkiojoukko sisältää ainakin yhden alkion jokaisesta tukirajoitteen lokerosta B_j. *Suljetussa tulkinnassa* kandidaattialkiojoukko C_k vastaa tukirajoitetta, mikäli alkiojoukko sisältää erillisen alkion jokaisesta tukirajoitteen SC_i lokerosta eikä alkiojoukossa ole muita alkioita. Tukirajoite $SC_0()$ on *oletusminimituki*, jota käytetään, kun mikään muu tukirajoite ei sovellu alkiojoukolle. Kandidaattialkiojoukon C_k minimituki, $min\_sup(C_k)$, on pienin tukirajoite, joka vastaa alkiojoukkoa. Kandidaattialkiojoukko C_k on usein esiintyvä, jos sen tuki $sup(C_k)$ toteuttaa alkiojoukon minimituen $sup(C_k) \geq min\_sup(C_k)$ (Wang & al., 2003).

Käymme seuraavaksi läpi Wangin & al. (2003) esimerkin tukirajoitteiden käytöstä kandidaattialkiojoukolle $\{b_0, b_1, b_2, b_3, b_4\}$, jossa $b_i \in B_i$. Huomaamme että tukirajoitteiden avoimella tulkinnalla alkiojoukko vastaa sekä tukirajoitetta $SC_1(B_1, B_2) \geq 0{,}1$ että tukirajoitetta $SC_2(B_3, B_4) \geq 0{,}2$. Kandidaattialkiojoukon $\{b_0, b_1, b_2, b_3, b_4\}$ tueksi

tietokannassa oletetaan 0,15. Tukirajoitteeksi valitaan joko SC_1 tai SC_2. Jos tukirajoitteeksi valitaan SC_2, niin kandidaattialkiojoukko ei ole usein esiintyvä. Wangin & al. (2003) mukaan tukirajoitteeksi valitaan vaihtoehdoista pienin, jolloin kandidaattialkiojoukon tuki ylittää minimituen ja on usein esiintyvä.

Wang & al. (2003) esittelevät neljä eri menetelmää, miten käyttäjä voi luokitella alkiot lokeroihin. Menetelmät perustuvat alkioiden tukeen, käsitteistöön, attribuutteihin sekä luettelointiin. Wang & al. (2003) käyttävät omassa tutkimuksessaan *tuen mukaista määrittelyä* (support-based specification). Tuen mukaisessa määrittelyssä käydään tietokanta läpi ja alkioiden tuet lasketaan tietokannassa. Alkiot, joilla on samantasoiset tuet, voidaan ryhmitellä samaan lokeroon. Lokeron tukirajoite θ_i voidaan määritellä funktiolla, jossa käytetään hyväksi lokeron alkioiden maksimi-, minimi- tai keskiarvotukea. Toinen menetelmä luokitella alkiot lokeroihin on *käsitepohjainen määrittely* (concept-based specification). Käsitepohjaisessa määrittelyssä alkioiden oletetaan olevan kuvan 2.3 kaltaisessa käsitehierarkiassa, jolloin minimituen määrittelyssä otetaan huomioon käsitehierarkian tasot ja alitasojen lukumäärät. Kolmas Wangin & al. (2003) mainitsema menetelmä alkioiden lokerointiin on *attribuuttien mukainen määrittely* (attribute-based specification). Attribuuttien mukaisessa määrittelyssä käytetään hyväksi relaatiotietokantaa. Tällöin lokeroon voidaan ryhmitellä tietokantataulun yhteenkuuluvia tietoja. Minimituen määrittelyssä otetaan huomioon monikkojen määrä tietokannassa. Wang & al. (2003) korostavat viimeisen menetelmän, *luettelointipohjaisen määrittelyn* (enumeration-based specification) olevan joustavin menetelmä näistä neljästä vaihtoehdosta. Luettelointipohjaisessa määrittelyssä luetteloidaan alkiot lokeroihin niiden yhteenkuuluvuuden perusteella. Käyttäjä voi luetteloida lokeroon haluamansa alkiot, kuten esimerkiksi maitotuotteet lokeroon $B_1 = \{\text{maito, juusto}\}$. Luetteloinnilla käyttäjä voi rajata mielenkiintoiset alkiot pieneen lokeroon sen sijaan, että lokero sisältäisi paljon alkioita.

Tarkastelemme seuraavaksi tukirajoitteiden käyttöä Wangin & al. (2003) esimerkin mukaisesti. Käytämme esimerkissä tukirajoitteiden avointa tulkintaa. Kuvassa 3.22 on tapahtumaperäinen tietokanta, lokerot sekä lokeroiden tukirajoitteet. Kandidaattialkiojoukko C_k, jolla on alkioita sekä lokerosta B_1 että lokerosta B_3 toteuttaa tukirajoitteet SC_1 ja SC_2. Alkiojoukon C_k minimitueksi $min\_sup(C_k)$ määräytyy 0,2 sillä se

on pienempi näistä kahdesta vaihtoehdosta. Esimerkiksi alkiojoukko {2, 3} toteuttaa tukirajoitteet SC_1 ja SC_2, sillä alkio 2 on lokerosta B_1 ja alkio 3 on lokerosta B_3.

TID	ALKIOT
1	{0,2,7}
2	{0,4,7,8}
3	{2,4,5,7,8}
4	{1,2,4,7,8}
5	{2,4,6,7,8}

LOKERO	ALKIOT
B_0	{1,7,8}
B_1	{2,6}
B_2	{4,5}
B_3	{0,3}

TUKIRAJOITE
$SC_3(B_2) \geq 0{,}6$
$SC_1(B_1, B_3) \geq 0{,}2$
$SC_2(B_3) \geq 0{,}4$
$SC_0() \geq 0{,}8$

Kuva 3.22 Tietokanta ja tukirajoitteet lokeroineen (Wang & al., 2003).

Wangin & al. (2003) menetelmässä kandidaattialkiojoukkoja luodaan kuten Apriori-algoritmissa. Tukirajoitteiden minimitukea käytetään, jotta saadaan kandidaattialkiojoukoista suodatettua usein esiintyviä alkiojoukkoja. Tästä menetelmästä Wang & al. (2003) käyttävät nimitystä adaptiivinen Apriori (Adaptive Apriori). Wang & al. (2003) painottavat, että heidän menetelmässään jokaiselle alkiojoukolle annettu minimituki määrätään yksilöllisesti. Adaptiivinen Apriori käyttää epäyhtenäistä minimituki-parametria ja se säilyttää kuitenkin Apriori-algoritmin olemuksen. Adaptiivinen Apriori voisi mielestäni olla kuvan 3.23 mukainen vaikkakin Wang & al. (2003) esittävät tarkemman toteutuksen.

Kuvan 3.23 toteutuksessa käytetään alkioiden lokeroinnissa luettelointipohjaista määrittelyä ja käytetään avointa tulkintaa. Algoritmille annetaan lähtötietoina tapahtumaperäinen tietokanta, käyttäjän luetteloimat alkiot lokeroissa, käyttäjän asettamat tukirajoitteet sekä minimituet tukirajoitteille. Algoritmin suorituksessa järjestetään ensin tukirajoitteet nousevaan järjestykseen minimituen perusteella (rivi 1), koska kandidaattialkiojoukolle annetaan pienin minimituki. Tämän jälkeen haetaan tietokannasta usein esiintyvät 1-alkiojoukot L_1-alkiojoukkoon funktiolla *frequent_itemset* (kuva 3.24). Algoritmin rivillä 4 muodostetaan C_k-kandidaattialkiojoukot edellisen kierroksen usein esiintyvistä alkiojoukoista L_{k-1}. Kandidaattialkiojoukoista karsitaan ne alkiojoukot, jotka eivät toteuta tukirajoitettaan. Algoritmin suoritusta jatketaan, kunnes usein esiintyviä alkiojoukkoja ei enää löydy.

Input: Database, D, of transactions; B: bins of items; SC: support constraints; θ:minimum support thresholds for SC.

Output: L, frequent itemsets in D.

Method:

(1) $sortedSC$ = sort(SC, θ); // sorted support constraints according to θ

(2) L_1 = frequent_itemset(D, B, D, $sortedSC$, θ);

(3) **for** (k = 2; $L_{k-1} \neq \phi$; k++) {

(4) C_k = apriori-gen(L_{k-1}); // get candidate itemsets

(5) L_k = frequent_itemset(C_k, B, D, $sortedSC$, θ);

(6) }

(7) **return** $L = \cup_k L_k$;

Kuva 3.23 Mukailtu Adaptiivinen Apriori

Usein esiintyvät alkiojoukot tuotetaan funktion *frequent_itemset* (kuva 3.24) karsimista kandidaattialkiojoukoista. Funktio saa parametreina kandidaattialkiojoukon, käyttäjän luetteloimat alkiot lokeroissa, tapahtumaperäisen tietokannan, käyttäjän asettamat tukirajoitteet suuruusjärjestyksessä sekä minimituet tukirajoitteille. Funktio käy tietokannan läpi tapahtumittain ja kandidaattialkiojoukoille lasketaan tuet. Alkiojoukkoon *bin_c* muunnetaan kandidaattialkiojoukon jokaisen alkion numero edustamaan lokeron numeroa, jolloin esimerkiksi kandidaattialkiojoukosta $\{b_0, b_1, b_2\}$ saadaan $bin\_c = \{B_1, B_2, B_3\}$, kun $b_i \in B_i$. Tämän jälkeen käydään järjestetyt tukirajoitteet läpi ja pysähdytään, jos kohdalla oleva tukirajoite vastaa alkiojoukon *bin_c* lokeroita. Tämän jälkeen haetaan tukirajoitteen minimituki ja verrataan sitä kandidaattialkiojoukon tukeen. Mikäli kandidaattialkiojoukon tuki toteuttaa tukirajoitteen minimituen, se lisätään usein esiintyviin alkiojoukkoihin. Jos kandidaattialkiojoukolle ei löydetä tukirajoitetta, sille sovelletaan oletusminimitukea.

Tarkastelemme seuraavaksi kuvan 3.24 mukailtua algoritmia esimerkin avulla. Annetaan algoritmille lähtötietoina tapahtumaperäinen tietokanta sekä käyttäjän luetteloimat lokerot B_1, B_2, B_3 ja B_4, $B_1 = \{i_1, i_2\}$, $B_2 = \{i_3,..., i_{20}\}$, $B_3 = \{i_{21}, i_{22}\}$, $B_4 = \{i_{23},..., i_{25}\}$ ja $B_4 = \{i_{26},..., i_m\}$. Alkiot lokerossa B_1 voisivat merkitä alkiojoukkoa {moottoripyörä, kevytmoottoripyörä} ja lokero B_2 voisi edustaa esimerkiksi vapaa-

ajan kaupan kaikkia muita moottoripyörätarvikkeita. Tukirajoitteeksi annetaan algoritmille käyttäjän määräämät tukirajoitteet näistä neljästä lokerosta $SC_1(B_1, B_2) = 0{,}1$ ja $SC_2(B_3, B_4) = 0{,}3$, $SC_0() = 0{,}4$. Kun tukirajoitteen SC_1 minimituki on 0,1, on käyttäjä kiinnostunut lähes kaikista säännöistä, joista selviää mitä tuotteita pyörien kanssa ostetaan.

```
function frequent_itemset(C_k: candidate itemset; B: bins of items; Database, D, of
        transactions; sortedSC: sorted support constraints; θ: minimum support thresholds
        for SC);
(1)   for each transaction t ∈ D { // scan D for counts
(2)     C_t = subset(C_k, t); // get the subsets of t that are candidates
(3)     for each candidate c ∈ C_t
(4)       c.count++;
(5)       bin_c = transform_itemset_to_bin_numbers(c, B); // compares items to bins
(6)       match = false;
(7)       for each support_constraint sc ∈ sortedSC { // scan the sortedSC
(8)         match = match_for_support_constraint(bin_c,sc);
(9)         if match then
(10)          L_k = {c ∈ C_k | c.count ≥ θ.sc_sup}
(10)      }
(11)      if not match then
(12)        L_k = {c ∈ C_k | c.count ≥ θ.default}
(13)  }
(14)  return L_k;
```

Kuva 3.24 Usein esiintyvien alkiojoukkojen muodostaminen

Algoritmin suorituksessa järjestetään ensin tukirajoitteet nousevaan järjestykseen mimimituen perusteella. Tämän jälkeen haetaan alkiojoukko L_1, josta muodostetaan liitoksella kandidaattialkiojoukko C_2. Löydettyjen kandidaattialkiojoukon tuki tietokannassa lasketaan. Mikäli kandidaattialkiojoukko c olisi esimerkiksi $\{i_2, i_{12}\}$, se muutetaan funktiolla *transform_itemset_to_bin_numers* alkiojoukoksi $\{B_1, B_2\}$, sillä alkio i_2 kuuluu lokeroon B_1 ja alkio i_{12} kuuluu lokeroon B_2. Tämän jälkeen käydään

minimituen mukaan järjestetyt tukirajoitteet läpi ja verrataan alkiojoukon *bin_c* lokeroita tukirajoitteen lokeroihin. Alkiojoukko *bin_c* vastaa näin ollen tukirajoitetta SC_1. Kandidaattialkiojoukko lisätään usein esiintyviin alkiojoukkoihih, mikäli sen tuki toteuttaa tukirajoitteen minimituen.

4. YHTEENVETO

Tietokantoihin tallentuu suuri määrä tärkeää tietoa, joka pitää vain etsiä sieltä. Tätä monivaiheista prosessia kutsutaan tietämyksen etsimiseksi. Tietämyksen etsimisen yksi vaihe on tiedonrikastus, johon sisältyy erilaisia tekniikoita. Yksi näistä tekniikoista on assosiaatiosääntöjen johtaminen. Tietokannasta etsitään usein toistuvia alkiojoukkoja ja näistä muodostetaan assosiaatiosääntöjä. Assosiaatiosäännöillä saadaan tietokannoista esille mielenkiintoisia mielleyhtymiä, sillä tietokantaan tallennetuilla tiedoilla on erilaisia suhteita ja yhteyksiä toisiinsa. Yksinkertaisin sovellus assosiaatiosäännöistä on ostoskorianalyysi, jossa tutkitaan mitä tuotteita kuluttaja ostaa usein samalla kerralla käydessään kaupassa.

Käytettävät assosiaatiosäännöt voivat olla yksinkertaisia tai sitten hyvin monimutkaisia. Assosiaatiosäännöissä oleva tiedon olemus voidaan jakaa neljään eri luokkaan, jotka ovat tiedon tyyppi, tiedon ulottuvuus, tiedon hierarkia ja tiedon harvinaisuus. Etsittäessä assosiaatiosääntöjä, jotka muodostuvat näistä tiedon olemuksen luokista, voidaan algoritmeilla löytää lukuisia assosiaatiosääntöjä. Suuri osa säännöistä ei kuitenkaan ole mielenkiintoisia. Assosiaatiosääntöjä arvioidaan sekä objektiivisilla että subjektiivisilla mittareilla, sillä assosiaatiosääntöjen johtamisessa haetaan nimenomaan mielenkiintoisia sääntöjä. Objektiiviset mittarit perustuvat käytettävien algoritmien rakenteeseen sekä tilastotietoihin. Subjektiivisiin mittareihin liittyy käyttäjän uskomukset ja kokemukset.

Assosiaatiosääntöjen johtamisen ensimmäisessä vaiheessa etsitään tietokannasta usein esiintyviä alkiojoukkoja minimituki-parametrilla, joka voi olla yhtenäinen tai epäyhtenäinen. Usein esiintyvien alkiojoukkojen tuki tietokannassa ylittää minimituki-parametrin. Yhtenäistä minimitukea käytetään esimerkiksi Apriori-algoritmissa, joka on lähtökohta useille myöhemmin kehitellyille assosiaatiosääntöjä etsiville algoritmeille. Usein esiintyviä alkiojoukkoja etsitään kandidaattialkiojoukkojen avulla, jotka muodostetaan edellisen kierroksen usein esiintyvistä alkiojoukoista. Kandidaattialkiojoukkoihin ei hyväksytä alkiojoukkoja, joiden alijoukot edelliseltä tasolta eivät ole usein esiintyviä. Yhtenäistä minimitukea käytettäessä törmätään harvinaisten alkioiden ongelmaan. Kun minimituki asetetaan alhaiseksi, saadaan suuria määriä sääntöjä

ja kohdataan mielenkiintoisuusongelma. Lukemattomista assosiaatiosäännöistä ei enää löydetä mielenkiintoisia sääntöjä ja assosiaatiosääntöjen johtamisesta ei ole mitään konkreettista hyötyä. Jos sen sijaan minimituki asetetaan liian korkeaksi, ei sääntöihin saada mukaan tietokannassa harvoin esiintyviä alkioita. Tämän ongelman ratkaisemiseksi voidaan käyttää epäyhtenäistä minimitukea.

Assosiaatiosääntöjä etsivissä algoritmeissa voidaan epäyhtenäistä minimitukea käyttää eri tavoin. Tässä tutkielmassa tutustuimme neljään menetelmään. Yhteistä näille kaikille on se, että niillä ratkaistaan harvinaisten alkioiden ongelma. Hierarkkiselle tiedolle voidaan käyttää yhtenäistä tai epäyhtenäistä minimitukea. Hierarkkisessa tiedossa alkiot on jaoteltu käsitehierarkian tasoille, jolloin ylempien tasojen alkioilla on suurempi tuki. Epäyhtenäistä minimitukea käyttämällä, tasoittain vähenevän minimituen menetelmässä, annetaan eri tasoille erilliset minimituki-parametrit. Tällöin saadaan käsitehierarkian alemmiltakin tasoilta assosiaatioita, ilman ylempien tasojen turhia assosiaatioita.

Alkiokohtaista minimitukea käyttämällä saadaan assosiaatiosääntöihin mukaan harvoin esiintyviä alkioita, sillä jokaiselle alkiolle määrätään minimituki yksilöllisesti. Kandidaattialkiojoukko on usein esiintyvä, jos sen tuki ylittää alkiojoukon pienimmän minimialkiotuen. Tietokannoissa harvoin esiintyville alkioille voidaan antaa pieni minimialkiotuki, jolloin ne saadaan mukaan assosiaatiosääntöihin. Alkiokohtaista minimitukea käyttämällä ei kuitenkaan saada esiin merkittävää, harvinaista dataa, sillä menetelmässä ei oteta huomioon alkiojoukkojen uskottavuutta. Merkittävällä, harvinaisella datalla on tietokannassa alhainen tuki, mutta korkea uskottavuus.

Suhteellisen tuen menetelmässä käytetään kahta minimituki-parametria sekä suhteellista tukea. Menetelmässä otetaan huomioon alkiojoukkojen uskottavuus, jolloin saadaan tietokannasta esille merkittävää, harvinaista dataa. Kandidaattialkiojoukkojen suhteellinen tuki lasketaan ja minimi suhteellisen tuen ylittävät alkiojoukot todetaan merkittäväksi, harvinaiseksi dataksi, mikäli alkiojoukko toteuttaa myös minimituki-parametrin.

Tukirajoitteisiin perustuvassa menetelmässä annetaan minimituki kokonaiselle alkiojoukolle. Annettu minimituki riippuu siitä, mitä alkioita alkiojoukko sisältää. Jokainen alkio kuuluu johonkin lokeroon. Tukirajoitteiden lokeroita verrataan alkiojoukon alkioiden edustamiin lokeroihin ja minimitueksi annetaan pienimmän sopivan tukirajoitteen minimituki.

VIITELUETTELO

Agrawal, R., Imielinski, T., Swami, A. (1993) Mining Association Rules between Sets of Items in Large Databases. *SIGMOD Conference 1993* (toim. Buneman, P., Jajodia, S.), ACM Press, Washington DC, 207-216.

Agrawal, R., Srikant, R. (1994) Fast Algorithms for Mining Association Rules in Large Databases. *Proceedings of 20th nternational Conference on Very Large Data Bases* (toim. Bocca, J. B., Jarke, M., Zaniolo, C.), Morgan Kaufmann, Santiago de Chile, Chile, 487-499.

Agrawal, R., Srikant, R. (1995) Mining Sequential Patterns. *Proceedings of the Eleventh International Conference on Data Engineering* (toim. Yu, P. S., Chen, A. L. P.), IEEE Computer Society, Taipei, Taiwan, 3-14.

Au, W., Chan, K. C. C (2003) Mining Fuzzy Association Rules in a Bank-Account Database. *IEEE Transactions on Fuzzy Systems* **11**(2), 238-248.

Chen, M., Han, J., Yu, P. S. (1996) Data Mining: An Overview from Database Perspective. *IEEE Transactions on Knowledge and Data Engineering* **8**(6), 866-883.

Chiu, K. S. Y., Luk, R. W. P., Chan, K. C. C., Chung, K. F. L. (2002) Market-Basket Analysis with Principal Component Analysis: An Exploration. *IEEE International Conference on Systems, Man and Cybernetics* Hammamet, Tunisia.

Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J. D., Yang, C. (2001) Finding Interesting Associations without Support Pruning. *Knowledge and Data Engineering* **13**(1), 64-78.

Cooley, R., Mobasher, B., Srivastava, J. (1999) Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems* **1**(1), 5-32.

Dunham, M. H., Xiao, Y., Gruenwald, L., Hossain, Z. (2000) *A Survey of Association* Rules. http://www.cs.uh.edu/~ceick/6340/grue-assoc.pdf (18.08.2003).

Elmasri, R., Navathe, S. B. (2000) *Fundamentals of Database Systems.* Addison-Wesley, Reading, Massachussets, USA.

Ganti, V., Gehrke, J., Ramakrishnan, R. (1999) Mining Very Large Databases. *IEEE Computer* **32**(8),38-45.

Garcia-Molina, H., Ullman, J. D., Widom, J. (2002) *Database Systems: The Complete Book*. Prentice Hall, New Jersey.

Han, J., Fu, Y. (1995) Discovery of Multiple-Level Association Rules from Large Databases. *Proceedings of 21th International Conference on Very Large Data Bases* (toim. Dayal, U., Gray, P. M. D., Nishio, S.), Morgan Kaufmann, Zurich, Switzerland.

Han, J., Fu, Y. (1999) Mining Multiple-Level Association Rules in Large Databases. *IEEE Transactions on Knowledge and Data Engineering* **(11)**5, 798-805.

Han, J., Kamber, M. (2001) *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers, San Fransisco, CA.

Hand, D., Mannila, P., Smyth, P. (2001) *Principles of Data Mining*. MIT Press, Cambridge, CA.

Houtsma, M., Swami, A. (1995) Set-Oriented Mining for Association Rules in Relational Databases. *Proceedings of the Eleventh International Conference on Data Engineering* (toim. Yu, P. S., Chen, A. L. P.), IEEE Computer Society, Taipei, Taiwan, 25-33.

Jin, L., Li, C., Mehrotra, S. (2002) *Efficient Similarity String Joins in Large Data Sets*. UCI ICS technical report TR-DB-02-04, University of California at Irvine, USA.

Kandel, A., Last, M., Bunke, H. (2001) *Data Mining and Computational Intelligence*. Physica-Verlag, Germany.

Kantardzic, M. (2003) *Data Mining Concepts, Models, Methods, and Algorithms.* Wiley-Interscience, Piscataway, NJ, USA.

Liu, B., Hsu, W., Mun, L., Lee, H. (1999a) Finding Interesting Patterns Using User Expectations. *IEEE Transactions on Knowledge and Data Engineering* **(11)**6, 817-832.

Liu, B., Hsu, W., Ma, Y. (1999b) Mining Association Rules with Multiple Minimum Supports. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, New York, NY, USA, 337-341.

Liu, B., Hsu, W., Chen, S., Ma, Y. (2000) Analyzing the Subjective Interestingness of Association Rules. *IEEE Intelligent Systems* **15**(5), 47-55.

Padmanabhan, B., Tuzhilin, A. (1999) Unexpectedness as a Measure of Interestingness in Knowledge Discovery. *Decision SupportSystems* **27**(3), 303-318.

Park, J. S, Chen, M., Yu, P. S. (1995) An Effective Hash-Based Algorithm for Mining Association Rules. *Proceedings of the 1995ACM SIGMOD International Conference on Management of Data* (toim. Carey, M. J., Schneider, D. A.), ACM Press, New York, NY, USA, 175-186.

Park, J. S, Chen, M., Yu, P. S. (1997) Using a Hash-Based Method with Transaction Trimming for Mining Association Rules. *IEEE Transactions on Knowledge and Data Engineering* **9**(5), 813-825.

Pinto, H (2001) *Multi-Dimensional Sequential Pattern Mining*. Master Thesis, Simon Fraser University, Canada.

Rantzau, R. (1997) *Extended Concepts for Association Rule Discovery*. Diplomarbeit Nr. 1554, Fakultät Informatik, Institut für Parallele und Vertailte Höchstleistungsrechner, Universität Stuttgart, Deutschland.

Sahar, S (1999) Interestingness Via What Is Not Interesting. *Proceedings of theFifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, San Diego, CA, USA, 332-336.

Savasere, A., Omiecinski, E., Navathe, S. (1995) An Efficient Algorithm for Mining Association Rules in Large Databases. *Proceedings of 21th International Conference on Very Large Data Bases*. (toim. Dayal, U., Gray, P. M. D., Nishio, S.), Morgan Kaufmann, Zurich, Switzerland, 432-444.

Silberschatz, A., Tuzhilin, A. (1995) On Subjective Measures of Interestingness in Knowledge Discovery. *Proceedings of the First International Conference on Knowledge Discovery and Data Mining* (toim. Fayyad, U. M., Uthurusamy, R.), AAAI Press, Montreal, Canada, 275-281.

Srikant, R., Agrawal, R. (1995) Mining Generalized Association Rules. *Proceedings of the 21th International Conference on Very Large Data Bases* (toim. Dayal, U., Gray, P. M. D,. Nishio, S.), Morgan Kaufmann, Zurich, Switzerland, 407-419.

Srivastava, J., Cooley, R., Deshpande, M., Tan, P. (2000) Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. *SIGKDD Explorations* **1**(2),12-23.

Toivonen, H. (1996) Sampling Large Databases for Association Rules. *Proceedings of 22th International Conference on Very Large Data Bases* (toim. Vijayamaran, T. M., Buchmann, A. P., Mohan, C., Sarda, N. L.), Morgan Kaufmann, Mumbai, India, 134-145.

Wang, K., He, Y., Han, J. (2003) Pushing Support Constraints Into Association Rules Mining. *IEEE Transactions on Knowledge and Data Engineering* **(15)**3, 642-658.

Witten, I. H, Frank, E. (2000) *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, San Fransisco, CA.

Wur, S., Leu, Y. (1999) An Effective Boolean Algorithm for Mining Association Rules in Large Databases. *Proceedings of the Sixth International Conference on Database Systems for Advanced Applications* (toim. Chen A. L. P, Lochovsky F. H.), IEEE Computer Society, Hsinchu, Taiwan, 179-186.

Yang, D., Pan, C., Chung, Y. (2001) An Efficient Hash-Based Method for Discovering the Maximal Frequent Set. *25th International Computer Software and Applications Conference*, IEEE Computer Society, Chicago IL, USA, 511-516.

Yen, S., Chen, A. L. P (2001) A Graph-Based Approach for Discovering Various Types of Association Rules. *IEEE Transactions on Knowledge and Data Engineering* **(13)**5, 839-845.

Yun, H., Ha, D., Hwang, B., Ryu, K. H. (2003) Mining association rules on significant rare data using relative support. *The Jornal of Systems and Software* **(67)**3, 181-191.

Yilmaz, E., Triantaphyllou, E., Chen, J. Liao, T. W. (2003) A Heuristic for Mining Association Rules in Polynomial Time. *Mathematical and Computer Modelling* 37, 219-233.