Semanttisten yhteyksien selvittämiseen soveltuva kyselykieli

Janne Jämsen

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi
Pro gradu -tutkielma
Toukokuu 2004

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi
Janne Jämsen: Semanttisten yhteyksien selvittämiseen soveltuva kyselykieli
Pro gradu -tutkielma, 86 sivua, 10 liitesivua
Toukokuu 2004

Tutkielmassa käsitellään keskeisiä tietokantaparadigmoja niihin liittyvine kyselykielineen sekä tarkastellaan seuraavan sukupolven tietojärjestelmille (Next Generation Information Systems - NGIS) ehdotettuja keskeisiä piirteitä. Erityisesti esitellään semanttisen yhteyden käsite ja selvitetään, miten semanttisia yhteyksiä tukeva kyselykieli mahdollistaa uudentyyppisten kyselyiden tekemisen tietokannassa. Huomiota kiinnitetään myös tieto- ja tekstiorientoitunutta käsittelyä tukevien piirteiden integroimiseen samaan kyselykieleen. Kehitetty semanttisten yhteyksien selvittämiseen kykenevä kyselykieli, jota on täydennetty vapaamuotoisten dokumenttien käsittelyyn soveltuvilla primitiiveillä, esitellään kieliopillisen syntaksin ja esimerkkikyselyiden avulla. Kyselykielen suunnittelussa, ilmaisuvoiman lisäksi, on erityistä huomiota kiinnitetty sen ilmausten deklaratiivisuuteen. Myös kyselykielen arkkitehtuuria, sen Prolog-toteutusta ja prosessoinnin tehostamiseen sovellettuja menetelmiä tarkastellaan tutkielmassa.

Avainsanat ja -sanonnat: semanttinen yhteys, kyselykieli, tietokanta, tietojärjestelmä, NGIS (seuraavan sukupolven tietojärjestelmät), XML.

Sisällys

1. Johdanto

Relaatiomallin puutteet (kts. esim. [Paton et al., 1996]) ovat synnyttäneet tarpeen kehittää uudenlaisia, relaatiomallia mallintamisominaisuuksiltaan rikkaampia ja ilmaisuvoimaisempia tietokantaparadigmoja. Osa niistä on jo aktiivisessa käytössä, kun taas osan merkitys on ainakin toistaiseksi ollut lähinnä teoreettinen. Tutkielmassa esitellään ensin keskeisimpiä näistä relaatiomallille vaihtoehtoisista tietokantaparadigmoista. Tarkastelunäkökulmana ovat erityisesti kyseisiin paradigmoihin perustuvien kyselykielten ilmaisuvoima ja käytettävyys tavallisen loppukäyttäjän näkökulmasta. Toisen luvun lopussa tehdään katsaus seuraavan sukupolven tietojärjestelmille (Next Generation Information Systems - NGIS) ehdotettuihin piirteisiin. Aihealueeseen liittyvää tutkimus- ja kehitystyötä on tehty runsaasti Tampereen yliopistossa [Niemi et al., 2002a, 2002b, 2004], ja tämä antaa luonnollisen tarkastelunäkökulman aiheeseen.

Perinteisissä ekstensionaalisia kyselyitä tukevissa tietokantojen kyselykielissä käyttäjän tulee hallita yksityiskohtaisesti kyselyn kannalta relevantit kaavion osat ja niiden rakenteet. Seuraavan sukupolven tietojärjestelmien tulee mahdollistaa lisäksi kyselyt, jotka kohdistuvat tietokannan kaaviotasoon, sekä kyselyt, joissa yhdistetään tietokannan kaavio- ja ilmentymätasoon liittyvää informaatiota [Niemi et al., 2002a, 2004]. Tutkielmassa käsitellään tähän liittyen kyselykielten laajentamista semanttisten yhteyksien selvittämispiirteellä. Tutkielman kolmannessa luvussa esitellään semanttisen yhteyden käsitettä ja pohditaan sen soveltamismahdollisuuksia seuraavan sukupolven tietojärjestelmissä. Erityisesti yritetään yhdistää se aiempaan käsitteistöön tarkastelemalla semanttisten yhteyksien selvittämiseen soveltuvalta kieleltä vaadittavaa ilmaisuvoimaa. Samoin pohditaan niitä uusia mahdollisuuksia, joita tällaiset kyselykielet tarjoavat loppukäyttäjälle. Myös aiheeseen mahdollisesti liittyviä ongelmakohtia tarkastellaan.

Tutkielman toinen painopistealue on tietokantaan liittyvän tekstiorientoituneen tiedon käsittely. Perinteisesti tietokantaan tallennetun tiedon ja samaan sovellusalueeseen liittyvien dokumenttien käsittely ovat olleet erillään toisistaan. Tämä on ilmennyt siten, että niiden käsittelyssä on sovellettu erillisiä kysely- ja tiedonhakujärjestelmiä. Seuraavan sukupolven tietojärjestelmien yhteydessä on kuitenkin ehdotettu tekstuaalisen ja muun tiedon käsittelyä integroidusti [Niemi et al, 2002a., 2004]. Tutkielmassa oletetaan vapaamuotoisten dokumenttien noudattavan XML-syntaksia [XML, 2004]. Tämän johdosta neljännessä luvussa esitellään ensin tärkeimmät XML-kyselykielet, minkä jälkeen pohditaan niille ominaisten piirteiden yhdistämistä osaksi varsinaista tietokannan kyselykieltä.

Tutkielmassa esitellään toteutettu semanttisten yhteyksien selvittämiseen soveltuva kyselykieli. Toteutettu kyselykieli tukee rajoitetusti myös tietokantaan liittyvien vapaamuotoisten dokumenttien käsittelyä. Luvuissa 5 ja 6 esitellään kyselykielen demonstrointiin tarkoitettu esimerkkitietokanta ja vapaamuotoiset dokumentit sekä niihin liittyvän tiedon loogiset ja fyysiset organisointitavat. Tutkielman kyselykielen syntaksi primitiiveineen annetaan kokonaisuudessaan seitsemännessä luvussa. Kyselykielen toteutustapaa ja arkkitehtuuria Prolog-ympäristössä käsitellään luvussa 8. Erityisesti huomiota kiinnitetään mahdollisuuksiin tehostaa semanttisten yhteyksien selvittämiseen tarkoitettujen kyselyiden prosessointia. Lopuksi yhdeksännessä luvussa esitetään kyselykielen kannalta keskeiset kyselytyypit ja annetaan jokaisesta kyselytyypistä yksi tai useampi esimerkkikysely.

2. Tietokantaparadigmat ja niihin pohjautuvat kyselykielet

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

Käyttäjä on vuorovaikutuksessa tietokannan kanssa jonkin kysely-, tiedonmäärittely- tai käsittelykielen välityksellä [Ullman, 1988]. Kyselykieli on erityisesti tietokannassa olevien tietojen hakemiseen ja niiden pohjalta muodostettavan informaation johtamiseen tarkoitettu korkean tason kieli. Tiedonmäärittelykielen avulla määritellään tietokannan kaaviotaso, johon liittyvää ilmentymätason tietoa päivitetään (lisätään, muokataan, poistetaan) käsittelykielen avulla. Usein kyselykieleen yhdistyy myös määrittely- ja päivitysominaisuuksia, minkä vuoksi rajanveto kyselykielten ja tiedonmäärittely- tai käsittelykielten välillä voi olla käytännössä hankalaa. [Samet, 1981; Date, 1989] Tämän tutkielman puitteissa rajaudutaan kuitenkin tarkastelemaan puhtaasti kyselykielille ominaisia piirteitä.

Tyypillisesti kyselykieli perustuu tietokannan organisoimisessa käytettyyn tietomalliin. Kyselykieli voi kuitenkin perustua myös johonkin muuhun tietomalliin. Esimerkiksi funktionaalisia kyselykieliä käytetään tavallisimmin relaatiomalliin tai olio-orientoituneisiin tietomalleihin perustuvien tietokantojen yhteydessä. Samassa kyselykielessä voi yhdistyä myös useiden erilaisten tietomallien piirteitä. [Paton et al., 1996]

Tutkielmassa erotetaan toisistaan tietokantaparadigma ja tietomalli. Eräät tietokantaparadigmat perustuvat täsmällisesti määriteltyyn tietomalliin (esimerkiksi relaatiomalli). Joihinkin tietokantaparadigmoihin liittyen (esimerkiksi olio-orientoituneet tietokannat) mitään yleisesti hyväksyttyä tietomallia ei ole

esitettävissä. Tietokantaparadigmalla viitataan tutkielmassa siis tietomallia laajempaan käsitekokonaisuuteen. Se ymmärretään abstraktiksi piirrejoukoksi, joka kuvailee paradigmaan perustuville tietomalleille ja käytännön toteutuksille tyypillisiä piirteitä. Sellaisenaan se on verrattavissa esimerkiksi ohjelmointiparadigman käsitteeseen.

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

2.1. Relaatiomalli ja relaatiotietokannat

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

Relaatiomallin yksinkertaisuus ja abstraktisuus perustuvat siihen, että mallin mukaisen relaatiotietokannan ainoita rakenteellisia konstruktioita ovat relaatiot ja niiden attribuutit. Ne kuvaavat puhtaasti tiedon loogisia ominaisuuksia, eivätkä siten ole riippuvaisia tiedon fyysisestä tallennusmuodosta. Varhaisiin tietokantaparadigmoihin (esim. verkkomalli, hierarkkinen malli) perustuvat tietomallit edellyttivät tavallisesti, että käyttäjä tuntee yksityiskohtia tavasta, jolla tieto on fyysisesti organisoitu. [Codd, 1970] Tämän vuoksi kyselyn muodostaminen on relaatiomalliin perustuvalla kyselykielellä tavallisen loppukäyttäjän kannalta huomattavasti yksinkertaisempaa.

Relaatiomallille voidaan antaa joukko-opillinen tulkinta. Joukko-opillisesti n-paikkainen relaatio R on siihen osallistuvien arvojoukkojen D_1, D_2, …, D_n muodostaman karteesisen tulon $D_1 \times D_2 \times \ldots \times D_n$ osajoukko. Yksittäistä relaatioon kuuluvaa arvojen järjestettyä joukkoa $(d_1, d_2, \ldots, d_n) \in R$ kutsutaan monikoksi. Käytännössä relaatio voidaan visualisoida tauluna, jonka rivit ovat monikkoja ja sarakkeet koostuvat nimettyihin attribuutteihin liittyvistä arvoista. [Ullman, 1988]

Kyselyt relaatiomallissa voidaan formalisoida E. Coddin vuonna 1970 esittämällä relaatioalgebralla. Vaihtoehtoisesti voidaan käyttää matemaattisen logiikan ilmaisutapaan perustuvaa relaatiokalkyylia. Karteesisen tulon lisäksi relaatioalgebra sisältää operationaaliset vastineet relationaalisissa kyselykielissä yleisesti esiintyville toiminnoille: liitos, projektio, valinta, unioni, leikkaus ja

erotus [Zaniolo et al., 1997]. Nykyaikaisten relationaalisten kyselykielten (kts. esim. SQL [Date, 1989]) ilmaisuvoima ylittää osin relaatioalgebran ilmaisuvoiman, sillä ne sisältävät relaatioalgebran operaatioiden lisäksi erilaisia aggregointifunktioita, joilla ei ole relaatioalgebrassa vastinetta [Järvelin and Niemi, 1999].

Relaatiomalliin perustuvat kyselykielet ovat luonteeltaan deklaratiivisia. Korkean deklaratiivisuuden asteen omaaville kyselykielille on ominaista, että kysely pohjautuu siihen, mitä halutaan lopputulokseksi, ei siihen, miten lopputulos tuotetaan. Deklaratiivisten kyselyiden mahdollistamiseksi vastuu kyselyn optimoinnista siirretään käyttäjältä kyselykielen toteuttavalle ohjelmistolle. Kieliä, jotka eivät ole deklaratiivisia, kutsutaan proseduraalisiksi [Ullman, 1988].

Relaatiomallin ilmeisistä eduista huolimatta siihen kohdistettu myös kritiikkiä. Keskeisimpiä ongelmia ovat relaatiomallin kykenemättömyys ilmaista kompleksisia rakenteita ja transitiivisia suhteita sekä sen heikko tuki tiedon johdettavuudelle eli deduktiolle. Käsitteellisellä tasolla relaatiomallin abstraktiotasoa on pidetty liian alhaisena verrattuna esimerkiksi ER-malliin, jota käytetään yleisesti relaatiotietokantojen kaavioiden suunnittelussa. Myös käytännön ohjelmointityön näkökulmasta relaatiomalli on ongelmallinen. Ohjelmallisia kyselyitä ja päivityksiä ei voida toteuttaa suoraan laajassa tuotantokäytössä olevien (yleensä proseduraalisten ja olio-orientoituneiden) ohjelmointikielten omin rakentein, vaan on suoritettava konversio kahden erilaisen esitystavan välillä. [Paton et al., 1996]

Relaatiomallin puutteet ovat johtaneet toisaalta uusien tietokantaparadigmojen kehittämiseen ja toisaalta pyrkimyksiin laajentaa relaatiomallia itseään. Edellä esitellyssä relaatiomallin ensimmäisessä normaalimuodossa relaatioiden attribuuteilla voi olla vain atomisia arvoja. Ensimmäinen normaalimuoto voidaan laajentaa NF^2-muotoon (ei- ensimmäinen normaalimuoto) sallimalla lisäksi relaatiot, joiden attribuuttien arvoina on toisia relaatioita. NF^2-muotoinen relaatiomalli soveltuu hyvin kompleksisten olioiden (engl. entity) esittämiseen. Myös relaatioalgebran ilmaisuvoimaa voidaan kasvattaa ottamalla käyttöön operaattori, joka soveltuu transitiivisten suhteiden käsittelyyn. [Järvelin and Niemi, 1999]

2.2. Funktionaaliset kyselykielet ja tietokannat

Funktionaalisia kysely- ja ohjelmointikieliä voidaan pitää proseduraalisten ja deklaratiivisten kielten välimuotona [Hillebrand and Kanellakis, 1994]. Niiden tärkein erottava piirre perinteisiin proseduraalisiin kieliin verrattuna on hävittävän sijoitusoperaation puuttuminen. Tämän vuoksi muuttujan arvo ei voi funktionaalisessa kielessä muuttua kesken prosessoinnin, vaan se on sidottu suhteessa käyttökontekstiin. Arvojen staattisuuden ansiosta funktiokutsujen

keskinäinen suoritusjärjestys ei ole puhtaan proseduraalisten kielten tapaan sidottu, mitä voidaan käyttää hyväksi kyselynoptimoinnissa. Funktionaalisissa kielissä kaksi saman arvon omaavaa ilmausta ovat ekvivalentit ts. ne voidaan korvata keskenään. Periaatteen nojalla puhtaan funktionaalisessa kielessä ei voi esiintyä arvosemantiikan ulkopuolisia operaatioita eli sivuvaikutuksia. [Paton et al., 1996]

Pelkistetyssä funktionaalisessa kielessä ei literaaliarvojen ja muuttujien lisäksi ole muita konstruktioita kuin funktiot. Funktio voi saada mielivaltaisen määrän parametreja, joiden perusteella se palauttaa yhden arvon. Sekä palautettava arvo että parametrit voivat olla literaaliarvoja, muuttujia tai funktioita. Tietoalkioista muodostuvia kokoelmia (esimerkiksi listoja) voidaan ilmaista erityisillä konstruktorifunktioilla. Tyypitetyissä funktionaalisissa kielissä on lisäksi mahdollisuus määritellä tietotyyppejä, jotka edustavat joko literaalityyppejä tai niistä johdettuja kompleksisia tyyppejä. [Paton et al., 1996; Knuutila, 2001]

Funktionaaliset kielet pohjautuvat lambda-kalkyyliin, jota voidaan pitää yksinkertaisimpana tapana määritellä matemaattinen funktion käsite. Lambda-kalkyylin esitti Alonzo Church 1930-luvulla. [Knuutila, 2001] Kalkyylista on olemassa myös tyyppimäärittelyn sisältävä versio, jota voidaan käyttää tyypitettyjen funktionaalisten kielten formalisointiin. [Hillebrand and Kanellakis, 1994] Lambda-kalkyylin ja samalla funktionaalisten kysely- ja ohjelmointikielten ilmaisuvoima on sama kuin Turingin koneen [Rosser, 1982]. Tämän laskennallisen täydellisyyden ansiosta funktionaaliset kyselykielet ovat huomattavasti esimerkiksi relaationaalisia kyselykieliä ilmaisuvoimaisempia. Funktionaaliset kielet soveltuvatkin hyvin juuri kyselykieliksi, sillä kyselyiden prosessointi ei edellytä tietokannan päivittämistä. Päivitysoperaatioiden ilmaiseminen ilman hävittävää sijoitusta on hankalaa. Funktionaalisia kieliä voidaan käyttää erilaisiin tietokantaparadigmoihin perustuvien tietokantojen kyselykielinä [Paton et al., 1996].

Seuraavaksi tarkastellaan funktionaalisten kyselykielten yleisiä ominaisuuksia vertailemalla lyhyesti kolmea funktionaalista kyselykieltä: DAPLEX:ia, FQUERY:a ja FQL:ää. Näistä DAPLEX toimii aidosti funktionaalisen tietokannan kyselykielenä tarjoten lisäksi funktionaalisen tietokannan päivitykseen ja sen kaavion ylläpitoon liittyviä toimintoja [Shipman, 1981]. FQUERY on tarkoitettu relaatiotietokantojen kyselykieleksi, kun taas FQL on periaatteessa tietokantaparadigmasta riippumaton; käytettävältä tietokannalta edellytetään, että sille on määritelty rajapinta, jonka avulla tieto voidaan kuvata FQL:n ymmärtämään yksinkertaiseen funktionaaliseen muotoon [Buneman and Frankel, 1979; Warner and Odle, 1981].

Tiedon funktionaaliselle kuvaukselle on DAPLEX:in ja FQL:n tapauksissa ominaista, että tieto esitetään funktioiden, olioiden (engl. entity) ja literaaliarvojen avulla. Olioiden ominaisuudet kuvataan funktioina, jotka saavat paramet-

rikseen kohdeolion ja palauttavat ominaisuuden arvoa vastaavan literaaliarvon. Vastaavasti olioiden keskinäiset suhteet ilmaistaan niiden välisinä funktioina. DAPLEX:issa myös oliotyypit esitetään funktioina. Ne ovat funktioita, jotka eivät saa parametrejä, mutta palauttavat tyyppiin kuuluvien olioiden joukon. [Shipman, 1981; Buneman and Frankel, 1979]

Funktionaalisen tiedon esittämisen huomattavin ongelma relaatiomalliin nähden on se, että funktioiden käsittely tapahtuu lähtökohtaisesti vain yksisuuntaisesti. Haluttaessa käsitellä suhteita jossain muussa järjestyksessä täytyy tätä varten määritellä uusi johdettu funktio. Binääristen suhteiden kohdalla tämä tapahtuu luontevasti käyttäen kielten tarjoamia valmiita primitiivejä, joilla voidaan johtaa funktion käänteisfunktio. Korkeampaa astelukua edustavien ja mahdollisesti omia attribuutteja sisältävien suhteiden käsittely on sen sijaan ongelmallisempaa. Ilmeinen ratkaisu on mallintaa tällaiset suhteet binäärisiksi funktioiksi, jotka liittyvät korkeampaa astelukua olevaa suhdetta edustavaan olioon.

FQUERY käsittelee relaatioita ja tukee siten relaatioalgebralle tyypillisiä operaatioita [Warner and Odle, 1981]. Sen syntaksi muistuttaa SQL:ää. Erottava piirre relaationaalisiin kyselykieliin nähden on se, että FQL:n ja DAPLEX:in tapaan kyselykieli sisältää primitiivit, joilla esitetään iterointi tietoalkioiden muodostamassa joukossa. Iterointia tarvitaan aina suoritettaessa funktiokutsu tai operaatio joukon jokaiselle alkiolle. Menettelyn vahvuus on se, että sen avulla voidaan muodostaa helposti esimerkiksi aggregointifunktiot, joiden esittäminen relaatioalgebralla ei ole mahdollista. Toisaalta se edellyttää käyttäjältä enemmän algoritmista ajattelua kuin puhdas relationaalinen kieli, mikä vähentää kielen deklaratiivisuuden astetta. Vaihtoehtoinen funktionaalisissa kyselykielissä käytetty tapa operoida kokoelman alkioilla on logiikan joukonmuodostus operaatiota vastaava merkintä (engl. comprehension). Sen avulla kokoelmasta voidaan valikoida esitettävät valintakriteerit täyttävät tietoalkiot [Paton et al., 1996]. Menettely on käyttäjän kannalta deklaratiivisempi, mutta toisaalta se ei itsessään edusta funktionaalista ilmaisutapaa.

Kolmesta kyselykielestä vain FQL:ssä kysely on funktiomäärittely. Kyselyllä voi olla funktioiden tapaan parametrejä, jolloin samaa kyselyrunkoa voidaan periaatteessa käyttää useaan eri kyselyyn. DAPLEX sen sijaan muistuttaa ohjausrakenteiltaan enemmän käskyorientoitunutta kieltä ja FQUERY SQL:n tapaista deklaratiivista kyselykieltä. Mielenkiintoista on, että sekä DAPLEX:in että FQUERY:n suunnittelijat ilmoittavat ratkaisuillaan korostaneensa kyselykielen helppokäyttöisyyttä ja intuitiivisuutta loppukäyttäjän näkökulmasta [Shipman, 1981; Warner and Odle, 1981]. Sitä vastoin FQL:n suunnittelijat esittävät varauksen oman kielensä soveltuvuudesta loppukäyttäjälle [Buneman and Frankel, 1979]. Näyttää siis ilmeiseltä, että puhtaan funktionaalisille kyselykielille ominaisen syntaksin ja ajattelutavan oppiminen edellyttää käyttäjältä keskimääräis-

tä enemmän ohjelmointimenetelmien tuntemusta. Tarvittavan käsitteistön vaativuutta on silti vaikeaa verrata esimerkiksi deduktiivisten kyselykielten käytön vaativuuteen.

2.3. Olio-orientoituneet tietokannat

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

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

On huomattava, että mitään yksikäsitteistä määritelmää tai kriteeristöä olio-orientoituneelle tietomallille ei ole hyväksytty [Bertino et al., 1992]. Tämä selittää osaltaan olemassa olevien olio-orientoituneiden tietokantajärjestelmien suurehkot keskinäiset erot. Kuitenkin vakiintuneen käytännön mukaan olio-orientaatiolla viitataan nykyisin olio-identiteetin ohella myös laajempaan joukkoon piirteitä, jotka ovat peräisin olio-orientoituneista ohjelmointiparadigmois-

ta [Kim, 1990]. Piirteet voidaan jakaa strukturaalisiin ja operationaalisiin (engl. behavioral) (vrt. esim. [Niemi et al., 2002b]).

Strukturaalisesti olio-orientoitunut tietomalli voidaan johtaa relaatiomallista antamalla monikoille olio-identiteetti ja määrittelemällä attribuuttien arvoalueeksi jokin järjestelmässä määritelty tietotyyppi eli luokka. Luokka edustaa literaalityyppiä tai se voi olla literaalityypeistä johdettu kompleksinen tietotyyppi. Relaatiota vastaa olio-orientoituneessa järjestelmässä luokka ja monikoita oliot. Luokat järjestetään hierarkkisesti niin, että hierarkiassa alempana olevan luokan ilmentymät ovat myös hierarkiassa ylempänä olevan luokan ilmentymiä. Määrittelystä seuraa, että hierarkiassa alempana olevan luokan eli aliluokan tulee sisältää kaikki attribuutit, jotka sisältyvät sen yläluokkaan. (vrt. [Kim, 1990]) Tämän lisäksi aliluokka voi (ja hyvässä olio-orientoituneessa suunnittelussa sen myös pitää) sisältää omia attribuutteja. Olion tilaksi kutsutaan sen kaikkien attribuuttien arvojen muodostamaa kokonaisuutta tietyllä ajanhetkellä [Koskimies, 2000].

Operationaalisesti luokkiin voidaan sisällyttää attribuuttien ohella ohjelmakoodilla toteutettuja metodeja [Kim, 1990]. Metodeita kutsutaan tavallisesti luokan ilmentymän kautta. Koska myös metodien palautusarvoille on attribuuttien tapaan on määritelty tyyppi, voidaan attribuutit ja metodit rinnastaa toisiinsa. Attribuutit vastaavat metodeita, joiden parametrien määrä on nolla (vrt. esim. [Niemi et al., 2002b]). Attribuuttien tavoin aliluokka perii kaikki yläluokan metodit, mutta aliluokan määrittelyn yhteydessä on myös mahdollista kuormittaa metodi eli määritellä sille uusi toteutus.

Olio-orientaatioon liittyy läheisesti myös niin sanottu tiedon kätkennän periaate eli kapselointi, jonka nojalla luokkaan sisältyvien olioiden attribuutteja ja metodeita voidaan kutsua vain määritellyn rajapinnan kautta. Yksittäinen olio voi rajapinnassa näkyvien attribuuttien ja metodien lisäksi sisältää myös kätkettyjä attribuutteja tai metodeita. Tämän ansioista se toimii eräänlaisena ”mustana laatikkona”, jonka palveluita voidaan käyttää ilman, että käyttäjä tuntee olion tilaa tai metodien toteutusta [Ullman, 1988; Koskimies, 2000]. Tämä parantaa sovellusten modulaarisuutta sekä helpottaa niiden suunnittelua ja päivitystä.

Varhaiset olio-orientoituneet kyselykielet olivat tyypillisesti proseduraalisia. Niitä vastaavien tietokantojen olio-orientoituneet piirteet rajoittuivat usein olio-identiteetin tukemiseen. Esimerkkejä ovat CODASYL DBTG ja IMS, joista ensin mainittu edusti verkkomallia ja jälkimmäinen hierarkkista tietomallia. [Ullman, 1988] Nykyaikaisten olio-orientoituneiden tietokantojen yleistymisen yhteydessä ongelmana ovat pitkään olleet olio-orientoituneiden tietomallien keskinäiset erot ja relaatioalgebraan verrattavan yhtenäisformalismin puute [Bagui, 2003]. Tästä johtuen olio-orientoituneiden tietokantojen yhteydessä esiin ei ole noussut SQL:n tapaista yhtä laajalti hyväksyttyä kyselykieltä, vaan

valmistajasta riippuen tietokantojen kyselyominaisuudet ovat vaihdelleet suuresti.

Viime aikoina olio-orientoituneissa tietomalleissa suoritettaville kyselyille on esitetty runsaasti formaaleja malleja, muun muassa olioalgebra ja olio-orientoitunut predikaattikalkyyli [Alhajj and Polat, 2000; Bertino et al., 1992]. Nämä formalismit osoittavat, että kyselyt olio-orientoituneissa tietomalleissa voidaan suorittaa deklaratiivisesti edellyttäen, että käytössä on tietty operaatiojoukko. Niiden käytännön sovellettavuus on kuitenkin ollut rajoitettu, sillä ne eivät sinällään tarjoa esimerkiksi relaatiomallille tyypillisiä kyselynoptimointimenetelmiä [Bagui, 2003]. Ne myös estävät laskennallisesti täydellisen käsittelyn [Niemi et al., 2000].

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

Olio-orientoituneessa tietomallissa myös olioiden väliset suhteet (assosiaatiot) kuvataan tyypillisesti osa-kokonaisuus –suhdetta vastaavalla tavalla olioiden olioarvoisina attribuutteina. Tämä tekee suhteiden käsittelystä epäsymmetrisen, ja usein onkin epäselvää kummassa suhteeseen osallistuvasta oliosta suhdetta vastaava attribuutti tulee antaa [Paton et al., 1996]. Kompleksisiin olioihin kohdistuvissa kyselyissä ongelmana on, että toteutetuissa olio-orientoituneissa kyselykielissä käyttäjän täytyy kyselyn formuloinnissa tuntea olion intensionaalinen rakenne. Ainoa tapa päästä käsiksi kokonaisuuden osiin on yleensä niin sanotun polkunotaation käyttö, jossa muuttujan nimeen liitetään attribuutin nimi erotinoperaattorilla (esimerkiksi piste ilmaisussa "`kokonaisuus.attribuutti`"). Ilmaus palauttaa attribuuttia vastaavan osaolion, jonka osa-olioon voidaan edelleen viitata liittämällä polkuun uusi erotinoperaattorilla erotettu attribuutti jne. (kts. esim. [Niemi et al., 2002a]) Polkunotaation etuna esimerkiksi relaatiomalliin nähden on, että se vähentää tarvittavien eksplisiittisesti annettujen liitosehtojen tarvetta [Bagui, 2003]. Toisaalta se edellyttää polun muodostavan attribuuttiketjun tuntemista, mikä rajoittaa tämän

ilmaisutavan käytettävyyttä kompleksisten olioiden käsittelyssä [Niemi et al., 2002a]. Polkunotaatio ei myöskään poista tarvetta käsitellä olioita jossain muussa järjestyksessä kuin kaaviossa esitetyllä tavalla.

Kyselyyn sisältyvät metodikutsut voivat olla ongelmallisia. Yksinkertaisen semantiikan määrittely kyselylle edellyttää, että metodit eivät tee muutoksia olion tilaan, vaan ovat yksinomaan lukuoperaatioita [Bertino et al., 1992]. Olion tilaa muuttavien päivitysmetodien käyttö estää pahimmillaan tiedon deklaratiivisen käsittelyn edellyttäen proseduraalista kyselynmuodostusta.

Tunnetuimpiin käytössä oleviin olio-orientoituneisiin kyselykieliin kuuluu O_2-tietokannan kyselykieleksi suunniteltu OQL. Kyselykielen suurimpia etuja ovat sen SQL:stä tuttu luonnollista kieltä muistuttava rakenne ja sen erityisesti yksinkertaisissa kyselyissä mahdollistama deklaratiivinen ilmaisutapa. Sen sijaan kompleksissa kyselyissä sen ilmaisutapaa ei voida pitää erityisen deklaratiivisena. Kielen ongelmallisia piirteitä ovat sen edellyttämä erilaisten rakentimien ja iteraattoreiden sisäkkäinen käyttö, mikä voi olla liian vaativaa loppukäyttäjälle. Muiden olio-orientoituneiden kyselykielten tapaan OQL edellyttää myös toisiinsa liittyvien olioiden välisen navigointipolun täydellistä tuntemista. [Niemi et al., 2000]

2.4. Deduktiiviset tietokannat

Relaatioalgebran ilmaisuvoima on rajallinen. Jos relaatiomallia täydennetään ensimmäisen kertaluvun predikaattilogiikkaan pohjautuvilla säännöillä, saadaan tietomalli, jossa tärkeimmät relaatiomallin laskennallisista rajoituksista poistuvat [Zaniolo et al., 1997]. Loogisen esitysformalismin etuina ovat siihen liittyvä vahva malliteoreettinen semantiikka, joka laajentaa relaatiomallin vastaavaa, sekä todistusteoria, jonka avulla eksplisiittisesti annetusta tiedosta voidaan johtaa loogisesti voimassa olevia päätelmiä. Looginen esitysmuoto on luonteeltaan deklaratiivinen, mikä tekee siitä käyttäjän kannalta intuitiivisen. [Paton et al., 1996] Relaatiomallin tavoin tiedon käsittely perustuu arvo-orientoituneisuuteen.

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

Datalogissa tieto mallinnetaan sääntöinä, jotka perustuvat predikaattilogiikkaan. Proseduraalisen tulkinnan helpottamiseksi säännöt esitetään tavallisesti niin sanotussa Hornin muodossa, jossa säännöillä on positiivista loogista atomia vastaava pää, jota implikoi loogisten atomien konjunktiosta muodostu-

va säännön runko. Säännön rungossa ei esiinny disjunktiota, vaan vaihtoehtoiset tavat suorittaa johdos ilmaistaan erillisinä sääntöinä. Faktat ovat sääntöjen erikoismuoto, joilla ei ole runkoa ja jotka siten tulkitaan aina tosiksi. [Niemi, 2003] Yksinkertaisimmassa relaatiomallin ensimmäistä normaalimuotoa vastaavassa esitystavassa predikaatit sisältävät ainoastaan vakioita tai muuttujia. Kielen ilmaisuvoimaa voidaan kasvattaa sallimalla sisäkkäiset predikaattiilmaukset eli funktiosymbolit. [Zaniolo et al., 1997]

Predikaattilogiikasta poiketen kvantifiointia ei Prologissa tai Datalogissa esitetä eksplisiittisesti kvanttoreilla, vaan eksistentiaalimuodossa olevat säännöt ilmaistaan vakioilla, kun taas universaalikvantifioinnin ilmaisemiseen käytetään muuttujia. Muuttujien arvotukset ovat funktionaalisten kielten tapaan sidottuja käyttökontekstissaan. Logiikkaohjelmointikielet (esim. Prolog) sisältävät automatisoidun teoreemantodistusmekanismin, jonka avulla sääntökokoelmaan eli tietokantaan voidaan kohdistaa kyselyitä. Loogisen deduktion onnistumiseen perustuvien totuusarvokyselyjen ohella voidaan muuttujien avulla esittää myös relationaalisille kielille tyypillisiä eksistentiaalikyselyjä, jotka palauttavat kyselyssä esiintyviin muuttujiin samaistettavissa olevat vakiot. [Niemi, 2003]

Deduktiivisiin tietokantakieliin perustuvat säännöt ovat luonnollinen tapa laajentaa relaatiomallin ilmaisuvoimaa. Ensimmäistä normaalimuotoa olevat relaatiot ilmaistaan faktoilla, joissa ei esiinny funktiosymboleita. Relaatioita vastaavat faktat yhdessä tietokantaan tallennettujen deduktiivisten sääntöjen kanssa muodostavat niin kutsutun suljetun maailman, jossa kyselyn tulokseen sisältyvät arvotukset ja totuusarvot määräytyvät. Suljetussa maailmassa väite on tosi vain, mikäli se voidaan johtaa tietokantaan tallennetuista säännöistä. [Paton et al., 1996]

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

2.5. Deduktiiviset olio-orientoituneet tietokannat

Puhdas Datalog ja ensimmäistä normaalimuotoa edustava relaatiomalli eivät sinällään sovellu kompleksisten entiteettien esittämiseen. Kuten todettiin, Datalogia voidaan kuitenkin laajentaa sallimalla funktiosymboleiden käyttö. Myös ensimmäistä normaalimuotoa oleva relaatiomalli voidaan laajentaa NF^2-muotoon sallimalla sisäkkäiset relaatiot [Järvelin and Niemi, 1999]. Toinen mahdollisuus, jota tarkastelemme tässä luvussa, on pyrkiä yhdistämään deduktiiviset ja olio-orientoituneet tietokantaparadigmat toisiinsa. Myös tämä lä-

hestymistapa on kerännyt huomiota tieteellisessä keskustelussa, ja useita erilaisia tietomalleja kyselykielineen on esitetty. [Niemi et al., 2002b]

Eräs suurimmista deduktiivisen ja olio-orientoituneen paradigman integrointiin liittyvistä ongelmista, koskee olio- ja arvo-orientoituneiden lähestymistapojen välistä eroa. Arvo-orientoituneessa lähestymistavassa tietoyksiköt ovat identtisiä, jos niiden kaikkien attribuuttien arvot ovat samat. Olio-orientoituneessa lähestymistavassa samuus määräytyy sen sijaan olio-identiteetin perusteella. On kuitenkin mahdollista, että käyttäjä haluaa myös olio-orientoituneessa järjestelmässä vertailla olioita niiden attribuuttien arvoihin perustuen. Kaksi ei-primitiivistä oliota ovat arvoiltaan identtiset, mikäli niillä on samat attribuutit ja attribuuttien arvot ovat arvoiltaan identtiset. Jos olio on kompleksinen, edellyttää tämä myös olioarvoisten attribuuttien arvoina olevien olioiden tutkimista rekursiivisesti. Yhden vertailuoperaattorin sijasta tarvitaan siis kaksi vertailuoperaattoria, joista toinen viittaa olion olio-identiteettiin ja toinen sen attribuuttien arvoihin [Bagui, 2003]. Loogisten sääntöjen muodostuksen kannalta tämä on hankalaa, sillä se vaikeuttaa deduktiivisissa tietokantakielissä tyypillisesti käytettävien samaistamisalgoritmien käyttöä ja tekee kyselyistä monimutkaisempia. Varsinaista teoreettista estettä näiden kahden paradigman yhdistämiselle ei kuitenkaan liene, vaikka myös vastaväitteitä on esitetty [Niemi et al., 2002b; Ullman, 1988].

Vanhin deduktiivinen olio-orientoitunut (DOOD – Deductive Object-Oriented Database) kieli on O-Logic. O-Logicin tuki olio-orientoituneille piirteille on rajallinen. Se ei tue metodeita tai luokkahierarkioita. Predikaattien tai monikoiden määrittely ei O-Logicissa ole mahdollista, vaan olioiden väliset suhteet esitetään olioarvoisilla attribuuteilla. Myös literaaliarvot ovat O-Logicissa olioita. F-Logic on O-Logic:in johdos, joka sisältää muun muassa tuen periytymishierarkioille. [Liu, 1999] ROL on O-Logicia ja F-Logicia uudempi deduktiivisten olio-orientoituneiden tietokantojen kyselykieleksi tarkoitettu ja myös implementoitu kieli. ROL:in mielenkiintoinen ominaispiirre on, että se antaa käyttäjälle mahdollisuuden valita, käyttääkö hän tiedon kuvaamiseen arvo- vai olio-orientoitunutta lähestymistapaa. Koska ROL sisältää Datalogin aitona osajoukkonaan, se mahdollistaa myös predikaattien määrittelyn. ROL:n kehittynyt piirre on, että se mahdollistaa perinteisten ekstensionaaliseen informaatioon kohdistuvien kyselyiden lisäksi intensionaaliset eli kaavioon (ROL:in tapauksessa luokkarakenteeseen) kohdistuvat kyselyt. Olio-orientoituneista piirteistä ROL ei tue metodeita. [Liu, 1996]

Loppukäyttäjän kannalta deduktiivisten olio-orientoituneiden kyselykielten käyttö on vaativuustasoltaan verrattavissa deduktiivisiin kyselykieliin, jonka lisäksi käyttäjän on tunnettava olio-orientoituneista ohjelmointiparadigmoihin liittyviä peruskäsitteitä. Valmiiden konstruktioidensa avulla ne helpottavat tie-

don kuvaamista. Samalla kyselyiden kompleksisuus kuitenkin kasvaa [Järvelin and Niemi, 1999].

2.6. Seuraavan sukupolven tietojärjestelmät

Seuraavaksi esitellään piirteitä, joiden on esitetty olevan tyypillisiä seuraavan sukupolven tietojärjestelmille eli NGIS-järjestelmille (Next Generation Information Systems) [Niemi et al., 2002b]. Aiheeseen liittyvää tutkimus- ja kehitystyötä on tehty runsaasti Tampereen yliopistossa, ja tämä kehitystyö valitaan lähtökohdaksi aihealueen tarkastelulle. Tietenkin on syytä muistaa, että vasta tulevaisuus tulee näyttämään, mitkä tämän päivän kehitteillä olevista malleista ja teknologioista vastaisuudessa saavat merkittävän aseman. Parhaimmillaankin tästä voidaan antaa vain hyviä ennusteita.

RDOOM [Niemi et al., 2002b, 2004] on Tampereen yliopistossa kehitetty tietomalli, joka yhdistää toisiinsa vahvuuksia relationaalisista ja deduktiivisista olio-orientoituneista tietokantaparadigmoista. Tietomallin huomattavimmat erot edellä kuvattuun deduktiiviseen olio-orientoituneeseen tietomalliin nähden ovat olioiden välisten assosiaatioiden (suhteiden) kuvaaminen relaatioina sekä tuki deduktiivisille olio- ja assosiaatiotyypeille. Relaatioiden käyttö olioiden välisten suhteiden mallintamisessa on edullista, sillä se mahdollistaa relaatiomallille tyypilliset tehokkaat käsittelyoperaatiot, poistaa olioarvoisista attribuuteista aiheutuvan kuvauksen asymmetrisyyden sekä parantaa sen intuitiivista tulkittavuutta. Näin valittuina RDOOM-tietomallin perusprimitiivit muistuttavat läheisesti ER-mallin (kts. esim. [Ullman, 1988]) primitiivejä. ER-mallia käytetään tavallisesti juuri tietokantakaavioiden suunnitteluun ja sitä voidaan siten pitää intuitiivisena tapana kuvata tietokannan rakenne [Paton et al., 1996]. Erityisesti relaatio- ja olio-orientoituneiden tietomallien yhdistämistä on esitetty myös muissa yhteyksissä (kts. esim. [Premeriani et al., 1990; Cattell and Rogers, 1986]).

RDOOM tarjoaa loppukäyttäjälle käsitteellisesti yksinkertaisen tavan hallita sovellusalueeseen liittyvää tietoa sallimalla deduktiivisesti määriteltyjen olio- ja suhdetyyppien sekä metodien määrittelyn. Tiedon kätkentä -periaatteen mukaisesti loppukäyttäjän ei tarvitse tuntea tapaa, jolla niihin liittyvä informaatio johdetaan tietokantaan eksplisiittisesti tallennetusta tiedosta. Deduktiivisten olio- ja suhdetyyppien käsittely kyselyissä tapahtuu samaan tapaan kuin varsinaisten olio- ja suhdetyyppien. Niiden avulla voidaan relaatiotietokannoille tyypillisiin näkymiin verrattuna tarjota käyttäjälle korkeammin jalostettua, muun muassa transitiivisen sulkeumien avulla johdettavaa tietoa. [Niemi et al., 2002b, 2004]

Toinen NGIS-järjestelmien kannalta hyödyllinen piirre on tehokas tuki kompleksisten entiteettien mallintamiselle ja niihin kohdistuville kyselyille. Edellä esitetyn mukaisesti olio-orientoituneissa tietomalleissa osa-

kokonaisuussuhde esitetään tavallisesti yksisuuntaisesti olioarvoisten attribuuttien avulla. Tämä tekee navigoinnista hankalaa haluttaessa viitata olioarvoisen attribuutin arvona olevasta oliosta viittaavaan olioon. Toteutetuissa kyselykielissä käyttäjän täytyy myös yleensä osoittaa polkuilmausten avulla eksakti navigointi kahden sisäkkäisyyden eri tasoilla olevan olion välillä, minkä vuoksi viittaaminen kokonaisuuden mielivaltaisella sisäkkäisyyden tasolla olevaan osaan on vaikeaa ellei mahdotonta. [Niemi et al., 2002a]

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

Kompleksisten olioiden tapauksessa erityisen hyödyllisiä ovat kyselyt, jotka mahdollistavat ekstensionaalisen eli ilmentymätason tiedon ohella tietokannan kaavioon eli sen ekstensioonaaliseen tasoon liittyvän informaation käsittelyn. Niiden ansiosta käyttäjän ei esimerkiksi tarvitse tuntea osia esittävien olioiden täsmällistä luokkaa tai asemaa osa-kokonaisuus –hierarkiassa, vaan hän voi kyselykielen sisältämien primitiivien avulla asettaa ehtoja, jotka kohdistuvat osaolioiden intensionaalisiin ja ekstensionaalisiin piirteisiin. [Niemi et al., 2002a]

Seuraavan sukupolven tietojärjestelmissä yhdistyvät tieto- ja tekstikeskeisen informaation käsittely. Tekstikeskeisellä informaatiolla tarkoitetaan tietoa, jota ei ole organisoitu tietokannalle tyypillisen tietomallin mukaisesti. Sen esittämiseen voidaan käyttää esimerkiksi XML-syntaksia noudattavia rakenteellisia tekstidokumentteja. Sovellusalueeseen liittyvää tekstikeskeistä informaatiota voidaan käsitellä kyselyissä rinnan tietokannan intensionaalisen ja ekstensionaalisen tasoihin kuuluvan informaation kanssa. [Niemi et al., 2002a, 2004] Tähän liittyviä kysymyksiä tarkastellaan lähemmin tutkielman neljännessä luvussa.

3. Semanttisten yhteyksien selvittäminen

Tässä luvussa esitetään RDOOM:in tapaisiin (deduktiivisiin) relationaalisiin ja olio-orientoituneisiin tietomalleihin perustuvien kyselykielten laajentamista uudella piirteellä: semanttisten yhteyksien selvittämiseen tarkoitetuilla primitiiveillä. RDOOM-tietomallille tyypillistä tiedon loogista johdettavuutta eli deduktiota ei tässä yhteydessä huomioida. Ehdotamme, että tämä uusi piirre integroidaan seuraavan sukupolven tietojärjestelmien (NGIS) ominaisuudeksi.

Seuraavaksi esitellään ja johdetaan semanttisen yhteyden käsite, perustellaan semanttisten yhteyksien selvittämisen tarpeellisuutta sekä pohditaan niitä ilmaisuvoimaan ja käytettävyyteen liittyviä aspekteja, jotka tulee ottaa huomioon semanttisten yhteyksien selvittämiseen soveltuvan kyselykielen suunnittelussa. Lopuksi tarkastellaan lyhyesti niitä potentiaalisia ongelmia, jotka liittyvät tällaisiin kyselyihin. Semanttisen yhteyden käsitteen luonnehdinta aiemmasta tutkimuskontekstista esitetään puhtaasti subjektiivisen ajatuskulun perusteella. Siten se ei välttämättä ole kehityshistoriallisesti oikea tai käytännön kannalta edes relevantti johdatus aiheeseen. Tarkoitus onkin tarjota vain yksi näkökulma, joka ainakin jollain tasolla jäsentää kehiteltävää käsitteistöä suhteessa aiempaan tietämykseen.

3.1. Semanttisen yhteyden käsite ja sen soveltaminen

Tietokannan oliot ja niiden väliset assosiaatiot voidaan esittää graafiin perustuvalla esitystavalla. GOOD [Gyssens et al., 1990] on esimerkki tietomallista, jossa koko tietokanta on organisoitu graafina. GOOD tukee myös olio-orientoituneita piirteitä, ja yleisesti funktionaaliset tai olio-orientoituneet tietokannat voidaankin esittää GOOD-tietomallia käyttäen. Käytettäessä RDOOM:in tapaista relaationaalista ja olio-orientoitunutta tietomallia (RDOOM:in deduktiivisia piirteitä ei tässä huomioida) luonnollinen tapa esittää tietokannan oliot ja assosiaatiot graafina on kuvata oliot graafin solmuiksi ja assosiaatiot niiden välisiksi kaariksi. Korkeampaa astelukua olevat assosiaatiot voidaan palauttaa binäärisiksi assosiaatioiksi funktionaalisen tietomallin esittelyn yhteydessä kuvatulla tavalla.

Koska tietokannan oliot ja niiden väliset assosiaatiot voidaan mieltää graafina, voidaan tietokantaan periaatteessa soveltaa graafiteoreettisiin menetelmiin perustuvia kyselyitä. Tyypillinen graafiteoreettisten menetelmien osajoukko ovat kahden solmun välillä olevien polkujen etsimiseen tarkoitetut algoritmit. Tällaisten menetelmien soveltaminen tietokantakyselyissä ei ole uusi ajatus. Erityisesti tieverkostojen ja prosessien suunnittelun yhteydessä on kirjallisuudessa esitetty kyselykieliä, jotka mahdollistavat esimerkiksi lyhimmän kahden kaupungin välisen tiereitin etsimisen (kts. esim. [Zhao and Zaki, 1994], [Mannino and Shapiro, 1990]). Huomiotta on kuitenkin jätetty olennainen aspekti, joka laajentaa graafikyselyiden sovellettavuutta myös laajaan joukkoon muunlaisia tietokantoja: oliota edustavien solmujen välisen polun semanttinen tulkinta.

Perinteisissä graafisovelluksissa solmujen välinen polku muodostetaan yleensä vain yhden assosiaatiotyypin yli. Assosiaatiotyyppi voi kuvata esimerkiksi tieverkon osia tai prosessin vaiheita. Tällaisissa sovelluksissa kahden polulla yhdistetyn solmun etsiminen vastaa transitiivisen sulkeuman muodostamista kyseiselle assosiaatiotyypille [Mannino and Shapiro, 1990]. Transitiivisen sulkeuman muodostaminen voidaan esittää Datalog-tyylisen pseudokoodiesityksen avulla ohjelmakatkelman 1 mukaisesti. Esimerkissä oletetaan, että pre-

dikaatin `pred/2` argumentit ovat järjestelmään tallennettuja olioita (esityksessä voidaan käyttää esimerkiksi niiden oliotunnisteita) ja että kyseisellä predikaatilla ilmaistaan jokin olioiden välinen assosiaatiotyyppi.

```
1  transitive_closure(A,C):-pred(A,C).

2  transitive_closure(A,C):-
3  pred(A,B),transitive_closure(B,C).
```

Ohjelmakatkelma 1. Transitiivisen sulkeuman muodostaminen Datalog-tyylisenä pseudokoodiesityksenä.

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

```
(SELECT A1:attr2 FROM rel A1 WHERE A1:attr1 = 'start')
UNION
((SELECT B2:attr2 FROM rel B1, rel B2 WHERE B1:attr1 = 'start'
AND B1:attr2 = B2:attr1)
UNION
(SELECT C3:attr2 FROM rel C1, rel C2, rel C3 WHERE C1:attr1 =
'start' AND C1:attr2 = C2:attr1 AND C2:attr2 = C3:attr1))
```

Ohjelmakatkelma 2. SQL-kysely, joka palauttaa transitiivisen sulkeuman syvyystasolle 3 relaatiossa `rel`. Relaatiolla on attribuutit `attr1` ja `attr2`. Sulkeuman muodostaminen aloitetaan monikosta, jonka attribuutin `attr1` arvo on "`start`".

Laajennamme nyt transitiivisen sulkeuman käsitettä sallimalla assosiaatioiden käytön sulkeuman muodostamiseen niiden tyypistä riippumatta. Yleisessä tapauksessa tällaiseen sulkeuman kannalta mielenkiintoiseen assosiaatioon voi osallistua `n` oliota, missä `n` on jokin positiivinen kokonaisluku. Tällöin sulkeuma voidaan muodostaa minkä tahansa kahden samaan assosiaatioon osallistuvan olion välityksellä. Negaatiolla täydennetyn Datalogin ilmaisuvoima riittää periaatteessa tällaisen sulkeuman muodostamiseen, jos korvaamme as-

sosiaatioiden normaalin Datalog-esitystavan uudella esitystavalla seuraavan mukaisesti.

Jokainen predikaatti-ilmaus ”PredName(X_1,X_2,...,X_n)”, missä PredName on assosiaation ilmaisevan predikaatin nimi ja X_1, X_2,..., X_n ovat n kappaletta assosiaatioon osallistuvia olioita, korvataan n:llä kappaleella predikaatti-ilmauksia, jotka ovat muotoa ”pred(PredName,PredID,i,X_i)”. Argumentti PredID on tunniste, joka yksilöi yksiselitteisesti samasta alkuperäisestä predikaatti-ilmauksesta generoidut uudet predikaatti-ilmaukset. Argumentti i ($1 \leq i \leq n$) on järjestysluku, joka osoittaa argumentin X_i aseman alkuperäisessä predikaatti-ilmauksessa. Esimerkiksi fakta “f(a,b,c)” voidaan muuntaa kolmeksi uutta muotoa olevaksi faktaksi, jotka ovat “pred(f,id_1,1,a)”, ”pred(f,id_1,2,b)” ja ”pred(f,id_1,3,c)”, missä id_1 on mielivaltaisesti valittu yksilöivä tunniste. Sulkeuma voidaan muodostaa nyt ohjelmakatkelman 3 pseudokoodiesityksellä.

```
1  closure(A,B):-
2  closure(A,B,notPredID).

3  closure(A,B,_):-
4  pred(_,PredID,Ind1,A),
5  pred(_,PredID,Ind2,B),
6  not (Ind1 = Ind2).

7  closure(A,C,PrevPredID):-
8  pred(_,PredID,Ind1,A),
9  pred(_,PredID,Ind2,B),
10 not (PredId = PrevPredId),
11 not (Ind1 = Ind2),
12 closure(B,C,PredID).
```

Ohjelmakatkelma 3. Pseudokoodiesitys sulkeumalle. Argumentti notPredID on vakio, joka ei yksilöi mitään alkuperäistä predikaatti-ilmausta.

Edellä esitetty pseudokoodi voi tietyissä tilanteissa johtaa syklien muodostumiseen sekä sulkeuman muodostamiseen käytettyjen olioiden että assosiaatioiden suhteen. Evaluoitaessa sulkeumaa esimerkiksi Prologin prosessointitavan mukaisesti johtaa tämä vääjäämättä päättymättömään prosessointiin. Syklien eliminoimiseksi tarvitaan tietorakenteita, joihin kerätään tiedot olioista ja assosiaatioista, joita on jo käytetty sulkeuman muodostamiseen. Koska lisäksi olemme kiinnostuneita nimenomaan löydetystä yhteydestä, on sen tallentami-

nen johonkin rekursiivisesti toteutettuun tietorakenteeseen (esimerkiksi listaan) ainoa mielekäs vaihtoehto. Käytännössä tarvitaan siis Datalogia, jossa negaation lisäksi sallitaan funktiosymboleiden käyttö.

Edellä olemme käyttäneet olioiden välillä vallitsevista assosiaatiosta hieman epätäsmällisesti myös nimityksiä relaatio, suhde tai predikaatti. Jatkossa korvaamme nämä enemmän tai vähemmän samaa tarkoittavat ilmaukset termillä semanttinen yhteys. Välitön semanttinen yhteys ilmaisee yhden assosiaation, ja sen pituus on yksi. Edellä esitetyn sulkeuman avulla voimme etsiä olioita, jotka liittyvät toisiinsa mielivaltaisen pituuden omaavan semanttisen yhteyden välityksellä. Välilliseksi semanttiseksi yhteydeksi kutsumme semanttista yhteyttä, jossa olioiden välinen yhteys muodostuu välillisesti yhden tai useamman muun olion kautta välittömien semanttisten yhteyksien välityksellä. Termin semanttinen yhteys käyttö on tässä yhteydessä mielekästä, koska haluamme korostaa näiden yhteyksien semanttista tulkittavuutta. Käytännössä semanttinen tulkinta voidaan esittää esimerkiksi luonnollisen kielen ilmausten avulla.

Olettakaamme RDOOM-tietomallia [Niemi et al., 2002b, 2004] mukaileva tietokanta, jossa on sekä oliota että niiden välillä vallitsevia semanttisia yhteyksiä. Oliot voivat olla esimerkiksi oliotyyppien henkilö, auto tai yritys ilmentymiä. Semanttiset yhteydet voivat puolestaan pitää sisällään perhe-, omistus- ja työskentelysuhteita. Täten tietokannassa voisi vallita semanttinen yhteys Mattinimisen henkilön ja rekisterinumeron ”GRP-454”omaavan auton välillä seuraavasti: ”Matin isä työskentelee yrityksessä, jossa työskentelee myös henkilö `x`, joka omistaa kyseisen auton”.

Olemassa olevat kyselykielet eivät tue semanttisten yhteyksien selvittämistä. Kuitenkin voidaan kuvitella suuri joukko erilaisia tilanteita, joissa kyselykielen kyky vastata muotoa: ”Miten kaksi oliota liittyvät semanttisesti toisiinsa?” tai ”Mitkä oliot liittyvät annettuun olioon semanttisesti ja millaisten suhteiden kautta?” oleviin kysymyksiin voisi selvästi olla hyödyllinen. Käytännön sovelluksia voisivat olla esimerkiksi potentiaalisten intressiristiriitojen ja jääviystilanteiden selvittäminen henkilöiden välillä, onnettomuuden uhrin tuntevien henkilöiden löytäminen, ilmiöiden välisten syy- ja seuraussuhteiden paljastaminen, erilaiset rikostutkinnalliset tarkoitukset ja tutkiva journalismi.

Ohjelmakatkelman 3 pseudokoodi kaikkine puutteineenkin antaa selvän kuvan siitä, että semanttisten yhteyksien selvittämiseen kykenevän kyselyn muodostaminen olemassa olevilla deduktiivisilla kyselykielillä on huomattavasti esimerkiksi transitiivisen sulkeuman muodostamista monimutkaisempi tehtävä. Käyttäjältä edellytetään vaativien ohjelmointitekniikoiden (esim. rekursio) hallintaa, mikä on liian kova vaatimus tavalliselle loppukäyttäjälle. Tarvitaan siis kyselykieltä, joka mahdollistaa semanttisten yhteyksien selvittämisen loppukäyttäjän kannalta helpolla ja deklaratiivisella tavalla.

3.2. Semanttisten yhteyksien selvittämiseen soveltuvan kyselykielen erityispiirteet ja ongelmat

Perinteiset tietokantojen kyselykielet mahdollistavat vain kyselyt, jotka palauttavat tuloksena tietokannan ekstensioon eli sen ilmentymätasoon liittyvää informaatiota. Seuraavan sukupolven tietojärjestelmissä kysely voi palauttaa myös tietokannan intensioon eli kaavioon liittyvää informaatiota. Semanttisten yhteyksien käsittelyn yhteydessä nämä kaksi vastaustyyppiä sekoittuvat toisiinsa. Toisaalta semanttinen yhteys sisältää tietokannan ekstensionaaliselle tasolle kuuluvaa informaatiota (välittömiin semanttisiin yhteyksiin osallistuvat oliot) ja toisaalta intensionaalista informaatiota (välittömien osayhteyksien tyypit).

Myös loppukäyttäjältä vaadittavan tietokannan kaavion tuntemuksen kannalta semanttisia yhteyksiä selvittävät kyselyt ovat mielenkiintoisia. Perinteisissä kyselykielissä, esimerkiksi SQL:ssä, kysely voidaan ymmärtää intensionaalisena spesifikaationa kyselyn tuloksena haluttavasta tiedosta [Motro, 1994]. Voidakseen muodostaa kyselyjä tällaisella kyselykielellä käyttäjän on siis aina tunnettava täysin kyselyiden kannalta relevantit kaavion osat. Semanttisia yhteyksiä muodostettaessa käyttäjän ei tarvitse periaatteessa tuntea kaaviota lainkaan. Tällainen ääritapausta edustava kysely vastaa tietokannan kaikkien semanttisten yhteyksien palauttamista, mikä ei kuitenkaan käytännössä ole relevantti kyselytyyppi. Yleisessä tapauksessa käyttäjä voi tuntea joko yksilöivästi tai vain joiltain piirteiltään oliot, joiden välisistä yhteyksistä hän on kiinnostunut. Lisäksi mahdollista on, että käyttäjä antaa rajoituksia, jotka liittyvät yhteyden pituuteen, sen muodostamisessa käytettyihin yhteystyyppeihin jne. Yhteenvetona voidaan siis todeta, että semanttisten yhteyksien selvittämistä tukevassa kyselykielessä käyttäjän ei tarvitse tuntea eksplisiittistä navigointipolkua kahden olion välillä, vaan tämä navigointipolku (so. semanttinen yhteys) siihen liitettävine tulkintoineen on itsessään kyselyn tulos.

On huomattava, että RDOOM:in [Niemi et al., 2002b, 2004] kaltainen (deduktiivinen) relationaalinen ja olio-orientoitunut tietomalli tarjoaa semanttisten yhteyksien selvittämiselle luonnollisen lähtökohdan. Tavallisestihan olemme kiinnostuneita juuri olioiden välisistä yhteyksistä; emme esimerkiksi kahden assosiaation välisestä yhteysketjusta. Esimerkiksi relaatiomallissa semanttisten yhteyksien automaattinen selvittäminen ei ole mahdollista, koska siinä relaatioiden välistä navigaatioita ei ole eksplisiittisesti määritelty, vaan yhteys voitaisiin periaatteessa muodostaa minkä tahansa kahden eri monikkoon kuuluvan samanarvoisen attribuutin kautta.[1] Myös relationaaliseen olio-orientoituneeseen mallintamistapaan perustuvissa järjestelmissä voi esiintyä

[1] Voitaisiin esimerkiksi muodostaa yhteys kaksivuotiaan lapsen ja kaksi euroa maksavan tuotteen välille vain sillä perusteella, että niiden ikä- ja hinta-attribuuttien arvot ovat samat.

tilanteita, joissa semanttinen yhteys voidaan mielekkäällä tavalla muodostaa muuten kuin olioiden välisten yhteyksien perusteella. Tällainen tilanne syntyy esimerkiksi, jos asuinkunta on kuvattu henkilön merkkijonoarvoiseksi attribuutiksi. Tällöin kahden henkilön välillä voitaisiin esittää semanttinen yhteys ”asuvat samalla paikkakunnalla” asuinkunta-attribuuttien arvojen samuuden perusteella (vrt. esim. [Bertino et al., 1992]). RDOOM-tietomallissa tämä voitaisiin välttää määrittelemällä uusi deduktiivinen assosiaatiotyyppi, joka perustuu kyseisten attribuuttien arvojen vertailuun. Suoraviivaisin vaihtoehto on jo tietokannan kaavion suunnittelun yhteydessä esittää kaikki vaihtoehtoiset navigointipolut olioiden välisinä eksplisiittisinä semanttisina yhteyksinä. Esimerkkitapauksessa tämä merkitsisi, että kunta määriteltäisiin oliotyypiksi ja asuminen semanttiseksi yhteystyypiksi henkilöiden ja kuntien välillä.

Keskeisin ongelma semanttisten yhteyksien muodostamisessa on niiden potentiaalinen suuri määrä. Jos oletamme, että tietokannassa on 3 oliota, joista jokainen on yhdistetty jokaiseen toiseen yhdellä välittömällä binäärisellä yhteydellä, on mielivaltaisen kahden olion välisten erilaisten syklittömien yhteyksien määrä 2. Vastaavasti olioiden määrän ollessa 5 erilaisia yhteyksiä on vastaavassa tapauksessa 16. Kuitenkin kahdeksan tällaisen olion tapauksessa erilaisten polkujen määrä lähestyy jo kahta tuhatta (1957)[1]. On siis selvää, että tällaisessa tilanteessa käyttäjän ei kannata tutkia koko vastausta. Käytännössä voitaneen kuitenkin olettaa, että useissa käytännön sovelluksissa yhteystiheys[2] ei ole näin suuri ja että käyttäjän semanttisten yhteyksien muodostamiselle antamat ehdot karsivat ainakin osan mahdollisista yhteyksistä. Todennäköistä on myös, että käyttäjä ei ole edes kiinnostunut kaikista mahdollisista olioiden välillä vallitsevista semanttisista yhteyksistä. Selattuaan muutaman joukon alkupään tuloksen käyttäjä voi antaa lisäehtoja, jotka parantavat hakutuloksen tarkkuutta ja karsivat epärelevanttien yhteyksien määrää.

Tämä kombinatoriikasta johtuva yhteysmäärän potentiaalinen suuruus koskee vain semanttisten yhteyksien lineaarista esitysmuotoa. Puhtaasti matemaattisesti ajatellen luontaisempi esitysmuoto tällaisen kyselyn tulokselle olisikin jokin graafia muistuttava esitystapa, jossa kaaret esittäisivät solmuja vastaavien olioiden välisiä välittömiä semanttisia yhteyksiä. Kuvaavaa on, että edellisen kahdeksan välittömästi toisiinsa yhdistetyn olion tapauksessa vastaus voitaisiin kuvata yhden kahdeksasta solmusta ja 28 kaaresta muodostuvan

---

[1] Jos jokainen olio on yhdistetty jokaiseen muuhun olioon yhdellä välittömällä binäärisellä yhteydellä ja olioiden määrä on n, saadaan mielivaltaisen kahden olion välisten syklittömien yhteyksien lukumäärä laskemalla yhteen (n – 2):n alkion joukosta otettujen 1, 2, …, (n – 2) -alkioisten variaatioiden lukumäärä, mihin lisätään yksi (olioiden välinen välitön yhteys).

[2] Yhteystiheydellä tarkoitetaan tässä välittömien semanttisten yhteyksien lukumäärää suhteutettuna olioiden lukumäärään.

graafiesityksen avulla. Tulevissa käytännön sovelluksissa tällainen graafinen esitysmuoto, mahdollisesti luonnollisella kielellä rikastettuna, saattaakin siis olla varteenotettava vaihtoehto.

Loppukäyttäjän kannalta olisi ideaalista, että järjestelmä osaisi tehdä päätelmiä semanttisten yhteyksien relevanttiudesta ja esimerkiksi järjestäisi vastauksen siihen perustuen. Yksinkertainen päättelysääntö voisi olla esimerkiksi, että lyhyempi semanttinen yhteys kuvastaa yleensä läheisempää ja siten vastauksen kannalta kiinnostavampaa suhdetta kuin pidempi yhteys. Valtaosassa tapauksia tämä ehkä pitääkin paikkaansa, mutta toisaalta vastaesimerkkien keksiminen on yksinkertaista. Kahden sisaruksen välinen suhde voi muodostua esimerkiksi kahden vanhemmuutta ilmaisevan välittömän osayhteyden välityksellä, kun taas tosille sisarista tammikuussa 2001 autonrenkaita myynyt henkilö voi liittyä tähän yhden välittömän semanttisen yhteyden välityksellä. Toisaalta kaksi henkilöä, joilla on vakuutuksia samassa vakuutusyhtiössä, eivät välttämättä liity toisiinsa siinä määrin läheisesti, että kolmansien olioiden välisiä semanttisia yhteyksiä kannattaisi ainakaan kaikissa tapauksissa muodostaa tämän osayhteyden kautta. Potentiaalisen sovellusalueen monialaisuudesta johtuen tällaisten päättelymenetelmien kehittäminen lienee varsin haasteellinen tehtävä. Tämän vuoksi jätämme tämän ongelma-alueen tutkielman ulkopuolelle. Tyydymme korostamaan, että toteutettavan kyselykielen on tarjottava käyttäjälle riittävät välineet, joilla hän voi karsia vastausjoukosta epärelevantteja yhteyksiä.

4. XML-muotoisen tiedon teksti-orientoitunut käsittely

XML (Extensible Markup Language) on W3C-konsortion standardiin [XML, 2004] pohjautuva, laajasti käytetty rakenteellisten dokumenttien merkintäkieli, joka määrittelee dokumenteille yhtenäisen, sovellusalueesta riippumattoman syntaksin. Toisin kuin HTML, XML on tarkoitettu ensisijaisesti tiedon loogisen rakenteen, ei sen ulkoisen esitysmuodon, määrittelyyn. XML-spesifikaation pohjalta voidaan muodostaa sovellusaluekohtaisia merkintäkieliä, jotka rajoittavat sovelluksen kannalta validien XML-dokumenttien määrää ja joihin yleensä liittyy jokin ennalta määritelty semantiikka. Merkintäkieltä vastaava XML-kielioppi voidaan määrittää joko W3C-konsortion XML-suositukseen perustuvalla DTD-esityksellä tai sitä monipuolisemmalla XML Schema –määrittelyllä [XML Schema, 2001].

XML-esitystavassa dokumentti muodostuu nimettyjen elementtien muodostamasta puumaisesta hierarkiasta, jonka ylin taso on juurielementti. Elementin sisältö voi olla tyhjä tai se voi koostua yhdestä tai useammasta lapsielementistä, rakenteettomasta tekstistä tai näiden erilaisista yhdistelmistä. Tämän lisäksi elementtiin voidaan liittää attribuutteja, jotka ovat attribuutin nimestä ja sen arvosta muodostuvia pareja. Useissa tapauksissa sama tieto voi-

taisiin syntaktisesti esittää joko lapsielementtien tai attribuuttiesityksen avulla. Attribuutin arvo on aina puhdasta merkkijonomuotoista dataa, joten rakenteellisen tiedon esittäminen ei attribuuttiesityksessä kuitenkaan ole mahdollista. Rakenteettoman tekstijakson ”sisältö” sisältävän elementin elementti, jolla on attribuutti attribuutti arvonaan ”arvo”, esitysmuoto XML-syntaksissa on ”<elementti attribuutti=”arvo”>sisältö</elementti>”. Tyhjän elementin tapauksessa erillisiä alku- ja loppumerkkejä ei tarvita, vaan nämä voidaan korvata merkinnällä ”<elementti/>”. [XML, 2004]

XML-spesifikaatio määrittelee lisäksi erityisiä syntaktisia rakenteita: kommentteja, prossessointiohjeita, entiteettiviittauksia- ja määrittelyjä, sisällöstään riippumatta rakenteettomana tekstinä tulkittavia CDATA-jaksoja sekä dokumenttityyppimäärittelyjä [XML, 2004]. Näillä ei kuitenkaan ole itse XML-esitystavan rakenteellisuuden kannalta keskeistä merkitystä, joten sivuutamme ne tässä. Todettakoon, että yksinkertaisimmillaan spesifikaation mukainen hyvinmääritelty XML-dokumentti muodostuu yhdestä juurielementistä sisältöineen ja attribuutteineen.

Tässä luvussa esittelemme ensin lyhyesti W3C-konsortion kehittämät XML-muotoisen tiedon kyselyyn tarkoitetut kyselykielet; XPathin, XSLT:n ja XQueryn; minkä jälkeen käsittelemme XML-kyselyominaisuuksien integrointia osaksi varsinaista tietokannan kyselykieltä.

4.1. XML-dokumenttien kyselyyn tarkoitetut kyselykielet

XPath (XML Path Language) [XPath, 1999] perustuu W3C-konsortion suositukseen. XPath on tarkasteltavista kielistä suppein. Se sisältyy osana sekä XSLT:hen että XQueryyn. XPath-ilmaukset ovat XML-dokumenttien loogista rakennetta kuvaavia polkuja, joilla voidaan viitata dokumenteissa esiintyviin yksittäisiin elementteihin, rakenteettoman tekstin jaksoihin ja attribuutteihin. Lisäksi esimerkiksi kommenttien ja prosessointiohjeiden käsittely on XPathin avulla mahdollista, vaikka ne eivät varsinaisesti kuulukaan itse dokumenttien XML-tietosisältöön.

XPath-syntaksi muistuttaa tiedostojärjestelmän hakemistohierarkian esittämiseen käytettyä polkumerkintää. Juurielementtiä merkitään symbolilla ”/”. Juurielementin välittömään alielementtiin eli lapsielementtiin lapsi viitataan merkinnällä ”/lapsi”. Polkuilmausta voidaan kasvattaa iteratiivisesti lisäämällä uusia syvyystasoja; esimerkiksi lapsielementin lapsi lapsielementtiin lapsenlapsi viitataan ilmauksella ”/lapsi/lapsenlapsi”. Kun polku edellisten tapaan alkaa symbolilla ”/”, on kyseessä absoluuttinen polku, jossa polun evaluointi aloitetaan dokumentin juuresta. Jos taas polku alkaa elementin nimellä, on kyseessä suhteellinen polku, jonka evaluointi aloitetaan prosessointikontekstissa aktiivisena olevalta tasolta.

Merkinnällä ”//” viitataan mielivaltaisella syvyystasolla olevaan elementtiin. Niinpä esimerkiksi ilmaus ”//elementti” viittaa dokumentissa millä tahansa syvyystasolla olevaan elementtiin, jonka nimi on elementti. Sama polku voi sisältää sekä ”/”, että ”//”-operaattoreilla erotettuja elementtejä.

Jos elementti sisältää useita samannimisiä lapsielementtejä, voidaan haluttu lapsielementti määrittää liittämällä sen nimen perään järjestysluku hakasulkeissa. Täten esimerkiksi ilmaus ”/elementti/alielementti[2]” viittaa elementti-nimisen elementin toiseen alielementti-nimiseen alielementtiin. Järjestyslukujen asemesta voidaan käyttää myös esimerkiksi funktiokutsuja ”first()” tai ”last()”, joista ensimmäinen palauttaa järjestyksessä ensimmäisen ja jälkimmäinen järjestyksessä viimeisen elementin. Näiden ohella XPath tarjoaa koko joukon muita elementtien keskinäiseen sijaintiin liittyviä funktioita. Erityisesti on huomattava, että polussa mikä tahansa elementin nimi voidaan korvata symbolilla ”*”. Tämän vuoksi ilmaus ”/*” viittaa kaikkiin juurielementin välittömiin lapsielementteihin ja ilmaus ”//*” mihin tahansa dokumentissa esiintyvään elementtiin.

Elementin attribuutti voidaan palauttaa käyttämällä ”/”-merkin sijasta elementin ja attribuutin nimen välisenä erottimena ”@”-merkkiä. Esimerkiksi ilmaus ”/elementti@attribuutti” palauttaa elementin elementti attribuutin attribuutti. Hakasuluissa annettu funktiokutsu ”text()” liitettynä elementin nimen perään palauttaa kaikki elementin sisältämät rakenteettoman tekstin jaksot.

XPathin kannalta XML:n kaikki rakenteelliset osat; elementit, attribuutit, rakenteeton teksti jne.; ovat solmuja, joille voidaan antaa tekstuaalinen esitys. Attribuuttien tapauksessa tekstuaalisen esityksen muodostaa attribuutin arvo ja rakenteettoman tekstin tapauksessa teksti itse. Elementtien tekstuaalinen esitys muodostuu puolestaan rekursiivisesti niiden sisällön tekstuaalisesta esityksestä. Tekstuaalinen esitys voidaan tarvittaessa konvertoida numeromuotoon. [XPath, 1999] Tämän ansiosta solmuihin voidaan kohdistaa useita teksti- ja numeromuotoisen datan käsittelyyn tarkoitettuja funktioita ja operaattoreita, joilla voidaan esimerkiksi tarkistaa osamerkkijonon sisältyminen merkkijonoon ja suorittaa aritmeettisia vertailuja. XPathin tekstuaalisen ja numeromuotoisen datan käsittelyyn tarkoitetut funktiot laajentavat XPathin ilmaisuvoimaa, joka muuten rajoittuisi yksin XML-dokumentin rakenteelliseen analyysiin. Suhteellisen köyhä syntaksi puuttuvine ohjausrakenteineen kuitenkin rajoittaa XPathin käytännön sovellettavuutta.

Myös XSLT (Extensible Stylesheet Language Transformations) [XSLT, 2003] perustuu W3C-konsortion suositukseen. Toisin kuin XPath, XSLT noudattaa itsessään XML-syntaksia. XSLT on ensisijaisesti tarkoitettu tekemään rakenteellisia muutoksia XML-dokumentissa. Vaikka XSLT on siis luonteeltaan enemmän uudelleenstrukturointikieli kuin kyselykieli, voidaan muunnos näh-

dä myös halutun kyselytuloksen esitysmuodon ja johtamistavan määrittämisenä. Tämän vuoksi myös XSLT:tä voidaan käyttää XML-dokumenttien kyselykielenä.

XSLT-muunnos ilmaistaan ”xsl:stylesheet” tai ”xsl:transform”-elementtien sisällä määriteltyinä malleina. Mallit ovat ”xsl:template”-elementtejä, joiden match-attribuutin arvona olevaa XPath-lauseketta sovitetaan muunnoksen kohteena oleviin XML-dokumentteihin. Mikäli malliin sopiva elementti löydetään, siihen sovelletaan mallin sisältämiä sääntöjä. Säännöt voivat olla XML-elementteinä ilmaistavia proseduraalisille ohjelmointikielille tyypillisiä ohjausrakenteita, eksplisiittisesti annettuja vakiomuotoisina tulostettavia rakenteita, esimerkiksi HTML:n head-elementin aloitusmerkki, tai ”xsl:value-of”-elementtejä, joiden select-attribuuttien arvoina olevilla, tavallisesti suhteellisilla, XPath-poluilla valitaan kohdedokumenteista kokonaisina kopioitavia rakenteellisia tai rakenteettomia osia. [XSLT, 2003]

XSLT:n ongelmia yleiskäyttöisenä XML-dokumenttien kyselykielenä ovat sen vahva keskittyminen uudelleenstrukturoinnin määrittelyyn, XML-elementteinä ilmaistavien tulosteessa näkymättömien ohjaus- ja siinä näkyvien tulosterakenteiden rinnakkainen käyttö sekä kielen vahva proseduraalinen orientaatio, joka edellyttää käyttäjältään ohjelmointikokemusta. Myöskään kokeneempien käyttäjien kannalta XML-syntaksi ei välttämättä ole optimaalinen tai edes selkeä tapa ilmaista proseduraalisia rakenteita. [Kumpulainen, 2003]

Kolmas tässä luvussa tarkasteltava W3C-konsortiossa kehitetty XML-dokumenttien kyselykieli on XQuery [XQuery, 2003]. XQuery laajentaa XSLT:n tapaan XPathia monipuolisilla ohjausrakenteilla ja mahdollisuudella määritellä tuloksen ulkoasu. XSLT:stä poiketen sillä on luettavaksi tarkoitettu muoto, joka ei perustu XML-syntaksiin. Tämän muodon sijasta voidaan kuitenkin käyttää myös ohjelmallisesti prosessoitavaksi tarkoitettua XML-esitystä.

XQuery perustuu samaan tietomalliin kuin XPathin versio 2.0. Täten myös XQueryssä XML-dokumentti kuvautuu puumaiseksi solmujen hierarkiaksi, jossa solmuja vastaavat elementit, niiden attribuutit, rakenteettomat tekstijaksot, kommentit, prosessointiohjeet jne.

XQueryn keskeinen laajennos XPathiin nähden on SQL:n syntaksia mukaileva FLWOR-ohjausrakenne (for, let, where, order by, return). Ohjausrakenteen for-osaa voidaan käyttää iteroimaan annettuihin XPath-lausekkeisiin sopivia elementtejä, jotka samassa yhteydessä sidotaan muuttujien arvoiksi. Tämän asemesta tai ohella voidaan käyttää let-osaa, joka on toiminnallisesti vastaava, mutta arvottaa muuttujat ei-iteratiivisesti. Rakenteen valinnaisesssa where-osassa sidottuihin muuttujiin kohdistetaan SQL:n tapaan lisäehtoja. Ilmausta ”order by” käytetään järjestämään tulosjoukko halutun muuttujan tai siitä riippuvan suhteellisen polkuilmauksen arvon mukaiseen järjestykseen. Kyselyn

tulosasu muotoillaan `return`-osassa, ja se voi XSLT:n tapaan olla rakenteellinen. [XQuery, 2003]

4.2. XML-kyselyominaisuuksien integrointi tietokannan kyselykieleen

XML-dokumentti voi olla dokumentti- tai tietokeskeinen riippuen sen määrittelyyn soveltuvien kielioppisääntöjen tiukkuudesta. Dokumenttikeskeisessä esitystavassa samaa kielioppia noudattavat dokumentit voivat erota toisistaan rakenteellisesti merkittävästi. Niissä XML-syntaksin tehtävä on korostaa esimerkiksi luonnollisella kielellä esitetyn dokumentin loogisia rakenneosia. Esimerkkinä voidaan ajatella tieteellistä artikkelia, jossa otsikot, kappaleet ja lähdeluettelo ilmaistaan kaikki erillisten XML-elementtien avulla. Sitä vastoin tietokeskeisissä kieliopeissa kaikki kielioppia noudattavat dokumentit ovat rakenteellisesti homogeenisia. Tämä merkitsee, että yksittäinen dokumentti muistuttaa läheisesti tietokannan tietuetta, jossa tiedon esityksessä käytetyt kentät on määritelty etukäteen. Ilmeisiä sovellusalueita tietokeskeisille XML-kieliopeille ovat esimerkiksi ostolaskut, vahinkoilmoitukset tai muut määrämuotoiset dokumentit. [Fuhr and Großjohann, 2001]

XML-syntaksin joustavuuden ansiosta ero dokumenttikokoelmien ja tietokantojen välillä on hämärtynyt [Chamberlin et al., 2000]. Usein sekä tieto- että dokumenttikeskeinen aineisto esiintyy rinnan samalla sovellusalueella (vrt. esim. [Niemi et al., 2002a]), jolloin tarve tietojen tehokkaaseen hallintaan ja monipuolisiin kyselymahdollisuuksiin kohdistuu yhtä lailla molempiin dokumenttityyppeihin. Standardimuotoisten, tietokeskeisten dokumenttien osalta tarve voidaan tyydyttää tallentamalla dokumenttien tiedot tietokantaan, joka tukee XML-esitysmuotoa. Tällöin kyselyvälineinä voidaan käyttää perinteisiä tietokantojen kyselykieliä, esimerkiksi SQL:ää. Tällainen ratkaisu ei kuitenkaan sovellu kompleksisten ja laajoja rakenteellisen tekstin jaksoja sisältävien XML-dokumenttien hallintaan. Muiden dokumenttien osalta onkin mahdollisesti käytetty erillisiä XPathin tai XSLT:n tapaisia XML-muotoisen tiedon kyselyyn tarkoitettuja kyselykieliä tai tiedonhaku- (IR) järjestelmiä. [Salminen and Tompa, 2001; Jagadish et al., 2002]

Vaihtoehtona XML:ää tukeville perinteisille tietokannoille on esitetty XML-dokumenttien tallentamiseen soveltuvaa XML-dokumenttitietokantaa. XML-dokumenttitietokannassa kaikki tieto taltioidaan ja luetaan XML-muodossa. XML-dokumenttitietokannan sijasta käytetään usein termiä XML-tietokanta, joka on kuitenkin syytä erottaa edellä kuvatun mukaisesta XML:ää tukevasta perinteisestä tietokannasta. Toisin kuin perinteisissä tietokannoissa, tällaisessa tietokannassa tiedon perusyksiköitä eivät ole relaatiot tai oliot vaan XML-dokumentit. Kyseessä on siis erityinen XML:lle ominainen tietomalli, jonka loogiset rakenteet perustuvat XML-spesifikaatiossa määriteltyyn XML-esitystapaan. [Salminen and Tompa, 2001; Kay, 2003]

Ainakin toistaiseksi teknisenä ongelmana on se, että mitään yhtenäistä XML-tietomallia ei ole hyväksytty. Jopa W3C-konsortio ylläpitää useita eritasoisia tietomalleja, joista mainittakoon XSLT:n, XPath 2.0:n ja XQueryn tietomallit sekä DOM-malli (Document Object Model). Mallit eroavat toisistaan muun muassa sen suhteen, mitä XML:n rakenteellisia osia niissä huomioidaan. Lisäksi vain XPath 2.0:n ja XQueryn tietomalli tukee jo mallin tasolla kokonaisen dokumenttikokoelman esittämistä, ei vain yksittäisen dokumentin muodostamaa puumaista hierarkiaa. [Salminen and Tompa, 2001]

Potentiaalisen XML-tietomallin heikkous useimpiin aiemmin esitettyihin tietokantaparadigmoihin perustuviin tietomalleihin verrattuna on monien keskeisten semanttisten konstruktioiden puute. Sinällään XML-esitys on esimerkiksi relaatiomallia ilmaisuvoimaisempi; voidaanhan relaatiomallin mukaisille relaatiotauluille antaa aina yksinkertainen XML-esitys, mutta toisin kuin relaatiomallilla, XML:llä voidaan lisäksi helposti mallintaa esimerkiksi kompleksisia, useita hierarkiatasoja sisältäviä entiteettejä. Sama pätee lähes mihin tahansa muuhun tietomalliin. Jopa tutkielmassa käytettävälle relationaaliselle ja olio-orientoituneelle tietomallille annetaan jäljempänä XML-esitys.

Ongelmana on kuitenkin, että XML-tietomalli ei loogisella tasolla tue esimerkiksi olioita, suhteita, olio- ja suhdetyyppejä, periytymishierarkioita tai muita reaalimaailman mallintamisen kannalta hyödyllisiä käsitteellisiä konstruktioita. Se ei myöskään tue tiedon johdettavuutta, joka on tyypillistä deduktiivisiille ja deduktiivisille olio-orientoituneille tietokannoille. Tämän vuoksi myöskään pelkkään XML-tietomalliin perustuva kyselykieli ei voi näitä piirteitä suoraan tukea. Vaikka XML-tietomalli soveltuukin hyvin rakenteellisten dokumenttien kuvaamiseen, on se reaalimaailman ilmiöiden kuvaamisen kannalta ontologisesti varsin heikko. Ontologisesti monipuolisemman tietomallin simulointi tällaisessa mallissa edellyttää, että käyttäjä tuntee yksityiskohtia tiedon fyysisestä tallennus- ja esitystavasta sekä käytetystä XML-kieliopista ja osaa käyttää tätä tietoa hyväkseen kyselynmuodostuksessa. (vrt. [Kay, 2003]) Kaikissa tapauksissa monipuolisemmalle mallille ei tietenkään ole tarvetta, jolloin tämä ei muodostu käyttäjän kannalta ongelmaksi.

Vaihtoehtoinen tässä tutkielmassa esiteltävä lähestymistapa on integroida samaan kyselykieleen kaksi erilaista tietomallia: perinteinen tietokannan tietomalli tietokantamaisten ja XML-tietomalli muiden dokumenttien hallintaan. Tällainen ratkaisumalli voisi periaatteessa olla toimiva paitsi yksittäisen XML-dokumenttitietokannan tapauksessa myös järjestelmissä, joissa yksittäisen toimijan ja tietojärjestelmän lisäksi on käytössä usean toimijan välinen avoin ympäristö, jossa tietojärjestelmän kannalta tuntemattomat mutta kuitenkin sen saavutettavissa olevat dokumentit voivat fyysisesti sijaita eri kohteissa. Toteutustavasta riippumatta jatkossa tietokannalla viitataan tietoon, joka on tallennettu tietokannan tietomallia noudattaville XML-dokumenteille, ja vapaamuo-

toisilla dokumenteilla muihin järjestelmään tallennettuihin tai järjestelmän kautta saatavilla oleviin dokumentteihin.

Ratkaisun etuna on mahdollisuus yhdistää samassa kyselyssä tieto- ja rakenteellisten tekstidokumenttien tekstiorientoitunut käsittely. Tutkielmassa myöhemmin esiteltävässä kyselykielessä käyttäjällä on tarkoitukseen suunnitellun primitiivin avulla mahdollisuus etsiä annettuun olioon liittyviä vapaamuotoisia dokumentteja tai kääntäen annettuun dokumenttiin liittyviä tietokannan olioita. Tämä tekee mahdolliseksi tietokantamaisten kyselyiden laajentamisen vapaamuotoisten dokumenttien rakenteeseen ja tekstisisältöön liittyvillä ehdoilla ja toisaalta vapaamuotoisiin dokumentteihin kohdistuvien kyselyiden laajentamisen tietokannan tietosisältöön liittyvillä ehdoilla. Kahden erillisen järjestelmän tapauksessa tällaisen interaktion toteuttaminen olisi käyttäjän kannalta varsin työlästä.

Dokumenttien rakenteeseen kohdistuvien ehtojen käsittely voidaan mahdollistaa sallimalla esimerkiksi XPath-tyylisten deklaratiivisten ilmausten käyttö osana tietokantakyselyä. XSLT:n tapainen XML-syntaksia noudattava formaatti ei tässä tapauksessa tule kysymykseen. Myös XQueryn ilmaisumuoto on tarkoitukseen nähden liian raskas ja proseduraalisia piirteitä omaava. Periaatteessa järjestelmään voitaisiin integroida myös tiedonhakujärjestelmille ominainen dokumenttien relevanssin arviointi tavallisesti niissä esiintyvien avainsanojen suhteellisen frekvenssin avulla, mistä XML-muotoisen tiedon yhteydessä ovat esimerkkeinä TIX-algebra [Khalifa et al., 2003] ja XIRQL-kyselykieli [Fuhr and Großjohann, 2001]. XML-dokumenttien hierarkkisen rakenteen vuoksi tämä on kuitenkin ongelmallista. Tässä yhteydessä tyydymme avainsanojen ja fraasien hakuun dokumenteista boolean filter –periaatteen mukaisesti.

Myös RDOOM-tietomallia on laajennettu dokumentin käsitteellä. Olioiden ohella dokumentteja voi liittyä myös mm. tietomallin intensionaalisiin primitiiveihin, esimerkiksi oliotyyppeihin. [Niemi et., 2002a, 2004]

5. Käytetty tiedon organisointitapa

Tutkielman kyselykieli ja esimerkkitietokanta perustuvat RDOOM-tietomallia läheisesti muistuttavaan tiedon loogiseen organisointitapaan. RDOOM-tietomallin deduktiivisia piirteitä ei ole siinä otettu huomioon. Assosiaatiotyyppiä vastaa käytetyssä organisointitavassa semanttinen yhteystyyppi ja yksittäisiä assosiaatioita välittömät semanttiset yhteydet. Semanttisten yhteyksien käsittely tapahtuu RDOOM-tietomallin assosiaatioiden tapaan arvo-orientoituneesti, kun taas olioita käsitellään olio-orientoituneesti (Eräiltä osin käsittely vastaa arvo-orientoitunutta lähestymistapaa. – kts. kohta 5.1.). Jokaiselle semanttiselle yhteystyypille on määritelty luonnollisella kielellä esitettävä semanttinen tulkinta, jota sovelletaan yhteystyyppiin kuuluviin välittömiin semanttisiin yhteyksiin. Koska tieto-orientoituneiden piirteiden ohella haluamme

integroida järjestelmään vapaamuotoisten olioihin liittyvien XML-dokumenttien tekstiorientoituneen käsittelyn, muodostavat tällaiset vapaamuotoiset dokumentit yhden tiedon organisointitavan loogisista rakenneosista.

Fyysisesti loogista organisointitapaa noudattava tieto esitetään XML-syntaksia noudattavalla merkintätavalla. Myös tiedon fyysinen organisointitapa esitetään tässä luvussa.

5.1. Tiedon looginen organisointitapa

Tutkielmassa esitettävän tiedon organisointitavan loogisia rakenneosia ovat välittömät semanttiset yhteydet, semanttiset yhteystyypit, oliot, oliotyypit sekä vapaamuotoiset dokumentit. Näistä tietokannan ekstensionaaliseen eli ilmentymätasoon kuuluvat välittömät semanttiset yhteydet ja oliot. Intensionaaliseen eli kaaviotasoon kuuluvat olio- ja yhteystyypit. Tutkielmassa vapaamuotoiset dokumentit voivat liittyä vain olioihin ts. tietokannan ilmentymätasoon. Myös välittömiin semanttisiin yhteyksiin sekä olio- ja assosiaatiotyyppeihin voidaan periaatteessa liittää dokumentteja, mutta esimerkkijärjestelmän laajuuden huomioon ottaen tämä mahdollisuus on rajattu käsittelyn ulkopuolelle.

Tiedon loogista organisointitapaa havainnollistaa taulukko 1. Taulukon ensimmäisessä sarakkeessa esitetään ilmentymätasoon liittyvien loogisten rakenneosien keskinäiset suhteet. (Myös olioihin liittyvät vapaamuotoiset dokumentit esitetään siinä, vaikka ne eivät tarkasti ottaen kuulu itse tietokannan ilmentymätasoon.) Toisessa sarakkeessa esitetään sellaiset suhteet , joihin osallistuu sekä ilmentymä- että kaaviotasoon kuuluvia rakenneosia. Kaaviotasoon kuuluvien rakenneosien keskinäiset suhteet esitetään kolmannessa sarakkeessa.

tarkastelu-taso / suhde	**ilmentymätaso (ja dokumentit)**	**ilmentymä- ja kaaviotaso**	**kaaviotaso**
IS-A -suhde / sisältymis-suhde		olio kuuluu oliotyyppiin ja oliotyypin jokaiseen ylätyyppiin	oliotyyppi on toisen oliotyypin yleistys ts. ylätyyppi / erikoistus ts. alityyppi
		välitön semanttinen yhteys kuuluu semanttiseen yhteystyyppiin	
assosiaatio	olio on semanttisessa yhteydessä toiseen olioon		oliotyyppi voi osallistua semanttiseen yhteystyyppiin jonkin toisen oliotyypin kanssa
viittaus	vapaamuotoinen dokumentti sisältää viittauksen olioon		

Taulukko 1. Tiedon loogiseen organisointitapaan sisältyvien loogisten rakenneosien väliset suhteet.

Taulukosta ilmenee, että ilmentymätasolla olio voi olla semanttisesti yhteydessä toiseen olioon tai siihen voi olla viittaus vapaamuotoisessa dokumentissa. Ilmentymä- ja kaaviotaso ovat vuorovaikutuksessa tapauksissa, joissa olio kuuluu oliotyyppiin tai välitön semanttinen yhteys semanttiseen yhteystyyppiin. Vastaavasti kaaviotason ilmiöitä ovat oliotyyppien välinen is-a –hierarkia sekä oliotyypin osallistuminen (rooli) semanttiseen yhteystyyppiin. Part-of –suhteiden käsittely (kts. [Niemi et al. 2002a]) on rajattu tutkielman ulkopuolelle. Seuraavassa esitellään yksityiskohtaisesti tiedon loogiset rakenneosat ja niiden keskinäiset suhteet.

Oliotyyppi määrittelee sen ilmentymille ominaisten attribuuttien joukon. Attribuuteilla tarkoitetaan tässä tarkasti ottaen niiden nimiä, sillä attribuuttien arvoja ei mallissa voida käsitellä muualla kuin ilmentymätasolla. Koska oliotyypit on järjestetty periytymishierarkiaan, perii oliotyyppi myös kaikki sen välittömille tai ei-välittömille ylätyypeille kuuluvat attribuutit. Mallissa oliotyypillä voi olla useita välittömiä ylätyyppejä, joten se tukee moniperiytyvyyttä. Periytymishierarkian huipulla mallissa on oliotyyppi `object`, joka on jokaisen muun oliotyypin ylätyyppi. Se ei sisällä omia määriteltyjä attribuutteja.

Traditionaalisen olio-orientaation vastaisesti jokaiselle oliotyypille on määritelty sen jostain attribuuttien osajoukosta muodostuva avain. Tämä mahdollistaa olioiden sekä arvo- että perinteisessä mielessä olio-orientoituneen käsittelyn. Arvo-orientoituneen käsittelyn mukaisesti suora viittaamisen olioon, jonka oliotunniste ei ole tiedossa, on mahdollista sen avainattribuuttien tunnettujen arvojen perusteella. Toisaalta käytön aikana järjestelmässä jokaiselle oliolle tuotetaan sen välittömästä ilmentymätyypistä ja avainattribuuttien arvosta riippuva yksikäsitteinen oliotunniste, jota voidaan käyttää osana kyselyä viittaamaan olioon olio-orientoituneella tavalla. Menettely mahdollistaa mm. olioarvoisten attribuuttien eli roolien käsittelyn (kts. semanttiset yhteydet) kyselykielessä ilman, että käyttäjän täytyisi eksplisiittisesti antaa kaikkia avainattribuutteja vastaavia liitosehtoja. Menettelyn haittana on puhtaan arvo-orientoituneen lähestymistavan tavoin, että avaimeen kuuluvien attribuuttien arvot eivät saa muuttua. Tämä ei kuitenkaan muodostu kynnyskysymykseksi, sillä luvussa 2 esitetyllä tavalla jokaiselle oliotyypille voidaan tarvittaessa määritellä keinotekoinen avainattribuutti eli surrogaatti. Tämä tietenkin lisää tietokantakaavion suunnittelijan vastuuta.

Olio perustetaan yhteen oliotyyppiin. Se on myös jokaisen sellaisen oliotyypin ilmentymä, joka erikoistamis- eli is-a –hierarkiassa on olioon välittömästi liittyvän oliotyypin yläpuolella ts. sen ylätyyppi. Olio sisältää attribuutteinaan sitä vastaavan oliotyypin attribuutit. Attribuutin arvona voi olla vain *primitiivinen literaaliarvo*. Primitiivisiä literaaliarvoja mallissa ovat merkkijonot, kokonais- ja desimaaliluvut sekä päivämäärät.

Semanttinen yhteystyyppi eroaa oliotyypistä siinä, että yhteystyyppejä ei ole järjestetty oliotyyppien tapaan hierarkkisesti. Primitiivisiä literaaliarvoja saavien attribuuttien lisäksi yhteystyypeillä on olioarvoisia attribuutteja, joita kutsumme rooleiksi. Rooliin osallistuvan olion tulee olla roolin määrittelyn yhteydessä annetun oliotyypin ilmentymä. Vastaavasti kuin olioiden tapauksessa, jokin attribuuttien ja roolien osajoukko voidaan määritellä yhteystyypin yhteydet identifioivaksi avaimeksi.

Semanttisiin yhteystyyppeihin liittyy myös luonnollisella kielellä esitetty semanttinen tulkinta, joka voi sisältää viittauksia yhteystyypin attribuuttien arvoihin ja rooleihin osallistuviin olioihin. Näin tyyppiä edustavan semanttisen yhteyden tulkinnassa yhdistyvät sekä kaaviotason että yksittäiseen yhteyteen liittyvä ilmentymätason tieto. Semanttisten yhteyksien selvittämisen ja tulkittavuuden kannalta tarkoituksenmukaista on, että semanttinen tulkinta sisältää viittauksen jokaiseen yhteystyypin sisältämään rooliin ja avainattribuuttiin.

Yksittäinen *semanttinen yhteys* sisältää kaikki sen yhteystyypin attribuutit ja roolit niihin liittyvine arvoineen. Järjestelmässä semanttisia yhteyksiä käsitellään täysin relationaalisesti; niille ei tuoteta olioiden tavoin oliotunnisteita, vaan niiden identifioimiseen käytetään yksinomaan niiden attribuuttien ja roolien arvoja.

Vapaamuotoiset dokumentit sisältävät viittauksia järjestelmässä määriteltyihin olioihin. Niillä voidaan esittää olioihin liittyvää, luonteeltaan tekstuaalista informaatiota, jota ei ole organisoitu tietokannalle tyypilliseen tapaan. Tutkielmassa vapaamuotoisten dokumenttien oletetaan noudattavan XML-syntaksia, joten ne soveltuvat myös rakenteellisten tekstisisältöjen esittämiseen. Viittaus olioon esitetään dokumentissa kohdan 5.2. mukaisella elementillä. Elementti voi sijaita elementtihierarkiassa mielivaltaisella syvyyden tasolla, ja samassa dokumentissa voi olla useita viittauksen sisältäviä elementtejä. Sama dokumentti voi siis sisältää viittauksen myös useampaan tietokannan olioon.

5.2. Tiedon fyysinen organisointitapa

Tieto organisoidaan fyysisesti XML-dokumentteina. Seuraavassa esitetään edellä esiteltyjä tiedon loogisia rakenneosia ja niiden välisiä suhteita vastaava XML-esitystapa esimerkkien avulla.

OLIOTYYPIT

Jokaisen järjestelmän sisältämän oliotyypin määrittely esitetään omassa XML-dokumentissaan. Määrittely sisältää oliotyypin nimen, attribuutit sekä välittömät ylätyypit. XML-esimerkissä 1 annetaan dokumentin person.xml sisältö. Dokumentissa määritellään oliotyyppi `person`.

```
<type>
   <name>person</name>
   <attributes>
      <attribute>forename</attribute>
      <attribute>surname</attribute>
      <attribute is_key="true">ssn</attribute>
      <attribute>sex</attribute>
      <attribute>address</attribute>
      <attribute>phone_number</attribute>
      <attribute>licence</attribute>
   </attributes>
   <supertypes>
      <supertype>legal_subject</supertype>
   </supertypes>
</type>
```

XML-esimerkki 1. Oliotyypin määrittely (oliotyyppi `person` dokumentissa person.xml).

Oliotyypin määrittely esitetään kokonaisuudessaan `type`-juurielementin sisältönä. Juurielementin tulee sisältää yksi `name`-elementti, jossa ilmaistaan määriteltävän oliotyypin nimi. Oliotyypin attribuutit esitetään juurielementin valinnaisen attributes-alielementin attribute-elementtien sisältönä. Myös oliotyypin perityt attribuutit esitetään eksplisiittisesti. Yksittäiseen `attribute`-elementtiin voi liittyä XML-attribuutti `is_key`, jolla on arvo ”`true`” tai ”`false`”. Mikäli XML-attribuutin arvo on ”`true`”, luetaan attribuutti osaksi avain-attribuuttien joukkoa. Mikäli XML-attribuuttia ei ole määritelty tai attribuutin arvona on ”`false`”, ei attribuuttia lueta osaksi avainta. Juurielementti voi sisältää myös valinnaisen elementin `supertypes`, jonka `supertype`-nimisten alielementtien sisältönä annetaan oliotyypin välittömien ylätyyppien nimet. Jos ainuttakaan ylätyyppiä ei ole määritelty, tulkitaan sellaiseksi oliotyyppi `object`.

OLIOT

Samassa dokumentissa voidaan määritellä yksi tai useampi olio. Määrittelyt ryhmitellään oliotyypin mukaisesti alielementteihin. Jokaisen olion osalta annetaan kaikki sen attribuuttien arvot. XML-esimerkki 2 on osa esimerkkitietokannan sisältämästä objects.xml –dokumentista, jossa määritellään kaikki tietokannan sisältämät oliot.

```
<objects>
   <type name="person">
      <object>
         <forename>Amber</forename>
         <surname>Copeland</surname>
         <ssn>120828-235A</ssn>
         <sex>female</sex>
         <address>East Abbey</address>
         <phone_number>331445</phone_number>
         <licence/>
      </object>
      <object>
         <forename>James</forename>
         <surname>Copeland</surname>
         <ssn>010833-020T</ssn>
         <sex>male</sex>
         <address>East Abbey</address>
         <phone_number>331445</phone_number>
         <licence/>
      </object>
      …
   </type>
   …
</objects>
```

XML-esimerkki 2. Olioiden määrittely (`person`-tyypin olioita dokumentissa objects.xml).

Olioiden määrittely esitetään kokonaisuudessaan juurielementin `object` sisällä. Samaan oliotyyppiin kuuluvat oliot ryhmitellään `type`-elementillä, jonka XML-attribuutti `name` ilmaisee oliotyypin nimen. Yksittäinen olio määritellään `type`-elementin `object`-alielementissä. Se sisältää olion attribuuttien mukaan nimetyt alielementit, joiden sisältönä annetaan vastaavien attribuuttien arvot. Merkkijonot sekä kokonais- ja desimaaliluvut esitetään normaaliin tapaan rakenteettomana tekstinä. Päivämäärien esityksessä on käytettävä ISO-standardin [ISO, 2000] mukaista esitysmuotoa ”`YYYY-MM-DD`”, missä `YYYY` vastaa nelinumeroista vuotta, `MM` kaksinumeroisena annettua kuukautta ja `DD` kaksinumeroisena esitettyä päivämäärää. Jos attribuuttia vastaava elementti on tyhjä, tulkitaan attribuutin arvo määrittelemättömäksi (`null`).

Oliotyyppien tavoin jokainen semanttinen yhteystyyppi määritellään omassa dokumentissaan. Semanttisen yhteystyypin osalta määrittelyssä esitetään sen nimi, attribuutit, roolit ja yhteystyypin vaihtoehtoiset semanttiset tulkinnat. XML-esimerkissä 3 annetaan esimerkkitietokantaan sisältyvän parenthood.xml-dokumentin sisältö. Dokumentissa määritellään semanttinen yhteystyyppi `parenthood`.

```
<sem_connection_type>
   <name>parenthood</name>
   <roles>
      <role is_key="true">
         <rolename>parent</rolename>
         <type>person</type>
      </role>
      <role is_key="true">
         <rolename>child</rolename>
         <type>person</type>
      </role>
   </roles>
   <attributes/>
   <sem_interpretations>
      <sem_interpretation>
         <parent><person sex="male"/></parent>
         is the father of
         <child/>
      </sem_interpretation>
      <sem_interpretation>
         <parent><person sex="female"/></parent>
         is the mother of
         <child/>
      </sem_interpretation>
   </sem_interpretations>
</sem_connection_type>
```

XML-esimerkki 3. Semanttisen yhteystyypin määrittely (semanttinen yhteystyyppi `parenthood` dokumentissa parenthood.xml).

Semanttisen yhteystyypin määrittely esitetään kokonaisuudessaan juurielementin `sem_connection_type` sisältönä. Juurielementin tulee sisältää yksi `name`-elementti, joka ilmaisee määriteltävän yhteystyypin nimen. Yhteystyypin roolit määritellään juurielementin `roles`-alielementissä ja vastaavasti yhteys-

tyypin attribuutit sen `attributes`-alielementissä. Attribuuttien määrittely tapahtuu kuten oliotyyppien yhteydessä. Jos yhteystyypillä ei ole attribuutteja, käytetään tyhjää attributes-elementtiä. Yhteystyypin roolit määritellään `roles`-elementin `role`-alielementeillä, joissa annetaan sekä roolin nimi että sitä vastaava oliotyyppi. Edellinen ilmaistaan `role`-elementin `rolename`- ja jälkimmäinen `type`-alielementillä. XML-attribuutti `is_key` voi esiintyä sekä rooleissa että attribuuteissa, ja sen merkitys on sama kuin oliotyypin yhteydessä.

Juurielementin `sem_connection_type` tulee sisältää `sem_interpretations`-elementti, jota käytetään ilmaisemaan yhteyden vaihtoehtoiset semanttiset tulkinnat. Semanttisia tulkintoja voidaan lähestymistavassamme määritellä useita eri oliotyypeille. Niinpä yllä olevalle `parenthood`-yhteydelle annetaan erilaiset tulkinnat riippuen siitä, onko vanhempi mies vai nainen. Edellisessä tapauksessa se voidaan tulkita semanttisesti isyys- ja jälkimmäisessä äitiyssuhteeksi. Erilaiset semanttiset tulkinnat esitetään `sem_interpretations`-elemetin `sem_interpretation`-alielementeissä. Alielementtien sisältö organisoidaan rakenteettoman tekstin tapaan, kuitenkin niin, että se voi sisältää viittauksia yhteystyyppiin liittyvien roolien ja attribuuttien arvoihin. Roolin tai attribuutin arvoon viitataan yksinkertaisesti sen nimeä vastaavalla elementillä. Rooliin viittaava elementti voi lisäksi sisältää yhden alielementin, jonka nimi on jonkin järjestelmässä määritellyn oliotyypin nimi. Tällöin kyseistä semanttista tulkintaa sovelletaan vain tämän oliotyypin ilmentymille. Mikäli oliotyyppiä ei ole annettu, semanttisen tulkinnan oletetaan kohdistuvan kaikkiin roolissa sallittuihin oliotyyppeihin. Oliotyypin ilmaiseva elementti voi sisältää lisäksi oliotyypin attribuuttien mukaan nimettyjä XML-attribuutteja, joiden arvoja käytetään asettamaan attribuuttien arvoihin perustuvia valintaehtoja. Näin semanttisen tulkinta voi riippua paitsi siihen osallistuvien olioiden intensionaalisesta rakenteesta (tyypistä) myös olioiden tilasta ts. niiden attribuuttien arvoista.

VÄLITTÖMÄT SEMANTTISET YHTEYDET

Olioiden tapaan samassa dokumentissa voidaan määritellä useita välittömiä semanttisia yhteyksiä. Yhteydet ryhmitellään dokumentissa niiden tyyppien mukaisesti. Jokaisen semanttisen yhteyden osalta esitetään sen attribuuttien arvot ja rooleihin osallistuvat oliot. XML-esimerkissä 4 annetaan dokumentin sem_connections.xml alkuosa. Siinä määritellään tietokannan sisältämät välittömät semanttiset yhteydet.

```
<sem_connections>
   <sem_connection_type name="parenthood">
      <sem_connection>
         <parent><person ssn="120828-235A"/></parent>
         <child><person ssn="151251-215Q"/></child>
      </sem_connection>
      <sem_connection>
         <parent><person ssn="010833-020T"/></parent>
         <child><person ssn="050555-313B"/></child>
      </sem_connection>
      …
   </sem_connection_type>
   …
</sem_connections>
```

XML-esimerkki 4. Välittömien semanttisten yhteyksien määrittely (`parenthood`-yhteystyyppiin kuuluvia välittömiä semanttisia yhteyksiä dokumentissa sem_connections.xml).

Järjestelmän sisältämät välittömät semanttiset yhteydet esitetään kokonaisuudessaan `sem_connections`-juurielementin sisällä, jossa eri yhteystyyppeihin kuuluvat semanttiset yhteydet ryhmitellään `sem_connection_type`-alielementtien avulla. Alielementin `sem_connection_type` XML-attribuutti `name` ilmaisee määriteltävien välittömien semanttisten yhteyksien tyypin. Yksittäinen tyyppiä edustava välitön semanttinen yhteys määritellään `sem_connection`-elementin sisältönä. Yhteyden attribuuttien arvot esitetään kuten olioiden tapauksessa. Rooleihin osallistuvat oliot määritellään roolin mukaan nimetyn elementin sisältönä alielementissä, jonka nimi vastaa olion tyypin nimeä. Se sisältää olion avainattribuuttien mukaiset XML-attribuutit, joiden arvot yksilöivät olion.

VAPAAMUOTOISET DOKUMENTIT

Vapaamuotoisten dokumenttien syntaksia ei ole rajoitettu seuraavia poikkeuksia lukuun ottamatta.

- Niiden tulee noudattaa XML-syntaksia.
- Niiden juurielementin nimen voidaan olettaa kuvaavan dokumentin luonnetta. Tutkielmassa oletetaan, että nämä nimet ovat riittävällä tasolla standardoituja.
- Dokumenteissa esiintyvät viittaukset olioihin tulee antaa alla esitettävien periaatteiden mukaisesti.

- Jäljempänä selvitettävällä tavalla viittauksessa käytettävä nimiavaruus (`ref`) on varattu yksinomaan olioihin tehtäviä viittauksia varten.

Vapaamuotoisessa XML-dokumentissa oleva viittaus olioon voidaan toteuttaa ainakin kahdella toisistaan eroavalla ratkaisuperiaatteella. Ensinnäkin olion identifioimiseen voidaan käyttää viittauksessa sen avainattribuuttien arvoja ja oliotyypin nimeä. Tällaisen arvo-orientoituneen ratkaisutavan asemesta voidaan hyödyntää XML-esitystavan rakenteellisuutta ja viitata olioon siihen liittyvän XML-polun avulla. Tämä tarkoittaa linkin luomista vapaamuotoisesta dokumentista kohdeolion määrittelyn sisältävään osadokumenttiin. Jälkimmäisen menettelytavan heikkoutena on viittauksen sitominen XML-dokumentin tiettynä ajanhetkenä vallitsevaan organisointitapaan. Dokumenttien rakenteiden muuttuessa myös näin muodostetut linkit pitäisi muodostaa uudelleen. Jos viittauksessa käytettäisiin esimerkiksi XPathin [XPath, 1999] sallimia indeksoituja elementtien nimiä, olioita vastaavien elementtien keskinäisen järjestyksen muuttaminen tai jonkin olion poistaminen aiheuttaisi linkkirakenteen sekaantumisen.

Vaihtoehtoisesti voitaisiin käyttää XPathin tekstihakuun soveltuvia funktioita, joiden avulla olioon voidaan viitata sen muuttumattomien attribuuttien arvoihin perustuen. Tällöin ratkaisusta tulisi kuitenkin arvo-orientoitunut, mikä vastaa ensin kuvattua ratkaisumallia. Olioiden määrittelyt sisältäviin elementteihin voitaisiin myös liittää yksilöllinen XML-attribuutti (esimerkiksi `id`), jonka arvo vastaa olion yksilöllistä oliotunnistetta. Koska tarkasteltavassa järjestelmässä oliotunnisteena käytetään olion avainattribuuttien arvoja yhdessä oliotyypin nimen kanssa, ei kysymys ole periaatteellinen, vaan liittyy käytettyyn esitystapaan. Esimerkkijärjestelmän toteutuksessa on päädytty XML-esimerkin 5 mukaiseen viittaamistapaan.

Huomionarvoisia XML-esimerkin 5 dokumentissa ovat elementtien `seller`, `buyer` ja `vehicle` sisällä olevat viittaukset tietokannan olioihin. Viittaamiseen käytetään elementtiä, joka sisältää XML-attribuutteinaan olion avaimen. Elementin nimi on muotoa ”`ref:typename`”, missä `typename` on olion tyypin nimi. Elementin nimeämisessä sovelletaan nimiavaruus- (engl. namespace) määrittelyä. Se ei sisälly itse XML-spesifikaatioon, vaan se on määritelty XML-konsortion erillisessä Namespaces in XML –suosituksessa [XML Namespaces, 1999]. XML-spesifikaation näkökulmasta ilmaus ymmärretään kokonaisuudessaan elementin nimeksi. Sen sijaan Namespaces–suosituksen perusteella ilmauksen ”`:`”-merkkiä edeltävää (tässä tapauksessa `ref`-), osaa pidetään nimiavaruutena, johon ”`:`”-merkkiä seuraava elementin nimi sisältyy. Nimiavaruusmäärittelyn ansiosta samassa dokumentissa voidaan käyttää useita samannimisiä elementtejä, jotka kuitenkin sisältyvät eri nimiavaruuksiin ja joilla siten on yleensä erilainen semanttinen merkitys. Tutkielman esimerkkijärjestelmän tapauksessa tätä voidaan käyttää hyväksi, jolloin vältetään kaikkien tietokannas-

sa esiintyvien oliotyyppien nimien muodostuminen varatuiksi sanoiksi. Tämän sijasta voimme yksinkertaisesti määritellä nimiavaruuden `ref` varatuksi kyseiseen tarkoitukseen.

```
<bill_of_sale>
   <date>2001-04-30</date>
   <seller>
      <ref:enterprise name="Cassius Automotives">
         Cassius Automotives
      </ref:enterprise>
   </seller>
   <buyer>
      <ref:person ssn="130845-772E">Allan Selsor</ref:person>
   </buyer>
   <vehicle>
      <ref:car register="USD-999">USD-999</ref:car>
   </vehicle>
   <ordometer_reading>0</ordometer_reading>
   <selling_price>65000 euros</selling_price>
   <warranties/>
</bill_of_sale>
```

XML-esimerkki 5. Vapaamuotoisen dokumentin bos.xml sisältö.

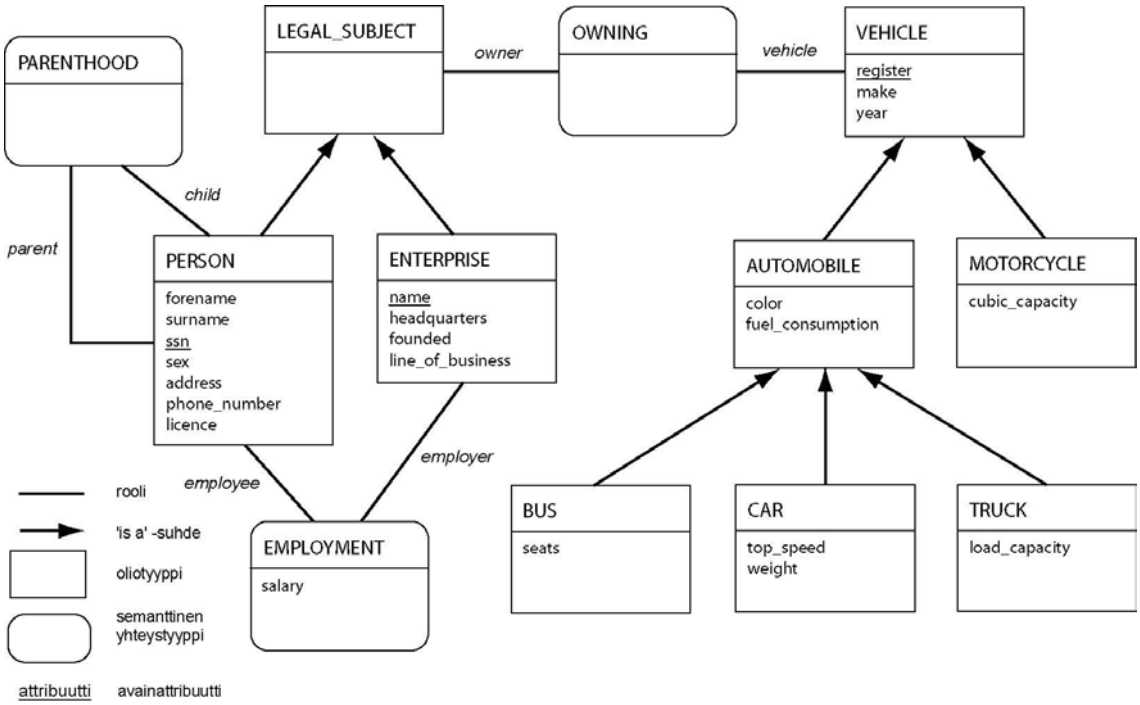
Viittauselementtiä voidaan käyttää HTML:n linkkien muodostamiseen tarkoitetun `A`-elementin tapaan. Toisin sanoen sen sisältönä annetaan teksti, jonka halutaan näkyvän selattaessa dokumenttia jollain sovellusohjelmalla.

6. Esimerkkitietokanta ja vapaamuotoiset dokumentit

Tässä luvussa esitellään kyselykielen demonstrointia varten luotu tietokanta sekä siihen liittyvät vapaamuotoiset dokumentit.

Esimerkkitietokanta sisältää oliotyypin oikeussubjekti (`legal_subject`) alatyyppeinään henkilö (`person`) ja yritys (`enterprise`) sekä oliotyypin ajoneuvo (`vehicle`) alatyyppeinään moottoripyörä (`motorcycle`) ja auto (`automobile`). Oliotyypillä auto on edelleen alatyypit henkilöauto (`car`), rekka (`truck`) ja linja-auto (`bus`). Tietokannan semanttiset yhteystyypit ilmaisevat henkilön ja yrityksen välisen työskentelysuhteen (`employment`), oikeussubjektin ja ajoneuvon välisen omistussuhteen (`owning`) sekä henkilötyypin sisällä vallitsevan vanhemmuussuhteen (`parenthood`). Olio- ja yhteystyyppeihin liittyvät attribuutit ja roolit sekä tyyppien tarkat keskinäiset suhteet esitetään kuvassa 1. Kuvassa kunkin oliotyypin kohdalla esitetään vain oliotyypin omat att-

ribuutit, ei sen perimiä attribuutteja. Roolit kuvataan yhdyssuorina, jotka yhdistävät toisiinsa yhteystyyppejä ja niihin osallistuvia oliotyyppejä.



Kuva 1. Esimerkkitietokannan kaaviotason kuvaus.

Kuvan 1 mukaista kaaviotasoa vastaa tietokannan ilmentymätaso yksittäisine olioineen ja semanttisine yhteyksineen. Tietokannassa esiintyvät oliot ja niihin liittyvät attribuuttien arvot on esitetty `enterprise` -tyyppiin liittyen taulukossa 2, `person` -tyyppiin liittyen taulukossa 3 sekä `vehicle`-tyyppiin liittyen taulukossa 4.

ENTERPRISE

name	headquarters	founded	line_of_business
Advanced Experts Unlimited	Scottsfield	1993	consulting
Cafe Round Table	Maymond	1880	cafeteria
Cassius Automotives	North End	2001	automotive sales
Darwin's Bakery	White Forest	1988	bakery
Everyway Transportation	Greenbridge	1952	transportation
Lifetime Insurance Company	Down Crossing	1933	insurance
Prominent Insurance Company	Finton	1962	insurance
South Bay Industries	Fourway	1977	steel industry
Tellurian Insurance Company	Kernington	1970	insurance
Westfield Transit	Westfield	1969	bus operating

Taulukko 2. Esimerkkitietokannan `enterprise`-tyypin oliot.

PERSON						
ssn	**surname**	**forename**	**sex**	**address**	**phone_number**	**licence**
010833-020T	Copeland	James	male	East Abbey	331445	no
021050-778D	Stryker	David	male	Wilpoint	290877	yes
030750-761E	Goodheart	Erica	female	Finton	108995	yes
040185-100Y	Selsor	Thomas	male	Tunmarch	258008	yes
040560-212F	Cassius	Lisa	female	Greenbridge	543116	yes
050495-620U	McCurley	Hugh	male	Kingmont	912355	yes
050555-313B	Freeman	Ethel	female	Tunmond	243995	yes
060659-219K	Castaldo	Ferdinand	male	Cambury	916654	yes
060685-517X	Castaldo	Maricela	female	Cambury	211569	no
070659-218L	Laakso	Darren	male	Evanroad	342828	yes
120828-235A	Copeland	Amber	female	East Abbey	331445	no
121272-459W	McCurley	Ann	female	Old Garden	512343	yes
130347-372S	Baker	Alvin	male	Finton	818555	yes
130845-772E	Selsor	Allan	male	Scottsfield	418955	yes
150372-152V	Stryker	Calvin	male	Greenbridge	193776	yes
150735-542Y	Cassius	Kathrine	female	North End	163222	no
150762-115S	Campbell	Paula	female	Oakdale	572404	yes
151251-215Q	Copeland	Dale	male	Finton	108995	yes
152260-140G	McCurley	Codey	male	Old Garden	512343	yes
170165-100I	Guida	Neil	male	West Yard	716543	yes
171166-333F	Castaldo	Annabelle	female	Cambury	916654	yes
280643-005J	Cabe	Kathy	female	North End	104500	no
311182-021A	Baltes	Mallory	female	Fairshire	683576	yes

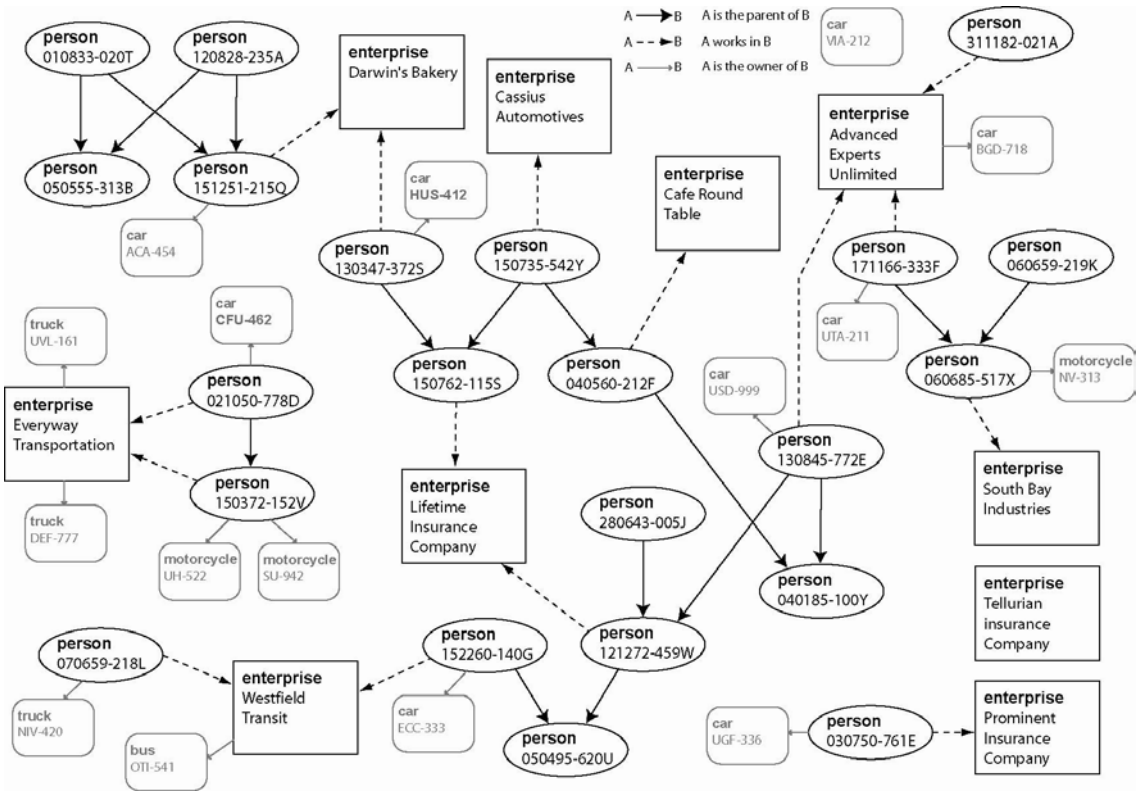
Taulukko 3. Esimerkkitietokannan `person`-tyypin oliot.

VEHICLE	**register**	**make**	**year**				
MOTORCYCLE				**cubic_capacity**			
	NV-313	honda	1996	650			
	SU-942	yamaha	1996	900			
	UH-523	kawasagi	1980	500			
AUTOMOBILE				**color**	**fuel_consumption**		
CAR						**top_speed**	**weight**
	ACA-454	skoda	1988	red	20	120	1100
	BGD-718	mercedes	1990	blue	15	160	1400
	CFU-462	fiat	1995	grey	17	170	1100
	ECC-333	audi	1994	white	17	200	1200
	HUS-412	saab	1990	green	14	150	1200
	UGF-336	opel	1998	red	12	200	1300
	USD-999	chrysler	2001	silver	12	220	1400
	UTA-211	fiat	2000	yellow	8	140	1100
	VIA-212	lada	1982	white	22	110	1000
TRUCK						**load_capacity**	
	DEF-777	hummer	1992	brown	32	25000	
	NIV-420	luna	1997	red	28	15000	
	UVL-161	sisu	1985	grey	30	20000	
BUS						**seats**	
	OTI-541	scania	1998	blue	22	50	

Taulukko 4. Esimerkkitietokannan `vehicle`-tyypin oliot.

Tietokannan semanttiset yhteydet on esitetty kuvassa 2. Kuvassa eri tyylejä edustavat suorakulmiot ja soikiot vastaavat eri oliotyyppejä edustavia yksittäisiä olioita, joiden avainattribuuttien arvot on annettu. Välittömiä semanttisia yhteyksiä vastaavat kuvassa olioiden väliset nuolet. Kuvan avulla lukijan on mahdollista hahmottaa sekä tietokannan välittömät semanttiset yhteydet että

niistä muodostuvat välilliset yhteydet. Esimerkkitietokanta on suunniteltu nimenomaisesti kyselykielen demonstrointia varten, joten siinä suurin osa olioista liittyy semanttisesti toisiinsa. Olioiden välillä ei esimerkkitietokannassa myöskään esiinny suurta määrää vaihtoehtoisia semanttisia yhteyksiä. Reaalimaailman tilanteissa voi tietokannan yhteystiheys olla sovellusalueesta riippuen huomattavastikin erilainen, jolloin toisiinsa semanttisesti yhdistettyjen olioiden keskenään yhdistämättömien osajoukkojen tai vaihtoehtoisten semanttisten yhteyksien määrä on suurempi.



Kuva 2. Esimerkkitietokannassa olevat semanttiset yhteydet.

Esimerkkitietokantaan liittyvät vapaamuotoiset dokumentit on esitetty kokonaisuudessaan liitteessä 2. Dokumenttien juurielementit on nimetty niiden tyypin mukaisesti. Esimerkkijärjestelmässä dokumenttityyppejä ovat ajoneuvojen kauppakirjat (juurielementti `bill_of_sale`), vahinkoilmoitukset (juurielementti `damage_report`) sekä henki- (juurielementti `life_insurance`) ja ajoneuvovakuutussopimukset (juurielementti `motor_vehicle_insurance`).

7. Kyselykielen syntaksi ja semantiikka

Tässä luvussa käsitellään ensin tutkielman kyselykielen suunnittelun lähtökohtia, minkä jälkeen esitellään kieleen sisältyvät muuttujan ja arvon käsitteet sekä kielen täsmällinen syntaksi primitiiveineen. Syntaksin ohella annetaan primitiiveihin liittyvä semantiikka.

7.1. Suunnittelun lähtökohdat

Tutkielman kyselykielen suunnittelun ensisijaisena tavoitteena on ollut mahdollistaa semanttisten yhteyksien selvittäminen, siten että loppukäyttäjä voi ilmaista kyselynsä intuitiivisella ja deklaratiivisella tavalla. Keskeinen tavoite on ollut myös kehittää primitiivejä, joilla käyttäjä voi määritellä voimakkaita suodatusehtoja kyselyn tulokseen sisältyville yhteyksille. Semanttisten yhteyksien etsimisen ohella kyselykielen odotetaan tukevan tietokannan olioihin liittyvien vapaamuotoisten dokumenttien käsittelyä.

Tavoitteiden saavuttamiseksi tutkielman kyselykieli sisältää kolme toisiaan täydentävää primitiivien osajoukkoa. Kyselykielen ydinilmaisuvoiman muodostavat primitiivit mahdollistavat perinteiset ekstensionaaliset kyselyt edellä annettujen periaatteiden mukaan organisoidussa tietokannassa, joka koostuu oliotyypeistä, olioista sekä semanttisista yhteystyypeistä ja välittömistä semanttisista yhteyksistä. Tämä primitiivien osajoukko muistuttaa läheisesti monia olemassa olevia kyselykieliä: SQL:ää johdannaisineen sekä erityisesti rajoitettua osaa Niemen et al. [2002] artikkelissa esitetystä DOOD-tietokantaparadigmaan perustuvasta kyselykielestä. Ydinilmaisuvoimaa on laajennettu primitiiveillä, jotka mahdollistavat välittömien ja välillisten semanttisten yhteyksien löytämisen. Kolmas olennainen osa kielen ilmaisuvoimaa muodostuu dokumenttien käsittelyyn tarkoitetuista primitiiveistä.

Koska kieli on tarkoitettu yksinomaan kyselykieleksi, ei sen sellaisenaan edellytetä olevan laskennallisesti täydellinen. Vertailukohtana voidaan käyttää RDOOM-tietomallia varten kehitettyä kyselykieltä [Niemi et al., 2002b, 2004], joka perustuu yksinkertaisen, ilmaisuvoimaltaan rajoitetun kysely-ympäristön tarjoamiseen loppukäyttäjälle kätkemällä tietokantaan sisältyvät ilmaisuvoimaiset deduktiiviset määrittelyt. Tämän periaatteen mukaisesti myös tutkielman kyselykieli sisältää valmiita semanttisten yhteyksien selvittämiseen tarkoitettuja primitiivejä, joiden määrittelyt edellyttävät laajaa ilmaisuvoimaa. Toisaalta kieli ei sellaisenaan tue esimerkiksi predikaattien tai aliohjelmien määrittelyä, mikä olisi tarpeen, jos sitä käytettäisiin varsinaisen ohjelmointikielen tapaan.

Tutkielman kyselykieli on tavoitteenasettelun mukaisesti luonteeltaan deklaratiivinen. Proseduraalisia kontrollirakenteita ei tarvita, vaan muuttuja arvotetaan automaattisesti jokaiseen tietokannassa ja dokumenttikokoelmassa ole-

vaan validiin arvoon, joka täyttää annetut ehdot. Jokainen tällainen arvotus sisältyy kyselyn vastaukseen. Mikäli kyselyn tulos sisältää useita muuttujia, muodostuu vastaus niiden arvotusten kaikista keskenään johdonmukaisista kombinaatioista. Koska muuttujien arvotukset ovat funktionaalisten ja deklaratiivisten kielten tapaan sidottuja niiden käyttökontekstissa, ei kyselyn sisältämien ehtojen järjestyksellä ole vastauksen kannalta merkitystä.

7.2. Käytetty notaatio ja termistö

Kielen syntaksiin liittyviä yksityiskohtia esitellään tässä luvussa sekä sanallisesti että BNF-notaatioon (Backus-Naur Form) perustuvien kielioppisääntöjen avulla. Kyselykielen kielioppi on annettu kokonaisuudessaan liitteessä 1. Kieliopin esitystapa perustuu notaatiosta annettuun Extended BNF –standardiin [ISO/IEC 14977, 1996]. Esityksessä nonterminaalia `A` vastaava kieliopillinen sääntö, joka voidaan muodostaa nonterminaali- ja/tai terminaalisymboleiden muodostamasta merkkijonosta α, ilmaistaan muodossa "`A ::= α;`". Terminaalisymbolit erotetaan nonterminaalisymboleista kirjoittamalla ne lainausmerkkien ("") sisään. Vaihtoehtoiset merkkijonot erotetaan toisistaan pystyviivalla (`|`). Valinnaiset merkkijonot, joiden esiintymismäärä on mielivaltainen, esitetään aaltosulkeiden välissä (`{}`). Vastaavasti merkkijonot, jotka esiintyvät joko kerran tai eivät lainkaan, esitetään hakasulkeiden välissä (`[]`).

Kielioppisäännöissä esitettävät tyhjät (välilyönti) merkkijonot ja osamerkkijonot on tarkoitettu ensisijaisesti kielen esitysasun selkiyttämiseen. Niiden osalta kielioppi on luonteeltaan ohjeellinen tyylisääntö, sillä kielen toteutus sallii lisäksi ylimääräisten tyhjämerkkien lisäämisen primitiivien ja niiden argumenttien välille sekä toisaalta tyhjämerkkien poisjättämisen esimerkiksi pilkulla (`,`) toisistaan erotettujen primitiivien välistä. Tarkoituksena on tältä osin ollut selkiyttää kieliopillista esitystä pikemminkin kuin esittää kaikkia kielessä hyväksyttäviä erilaisia kombinaatioita.

Kyselykieli perii eräitä deduktiivisille tietokantakielille, erityisesti Prologille tyypillisiä syntaktisia piirteitä. Seuraavassa esittelemme nämä piirteet sekä niihin liittyvän termistön.

Muuttujaa (variable) merkitään kielessä merkkijonolla, joka alkaa isolla kirjaimella tai alaviivalla (_) ja sisältää kirjain- tai numeromerkkejä tai alaviivoja. Esimerkkejä kielessä hyväksyttävistä muuttujanimistä ovat "`A`", "`_2`", "`Koe`", ja "`B2_`". Muuttujanimi "_" on varattu anonyymimuuttujalle, jonka käyttöön liittyvät säännöt on esitetty kohdassa 7.4. *Merkkijonovakio* (constant string) on pienellä kirjaimella alkava merkkijono, joka voi muuttujan tavoin sisältää kirjain- tai numeromerkkejä sekä alaviivoja. Esimerkkejä merkkijonovakioista ovat "`car`","`a1`" ja "`abc_2`". *Merkkijonolla* (string) viitataan jatkossa heittomerkeillä (`''`) erotettuun merkkijonoon, joka voi sisältää kirjain- tai numeromerkkien ohella erikoismerkkejä heittomerkkejä lukuun ottamatta. *Kokonais-* (integer) ja

desimaalilukujen (decimal number) merkitsemiseen käytetään tavanomaista esitystä, missä desimaalierottimena käytetään pistettä (.). Luku voidaan varustaa positiivisella (+) tai negatiivisella (-) etumerkillä, joka liitetään välittömästi lukuvartalon eteen. Esimerkkejä kyselykielen hyväksymistä kokonais- ja desimaaliluvuista ovat "+1", "1", ja "-3.5".

7.3. Arvot ja arvoviittaukset

Kyselykielessä esiintyvät arvot voidaan jakaa primitiivisiin literaaliarvoihin ja ei-primitiivisiin arvoihin. *Primitiiviset literaaliarvot* (primitive literal) esiintyvät olioiden ja välittömien semanttisten yhteyksien attribuuttien arvoina. Niitä ovat edellä esitetyt merkkijono, kokonaisluku, desimaaliluku sekä niistä erikseen johdettu päivämäärä (date). Päivämäärän BNF-määrittely on muotoa

```
date ::= integer "-" integer "-" integer;.
```

Tämä mukailee päivämäärän esitysmuodon ISO-standardia [ISO, 2000], missä ensimmäinen kokonaisluku tarkoittaa vuotta, toinen kuukautta ja kolmas päivää. Vaikka päivämäärä ei siis kirjaimellisesti ottaen ole luonteeltaan primitiivinen arvo, sitä käytetään syntaktisesti samoissa yhteyksissä kuin varsinaisia primitiivisiä arvoja, ja sen katsotaan siksi kieliopillisesti kuuluvan niihin[1].

Kielen *ei-primitiivisiä arvoja* ovat olio, semanttinen yhteys ja dokumentti. Ei-primitiivisille arvoille on kielessä ominaista, että niihin voidaan viitata muuttujalla. Myös ei-primitiivisille arvoille on kielessä literaaliesitys, jota käytetään jäljempänä mainittavaa poikkeusta lukuun ottamatta sekä osana muodostettavaa kyselyä että kyselyn vastauksessa.

Olioviittaus (object reference) on viittaus olioon. Seuraavan määritelmän mukaisesti olioviittaus voi olla joko muuttuja tai olion literaalinen esitys (literal object reference).

```
object_reference ::= variable | literal_object_reference;
```

Olion literaaliesitys kielessä on

```
literal_object_reference ::= typename " " key_part;.
```

Tässä `typename` on olion tyypin nimi. Oliotunnisteen avainosa `key_part` muodostetaan olion avainattribuuttien arvoista. Mikäli oliolla on vain yksi avainattribuutti, käytetään kyseisen attribuutin arvoa. Mikäli avainattribuutteja on useita, käytetään näiden pilkulla (,) toisistaan erotettuja arvoja, jotka on esitetty hakasulkeiden ([]) sisällä. Avainosan tarkka syntaksi on seuraava.

```
key_part ::= literal |
"[" literal {"," literal} "]";
```

Avainosassa esiintyvien avainattribuuttien arvojen järjestys on sama kuin järjestys, jossa attribuutit on esitetty oliotyypin määrittelevässä XML-dokumentissa. Tämä merkitsee, että loppukäyttäjän ei voida olettaa ennalta

[1] Myös desimaaliluku on johdettu kieliopissa kokonaisluvun määritelmän avulla.

tuntevan tapaa, jolla oliotunniste on generoitu. Oliotunnisteen asemesta voidaan aina käyttää muuttujaa, johon on liitetty avainattribuuttien arvoja vastaavat valintaehdot. Olioiden literaalinen esitystapa onkin tarkoitettu käytettäväksi ensisijaisesti kyselyn vastauksessa.

Viittaus semanttiseen yhteyteen (semantic connection reference) voidaan osana kyselyä ilmaista muuttujalla seuraavasti.

```
semantic_connection_reference ::= variable;
```

Myös semanttisilla yhteyksillä on literaaliesitys. Toisin kuin olioiden tapauksessa, esitystapaa ei kuitenkaan voida käyttää osana muodostettavaa kyselyä, vaan sitä sovelletaan yksinomaan kyselyn vastauksessa. Tämän vuoksi sitä ei ole annettu osana kyselykielen määrittelyä.

Välittömän semanttisen yhteyden literaaliesitys on sen yhteystyypin luonnollisella kielellä esitetty semanttinen tulkinta, johon sisältyvät viittaukset yhteyden attribuuttien arvoihin ja rooleissa esiintyviin olioihin on korvattu niiden literaaliesityksillä. Välillisen semanttisen yhteyden literaaliesitys muodostetaan liittämällä siihen sisältyvien välittömien semanttisten yhteyksien literaaliesitykset toisiinsa `and`-konnektiivilla. Esimerkiksi erään esimerkkitietokannasta johdettavissa olevan välillisen semanttisen yhteyden literaaliesitys on: ”`person '120828-235A' is the mother of person '151251-215Q' and person '010833-020T' is the father of person '151251-215Q'.`”.

Viittaus dokumenttiin (document reference) ilmaistaan kielessä muuttujalla tai dokumentin literaaliesityksellä (literal document reference). Sen BNF-määrittely on seuraava.

```
document_reference ::=
variable | literal_document_reference;
```

Tutkielman kyselykielen kannalta dokumentti on kokonainen XML-dokumentti tai siihen mielivaltaisella syvyyden tasolla sisältyvä elementti eli osadokumentti. Dokumenttien literaaliesityksenä käytetään niihin liittyviä polkuilmauksia.

Kyselykielen polkuilmaukset muistuttavat yksinkertaisia XPath-tyylisiä lausekkeita [XPath, 1999]. Tiedostonimen ja elementtien nimien välisinä erottimina käytetään XPathin tapaan operaattoreita ”`/` ” ja ”`//` ”, joiden merkitys on sama kuin XPathissa. Mikäli elementti sisältää useita samannimisiä alielementtejä, voidaan näistä jokaiseen viitata käyttämällä polussa elementtien yhteistä nimeä. Siis esimerkiksi ilmaus ”`tiedostonimi/juurielementti/alielementti`” arvotetaan vastauksessa vuorollaan jokaiseen juurielementin `alielementti`-nimiseen välittömään alielementtiin. Mikäli haluttaisiin viitata samannimisistä sisarelementeistä järjestyksessä toisena olevaan, voitaisiin tämä XPathin tavoin tehdä ilmauksella ”`tiedostonimi/juurielementti/alielementti[2]` ”.

XPathin ”*”-merkin sijaan kielessä polun mielivaltainen osa voidaan ilmaista muuttujalla. Muotoa ”X/Y” oleva ilmauksen yhtäsuuruusvertailu polun ”tiedostonimi/juurielementti/alielementti” kanssa saa aikaan muuttujan X arvottumisen osapoluksi tiedostonimi ja muuttujan Y arvottumisen osapoluksi ”juurielementti/alielementti”. Muuttujat siis arvotetaan polussa vasemmalta alkaen ilmauksessa viimeisen muuttujan arvottuessa polun koko loppuosaan eli häntään. Ääritapauksessa, mikäli ilmaus koostuu vain yhdestä muuttujasta, arvottuu se koko kohdepolkuun. Mikäli ilmaus sisältää ”//”-operaattoreita, suoritetaan arvotus näiltä osin samaan tapaan kuin ”/”-operaattorin kohdalla. Sen lisäksi otetaan huomioon tapaukset, joissa ”//”-operaattorin osoittamassa kohdassa jätetään huomiotta välistä yksi tai useampi alielementti. Siis ilmauksen ”X//Y” arvottaminen poluksi ”tiedostonimi/juurielementti/alielementti” kanssa johtaa X:n arvottumiseen osapolulla tiedostonimi ja Y:n arvottumiseen osapoluiksi ”juurielementti/alielementti” ja ”alielementti”.

Jokaiselle dokumentille voidaan antaa useita ei-täydellisiä ja täsmälleen yksi täydellinen polku. Kutsumme polkua tässä täydelliseksi, mikäli se ei sisällä operaattoria ”// ”, muuttujia tai ilman järjestyslukuindeksiä annettua viittausta elementtiin, jolla on useita samannimisiä sisarelementtejä. Vastauksessa jokainen XML-elementtiin viittaava muuttuja arvotetaan täydellisellä polulla.

Kyselykielen BNF-kieliopissa esitetään lisäksi käsite *arvoviittaus* (value reference), joka määritellään seuraavasti.

```
value_reference ::=
object_reference | semantic_connection_reference |
document_reference | attribute_reference |
primitive_literal;
```

Määritelmän mukaisella arvoviittauksella tarkoitetaan ilmausta, johon joko suoraan tai suoritettavan kyselyn tuloksena liittyy jokin primitiivinen tai ei-primitiivinen arvo.

Arvoviittauksen määritelmään sisältyy *attribuuttiviittauksen* (attribute reference) käsite. Attribuuttiviittauksen avulla voidaan palauttaa olion tai välittömän semanttisen yhteyden attribuutin arvo tai olio, joka osallistuu semanttiseen yhteyteen määrätyssä roolissa. Lisäksi jokaiseen dokumenttiin liittyy kyselykielessä näennäinen attribuutti content, joka palauttaa sitä vastaavan elementin jäsennetyn esityksen (kts. esimerkkikysely 8). Attribuuttiviittauksen syntaksi on muotoa

```
attribute_reference ::=
(object_reference | semantic_connection_reference) ":"
attribute_name | document_reference ":content";.
```

Tässä attribute_name on jonkin attribuutin tai roolin nimi.

7.4. Kyselyn rakenne ja kielen primitiivit

Kyselykielellä suoritettavan kyselyn yleinen syntaksi on muotoa

```
query ::= result_part [" where " condition_part] ".";.
```

Pakollisen *tulososan* (result part) syntaksi noudattaa seuraavaa BNF-määrittelyä.

```
result_part ::= value_reference {", " value_reference};
```

Tulososa koostuu siis pilkulla toisistaan erotetuista arvoviittauksista. Tämän mukaisesti ehkäpä triviaalein kielen sallima kysely on ”`1.`”, joka antaa vastaukseksi arvon `1`. Käytännössä literaalien sisällyttämiseen tulososaan ei kuitenkaan ole tarvetta. Tavallisesti siinä käytetään kyselyn ehto-osassa esiintyviä muuttujia tai niihin liittyviä attribuuttiviittauksia, jotka arvotetaan annettujen ehtojen perusteella. Mikäli tulososassa kuitenkin esiintyy vapaita muuttujia, joilla ei ole vastinetta kyselyn ehto-osassa, arvotetaan ne tietokannan olioihin. Semanttisten yhteyksien ja elementtihierarkian eri tasoilla olevien dokumenttien potentiaalisen suuren määrän johdosta ei niiden sitominen vapaisiin muuttujiin ole prosessoinnin kannalta tehokas ratkaisu.

Erityisesti anonyymimuuttujan tapauksessa on huomattava, että jokainen anonyymimuuttujan esiintymä tulkitaan kyselyssä itsenäiseksi muuttujaksi. Näin ollen tulososassa ei voida viitata kyselyn ehto-osassa esiintyvän anonyymimuuttujan arvoon. Vastaavasti ehto-osassa mahdollisesti useassa erillisessä ehdossa esiintyvät anonyymimuuttujat tulkitaan eri muuttujiksi.

Tulososa voi myös sisältää implisiittisesti annettuja ehtoja muuttujan arvotuksen intensionaalisesta rakenteesta. Tätä havainnollistaa esimerkkikysely 1. Koska kyselyn tulososassa esiintyy attribuuttiviittaus ”`X:name`”, voi muuttuja `X` saada arvokseen vain olioita tai välittömiä semanttisia yhteyksiä, joilla on `name`-niminen attribuutti tai välittömiä semanttisia yhteyksiä, joilla on `name`-niminen rooli. Koska esimerkkikyselyssä 1 muuttujaan `X` ei ole liitetty eksplisiittisiä ehto-osassa määriteltäviä ehtoja, voidaan muuttuja arvottaa vain olioksi. Esimerkkikyselyssä 1 muuttuja `X` arvottuu oliotyypin `enterprise` ilmentymiksi, sillä esimerkkitietokannassamme vain niillä on vaadittu attribuutti. Kyselyn vastaukseen sisältyvistä arvotuksista on esitetty esimerkissä vain kolme ensimmäistä.

kysely
`X:name.`
vastaus
`X:name = 'Prominent Insurance Company'`
`X:name = 'Lifetime Insurance Company'`
`X:name = 'Tellurian Insurance Company'`
`...`

Esimerkkikysely 1. Kyselyn tulososan sisältämä implisiittinen ehto, joka valitsee tietokannan olioita niiden intensionaalisen rakenteen perusteella.

Tulososan lisäksi kysely sisältää tavallisesti myös *ehto-osan*, jossa määritellään tulososan muuttujien arvottamiselle asetettavat ehdot. Tämä erotetaan tulososasta avainsanalla ”`where`”. Ehto-osan syntaksi on muotoa

```
condition_part ::= condition {", " condition};.
```

Muodollisesti ehto-osa on siihen sisältyvien ehtojen konjunktio. Ehdot erotetaan toisistaan pilkulla (,). Ehto (condition) on jokin seuraavissa alakohdissa esiteltävistä kielen primitiiveistä.

7.4.1. Perusprimitiivit

Ekstensionaaliset kyselyt ovat perinteisiä tietokantakyselyitä. Ekstensionaaliset kyselyt mahdollistavia primitiivejä kyselykielessä ovat tyyppimäärittely, negaatio ja vertailuoperaatiot. Näiden primitiivien avulla voidaan suoraan tai epäsuorasti suorittaa relaatioalgebran kanssa analogiset operaatiot: valinta, liitos, projektio ja karteesinen tulo, kun operandirelaatiot rinnastetaan välittömiin semanttisiin yhteyksiin ja olioihin. Sen sijaan kieli ei sisällä vastinetta relaatioalgebran muille operaatioille tai sisäkkäisille kyselyille. Kielen tarkoituksena onkin tältä osin ollut lähinnä tarjota käyttäjälle yksinkertaiset ekstensionaaliset kyselyt mahdollistava ilmaisumuoto, joka tukee kehittyneempien kyselyominaisuuksien integrointia kyselykieleen.

Tyyppimäärittelyllä muuttujan arvoksi määritellään jonkin olio- tai yhteystyypin ilmentymä. Tyyppimäärittely muistuttaa SQL-kyselyn `WHERE`-osassa annettavia muuttujien esittelyjä. Tyyppimäärittelyn syntaksin BNF-määrittely on seuraava.

```
type_declaration ::= typename " "
(object_reference | semantic_connection_reference);
```

Tässä `typename` on olio- tai yhteystyypin nimi. Tyyppimäärittelyn käyttöä havainnollistetaan esimerkkikyselyssä 2, jossa vastaukseen halutaan kaikki tietokannassa olevat `person`-tyyppiset oliot.

kysely

```
X where person X.
```

vastaus

```
X = person '120828-235A'
X = person '010833-020T'
X = person '050555-313B'
...
```

Esimerkkikysely 2. Tyyppimäärittelyn käyttö osana kyselyä.

Negaatio-operaattorina kielessä on ”`neg`”. Negaation BNF-määritelmä on

```
negation ::= "neg " condition;.
```

Tarkasteltavassa kyselykielessä muuttuja pyritään automaattisesti arvottamaan jokaiseen tietokannan kannalta validiin arvoon myös ehtojen negaatioihin liittyen. Tämän mukaisesti kysely ”`X where neg person X.`” palauttaa tietokannan kaikki oliot, jotka eivät ole luokan `person` ilmentymiä.

Vertailuoperaattoreita ovat ”<”, ”=”, ”>” ja ”><”, joiden merkitys vasemmalta alkaen on: pienempi kuin, yhtä suuri kuin (sama kuin), suurempi kuin ja eri suuri kuin (eri kuin). Vertailu voidaan periaatteessa suorittaa minkä tahansa kahden arvoviittauksen välillä, mutta käytännössä vertailu tarkoituksenmukaista suorittaa vain, mikäli vertailtavat arvot ovat samaa tyyppiä. Vertailuilmauksen syntaksin BNF-esitys on seuraava.

```
comparison ::= value_reference " " (">" | "=" | "<" | "><")
" " value_reference;
```

Erityisesti seuraavat kyselykielen primitiivisten arvojen vertailut ovat semanttisesti mielekkäitä:

- lukujen vertailu suuruuden perusteella
- merkkijonojen vertailu aakkosjärjestyksen perusteella
- päivämäärien vertailu aikajärjestyksen perusteella

Vertailtaessa lukuja ja merkkijonoja merkkijono on järjestyksessä luvun jälkeen ts. se on vertailuarvoltaan tätä suurempi. Tilanne on vastaava vertailtaessa päivämääriä ja merkkijonoja. Lukujen ja päivämäärien keskinäinen järjestys sen sijaan ei ole hyvin määritelty.

Kahden arvoviittauksen, joista molempien arvona on joko olio, semanttinen yhteys tai dokumentti, välillä semanttisesti mielekkäitä ovat yhtäsuuruus- (sama kuin) ja erisuuruus- (eri kuin) vertailut. Mikäli toinen yhtäsuuruusvertailun kohteena olevista arvoviittauksista on muuttuja, merkitsee tämä muuttujan arvottamista vertailtavan arvoviittauksen arvoon. Jos molemmat vertailtavista arvoviittauksista ovat muuttujia, tulkitaan muuttujat samoiksi. Dokumentteihin liittyvien XML-polkujen osalta muuttujia sisältävissä yhtäsuuruusvertailuissa

noudatetaan edellä esitettyjä periaatteita. Erityisesti samaan dokumenttiin liittyvän täydellisen ja/tai siihen liittyvien ei-täydellisten polkujen väliset yhtäsuuruusvertailut onnistuvat aina.

Vertailuoperaattoreiden ja tyyppimäärittelyjen avulla voidaan suorittaa esimerkkikyselyn 3 mukainen tavanomainen ekstensionaalinen kysely. Tällä kyselyllä etsitään henkilötunnuksen ”`151251-215Q`” omaavan henkilön (muuttuja `P`) äiti (muuttuja `M`).

kysely

```
M where person P, parenthood H, H:child = P,P:ssn = '151251-215Q', person M,
H:parent = M, M:sex = 'female'.
```

vastaus

```
M = person '120828-235A'
```

Esimerkkikysely 3. Esimerkki ekstensionaalisesta kyselystä.

7.4.2. Semanttisten yhteyksien selvittämiseen tarkoitetut primitiivit

Kyselykieli sisältää seitsemän primitiiviä, joilla voidaan selvittää semanttisia yhteyksiä. Kyselykielen kannalta kelvollinen semanttinen yhteys täyttää seuraavat ehdot.

- Muodostettaessa välillistä yhteyttä olioiden A ja B välille voi välitön semanttinen yhteys, johon osallistuu olio A tai B, olla vain yhteyden ensimmäinen tai viimeinen osayhteys.
- Yhteys ei saa sisältää syklejä yhteyden muodostavien olioiden tai välittömien semanttisten yhteyksien kautta. Täsmällisesti tämä merkitsee, että:
    - Yhteyttä muodostettaessa jokaista kahden olion yhdistämiseen käytettyä oliota voidaan käyttää tässä tarkoituksessa vain kerran.
    - Sama välitön semanttinen yhteys ei voi esiintyä välillisessä yhteydessä useammin kuin kerran.

Ehtojen keskeinen tavoite on rajoittaa hyväksyttävien yhteyksien määrää ja niissä esiintyvää redundanssia. Ehdot estävät myös kyselyn prosessoinnin päättymättömyyden.

Edellä esitetty *tyyppimäärittely* sopii tietokannan sisältämien välittömien semanttisten yhteyksien löytämiseen, kun yhteystyyppi on käyttäjän tuntema. Jatkossa esiteltävät primitiivit laajentavat kielen ilmaisuvoimaa sallimalla näiden lisäksi välillisten ja tyypiltään tuntemattomien välittömien semanttisten yhteyksien selvittämisen.

Primitiivi sc_between on tarkoitettu kaikkien kahden olion välillä vallitsevien semanttisten yhteyksien selvittämiseen. Sen täsmällinen syntaksi on seuraava.

```
primitive1 ::= "sc_between(" object_reference ", "
object_reference ", " semantic_connection_reference ")";
```

Molempien olioviittausten ollessa muuttujia, joihin ei kohdistu lisäehtoja, primitiiviä voidaan käyttää löytämään kaikki semanttisesti yhdistettävissä olevat oliot ja niiden väliset semanttiset yhteydet. Jos vain toinen olioviittauksista viittaa ennalta tuntemattomaan olioon, evaluoinnin tuloksena voidaan selvittää tunnettuun olioon semanttisesti liittyvät oliot sekä niiden yhteydet tunnettuun olioon. Primitiivin käyttö voi perustua myös kahden tunnetun olion välisen semanttisten yhteyden palauttamiseen tai tällaisen yhteyden olemassaolon tarkistamiseen. Erilaisten käyttötapojen määrä on huomattavan suuri, sillä yleisessä tapauksessa olioiden tunnettuus vaihtelee ääripäiden yksikäsitteisesti tunnettu ja täysin tuntematon välillä.

Esimerkkikyselyssä 4 tällä primitiivillä selvitetään, onko olemassa semanttista yhteyttä (muuttuja SC) henkilön (muuttuja P), jonka henkilötunnus on ”151251-215Q”, ja rekisterinumeron ”HUS-412” omaavan auton (muuttuja C) välillä.

kysely

```
SC where person P, P:ssn = '151251-215Q', car C, C:register = 'HUS-412',
sc_between(P, C, SC).
```

vastaus

```
SC =
  "person '151251-215Q' works in enterprise 'Darwin's Bakery',
  and person '130347-372S' works in enterprise 'Darwin's Bakery',
  and person '130347-372S' is the owner of car 'HUS-412'."
```

Esimerkkikysely 4. Primitiivin sc_between soveltaminen kyselyssä.

Primitiivillä sc_via selvitetään välilliset semanttiset yhteydet, joiden muodostamiseen on käytetty annettuja olioita. Ehtona on, että mikään annetuista olioista ei ole yhteyden pääteolio ts. toinen niistä olioista, joiden välistä semanttista yhteyttä yritetään selvittää. Annettujen olioiden lisäksi yhteyden muodostamiseen voidaan käyttää myös muita tietokannassa esiintyviä olioita. Kyselykielessä primitiivi esitetään syntaktisesti seuraavasti.

```
primitive2 ::= "sc_via([" object_reference_list "], "
semantic_connection_reference ")";
```

Määritelmässä object_reference_list tarkoittaa yhtä olioviittausta tai useaa tosistaan pilkuilla (,) erotettua olioviittausta.

Primitiivin ilmeisimmässä soveltamistavassa kaikki olioviittaukset liittyvät ennalta määriteltyihin olioihin. Sitä voidaan soveltaa myös kaikkien niiden olioiden selvittämiseen, joita välillisesti tarvitaan kahden olion yhdistämiseen. Tällöin on suositeltavaa, mikäli mahdollista, usean muuttujan listan sijasta käyttää vain yhtä muuttujaa, joka vastauksessa arvotetaan jokaiseksi näistä olioista. Menettelyllä vältetään erilaisten arvotusten kombinoinnista johtuva vastauksen koon huomattava kasvu.

Esimerkkikyselyssä 5 sovelletaan primitiiviä `sc_via`. Siinä etsitään primitiivin `sc_between` avulla mahdollinen semanttinen yhteys (muuttuja `SC`) henkilöiden, joiden henkilötunnukset ovat ”`130347-372S`” (muuttuja `P1`) ja ”`121272-459W`” (muuttuja `P2`), välillä. Esimerkkitietokannan semanttisia yhteyksiä esittävästä kuvasta 2 kuitenkin havaitaan, että tällaisia yhteyksiä on kaksi kappaletta. Primitiivin `sc_via` avulla voidaan vaatia, että yhteyden muodostamisessa tarvitaan yritystä Lifetime Insurance Company (muuttuja `E`), jolloin vastaus sisältää vain yhden arvotuksen muuttujalle `SC`.

kysely

```
SC where person P1, P1:ssn = '130347-372S', person P2, P2:ssn =
'121272-459W', enterprise E, E:name = 'Lifetime Insurance Company',
sc_between(P1,P2,SC), sc_via([E], SC).
```

vastaus

```
SC =
  "person '130347-372S' is the father of person '150762-115S',
  and person '150762-115S' works in enterprise 'Lifetime Insurance Company',
  and person '121272-459W' works in enterprise 'Lifetime Insurance Company'.
  "
```

Esimerkkikysely 5. Primitiivin `sc_via` soveltaminen kyselyssä.

Primitiivi `sc_not_via` on syntaktisesti analoginen primitiivin `sc_via` kanssa. Primitiivin avulla käyttäjä voi määrätä, että annettuja olioita ei voida käyttää semanttisen yhteyden muodostamiseen. Yleisessä tapauksessa tämä ei vastaa primitiivin `sc_via` negaatiota, joka hyväksyy myös kaikki sellaiset semanttiset yhteydet, joissa jokin tai jotkin annetuista olioista (mutta eivät kaikki) osallistuvat yhteyteen välillisesti. Primitiivin `sc_not_via` syntaksi on seuraava.

```
primitive3 ::= "sc_not_via([" object_reference_list "], "
semantic_connection_reference ")";
```

Myös primitiivissä `sc_not_via` voidaan periaatteessa käyttää tuntemattomiin olioihin viittaavia muuttujia. Tällöin käyttäjä on kiinnostunut kaikista niistä olioista, jotka eivät välillisesti osallistu semanttiseen yhteyteen. Koska ehtoa ei kuitenkaan voida kohdistaa yhtä aikaa kaikkiin semanttisiin yhteyksiin (kieli

ei tue universaalikvantifiointia), ei näin voida löytää esimerkiksi kaikkia niitä tietokannan olioita, jotka eivät ole semanttisesti yhdistettävissä mihinkään muuhun olioon. Näin tämä primitiivin käyttötapa ei ole yleensä tarkoituksenmukainen.

Oletuksena kyselykielessä on, että semanttisen yhteyden muodostamiseen voidaan käyttää kaikkia semanttisia yhteystyyppejä. Primitiivillä `sc_using` voidaan tämä oletuskäyttäytyminen kumota rajoittamalla yhteyden muodostamisessa käytettävien semanttisten yhteystyyppien joukkoa. Ehto ymmärretään inklusiivisen disjunktion tapaan; riittää, että yhteydessä on hyödynnetty ainakin yhtä annetuista tyypeistä. Toisaalta yhteys ei saa sisältää muita kuin primitiivillä annettuihin yhteystyyppeihin kuuluvia välittömiä osayhteyksiä. Primitiivin `sc_using` syntaksi esitetään seuraavassa kielioppisäännössä.

```
primitive4 ::= "sc_using([" semantic_connection_type_list
"], " semantic_connection_reference ")";
```

Nonterminaali `semantic_connection_type_list` muodostuu yhdestä semanttisen yhteystyypin nimestä tai useasta pilkulla (,) toisistaan erotetusta yhteystyypin nimestä.

Esimerkkikyselyssä 6 on esimerkkikyselyn 5 alkuasetelma. Siinä kuitenkin primitiivin `sc_using` avulla vaaditaan, että semanttisen yhteyden (muuttuja `SC`) muodostamiseen voidaan käyttää vain `parenthood`-yhteystyypin ilmentymiä.

kysely

```
SC where person P1, P1:ssn = '130347-372S', person P2, P2:ssn = '121272-
459W', sc_between(P1,P2,SC), sc_using([parenthood], SC).
```

vastaus

```
SC =
  "person '130347-372S' is the father of person '150762-115S',
  and person '150735-542Y' is the mother of person '150762-115S',
  and person '150735-542Y' is the mother of person '040560-212F',
  and person '040560-212F' is the mother of person '040185-100Y',
  and person '130845-772E' is the father of person '040185-100Y',
  and person '130845-772E' is the father of person '121272-459W'."
```

Esimerkkikysely 6. Primitiivin `sc_using` soveltaminen kyselyssä.

Annettuihin yhteystyyppeihin kuuluvien välittömien osayhteyksien esiintyminen semanttisessa yhteydessä voidaan kieltää primitiivillä `sc_not_using`. Primitiivin `sc_using` negaatio ei ole tähän yleisesti riittävän rajoittava ehto, sillä se hyväksyy myös yhteydet, joissa esiintyy primitiivissä annettuihin yhteystyyppeihin kuulumattomien yhteyksien lisäksi yksi tai useampi annettuihin yh-

teystyyppeihin kuuluva välitön osayhteys. Primitiivin `sc_not_using` täsmällinen syntaksi on seuraava.

```
primitive5 ::= "sc_not_using(["
semantic_connection_type_list "], "
semantic_connection_reference ")";
```

Viimeinen esiteltävistä semanttisten yhteyksien selvittämiseen tarkoitetuista primitiiveistä on `sc_max_length`. Primitiivin avulla voidaan hyväksyttävien semanttisten yhteyksien pituudelle asettaa yläraja. Yhdessä negaationsa kanssa sitä voidaan epäsuorasti käyttää myös yhteyksien pituuden vaihteluvälin määräämiseen. Yhteyden enimmäispituuden määrittely lienee kuitenkin käytännössä primitiivin tarkoituksenmukaisin käyttötapa. Luvussa 2 esitetyllä tavalla sitä voidaan käyttää usein, mutta ei aina, semanttisen yhteyden relevanssin arvioimiseen. Primitiivin syntaksi on esitetty seuraavassa.

```
primitive6 ::= "sc_max_length(" integer ", "
semantic_connection_reference ")";
```

7.4.3. Dokumenttien käsittelyyn tarkoitetut primitiivit

Dokumenttien ja olioiden käsittelyn integrointi tapahtuu kyselykielessä primitiivin `refers` avulla. Primitiivillä voidaan selvittää spesifein osadokumentti, josta on viittaus olioon. Tämä kyselykielen suunnittelun yhteydessä tehty valinta on merkityksellinen. Vaihtoehtoisesti vastaukseen voitaisiin ottaa mukaan myös kaikki sellaiset dokumentit ja osadokumentit, joiden mielivaltaisella sisäkkäisyyden tasoilla olevista alielementeistä viittaus olioon on tehty. Mahdollista olisi myös huomioida vastauksessa ainoastaan viittauksen sisältävät kokonaiset XML-dokumentit. Valitun viittauspiirteen etuina ovat kyselyn vastauksen koon rajaaminen minimiin samalla säilyttäen siihen sisältyvän informaation yksityiskohtaisuus. Jos kaikki viittaavan elementin yläelementit tai ainoastaan ylimmän tason elementit otettaisiin mukaan vastaukseen, ei käyttäjällä olisi enää yksinkertaista mahdollisuutta erottaa sitä välitöntä elementtiä (so. osadokumenttia), joka sisältää viittauksen. Kuitenkin tämä tieto saattaa olla loppukäyttäjän kannalta informatiivinen. Esimerkiksi kauppakirjaa vastaavan dokumentin osalta voidaan olla kiinnostuneita siitä, esiintyykö viitattava henkilö siinä myyjän vai ostajan roolissa. Tieto-orientoituneessa XML-dokumentissa tämä tieto käy yleensä ilmi jo olioon viittaavan elementin nimestä. Elementtiin liittyvä polkuilmaus voidaankin usein tulkita olion dokumentissa olevaksi roolimäärittelyksi. Esimerkinomainen polku ”`bos1.xml/bill_of_sale/buyer`” havainnollistaa tätä. Primitiivillä `refers` on seuraava syntaksi.

```
primitive7 ::= "refers(" document_reference ", "
object_reference ")";
```

Molempien viittauksien kohdistuessa ennalta määrittelemättömiin olioihin tai dokumentteihin (ts. ne ilmaistaan muuttujilla), voidaan primitiiviä käyttää

palauttamaan kaikki viitatut oliot niihin liittyvine dokumentteineen. Vastaavasti, jos vain toinen ilmauksen argumenteista on muuttuja, saadaan tulokseksi joko kaikki annettuun olioon liittyvät dokumentit tai kaikki annettuun dokumenttiin liittyvät oliot. Jos halutaan esimerkiksi selvittää kaikki oliot, joihin on viitattu XML-dokumentista koe.xml jollain sisäkkäisyyden tasolla, voidaan tämä ilmaista ehdolla "`refers('koe.xml'/_, O)`". Muuttuja "`_`" voi siinä saada arvokseen XML-dokumentin koe.xml jokaiseen osadokumenttiin liittyvän osapolun. Primitiivi ei siis estä kokonaisten XML-dokumenttien käsittelyä yksittäisten osadokumenttien sijasta.

Esimerkkikyselyssä 7 haetaan kaikki henkilötunnuksen "`130347-372S`" omaavaan henkilöön (muuttuja `P`) liittyvät kokonaiset XML-dokumentit (muuttuja `MD`) ja spesifeimmät viittaavat osadokumentit (muuttuja `D`).

kysely

```
MD,D where person P, P:ssn = '130347-372S', refers(D,P), D = MD/_.
```

vastaus

```
MD = 'C:\dr4.xml'
D = 'C:\dr4.xml'/damage_report/vehicle/driver

MD = 'C:\bos6.xml'
D = 'C:\bos6.xml'/bill_of_sale/buyer
```

Esimerkkikysely 7. Primitiivin `refers` soveltaminen kyselyssä.

Kyselykieli sisältää myös primitiivin yksinkertaisen sana- ja fraasihaun suorittamiseen dokumenteissa. Fraasilla tarkoitetaan tässä yhteydessä mitä tahansa dokumentissa peräkkäin esiintyvistä sanoista muodostuvaa jonoa. Ratkaisevaa fraasin muodostumisen kannalta on sanojen fyysinen järjestys dokumentissa. Samaan fraasiin sisältyvät sanat voivat tämän mukaisesti sijaita myös eri alielementeissä. Tällainen fraasihaku on hyödyllinen erityisesti dokumenttiorientoituneiden XML-dokumenttien yhteydessä, joissa XML-elementeillä HTML:n tapaan korostetaan tekstin loogisia rakenneosia. Tarkkuusperiaatteen mukaiseksi fraasin sisältäväksi elementiksi tulkitaan tässä tapauksessa spesifein elementti, johon kaikki fraasin sisältämät sanat sisäkkäisyyden jollain tasolla sisältyvät.

Sana- ja fraasihaku ilmaistaan kyselykielessä primitiivillä `contains`. Primitiivillä on seuraava syntaksi.

```
primitive8 ::= "contains(" document_reference ", "
string ")";
```

Syntaksin mukainen merkkijono tulkitaan sanajonoksi. Näin ollen sen täsmäytyksessä ei oteta huomioon sanojen välillä esiintyvien tyhjämerkkien määrää, erikoismerkkejä tai isojen ja pienten kirjainten välistä eroa.

Esimerkkikyselyssä 8 haemme kaikki spesifeimmät (osa-)dokumentit (muuttuja `D`), jotka sisältävät fraasin "`front wheel`". Dokumenteista annetaan vastauksessa myös niiden elementtitason jäsennetty esitys näennäisen attribuutin `content` avulla.

kysely

```
D, D:content where contains(D,'front wheel').
```

vastaus

```
D = 'C:\dr2.xml'/damage_report/vehicle1/description_of_damages
D:content =
  <description_of_damages>
    the front wheel broke off
  </description_of_damages>
```

Esimerkkikysely 8. Primitiivin `contains` soveltaminen kyselyssä.

8. Kyselykielen toteutus

Tämän luvun tarkoituksena on antaa lukijalle yleiskuva kyselykielen toteutustavasta. Kohdassa 8.1. esitellään kyselykielen toteutuksen tärkeimmät osat: niiden tehtävät ja karkea toimintamekanismi. Tätä yleisemmällä ja pohdinnallisemmalla tasolla tarkastellaan kohdassa 8.2. menetelmiä, joilla semanttisten yhteyksien selvittämiseen tarkoitettujen kyselyiden evaluointia voidaan tehostaa Prolog-ympäristössä. Lukijan oletetaan tuntevan Prolog-ohjelmoinnin keskeiset piirteet.

8.1. Toteutuksen osat

Kyselykielen prototyyppi on toteutettu LPA-WinProlog –ympäristössä. Prologin valinta toteutusympäristöksi nopeuttaa toteutusta, sillä Prologin sisältämää automaattista teoreemantodistusmekanismia voidaan käyttää sellaisenaan deklaratiivisten kyselyjen prosessointiin, mikä vähentää tarvittavan ohjelmakoodin määrää. DCG:n (Definite Clause Grammar) avulla toteutukseen liittyvät jäsennystehtävät, XML-dokumenttien lukeminen ja käyttäjän antamien kyselyiden jäsennys, voidaan esittää perinteisten kielioppisääntöjen avulla.

Kyselykielen toteuttava ohjelmisto muodostuu seuraavista pääosista:

1. käyttöliittymä
2. XML-tietokannan lukija
3. vapaamuotoisten dokumenttien lukija

4. kyselykielen jäsennin
5. kyselyprimitiivejä vastaavat predikaatit
6. XML-jäsennin

Nämä esitellään seuraavissa alakohdissa.

8.1.1. Käyttöliittymä

Käyttäjän käyttöliittymä vastaa sekä kyselyiden vastaanotosta ja niiden käsittelyn delegoimisesta ohjelmiston muille osille että kyselyn tuloksen esittämisestä. Käyttöliittymä käynnistetään predikaatilla `do_query/0`. Käyttöliittymän keskeinen osa on vuorovaikutteinen silmukka, joka vastaanottaa käyttäjän syötteet ja välittää ne edelleen prosessoitavaksi. Se on toteutettu valmispredikaattien `repeat/0` ja `fail/0` avulla. Jos silmukka muodostettaisiin sijoittamalla silmukassa toistettava osa erilliseen predikaattiin, joka prosessoinnin päätteeksi kutsuisi rekursiivisesti itseään, kasvaisi Prologin prosessointiin käytetty muistipinon koko jokaisella silmukan suorituskerralla. Tämä voi lopulta johtaa muistitilan loppumiseen. Peruutuksen ansiosta muistitila vapautetaan ennen silmukan uudelleen suorittamista.

Kyselyiden lisäksi käyttäjä voi antaa kyselytulkille komentoja. Komento ”`quit.`” lopettaa ohjelmiston suorituksen. Erityinen istuntokohtainen asetus on sorting, jota käytetään poistamaan vastauksesta siinä useammin kuin kerran esiintyvät arvotukset eli duplikaatit. Asetuksen tilaa voidaan muuttaa antamalla kyselytulkille komento ”`sort.`”. Yksinkertainen duplikaatteja palauttava kysely on ”`A where neg A = B.`”, jossa muuttujien `A` ja `B` arvotuksessa otetaan huomioon kaikki tietokannasta johdettavissa olevat erilaiset kahden olion muodostamat parit. Duplikaattien poistamiseen käytetty `setof/3`-valmispredikaatti aiheuttaa lisäksi arvotusten järjestämisen. Järjestäminen edellyttää kaikkien arvotusten prosessointia ja tallentamista johonkin tietorakenteeseen ennen ensimmäisen vastaukseen sisältyvän arvotuksen palauttamista käyttäjälle. Koska erityisesti semanttisia yhteyksiä käsittelevät kyselyt voivat palauttaa huomattavan suuren määrän arvotuksia, joista käyttäjä on mahdollisesti kiinnostunut vain muutamasta ensimmäisestä, vaatii tällainen prosessointistrategia runsaasti ylimääräisiä muisti- ja aikaresursseja. Tämän vuoksi duplikaattien poistamista ei oletusarvoisesti tehdä.

Käyttöliittymässä käyttäjän antamat syötteet luetaan valmispredikaatilla `etoks/2`. Predikaatti jäsentää automaattisesti syötetyt merkit Prologin ISO-standardin mukaisiksi termeiksi: kokonaisluvuiksi, desimaaliluvuiksi, merkkijonoiksi, muuttujiksi jne. Termit niihin liittyvine tyyppitietoineen välitetään kyselykielen jäsentimelle, joka yrittää niiden jäsentämistä kyselyiksi. Jos jäsentäminen onnistuu, saadaan jäsennyksen tuloksena kyselyä vastaavien Prolog-tavoitteiden konjunktio, jonka oikeaksi todistamista yritetään valmispredikaatilla `call/1`. Oikeaksi todistamisen onnistuessa tavoitekonjunktiossa esiintyvät

tulokseen sisältyvät alustamattomat muuttujat arvottuvat, ja näin muodostuva kyselyn tulos esitetään näytöllä. Jos syötteen jäsentäminen kyselyksi ei onnistu eikä syöte vastaa mitään järjestelmän tuntemaa komentoa, annetaan virheilmoitus ”`Syntax error`”.

8.1.2. XML-tietokannan lukija

XML-tietokannan lukija muuntaa tietokannan XML-esityksen faktoilla ilmaistuksi Prolog-esitykseksi ja tallentaa näin luodun Prolog-tietokannan istunnon ajaksi muistiin valmispredikaatilla `assert/1`. Tietokannan luku käynnistetään predikaatilla `open_xml_db/0`. Predikaatin suorituksen ensimmäisessä vaiheessa etsitään kaikki tietokantaan kuuluvat XML-dokumentit, ja lajitellaan perustuen siihen, onko kyseessä oliotyypin, olioiden, semanttisen yhteystyypin vai semanttisten yhteyksien määrittelyyn tarkoitettu dokumentti. Haussa huomioidaan vain ne tiedostot, joiden tiedostopäätteenä on ”`.xml`”. Käsittelemättä jätetään XML-dokumentit, joiden juurielementtiä ei ole nimetty edellä esitetyn XML-esitystavan mukaisesti. Juurielementin nimi selvitetään lukemalla tiedoston alusta riittävä määrä merkkejä, minkä ansiosta koko dokumenttia ei tarvitse tutkia. Dokumenttien etsiminen alkaa aktiivisena olevasta hakemistosta ja etenee sen kaikkiin alihakemistoihin.

Kun dokumentit on lajiteltu, luetaan ensin oliotyyppien ja semanttisten yhteystyyppien määrittelyt sisältävät dokumentit ja tallennetaan niitä vastaavat faktat Prolog-tietokantaan. Tiedostojen lukemiseen käytetään toteutuksen XML-jäsennintä. Mikäli jokin tiedostoista ei noudata XML-syntaksia tai määriteltyä XML-esitystapaa, annetaan virheilmoitus, ja kyseinen tiedosto sivuutetaan käsittelyssä. Olio- ja yhteystyyppejä vastaavien dokumenttien lukemisen jälkeen luetaan dokumentit, joissa määritellään tietokannan oliot ja välittömät semanttiset yhteydet. Mikäli tiedostot rikkovat XML-syntaksia tai määriteltyä XML-esitystapaa, niitä ei käsitellä ja annetaan virheilmoitus. Jos vain yksittäistä oliota tai semanttista yhteyttä vastaava elementti ei noudata määriteltyä XML-esitystä tai sen määrittely ei ole minkään tietokannassa esiintyvän olio- tai yhteystyypin mukainen, sivuutetaan vain kyseinen elementti. Luettu tietokanta poistetaan muistista predikaatilla `close_xml_db/0`. Tämä hävittää valmispredikaatin `abolish/1` avulla muistista kaikki tietokantaan liittyvät faktat.

8.1.3. Vapaamuotoisten dokumenttien lukija

Toteutukseen sisältyvää vapaamuotoisten dokumenttien lukijaa voitaisiin tietyin edellytyksin verrata hakukoneeseen, sillä se ylläpitää pysyviä indeksejä vapaamuotoisista dokumenteista sekä niissä esiintyvistä sanoista ja viittauksista tietokannan olioihin. Sana- ja viittausindeksien organisoinnissa on kyse tavanomaista käänteistiedostoa vastaavasta tietorakenteesta [Järvelin ja Kekäläinen, 2002]. Indeksien tarkoituksena on nopeuttaa kyselyiden prosessointia siten, ettei jokaisen dokumenttiin kohdistuvan ehdon evaluoimiseksi ole tarvetta

jäsentää ja lukea kaikkia vapaamuotoisia dokumentteja. Toisaalta käänteisrakenteeseen ei toteutuksessa tallenneta kaikkea dokumentteihin sisältyvää tietoa, esimerkiksi dokumenttien rakennepuita tai niihin sisältyvien elementtien XML-attribuutteja. Tämän johdosta kyselyyn vastaaminen saattaa edellyttää myös dokumenttien lukuoperaatioita, mikä on käänteistiedostoja käyttävissä kyselykielissä tavallista (vrt. [Järvelin ja Kekäläinen, 2002]).

Dokumentti-indeksiin tallennetaan tiedot tunnetuista vapaamuotoisista dokumenteista: dokumenttien tiedostonimet sekä niiden viimeisimmät muokkausajankohdat. Sanaindeksi on organisoitu dokumenteissa esiintyvien sanojen mukaisesti. Jokaiselle sanalle on määritelty indeksissä lista, joka sisältää sanan sijaintitiedot dokumenteissa. Yksittäinen sijaintitieto sisältää sekä XML-polun spesifeimpään sanan sisältävään elementtiin että järjestysluvun, joka osoittaa sanan fyysisen sijainnin dokumentissa suhteessa muihin dokumentin sisältämiin sanoihin. Viittausindeksi on analoginen sanaindeksin kanssa, mutta sanojen sijasta se sisältää tiedot dokumenteissa viitatuista olioista. Viittauksien osalta tallennetaan vain niihin liittyvät XML-polut. Nimiavaruudella `ref` määriteltyä viittauselementtiä ei itseään huomioida indeksoinnissa.

Vapaamuotoisten dokumenttien lukija käynnistetään predikaatilla `scan_doc_paths/0`. Kyselyohjelmiston suorituksen aluksi vapaamuotoisten dokumenttien lukija päivittää indeksejä vain siltä osin kuin on tarpeellista. XML-tietokannan lukijan tavoin vapaamuotoisten dokumenttien lukija aloittaa dokumenttien etsimisen ja lukemisen aktiivisena olevasta hakemistosta ja etenee rekursiivisesti sen kaikkiin alihakemistoihin. Lukija käsittelee tiedostot, joiden tiedostopäätteenä on ”`.xml`”. Kutakin dokumenttia verrataan sen indeksissä olevaan aikaleimaan. Mikäli dokumenttia ei löydy indeksistä, luetaan dokumentti ja lisätään indekseihin sitä vastaavat tiedot. Jos taas dokumenttia on aikaleiman perusteella päivitetty, poistetaan ensin indekseistä kaikki siihen liittyvät tiedot, minkä jälkeen lisätään uudet päivitetyt tiedot. Jos jotakin dokumentti-indeksiin sisältyvää dokumenttia ei löydetä etsintäprosessin missään vaiheessa, poistetaan indekseistä lopuksi kaikki tällaiseen dokumenttiin liittyvät tiedot. Indeksoinnissa ei huomioida dokumentteja, jotka kuuluvat varsinaiseen XML-tietokantaan.

8.1.4. Kyselykielen jäsennin

Kyselykielen jäsennin on toteutettu Prologiin sisältyvällä DCG-logiikkakieliopilla (Definite Clause Grammar). DCG soveltuu paitsi sen toteamiseen, että kysely on tarkasteltavan kieliopin mukainen, myös muuntamaan kyselyyn sisältyvät ehdot niitä vastaavaksi Prologin tavoitekonjunktioksi. LPA-WinPrologin sisältämä DCG-toteutus sallii minkä tahansa Prologin listamuotoa noudattavan tiedon jäsentämisen. Näin ollen valmispredikaatilla `etoks/2` suoritetun alustavan jäsennyksen tuloksena saatu lista syötteeseen si-

sältyvistä termeistä voidaan jäsentää sellaisenaan DCG:llä. Alustavan jäsennyksen ansiosta kielioppi voidaan ilmaista huomattavasti tiiviimmin kuin muutoin olisi mahdollista.

Toteutuksessa on pyritty noudattamaan DCG-sääntöjen vasemmalle ositusta, mitä on käsitelty mm. Kumpulaisen [2003] pro-gradu –tutkielmassa. Vasemmalle osituksen ansiosta kielioppisääntöjen prosessointia on mahdollista nopeuttaa.

8.1.5. Kyselyprimitiivejä vastaavat predikaatit

Kyselyn tulososan jäsentämisen seurauksena saadaan tavoitekonjunktio, joka sisältää kyselyprimitiivejä vastaavat predikaatit. Pääsääntöisesti jokaista kyselyprimitiiviä vastaa tavoitekonjunktiossa yksi predikaatti.

Negaatio on määritelty erikseen jokaiselle tavoitekonjunktioon sisältyvälle predikaatille. Kyselykielen negaatiota ei voida suoraan ilmaista Prologin negaatiolla (operaattori ”`\+`”), koska ne eroavat toisistaan. Prologissa yksinkertainen kysely ”`\+ tuntee(matti,X).`”, missä `matti` on vakio ja `X` muuttuja, on tosi vain, jos tietokannassa ei ole löydettävissä ilmausta ”`tuntee(matti,X).`”, missä `X`:n paikalla on jokin vakio tai muuttuja. Jos ilmaukseen ”`tuntee(a,b).`” on liitettävissä tulkinta ”a tuntee b:n”, on tällaiseen kyselyyn liitettävissä Prolog-tietokannan suljetussa maailmassa tulkinta ”matti ei tunne ketään”. Koska tarkasteltavassa kyselykielessä muuttuja pyritään automaattisesti arvottamaan jokaiseen tietokannan sisältämään olioon, vastaava epäsuoran universaalikvantifioinnin sisältävä kyselytyyppi ei ole kielessä mahdollinen. Prologin yhteydessä tämä merkitsisi, että muuttuja `X` pyrittäisiin samaistamaan johonkin tietokannassa muussa yhteydessä esiintyvään vakioon, esimerkiksi vakioon `z`, jolloin väite ”`\+ tuntee(matti,z).`” olisi tosi.

Semanttisen yhteyksien käsittely on yhdistetty niin, että kaikki samaan semanttiseen yhteyteen kohdistuvat ehdot ilmaistaan yhdellä predikaatilla. Predikaatissa on määritelty argumentti jokaiselle semanttisten yhteyksien selvittämiseen tarkoitetulle primitiiville ja sen negaatiolle. Perustelut kaikkien semanttiseen yhteyteen kohdistuvien ehtojen yhdistämiselle samaan predikaattiin esitetään kohdassa 8.2. Nykyinen toteutustapa ei vielä salli useamman samaan semanttiseen yhteyteen kohdistuvan samalla primitiivillä ilmaistavan ehdon sisällyttämistä kyselyyn. Tämän mahdollistamiseksi jokaisen argumentin paikalla tulisi käyttää listaa, jonka alkioina olisivat edellisen mukaiset yksittäistä primitiiviä tai sen negaatiota vastaavat argumentit.

8.1.6. XML-jäsennin

Toteutuksessa on käytetty XML-jäsennintä, joka on toteutettu DCG-logiikkakieliopilla. Jäsennin tukee vain yksinkertaisia XML-dokumentteja, jotka eivät elementtien lisäksi sisällä muita XML:n rakenteellisia piirteitä: kommentteja, prolog-osuutta jne. Suorituksen aluksi XML-tiedosto luetaan ja muunne-

taan ASCII-koodeista muodostuvaksi listaksi. Tiedoston sisältämiä sarkain- ja rivivaihtomerkkejä ei muunnoksessa huomioida. Tämän jälkeen kutsutaan valmispredikaatin `phrase/2` avulla XML-dokumentin jäsentämiseen tarkoitettua kielioppisääntöä, jolle muodostettu lista annetaan syötteenä. Jäsennyksen tuloksena saadaan XML-dokumentin rakenteen Prolog-faktalla ilmaistu puumainen esitys. Jäsennin voi käsitellä dokumenttiin sisältyviä rakenteettoman tekstin jaksoja sellaisenaan tai purkaa ne yksittäisistä sanoista muodostuviksi osalistoiksi. Jälkimmäisessä tapauksessa jätetään huomiotta välilyönnit, välimerkit ja muut sanoihin kuulumattomat merkit sekä muunnetaan kaikki sanoissa esiintyvät isot kirjaimet pieniksi. Näin XML-jäsennintä voidaan hyödyntää helposti esimerkiksi sanaindeksien luomisessa.

8.2. Toteutuksen tehostaminen

Suoraviivainen prosessointitapa semanttisten yhteyksien selvittämiseen tarkoitetuille kyselyille on jokaisen tietokannasta johdettavissa olevan semanttisen yhteyden muodostaminen ja vertaaminen kyselyssä annettuihin ehtoihin. Tämä prosessointitapa ei kuitenkaan ole aina tehokkain mahdollinen. Tehokkaamman prosessoinnin mahdollistamiseksi tarvitaan menetelmiä, joilla voidaan rajoittaa käsiteltävien semanttisten yhteyksien määrää tai ainakin keskeyttää kyselyn kannalta epärelevanttien semanttisten yhteyksien käsittely mahdollisimman aikaisessa vaiheessa. Koska toteutuskielenä käytetään Prologia, on lisäksi huomioitava Prologin omasta prosessointitavasta johtuvat erityispiirteet. Menetelmissä keskeisenä tehokkuuskriteerinä pidetään prosessointiin kuluvaa aikaa: ei niinkään vaadittavaa muistitilaa.

8.2.1. Semanttisten yhteyksien prosessointi ja käänteinen käsittely

Ohjelmakatkelmassa 4 määritelty `semantic_connection1/3` on yksinkertainen, semanttisten yhteyksien selvittämiseen soveltuva predikaatti. Se ei sellaisenaan sisälly toteutukseen, mutta auttaa Prolog-ohjelmoinnin perusteet tuntevaa lukijaa ymmärtämään tavan, jolla semanttisia yhteyksiä prosessoidaan.

Predikaatin `semantic_connection1/3` argumenteista `A` ja `B` ovat semanttisesti yhdistetyt oliot ja `Cons` niiden välinen semanttinen yhteys, joka esitetään välittömistä semanttisista yhteyksistä koostuvana listana. Semanttisen yhteyden muodostaminen aloitetaan oliosta `A` ja päätetään olioon `B`. Jatkossa kutsutaan `A`:ta yhteyden alkuolioksi, `B`:tä yhteyden loppuolioksi ja molempia sen pääteolioiksi. Predikaatille `semantic_connection1/4` on annettu kaksi määrittelyä. Riviltä 6 alkava määrittely vastaa välillisen yhteyden muodostamiseen käytettävää rekursioaskelta ja vastaavasti riviltä 3 alkava määrittely rekursion päättävää loppuehtoa. Predikaatilla `directly_connected/3` oletetaan voitavan hakea tietokannasta kaikki kahden olion väliset välittömät semanttiset yhteydet. Sen määrittelyssä voidaan hyödyntää intensionaalisen tason tietoa niistä oliotyypeistä, jotka osallistuvat semanttisiin yhteystyyppeihin. Näin vältetään

kaikkien välittömien semanttisten yhteyksien läpikäynti. Esimerkissä oletamme yksinkertaistaen, että välittömään semanttiseen yhteyteen voi osallistua vain kaksi oliota. Tämä helpottaa käsittelyä; ei ole esimerkiksi tarpeen erikseen tarkistaa syklien muodostumista välillisen yhteyden osayhteyksien suhteen.

```
1  semantic_connection1(A,B,Cons):-
2  semantic_connection1(A,B,[A],Cons).

3  semantic_connection1(A,B,AccObjs,[Con]):-
4  directly_connected(A,B,Con),
5   \+ member(B,AccObjs).

6  semantic_connection1(A,C,AccObjs,[Con|Cons]):-
7  directly_connected(A,B,Con),
8  B \== C,
9  \+ member(B,AccObjs),
10 semantic_connection1(B,C,[B|AccObjs],Cons).
```

Ohjelmakatkelma 4. Semanttisen yhteyden muodostamiseen soveltuva predikaatti `semantic_connection1/3`.

Määrittelystä huomataan, että sekä rekursioaskelta että rekursion loppuehtoa vastaavien sääntöjen rungoissa esiintyy kaksi identtistä tavoitetta. Pahimmassa tapauksessa rekursion mielivaltaisessa vaiheessa voidaan soveltaa sekä rekursioaskelta että loppuehtoa, jolloin samat tavoitteet todistetaan oikeaksi kahteen kertaan. Tämä voidaan välttää määrittelemällä predikaatin semantic_connection1/4 sijasta kaksi keskenään rekursiivista predikaattia. Tätä vastaa ohjelmakatkelma 5.

Semanttisten yhteyksien selvittämiseen soveltuvaa predikaattia `semantic_connection2/3` voidaan Prologissa käyttää usealla erilaisella tavalla. Tunnettujen alku- ja pääteolioiden sijasta voimme käyttää niiden paikalla alustamattomia muuttujia. Alustamaton muuttuja on samaistettavissa mihin tahansa tietokannassa esiintyvään olioon. Näin predikaatin käyttö ei rajoitu kahden tunnetun olion välisten kaikkien semanttisten yhteyksien etsimiseen, vaan sen avulla voidaan hakea myös kaikki tunnetusta oliosta alkavat semanttiset yhteydet ja niihin liittyvät tuntemattomat pääteoliot. Päinvastaisessa tapauksessa voidaan selvittää kaikki tunnettuun olioon päättyvät semanttiset yhteydet tuntemattomine alkuolioineen. Sitä voidaan käyttää myös kaikkien tietokannasta johdettavissa olevien semanttisten yhteyksien selvittämiseen. Tällöin sekä alku-, että pääteoliot ovat tuntemattomia eli niitä vastaavat alustamattomat muuttujat.

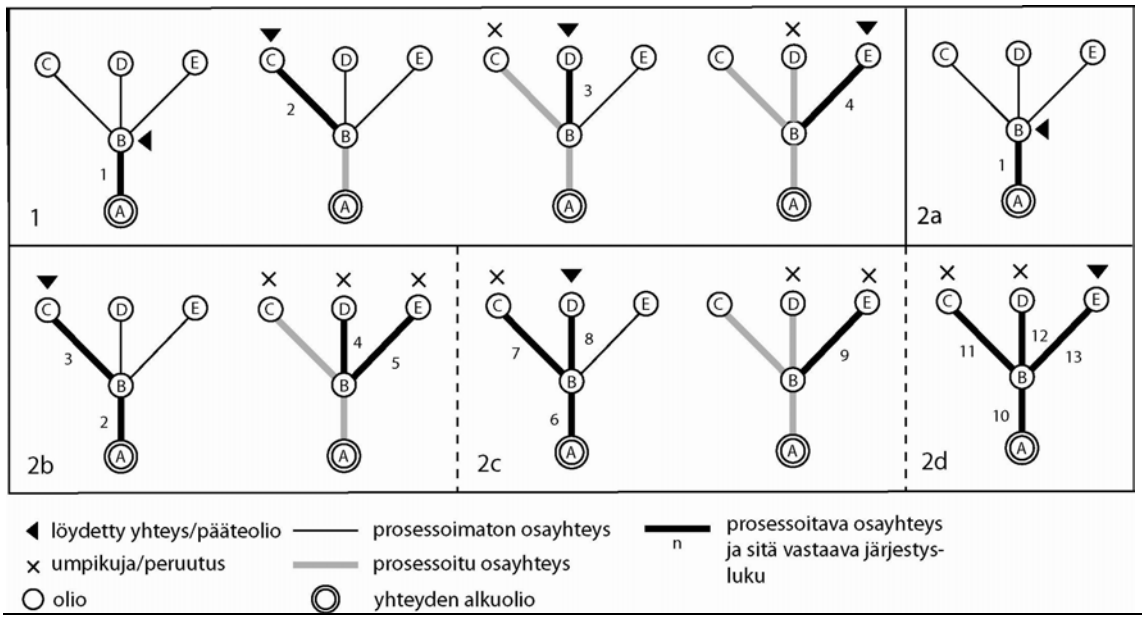
```
1  semantic_connection2(A,C,Cons):-
2  semantic_connection2(A,C,[A],Cons).

3  semantic_connection2(A,C,AccObjs,[Con|Cons]):-
4  directly_connected(A,B,Con),
5  \+ member(B,AccObjs),
6  semantic_connection2(A,B,C,AccObjs,Cons).

7  semantic_connection2(A,B,B,_,[]).

8  semantic_connection2(A,B,C,AccObjs,Cons):-
9  B \== C,
10 semantic_connection2(B,C,[B|AccObjs],Cons).
```

Ohjelmakatkelma 5. Semanttisen yhteyden muodostamiseen soveltuva predikaatti `semantic_connection2/3`.

Mikäli alkuolion paikalla käytetään alustamatonta muuttujaa, sovelletaan predikaatin `semantic_connection2/5` rekursioaskelta jokaiseen tietokannan sisältämään välittömään semanttiseen yhteyteen. Jos jokainen olio osallistuu vähintään yhteen semanttiseen yhteyteen, samaistetaan yhteyden alkuoliota vastaava muuttuja `A` tämän seurauksena ainakin kerran jokaiseen tietokannassa määriteltyyn olioon. Jokaisesta tällaisesta arvotuksesta saa alkunsa itsenäinen rekursiohaara, joka vastaa syvyyshakua. Kun prosessoinnissa saavutetaan umpikuja, jossa semanttista yhteyttä ei voida enää kasvattaa, suoritetaan peruutus. Jos peruutuksen seurauksena palataan ensimmäiseen rekursioaskeleeseen, rekursiohaaran suoritus lopetetaan ja siirrytään mahdolliseen seuraavaan rekursiohaaraan. Jos pääteolio on tunnettu, jatketaan rekursiota, kunnes ensimmäinen pääteolioon päättyvä mahdollinen yhteys löydetään. Tämän jälkeen rekursiohaaran suoritusta jatketaan mahdollisten vaihtoehtoisten yhteyksien löytämiseksi. Jos loppuoliota ei tunneta, palauttaa rekursiohaara tuloksenaan vuorollaan jokaisen prosessoinnin yhteydessä kohdatun olion sekä siihen liittyvän semanttisen yhteyden.

Jos pääteolio on tuntematon ja alkuolio tunnettu, on tämä prosessointitapa tehokas. Jokaisen löydetyn pääteolion ja sitä vastaavan semanttisen yhteyden palauttamisen jälkeen ei rekursiohaarassa palata takaisin ensimmäiseen rekursioaskeleeseen ja aloiteta prosessointia alusta. Sen sijaan uusia yhteyksiä pyritään muodostamaan kumulatiivisesti aiemmin samassa rekursiohaarassa löydettyjen yhteyksien pohjalta. Huomattavasti tehottomampi olisi ratkaisu, jossa prosessoinnin aluksi tuntematonta pääteoliota vastaava muuttuja samaistettai-

siin vuorollaan jokaiseen tietokannan olioon ja jokaista tällaista alku- loppuolio –kombinaatiota kohden suoritettaisiin erillinen prosessointi. Tällöin kumulatiivisuudesta saavutettava hyöty menetettäisiin ja prosessointi palautuisi jokaisen käsitellyn pääteolion jälkeen tunnettuun alkuolioon. Tätä havainnollistetaan kuvassa 3.



Kuva 3. Semanttisten yhteyksien prosessointi ja siihen liittyvä syvyyshaun vaiheiden määrä, kun yhteyden alkuolio (A) on tunnettu ja loppuolio on tuntematon. Tapauksessa 1 yhteyden loppuoliota vastaavaa muuttujaa ei alusteta ennen prosessointia. Tapauksessa 2 loppuolio alustetaan ennen prosessointia vuorollaan tietokannan olioiksi B (tapaus 2a), C (tapaus 2b), D (tapaus 2c) ja E (tapaus 2d). Tapauksessa 2 prosessoinnin kumulatiivisuudesta saatu hyöty menetetään ja syvyyshaun vaiheiden määrä (13 kpl) on suurempi kuin tapauksessa 1 (4 kpl).

Prosessointitavan kannalta ongelmallisempi on se tilanne, jossa alkuolio on tuntematon ja pääteolio tunnettu. Myös tässä tapauksessa kumulatiivisuusvaikutus menetetään. Pahimmassa tapauksessa alkuoliota vastaava muuttuja samaistetaan vuorollaan jokaiseen tietokannan sisältämään olioon, ja kaikki niistä alkunsa saavat rekursiohaarat käsitellään kokonaisuudessaan. Tässä esiintyy kuitenkin mahdollisuus suorittaa yksinkertainen optimointi. Tällaista tapausta voidaan nimittäin käsitellä käänteisesti. Jos siis alkuolion paikalla on alustamaton muuttuja ja pääteolio on tunnettu, tapausta voidaan käsitellä ikään kuin alkuoliona käytettäisiin tunnettua pääteoliota ja pääteoliona tuntematonta alkuoliota. Tällöin tilanne palautuu edellisen mukaiseksi, ja prosessointi on mahdollista suorittaa tehokkaasti vain yhdestä oliosta alkavilla rekursiohaaroilla.

Käänteisen käsittelyn ongelmana on kuitenkin, että myös löydettävien semanttisten yhteyksien listaesitykset ovat alkuperäiseen nähden käänteisiä. Suoraviivainen ratkaisu olisi niiden kääntäminen valmispredikaatilla `reverse/2`. Ongelmana on, että tuloksena saatavien semanttisten yhteyksien kääntäminen vaatii ylimääräistä suoritusaikaa. Yleensä listan kääntäminen on suoritettavissa lineaarisessa ajassa sen pituuteen nähden. Koska kääntäminen suoritetaan erikseen jokaiselle tuloksena saatavalle yhteydelle, ei kumulatiivisesti rekursiohaaran suorituksen aikana, joudutaan sama osayhteys kääntämään mahdollisesti useaan kertaan. Tämän vuoksi ei ole helposti todettavissa, onko rekursiohaarojen vähenemisen seurauksena säästyvä suoritusaika suurempi kuin aika, joka menetetään kääntämisen seurauksena.

```
1  tr1(A,C,[f(A,C)]):-
2  f(A,C).

3  tr1(A,C,[f(A,B)|Xs]):-
4  f(A,B),
5  tr1(B,C,Xs).

6  tr2(A,C,Xs):-
7  tr2(A,C,[],Xs).

8  tr2(A,C,Acc,[f(A,C)|Acc]):-
9  f(A,C).

10 tr2(A,C,Acc,Xs):-
11 f(A,B),
12 tr2(B,C,[f(A,B)|Acc],Xs).
```

Ohjelmakatkelma 6. Listarakenteiden muodostaminen normaaliin rekursiiviseen tapaan (predikaatti `tr1/3`) ja kerääjätekniikalla (predikaatti `tr2/3`).

Selvästi parempi tapa on kerryttää käänteistä yhteyttä jo rekursioaskeleessa. Tällöin jokainen osayhteys joudutaan rekursiohaaran suorituksen yhteydessä kääntämään vain kerran. Voimme soveltaa havaintoa, että Prologin kerääjätekniikka ja normaali rekursiivinen tapa tuottaa listoja johtavat toisiinsa nähden käänteisten listojen muodostumiseen. Olkoon Prolog-tietokannassa faktat, ”`f(a,b).`”, ”`f(b,c).`” ja ”`f(c,d).`". Ohjelmakatkelmassa 6 on määritelty näiden faktojen transitiiviseen läpi käymiseen soveltuvat predikaatit `tr1/3` ja `tr2/3`. Predikaatti `tr/1` on toteutettu tavallisella rekursiivisella menetelmällä, kun taas predikaatin `tr/2` määrittelyssä on käytetty kerääjätekniikkaa. Predi-

kaatit laskevat transitiivisen sulkeuman ja keräävät siihen liittyvät faktat listaan.

Kun suoritamme kyselyn ”`tr1(a,d,X).`”, saamme `X`:n arvotukseksi listan ”`[f(a,b),f(b,c),f(c,d)]`”. Vastaavasti suoritettaessa kysely ”`tr2(a,d,X).`” `X`:n arvotukseksi saadaan lista ”`[f(c,d),f(b,c),f(a,b)]`”.

```
1  semantic_connection3(A,B,Cons):-
2  reverse_handling(A,B),
3  reverse_semantic_connection(B,A,[B],[],Cons).

4  semantic_connection3(A,B,Cons):-
5  \+ reverse_handling(A,B),
6  semantic_connection2(A,B,[A],Cons).

7  reverse_semantic_connection(A,C,AccObjs,AccCons,Cons):-
8  directly_connected(A,B,Con),
9  \+ member(B,AccObjs),
10 reverse_semantic_connection(A,B,C,AccObjs,[Con|AccCons],Cons).

11 reverse_semantic_connection(A,B,B,_,Cons,Cons).

12 reverse_semantic_connection(A,B,C,AccObjs,AccCons,Cons):-
13 B \== C,
14 reverse_semantic_connection(B,C,[B|AccObjs],AccCons,Cons).

15 reverse_handling(A,B):-
16 var(A),
17 nonvar(B).
```

Ohjelmakatkelma 7. Semanttisen yhteyden muodostamiseen soveltuva predikaatti `semantic_connection3/3`.

Voimme nyt soveltaa tätä semanttisten yhteyksien muodostamiseen. Korvaamme ohjelmakatkelmassa 5 annetun predikaatin `semantic_connection2/3` predikaatilla `semantic_connection3/3`. Tämän lisäksi määrittelemme uuden predikaatin `reverse_semantic_connection/5`, jolla voimme selvittää käänteisen semanttisen yhteyden. Se on analoginen predikaattiin `semantic_connection2/4` nähden, mutta yhteyden kerryttämiseen käytetään siinä kerääjätekniikkaa. Predikaattia `reverse_handling/2` käytetään sen päättämiseksi, onko yhteys tarpeen muodostaa käänteisesti. Predikaattien määrittelyt on annettu ohjelmakatkelmassa 7.

Käänteisen käsittelyn ansiosta prosessoitavien rekursiohaarojen määrä vähenee merkittävästi. Sen sijaan että alkuolion ilmaiseva muuttuja pahimmassa tapauksessa samaistettaisiin vuorollaan jokaiseen tietokannan olioon (mahdollisesti useaan kertaan), suoritetaan nyt ainoastaan tunnetusta pääteoliosta alkavat rekursiohaarat. Käänteisessä käsittelyssä prosessointi etenee vain sellaisiin olioihin, jotka ovat semanttisesti yhteydessä pääteolioon. Jos prosessointia ei suoritettaisi käänteisesti, jouduttaisiin lisäksi käymään läpi pääteolioon semanttisesti yhdistämättömistä olioista alkavat rekursiohaarat.

Mikäli molemmat pääteoliot ovat tunnettuja, on prosessointi aina tehokasta ja edellyttää vain yhdestä oliosta alkavien rekursiohaarojen suorittamista. Jos molemmat pääteoliot ovat tuntemattomia, joudutaan joka tapauksessa käymään läpi jokainen erilainen rekursiohaara, eikä toteutusta ei ole mahdollista tehostaa millään suoraviivaisella tavalla.

8.2.2. Rekursion keskeyttäminen mahdollisimman aikaisessa vaiheessa

Kyselykielen semanttisten yhteyksien selvittämiseen tarkoitetut primitiivit eroavat toisistaan sen suhteen, kuinka aikaisessa prosessoinnin vaiheessa yhteys voidaan todeta epärelevantiksi ja hylätä. Semanttisten yhteyksien selvittämiseen tarkoitetut primitiivit negaatioineen jaetaan tässä yksinkertaisesti kolmeen luokkaan. Jaottelussa myös primitiivien negaatiot otetaan huomioon, sillä useissa tapauksissa ne edustavat eri luokkaa kuin primitiivi itse. Ensimmäiseen luokkaan kuuluviin ehtoihin liittyen syvyyshakua vastaava rekursio voidaan keskeyttää. Toiseen luokkaan kuuluvien ehtojen osalta keskeytys voidaan suorittaa vain osissa tapauksissa. Kolmanteen luokkaan kuuluvien ehtojen tapauksessa rekursiohaara on aina suoritettava kokonaisuudessaan, ja kaikkia löydettyjä semanttista yhteyksiä on verrattava primitiivillä annettuun ehtoon. Tarkastelu suoritetaan perustuen siihen, että semanttisten yhteyksien muodostamiseen käytetään jotain edellä määriteltyä predikaattia yhteyden alku- ja pääteolioiden ollessa tuntemattomia. Toisin sanoen tarkastelemme tilannetta, jossa prosessointimekanismi ilman primitiiveillä annettavia ehtoja pyrkisi muodostamaan kaikki tietokannasta johdettavissa olevat semanttiset yhteydet.

Primitiivit negaatioineen sijoittuvat luokkiin seuraavasti.

1. `sc_using`, `sc_not_using`, `sc_max_length`
2. `sc_between`, `sc_not_via`, `neg sc_via`, `neg sc_between`
3. `sc_via`, `neg sc_not_via`, `neg sc_using`, `neg sc_not_using`, `neg sc_max_length`

Ensimmäiseen luokkaan kuuluvien primitiivien osalta jaottelu on mahdollista suorittaa yksiselitteisesti. Primitiiveihin `sc_not_using` ja `sc_using` liittyen rekursio voidaan keskeyttää heti, kun yhteyden muodostamiseen on käytetty välitöntä semanttista yhteyttä, joka edustaa primitiivin kieltämää tai jotain muuta kuin sen sallimaa semanttista yhteystyyppiä. Primitiivin

sc_max_length perusteella rekursio voidaan keskeyttää heti, kun muodostettavan semanttisen yhteyden pituus on ylittänyt sallitun rajan.

Toiseen luokkaan kuuluvien ehtojen rajoittavuus riippuu niissä esiintyvistä alustamattomista muuttujista. Primitiivi sc_between vastaa käytöltään edellä määriteltyjä semanttisen yhteyden selvittämiseen soveltuvia predikaatteja, joten sille on voimassa kaikki edellä esitetty. Erityisesti tapauksessa, jossa molemmat olioihin viittaavat muuttujat ovat alustamattomia, ei prosessointia voida rajoittaa. Primitiiviin sc_not_via liittyen rekursio voidaan keskeyttää heti, kun yhteyden muodostamiseen on käytetty kiellettyä oliota. Jos jotain kielletyistä olioista vastaa alustamaton muuttuja, ei keskeyttäminen ole siihen perustuen kuitenkaan mielekästä. Tällaisen muuttujan arvotus on syytä suorittaa vasta yhteyden muodostamisen jälkeen. Silloin se voidaan vuorollaan arvottaa jokaiseen yhteydessä esiintymättömään olioon, joka ei ole mikään primitiivillä annetuista muista olioista. Primitiivin sc_via negaation perusteella prosessoitava yhteys voidaan todeta epärelevantiksi heti, kun sen muodostamisessa on käytetty jokaista primitiivillä annettua oliota. Jos jotain olioista vastaa alustamaton muuttuja, on sen kohdalla käsittely suoritettava kuten edellä. Primitiivin sc_between negaatio on rajoittavuudeltaan luokkaan sisältyvistä ehdoista heikoin. Siihen perustuen rekursiohaaran suoritus voidaan keskeyttää vain, mikäli yhteyden kielletty alkuolio on tunnettu. Tässä parhaassakin tapauksessa voidaan keskeyttää rekursiohaaroista vain ne, jotka alkavat kyseisestä oliosta. Prosessointiin voitaisiin sc_between-primitiivin tavoin soveltaa edellä esitettyä käänteistä prosessointitapaa, joskaan näin saatava hyöty ei olisi kovin merkittävä.

Primitiivi sc_via on kolmannen luokan tyypillinen edustaja. Sitä vastaavan ehdon pätemättömyyden osoittaminen ts. sen todentaminen, että jokin primitiivillä annettu olio ei välillisesti osallistu yhteyteen, edellyttää aina koko yhteyden muodostamista (yhteyden viimeistä välitöntä osayhteyttä lukuun ottamatta). Primitiivin sc_not_via negaatio vastaa sc_via-ehtoa jonkin siinä määritellyn olion suhteen. Täten myös se kuuluu selvästi kolmanteen luokkaan. Myös primitiivien sc_using ja sc_not_using negaatiot ovat liian heikkoja rekursion keskeyttämiseen. Edellisessä tapauksessa joudumme osoittamaan, että jokin yhteyteen sisältyvä välitön osayhteys (mahdollisesti vasta viimeinen) edustaa yhteystyyppiä, joka ei sisälly annettuihin yhteystyyppeihin. Ymmärrettävästi yhteyttä ei voida tämän perusteella todeta epärelevantiksi ennen sen muodostamista kokonaisuudessaan. Jälkimmäisessä tapauksessa ehdon todentaminen edellyttää, että jokin yhteyteen sisältyvä välitön osayhteys edustaa yhteystyyppiä, joka sisältyy annettuihin yhteystyyppeihin. Myös tämän ehdon evaluointi edellyttää aina koko yhteyden muodostamista. Primitiivin sc_max_length negaatio, jolla asetetaan yhteyden pituuden (avoin) alaraja, ei sekään mahdollista prosessoinnin keskeyttämistä.

Suoraviivainen kyselykielen toteutustapa Prolog-ympäristössä on kyselyyn sisältyvien ehtojen muuntaminen kielen primitiivejä vastaavista predikaateista muodostuvaksi tavoitekonjunktioksi, joka sitten pyritään todistamaan oikeaksi valmispredikaatilla `call/1`. Tämän mukaisesti voisimme määritellä jokaiselle semanttisten yhteyksien selvittämiseen soveltuvalle primitiiville sitä vastaavan predikaatin, joka määriteltäisiin edellä esiintyneen predikaatin `semantic_connection3/3` tapaan ja sisältäisi lisäksi joko luokkien 1 ja 2 mahdollistamat rekursioaskelten aikaiset tarkistukset tai/ja luokkien 2 ja 3 edellyttämät jälkitarkistukset. Toteutuksesta tulisi tällöin kuitenkin varsin tehoton. Tehottomuutta aiheuttaa se, että tällaisessa ratkaisussa joudumme jokaisen samaan semanttiseen yhteyteen kohdistuvan ehdon evaluoinnin yhteydessä suorittamaan kaikki yhteyden muodostamiseen tarvittavat prosessoinnit.

Tehokkuutta voitaisiin parantaa sisällyttämällä predikaattien määrittelyihin metaloogiset tarkastelut, joilla erotetaan toisistaan tapaukset, joissa yhteyttä vastaa alustamaton muuttuja sekä tapaukset, joissa yhteyttä vastaava muuttuja on jo arvotettu siihen kohdistuvien aiempien ehtojen evaluoinnin seurauksena. Jälkimmäisessä tapauksessa olisi riittävää tarkistaa valmiiksi muodostetun semanttisen yhteyden täsmäävyys primitiivillä annettuihin ehtoihin. Tällainenkaan ratkaisu ei kuitenkaan hyödyntäisi täysimääräisesti luokkiin 1 ja 2 kuuluvien primitiivien rekursiohaaroja pienentävää vaikutusta. Olisi esimerkiksi mahdollista, että kyselyä vastaavassa konjunktiossa esiintyisi ennen luokkaan 1 kuuluvaa ehtoa jokin samaan semanttiseen yhteyteen kohdistuva luokan 3 ehto. Tällöin prosessoiduksi tulisi joka tapauksessa jokainen tietokannan sisältämä semanttinen yhteys.

Voimme edelleen tehostaa toteutusta järjestämällä kyselyitä vastaavat konjunktiot niin, että niissä predikaatit esiintyvät niiden luokan mukaisessa järjestyksessä. Tämäkään ei kuitenkaan ole optimaalisen tehokasta. Koska huomioon otetaan vain yksi luokkiin 1 tai 2 kuuluva ehto kerrallaan, voi tällaisen ehdon käsittely kuitenkin aiheuttaa sitä seuraavan, jompaankumpaan luokkaan kuuluvan ehdon kannalta turhaa prosessointia. Sitä tehokkaampaa on kaikkien samaan semanttiseen yhteyteen kohdistuvien ehtojen huomioon ottaminen yhdessä predikaatissa.

Yhteen predikaattiin perustuvan toteutustavan seurauksena kielen jäsentäminen Prolog-ehtojen konjunktioksi ja ehtojen prosessointi kuitenkin hankaloituvat. Semanttisten yhteyksien selvittämiseen tarkoitettuun predikaattiin täytyy tällöin määritellä alkuarvoiltaan alustamattomat argumentit jokaista primitiiveissä ja niiden negaatioissa esiintyvää argumenttia kohti. Ennen kuin uutta semanttisten yhteyksien hakuun tarkoitettua primitiiviä vastaavat ehdot voidaan lisätä konjunktioon, täytyy tarkistaa, sisältyykö kerrytettyyn konjunktioon jo kyseiseen semanttiseen yhteyteen liittyvä tavoite. Mikäli sisältyy, valitaan argumentti, johon ehto kohdistuu ja liitetään siihen primitiivillä annetut para-

metrit. Muussa tapauksessa lisätään konjunktioon uusi tavoite. Tavoitetta evaluoitaessa täytyy olla mahdollista selvittää, mihin tavoitteessa esiintyviin argumentteihin ehtoja on kohdistettu, ja sopeuttaa suoritettava prosessointi niihin.

8.2.3. Kyselyprimitiivejä vastaavien ehtojen järjestäminen tavoitekonjunktiossa

Kysely sisältää yleensä myös muita kuin semanttisten yhteyksien selvittämiseen tarkoitettuja primitiivejä kuten tyyppimäärittelyjä, vertailuja ja dokumenttien käsittelyyn tarkoitettuja primitiivejä negaatioineen. Tällaisilla primitiiveillä voidaan muun muassa alustaa jokin `sc_between`-, `sc_via`- tai `sc_not_via`-ehdossa tai niiden negaatiossa esiintyvä olioon viittaava muuttuja. Suoritusaikaan vaikuttaa usein se, tapahtuuko tällaisen ehdon evaluointi ennen vai jälkeen vastaavan semanttisen yhteyden prosessointia. Tarkastellaan tilannetta, jossa tällaisella ehdolla alustetaan muuttuja ennen siihen liittyvän semanttisen yhteyden muodostamista. Jos muuttuja voidaan ehdon seurauksena arvottaa useammaksi kuin yhdeksi olioksi, seuraa jokaista uudelleenarvotusta kohti myös semanttisen yhteyden selvittämiseen tarkoitetun predikaatin uudelleenevaluointi.

Suoritusajan kannalta erityisen epätoivottavia ovat tilanteet, joissa semanttiseen yhteyteen kohdistuu vain luokkaan 3 kuuluvia ehtoja. Tällöin jokaisen uudelleenevaluoinnin yhteydessä joudutaan muodostamaan kaikki tietokannasta johdettavissa olevat semanttiset yhteydet. Toisaalta, jos semanttiseen yhteyteen ja muuttujaan kohdistuu rekursiota voimakkaasti rajoittava (luokkaan 1 tai 2 kuuluva) ehto, voi tarvittavan prosessoinnin määrä jopa vähentyä. Näin käy esimerkiksi tilanteessa, jossa ehdon seurauksena yhteyden alkuolion ilmaiseva alustamaton muuttuja voidaan kaikkien olioiden sijasta arvottaa joksikin olioiden osajoukoksi.

Pääsääntönä voidaan siis pitää, että luokkaan 3 kuuluviin ehtoihin liittyviä olioon viittaavia muuttujia ei pidä alustaa ennen semanttisen yhteyden prosessointia. Kyselyn evaluointia voidaan tehostaa järjestämällä tavoitekonjunktiossa semanttisen yhteyden muodostava tavoite ennen muita muuttujan alustavia tavoitteita. Luokkaan 2 kuuluvien ehtojen osalta tehokasta suoritusjärjestystä ei voida todeta yhtä suoraviivaisesti. Esimerkiksi primitiivi `sc_not_via` mahdollistaa rekursion rajoittamisen tehokkaasti vain, jos siinä esiintyvät muuttujat ovat alustettavissa yksikäsitteisesti. Pahimmassa tapauksessa, jos muuttuja voidaan ehtojen perusteella alustaa jokaiseksi tietokannan olioksi, voi tarvittavan prosessoinnin määrä jopa lisääntyä. Näin ollen ei ole suoraviivaisesti osoitettavissa, onko muuttujan alustavan ehdon evaluointi primitiiviin liittyen tehokasta ennen vai jälkeen yhteyden prosessoinnin.

Optimaalinen suoritusjärjestys riippuu siis vahvasti annettujen ehtojen tosiasiallisesta rajoittavuudesta ja primitiivien yleisimmistä käyttötavoista. Tämän vuoksi mahdollisuus yleispätevien optimointisääntöjen esittämiseen on rajallinen. Mahdollisesti kyseeseen voisivat tulla luonteeltaan heuristiset optimointimenetelmät. Koska toteutus on tarkoitettu yksinomaan kyselykielen demonstrointiin, ei näin laajaan optimointiin ole tässä yhteydessä kuitenkaan tarvetta.

9. Kyselytyypit ja esimerkkikyselyt

Tässä luvussa esitellään tutkielman kyselykielen kannalta keskeiset kyselytyypit. Jokaisesta kyselytyypistä annetaan yksi tai useampi esimerkkikysely. Tutkielman kyselykielen kannalta mielenkiintoiset kyselyt jakaantuvat kahteen pääluokkaan, joita ovat:

1. semanttisten yhteyksien selvittämiseen tarkoitetut kyselyt tietokannassa
2. kyselyt tietokannan ja vapaamuotoisten dokumenttien välillä

Näiden lisäksi kyselykieli mahdollistaa luvussa 7 esitetyt perinteiset ekstensionaaliset tietokantakyselyt sekä vapaamuotoisten dokumenttien joukkoon rajoittuvat XML-kyselyt, joissa ei anneta tietokantaan kohdistuvia ehtoja. Nämä ovat kuitenkin niin tavanomaisia jo vakiintuneissa tietokanta- ja XML-kyselykielissä, että niitä ei tässä luvussa erikseen käsitellä. Kyselytyyppien ja -esimerkkien tarkoituksena onkin esitellä monipuolisesti kielen tarjoamia uusia mahdollisuuksia ja käyttömuotoja pikemminkin kuin esittää tyhjentävä, jokaisen käytännön tilanteen kattava typologia.

9.1. Semanttisten yhteyksien selvittämiseen tarkoitetut kyselyt tietokannassa

Semanttisten yhteyksien selvittämiseen soveltuvat kyselyt jaetaan tässä kolmeen käyttötapaukseen ja niitä edustavaan kyselytyyppiin. Niistä ilmeisin on tilanne, jossa haluamme selvittää olioiden välisen semanttisen yhteyden. Riippuen siitä, ovatko oliot yksikäsitteisesti tunnettuja vai eivät, voidaan tällaista kyselytyyppiä hyödyntää usealla erilaisella tavalla. Alakohdassa 9.1.1. näistä kaikista annetaan esimerkit. Toiseksi voimme olla kiinnostuneita niistä olioista, joiden kautta kaksi oliota voidaan välillisesti yhdistää toisiinsa. Tällainen käyttötapa on hyödynnettävyydeltään edellistä marginaalisempi. Yhtä hyvin voisimme tarkastella kaikkia niitä oliota, jotka liittyvät semanttisesti molempiin olioihin. Tällöin vastauksen koko olisi kuitenkin edellistä suurempi (Huomioon otettaisiin myös oliot, jotka liittyvät toiseen olioon välillisesti vain toisen olion kautta.), eivätkä siihen sisältyvät oliot yhtä suurella todennäköisyydellä liittyisi relevantisti molempiin olioihin. Joka tapauksessa kyselykieli mahdollistaa myös tällaisen kyselytyypin, ja siitä annetaan esimerkki alakohdassa 9.1.2. Kolmanneksi voimme olla kiinnostuneita kaikesta siitä olioon liittyvästä eksplisiittisestä tiedosta, joka on ilmaistavissa välittömillä semanttisilla yhteyksillä.

Tämä vastaa syntaktisesti ensin mainittua kyselytyyppiä toisen olioista ollessa tuntematon ja semanttisen yhteyden maksimipituuden ollessa yksi. Kyselyn käyttötarkoitus ja sen käyttömotivaatio ovat tässä tapauksessa kuitenkin niin erilaiset, että tämä käsitellään erillisenä kyselytyyppinä alakohdassa 9.1.3.

Yksikäsitteisesti tunnetulla oliolla tarkoitetaan esimerkkikyselyissä oliota, jonka avainattribuuttien arvot ovat käyttäjän tuntemat. Jos käyttäjä ei tunne oliota yksikäsitteisesti, voidaan olion tunnettujen piirteiden perusteella erottaa seuraavat tapaukset.

1. Olio on käyttäjälle täysin tuntematon.
2. Käyttäjä tuntee olion tyypin (esimerkiksi olio, joka on luokan `person` ilmentymä).
3. Käyttäjä tuntee olion joidenkin ei-yksilöivien attribuuttien arvot (esimerkiksi olio, jonka `size`-attribuutin arvo on `12`).
4. Käyttäjä tuntee olion tyypin ja joidenkin ei-yksilöivien attribuuttien arvot (esimerkiksi `car`-tyypin olio, jonka `color`-attribuutin arvo on ”`yellow`”).

Lisäksi olio voidaan kyselykielessä tunnistaa sen omista piirteistä riippumatta niiden semanttisten yhteyksien ja vapaamuotoisten dokumenttien tai niiden osadokumenttien perusteella, jotka liittyvät olioon tai joissa on viittaus siihen.

9.1.1. Kyselytyyppi 1: Olioiden välisten semanttisten yhteyksien selvittäminen

Esiteltävistä kyselytyypeistä keskeisin on olioiden välisten semanttisten yhteyksien selvittäminen. Esimerkkikysely 9 edustaa tapausta, jossa oliot ovat yksikäsitteisesti tunnettuja.

Kyselyllä selvitetään, onko henkilötunnuksen "`060685-517X`" omaava henkilö (muuttuja `P`) semanttisesti liitettävissä autoon (muuttuja `C`), jonka rekisterinumero on "`BGD-718`".

kysely

```
SC where person P, P:ssn = ’060685-517X’, car C, C:register = ’BGD-718’,
sc_between(P,C,SC).
```

vastaus

```
SC =
"person '171166-333F' is the mother of person '060685-517X',
and person '171166-333F' works in enterprise 'Advanced Experts Unlimited',
and enterprise 'Advanced Experts Unlimited' is the owner of car 'BGD-718'."
```

Esimerkkikysely 9. Kahden olion välillä mahdollisesti vallitsevien semanttisten yhteyksien selvittäminen, kun molemmat olioista ovat yksikäsitteisesti tunnettuja.

Esimerkkikysely 10 edustaa tapausta, jossa vain toinen oliosta tunnetaan yksikäsitteisesti. Kyselyssä etsitään henkilöt (muuttuja `P2`), jotka liittyvät perhesuhteiden kautta rekisterinumeron `"NV-313"` omaavan moottoripyörän (muuttuja `M`) omistajaan (muuttuja `P1`). Myös löydetyt semanttiset yhteydet (muuttuja `SC`) esitetään tuloksessa. Tämä vastaa siis semanttista yhteyttä yksikäsitteisesti tunnetun moottoripyörän `M` ja tuntemattoman henkilön `P2` välillä. Semanttisen yhteyden ensimmäinen välitön osayhteys, omistussuhde, määritellään eksplisiittisesti, ja siihen viitataan kyselyssä muuttujalla `O`.

kysely

```
P2:forename,P2:surname,P2,O,SC where motorcycle M, M:register = ’NV-313’,
person P1, owning O, O:owner = P1, O:vehicle = M, person P2,
sc_between(P2,P1,SC), sc_using([parenthood],SC).
```

vastaus

```
P2:forename = 'Annabelle'
P2:surname = 'Castaldo'
P2 = person '171166-333F'
O = "person '060685-517X' is the owner of motorcycle 'NV-313'."
SC = "person '171166-333F' is the mother of person '060685-517X'."
```

vastaus jatkuu

```
P2:forename = 'Ferdinand'
P2:surname = 'Castaldo'
P2 = person '060659-219K'
O = "person '060685-517X' is the owner of motorcycle 'NV-313'."
SC = "person '060659-219K' is the father of person '060685-517X'."
```

Esimerkkikysely 10. Olioiden välillä mahdollisesti vallitsevien semanttisten yhteyksien selvittäminen, kun vain toinen olioista on yksikäsitteisesti tunnettu.

Esimerkkikysely 11 edustaa tapausta, jossa kumpaakaan olioista ei tunneta yksikäsitteisesti. Siinä ylinopeutta ajaneen henkilön (muuttuja `P`) henkilötunnus ja auton (muuttuja `C`) rekisterinumero eivät ole tunnettuja. Sen sijaan molemmista olioista tunnetaan joidenkin avaimeen kuulumattomien attribuuttien arvot. Esimerkissä ajajan tiedetään olevan mies, ja hänellä oletetaan olevan ajokortti. Auton tiedetään olevan hopean värinen ja sen huippunopeuden suurempi tai yhtä suuri kuin 200 km/h. Tarkoituksena on etsiä potentiaalisia ajaja-ajoneuvo –kombinaatioita tarkastelemalla tietokannan semanttisia yhteyksiä. Koska vain relevanteimmat yhteydet halutaan huomioida, määritellään semanttisen yhteyden (muuttuja `SC`) maksimipituudeksi 4.

kysely

```
C,P,SC where car C, C:color = 'silver', C:top_speed >= 200, person P, P:sex
= 'male', P:licence = 'yes', sc_between(C,P,SC), sc_max_length(4,SC).
```

vastaus

```
C = car 'USD-999'
P = person '130845-772E'
SC = "person '130845-772E' is the owner of car 'USD-999'."

C = car 'USD-999'
P = person '040185-100Y'
SC =
  "person '130845-772E' is the owner of car 'USD-999'
  and person '130845-772E' is the father of person '040185-100Y'."
```

vastaus jatkuu

```
C = car 'USD-999'
P = person '152260-140G'
SC =
  "person '130845-772E' is the owner of car 'USD-999',
  and person '130845-772E' is the father of person '121272-459W',
  and person '121272-459W' is the mother of person '050495-620U',
  and person '152260-140G' is the father of person '050495-620U'."

C = car 'USD-999'
P = person '050495-620U'
SC =
  "person '130845-772E' is the owner of car 'USD-999',
  and person '130845-772E' is the father of person '121272-459W',
  and person '121272-459W' is the mother of person '050495-620U'."
```

Esimerkkikysely 11. Olioiden välillä vallitsevien mahdollisten semanttisten yhteyden selvittäminen, kun kumpaakaan olioista ei tunneta yksikäsitteisesti.

9.1.2. Kyselytyyppi 2: Tunnettujen olioiden toisiinsa liittävien tuntemattomien olioiden selvittäminen

Primitiivi `sc_via` soveltuu semanttisen yhteyden muodostamisessa tarvittavien olioiden selvittämiseen. Esimerkkikyselyssä 12 haluamme selvittää kaikkien niiden henkilöiden (muuttuja `P2`) etu- ja sukunimet sekä puhelinnumerot, jotka liittävät toisiinsa henkilötunnuksen `"280643-005J"` omaavan henkilön (muuttuja `P1`) ja auton (muuttuja `C`), jonka rekisterinumero on `"ECC-333"`.

Käytännössä kyselytyyppi 2 voisi tulla kyseeseen esimerkiksi ajoneuvon luvattoman käyttöönoton tapauksessa, kun haluamme selvittää kaikkien tapauksen kannalta relevanttien henkilöiden yhteystiedot. Suoraviivaisempaa olisi tällöinkin selvittää tavallisella ekstensionaalisella kyselyllä kyseisen ajoneuvon omistaja ja tarvittaessa alakohdan 9.1.1. mukaisilla kyselyillä epäillyn tai ajoneuvon omistajan lähipiiri.

kysely

```
P2:forename, P2:surname, P2:phone_number, P2, SC where person P1, P1:ssn =
'280643-005J', car C, C:register = 'ECC-333', sc_between(P1,C,SC), person
P2, sc_via([P2],SC).
```

vastaus

```
P2:forename = 'Ann'
P2:surname = 'McCurley'
P2:phone_number = 512343
P2 = person '121272-459W'
SC =
  "person '280643-005J' is the mother of person '121272-459W',
  and person '121272-459W' is the mother of person '050495-620U',
  and person '152260-140G' is the father of person '050495-620U',
  and person '152260-140G' is the owner of car 'ECC-333'."

P2:forename = 'Codey'
P2:surname = 'McCurley'
P2:phone_number = 512343
P2 = person '152260-140G'
SC =
  "person '280643-005J' is the mother of person '121272-459W',
  and person '121272-459W' is the mother of person '050495-620U',
  and person '152260-140G' is the father of person '050495-620U',
  and person '152260-140G' is the owner of car 'ECC-333'."

P2:forename = 'Hugh'
P2:surname = 'McCurley'
P2:phone_number = 912355
P2 = person '050495-620U'
SC =
  "person '280643-005J' is the mother of person '121272-459W',
  and person '121272-459W' is the mother of person '050495-620U',
  and person '152260-140G' is the father of person '050495-620U',
  and person '152260-140G' is the owner of car 'ECC-333'."
```

Esimerkkikysely 12. Kahden olion välillä vallitseviin semanttisiin yhteyksiin välillisesti osallistuvien olioiden selvittäminen.

9.1.3. Kyselytyyppi 3: Olioon liittyvien välittömien semanttisten yhteyksien selvittäminen

Esimerkkikyselyllä 13 haluamme selvittää, mitä eksplisiittistä relationaalista tietoa tietokannassa on henkilötunnuksen `"150762-115S"` omaavasta henkilöstä.

kysely

```
SC where P:ssn = '150762-115S', sc_between(P, _,SC), sc_max_length(1,SC).
```

vastaus

```
SC = "person '150762-115S' works in enterprise 'Lifetime Insurance Compa-
ny'."

SC = "person '130347-372S' is the father of person '150762-115S'."

SC = "person '150735-542Y' is the mother of person '150762-115S'."
```

Esimerkkikysely 13. Olioon liittyvien välittömien semanttisten yhteyksien selvittäminen.

Tällaisella kyselyllä voidaan siis selvittää tunnettuun olioon liittyviä keskeisiä faktoja, jotka ilmaisevat sen, mihin olioihin oliolla on eksplisiittisesti ilmaistu semanttinen yhteys ja minkä yhteyden kautta. Onkin mielenkiintoista, että kyselykieli soveltuu myös tällaiseen selailutyyppiseen käyttöön. Koska välittömien semanttisten yhteyksien määrä on aina selvästi rajatumpi kuin välillisten yhteyksien määrä eikä niihin sisälly välillisille yhteyksille tyypillistä redundanssia, on vastaus käyttäjän kannalta tiivis ja informatiivinen. Kyselytyyppiä voidaan soveltaa yksikäsitteisesti tunnettujen olioiden ohella olioihin, joista tunnetaan vain tyyppi ja/tai joidenkin ei-yksilöivien attribuuttien arvot. Näin voitaisiin selvittää esimerkiksi kaikki nopeisiin autoihin liittyvät välittömät semanttiset yhteydet.

9.2. Kyselyt tietokannan ja vapaamuotoisten dokumenttien välillä

Tietokannan ja vapaamuotoisten dokumenttien väliset kyselyt sisältävät primitiivillä `refers` ilmaistuja ehtoja, jotka kytkevät toisiinsa dokumentit ja tietokannan loogiset rakenneosat. Oliot, niihin liittyvät välittömät semanttiset yhteydet ja vapaamuotoiset dokumentit tai niiden osadokumentit voivat olla käyttäjän yksikäsitteisesti tuntemia, tämän vain joiltaan piirteiltään tuntemia tai käyttäjälle täysin tuntemattomia. Erityisesti, jos käyttäjä ei tunne dokumenttia tai osadokumenttia yksikäsitteisesti, voidaan erottaa seuraavat tapaukset.

1. Se on käyttäjälle täysin tuntematon.
2. Käyttäjä tuntee osia sen rakenteesta (esimerkiksi, että se sisältää elementin `seller`).
3. Käyttäjä tuntee osia sen tekstisisällöstä (esimerkiksi, että se sisältää fraasin ”`as is`”).
4. Käyttäjä tuntee (yksikäsitteisesti tai ei) olion, johon on viittaus dokumentissa tai osadokumentissa (esimerkiksi, että dokumentti sisältää viittauksen `car`-tyypin olioon, jonka `color`-attribuutin arvo on ”`blue`”).

5. (Jokin kohtien 2 – 4 kombinaatio.)

Seuraavissa alakohdissa tietokannan ja vapaamuotoisten dokumenttien väliset kyselyt luokitellaan sen mukaan, halutaanko niillä selvittää tietokantaan, vapaamuotoisiin dokumentteihin vai molempiin sisältyvää informaatiota.

9.2.1. Kyselytyyppi 4: Tietokantaan kohdistuva kysely, jossa hyödynnetään vapaamuotoisissa dokumenteissa olevaa informaatiota

Tietokantaan kohdistuvissa kyselyissä voidaan vapaamuotoisissa dokumenteissa olevaa informaatiota hyödyntää monella tapaa. Voimme esimerkiksi liittää toisiinsa tietokannan olioita ja semanttisia yhteyksiä vapaamuotoisissa dokumenteissa olevien olioviittauksien perusteella. Voimme myös etsiä annetussa dokumentissa tai osadokumentissa viitattuja olioita ja niihin liittyviä semanttisia yhteyksiä.

Esimerkkikyselyssä 14 etsitään autoja (muuttuja `C`), jotka on vakuutettu vakuutusyhtiössä Tellurian Insurance Company (muuttuja `E`). Vakuutussopimusten juurielementit oletetaan nimetyn yhtenäisesti (nimenä `motor_vehicle_insurance`).

kysely

```
C where enterprise E, E:name = 'Tellurian Insurance Company', car C,
refers(D/motor_vehicle_insurance/_,E), refers(D/_,C).
```

vastaus

```
C = car 'CFU-462'

C = car 'UTA-211'
```

Esimerkkikysely 14. Tietokannan olioiden liittäminen toisiinsa vapaamuotoisten dokumenttien avulla.

Esimerkkikyselyssä 15 etsitään kolaridokumentin ”`C:\dr1.xml`” ”`injured_persons`”-elementeissä viitatut loukkaantuneet henkilöt (muuttuja `P1`) ja näihin läheisesti liittyvien henkilöiden (muuttuja `P2`) etu- ja sukunimet sekä puhelinnumerot. Myös henkilöt yhdistävät semanttiset yhteydet (muuttuja `SC`) esitetään tuloksessa. Läheisyyden kriteerinä käytetään hyväksyttäville semanttisille yhteyksille asetettavaa maksimipituutta 2.

kysely

```
P1, P2, P2:forename, P2:surname, P2:phone_number, SC where re-
fers('C:\dr1.xml'//injured_persons/_,P1), person P2, sc_between(P1,P2,SC),
sc_max_length(2,SC).
```

vastaus

```
P1 = person '040560-212F'
P2 = person '150735-542Y'
P2:forename = 'Kathrine'
P2:surname = 'Cassius'
P2:phone_number = 163222
SC = "person '150735-542Y' is the mother of person '040560-212F'."

P1 = person '040560-212F'
P2 = person '150762-115S'
P2:forename = 'Paula'
P2:surname = 'Campbell'
P2:phone_number = 572404
SC =
  "person '150735-542Y' is the mother of person '040560-212F'
  and person '150735-542Y' is the mother of person '150762-115S'."

P1 = person '040560-212F'
P2 = person '130845-772E'
P2:forename = 'Allan'
P2:surname = 'Selsor'
P2:phone_number = 418955
SC =
  "person '040560-212F' is the mother of person '040185-100Y'
  and person '130845-772E' is the father of person '040185-100Y'."

P1 = person '040560-212F'
P2 = person '040185-100Y'
P2:forename = 'Thomas'
P2:surname = 'Selsor'
P2:phone_number = 258008
SC = "person '040560-212F' is the mother of person '040185-100Y'."
```

Esimerkkikysely 15. Kysely vapaamuotoisesta dokumentista tietokantaan.

9.2.2. Kyselytyyppi 5: Vapaamuotoisiin dokumentteihin kohdistuva kysely, jossa hyödynnetään tietokannassa olevaa informaatiota

Yksinkertaisimmillaan vapaamuotoisiin dokumentteihin kohdistuva, tietokannan informaatiota hyödyntävä kysely tarkoittaa kyselyä, jossa haluamme löytää tunnettuun olioon viittauksen sisältäviä dokumentteja. Esimerkkikyselyssä 16

etsitään käytettyjen ajoneuvojen myyntidokumentteja (muuttuja `D`), joissa toisena sopimuskumppanina on henkilötunnuksen "`130845-772E`" omaava henkilö (muuttuja `P`). Esimerkissä oletetaan, että kaikki kauppakirjoja vastaavat XML-dokumentit (muuttuja `D`) sisältävät juurielementin `bill_of_sale`. Käytettyjä ajoneuvoja koskevat kauppakirjat tunnistetaan niissä esiintyvän fraasin "`as is`" perusteella.

kysely

```
D/bill_of_sale:content where person P, P:ssn = '130845-772E',
refers(D/bill_of_sale/_,P), contains(D/_,'as is').
```

vastaus

```
'C:\bos6.xml'/bill_of_sale:content =
  <bill_of_sale>
    The Seller
    <seller> <ref:person ssn = '130845-772E'> Allan Selsor </ref:person>
    </seller>
    , for payment of
    <price> 9500 euros </price>
    by the Buyer
    <buyer> <ref:person ssn = '130347-372S'> Alvin Baker </ref:person>
    </buyer>
    , has sold the vehicle
    <sold_vehicle> <ref:car register = 'HUS-412'> saab HUS-412 </ref:car>
    </sold_vehicle>
    to Buyer on
    <date> 2003-05-20 </date>
    . The ordometer reading of the vehicle at the moment of the sale is
    <ordometer_reading> 20000 km </ordometer_reading>
    . The Seller certifies that the Vehicle's odometer has not been altered.
    The vehicle is sold "as is".
  </bill_of_sale>
```

Esimerkkikysely 16. Kysely tietokannasta vapaamuotoisiin dokumentteihin.

9.2.3. Kyselytyyppi 6: Kysely, joka kohdistuu sekä tietokantaan että vapaamuotoisiin dokumentteihin

Usein on hyödyllistä selvittää kyselyllä sekä tietokantaan että vapaamuotoisiin dokumentteihin sisältyvää informaatiota. Tällainen kyselytyyppi on erityisen käytännöllinen tilanteissa, joissa mitään käsiteltävistä dokumenteista, olioista tai semanttisista yhteyksistä ei tunneta yksikäsitteisesti.

Esimerkkikyselyssä 17 etsitään kauppakirjoja (muuttuja `D`), joissa sopimusosapuolet ovat toisiinsa sukulaisuussuhteessa olevia henkilöitä (muuttujat `P1` ja `P2`). Erityisesti halutaan selvittää, missä rooleissa kyseiset henkilöt dokumenteissa esiintyvät (muuttujat `D1` ja `D2`). Kauppakirjojen juurielementit oletetaan nimetyn yhtenäisesti (nimenä `bill_of_sale`). Myös henkilöiden sukulaisuussuhdetta vastaavat semanttiset yhteydet (muuttuja `SC`) sisällytetään tulokseen.

kysely

```
D1,P1,D2,P2,SC where person P1, person P2, refers(D1,P1),refers(D2,P2),
D1 = D/bill_of_sale/_, D2 = D/_, sc_between(P1,P2,SC),
sc_using([parenthood],SC).
```

vastaus

```
D1 = 'C:\bos6.xml'/bill_of_sale/seller
P1 = person '130845-772E'
D2 = 'C:\bos6.xml'/bill_of_sale/buyer
P2 = person '130347-372S'
SC =
  "person '130845-772E' is the father of person '040185-100Y',
  and person '040560-212F' is the mother of person '040185-100Y',
  and person '150735-542Y' is the mother of person '040560-212F',
  and person '150735-542Y' is the mother of person '150762-115S',
  and person '130347-372S' is the father of person '150762-115S'."

D1 = 'C:\bos6.xml'/bill_of_sale/buyer
P1 = person '130347-372S'
D2 = 'C:\bos6.xml'/bill_of_sale/seller
P2 = person '130845-772E'
SC =
  "person '130347-372S' is the father of person '150762-115S',
  and person '150735-542Y' is the mother of person '150762-115S',
  and person '150735-542Y' is the mother of person '040560-212F',
  and person '040560-212F' is the mother of person '040185-100Y',
  and person '130845-772E' is the father of person '040185-100Y'."
```

Esimerkkikysely 17. Kysely, joka kohdistuu sekä tietokannassa että vapaamuotoisissa dokumenteissa olevaan informaatioon.

Huomaamme, että molemmat vastaukseen sisältyvät arvotukset liittyvät dokumenttiin "`C:\bos6.xml`". Arvotuksia saadaan kaksi kappaletta, sillä nor-

maaliin tapaan kumpikin henkilö otetaan vuorollaan huomioon muuttujien `P1` ja `P2` arvottamisessa.

10. Yhteenveto ja loppupäätelmät

Seuraavan sukupolven tietojärjestelmiksi (NGIS – Next Generation Information Systems) on ehdotettu deduktiivisista olio-orientoituneista tietomalleista johdettuja, mahdollisesti arvo- ja olio-orientoituneita käsittelytapoja yhdistäviä järjestelmiä. Ne tukevat mm. kyselykielten deklaratiivisuutta, tiedon loogista johdettavuutta eli deduktiota, kompleksisten entiteettien mallintamista, ekstensionaaliseen eli ilmentymätasoon kohdistuvien kyselyiden ohella myös intensionaalisen eli kaaviotasoon kohdistuvia kyselyitä sekä kyselykielten laajentamista dokumenttien käsittelyyn soveltuvilla primitiiveillä. Tampereen yliopistossa kehitetyssä RDOOM-tietomallissa oliot kuvataan olio-orientoituneesti, kun taas niiden väliset assosiaatiot esitetään arvo-orientoituneesti. RDOOM mahdollistaa mm. deduktiivisten olio- ja assosiaatiotyyppien määrittelyn tietokantaan eksplisiittisesti määriteltyjen olio- ja assosiaatiotyyppien sekä niihin liittyvän ilmentymätason tiedon ja dokumenttien perusteella.

Tutkielman esimerkkijärjestelmässä käytetty tiedon looginen organisointitapa muistuttaa läheisesti RDOOM-tietomallia, jonka deduktiivisia piirteitä ei ole otettu huomioon. RDOOM:in assosiaatioita vastaavat siinä välittömät semanttiset yhteydet ja assosiaatiotyyppejä semanttiset yhteystyypit. Jokaiselle semanttiselle yhteystyypille on määritelty luonnollisella kielellä esitettävä semanttinen tulkinta, jota sovelletaan yhteystyyppiin kuuluviin välittömiin semanttisiin yhteyksiin. Välillinen semanttinen yhteys on olioista ja niitä yhdistävistä välittömistä semanttisista yhteyksistä muodostuva kahden olion välinen navigaatiopolku. Myös välillisellä semanttisella yhteydellä on semanttinen tulkinta, joka koostuu sen välittömien osayhteyksien tulkinnoista.

Tutkielmassa on ehdotettu NGIS-järjestelmien kyselykielten laajentamista semanttisten yhteyksien selvittämiseen liittyvällä piirteellä. Tavanomaiset ekstensionaaliset tietokantakyselyt edellyttävät, että käyttäjä tuntee kokonaisuudessaan olioiden välisen navigaatiopolun, kun taas tutkielman kyselykielessä tämä polku (so. semanttinen yhteys) saadaan kyselyn tuloksena. Tuloksena saatava semanttinen yhteys edustaa toisaalta tietokannan ekstensionaaliselle tasolle kuuluvaa informaatiota (välittömien semanttisten yhteyksien attribuuttien arvot ja niihin osallistuvat oliot) ja toisaalta intensionaalista informaatiota (välittömien osayhteyksien tyypit).

Tutkielman kyselykielellä voidaan saada vastaukset esimerkiksi kysymyksiin: ”Miten kaksi oliota voivat semanttisesti liittyä tosiinsa?” tai ”Mitkä oliot voidaan semanttisesti liittää annettuun olioon?”. Periaatteessa deduktiivisten kyselykielten ilmaisuvoima riittää tällaisten kyselyiden tekemiseen. Kyselynmuodostus näillä kielillä on kuitenkin selvästi vaativampi tehtävä kuin mihin

tavallisen loppukäyttäjän taidot riittävät. Tutkielman kyselykieli sisältää valmiita primitiivejä, joilla käyttäjä voi korkealla abstraktiotasolla sekä selvittää semanttisia yhteyksiä että määritellä suodattavia rajoituksia kyselyn vastauksena tuotettaville yhteyksille.

RDOOM-tietomallin perusprimitiivit sisältävät viimeisen laajennuksen jälkeen myös rakenteiset tekstidokumentit. Ne voivat liittyä paitsi tietokannan intensionaaliseen tasoon (esimerkiksi oliotyyppeihin) myös sen ekstensionaaliseen tasoon (esimerkiksi olioihin). Tämän mukaisesti myös tietokannan kyselykieleen on integroitu dokumenttien käsittelyyn tarkoitettuja primitiivejä, joilla annetaan dokumenttien rakenteeseen tai tekstisisältöön kohdistuvia ehtoja. Integroinnin ansiosta ei tarvita erillisiä hakujärjestelmiä tietokantaan tallennetulle tiedolle ja siihen liittyvälle vapaamuotoiselle tekstitiedolle. Usein samassa sovellusalueessa tarvitaan sekä tieto-orientoitunutta että teksti-orientoitunutta käsittelyä, jolloin niiden piirteiden integrointi on loppukäyttäjälle tarjottavassa kyselykielessä välttämätöntä.

Tutkielman kyselykieli tukee olioihin liittyvien vapaamuotoisten XML-dokumenttien käsittelyä. Vapaamuotoisten dokumenttien rakenteelle ei luvussa 5 esitettyjä poikkeuksia lukuun ottamatta ole asetettu rajoitteita. Kyselykieli soveltuu sekä olioviittauksien sisältävien kokonaisten dokumenttien että spesifeimpien viittauksen sisältävien osadokumenttien selvittämiseen. Erityisesti tieto-orientoituneiden XML-dokumenttien kohdalla jälkimmäinen piirre on hyödyllinen, sillä spesifeintä olioon viittaavaa osadokumenttia vastaava XML-polku voidaan usein tulkita olion roolimäärittelyksi dokumentissa.

Yhdessä semanttisten yhteyksien selvittämiseen soveltuvien primitiivien kanssa dokumenttien käsittelyyn tarkoitetut primitiivit mahdollistavat tavallisen loppukäyttäjän kannalta uudenlaisen informaation johtamisen tietokannasta. Semanttiseen yhteyteen osallistuva olio voidaan tunnistaa (yksikäsitteisesti tai ei) sen tyypin ja attribuuttien arvojen lisäksi myös siihen viittauksen sisältävien dokumenttien ja osadokumenttien perusteella. Kyselyn tulos voi sisältää paitsi tuntemattomia semanttisia yhteyksiä ja olioita myös niihin liitettävissä olevia tuntemattomia dokumentteja tai osadokumentteja.

Kaiken kaikkiaan semanttisen yhteyden käsitteen hyödynnettävyys riippuu sovellusalueesta ja tietokannan tietosisällöstä - esimerkiksi siinä olevien semanttisten yhteyksien lukumäärästä. Potentiaalisia sovellusalueita ovat esimerkiksi tutkiva journalismi, henkilöiden ja/tai muiden objektien välisten suhteiden kartoitus rikostutkinnallisissa yhteyksissä, intressiristiriitojen ja jääviystilanteiden selvittäminen, ilmiöiden välisten syy- ja seuraussuhteiden löytäminen sekä kaupunkien välisten tie-, lento-, yms. vaihtoehtoisten yhteyksien etsiminen yksinkertaisessa reittitietokannassa. Semanttisten yhteyksien selvittämisen suurimpia ongelmia ovat yhteyksien potentiaalinen suuri määrä ja niiden keskinäisen relevanssin arviointi. Tutkielman kyselykielellä käyttäjä voi itse ra-

joittaa hyväksyttävien vastauksien määrää ja laatua. Sen sijaan kyselykieli ei nykyisessä muodossaan tarjoa mitään yhteyksien automaattisen relevanssin arviointiin soveltuvaa menetelmää. Myös nykyisiä primitiivejä monipuolisempien ja tehokkaammin suodattavien ehtojen kehittäminen voi olla mahdollista sallimalla kielessä esimerkiksi universaalikvantifiointi tai yhteydessä esiintyviin oliotyyppeihin kohdistuvat rajoitukset. Vapaamuotoisten XML-dokumenttien käsittelyn osalta tutkielman kyselykieli mahdollistaa vain suppeat rakenteeseen, tekstisisältöön ja viitattuihin olioihin kohdistuvat ehdot, jotka tuskin ovat riittäviä käytännön sovelluksissa. Tällaisenaankin kyselykieli tarjoaa silti hyvän pohjan mahdolliselle jatkokehitykselle.

Viiteluettelo

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

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

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

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

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

[Chamberlin et al., 2000] Don Chamberlin, Jonathan Robie and Daniela Florescu, Quilt: An XML query language for heterogeneous data sources. *Lecture Notes in Computer Science* **1997**, (2001), 1 – 25.

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

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

[Fuhr and Großjohann, 2001] Norbert Fuhr and Kai Großjohann, XIRQL: A query language for information retrieval in XML documents. In: *Proc. of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 172 – 180.

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

[Hillebrand and Kanellakis, 1994] Gerd G. Hillebrand and Paris C. Kanellakis, Functional database query languages as typed lambda calculi of fixed or-

der. In: *Proc. of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 222 – 231.

[Jagadish et al., 2002] H. V. Jagadish, Shurug Al-Khalifa, Adriane Chapman, Laks V. S. Lakshmanan, Andrew Nierman, Stelios Paparizos, Jignesh M. Patel, Divesh Srivastava, NuweeWiwatwattana, Yuqing Wu and Cong Yu, Jagadish et al., 2002: A native XML database. *The International Journal on Very Large Data Bases* **11**, (Dec. 2002), 274 – 291.

[ISO 8601, 2000] Representation of dates and times. *International Standard* **ISO 8601 : 2000**, 2000.

[ISO/IEC 14977, 1996] Extended BNF. *International Standard* **ISO/IEC 14977 : 1996(E)**, 1996.

[Järvelin ja Kekäläinen, 2002] Kalervo Järvelin ja Jaana Kekäläinen, Tiedonhaun menetelmät/ -opintoaineisto. Internetix-verkkoportaali, 2000. Saatavilla: http://www.internetix.fi/opinnot/opintojaksot/0viestinta/\ informaatiotutkimus/po4/ [2.5.2004]

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

[Kay, 2003] Michael H. Kay, XML five years on: a review of the achievements so far and the challenges ahead. In: *Proc. of the 2003 ACM Symposium on Document Engineering*, 29 – 31.

[Khalifa et al., 2003] Shurug Al-Khalifa, Cong Yu and H. V. Jagadish, Querying structured text in an XML database. In: *Proc. of the 2003 ACM SIGMOD International Conference on Management of Data*, 4 – 15.

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

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

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

[Kumpulainen, 2003] Teemu Kumpulainen, XML-dokumenttien deklaratiivinen kyselykieli/ –pro gradu-tutkielma. Tampereen yliopisto, Tietojenkäsittelytieteiden laitos, 2003.

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

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

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

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

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

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

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

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

[Niemi et al., 2004] Timo Niemi, Marko Junkkari and Kalervo Järvelin, Query language approach for next generation information systems. University of Tampere, Department of Computer Sciences, Report **A-2004-5**, 2004.

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

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

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

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

[Salminen and Tompa, 2001] Airi Salminen and Frank Wm. Tompa, Requirements for XML document database systems. In: *Proc. of the 2001 ACM Symposium on Document Engineering*, 85 – 94.

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

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

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

[XML, 2004] François Yergeau, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen and Eve Maler, Extensible Markup Language (XML) 1.0 (Third Edition). *W3C Recommendation*, (Feb. 2004). Available as: http://www.w3.org/TR/2004/REC-xml-20040204/ [5.3.2004].

[XML Namespaces, 1999] Tim Bray, Dave Hollander and Andrew Layman, Namespaces in XML. *W3C Recommendation*, (Jan. 1999). Available as: http://www.w3.org/TR/1999/REC-xml-names-19990114/ [21.3.2004].

[XML Schema, 2001] David C. Fallside, XML Schema part 0: primer. *W3C Recommendation*, (May. 2001). Available as: http://www.w3.org/TR/xmlschema-0/ [5.3.2004].

[XPath, 1999] James Clark and Steve DeRose, XML Path Language (XPath) 1.0. *W3C Recommendation*, (Nov. 1999). Available as : http://www.www.org/TR/xpath [5.3.2004].

[XQuery, 2003] Scott Boag, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie and Jérôme Siméon, XQuery 1.0: An XML query language, *W3C Working Draft*, (Nov. 2003). Available as : http://www.w3.org/TR/xquery/ [5.3.2004]

[XSLT, 2003] Michael Kay. XSL Transformations (XSLT) 2.0. *W3C Working Draft*, (Nov. 2003). Available as: http://www.w3.org/TR/xslt20/ [5.3.2004].

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

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

Kyselykielen syntaksi, BNF-notaatio

```
query ::= result_part [" where " condition_part] ".";
result_part ::= value_reference {", " value_reference};
condition_part ::= condition {", " condition};

value_reference ::= object_reference | semantic_connection_reference |
    document_reference | attribute_reference |primitive_literal;

object_reference ::= variable | literal_object_reference;
literal_object_reference ::= typename " " key_part;
typename ::= constant_string;
key_part ::= literal | "[" literal {"," literal} "]";

semantic_connection_reference ::= variable;

document_reference ::= variable | literal_document_reference;
literal_document_reference ::= filename subpath;
filename ::= variable | constant_string | string;
subpath ::= "/" path_element [subpath] | "//" path_element [subpath];
path_element ::= (variable | constant_string | string) ["[" integer "]"];

attribute_reference ::= (object_reference | semantic_connection_reference) ":"
    attribute_name | document_reference ":content";
attribute_name ::= constant_string;

primitive_literal ::= string | integer | decimal_number | date;

condition ::= negation | type_declaration | comparison | primitive1 |
    primitive2 | primitive3 | primitive4 | primitive5 | primitive6 | primitive7 |
    primitive8;

negation ::= "neg " condition;
type_declaration ::= typename " "
    (object_reference | semantic_connection_reference);
comparison ::= value_reference " " (">" | "=" | "<" | "><") " " value_reference;
primitive1 ::= "sc_between(" object_reference ", " object_reference ", "
    semantic_connection_reference ")";
```

```
primitive2 ::= "sc_via([" object_reference_list "], "
    semantic_connection_reference ")";.
primitive3 ::= "sc_not_via([" object_reference_list "], "
    semantic_connection_reference ")";
primitive4 ::= "sc_using([" semantic_connection_type_list "], "
    semantic_connection_reference ")";
primitive5 ::= "sc_not_using([" semantic_connection_type_list "], "
    semantic_connection_reference ")";
primitive6 ::= "sc_max_length(" integer ", " semantic_connection_reference ")";
primitive7 ::= "refers(" document_reference ", " object_reference ")";
primitive8 ::= "contains(" document_reference ",  " string ")";.

object_reference_list ::= object_reference {", " object_reference};
semantic_connection_type_list ::= semantic_connection_type {", "
    semantic_connection_type};
semantic_connection_type ::= constant_string;

constant_string ::= lower_alpha [alphanum_string_part];
variable ::= ("_" | upper_alpha) [alphanum_string_part];
date ::= integer "-" integer "-" integer;
string ::= """ string_part """;
string_part ::= ("_" | lower_alpha | upper_alpha | number | special_symbol)
    [string_part];
alphanum_string_part ::= ("_" | lower_alpha | upper_alpha | number)
    [alphanum_string_part];
decimal_number ::= integer "." integer;
integer ::= number {number};
lower_alpha ::=
    ("a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "
    s" | "t" | "u" | "v" | "w" | "x" | "y" | "z");
upper_alpha ::=
    ("A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q
    " | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z");
number ::= ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9");
special_symbol ::=
" " | "!" | """ | "#" | "$" | "%" | "&" | "(" | ")" | "*" | "+" | "," | "-
    " | "." | "/" | ":" | ";" | "<" | "=" | ">" | "?" | "@" | "[" | "\" | "]" | "^" | "{" | "|" | "}";
```

dokumentti bos1.xml

```
<bill_of_sale>
  <date>2001-04-30</date>
  <seller>
    <ref:enterprise name="Cassius Automotives">Cassius
      Automotives</ref:enterprise>
  </seller>
  <buyer>
    <ref:person ssn="130845-772E">Allan Selsor</ref:person>
  </buyer>
  <vehicle>
    <ref:car register="USD-999">USD-999</ref:car>
  </vehicle>
  <ordometer_reading>0</ordometer_reading>
  <selling_price>65000 euros</selling_price>
  <warranties/>
</bill_of_sale>
```

dokumentti bos2.xml

```
<bill_of_sale>
  <date>1998-01-13</date>
  <seller>
    <ref:enterprise name="Cassius Automotives">Cassius
      Automotives</ref:enterprise>
  </seller>
  <buyer>
    <ref:person ssn="150372-152V">Calvin Stryker</ref:person>
  </buyer>
  <vehicle>
    <ref:motorcycle register="SU-942">SU-942</ref:motorcycle>
  </vehicle>
  <ordometer_reading>0</ordometer_reading>
  <selling_price>7500 euros</selling_price>
  <special_conditions/>
</bill_of_sale>
```

dokumentti bos3.xml

```
<bill_of_sale>
  <date>1995-04-12</date>
  <seller>
    <ref:enterprise name="Cassius Automotives">Cassius
      Automotives</ref:enterprise>
  </seller>
  <buyer>
    <ref:person ssn="152260-140G">Codey
      McCurley</ref:person>
  </buyer>
  <vehicle>
    <ref:car register="ECC-333">ECC-333</ref:car>
  </vehicle>
  <ordometer_reading>0</ordometer_reading>
  <selling_price>22000 euros</selling_price>
  <special_conditions/>
</bill_of_sale>
```

dokumentti bos4.xml

```
<bill_of_sale>
    I
    <ref:person ssn="151251-215Q">Dale
      Copeland</ref:person>
     in consideration of 12000 euros sell to
    <ref:person ssn="030750-761E">Erica
      Goodheart</ref:person>
     the vehicle
    <ref:car register="VIA-212">lada VIA-212</ref:car>
     on 1982-12-21. I confirm that the ordometer reading
      14000 km of the vehicle is correct. The vehicle is sold
      in "as is" condition. I warrant that I have full authority
      to sell and transfer said vehicle.
</bill_of_sale>
```

dokumentti bos5.xml

```
<bill_of_sale>
  I
  <ref:person ssn="030750-761E">Erica
    Goodheart</ref:person>
  in consideration of 10000 euros sell to
  <ref:person ssn="021050-778D">David
    Stryker</ref:person>
  the vehicle
  <ref:car register="VIA-212">lada VIA-212</ref:car>
  on 1990-01-15. I confirm that the ordometer reading
    15000 km of the vehicle is correct. The vehicle is sold
    in "as is" condition. I warrant that I have full authority
    to sell and transfer said vehicle.
</bill_of_sale>
```

dokumentti bos6.xml

```
<bill_of_sale>
  The Seller
  <seller>
    <ref:person ssn="130845-772E">Allan
      Selsor</ref:person>
  </seller>
  , for payment of
  <price>9500 euros</price>
  by the Buyer
  <buyer>
    <ref:person ssn="130347-372S">Alvin
      Baker</ref:person>
  </buyer>
  , has sold the vehicle
  <sold_vehicle>
    <ref:car register="HUS-412">saab HUS-412</ref:car>
  </sold_vehicle>
  to Buyer on
  <date>2003-05-20</date>
  . The ordometer reading of the vehicle at the moment of
    the sale is
  <ordometer_reading>20000 km</ordometer_reading>
  . The Seller certifies that the Vehicle's odometer has not
    been altered. The vehicle is sold "as is".
</bill_of_sale>
```

dokumentti bos7.xml

```
<bill_of_sale>
  The Seller
  <seller>
    <ref:person ssn="171166-333F">Annabelle
      Castaldo</ref:person>
  </seller>
  , for payment of
  <price>9500 euros</price>
  by the Buyer
  <buyer>
    <ref:person ssn="151251-215Q">Dale
      Copeland</ref:person>
  </buyer>
  , has sold the vehicle
  <sold_vehicle>
    <ref:car register="ACA-454">ACA-454</ref:car>
  </sold_vehicle>
  to Buyer on
  <date>2000-11-15</date>
  . The ordometer reading of the vehicle at the moment of
    the sale is
  <ordometer_reading>50000 km</ordometer_reading>
  . The Seller certifies that the Vehicle's odometer has not
    been altered. The vehicle is sold "as is".
</bill_of_sale>
```

dokumentti dr1.xml

```
<damage_report>
  <date_of_accident>2002-01-03</date_of_accident>
  <time>16:30</time>
  <accident_location>Greenbridge</accident_location>
  <vehicle1>
    <driver>
      <ref:person ssn="130845-772E">Allan
      Selsor</ref:person>
    </driver>
    <vehicle>
      <ref:car register="USD-999">USD-999</ref:car>
    </vehicle>
    <injured_persons/>
    <estimated_damages>700</estimated_damages>
    <description_of_damages>The front frame of the car
      got smashed. </description_of_damages>
  </vehicle1>
  <vehicle2>
    <driver>
      <ref:person ssn="040560-212F">Lisa
      Cassius</ref:person>
    </driver>
    <vehicle>
      <ref:car register="VIA212">VIA212</ref:car>
    </vehicle>
    <injured_persons>
      <injured_person>
        <person>
          <ref:person ssn="040560-212F">Lisa
      Cassius</ref:person>
        </person>
        <description_of_injuries>minor leg and head
      injuries.</description_of_injuries>
      </injured_person>
    </injured_persons>
    <estimated_damages>9000</estimated_damages>
    <description_of_damages>not
      driveable</description_of_damages>
  </vehicle2>
  <details_of_accident>I was trying to do a U-turn and I
      didn't notice Ms Cassius's car which was coming from
      another direction.</details_of_accident>
</damage_report>
```

dokumentti dr2.xml

```
<damage_report>
  <date_of_accident>1999-02-28</date_of_accident>
  <time>10:00 PM</time>
  <location>Oakdale</location>
  <vehicle1>
    <driver>
      <ref:person ssn="150372-152V">Calvin
      Stryker</ref:person>
    </driver>
    <injured_persons>
      <injured_person>
        <ref:person ssn="150372-152V">Calvin
      Stryker</ref:person>
        <description_of_injuries>I broke my left
      arm</description_of_injuries>
      </injured_person>
    </injured_persons>
    <vehicle>
      <ref:motorcycle register="SU-942">SU-
      942</ref:motorcycle>
    </vehicle>
    <estimated_damages>500</estimated_damages>
    <description_of_damages>the front wheel broke
      off</description_of_damages>
  </vehicle1>
  <vehicle2>
    <driver>
      <ref:person ssn="151251-215Q">Dale
      Copeland</ref:person>
    </driver>
    <injured_persons/>
    <vehicle>
      <ref:car register="ACA-454">ACA-454</ref:car>
    </vehicle>
    <estimated_damages>0</estimated_damages>
    <description_of_damages/>
  </vehicle2>
  <details_of_accident>I lost the control of my motorcycle
      and collided with Mr. Copeland's car due to the icy
      conditions.</details_of_accident>
</damage_report>
```

dokumentti dr3.xml

```
<damage_report>
  <type>motor vehicle collision</type>
  <details_of_accident>
    <date>1991-06-13</date>
    <time>8:15 AM</time>
    <place>Scottsfield</place>
    <cause_and_details>I hit the parked truck while
      backing the car.</cause_and_details>
    <reported_to_police>yes</reported_to_police>
  </details_of_accident>
  <vehicle>
    <vehicle>
      <ref:car register="BGD-718">BGD-718</ref:car>
    </vehicle>
    <driver>
      <ref:person ssn="311182-021A">Mallory
      Baltes</ref:person>
    </driver>
    <driver_was_sober>yes</driver_was_sober>
    <estimated_damage>200</estimated_damage>
    <damage_description>few
      scratches</damage_description>
  </vehicle>
  <other_vehicles>
    <vehicle>
      <vehicle>
        <ref:truck register="UVL-161">UVL-161</ref:truck>
      </vehicle>
      <driver/>
      <driver_was_sober/>
      <estimated_damage>50</estimated_damage>
      <damage_description>the rear lights got
      shattered</damage_description>
    </vehicle>
  </other_vehicles>
  <injured_persons>
    <injured_person>
      <person/>
      <description_of_injuries/>
    </injured_person>
  </injured_persons>
</damage_report>
```

dokumentti dr4.xml

```
<damage_report>
  <type>motor vehicle collision</type>
  <details_of_accident>
    <date>2003-10-10</date>
    <time>3:00 PM</time>
    <place>Kingsbury</place>
    <cause_and_details>I ran red light and collided with
      Mrs. Castaldo's car.</cause_and_details>
    <reported_to_police>yes</reported_to_police>
  </details_of_accident>
  <vehicle>
    <vehicle>
      <ref:car register="HUS-412">HUS-412</ref:car>
    </vehicle>
    <driver>
      <ref:person ssn="130347-372S">Alvin
      Baker</ref:person>
    </driver>
    <driver_was_sober>yes</driver_was_sober>
    <estimated_damage>1200</estimated_damage>
    <damage_description>several deep
      dents</damage_description>
  </vehicle>
  <other_vehicles>
    <vehicle>
      <vehicle>
        <ref:car register="UTA-211">UTA-211</ref:car>
      </vehicle>
      <driver>
        <ref:person ssn="171166-333F">Annabelle
      Castaldo</ref:person>
      </driver>
      <driver_was_sober>yes</driver_was_sober>
      <estimated_damage>900</estimated_damage>
      <damage_description>a dent and two deep
      scratches</damage_description>
    </vehicle>
  </other_vehicles>
  <injured_persons>
    <injured_person>
      <person/>
      <description_of_injuries/>
    </injured_person>
  </injured_persons>
</damage_report>
```

dokumentti dr5.xml

```
<damage_report>
  <type>motor vehicle collision</type>
  <details_of_accident>
    <date>2001-09-26</date>
    <time>1:00 AM</time>
    <place>East Abbey</place>
    <cause_and_details>The bus in front of me stopped at
      the red lights. I drove such a speed that I didn't
      manage to stop my car early
      enough.</cause_and_details>
    <reported_to_police>yes</reported_to_police>
  </details_of_accident>
  <vehicle>
    <vehicle>
      <ref:car register="CFU-462">CFU-462</ref:car>
    </vehicle>
    <driver>
      <ref:person ssn="021050-778D">David
      Stryker</ref:person>
    </driver>
    <driver_was_sober>yes</driver_was_sober>
    <estimated_damage>2000</estimated_damage>
    <damage_description>the front of the car was
      crushed</damage_description>
  </vehicle>
  <other_vehicles>
    <vehicle>
      <vehicle>
        <ref:bus register="OTI-541">OTI-541</ref:bus>
      </vehicle>
      <driver>
        <ref:person ssn="152260-140G">Codey
      McCurley</ref:person>
      </driver>
      <driver_was_sober>yes</driver_was_sober>
      <estimated_damage>100</estimated_damage>
      <damage_description>the back bumper got hung
      up</damage_description>
    </vehicle>
  </other_vehicles>
  <injured_persons>
    <injured_person>
      <person/>
      <description_of_injuries/>
    </injured_person>
  </injured_persons>
</damage_report>
```

dokumentti lic1.xml

```
<life_insurance>
  <insurant>
    <ref:person ssn="171166-333F">Annabelle
      Castaldo</ref:person>
  </insurant>
  <beneficiary>
    <ref:person ssn="060685-517X">Maricela
      Castaldo</ref:person>
  </beneficiary>
  <insurance_company>
    <ref:enterprise name="Lifetime Insurance
      Company">Lifetime Insurance
      Company</ref:enterprise>
  </insurance_company>
  <face_amount>200000</face_amount>
  <effective_date>1985-07-01</effective_date>
  <duration>30 years</duration>
  <insurance_payment>1800</insurance_payment>
  <special_policy_conditions/>
  <commencement>1985-07-01</commencement>
</life_insurance>
```

dokumentti lic2.xml

```
<life_insurance>
  <insurant>
    <ref:person ssn="030750-761E">Erica
      Goodheart</ref:person>
  </insurant>
  <beneficiary>
    <ref:person ssn="151251-215Q">Dale
      Copeland</ref:person>
  </beneficiary>
  <insurance_company>
    <ref:enterprise name="Lifetime Insurance
      Company">Lifetime Insurance
      Company</ref:enterprise>
  </insurance_company>
  <face_amount>100000</face_amount>
  <effective_date>1978-06-17</effective_date>
  <duration>entire life</duration>
  <insurance_payment>1500</insurance_payment>
  <special_policy_conditions/>
  <commencement>1978-06-17</commencement>
</life_insurance>
```

dokumentti lic3.xml

```
<life_insurance>
  <insurance_company>
    <ref:enterprise name="Prominent Insurance
      Company">Prominent Insurance
      Company</ref:enterprise>
  </insurance_company>
  <insurant>
    <ref:person ssn="152260-140G">Codey
      McCurley</ref:person>
  </insurant>
  <policy_no>12411</policy_no>
  <insurance_type>optional life insurance
      C2</insurance_type>
  <effective_period>
    <effective_date>1997-11-13</effective_date>
    <duration>entire life</duration>
  </effective_period>
  <beneficiary>
    <ref:person ssn="121272-459W">Ann
      McCurley</ref:person>
  </beneficiary>
  <insurance_payment>1000</insurance_payment>
  <face_amount>15000</face_amount>
  <date_of_agreement>1997-11-13</date_of_agreement>
</life_insurance>
```

dokumentti lic4.xml

```
<life_insurance>
  <insurance_company>
    <ref:enterprise name="Prominent Insurance
      Company">Prominent Insurance
      Company</ref:enterprise>
  </insurance_company>
  <insurant>
    <ref:person ssn="021050-778D">David
      Stryker</ref:person>
  </insurant>
  <policy_no>12398</policy_no>
  <insurance_type>optional life insurance
      C1</insurance_type>
  <effective_period>
    <effective_date>1980-01-23</effective_date>
    <duration>entire life</duration>
  </effective_period>
  <beneficiary>
    <ref:person ssn="150762-115S">Paula
      Campbell</ref:person>
  </beneficiary>
  <insurance_payment>1100</insurance_payment>
  <face_amount>100000</face_amount>
  <date_of_agreement>1980-01-23</date_of_agreement>
</life_insurance>
```

dokumentti mvic1.xml

```
<motor_vehicle_insurance>
  <insurance_company>
    <ref:enterprise name="Prominent Insurance
      Company">Prominent Insurance
      Company</ref:enterprise>
  </insurance_company>
  <insurant>
    <ref:person ssn="130845-772E">Allan
      Selsor</ref:person>
  </insurant>
  <policy_no>998</policy_no>
  <insurance_type>comprehensive</insurance_type>
  <vehicle>
    <ref:car register="USD-999">USD-999</ref:car>
  </vehicle>
  <effective_date>2001-05-01</effective_date>
  <insurance_payment>1400</insurance_payment>
  <deductible>10%</deductible>
  <date_of_agreement>2001-05-01</date_of_agreement>
</motor_vehicle_insurance>
```

dokumentti mvic2.xml

```
<motor_vehicle_insurance>
  <insurance_company>
    <ref:enterprise name="Prominent Insurance
      Company">Prominent Insurance
      Company</ref:enterprise>
  </insurance_company>
  <insurant>
    <ref:person ssn="030750-761E">Erica
      Goodheart</ref:person>
  </insurant>
  <policy_no>1056</policy_no>
  <insurance_type>comprehensive</insurance_type>
  <vehicle>
    <ref:car register="UGF-336">UGF-336</ref:car>
  </vehicle>
  <effective_date>1998-07-12</effective_date>
  <insurance_payment>750</insurance_payment>
  <deductible>40%</deductible>
  <date_of_agreement>1998-07-12</date_of_agreement>
</motor_vehicle_insurance>
```

dokumentti mvic3.xml

```
<motor_vehicle_insurance>
  <insurance_company>
    <ref:enterprise name="Tellurian Insurance
      Company">Tellurian Insurance
      Company</ref:enterprise>
  </insurance_company>
  <insurance_type>motor vehicle insurance,
      comprehensive</insurance_type>
  <insurant>
    <ref:person ssn="171166-333F">Annabelle
      Castaldo</ref:person>
  </insurant>
  <insured_item>
    <ref:car register="UTA-211">UTA-211</ref:car>
  </insured_item>
  <effective_date>2000-10-30</effective_date>
  <insurance_payment>1000</insurance_payment>
  <deductible>20%</deductible>
  <special_policy_conditions/>
  <commencement>2000-10-30</commencement>
</motor_vehicle_insurance>
```

dokumentti mvic4.xml

```
<motor_vehicle_insurance>
  <insurance_company>
    <ref:enterprise name="Tellurian Insurance
      Company">Tellurian Insurance
      Company</ref:enterprise>
  </insurance_company>
  <insurance_type>motor vehicle insurance,
      comprehensive</insurance_type>
  <insurant>
    <ref:enterprise name="Westfield Transit">Westfield
      Transit</ref:enterprise>
  </insurant>
  <insured_item>
    <ref:bus register="OTI-541">OTI-541</ref:bus>
  </insured_item>
  <effective_date>1990-05-05</effective_date>
  <insurance_payment>1900</insurance_payment>
  <deductible>40%</deductible>
  <special_policy_conditions/>
  <commencement>1990-05-05</commencement>
</motor_vehicle_insurance>
```

dokumentti mvic5.xml

```
<motor_vehicle_insurance>
  <insurance_company>
    <ref:enterprise name="Prominent Insurance
      Company">Prominent Insurance
      Company</ref:enterprise>
  </insurance_company>
  <insurant>
    <ref:enterprise name="Everyway
      Transportation">Everyway
      Transportation</ref:enterprise>
  </insurant>
  <policy_no>980</policy_no>
  <insurance_type>comprehensive</insurance_type>
  <vehicle>
    <ref:truck register="UVL-161">UVL-161</ref:truck>
  </vehicle>
  <effective_date>1986-07-22</effective_date>
  <insurance_payment>1700</insurance_payment>
  <deductible>50%</deductible>
  <date_of_agreement>1986-07-22</date_of_agreement>
</motor_vehicle_insurance>
```

dokumentti mvic6.xml

```
<motor_vehicle_insurance>
  <insurance_company>
    <ref:enterprise name="Prominent Insurance
      Company">Prominent Insurance
      Company</ref:enterprise>
  </insurance_company>
  <insurant>
    <ref:enterprise name="Everyway
      Transportation">Everyway
      Transportation</ref:enterprise>
  </insurant>
  <policy_no>1006</policy_no>
  <insurance_type>comprehensive</insurance_type>
  <vehicle>
    <ref:truck register="DEF-777">DEF-777</ref:truck>
  </vehicle>
  <effective_date>1992-06-02</effective_date>
  <insurance_payment>2000</insurance_payment>
  <deductible>50%</deductible>
  <date_of_agreement>1992-06-02</date_of_agreement>
</motor_vehicle_insurance>
```

dokumentti mvic7.xml

```
<motor_vehicle_insurance>
  <insurance_company>
    <ref:enterprise name="Tellurian Insurance
      Company">Tellurian Insurance
      Company</ref:enterprise>
  </insurance_company>
  <insurance_type>motor vehicle insurance,
      comprehensive</insurance_type>
  <insurant>
    <ref:person ssn="021050-778D">David
      Stryker</ref:person>
  </insurant>
  <insured_item>
    <ref:car register="CFU-462">CFU-462</ref:car>
  </insured_item>
  <effective_date>1998-12-05</effective_date>
  <insurance_payment>800</insurance_payment>
  <deductible>15%</deductible>
  <special_policy_conditions/>
  <commencement>1998-12-05</commencement>
</motor_vehicle_insurance>
```