Mikko Malinen

LIIKETOIMINTAPROSESSIN KUVAAMINEN TRANSAKTIONA ERI TRANSAKTIOMALLIEN AVULLA

Tietojärjestelmätieteen kandidaatintutkielma

20. helmikuuta 2009

JYVÄSKYLÄN YLIOPISTO

TIETOJENKÄSITTELYTIETEIDEN LAITOS

TIIVISTELMÄ

Malinen, Mikko
Tietojärjestelmätieteen kandidaatintutkielma / Mikko Malinen
Jyväskylä: Jyväskylän yliopisto, 2009, 41 s.

Kandidaatintutkielma

Yrityksen liiketoimintaprosessia tarkasteltaessa voidaan siinä huomata olevan transaktiolle ominaisia piirteitä. Tässä tutkielmassa liiketoimintaprosessilla tarkoitetaan sitä tehtävien ketjua, joka yrityksen täytyy suorittaa saavuttaakseen tietyn liiketoiminnallisen lopputuloksen. Vastaavasti transaktiolla tarkoitetaan toimintojen ketjua, joka suoritetaan tietyn lopputuloksen saavuttamiseksi esimerkiksi tietokannassa. Transaktion ACID -ominaisuudet (jakamattomuus, ristiriidattomuus, erillisyys ja pysyvyys) nousevat keskeisiksi tekijöiksi käsitteiden yhteyttä arvioitaessa. Liiketoimintaprosessia voidaan pitää ACID -ominaisuuksia noudattavana transaktiona, jolloin liiketoimintaprosessi saa käyttöönsä ACID -ominaisuuksien tarjoamat hyödyt muun muassa prosessin jakamattomuuteen ja erillisyyteen liittyen. Liiketoimintaympäristö asettaa kuitenkin haasteensa ACID -ominaisuuksien tiukalle noudattamiselle.

Tässä tutkielmassa tarkastellaan kehittyneiden transaktiomallien tuomia helpotuksia liiketoimintaprosessin kuvaamiseen transaktiona. Eri transaktiomallien soveltamista on tärkeä arvioida, sillä liiketoimintaprosessin toteuttaminen soveltumattoman transaktiomallin avulla saattaa aiheuttaa liiketoimintaprosessin tehottomuutta tai jopa estää liiketoiminnan suorittamisen. Tutkielmassa keskitytään arvioimaan kehittyneiden transaktiomallien soveltuvuutta liiketoimintaprosessien kuvaamiseen ACID -ominaisuuksien näkökulmasta ja tarkastellaan kevennettyjen ACID -ominaisuuksien tarjoamia etuja perinteiseen transaktiomalliin verrattuna.

Tutkielma on toteutettu kirjallisuuskatsauksena aiheesta jo kirjoitettua tieteellistä aineistoa lähteenä käyttäen. Keskeisenä tuloksena tutkielmassa todetaan transaktionäkökulman toimimattomuus liiketoimintaympäristössä johtuen tietokantamaisen lähestymistavan soveltumattomuudesta liiketoimintaympäristöön. Edes kehittyneillä transaktiomalleilla ei voida täysin kuvata kaikkia mahdollisia liiketoimintaan liittyviä rakenteita ja poikkeustilanteita tehokkaasti. Lisäksi tutkielmassa on lyhyesti selvitetty kuinka transaktioihin perustuvaa liiketoimintaprosessien kuvaamista on siirrytty korvaaman työkulkujärjestelmillä.

AVAINSANAT: transaktio, ACID -ominaisuudet, liiketoimintaprosessi, kehittyneet transaktiomallit, sisäkkäiset transaktiot, saga-transaktiot

ABSTRACT

Malinen, Mikko
Bachelor's thesis in Information Systems Science / Mikko Malinen
Jyväskylä: University of Jyväskylä, 2009, 41 p.

Bachelor's thesis

Business processes can be seen to have transactional features. In this thesis, a business process refers to a chain of tasks which must be accomplished to achieve a certain business outcome. Respectively, a transaction is defined as a sequence of distinct actions which is executed in order to reach a predefined outcome for example in a database. The ACID properties (atomicity, consistency, isolation and durability) have an important role when the similarity of these two concepts is evaluated. In some situations, business processes can be considered as transactions following ACID properties. Atomicity and isolation are known to be important properties to business processes which can benefit from these properties because of their transactional nature. Although ACID properties seem to fit well with the needs of business processes, strict ACIDity cannot be implemented in a business environment.

This thesis concentrates on evaluating the benefits gained from extended transaction models in describing a business process as a transaction. Different transaction models need to be evaluated because modeling a business process with an unsuitable transaction model can cause efficiency problems or even prevent a business process from functioning properly. In this thesis, the focus will be on evaluating the applicability of extended transaction models to business process modeling from the ACID perspective. In addition, the benefits of using relaxed ACID properties instead of full ACID properties will be discussed. Evaluations are based on a literature research exploiting literature from both database and business process management fields.

The most important finding of this thesis is that even extended transaction models cannot solve all the problems that are related to modeling a business process as a transaction. The database-centric approach of transaction models cannot be applied to a business environment: business processes are not as simple as transactions in terms of their internal structure and they normally encounter many exceptions in their execution, which cannot be handled by using transactions. Nowadays, resulting from the problems presented above, business processes are often managed by workflow management systems. This approach will be briefly presented in the third chapter of this thesis.

KEYWORDS: transaction, ACID properties, business process, extended transaction models, nested transactions, saga transactions

SISÄLTÖ

1 JOHDANTO

Transaktion (*transaction*) määritellään muodostuvan ketjusta erillisiä toimintoja (*sequence of actions*), jotka suoritetaan yhdessä tavoitellun lopputuloksen aikaansaamiseksi (esim. Garcia-Molina, 1983, 187; Gray, 1981, 145; Wang, Vonk, Kratz & Grefen, 2008, 237). Transaktioiden tarkoitus on mahdollistaa toimintoketjun käsittely yhtenä jakamattomana operaationa, jolloin ketjun yksittäisten toimien vaikutuksista ja suorituksesta ei tarvitse kantaa huolta sovellustasolla. Transaktiosta käytetään usein suomennosta *tapahtuma*, joka kuvastaa hyvin transaktion jakamattomuutta: vaikka transaktio koostuukin toimintojen ketjusta, ei tapahtumaa tule tarkastella sen osien kautta vaan niiden yhteisvaikutuksen kautta. Transaktioiden avulla voidaan myös huolehtia osittain suoritettujen operaatioiden yksinkertaisesta kumoamisesta. Transaktioiden käyttö tietokannoissa on ollut merkittävä askel luotettavan tietojenkäsittelyn varmistamisessa: Wang ym. (2008, 237) toteavat, että erityisesti monen yhtäaikaisen käyttäjän tilanteessa ACID -ominaisuuksia noudattavalla transaktiolla on korvaamaton rooli tietojenkäsittelyn virheettömyyden takaamisessa. Lisäksi on huomattu, että transaktioilla on erittäin suuri rooli myös liikemaailman sovelluksissa (Chen & Buchs, 2006). Ihannetilanteessa yritykset voivat mallintaa liiketoimintaprosessinsa transaktioina, sillä liiketoimintaprosessi muistuttaa pohjimmiltaan hyvin paljon transaktiota.

Tässä tutkielmassa keskitytään tarkastelemaan transaktioiden soveltuvuutta liiketoimintaympäristöön, jossa luotettava tietojenkäsittely on erittäin tärkeässä asemassa. Transaktioihin liittyvässä tutkimuksessa on todettu, että transaktioiden ACID -ominaisuuksia voidaan soveltaa lähtökohtaisesti myös liiketoimintaprosessien kuvaamisessa, vaikkakin niiden tiukka noudattaminen ei ole mahdollista liiketoimintaympäristössä samoin kuin tietokantaympäristössä (esim. Greenfield, Fekete, Jang & Kuo, 2003). Transaktioiden ACID -ominaisuudet sopisivat hyvin esimerkiksi liiketoimintaprosessin jakamattomuuden sekä

yhtäaikaisuuden hallintaan, jolloin liiketoimintaprosesseja voitaisiin suorittaa monien yhtäaikaisten käyttäjien kesken ilman häiriöitä (ks. Bennett ym., 2000, 333–334). Liiketoiminnan luonne on kuitenkin asettanut haasteensa transaktiolähestymistavan hyödyntämiselle varsinkin monimutkaisten ja pitkien liiketoimintaprosessien tapauksessa. Tästä johtuen perinteiselle ACID -ominaisuuksia noudattavalle transaktiolle on täytynyt etsiä korvaajaa, joka soveltuisi ACID -ominaisuuksien keventämisen myötä paremmin liiketoimintakontekstiin (Greenfield ym., 2003). Chen ja Buchs (2006) toteavat soveltumattomuuden johtuvan siitä, että liiketoimintakontekstin transaktiot eivät ole enää luonteeltaan lyhytkestoisia ja yksinkertaisia kuten tietokantaympäristössä. Tästä johtuen myöskään liiketoimintaprosessi ei tällöin voi olla ACID -ominaisuuksia noudattavan transaktion kaltainen.

Tässä tutkielmassa aihetta rajataan keskittymällä tiukasti sidottuihin (*tightly coupled*) liiketoimintaprosesseihin, jossa prosessin kaikki vaiheet ovat ennalta suunniteltuja ja toteutukseen kiinnitettyjä. Tarkastelun ulkopuolelle jätetään liiketoimintaprosessien käytännöntoteutus transaktioina, jonka sijasta keskitytään pohtimaan transaktiolähestymistavan toimivuutta ACID -ominaisuuksien näkökulmasta. Tutkielmassa vertaillaan eri transaktiomallien soveltuvuutta liiketoimintaprosessien kuvaamiseen teoriatasolla, mutta ei oteta kantaa tai vertailla yksittäisten liiketoimintaprosessien käytännön toteutuksia transaktiomallien avulla. Tarkastelussa pyritään tuomaan ilmi eri transaktiomallien perusperiaatteet sekä kuvaamaan niiden toteuttamat ACID -ominaisuudet painottaen niitä ACID -ominaisuuksia, joista on pyritty luopumaan tai joita on kevennetty.

Tutkielma on kirjallisuuskatsaus aihetta käsittelevään teoriaan, jossa merkittävimpinä lähteinä toimivat sekä Wangin ym. (2008) transaktioita käsittelevä julkaisu että Bennettin ym. (2000) ja Papazogloun (2003) transaktiomaisia liiketoimintaprosesseja käsittelevät julkaisut. Tutkielman tavoitteena on selvittää, kuinka transaktiot soveltuvat liiketoimintaprosessien kuvaamiseen ja kuinka

soveltuvuus on muuttunut kehittyneiden transaktiomallien synnyn myötä. Tutkielmassa on kaksi keskeistä tutkimusongelmaa, jotka ovat:

1. Mitä ovat liiketoimintaprosessit ja kuinka niitä voidaan kuvata transaktioiden avulla?

2. Mitä ovat kehittyneet transaktiomallit ja ratkaisevatko ne liiketoimintaprosessien transaktioilla kuvaamiseen liittyvät ongelmat?

Vastauksia tutkimusongelmiin haetaan vertailemalla perinteisten ACID -ominaisuuksia noudattavien transaktioiden sekä kahden kehittyneen transaktiomallin soveltumista liiketoimintaprosessien kuvaamiseen ja esittelemällä niiden ja perinteisen ACID -ominaisuuksia noudattavan transaktion eroja. Tutkimustuloksista voidaan hyötyä liiketoimintaprosesseja suunniteltaessa silloin, kun suunnitellaan liiketoimintaprosessien käytännön toteutusta tai silloin kun muokataan jo olemassa olevia liiketoimintaprosesseja paremmin liiketoimintaympäristöön soveltuviksi.

Tutkielma etenee siten, että luvussa kaksi tarkastellaan transaktiota ja siihen liitettäviä ACID -ominaisuuksia sekä määritellään mitä ovat pitkät transaktiot ja niiden syntyyn vaikuttavat tekijät. Luvussa kolme käsitellään liiketoimintaprosessin rakennetta sekä esitellään transaktion suhde liiketoimintaprosessiin. Tämän lisäksi luvussa tarkastellaan ACID -ominaisuuksia noudattavan transaktion hyödyntämiseen liittyviä ongelmia liiketoimintaprosessin kuvaamisessa. Luvussa neljä haetaan ratkaisuja luvussa kolme esitettyihin ongelmiin kahden kehittyneen transaktiomallin avulla sekä esitellään pintapuolisesti transaktioajattelusta poikkeava tapa mallintaa liiketoimintaprosesseja. Tutkielma päättyy yhteenvetoon luvussa viisi.

2 YKSITTÄISISTÄ OPERAATIOISTA TRANSAKTIOIHIN

Tässä luvussa käsitellään lähemmin transaktion teoriaa ja sen ominaispiirteitä ACID -ominaisuuksien muodossa. Luvun lopuksi käsitellään transaktion suoritusajan pidentymiseen vaikuttavia tekijöitä sekä pidentymisen seurauksia. Tämän luvun on tarkoitus taustoittaa seuraavaa lukua, jossa keskitytään liiketoimintaprosessin transaktiomaiseen kuvaamiseen.

2.1 Transaktiot

Transaktio on käsitteenä hyvin vanha, ja sitä voidaan soveltaa hyvin useassa eri kontekstissa (Wang ym., 2008). Transaktiolla tarkoitetaan Grayn (1981, 144) sekä Wangin ym. (2008, 237) mukaan jonkin asian tilan muuttumista toiseksi. Kontekstin vaihtuessa tilan muuttuminen voidaan ymmärtää hyvin monella eri tavalla. Wangin ym. (2008, 237) mukaan esimerkiksi ostajan ja myyjän välinen sopimus siitä, että kauppa tapahtuu sillä tavoin kuin on sovittu, on eräänlainen transaktio. Transaktion toteuduttua omistajuussuhteiden tila on muuttunut: ostaja on vaihtanut rahansa haluamaansa tuotteeseen ja tuotteen omistajuus on siirtynyt myyjältä ostajalle. Myöhemmin tässä luvussa keskitytään tarkastelemaan transaktiota tietokantaympäristössä. Tietokannan tilat muodostuvat tietokannan tietueiden arvoista, jolloin transaktion aiheuttama siirtymä tilasta toiseen tapahtuu tietueiden arvoja muuttamalla (Gray, 1981, 145).

Kun transaktiota yritetään määritellä konkreettisemmin, voidaan huomata, että erilaisia hieman toisistaan poikkeavia määritelmiä on olemassa runsaasti. Breitbart, Garcia-Molina ja Silberschatz (1992, 24) sekä Korth, Levy ja Silberschatz (1990, 97) tarkentavat Garcia-Molinan (1983), Grayn (1981) sekä Wangin ym. (2008) esittämää määritelmää kuvaamalla transaktion operaatioketjuksi (*chain of operations*), joka syntyy jonkin ohjelman suorituksen lopputuloksena. Lynch, Merrit, Weihl ja Fekete (1993, 13) eivät puolestaan pidä transaktiota vain suora-

viivaisten operaatioiden ketjuna vaan pikemminkin ohjelmana, joka kykenee mukailemaan suoritustaan valintoihin ja edellisten vaiheiden tulosten perusteella. Tämä transaktion määritelmä heijastelee transaktioiden myöhempien kehitysvaiheiden piirteitä, ja poikkeaa siksi perinteisestä transaktion määritelmästä. Edellä esitetyille määritelmille on yhteistä transaktion hahmottaminen operaatioketjuksi, joka tulee suorittaa yhtenä operaationa.

Tässä tutkielmassa keskitytään käsittelemään transaktiota Garcia-Molinan (1983) esittämän määritelmän mukaisesti. Hän nimittää transaktiota käyttäjän tekemäksi pyynnöksi (*user request*), joka esitetään suoritettavaksi kohdejärjestelmässä. Pyyntö voi koostua yhdestä tai useammasta eri toimesta, mutta pyynnön toteuttaminen vaatii kuitenkin kaikkien toimien toteuttamista tietyssä järjestyksessä. (Garcia-Molina, 1983, 187.) Tämä näkökulma noudattaa yleistä transaktion määritelmää, sekä palvelee hyvin myös tietokantatransaktion tarkastelua käytännönläheisyytensä vuoksi.

Tietotekniikassa transaktiot kehitettiin alun perin tietokantaympäristöön suorittamaan pieniä tehtäviä, joita esiintyi esimerkiksi rahankäsittelyyn liittyvissä sovelluksissa (Wang ym., 2008, 235). Tietokantaympäristössä yksittäinen tietokanta toimii eräänlaisena tietovarastona ja resurssina sovelluksille, jotka pyrkivät muuttamaan tietokannan tilaa halutulla tavalla. Garcia-Molinan (1983) määritelmän mukaisesti transaktiot voidaan nähdä sovellusten luomina pyyntöinä, jotka tulee toteuttaa tietokannassa. Esimerkkinä yksinkertaisesta tietokantatransaktiosta Wang ym. (2008) esittelevät artikkelissaan tilisiirtotransaktion, jossa asiakkaan tililtä siirretään tietty summa rahaa myyjän tilille. Esimerkissä suoritettava transaktio vähentää asiakkaan tililtä tietyn summan rahaa, jonka jälkeen samansuuruinen rahasumma lisätään myyjän tilille – näin tietokannan tila on päivitetty vastaamaan todellista tilaa, joka seuraa tehdystä tilisiirrosta. Transaktio koostuu esimerkissä kahdesta erillisestä operaatiosta: asiakkaan saldon vähentämisestä ja myyjän saldon kasvattamisesta, joiden molempien suorituksen

täytyy onnistua, jotta tilisiirtotransaktio voidaan katsoa onnistuneeksi. (Wang ym., 2008, 241–242.)

2.2 ACID -ominaisuudet

Edellä kuvatusta ketjumaisesta ja vaiheittaisesta rakenteesta johtuen transaktioihin liitetään neljä ominaisuutta, joiden tulee toteutua tiedonkäsittelyn luotettavuuden ja oikeellisuuden säilyttämiseksi (Grefen, Pernici & Sánchez, 1999, 5; Wang ym., 2008, 239). Näitä ominaisuuksia kutsutaan transaktioiden ACID -ominaisuuksiksi, joihin kuuluvat jakamattomuus (*atomicity*), ristiriidattomuus (*consistency*), erillisyys (*isolation*) sekä pysyvyys (*durability*) (Breitbart ym., 1992, 24; Ferreira, 2002, 3; Haerder & Reuter, 1983, 289–290; Wang ym., 2008, 239–240). ACID -ominaisuuksia noudattavan transaktion ajatus esiteltiin jo 1960-luvulla, mutta *ACID* -termi vakiintui käyttöön vasta 1980-luvulla (Gray & Reuter, 1993, Chessell ym., 2002, 744 mukaan).

Jakamattomuus. Jakamattomuuden mukaan transaktiota tulee käsitellä yhtenäisenä operaatiosarjana, joka joko suoritetaan täysin tai jätetään kokonaan suorittamatta (Breitbart ym., 1992, 24; Ferreira, 2002, 3; Garcia-Molina, 1983, 187; Greenfield ym., 2003; Haerder & Reuter, 1983, 289; Laiho, 2005). Jakamattomalla transaktiolla ei myöskään voida nähdä olevan välitiloja ulkopuolelta tarkasteltuna. Sisäisen rakenteen puutteesta johtuen perinteistä ACID -ominaisuuksia noudattavaa transaktiota kutsutaankin Wangin ym. (2008) mukaan litteäksi transaktioksi (*flat transaction*). (Wang ym., 2008, 239–241.)

Ristiriidattomuus. Ristiriidattomuudella tarkoitetaan sitä, ettei transaktio voi muuttaa kohdejärjestelmän tilaa ristiriitaiseksi vaan tehty muutos tapahtuu aina järjestelmän asettamien ehtojen rajoissa (Haerder & Reuter, 1983, 289; Laiho, 2005; Wang ym., 2008, 240). Tietokannassa ristiriidattomuus näkyy siten, että tietokannan kaikki tietueet noudattavat tietokannan kaikkia eheysrajoitteita (*integrity constraints*) (Barghouti & Kaiser, 1991, 273; Eswaran, Gray, Lorie & Trai-

ger, 1976, 624; Garcia-Molina, 1983, 187). Tietokannan eheysrajoitteet voivat esimerkiksi määrätä, että lentolipunvarauksessa kukin istumapaikka voidaan varata vain yhdelle henkilölle kerrallaan (Barghouti & Kaiser, 1991, 273). Breitbart ym. (1992, 24) sekä Ferreira (2002, 3) painottavat omissa ristiriidattomuuden määritelmässään, että ristiriidattomuus säilyy ainoastaan, mikäli transaktio suoritetaan kokonaan: mahdolliset jakamattomuuden puutteesta johtuvat osittaiset suoritukset eivät puolestaan takaa järjestelmän ristiriidattomuutta. Näin jakamattomuus liittyykin läheisesti ristiriidattomuuteen.

Erillisyys. Erillisyydellä tarkoitetaan transaktion suorittamista muista mahdollista samanaikaisista transaktioista riippumattomana (Breitbart ym., 1992, 24; Ferreira, 2002, 3; Haerder & Reuter, 1983, 289; Laiho, 2005; Wang ym., 2008, 240). Samanaikaiset transaktiot suoritetaan ikään kuin ne olisivat peräkkäisiä transaktioita, jolloin transaktiot eivät aiheuta vääristyneitä tuloksia, joita voisi aiheutua samanaikaisesta resurssien käsittelystä (Ferreira, 2002, 3; Wang ym., 2008, 240). Transaktiokäsittelyn sanotaan tällöin olevan sarjallistuvaa (*serializable*): samanaikaiset transaktiot asetetaan suorituslistaan (*schedule*) siten, että niiden aiheuttama lopputulos tietokannassa vastaa tilannetta, jossa transaktiot olisi suoritettu peräkkäin. Sarjallistuvuus ei kuitenkaan tarkoita sitä, että transaktiot suoritettaisiin todellisuudessa peräkkäin. Esimerkiksi useat lukuoperaatiot voidaan suorittaa ennen kirjoitusoperaatioita mikäli kirjoitukset käsittelevät kukin omaa resurssiaan. (Laiho, 2005.)

Pysyvyys. Pysyvyydellä varmistetaan transaktion tekemien muutosten säilyvyys virhetilanteessa: tehtyjen muutosten tulee olla sekä pysyviä riippumatta mahdollisista transaktiota seuraavista järjestelmävirheistä että voimassa heti transaktion suorituksen jälkeen (Breitbart ym., 1992, 24; Ferreira, 2002, 3; Greenfield ym. 2003; Haerder & Reuter, 1983, 289; Wang ym. 2008, 240).

ACID -ominaisuuksista jakamattomuus, erillisyys ja pysyvyys ovat edellytyksiä tietokannan pitämiseen ristiriidattomana virhetilanteista ja transaktioiden sa-

manaikaisesta suorituksesta huolimatta. Tällöin ristiriidattomuus on pikemminkin tietokannan ominaisuus: tietokanta valvoo ristiriidattomuutta suorittaessaan transaktioita ACID -ominaisuuksiin perustuen. (Greenfield ym., 2003.)

ACID -ominaisuuksien ansiosta transaktiota voidaan hyödyntää monien yhtäaikaisten käyttäjien tilanteessa kiinnittämättä erityistä huomiota yhtäaikaisuuden aiheuttamiin ongelmiin. Edellä kuvatulla tavalla jakamattomuus ja erillisyys yhdessä varmistavat sen, että yhtäaikaiset transaktiot eivät häiriinny edellä esitetyllä tavalla toistensa aiheuttamista tietokannan keskeneräisistä ja ristiriitaisista tiloista ja suorita näin omia operaatioitansa ristiriitaisiin arvoihin perustuen. Pysyvyys puolestaan varmistaa että tietokantaan tehdyt muutokset ovat pysyviä myös virheiden sattuessa transaktion suorituksen jälkeen.

Ennen ACID -ominaisuuksien tunnistamista sovelluslogiikan tehtävänä oli huolehtia tietokannan ristiriidattomuudesta ja virhetilanteiden hallinnasta. Nykyisissä tietokantaympäristöissä ACID -ominaisuuksien noudattamisesta vastaa tapahtumankäsittelyjärjestelmä eli TKJ (*transaction processing system, TPS*). (Wang ym., 2008, 241.) ACID -ominaisuuksien noudattaminen TKJ:n toimesta helpottaa ohjelmien suunnittelijoiden työtä, sillä heidän ei enää tarvitse ottaa kantaa tiedonkäsittelyn toimivuuteen TKJ:n pitäessä huolta siitä, että toiminnot suoritetaan jakamattomana operaationa ilman muista samanaikaisista operaatioista aiheutuvia vaikutuksia (Chessell ym., 2002, 744; Greenfield ym., 2003). TKJ huolehtii ACID -ominaisuuksien noudattamisesta käyttämällä hyväkseen tietokannan tietueiden lukituksia sekä pitämällä lokia tietokannan tilan muutoksista. Lukitukset mahdollistavat yhtäaikaisten resurssien käytön estämisen, kun taas lokien avulla voidaan tarvittaessa kumota tai palauttaa ennalleen transaktion tekemät muutokset. (Greenfield ym., 2003; Wang ym., 2008, 241.) Lukituksilla transaktioiden tarvitsemat resurssit varataan tietokannasta transaktion käyttöön suorituksen ajaksi, jonka jälkeen lukitukset puretaan. Näin muut mahdolliset resurssien tarvitsijat joutuvat odottamaan lukitusten purkamista ennen kuin pääsevät itse muokkaamaan tarvitsemiaan resursseja. (Greenfield

ym., 2003; O'Neil, 1986, 406.) Greenfield ym. (2003) huomauttavat lukitusten käyttämisen olevan edellytyksenä myös lokien tallentamiselle, sillä tietueet tulee pystyä lukitsemaan lokin kirjoituksen ajaksi mahdollisilta muutoksilta.

Transaktioiden yhtäaikaista suoritusta voidaan hallita lukitusten sijasta myös optimistisuuteen perustuvilla menetelmillä (*optimistic non-locking mechanisms*). Tällöin transaktioiden tarvitsemia resursseja ei lukita tietokannasta transaktioiden käyttöön, vaan transaktiota suoritettaessa valvotaan, että ennen tietokantaan tehtäviä muutoksia tietokannan tilassa ei ole tapahtunut muutoksia lähtötilanteeseen verrattuna. Mikäli tietueiden arvoja on muutettu transaktion suorituksen aikana, aloitetaan transaktio alusta. Nämä menetelmät soveltuvat hyvin ympäristöihin, joissa yhtäaikaiset transaktiot ovat harvinaisia ja transaktioiden kestot ovat lyhyitä, mutta optimistisuudesta aiheutuvat transaktioiden uudelleensuoritukset aiheuttavat ongelmaa erityisesti pitkäkestoisten transaktioiden tapauksessa. (Barghouti & Kaiser, 1991, 275–280.) Käyttökontekstista riippuu, mitä samanaikaisuudenhallintamenetelmää on kannattavinta soveltaa.

2.3 Pitkät transaktiot

Puhuttaessa pitkistä transaktioista (*long-running transaction* tai vaihtoehtoisesti *long lived transaction*) viitataan pituudella transaktion suorituksen keston pituuteen (Barghouti & Kaiser, 1991, 282; Fischer & Majumdar, 2007, 54; Garcia-Molina & Salem, 1987, 249). Pitkässä transaktiossa operaatioketju on merkittävästi pitkäkestoisempi kuin perinteisessä tietokantatransaktiossa, joka Haerderin ja Reuterin (1983, 291) mukaan voidaan suorittaa jopa alle sekunnissa. Barghoutin ja Kaiserin (1991, 282) sekä Garcia-Molinan ja Salemin (1987, 249) mukaan pitkäkestoiset transaktiot voivat kestää tunneista jopa viikkoihin. Tarkasteltaessa transaktiota on kuitenkin muistettava, että transaktion operaatioketjun pituus ei välttämättä tarkoita, että transaktio olisi kestoltaan pitkä, sillä oikeanlaiset suoritusolosuhteet voivat mahdollistaa myös monivaiheisen transaktion nopean suorittamisen.

Transaktiot voivat muuttua pitkäkestoisiksi, mikäli niiden yksittäiset operaatiot monimutkaistuvat tai niiden operaatioketju pidentyy. Garcia-Molinan ja Salemin (1987, 249) mukaan transaktioiden pidentyneeseen kestoon vaikuttavat muun muassa useiden tietokantaobjektien käyttö sekä pitkäkestoiset laskentaoperaatiot. Lisäksi automatisoidut operaatioketjut voivat vaatia ihmisen aktiivista osallistumista ja myös yhteistoimintaa manuaalisten syötteiden (*input*) muodossa jatkaakseen suoritustaan, jolloin transaktion kesto voi vaihdella ihmisen läsnäolosta johtuen hyvinkin paljon. (Garcia-Molina & Salem, 1987, 249; Greenfield ym., 2003.) Varsinkin yhteistoiminnan mahdollistumisen voidaan nähdä aiheutuneen tekniikan kehittymisestä, jonka ansiosta nykyiset transaktiot eivät enää välttämättä ole pelkkiä tietokannan sisällä suoritettavia ennalta määriteltyjä tapahtumia vaan suorituksen aikana tarkentuvia operaatioketjuja. On myös mahdollista, että transaktion pituuteen vaikuttavat edellä mainitut tekijät yhdessä (Garcia-Molina & Salem, 1987, 249).

Transaktioista voi myös tulla pitkäkestoisia, mikäli muut transaktiot hidastavat niiden suoritusta. Tällöin transaktioiden keston pidentymisen syynä voidaan pitää ACID -ominaisuuksista aiheutuvaa käsittelytapaa (Garcia-Molina & Salem, 1987). Koska transaktion tulee olla jakamaton ja suoriutua muita transaktioita huomioimatta, käsitellään pitkääkin transaktiota yhtenä kokonaisuutena, jonka suoritusta varten varataan transaktion tarvitsemat resurssit koko transaktion suorituksen ajaksi. Näin pitkät transaktiot ovat haitaksi muille suoritettaville transaktioille, joiden suoritus estyy pitkän transaktion lukitusten vuoksi. (Garcia-Molina & Salem, 1987, 249.)

Pitkien transaktioiden sanotaankin aiheuttavan useimmissa tapauksissa transaktionkäsittelylle suorituskykyongelmia, mikäli transaktioita käsitellään ACID -ominaisuuksia tiukasti noudattaen (Garcia-Molina & Salem, 1987, 249). Lukitusten käyttö transaktioiden ollessa pitkäkestoisia voi lisäksi saada aikaan transaktioiden ajautumisen niin sanottuun lukkiumatilanteeseen (*deadlock*), jossa transaktioiden suorituksen jatkaminen ei ole mahdollista toisen transaktion

varattua seuraavan vaiheen resurssit omaan käyttöönsä. Ongelmaksi lukkiumatilanteessa nousee resurssien lukitusten vastavuoroisuus: resursseja varaava transaktio ei pääse purkamaan lukituksia ja jatkamaan seuraavaan vaiheeseen, sillä vastaavasti toisella transaktiolla on hallussaan lukitus resursseihin, joita se tarvitsee omaan suoritukseensa. Näin kaikki transaktiot pysähtyvät odottamaan resurssien vapautumista, mitä ei koskaan tapahdu. (Wolfson ja Yannakakis, 1985, 106.) Garcia-Molina ja Salem (1987, 249) toteavat Grayhyn, Homaniin, Obermarckiin ja Korthiin (1981) viitaten lukkiumatilanteiden todennäköisyyden kasvavan neljännessä potenssissa transaktion pituuden kasvaessa.

Transaktioiden käsittely ACID -ominaisuuksiin perustuen on edellä kuvattuun viitaten kannattavaa vain, jos transaktioiden suoritus ei kestä pitkään – Chenin ja Buchsin (2006) mukaan ACID -ominaisuuksia noudattavan transaktion perusolettamus onkin transaktion lyhyt suoritusaika. Tällöin lukkiumatilanteiden todennäköisyys pysyy pienenä ja transaktiot eivät pidennä toistensa suoritusaikaa. Tästä syystä ACID -ominaisuuksien noudattamisen on todettu olevan yksinkertaisinta ja suositeltavinta tilanteessa, jossa transaktio on lyhytkestoinen ja suoritetaan keskitetysti yhden tietokannan alaisena (mm. Wang ym., 2008, 242). Tämä on nähtävissä tarkasteltaessa transaktionkäsittelyn tehokkuutta: Hrastnikin ja Winiwarterin (2007, 856) mukaan ACID -ominaisuuksia noudattavan järjestelmän tehokkuus on verrannollinen transaktion pituuteen tehokkuuden laskiessa transaktion pituuden kasvaessa. Perinteisten ACID -ominaisuuksia noudattavien transaktioiden turvallisuus ja yksinkertainen toimintaperiaate ovat kuitenkin saaneet aikaan sen, että niitä käytetään yhä edelleen erittäin paljon, vaikka ne eivät sovellukaan pitkiin transaktioihin ja monimutkaisiin ympäristöihin (Wang ym. 2008, 242).

3 LIIKETOIMINTAPROSESSIN KUVAAMINEN TRANSAKTIONA

Tässä luvussa tarkastellaan perinteisen transaktiomallin soveltamista liiketoimintakontekstissa. Aluksi määritellään mitä ovat liiketoimintaprosessit sekä millaisia piirteitä niihin liittyy. Luvun lopussa siirrytään tarkastelemaan perinteisten ACID -ominaisuuksia noudattavien transaktioiden soveltuvuutta liiketoimintaprosessien kuvaamiseen.

3.1 Liiketoimintaprosessit

Davenportin ja Shortin (1990, 12) sekä Papazogloun (2003, 49) määritelmien mukaisesti liiketoimintaprosessi (*business process*) on loogisesti toisiinsa liittyvien tehtävien (*tasks*) muodostama sarja, jonka tarkoituksena on saavuttaa tietty liiketoiminnallinen lopputulos. Schuldt, Alonso, Beeri ja Schek (2002, 64) täydentävät määritelmää toteamalla liiketoimintaprosessin kuvastavan vaiheita yhden tai useamman yrityksen sisällä, jotka täytyy käydä läpi tietyn liiketoiminnallisen lopputuloksen tai tehtävän saavuttamiseksi. Liiketoimintaprosessi voi näin olla esimerkiksi verkkokirjakaupan tilauksenkäsittelystä huolehtiva prosessi, joka koostuu erilaisista tilauksenkäsittelyyn liittyvistä tehtävistä (Veijalainen, 1999, 213). Tällöin liiketoimintaprosessin tarkoituksena on mahdollistaa asiakkaalle ostosten teko verkkokirjakaupassa. Davenportin ja Shortin (1990, 12–13) sekä Papazogloun (2003, 50) mukaan liiketoimintaprosessit voivat olla organisaatioiden tai organisaatioyksiköiden välisiä, jolloin niiden tehtävät voivat olla hajautettu usealle eri osapuolelle. Bennettin ym. (2000, 331) mukaan monet liiketoimintaprosessit ovat lisäksi luonteeltaan pitkäkestoisia. Tämän voidaan todeta johtuvan osittain sekä prosessien monimutkaisesta rakenteesta että tehtävien hajautuksesta usean osapuolen kesken.

Liiketoimintaprosessi monimutkaistuu ja sen kesto pitenee sitä mukaa kun siihen liitetään uusia vaiheita tai osapuolia. Veijalainen (1999, 213) esittelee artikkelissaan verkkokirjakaupan tilauksenkäsittelyyn liittyvän liiketoimintaprosessin, josta on nähtävissä prosessin kompleksinen rakenne ja pitkä suoritusaika. Tilauksenkäsittely saattaa Veijalaisen (1999) mukaan kestää jopa yli neljä viikkoa, sillä liiketoimintaprosessin voidaan katsoa päättyneen vasta tuotteen palautusajan loputtua tai kaupan perumisen ja takaisinmaksun tapahduttua. Esimerkin mukaisesti prosessi alkaa hetkestä, jolloin asiakas aloittaa kirjatilauksen tekemisen ja päättyy siihen, kun asiakas on sitoutuvasti päättänyt pitää tuotteet. Monimutkaisuus on nähtävissä Veijalaisen (1999) esimerkissä siten, että tilausprosessiin liittyy asiakkaan ja yrityksen lisäksi kolmantena osapuolena maksujärjestelyistä huolehtiva VISA, joka myös osaltaan lisää liiketoimintaprosessin kestoa. Tässä ja monissa muissa tapauksissa liiketoimintaprosessin kompleksinen rakenne ei ole näkyvissä prosessin kaikille osapuolille, vaan monet tehtävistä ovat osapuolten välillä kahdenkeskeisiä. Veijalaisen (1999) esimerkissä asiakas ei suoranaisesti tiedä mitä tehtäviä myyjä suorittaa kahdenkeskeisesti VISA:n kanssa.

3.2 Liiketoimintaprosessi transaktiona

Automatisoitua liiketoimintaprosessia voidaan pitää tarkoin järjestettyjen toimintojen ketjuna (*sequence of activities*), joka tähtää tietyn liiketoimintatehtävän suorittamiseen (Papazoglou, 2003, 49). Tämän määritelmän mukaan liiketoimintaprosessin voidaan ajatella koostuvan yksittäisistä toiminnoista, jotka sisältyvät aiemmin mainittuihin liiketoimintaprosessin tehtäviin. Toiminnot ovat varsinaisia liiketoimintaprosessin suorittavia osia ja ne voivat vaihdella yksinkertaisista viestinvälitystehtävistä monimutkaisiin koordinointitehtäviin liiketoimintaprosessista riippuen. (Papazoglou, 2003, 49–50.) Kun verrataan Papazogloun (2003) määritelmää automatisoidusta liiketoimintaprosessista transaktion määritelmään, voidaan huomata, että käsitteissä on samoja piirteitä. Sekä

transaktio että automatisoitu liiketoimintaprosessi koostuvat samalla tavoin toimintojen tai operaatioiden ketjusta muodostaen jonkun merkityksellisen kokonaisuuden yhdessä suoritettuna. Molemmille on tavoitteellista suorittaa joko koko operaatioketju, tai olla suorittamatta mitään; tämän lisäksi mahdolliset rinnakkaiset operaatioketjut eivät saisi häiritä toistensa suoritusta.

Niissä tilanteissa, joissa liiketoimintaprosessit muistuttavat piirteiltään transaktioita, on alettu puhua niin sanotuista transaktiovaatimuksia noudattavista liiketoimintaprosesseista (*transactional business process*). Ne eroavat tavallisista liiketoimintaprosesseista siten, että niihin ja niiden suorittamiseen voidaan havaita liittyvän transaktiomaisia piirteitä. (Bennett ym., 2000; Chen & Buchs, 2006.) Bennettin ym. (2000) mukaan esimerkiksi vakuutusyhtiön voidaan katsoa tarvitsevan liiketoimintaprosessilleen (vakuutusyhtiö tarjoaa asiakkailleen vakuutuksia) ACID -ominaisuuksien tarjoamia piirteitä seuraavien syiden vuoksi:

Jakamattomuus. Liiketoimintaprosessin tuloksena joko syntyy vakuutuskirja tai ei synny mitään. Mahdollisia liiketoimintaprosessin osittaisia suorituksia ei pidetä hyväksyttävinä, joten niitä ei sallita.

Ristiriidattomuus. Liiketoimintaprosessi voi muuttaa liiketoiminnan tilan vain sallitusta tilasta toiseen.

Erillisyys. Vakuutuskäsittelyn keskeneräiset tilat eivät saa olla näkyvissä muille tarkastelijoille ennen vakuutuksen tietojen lopullista käsittelyä ja hyväksyntää.

Pysyvyys. Vakuutuksen käsittelyn välivaiheiden tulee olla pysyviä vakuutuskirjan valmistumisen jälkeen. (Bennett ym., 2000, 333–334.)

Yllä kuvatut ominaisuudet vastaavat läheisesti perinteisten transaktioiden noudattamia ACID -ominaisuuksia (ks. luku 2.2). Voidaankin siis todeta, että transaktioihin sovellettavat ACID -ominaisuudet ovat ideaalisia hyödynnettäväksi myös muualla kuin tietokantaympäristössä. Liiketoimintaprosessia voidaan

ajatella ACID -ominaisuuksia noudattavana transaktiona silloin, kun edellä kuvattujen ominaisuuksien tulee täyttyä liiketoimintaprosessin suorituksen onnistumiseksi.

Perinteisesti liiketoimintaprosessin on ajateltu koostuvan useista lyhyistä ja löyhästi toisiinsa liittyvistä transaktioista, jotka suoritetaan liiketoimintaprosessin etenemisen mukaisesti (Bennett ym., 2000, 332). Liiketoimintaprosessin kuvaaminen ACID -ominaisuuksia noudattavana transaktiona on tuonut tähän lähestymistapaan helpotusta ACID -ominaisuuksien myötä. Edellä käsitellyssä Bennettin ym. (2000) esimerkissä jakamattomuus mahdollistaa esimerkiksi keskeneräisen vakuutuskäsittelyn täydellisen hävittämisen, mikäli asiakas päättääkin perua vakuutushakemuksensa vakuutuskäsittelyn alettua. Myös erillisyys on tärkeää keskeneräisten tietojen peittämisen kannalta, kun keskeneräisten ja varmentamattomien vaakutuksen tietojen ei haluta häiritsevän muita vakuutuskäsittelijöitä. (Bennett ym., 2000, 331–332.)

Transaktio kokoaa yhteen sekä liiketoimintaprosessin transaktiomaiset että ei-transaktiomaiset osat ja muodostaa näin operaatioiden ketjun tilanteessa, jossa liiketoimintaprosessin katsotaan onnistuvan. Koska liiketoimintaprosessi on yleensä pitkäkestoinen, muodostuu transaktioksi mallinnetusta liiketoimintaprosessista myös pitkäkestoinen transaktio. Perinteisenä transaktiona käsiteltynä liiketoimintaprosessin ei-transaktiomaisia osia ei voida kuitenkaan automatisoida, vaan esimerkiksi kirjeiden lähetykset ja puhelinsoitot on hoidettava manuaalisesti ilman että transaktionhallintajärjestelmä voi valvoa niiden suoritusta.

3.3 ACID -ominaisuudet liiketoimintaympäristössä

Papazogloun (2003, 55) mukaan transaktioihin sovellettavat ACID -ominaisuudet ovat kuitenkin liian rajoittavia noudatettaviksi liiketoimintaympäristössä. Liiketoimintaprosessit ovat harvoin niin suoraviivaisia prosesseja, että niitä

kyettäisiin kuvaamaan täysin suoraviivaisena transaktiona. Butlerin, Hoaren ja Ferreiran (2005) mukaan liiketoimintaprosessin tehtäviä saatetaan joutua koordinoimaan eri hierarkiatasojen ja liiketoimintakumppaneiden kesken, jotta niiden suoritus olisi mahdollista. Tällöin liiketoimintaprosessin kuvaaminen litteänä transaktiona hankaloituu tai on jopa mahdotonta.

Lähtökohtaisesti voidaan todeta, että transaktioina kuvattujen liiketoimintaprosessien pitkä kesto ei sovellu ACID -ominaisuuksia noudattavalle transaktiolle. Kuten luvussa 2.3 todettiin, ACID -ominaisuuksia tulisi hyödyntää vain transaktioiden suoritusajan ollessa lyhyt – liiketoimintaprosessia mallinnettaessa tämä harvoin toteutuu. Koska liiketoimintaprosessista muodostuva transaktio on pitkäkestoinen, ajaudutaan väistämättä luvussa 2.3 esitettyihin ongelmiin.

Pitkä liiketoimintaprosessia kuvaava transaktio lukitsee käyttöönsä tietokantaresursseja koko transaktion suorituksen ajaksi, koska transaktio noudattaa ACID -ominaisuuksia. Tämä puolestaan estää muita samanaikaisia käyttäjiä pääsemästä käsiksi tietokantaresursseihin, jonka lopputuloksena muiden transaktioiden suoritus hidastuu ja transaktiokäsittelyn tehokkuus laskee. Lukituksista voi aiheutua ongelmaa myös tiedonvälitykselle: esimerkiksi varastosaldon päivitys tapahtuu pitkien transaktioiden takia vasta liiketoimintaprosessin valmistuttua, jolloin kirjanpito ja varastosaldon seuranta ei ole mahdollista reaaliajassa (Papazoglou, 2003, 55).

Papazogloun (2003) mukaan lukitusten käyttö saattaa lisäksi estää liiketoimintaprosessille ominaisen ja tärkeän osapuolten yhteistyön, mikäli muilla osapuolilla ei ole mahdollisuutta päästä käsiksi transaktion varaamiin tietokantaresursseihin. Tämä on erityisen haitallista juuri e-liiketoimintasovelluksissa, joille liiketoimintaprosessiin yhteisosallistumisen mahdollistaminen on erityisen tärkeää (Papazoglou, 2003, 55). Esimerkiksi Bennettin ym. (2000, 333–334) esittämässä vakuutusyhtiön liiketoimintaprosessissa vakuutuskirjan tietoihin tulisi tietokannassa päästä käsiksi usean eri tahon toimesta yhtä aikaa. Mikäli

tämä ei olisi mahdollista, hidastuisi vakuutuskäsittely merkittävästi (esim. vakuutustietoja ei voitaisi tarkastella samalla kun niitä esimerkiksi varmennetaan viranomaisten toimesta). Lisäksi liiketoiminnassa lukkiumatilanteiden vaara on erittäin tärkeää huomioida transaktioita suoritettaessa, sillä jotkut liiketoimintaprosessin vaiheet saattavat olla merkittävän pitkäkestoisia aiheuttaen lukkiumatilanteiden riskin eksponentiaalisen kasvun. Lisäksi on tärkeä muistaa, että yksikin liiketoimintaprosessin pitkä vaihe riittää tekemään transaktiosta pitkäkestoisen ACID -ominaisuuksia noudatettaessa.

ACID -ominaisuuksista johtuvien lukitusongelmien lisäksi myös tietoturvaongelmat ovat relevantteja liiketoimintaprosesseja mallinnettaessa. Liiketoiminnassa ei voida käyttää hyväksi paikallisten tietokantojen lukituksia samalla tavoin kuin tietokantatransaktioissa, sillä lukitukset aiheuttavat tietoturvariskin varsinkin pitkien transaktioiden yhteydessä. Yhteistyökumppaneille ei haluta antaa mahdollisuutta vaikuttaa yrityksen omiin tietoresursseihin ja niiden saatavuuteen rajoittavasti, joten paikallisten tietokantaresurssien lukituksia ei pidetä käytännöllisinä tietoturvauhasta johtuen. (Papazoglou, 2003, 55; Greenfield ym., 2003.) Lukituksia kuitenkin tarvitaan mahdollistamaan ACID -ominaisuuksien takaama yhtäaikainen käyttö ja ristiriidattomuus, joten lukituksia ei voida kokonaan unohtaa. Joissain tapauksissa myös aikaleimaukseen perustuvat menetelmät ovat käyttökelpoisia, mutta mikäli niissä ajaudutaan uudelleensuorituksiin virhetilanteiden takia, pidentyy transaktioiden suorittamiseen vaadittava aika yhä lisää (Barghouti & Kaiser, 1991, 277–278). Schuldt ym. (2002, 64) toteavatkin varsinaisen ongelman olevan, kuinka voitaisiin sallia mahdollisimman paljon transaktioiden välistä viestintää ja resurssien yhteiskäyttöä samaan aikaan pitäen huolta tiedonkäsittelyn oikeellisuudesta.

Myöskään liiketoimintaprosessia kuvaavan transaktion kumoaminen ei ole yhtä yksinkertaista kuin perinteisen litteän transaktion kumoaminen. Perinteiset tietokantatransaktiot saatettiin kumota ainoastaan palauttamalla tietokannan vanhat arvot lokien avulla, mutta liiketoimintaprosessia kuvaavaa transaktiota

ei voida kumota vain perumalla tehdyt tehtävät vaan täytyy huomioida tilannekohtaisesti mitä kunkin tehtävän kumoaminen vaatii käytännössä (Butler ym., 2005). Virhetilanteiden hallinnalla on kuitenkin erittäin suuri merkitys liiketoimintaprosesseissa, sillä Greenfield ym. (2003) toteavat Peltziin (2003) viitaten, että lähes 80 % liiketoimintaprosessin suoritukseen kuluvasta ajasta kuluu poikkeusten käsittelyyn. Liiketoimintaprosesseja kuvatessa täytyy myös pystyä ottamaan huomioon, että uusia virhetilanteita saattaa esiintyä myös virhetilanteiden käsittelyvaiheessa (Greenfield ym., 2003).

Usein liiketoimintaprosessin manuaalisista tehtävistä johtuen täydellinen kumoaminen ei kuitenkaan ole toteutettavissa perinteisen transaktiomallin avulla, koska manuaalisiin tapahtumiin ei voida vaikuttaa jälkikäteen. Esimerkiksi vakuutuskäsittelyssä luotu vakuutuskirjaelementti voidaan poistaa tietokannasta pelkästään perumalla transaktio, mutta tehtyjä puhelinsoittoja tai sovittuja asiakastapaamisia ei voida perua näin. Schuldtin ym. (2002, 65) mukaan liiketoimintaprosessissa saattaa esiintyä lisäksi vaiheita, joiden kumoaminen ei ole mahdollista (*no-return point*). Tällöin virheen sattuessa liiketoimintaprosessi pitää saattaa loppuun esiintyneestä virheestä huolimatta yrittämällä uudelleen epäonnistunutta vaihetta tai valitsemalla vaihtoehtoinen liiketoimintaprosessin suorituspolku (Schuldt ym., 2002). Tämänkaltaisia tilanteita ei luonnollisestikaan voida toteuttaa perinteisenä ACID -transaktiona, jossa ainoana kumouskelvottomana pisteenä voidaan nähdä transaktion päättävä operaatio (Schuldt ym., 2002, 65). Esimerkkinä kumoamiskelvottomasta pisteestä voidaan pitää tuotteen (esim. vakuutuskirja) vastaanottamista, sillä mikäli asiakas on maksanut vakuutuksestaan, mutta luottoyhtiö ilmoittaa maksun epäonnistuneen vasta jälkikäteen, täytyy tilanteesta pystyä selviytymään vakuutuskirjaa perumatta.

Edellä kuvatuista syistä johtuen virhetilanteiden hallinnasta tulee erittäin kallista varsinkin pitkien liiketoimintaprosessia kuvaavien transaktioiden tapauksessa etenkin kun tiettyjä transaktion osia ei voida kumota (Ferreira, 2002, 3; Chessell ym., 2002, 744). Tähän liittyen myös Papazoglou (2003, 55) myöntää artikke-

lissaan, että pitkien transaktioiden yhteydessä olisi usein toivottavaa, että transaktioiden tiettyjä osia voitaisiin perua ja suorittaa uudelleen vaikuttamatta itse transaktion onnistumiseen. Yleisesti voidaankin väittää, että perinteinen ACID -ominaisuuksia noudattava transaktio ei sovellu tilanteisiin, jossa esiintyy runsaasti poikkeustilanteita.

4 KEHITTYNEET TRANSAKTIOMALLIT LIIKETOIMINTAPROSESSSIN KUVAAMISESSA

Tässä luvussa tarkastellaan kehittyneiden transaktiomallien soveltumista liiketoimintaprosessien kuvaamiseen. Aluksi määritellään mitä ovat kehittyneet transaktiomallit, jonka jälkeen esitellään kaksi kehittynyttä transaktiomallia ja tarkastellaan niiden soveltuvuutta liiketoimintaprosessien kuvaamiseen. Luvun lopuksi esitetään yhteenveto kehittyneiden transaktiomallien hyväksikäytön onnistumisesta liiketoimintaympäristössä jossa samalla esitetään uusi, transaktiolähestymistavasta poikkeava tapa kuvata liiketoimintaprosesseja.

4.1 Kehittyneet transaktiomallit

Greenfieldin ym. (2003) mukaan on yleisesti tunnustettu, että pitkäkestoiset liiketoimintaprosessit eivät voi edellä kuvatuista syistä noudattaa ACID -ominaisuuksia tarkasti, mutta silti ACID -ominaisuuksien tarjoamalle ristiriidattomuudelle ja toteutuksen helppoudelle on olemassa tarvetta. Tästä syystä on suunniteltu erityisiä jatkettuja transaktiomalleja (*extended transaction models*), joissa ACID -ominaisuuksia on kevennetty siten, että ne soveltuisivat paremmin pitkiin transaktioihin, mutta tarjoaisivat silti tuen virheiden ja yhtäaikaisuuden hallitsemiselle (Greenfield ym., 2003; Wang ym., 2008, 247). Jatketuista transaktiomalleista käytetään myös joissakin lähteissä nimitystä kehittyneet transaktiomallit (*advanced transaction models*, ATMs) (mm. Alonso ym., 1996; Chen & Buchs, 2006; Wang ym., 2008).

Wangin ym., (2008, 247) mukaan ACID -ominaisuuksia keventämällä transaktion sisäistä rakennetta voidaan kuvata kahdella vaihtoehtoisella tavalla. Molemmat tavat perustuvat transaktion jakamiseen pienempiin alitransaktioihin (*subtransaction*) hieman eri tavoin riippuen transaktion semantiikasta. Näin voidaan suorittaa pitkiä ja monimutkaisia transaktioita välttämällä niistä aiheutu-

via ongelmia yhden jakamattoman transaktion tapauksessa. (Wang ym., 2008, 242.) Tämä on merkityksellistä transaktiovaatimuksia noudattavien liiketoimintaprosessien näkökulmasta, sillä näin voidaan välttää ACID -ominaisuuksia noudattavan transaktion hyödyntämisestä aiheutuneita ongelmia.

Pitkien transaktioiden jakaminen alitransaktioihin soveltuu hyvin juuri liiketoimintaprosessin käsittelyyn, sillä liiketoimintaprosessin voidaan Papazogloun (2003, 52) mukaan nähdä muodostuvan useista erillisistä liiketoimintatransaktioista (*business transaction*) alitransaktiorakenteen mukaisesti. Papazoglou (2003) määrittelee liiketoimintatransaktion muutokseksi liiketoiminnan tilassa, jota ohjaa tarkasti määritelty liiketoimintaprosessi. Youngin (2006) esittämä hieman konkreettisempi määritelmä perustuu Yhdysvaltojen sähköisten transaktioiden säädökseen (US Uniform Electronic Transactions Act) vuodelta 1999, jonka mukaan liiketoimintatransaktio voidaan määritellä tapahtumien ketjuksi kahden tai useamman osapuolen välillä, jotka liittyvät jollain tavalla liiketoimintaan tai liiketoimiin. Esimerkkinä liiketoimintatransaktiosta voidaan nähdä tavaran tilaaminen joltain yritykseltä (Papazoglou, 2003, 52). Tällöin tehty tilaus ei kuitenkaan saa olla liiketoimintaprosessin päätarkoitus, vaan tilaus auttaa saavuttamaan jonkun suuremman tavoitteen, olemalla näin selvästi vain liiketoimintaprosessin yksi osa. Liiketoimintatransaktio voi pitää sisällään esimerkiksi tietokantojen päivitystä tai muita toimenpiteitä, jotka saavat aikaan liiketoiminnan tilan muutoksen (Papazoglou, 2003, 52).

Seuraavaksi tarkastellaan, kuinka transaktio voidaan jakaa hierarkkiseksi puurakenteeksi, joka muodostuu alitransaktioista. Sen jälkeen esitellään alitransaktioiden ketjuttaminen. Tarkastelussa käytetään apuna sisäkkäisiä transaktioita sekä saga-transaktioita, joiden perusperiaatteet kuvataan lyhyesti alaluvuissa 4.1.1 ja 4.1.2. Molemmissa malleissa pyritään tarjoamaan pitkille transaktioille mahdollisuus selviytyä virhetilanteista ilman, että koko transaktio joudutaan aloittamaan alusta (Wang ym., 2008, 242). Tutkielmassa on keskitytty juuri näihin kehittyneisiin transaktiomalleihin, sillä Wangin ym. (2008) mukaan niissä

esitellään uudenlainen transaktiorakenne. Muut kehittyneet transaktiomallit lähinnä käyttävät näitä malleja hyväkseen ja jalostavat niitä eteenpäin. Tästä syystä valittuja transaktiomalleja voidaankin pitää vaikutusvaltaisimpina (Wang ym., 2008, 247). Transaktiomallien tarjoamia etuja ja kohtaamia ongelmia liiketoimintaprosessien mallintamista käsiteltäessä käytetään esimerkkinä luvussa 3.2 kuvattua vakuutusyhtiön liiketoimintaprosessia, jossa asiakas hakee yhtiöltä vakuutusta autolleen.

4.1.1 Sisäkkäiset transaktiot

Sisäkkäisellä transaktiolla (*nested transaction*) tarkoitetaan Mossin (1982, Chessellin ym., 2002, 754 mukaan) mukaan transaktiota, joka koostuu alitransaktioiden hierarkiasta. Alitransaktiot muodostavat puumaisen rakenteen, jonka ylimpänä osana eli juurena on ylätason transaktio (*top-level transaction*) kuvastaen ositettavaa transaktiota (Miettinen, 2002, 45). Kaikki juuresta haarautuvat alitransaktiot ovat itse joko litteitä transaktioita tai uusia sisäkkäisiä transaktioita (Barghouti & Kaiser, 1991, 279; Haerder & Rothermel, 1987, 240; Lynch ym., 1993, 15; Miettinen, 2002, 45; Wang ym., 2008, 244). Näin puurakenne voi ulottua hyvinkin laajaksi sisäkkäisten transaktioiden ketjuksi, jossa Wangin ym. (2008) mukaan vasta alimman eli lehtitason alitransaktiot (*leaf level transaction*) ovat sisäkkäisen transaktion varsinaisia suorittavia osia ollen samalla myös litteitä transaktioita. Toisin sanoen ainoastaan lehtitason litteät alitransaktiot voivat sisältää tietokantaoperaatioita. (Miettinen, 2002, 45; Wang ym., 2008, 244). Alitransaktioiden suhteista puhuttaessa käytetään joko nimityksiä esi-isä (*ancestor*) ja jälkeläinen (*descendant*) tai vanhempi (*parent*) ja lapsi (*child*): lapsi-transaktio on vanhempi-transaktiosta haarautuva alitransaktio, joka on itse joko lapsi-transaktio tai uusi vanhempi-transaktio jolla on lapsi-transaktioita (Haerder & Rothermel, 1987, 240; Wang ym., 2008, 244). Wangin ym. (2008, 244) mukaan vanhempi-transaktion roolina on siis toimia koordinaattorina alemman tason lapsi-transaktioille.

Sisäkkäinen transaktio noudattaa kokonaisuutena ACID -ominaisuuksia, josta johtuen muut transaktiot voivat nähdä sisäkkäisestä transaktiosta vain ylätason transaktion (Barghouti & Kaiser, 1991, 279; Miettinen, 2002, 45). Sisäkkäisen transaktion suoritus tapahtuu lapsi-transaktioiden avulla siten, että ylätason transaktion aloituksen jälkeen aloitetaan lehtitason alitransaktioiden suoritus (Wang ym., 2008, 244). Sisäkkäisessä transaktiossa lehtitason transaktiot voidaan suorittaa samanaikaisesti transaktioiden ollessa itsessään jakamattomia ja toisistaan riippumattomia. Tästä aiheutuu myös se, että suorituksessa oleva alitransaktio voi epäonnistua aiheuttamatta ongelmia muille vanhempi-transaktion lapsi-transaktiolle. (Wang ym., 2008, 244.). Vasta kun vanhempi-transaktion kaikki lapsi-transaktiot ovat suorittaneet tehtävänsä onnistuneesti, voi vanhempi-transaktio ilmoittaa suorittaneensa tehtävänsä (Lynch ym., 1993, 16; Wang ym., 2008, 244). Kun vanhempi-transaktiot vähitellen valmistuvat, päästään transaktiohierarkiassa etenemään ylöspäin kohti ylätason transaktiota ja lopulta koko sisäkkäinen transaktio saadaan suoritetuksi, jolloin transaktion muutokset asetetaan voimaan ja tehdään näkyviksi muille transaktioille (Miettinen, 2002, 45; Wang ym., 2008, 244).

Haerderin ja Reuterin (1983, 292) sekä Miettisen (2002, 45) mukaan lehtitason transaktiot noudattavat vain ACI -ominaisuuksia, jolloin alitransaktioiden tekemät muutokset eivät ole alitransaktion pysyvyys -ominaisuuden puutteen vuoksi voimassa vielä alitransaktion valmistuttua, vaan vasta ylätason transaktion valmistuttua. Haerder ja Rothermel (1987, 240) toteavatkin edelliseen viitaten, että alitransaktion onnistuminen riippuu sen oman suorituksen lisäksi myös alitransaktion esi-isien onnistumisesta. Jos ylätason transaktion käynnistyttyä transaktion suoritus katkeaa esimerkiksi laiterikkoon, eivät pysyvyyden puuttumisesta johtuen alitransaktioiden tekemät muutokset jää järjestelmään pysyviksi ennen ylätason transaktion valmistumista. Haerderin ja Rothermelin (1987, 240) mukaan myös ristiriidattomuuden ominaisuudesta voidaan luopua, mikäli tarkoituksena on rakentaa sisäkkäinen transaktio hajautettuun tietokantaympäristöön. Ristiriidattomuuden ehdoton noudattaminen tekisi mahdotto-

maksi esimerkiksi perinteisen tilisiirtotransaktion toteuttamisen sisäkkäisenä transaktiona hajautetussa ympäristössä (Haerder & Rothermel, 1987).

Sisäkkäisten transaktioiden hyödyntäminen tarjoaa mahdollisuuden selvitä transaktion suorituksessa esiintyvistä virhetilanteista. Suorituksessa oleva lapsi-transaktio voi keskeyttää suorituksensa virheen takia, jolloin vanhemman tehtäväksi jää huolehtia mahdollisesta korvaavasta lapsi-transaktiosta tai vaihtoehtoisesti ilmoittaa lapsi-transaktion epäonnistumisesta asettamalla itsensä peruutustilaan (Barghouti & Kaiser, 1991, 279; Lynch ym., 1993, 17; Wang ym., 2008, 244). Virheiden mahdollistaminen on Barghoutin ja Kaiserin (1991, 279) mukaan erittäin tärkeää, koska näin vältetään kokonaisen sisäkkäisen transaktion peruminen, mikäli virheestä on mahdollista palautua uudella yrityksellä tai vaihtoehtoisella transaktiolla. Mikäli vanhempi-transaktio peruutetaan, ei sille kuuluvien lapsi-transaktioiden vaikutuksia aseteta voimaan transaktion ulkopuolelle vaan tehdyt muutokset kumotaan (Chessell ym., 2002, 754; Haerder & Rothermel, 1987, 240). Mikäli peruutus tapahtuu ylemmällä tasolla, käy vastaavalla tavalla sen kaikille alemman tason lapsi-transaktioille (Haerder & Rothermel, 1987, 240).

Sisäkkäisten transaktioiden etuna voidaan pitää niiden kykyä mahdollistaa sisäisten kontrollirakenteiden käyttö transaktioiden suunnittelussa (Haeder & Rothermel, 1987, 239; Lynch ym., 1993, 17). Esimerkiksi rinnakkain suoritettavissa olevat transaktion osat voidaan mallintaa helposti saman vanhempi-transaktion lapsiksi, jolloin osien samanaikainen suoritus toteutuu. Mitä suurempi transaktio on, sitä varmemmin siinä voidaan hyödyntää alitransaktioiden yhtäaikaista suoritusta. Yhtäaikainen suoritus on sisäkkäisissä transaktioissa toteutettu luotettavasti ja turvallisesti, jonka ansiosta transaktion tehokkuus kasvaa transaktion keston lyhetessä. (Haerder & Rothermel, 1987, 239.) Edellä kuvatuista ominaisuuksista voidaan nähdä olevan hyötyä erityisesti liiketoimintaprosessia mallinnettaessa. Vakuutusyhtiön liiketoimintaprosessissa samanaikaisuutta voitaisiin hyödyntää esimerkiksi vakuutustietoja käsiteltäessä

siten, että samalla kun vakuutukseen annettuja tietoja vahvistetaan esimerkiksi viranomaisten toimesta, voidaan vakuutuskirjaa valmistella etukäteen, jotta kaikki on valmiina viranomaisvahvistuksen valmistuttua. Tapahtumat eivät näin ole suoranaisesti sidoksissa toisiinsa, mutta niiden molempien onnistunut suoritus on edellytyksenä sille, että liiketoimintaprosessissa voidaan edetä seuraavaan vaiheeseen. Liiketoimintaprosessia mallinnettaessa sisäkkäisiä transaktioita tulisi hyödyntää varsinkin niissä tilanteissa, joissa useita toisistaan riippumattomia vaiheita voidaan suorittaa yhtä aikaa tai ennakoiden.

Miettisen (2002, 45) mukaan sisäkkäisissä transaktioissa on virhetilanteiden sallimisen ansiosta mahdollista toteuttaa hienorakeisempaa virhetilanteiden käsittelyä kuin perinteisissä litteissä transaktioissa. Tämä on tärkeää myös mallinnettaessa liiketoimintaprosessia, jossa virhetilanteiden suuri määrä aiheuttaisi transaktion jatkuvaa keskeytymistä, jos transaktion virheistä ei kyettäisi palautumaan koko transaktiota peruuttamatta. Esimerkiksi vakuutusyhtiön liiketoimintaprosessia ei ole mielekästä keskeyttää, mikäli asiakas ei ilmoita heti kaikkia tarvittavia tietoja hakiessaan vakuutusta. Kun eri tietojen keräämiset on mallinnettu eri alitransaktioiksi, voidaan puuttuvia tietoja kysyä uudelleen, ja vasta mikäli tietoja ei toimiteta pyynnöistä huolimatta, vakuutuksenhakuprosessi voidaan katkaista. Suurin etu sisäkkäisistä transaktioista voidaan kuitenkin Haerderin ja Rothermelin (1987, 239), Lynchin ym. (1993, 17) sekä Wangin ym. (2008, 247) mukaan saavuttaa vasta hajautetuissa ympäristöissä.

Haittapuolena sisäkkäisissä transaktioissa voidaan nähdä mallin yleisellä tasolla säilyttämä transaktion jakamattomuus ja erillisyys. Koska sisäkkäisen transaktion tekemät muutokset astuvat voimaan vasta transaktion suorituksen päätyttyä, transaktion käyttämät resurssit ovat lukittuna koko suorituksen ajan. Tämä aiheuttaa luvussa 2.3 esitetyllä tavalla transaktioiden suoritusajan pidentymistä ja transaktioiden suorituksen ruuhkautumista. Barghoutin ja Kaiserin (1991, 280) mukaan ainoa sisäkkäisten transaktioiden tuoma hyöty on transaktion parempi tehokkuus, mikä johtuu transaktion osien rinnakkaisen suorituk-

sen mahdollistamisesta. Näin luonnostaan pitkien transaktioiden kesto saattaa lyhentyä, joka taas puolestaan helpottaa muiden transaktioiden suoritusta resurssien vapautuessa aikaisemmin niiden käyttöön. Wangin ym. (2008, 246) mukaan sisäkkäiset transaktiot eivät kuitenkaan ole tarkoitettuja pitkäkestoisten transaktioiden ympäristöihin. Tästä syystä sisäkkäisiä transaktioita ei myöskään tulisi käyttää liiketoimintaprosessien kuvaamisessa. Perinteisten sisäkkäisten transaktioiden konsepti on kuitenkin tarjonnut hyvän kehityspohjan muille sisäkkäistämistä hyödyntäville transaktiomalleille, jotka pyrkivät helpottamaan ylätason transaktion erillisyyden ominaisuutta, jotta transaktioiden yhtäaikainen suoritus helpottuisi (Wang ym., 2008, 245-246).

4.1.2 Saga-transaktiot

Transaktion ketjuttamisella tarkoitetaan Wangin ym. (2008, 246) mukaan jakamattoman transaktion osittamista peräkkäin suoritettaviin alitransaktioihin. Miettisen (2002, 45) mukaan kukin alitransaktio suorittaa erillisen osan transaktiosta ja on itsessään litteä transaktio eli säilyttää järjestelmän ristiriidattomuuden suorituksensa jälkeen. Ketjutettua transaktiota (*chained transaction*) ja yksittäisiä transaktioita ei kuitenkaan pidä sekoittaa toisiinsa, sillä ketjutetun transaktion pääajatus on mahdollistaa alitransaktioiden käsittely ketjuna eikä erillisinä transaktioina (vrt. luvussa 3.2 esitetty perinteinen tapa liittää transaktioita liiketoimintaprosessiin). Näin voidaan valvoa transaktion suoritusta kokonaisuutena ja estää osittain onnistuneen transaktion voimaantulo (Garcia-Molina & Salem, 1987, 250).

Chessellin ym. (2002, 745) sekä Worahin ja Shethin (1997, 8) mukaan ketjutetuilla transaktioilla voidaan keventää transaktion ACID -ominaisuuksia ylätason transaktion erillisyyteen liittyen ja täten helpottaa useiden transaktioiden yhtäaikaista suoritusta. Alitransaktion suorituksen valmistuttua seuraavan alitransaktion suoritus käynnistetään – näin edetään, kunnes kaikki alitransaktiot on suoritettu onnistuneesti ennalta määrätyn järjestyksen mukaisesti (Miettinen,

2002, 45; Wang ym., 2008, 246). Alitransaktiot ovat jakamattomia, ACID -ominaisuuksia noudattavia transaktioita, joten niiden vaikutukset ovat näkyvissä muille transaktioille välittömästi alitransaktion suorituksen valmistuttua (Alonso ym., 1996, 578). Näin ollen transaktioiden yhtäaikainen suoritus helpottuu alitransaktioiden tarvitseman resurssien lukitusajan ollessa pienempi kuin transaktiolla itsellään. Alitransaktioiden suorituksen valmistuttua lukitukset voidaan purkaa vaikka itse transaktio on vielä kesken, jolloin muut transaktiot pääsevät resursseihin nopeammin käsiksi (Alonso ym., 1996, 578; Chessell ym., 2002, 745). Lisäksi jokainen alitransaktio säilyttää tietokannan tilan ristiriidattomana (Alonso ym., 1996, 578; Garcia-Molina & Salem, 1987, 250).

Mikäli ketjutetun transaktion suorituksessa havaitaan virhe, tarvitsee ainoastaan suorituksessa ollut alitransaktio kumota ja aloittaa alusta, sillä aikaisemmin valmistuneet alitransaktiot ovat jo tehneet muutoksensa järjestelmään (Wang ym., 2008, 246). Tästä syystä ketjutettua transaktiota ei voida myöskään kokonaisuudessaan kumota, vaan epäonnistuneen alitransaktion suoritusta tulee yrittää yhä uudestaan, kunnes sen suoritus saadaan onnistumaan (Greenfield ym., 2003). Wangin ym. (2008, 246) mukaan näin ollen ketjutettu transaktio ei kokonaisuutena säilytä myöskään transaktion jakamattomuutta jo mainitun erillisyyden lisäksi.

Perustuen ketjutettujen transaktioiden ajatukseen Garcia-Molina ja Salem (1987) ovat esitelleet oman versionsa transaktiomallista, jossa valmistuneiden alitransaktioiden kumoaminen on mahdollista. Transaktiomallin mukaan ketjutettua transaktiota, joka sallii suoritettujen alitransaktioiden kumoamisen, kutsutaan saga-transaktioksi (*saga transaction*) tai lyhyemmin pelkäksi saagaksi (Butler ym., 2005; Wang ym., 2008, 246; Worah & Sheth, 1997, 8). Virhetilanteita varten jokaiselle alitransaktiolle viimeistä lukuun ottamatta määritellään saagassa erillinen kompensaatiotransaktio, joka suoritetaan kaikissa jo suoritetuissa alitransaktioissa, mikäli transaktion jokin alitransaktio epäonnistuu (Butler ym., 2005; Chessell ym., 2002, 744; Wang ym., 2008; Worah & Sheth, 1997). Kompensaatio-

transaktiot suoritetaan transaktiolle käänteisessä järjestyksessä (Butler ym., 2005; Chessell ym., 2002). Samoin toimitaan silloin, kun transaktio halutaan peruuttaa kesken suorituksen (Greenfield ym., 2003).

Kompensaatiotransaktion tehtävä on kumota alitransaktion tekemät muutokset, mutta olla palauttamatta tietokantaa aikaisempaan tilaan (Alonso ym., 1996, 578; Chessell ym., 2002, 744; Garcia-Molina & Salem, 1987, 250; Wang ym., 2008, 246). Kompensointi voi yksinkertaisimmillaan tarkoittaa esimerkiksi tietokannan päivitystä, mutta myös muut lisätoimenpiteet ovat mahdollisia. Perinteinen tehtyjen muutosten kumoaminen lokiin perustuen ei ole saagassa sallittua, sillä alitransaktioiden lukitusten vapauduttua muut transaktiot ovat saattaneet muokata resursseja alitransaktion suorituksen jälkeen saagan suorituksen ollessa yhä kesken. (Garcia-Molina & Salem, 1987, 250.) Kompensaatiotransaktioiden ansiosta saaga noudattaa ylätasolla jakamattomuuden vaatimusta, sillä saagaa voidaan pitää jakamattomana kaikkien vaiheiden kumoamisen mahdollisuuden takia. Saagan alitransaktiot ovat yksinkertaisten ketjutettujen transaktioiden alitransaktioiden tavoin myös jakamattomia, ja ne noudattavat ACID -ominaisuuksia. Näin saagalla voidaan nähdä olevan kaikki perinteisten ketjutettujen transaktioiden edut, mutta ne eivät kärsi ketjun jakamattomuuden puutteesta johtuvasta täydellisen kumoamisen ongelmasta perinteisten ketjutettujen transaktioiden tavoin. (Wang ym., 2008, 246.)

Saagat ovat erittäin käyttökelpoisia pitkien liiketoimintaprosessien mallintamisessa niiden kompensaatiomahdollisuuden ansiosta (Chessell ym., 2002, 756). Greenfieldin ym. (2003) mukaan saagat ovatkin olleet yksi merkittävimmistä transaktiomalleista liiketoimintaprosessien kuvaamiselle. Erityisesti liiketoiminnassa joudutaan usein tilanteeseen, jossa tietyn tehtävän kumoaminen ei ole mahdollista perinteisin transaktiohallinnan keinoin (ks. luku 3.3 kumoamiseen liittyvät ongelmat). Tästä syystä Butler ym. (2005) korostavatkin kompensoinnin tärkeyttä niissä tilanteissa, joissa järjestelmällä ei ole suoraa mahdollisuutta vaikuttaa suoritettuun toimenpiteeseen. Esimerkkinä tämänkaltaisista tilanteis-

ta voidaan nähdä ihmisen kanssa tapahtuva kanssakäyminen (Butler ym., 2005). Vakuutusyhtiön liiketoimintaprosessissa esimerkiksi virheellisen vakuutuskirjan lähettämistä asiakkaalle postitse ei voida tapahtumana kumota tietoteknisten keinojen avulla, vaan asiakasta täytyy informoida virheestä mahdollisesti toisella kirjeellä tai puhelinsoitolla. Samassa kompensaatiotransaktiossa vakuutuskirja voidaan myös esimerkiksi mitätöidä jollain tietoteknisillä keinoilla, jotka kuitenkin voivat pohjautua täysin eri resursseihin kun itse vakuutuskirjan lähettämistä koskenut alitransaktio. Näin alkuperäisen transaktion vaikutuksia ei yritetäkään poistaa, vaan ne kumotaan käyttäen hyväksi kompensaatiota. Saagojen ansiosta poikkeustilanne tulee käsiteltyä kompensaatiotransaktiossa, eikä manuaalisiin toimiin välttämättä jouduta transaktiokäsittelyn ulkopuolella transaktion kumoamisen jälkeen. Lisäksi saagojen etuna voidaan pitää niiden toteuttamisen helppoutta olemassa olevilla tietokannanhallintajärjestelmillä (Garcia-Molina & Salem, 1987, 251).

Joissain tapauksissa saagojen huonona puolena voidaan pitää transaktion jakamista vain kahteen tasoon. Tästä syystä transaktiomallilla ei voida kuvata kovin monimutkaisia transaktiorakenteita, joita voi esiintyä kehittyneissä järjestelmissä tai sovelluksissa. (Barghouti & Kaiser, 1991, 294; Garcia-Molina & Salem, 1987, 251.). Fischer ja Majumdar (2007, 54) myös huomauttavat, että saagat eivät ota kantaa tapahtumiin, joihin ei voi soveltaa kompensointia. Lisäksi kompensointia ei voida toteuttaa ehtorakenteisiin perustuen (Fischer & Majumdar, 2007, 54). Edellä mainittuja ongelmia on pyritty korjaamaan kehittämällä muun muassa useampitasoisia transaktiorakenteita hallitsevia malleja, jotka hyödyntävät kompensaatiota saga-transaktioiden tavoin (Barghouti & Kaiser, 1991, 294; Greenfield ym, 2003; Worah & Sheth, 1997, 8).

4.2 Kehittyneiden transaktiomallien soveltuvuus liiketoimintaprosessien kuvaamiseen

Kuten käsitellyistä kehittyneistä transaktiomalleista kävi ilmi, ei sisäkkäisillä transaktiolla tai saagoilla voida kuvata liiketoimintaprosesseja ongelmitta. Alonson ym. (1996) mukaan monimutkaisissa sovelluksissa esiintyy monenlaisia ongelmia, joihin ei pystytä vastaamaan kehittyneillä transaktiomalleilla. Syyn epäillään olevan kehittyneiden transaktiomallien soveltumattomuudessa tosielämän tilanteisiin. Alonso ym. (1996, 574) pitävät kehittyneitä transaktiomalleja liian tietokantakeskeisinä, joka rajoittaa mallien käyttömahdollisuuksia ja ulottuvuutta.

Vähitellen transaktioiden käyttöä monimutkaisten ja pitkien liiketoimintaprosessien kuvaamisessa ovat syrjäyttäneet niin sanotut työnkulun hallintajärjestelmät (*workflow management system, WFMS*), joissa on pyritty luopumaan transaktioiden ACID -ominaisuuksien noudattamisesta ja keskitytty vastaamaan paremmin työkulun vaatimuksiin (Alonso ym., 1996, 574; Wang ym., 2008). Niissä pääajatuksena on keskittyä kuvaamaan prosessin etenemistä ACID -ominaisuuksien tarkastelun sijasta (Wang ym., 2008, 251). Casati, Ceri, Paraboschi ja Pozzi (1999, 2) määrittelevät Georgakopoulokseen ym. (1995) viitaten työkulun hallintajärjestelmät järjestelmiksi, joiden tarkoituksena on tukea organisaation jäsenten yhteistoimintaa ja koordinointia helpottamaan monimutkaisten liiketoimintaprosessien suoritusta. Alonson, Agrawalin, Abbadin ja Mohanin (1997, 2) sekä Papazogloun (2003, 49–50) mukaan työkulun hallintajärjestelmiä käytetäänkin perinteisesti juuri liiketoimintaprosessien yhteydessä: liiketoimintaprosessi kuvataan työkulkuna (*workflow*), jossa esitetään prosessin jokainen vaihe ja vaiheeseen liittyvät parametrit, joita vaaditaan prosessin suorittamiseen. Työkulkua voidaan näin pitää tietokoneistettuna liiketoimintaprosessin mallina (Alonso ym., 1997, 2).

Työkulut muistuttavat paljon kehittyneitä transaktiomalleja, sillä myös niissä liiketoimintaprosessi voidaan osittaa pienempiin osiin kahdella jo esitellyllä ta-

valla (Wang ym., 2008, 248). Työkulun hallintajärjestelmät hyödyntävät olemassa olevia kehittyneitä transaktiomalleja pitkien ja kompleksisten työkulkujen kuvaamisessa: esimerkiksi niin sanottu WIDE -malli jakaa työkulun kahdeksi kerrokseksi (*layer*), joissa hyödynnetään jatkettua saga-transaktioiden mallia ylätason työkulun mallintamisessa ja sisäkkäisten transaktioiden mallia alatason tapahtumien kuvaamisessa (Grefen, Vonk, Boertjes & Apers, 1997; Grefen, Vonk & Apers, 2001; Wang ym., 2008, 247). Yhdistelemällä jatkettuja transaktiomalleja edistyneellä tavalla työkuluiksi voidaan näin saavuttaa parempia lopputuloksia, kuin kuvaamalla liiketoimintaprosesseja puhtaasti transaktioina, jotka noudattavat edes kevennettyjä ACID -ominaisuuksia.

5 YHTEENVETO

Tässä tutkielmassa tarkasteltiin sekä perinteisen ACID -ominaisuuksia noudattavan transaktion että kehittyneempien transaktiomallien soveltuvuutta liiketoimintaprosessien kuvaamiseen. Aluksi keskityttiin tarkastelemaan transaktioita tietokantaympäristössä sekä hahmottelemaan transaktioiden ja liiketoimintaprosessien suhdetta toisiinsa, jonka jälkeen tutkittiin liiketoimintaympäristön asettamia haasteita ACID -ominaisuuksien noudattamiselle. Tämän jälkeen tutkielmassa selvitettiin, ratkaisevatko kehittyneet transaktiomallit ACID -ominaisuuksia noudattavan transaktion kohtaamat ongelmat liiketoimintaprosessin kuvaamisessa.

Yrityksen liiketoimintaprosessin määritelmä muistuttaa hyvin paljon perinteistä transaktion määritelmää, jossa transaktion sanotaan muodostuvan ketjusta erillisiä toimintoja, joiden suorituksella on tarkoitus saavuttaa halutun kaltainen lopputulos. Vastaavalla tavalla yrityksen liiketoimintaprosessin voidaan sanoa koostuvan toimintojen ketjusta, jonka suoritus tähtää jonkun liiketoimintatehtävän suorittamiseen. Transaktion ja liiketoimintaprosessin käsitteet ovat näin rinnastettavissa – Bennett ym. (2000) sekä Chen ja Buchs (2006) käsittelevät julkaisuissaan liiketoimintaprosesseja, joiden suoritukseen on liitettävissä transaktiomaisia piirteitä tehden niistä eräänlaisia transaktiovaatimuksia noudattavia liiketoimintaprosesseja.

Vaikka tietokantaympäristöstä tutut ACID -ominaisuudet soveltuvat lähtökohtaisesti hyvin liiketoimintaprosessien kuvaamiseen, aiheuttaa niiden noudattaminen liiketoimintaympäristössä merkittäviä ongelmia. Tämä johtuu muun muassa liiketoimintaprosessien pituudesta ja monimutkaisuudesta, joka tekee syntyvistä transaktioista pitkäkestoisia. Transaktioksi mallinnetut liiketoimintaprosessit lukitsevat käyttöönsä tietokantaresursseja pidentäen muiden transaktioiden suoritusaikaa ja estäen liiketoimintaprosessiin osallistumisen. Lisäksi liiketoimintaprosessin vaiheiden kumoaminen ei ole enää täysin mahdollista

joidenkin tehtävien ollessa manuaalisia tai ei-transaktiomaisia. Myöskään pitkiä liiketoimintaprosesseja ei ole aina mielekästä kumota täydellisesti ACID -ominaisuuksien vaatimalla tavalla, sillä prosessin kumoamisen on todettu olevan erittäin kallista.

Kehittyneet transaktiomallit noudattavat transaktion ACID -ominaisuuksia kevennetysti siten, että ne pyrkivät luopumaan tietyistä ACID -ominaisuuksista helpottaakseen pitkien transaktioiden selviytymistä virhetilanteista ilman koko transaktion kumoamista. Sisäkkäiset transaktiot helpottavat virheistä selviytymistä hierarkkisella alitransaktiorakenteella, jossa transaktio jaetaan pienempiin osatransaktioihin eräänlaiseksi puumaiseksi rakenteeksi. Transaktiomalli noudattaa ylätasolla ACID -ominaisuuksia, mutta yksittäiset alitransaktiot suoritetaan joko ACI tai AI -ominaisuuksiin perustuen. Tällöin yksittäisille alitransaktioille sallitaan epäonnistuminen sen vaarantamatta koko transaktion onnistumista. Lähestymistapa tuo samalla tehokkuutta transaktion suorittamiseen, sillä alitransaktioita voidaan suorittaa yhtä aikaa niiden ollessa toisistaan riippumattomia. Sisäkkäiset transaktiot eivät kuitenkaan onnistu välttämään pitkien transaktioiden ongelmaa, sillä ylätason transaktiota käsitellään edelleen jakamattomana.

Saga-transaktiot puolestaan jakavat transaktiot peräkkäin suoritettaviksi alitransaktioiksi, joilla kullakin on oma, alitransaktion vaikutukset kumoava kompensaatiotransaktio. Ylätasolla saaga ei enää pyri noudattamaan jakamattomuuden ja erillisyyden ominaisuuksia, mutta alitransaktiot ovat kukin ACID -ominaisuuksia noudattavia transaktioita. Saagaa suoritetaan yksi alitransaktio kerrallaan kunnes viimeinenkin alitransaktio on valmis. Virheen sattuessa kaikille jo suoritetuille alitransaktioille suoritetaan niiden oma kompensaatiotransaktio – näin saagan sanotaan säilyttävän ylätasolta katsottuna jakamattomuuden ominaisuutensa täydellisen kumoamisen ansiosta. Saagoilla voidaan vähentää pitkään transaktioon liittyvää resurssien lukitusta, sillä alitransaktiot tekevät muutoksensa pysyviksi järjestelmään heti valmistuttuaan. Saa-

gojen huonona puolena voidaan kuitenkin pitää transaktioiden jakamista vain kahteen tasoon. Lisäksi kompensointi on joissain tapauksissa riittämätöntä ehtorakenteiden tai kompensointikelvottomien tapahtumien osalta. Kehittyneetkään transaktiomallit eivät siis ole riittäviä liiketoimintaprosessien kuvaamiseen transaktiomaisesti.

Tutkimustulokset ovat merkityksellisiä ja hyödyllisiä pohdittaessa liiketoimintaprosessien käytännön toteutusta ja automatisointia liiketoimintaympäristössä tai kehitettäessä olemassa olevia liiketoimintaprosesseja paremmin liiketoimintaympäristöön soveltuviksi. Mitä lähempänä liiketoimintaprosessit ovat tietokantamaailman toteutusta, sitä hyödyllisempinä tämän tutkielman tuloksia voidaan pitää käytännön hyötyjen saannin kannalta, johtuen tarkastelun tietokantalähtöisyydestä.

Tässä tutkielmassa keskityttiin tarkastelemaan liiketoimintaprosessin mallintamista transaktiona ACID -ominaisuuksien näkökulmasta tarkastellen. Mahdolliseksi jatkotutkimusaiheeksi ehdottaisin luvussa 4.2 lyhyesti esiteltyjen työkulkujen hyödyntämismahdollisuuksien lähemmän tarkastelun liiketoimintaprosessien mallintamisessa ja automatisoinnissa. Lisäksi tässä tutkielmassa sivuutettiin täysin transaktiolähestymistavan käytännön toteutuksen arviointi, johon tulevaisuudessa voisi olla hyvä ottaa kantaa, kun verrataan sitä työkulkujen käytännön toteutukseen.

LÄHTEET

Alonso, G., Agrawal, D., Abbadi, A. E., & Mohan, C. (1997). Functionality and limitations of current workflow management systems. *IEEE Expert, 12*

Alonso, G., Agrawal, D., Abbadi, A. E., Kamath, M., Günthör, R., & Mohan, C. (1996). Advanced transaction models in workflow contexts. *ICDE '96: Proceedings of the Twelfth International Conference on Data Engineering,* 574–581.

Barghouti, N. S. & Kaiser, G. E. (1991). Concurrency control in advanced database applications. *ACM Comput.Surv., 23*(3), 269–317.

Bennett, B. T., Hahm, B., Leff, A., Mikalsen, T. A., Rasmus, K., Rayfield, J. T., & Rouvellou, I. (2000). A distributed object oriented framework to offer transactional support for long running business processes. *Middleware,* 331–348.

Breitbart, Y., Garcia-Molina, H., & Silberschatz, A. (1992). Overview of multidatabase transaction management. *CASCON '92: Proceedings of the 1992 Conference of the Centre for Advanced Studies on Collaborative Research,* Toronto, Ontario, Canada. 23–56.

Butler, M. J., Hoare, C. A. R., & Ferreira, C. (2004). A trace semantics for long-running transactions. *25 Years Communicating Sequential Processes,* 133–150.

Casati, F., Ceri, S., Paraboschi, S., & Pozzi, G. (1999). Specification and implementation of exceptions in workflow management systems. *ACM Trans.Database Syst., 24*(3), 405–451.

Chen, A. & Buchs, D. (2006). Transactional Business Process Design and Prototyping: A COOPN approach. University of Geneva. CUI Technical Report.

Chessell, M., Griffin, C., Vines, D., Butler, M., Ferreira, C., & Henderson, P. (2002). Extending the concept of transaction compensation. *IBM Syst.J., 41*(4), 743–758.

Davenport, T. H. & Short, J. E. (1990). The new industrial engineering: Information technology and business process redesign. *Sloan Management Review, 31*(4), 11–27.

Eswaran, K. P., Gray, J. N., Lorie, R. A., & Traiger, I. L. (1976). The notions of consistency and predicate locks in a database system. *Commun.ACM, 19*(11), 624–633.

Ferreira, C. (2002). Precise Modelling of Business Processes with Compensation. PhD thesis, University of Southampton.

Fischer, J. & Majumdar, R. (2007). Ensuring consistency in long running transactions. *ASE '07: Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering,* Atlanta, Georgia, USA. 54–63.

Garcia-Molina, H. & Salem, K. (1987). Sagas. *SIGMOD Rec., 16*(3), 249–259.

Garcia-Molina, H. (1983). Using semantic knowledge for transaction processing in a distributed database. *ACM Trans.Database Syst., 8*(2), 186–213.

Gray, J. (1981). The transaction concept: Virtues and limitations (invited paper). *Very Large Data Bases, 7th International Conference, September 9-11, 1981, Cannes, France, Proceedings,* 144–154.

Greenfield, P., Fekete, A., Jang, J., & Kuo, D. (2003). Compensation is not enough. *EDOC '03: Proceedings of the 7th International Conference on Enterprise Distributed Object Computing,* 232–239.

Grefen, P. W. P. J., Vonk, J., Boertjes, E. M., & Apers, P. M. G. (1997). Two-layer transaction management for workflow management applications. *Lecture Notes in Computer Science; Proceedings of the 8th International Conference on Database and Expert Systems Applications (DEXA 1997), Toulouse, France,* Toulouse, France. , *1308* 430–439.

Grefen, P., Pernici, B., & Sanchez, G. (1999). *Database support for workflow management: The wide project*. Norwell, MA, USA: Kluwer Academic Publishers.

Grefen, P., Vonk, J., & Apers, P. (2001). Global transaction support for workflow management systems: From formal specification to practical implementation. *The VLDB Journal, 10*(4), 316–333.

Haerder, T. & Reuter, A. (1983). Principles of transaction-oriented database recovery. *ACM Comput.Surv., 15*(4), 287–317.

Haerder, T. & Rothermel, K. (1987). Concepts for transaction recovery in nested transactions. *SIGMOD Rec., 16*(3), 239–248.

Hrastnik, P. & Winiwarter, W. (2007). Coordination in service oriented architectures using transaction processing concepts. *DEXA '07: Proceedings of the 18th International Conference on Database and Expert Systems Applications,* 855–860.

Korth, H. F., Levy, E., & Silberschatz, A. (1990). A formal approach to recovery by compensating transactions. *In Proceedings of the 16th International Conference on very Large Data Bases,* 95–106.

Laiho, M. (2005). SQL-transaktiot [SQL transactions]. Retrieved November 13, 2008, from http://myy.helia.fi/~ict1td003/rdbms/mats/Transaktiot_Toipuminen.pdf

Lynch, N. A., Merritt, M., Weihl, W. E., & Fekete, A. (1993). *Atomic transactions.* San Mateo, CA, USA: Morgan Kaufmann.

Miettinen, A. (2002). Distributed transaction management. In: J. Porrasmaa (Ed.), *Database Solutions for Componet-Based Software Development in Healthcare* (p. 43–50). Kuopio University Occasional Reports C. Natural and Environmental Sciences 9.

O'Neil, P. E. (1986). The escrow transactional method. *ACM Trans.Database Syst., 11*(4), 405–430.

Papazoglou, M. P. (2003). Web services and business transactions. *World Wide Web, 6*(1), 49–91.

Schuldt, H., Alonso, G., Beeri, C., & Schek, H. (2002). Atomicity and isolation for transactional processes. *ACM Trans.Database Syst., 27*(1), 63–116.

Wang, T., Vonk, J., Kratz, B., & Grefen, P. (2008). A survey on the history of transaction management: From flat to grid transactions. *Distrib.Parallel Databases, 23*(3), 235–270.

Veijalainen, J. (2000). Transactions in mobile electronic commerce. *Selected Papers from the Eight International Workshop on Foundations of Models and Languages for Data and Objects, Transactions and Database Dynamics,* 203–224.

Wolfson, O. & Yannakakis, M. (1985). Deadlock-freedom (and safety) of transactions in a distributed database. *PODS '85: Proceedings of the Fourth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems,* Portland, Oregon, United States. 105–112.

Worah, D. & Sheth, A. P. (1997). Transactions in transactional workflows. *Advanced transaction models and architectures* (pp. 3–34)

Young, C. (2006). BizTalk Server 2006 The Compensation Model. Retrieved December 5, 2008, from http://geekswithblogs.net/cyoung/articles/100424.aspx