Työnkulun formalisointi ja samanaikaisuuden oikeellisuus

Jan Lindström

Työnkulkujärjestelmät seminaari

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

Sisältö

1 Johdanto

Työnkulkujärjestelmät (Workflow Management Systems, WFMSs) ovat vakiinnuttaneet paikkansa yhtenä menetelmänä toteuttaa suuria ja hajautettuja suoritusympäristöjä, joissa joukko yhteenkuuluvia aktiviteetteja suoritetaan tehokkaasti ja tarkasti valvotusti. Työnkulkujärjestelmiin liittyy myös standardointijärjestö. Workflow Management Coalition (WfMC) on yrityspohjainen yhteisö, jonka tavoitteena on yhtenäistää käsitteistö ja standardoida työnkulkujärjestelmien avainkomponentit. WfMC on tunnistanut kuusi primitiiviä, joiden avulla on mahdollista kuvata *ohjausvuo* (control-flow) ja siten luoda työnkulun määritelmä.

Työnkulkuprosessi (workflow process) on määritelty joukoksi suoritusaskelia, jotka ovat organisoitu muodostamaan jokin liikeprosessi (busines process). Aktiviteetin voi suorittaa yksi tai useampi ohjelmisto tai kone, ihminen tai ryhmä tai yhdistelmä näistä. Työnkulkuprosessi sisältää joukon aktiviteetteja ja määrittelee järjestyksen aktiviteettien suoritukselle tai ehdot joiden täytyttyä aktiviteetti täytyy käynnistää. Työnkulkuprosessi määrittelee myös *tietovuon* (data-flow) aktiviteettien välillä. Aktiviteetti voi olla työnkulku. Lisäksi aktiviteetti voi koostua useammasta kutsusta paikalliseen järjestelmään (esimerkiksi tietokantaan) ja tämä osuus on piilotettu työnkulkutasolla.

Aktiviteetit voivat olla *transaktionaalisia* (transactional) tai *ei-transaktionaalisia* (non-transactional) [RS94a, RS94b, PTV96, PTV97, Puu99]. Tranksaktionaalisissa aktiviteeteissä tiedon käsittelyä kontrolloidaan *resurssihallitsimella* (Resource Manager, RM), jolla on transaktionaalisia ominaisuuksia (siis ACID ominaisuuksia). Transaktionaalisten aktiviteettien täytyy vähintään tarjota atomisuusominaisuuden ja maksimissaan tukea kaikkia tavanomaisen tapahtumamallin ACID ominaisuuksia. Transaktionaalisiin aktiviteetteihin tyypillisesti luetaan aktiviteetit, jotka keskustelevat tietokannanhallintajärjestelmän kanssa sitoudu (commit) ja peruuta (abort) operaatioilla, tallettetuilla proseduureilla ja kaksivaiheisella sitoutumisella (2PC).

Ei-transaktionaaliset aktiviteetit yleensä käsittelevät tietoa, jota hallitsee resurssihallitsin ilman transaktionaalisia ominaisuuksia. Tällaisiin ei-transaktionaalisia toimintoja ovat esimerkiksi tiedostojärjestelmä, ihmiset, HTTP palvelimet, tekstinkäsittely- ja taulukkolaskentaohjelmat.

Georgakopoulos ym. [GHS95] ovat artikkelissaan esittäneet, että aktiviteetin toteuttaja on vastuussa siitä, että yksinään suoritettu aktiviteetti tuottaa oikean tuloksen. Koska työnkulkujärjestelmät ovat pitkäkestoisia, voidaan aktiviteetin olettaa päättyvän (sitoutuvan) työnkulun ilmentymän aikana. Siksi aktiviteettien muuttama tieto, joka tulee näkyviin muille aktiviteeteille samassa työnkulussa tai muille työnkulun ilmentyville, voi aiheuttaa epäeheyttä järjestelmää virheellisen aktiviteettien lomittumisen takia. Kuitenkin useat työnkulkujärjestelmän operaatiot vaativat eheyden säilymistä ainakin osasta tietoalkioita. Siksi työnkulkujärjestelmien on huomioitava seuraavat oikeellisuuskohteet:

- Samaan työnkulkuilmentymään kuuluvien aktiviteettien samanaikaisen suorituksen oikeellisuus.
- Eri työnkulkuilmentymien aktiviteettien samanaikaisen suorituksen oikeellisuus.

Worah ja Seth [WS96] ovat artikkelissaan esittäneetkin, että tällä hetkellä työnkulkujärjestelmistä puuttuu selkeä teoreettien pohja, oikeellisuuskriteerit ja tuki samanaikaisten työnkulkujen eheydelle. Siksi Arpinar ym. [AHAD99] tämän työn pohjana olevassa artikkelissa ovat esittäneet teoreettisen pohjan työnkulun formalisointiin, määritelleet oikeellisuuskriteerin samanaikaisesti suoritettaville työnkuluille pohjaten tähän formalisointiin ja ehdottaneet samanaikaisuudenhallintamenetelmää, joka tarjoaa määritellyn oikeellisuuden.

Tämä esitys etenee seuraavasti. Kappaleessa 2 lyhyesti kuvataan työnkulun osien teoretisoinnit intuitiivisella tasolla (halutessaan matemaattisesti täsmälliset määritelmät lukija voi konsultoida alkuperäisartikkelia). Kappaleessa 3 esitetaan työnkulkuun liittyvät oikeellisuuskriteerit. Rajoitteisiin perustuva samanaikaisuudenhallintamenetelmä esitellään kappaleessa 4. Lopuksi kappaleessa 5 esitetään yhteenveto.

2 Työnkulun osien formalisointi

Tässä kappeleessa esitetään työnkulkujärjestelmien pääkomponenttien formalisointi. Pääkomponentteja ovat komponentit, jotka ovat merkittäviä samanaikaisessa suorituksessa. Formalisointi perustuu joukko- ja graafiteoriaan.

Formalisointi muodostaa teoreettisen perustan oikeellisuuskriteerille ja voidaan esittää lyhyesti seuraavasti:

- Työnkulun aktiviteetit esitetään joukkoteoriaan perustuvalla notaatiolla, jonka avulla voidaan formalisoida aktiviteettien käsitteellinen ryhmittely.
- Ohjausvuo esitetään erityisellä graafilla, joka perustuu joukon määritelmään. Ohjausvuo sisältää peräkkäisryhmittelyn (serial composition), rinnakkaisryhmittelyn (parallel composition), ehdollisen haarautumisen (conditional branching) ja yksittäisten aktiviteettien ja käsitteellisten aktiviteettien sisäkkäisrakenteen (nesting).
- Tietovuo esitetään suunnatulla syklittömällä verkolla samaan tapaan kuin ohjausvuo.

Samanaikaisesti suoritettavien työnkulkuilmentymien oikeellisuuden formalisointi perustuu tähän malliin määrittelemällä työnkulun ympäristön rajoitteille kaksi kategoriaa. Työnkulun ympäristössä työnkulun ilmentymät ja niiden aktiviteetit vaikuttavat toisiinsa. Kategoriat ovat:

- *Perusrajoitteet* (basic constraints), jotka määrittelevät työnkulkuympäristön eheät tilat.
- *Aktiviteetin väliset rajoitteet* (inter-activity constraints), joka määrittelevät semanttiset riippuvuudet aktiviteettien välillä. Kuten aktiviteetti tarvitsee rajoitteen oikeellisuutta, joka asetetaan tai varmistetaan kyseistä aktiviteettiä edeltävässä aktiviteetissä.

Perusrajoitegraafi ja aktiviteettien välisten rajoitteiden graafi ovat yhdenmukaisia ohjausvuon kanssa. Tietovuokaaviot määritellään esittämään näitä rajoitteita. Graafeja käytetään formalisoimaan aktiviteettien taukoja (intervals), joissa aktiviteetin väliset rajoitteet tulisi säilyttää ja taukoja, joissa perusrajoitteet eivät päde.

Oikeellisuuskriteerio määritellään lomittuvien työnkulkuilmentymien suorituksille käyttäen rajoitegraafeja. Samanaikaisuudenhallintamenetelmä, jota nimitetään *rajoitteisiin perustuvaksi samanaikaisuudenhallinnaksi* (Constraint Based Concurrency Control, CBCC) perustuu tähän oikeellisuuskriteeriin.

Käytännössä artikkelissa on esitetty formaali määrittely työnkululle, joka on yhdenmukainen WfMC mallin ohjausvuoprimitiiveihin. Formaali määrittely perustuu joukko- ja graafiteoriaan.

Arpinarin ym. artikkeli määrittelee erityisen joukon, jonka jäseniä voivat olla myös joukot. Tätä joukkoa käytetään esittämään työnkukujärjestelmän aktiviteettien käsitteellistä ryhmittelyä. Ohjausvuo asetetaan tähän joukkoon lisäämällä yhteenkuuluvia särmiä. Jako- ja yhdistyssolmut lisätään graafiin, jos ohjausvuo jakaantuu useampaan haaraan ja yhdistyy yhteen vuohon myöhemmin. Työnkulun tietovuo esitetään yksinkertaisella suunnatulla syklittömällä verkolla.

Työnkulun aktiviteetit käyttävät resursseja, jotka ilmaisevat kaikkia olioita työnkulkuympäristössä. Arpinar ym. artikkelissaan määritelvät oikean aktiviteettien suorituksen syöte- ja tulosrajoitteiden avulla.

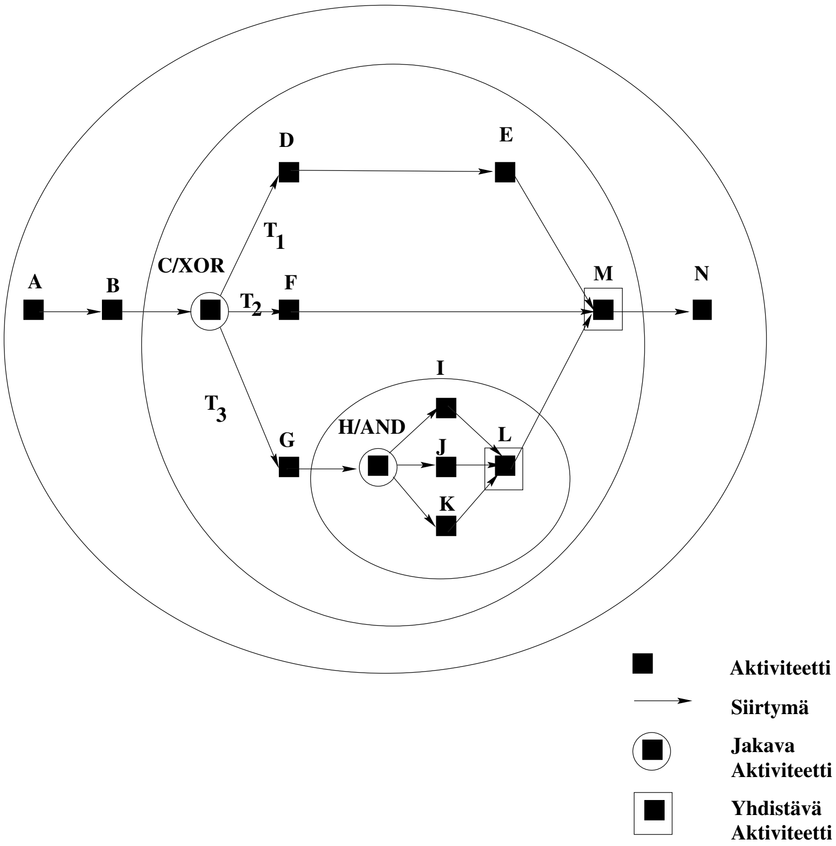
Esimerkiksi perusrajoite voi ilmaista rahan siirrännän kahden pankin konttorin välillä Nosta-Talleta työnkululla. Perusrajoite pysyy epäeheänä Nosta ja Talleta aktiviteettien suorituksen aikana. Toinen esimerkki on LisääVarasto aktiviteetti. Koska tuloksena syntyvä määrä raaka-aineita täytyy pysyä varastossa LisääVarasto aktiviteetin suorituksen jälkeen, tämä rajoite voidaan esittää aktiviteetin välisillä rajoitteella LisääVarasto aktiviteettin ja TuotaTuote aktiviteetin välillä.

Aktiviteettien väliset välit, joissa aktiviteettien väliset rajoitteet pitäisi säilyttää ja perusrajoitteet pysyvät epäeheinä, formalisoidaan vastaaviin rajoitteisiin liittyvillä graafeilla. Näitä graafeja käytetään kehitettäessa oikeellisuuskriteeriota lomittain suoritettaville työnkuluille, mitkä esitetään täydellisen suoritushistorian avulla. Oikeellisuuskriteerio vaatii kahden ehdon pätevyyttä:

1. Aktiviteettien väliset rajoitteiden tulisi esittää aktiviteettien semanttisia riippuvuuksia. Rajoitteiden tulisi estää suorittamasta aktiviteetteja, jotka rikkoisivat aktiviteettien välisiä rajoitteita.

2. Estetään perusrajoitteiden eheyttä vaativien aktiviteettien suoritus aktiviteettien välissä, jossa perusrajoitteet eivät ole voimassa.

Tässä esityksessä jätän väliin lähes kaiken formaalin esityksen artikkelista. Tässä esitetään vain työnkulun määritelmä ideatasolla käyttäen kuvassa 1 olevaa esimerkkiä.



Kuva 1: Esimerkkityönkulku

MÄÄRITELMÄ 2.1 *Työnkulku W on viisikko muotoa $W = (N, CF, DF, IC, BC)$, missä*

1. *N on sisäkkäinen hyperjoukko (hyperSet), jonka koostuu elementeistä $base(N) = T \cup S \cup J \cup \{f, l\}$, missä T on joukko yksittäisiä aktiviteetteja, S on joukko jakavia aktiviteetteja (split activities), J on joukko yhdistäviä aktiviteetteja (join activities), ja f on ensimmäinen aktiviteetti ja l viimeinen aktiviteetti työnkulussa.*

2. *$CF = (N, E_{CF}, L, TC)$ on nimetty jako-yhdistys hypersolmugraafi (split-join hyperNodeDAG) joukolle N vastaten ohjausvuota. L on funktio joukosta S joukkoon $\{and, or, xor\}$, joka esittää jakavien solmujen tyyppejä. TC on funktio jokaisesta $< s, j >$-polusta ohjausvuossa $\{T_1, T_2, ..., T_i, ..., T_n\}$, missä $s \in S$ on or/xor-jakava aktiviteetti, $j \in J$ on vastaava yhdistävä aktiviteetti, ja T_i on ehtolauseke.*

3. *$DF = (T, E_{DF})$ on suunnattu syklitön verkko, joka esittää tietovuota siten, että DF on yhtäpitävä CF:n kanssa.*

4. *$IC = (V_{IC}, E_{IC}, L_{IC})$ on nimetty hypergraafi, joka esittää aktiviteettien välisten rajoiden graafin.*

5. *$BC = (V_{BC}, E_{BC}, CL_{BC}, VL_{BC})$ on nimetty hypergraafi, joka esittää perusrajoitegraafin.*

ESIMERKKI 2.1 *Kuva 1 esittää yhden työnkulun. Kuva muodostaa samalla jako-yhdisty hypersolmugaafin, joka vastaa ohjausvuota. Kuvasta muodostuvassa graafissa $N = \{a, b, \{c, d, e, f, g, \{h, i, j, k, l\}, m\}, n\}$, $T = \{b, d, e, f, g, i, j, k\}$, $S = \{c, h\}$, $J = \{l, m\}$, $f = a$, ja $l = n$. Lisäksi $L(c) = xor$, $L(h) = and$, $TC(< c, m > -path_1) = T_1$, $TC(< c, m > -path_2) = T_2$, ja $TC(< c, m > -path_3) = T_3$.*

3 Aktiviteettien ja työnkulun oikeellisuus

Tässä kappaleessa esitetetään työnkulun oikeellisuus samanaikaisuuden vallitessa. [1] Työnkulku sisältää useita aktiviteetteja, joista jokaisen suorittaa agentti. Akvititeetit voivat käsitellä resursseja, jotka ilmaisevat joukkoa olioita muodostaen työnkulkuympäristön. Virheetön aktiviteetin suoritus määritellään syöttö- ja tulosspesifikaatiolla, jotka voivat olla joukko rajoitteita työnkulkuympäristöstä. Rajoitteet luokitellaan kahteen ryhmään, joita kutsutaan perusrajoitteiksi ja aktiviteettien välisiksi rajoitteiksi. Rajoitteet voidaan määritellä formaalisti ensimmäisen kertaluokan logiikan kaavoina.

Lomittaiset työnkulkujen suoritukset esitetään täydellisen historian avulla. Käyttämällä syöttö- ja tulosrajoitteita voidaan määritellä saadun historian oikeellisuus. Syöterajoite voi koostua kahdentyyppisistä rajoitteista: perusrajoitteista ja aktiviteettien välisistä rajoitteista. Perusrajoite määrittelee työnkulkuympäristön eheät tilat. Aktiviteetin väliset rajoitteeet määrittelevät semanttiset riippuvuudet rajoitteiden välillä. Esimerkiksi rajoitteet, jotka asetetaan tai varmistetaan ennen aktiviteetin suoritusta. Täydellinen historia on oikea jos syöterajoite jokaiselle

[1] Formaalit määritelmät esitetään vain intuitiivisella tasolla. Halutessaan lukija voi konsultoida Arpinarin ym. alkuperäisartikkelia.

aktiviteetille tässä historiassa on voimassa aktiviteetin alkaessa ja jos aktiviteetin alussa voimassa ollut perusrajoite on voimassa myös historian lopussa.

MÄÄRITELMÄ 3.1 *Akvititeetti t on viisikko $t = (IP, OP, RS, WS, AS)$, missä IP on syöteparametrit, OP on paluuparametrit, RS on t:n lukemat oliot, WS on t:n kirjoittamat oliot ja AS on aktiviteetin spesifikaatio.*

Aktiviteetti on virheetön spefikaationsa suhteen jos aktiviteetin t suoritus alkaessaan alkutilasta S_t päättyy johonkin lopputilaan $S_t^{'} = t(S_t)$. Aktiviteettien oletetaan olevan virheettömiä. Aktiviteetin tulosrajoite käsittää rajoitteet työnkulkuympäristölle, jossa työnkulkujärjestelmän täytyy olla aktiviteetin suorituksen päätyttyä.

ESIMERKKI 3.1 *PoistaVarastastosta (lyhyesti t_{WFS}) aktiviteetin tarkoituksena on poistaa vatastosta tietty määrä tuotteita m_i. Aktiviteetin tulosrajoite voidaan määritellä seuraavasti: $O_{t_{WTS}} \equiv$ (määrä$(M_i)'$ = määrä(m_i) − lkm(m_i)). $O_{t_{WTS}}$ määrittelee, että m_i:n varaston kokoa vähennetään $lkm(m_i)$ määrällä.*

Syöterajoite kuvaa joukkoa alkutiloja siten, että aktiviteetin päättyminen johtaa tulosrajoitteen eheänä säilyttävään lopputilaan. Siis aktiviteetin syöterajoite kuvaa työnkulkuympäristön tilat, joissa aktiviteetti voidaan suorittaa virheettömästi. Aktiviteetin syöterajoitteen tilasta riippuen kolme tapausta on mahdollista:

1. Aktiviteetin t aktivointi johtaa eheään lopputilaan.

2. Aktiviteetin t aktivointi johtaa epäeheään lopputilaan.

3. Aktiviteetin t aktivointi ei johda lopputilaan eli aktiviteetin suoritus epäonnistuu.

Jos syöterajoite on voimassa ja koska aktiviteetti oletetaan virheettömäksi, vain ensimmäinen kohta on mahdollinen. Jos syöterajoite ei ole voimassa, mikä tahansa kolmesta tapauksesta on mahdollinen.

ESIMERKKI 3.2 *Aktiviteetin t_{WFS} syöterajoite ilmaisee, että riittävä määrä raaka-ainetta m_i on oltava varastossa: $I_{t_{WFS}} \equiv$ (määrä$(m_i) \geq$ lkm(m_i)).*

Seuraavaksi esitetään funktio, jonka avulla tutkitaan säilyykö jokin tietty rajoite aktiviteetin suorituksen jälkeen eheänä.

MÄÄRITELMÄ 3.2 *Säilytysfunktio (preserve funktion). Olkoon t aktiviteetti ja T työnkulkuympäristöön liittyvä ensimmäisen kertaluokan logiikan lause. $S_{t_\forall}$ on kaikkien tilojen joukko. Säilytä(t, F) on kolmiarvoinen funktio, joka määritellään seuraavasti:*

1. *Säilytä$(t, F) =$ true jos $(\forall S_t \in S_{t_\forall}) : ((S_t \models (t(S_t)F)) \Rightarrow (t(S_t) \models F))$. Tässä tapauksessa sanotaan, että t säilyttää F:n.*
2. *Säilytä$(t, F) =$ false jos $(\forall S_t \in S_{t_\forall}) : ((S_t \models (t(S_t)F)) \Rightarrow (t(S_t) \not\models F))$. Tässä tapauksessa sanotaan, että t rikkoo F:n.*
3. *Säilytä$(t, F) =$ ehkä jos $(\exists S_t \in S_{t_\forall}) : ((S_t \models (t(S_t)F)) \Rightarrow (t(S_t) \not\models F))$. Tässä tapauksessa sanotaan, että t saattaa rikkoa F:n.*

Intuitiivisesti on selvää, että Säilytä$(t, F) =$ false tai ehkä vaatii, että $WS(t) \cap O(F) \neq \emptyset$, missä $O(F)$ sisältää syöteparametrit ja tämänhetkiset muuttujat.

TEOREEMA 3.1 *Täydellisen historian oikeellisuus. Olkoon $CH = (T_{CH}, E_{CH}, L_{CH})$ täydellinen suoritushistoria joukolle täydellisiä suorituksia $CH = \{CE_1, CE_2, ..., CE_n\}$, missä CE_1, CE_2, ..., CE_n ovat muodostuneet joukosta työnkulkuja $W = \{W_1, W_2, ..., W_m\}$. Jokainen työnkulku $W_i \in W$ esitetään muodossa $W_i = (N_i, CF_i, DF_i, IC_i, BC_i)$, missä $IC_i = (V_{IC_i}, E_{IC_i}, L_{IC_i})$ ja $BC_i = (V_{BC_i}, E_{BC_i}, CL_{BC_i}, VL_{BC_I})$. Täydellinen suoritushistoria CH on virheetön, jos seuraavat ehdot ovat voimassa:*

1. *Perusrajoitteiden (B) täytyy olla eheitä täydellisen historian suorituksen alkaessa. Eli ensimmäisen aktiviteetin f_{CH} alkaessa.*
2. *Jos $E =< t_j, VS = \{t_k, t_l, ...\}, CL_{BC_i}(E) = \neg B_n, VL_{BC_i}(E) = and/or >$ on särmä perusrajoitegraafissa, ja jos $\neg CL_{BC_i}(E) = B_n$ liittyy toisen aktiviteetin t_X syöterajoitteeseen, tällöin aktiviteetteihin $E(TI_E)$ ja TI_x liittyvät aikavälit (time interval) eivat saa lomittua.*

3. *Jos $E =< t, \{t_k, t_l, ...\}, L_{IC_i}(E) = F >$ on särmä aktiviteettien välisessä rajoitegraafissa ja toinen aktiviteetti t_X rikkoo F:n (Säilytä(t_x, F)=0), tällöin TI_E ja TI_x eivät saa lomittua.*

4. *Jos $E =< t, \{t_k, t_l, ...\}, L_{IC_i}(E) = F >$ on särmä aktiviteettien välisessä rajoitegraafissa ja toinen aktiviteetti t_X ehkä rikkoo F:n (Säilytä($t_x, F) = 1/2$), tällöin F täytyy olla eheä t_X suorituksen loputtua.*

Todistus: *Ohitetaan vaikeana.*□

4 Rajoitteisiin perustuva samanaikaisuusmenetelmä

Tässä kappaleessa esitetään Arpinarin ym. artikkelin pohjalta rajoitteisiin perustuva samanaikaisuudenhallinta (Constraint Based Concurrency Control, CBCC). CBCC takaa teoreeman 3.1 ehdot. CBCC koostuu kolmesta mekanismista:

1. Lukitusvaihe ennen aktiviteetin todellista suoritusta.
2. Varmistusvaihe ennen aktiviteetin todellista päättymistä.
3. Lukkojen vapautusvaihe aktiviteetin päättymisen jälkeen.

Lukkojen yhteensopivuustaulukko on esitetty taulukossa 1, missä K tarkoittaa että lukko voidaan myöntää ja E tarkoittaa konflitktia.

Taulukko 1: Lukkotyyppien yhteensopivuustaulukko.

Pyydetty lukko	Olemassaoleva lukko	
	Jaettu	Poissulkeva
Jaettu	K	E
Poissulkeva	E	K

Akviteettien välinen rajoite F voidaan aktiviteetissa t_x lukita yhdellä seuraavista tavoista:

- **Jaettu:** Tämä lukkotapa pyydetään aktiviteettissa t_x, jos aktiviteetti t_x pyrkii säilyttämään F:n kunnes joukko muita aktiviteetteja päättyy.
- **Poissulkeva:** Tätä lukkotapaa käytetään, jos aktivitetti t_x rikkoo F:n.

Perusrajoite B voidaan aktiviteetissa t_x lukita yhdellä seuraavista tavoista:

- **Jaettu:** Jos t_x vaatii rajoitteen B voimassaoloa, pyydetään jaettu lukko rajoitteeseen B.
- **Poissulkeva:** Jos t_x rikkoo tai ehkä rikkoo B:tä pyydetään yksityinen lukko rajoitteeseen B.

Seuraavassa esitetään Arpinarin ym. alkuperäisartikkelin pohjalta CBCC mentelelmän algoritmit.

Mikä tahansa aktiviteetti t_x tarvitsee poissulkevan lukon jokaiselle aktiviteettien väliselle rajoitteelle, jonka aktiviteetti rikkoo alkaessaan (kohdat 1 ja 2 kuvassa 2). Tämä on mahdollista vain jos mikään muu aktiviteetti ei ole varannut F:ää jaetulla lukolla. Lisäksi aktiviteetti t_x pyytää jaetun lukon kaikkiin perusrajoitteisiin, jotka liittyvät syöterajoitteeseen (kohdat 3 ja 4). Tämän jälkeen jokaiselle aktiviteetin väliselle rajoitteelle, joka on lähtöisin aktiviteetista t_x, pyydetään jaettu lukko (kohdat 5 ja 6). Kohdissa 7–10 pyydetään poissulkeva lukko kaikkiin perusrajoitteisiin, jotka rikotaan aktiviteetissa t_x. Aktiviteettin väliset rajoitteet, jotka ehkä rikotaan hallitaan optimistisella menetelmällä (kriittinen alue kohdassa 11).

```
begin
1.    for ∀ F ∈ F(t_x) do
2.        ExclusiveLock(F);
3.    for ∀ B ∈ B(t_x) do
4.        SharedLock(B);
5.    for ∀ F ∈ C_OUT(t_x) do
6.        SharedLock(F) with Counter = size(S_(t_x,F));
7.    for ∀ B ∈ ( ⋃_VS SB_{t_x,VS,and}) do
8.        ExclusiveLock(B) with Counter = size(VS);
9.    for ∀ B ∈ ( ⋃_VS SB_{t_x,VS,or}) do
10.       ExclusiveLock(B);
<
      ALF(t_x) ← (LF(t_x) ∩ ActiveICS);
11.   PLF(t_x) ← (LF(t_x) − ActiveICS);
      for ∀ F ∈ PLF(t_x) do
          ExclusiveLock(F);
>
end
```

Kuva 2: Aktiviteetin alkuoperaatiot.

Aktiviteetti päättyy kun kaikki operaatiot ovat päättyneet. Ennen lopettamista kuvassa 3 tarkistetaan onko aktiviteettien välisiä rajoitteita rikottu aktiviteetin suorituksessa. Tämä tehdään kohdassa 1 evaluoimalla rajoite $ALF(t_x)$ rinnaikkaisesti. Jos yksikin rajoite on rikottu palautetaan $false$ ja peruutetaan aktiviteetti.

```
begin
1. if (EvalInParallel(ALF(t_x)) = false) then
2.     Rollback(t_x), Resubmit(t_x);
end
```

Kuva 3: Aktiviteetin loppuoperaatiot.

Aktiviteetin t_x loputtua kaikki varatut lukot rajoitteisiin vapautetaan kohdissa 1–6 (kuva 4). Aktiviteettien välisten rajoitteiden lukot vapautetaan kohdissa 7 ja 8. Perusrajoitteiden lukot vapautetaan kohdissa 9–12.

begin
for $\forall\ F \in PLF(t_x)$ **do**
    Unlock(F);
for $\forall\ F \in F(t_x)$ **do**
    Unlock(F);
for $\forall\ B \in B(t_x)$ **do**
    Unlock(B);
for $\forall\ F \in C_{IN}(t_x)$ **do**
    Unlock(F);
for $\forall\ B \in (\bigcup_{t_i} SB_{\{t_i,VS,and\}})$ missä $t_x \in VS$ or $t_x = VS$ **do**
    Unlock(B);
for $\forall\ B \in (\bigcup_{t_i} SB_{\{t_i,VS,or\}})$ missä $t_x = first(VS)$ **do**
    Unlock(B);
end

Kuva 4: Aktiviteetin jälkikäsittely.

On olemassa useampia tapoja toteuttaa rajoitteisiin perustuva samanaikaisuudenhallintamekanismi:

- Pessimistinen. Pessimistisessä menetelmessä selvitetään etukäteen perusrajoitteet ja aktiviteettien väliset rajoitteet, jotka voidaan rikkoa tämän aktiviteetin suorituksessa. Aktiviteetti pyytää lukot aktiviteettien välisiin rajoitteisiin, jotka rikotaan tai ehkä rikotaan aktiviteetissa. Koska tämä pessimistinen menetelmä perustuu lukkoihin, kutsutaan tätä menetelmää *rajoitelukkosamanaikaisuusmenetelmäksi* (Constraint Locking Concurrency Control, CLCC).
- Optimistinen. Optimistisessä menetelmässä aktiviteetit validoivat syöterajoitteet.
- Dynaamisesti pessimistinen. CBCC käyttää tätä menetelmää.

5 Yhteenveto

Työssä esiteltiin alkuperäisartikkeliin pohjautuva menetelmä, jolla avulla työnkulku voidaan matemaattisen täsmällisellä formalismilla esittää. Lisäksi käsiteltiin työnkulkujärjestelmien samanaikaisuudenhallinnan piirteitä. Samanaikaisuudenhallinta on erityisen tärkeää työnkulkujärjestelmissä, joissa vaaditaan eheää kuvaa suoritusympäristöstä. Työnkulkujärjestelmän oikeellisuuskriteerio määritellään perusrajoitteiden ja aktiviteettien välisten rajoitteiden avulla käyttäen työnkulun suunnitteluvaiheen semanttista informaatiota. Rajoitteisiin perustuva samanaikaisuudenhallintamenetelmä perustuu näihin rajoitteisiin.

Viitteet

[AHAD99] İ. B. Arpinar, U. Halici, S. Arpinar, , Doĝac. Formalization of Workflows and Correctness Issues in the Presence of Concurrency. *Distributed and Parallel Databases*, 7(2):199–248, Huhtikuu 1999.

[GHS95] D. Georgakopoulos, M. Hornick, , A. Sheth. An Overview of Workflow Management: From Process Modelling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, Huhtikuu 1995.

[PTV96] J. Puustjärvi, H. Tirri, , J. Veijalainen. Managing Overlapping Transactional Work flows. *Lecture Notes in Computer Science*, 1080, 1996.

[PTV97] J. Puustjärvi, H. Tirri, , J. Veijalainen. Reusability and Modularity in Transactional Workflows. *Information Systems*, 22(2/3):101–120, 1997.

[Puu99] J. Puustjärvi. *Transactional Workflows*. PhD thesis, University of Helsinki, Department of Computer Science, 1999.

[RS94a] M. Rusinkiewicz, A. Sheth. Specification and Execution of Transactional Workflows. W. Kim, editor, *Modern Database Systems: The Object Model, Interoperability, and Beyond*. ACM Press, Cambridge, Massachusetts, 1994.

[RS94b] M. Rusinkiewicz, A. Sheth. Transactional workflow management systems. *Advances in Database and Information Systems, ADBIS'94*, Moscow, Toukokuu 1994.

[WS96] D. Worah, A. Sheth. What do advanced transaction models have to offer for workflows? *Intl. Workshop on Advanced Transaction Models and Architectures*, India, Elokuu 1996.