Keskusmuistitietokantojen samanaikaisuuden hallinta

Ilkka Pullinen

Helsinki 09.03.2012

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta – Fakultet –Faculty	Laitos – Institution – Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä – Författare – Author

Ilkka Pullinen

Työn nimi – Arbetets titel – Title

Keskusmuistitietokantojen samanaikaisuuden hallinta

Oppiaine – Läroämne – Subject

Tietojenkäsittelytiede

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
Seminaarityö	9.3.2012	12 sivua

Tiivistelmä – Referat – Abstract

Levynkäsittelystä aiheutuvien levyviipeiden puuttuessa keskusmuistitietokannoissa, ovat muiden transaktioiden suorituksesta vastaavien komponenttien operointikustannukset nousseet entistä suurempaan merkitykseen. Havainnon johdosta on keskusmuistitietokannoille kehitetty uusia sekä pessimistisiä että optimistisia samanaikaisuuden hallinnan menetelmiä.

Kehitetyt pessimistiset lukituskäytäntöön perustuvat menetelmät ovat kasvattaneet käytetyn lukitusrakeisuuden kokoa sekä sijoittaneet lukitustietojen ylläpidon itse lukittavaan tietoalkioon keskitetyn lukkotietorakenteen sijaan vähentäen lukkojen ylläpidosta aiheutuvia operointikustannuksia. Lukitusrakeisuuden kokoa on kasvatettu aina koko tietokannan lukitsemiseen asti, jolloin transaktioita suoritetaan sarjallisesti ilman tarvetta samanaikaisuuden hallintaan.

Moni transaktioita sarjallisesti suorittava keskusmuistitietokanta hyödyntää tietokannan hajauttamista suorituskyvyn kasvattamiseksi. Tällöin hajautettujen transaktioiden suorituksesta aiheutuvat verkkoviipeet ja sitoutumiskäytännöstä aiheutuvat odotukset voidaan hyödyntää suorittamalla muita transaktioita odotuksien aikana samanaikaisuuden hallinnan avulla.

ACM Computing Classification System (CCS):
H.2.4 [Systems]

Avainsanat – Nyckelord – Keywords

keskusmuistitietokanta, samanaikaisuuden hallinta, hajautettu tietokanta, hajautetun tietokannan spekuloiva samanaikaisuuden hallinta, kaksivaiheinen lukituskäytäntö, kaksivaiheinen sitoutumiskäytäntö

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

Sisältö

1 Johdanto

Levynkäsittelystä aiheutuvien viipeiden poistuessa ja kaiken käsiteltävän tiedon sijaitessa suoraan muistissa suoriutuvat transaktiot entistä nopeammin keskusmuistitietokannoissa nostaen muiden transaktion suorituksesta vastaavien komponenttien operointikustannukset entistä suurempaan merkitykseen [HAM08, KCK02 s. 635]. Yksi näistä komponenteista on samanaikaisuuden hallinta, jolla pyritään sekä sallimaan samanaikaisten transaktioiden suoritus että varmistamaan suorituksessa olevien transaktioiden eristyvyys niiden vaatimalla tasolla ylläpitäen samalla tietokannan eheyttä [SKS11, s. 651, KCK02, s.635]. Samanaikaisuuden hallinnan perustuessa tietoalkioiden lukitukseen, voivat lukkojen hallinnasta ja yhteisten lukkotietorakenteiden salpauskäytännöistä aiheutuvat operointikustannukset nousta merkittävään osaan transaktion käsittelyn kokonaiskustannuksissa keskusmuistitietokannoilla [HAM08, s. 982]. Optimoimalla transaktioiden käsittelyyn osallistuvien tietokantajärjestelmän komponenttien toiminta keskusmuistitietokannoille, voidaan tietokannan suorituskykyä kasvattaa entisestään [HAM08, s. 982]. Havaintojen seurauksena keskusmuistitietokannoille on kehitetty uusia sekä pessimistisiä että optimistisia menetelmiä samanaikaisuuden hallintaan, jotka ottavat paremmin huomioon keskusmuistitietokantojen ominaisuudet [LBD11, JAM10, KCK02].

Tässä seminaarityössä esitetään kaksi keskusmuistitietokannoille kehitettyä samanaikaisuuden hallinnan menetelmää. Ensimmäinen esitettävistä menetelmistä on pessimistinen menetelmä, joka pyrkii kasvattamaan tietokannan suorituskykyä suorittamalla spekulatiivisesti transaktioita hajautetun transaktion sitoutumiskäytännön aikana. Toinen esitettävistä menetelmistä perustuu kaksivaiheiseen lukituskäytäntöön ja on ensimmäisen menetelmän tavoin pessimistinen.

2 Samanaikaisuuden hallinnassa hyödynnettyjä keskusmuistitietokantojen ominaisuuksia

Pessimistisessä lukituskäytäntöön perustuvassa samanaikaisuuden hallinnassa käytetään usein pientä lukitusrakeisuuden kokoa, jotta kilpailua lukittavista tietoalkioista saataisiin vähennettyä mahdollistaen mahdollisimman monen eri tietoalkioita käsittelevän transaktion esteettömän samanaikaisen suorituksen [GaS92, s. 511]. Transaktioiden suoriutuessa keskusmuistitietokannoissa entistä nopeammin, pitävät ne myös entistä lyhyemmän ajan lukkoja hallussaan [GaS92, s. 511]. Lyhentyneen lukitusajan oletetaan vähentävän myös kilpailua lukittavista tietoalkioista, jolloin pieneen lukitusrakeisuuden kokoon ei ole enää tarvetta mahdollistaen lukitusrakeisuuden koon kasvattamisen [GaS92, s. 511]. Kasvattamalla lukitusrakeen kokoa saadaan vähennettyä lukkojen ylläpidosta aiheutuvia operointikustannuksia, mutta koon kasvattaminen heikentää samalla transaktioiden samanaikaisuutta, jolloin kaikkein suurimmat lukitusrakeisuuden koot eivät välttämättä ole asianmukaisia [KCK02, s. 636]. Lukkojen ylläpidosta aiheutuvia operointikustannuksia voidaan vähentää sijoittamalla lukitustiedot itse lukittavaan tietoalkioon kaikille transaktioille yhteisen lukitustietorakenteen sijaan, jolloin esimerkiksi lukkorakenteiden samanaikaisen päivittämisen estävän salpauskäytännön aiheuttamat operaatiokustannukset vähenevät [KCK02, s. 637]. Lukitusrakenteiden sijoitus tietoalkioon kasvattaa kuitenkin tietoalkion kokoa vieden varsinaiselta hyötykuormalta tilaa [KCK02, s. 637]. Kim ja kumppanit [KCK02, s. 636] havaitsivat, että keskusmuistitietokannoissa sopiva lukittavan tietoalkion koko on tietokannan fyysistä sivun kokoa ja sen rakennetta vastaava tietoalkio, jossa tietoalkio sisältää useita monikoita. Esitetyllä lukitusrakeisuuden koolla vältetään matala transaktioiden samanaikaisuuden taso ja lukkojen ylläpidosta aiheutuvat korkeat operointikustannukset tietoalkioissa sijaitsevien lukitustietojen viemän tilan pysyessä samalla riittävän pienenä [KCK02, s. 637]. Suurimmalla rakeisuuden koolla koko tietokanta lukitaan transaktion suoritusta varten johtaen transaktioiden sarjalliseen suoritukseen.

Transaktioiden sarjallisessa suorituksessa ei ole tarvetta samanaikaisuuden hallintaan, jolloin lukituskäytännöstä tai muiden samanaikaisuuden hallintaan käytettyjen transaktioille yhteisten tietorakenteiden salpauskäytännöistä aiheutuvat operointikustannukset poistuvat [GaS92, s. 511]. Sarjallisen suorittamisen suorituskykyä heikentävät kuitenkin pitkäkestoiset transaktiot, jotka estävät lyhytkestoisten transaktioiden tehokkaan suorit-

tamisen niiden joutuessa odottamaan suoritusvuoroaan. Pitkäkestoisia transaktioita ovat esimerkiksi tosiaikaisessa tiedonjalostuksessa (on-line analytic processing, OLAP) esiintyvät useaa monikkoa käsittelevät kyselyt tai käyttäjän kanssa vuorovaikutusta sisältävät transaktiot, joiden kesto on mielivaltaisen pituinen. Pitkäkestoisten transaktioiden ongelma poistuu esimerkiksi sallimalla vain ennalta määriteltyjen lyhytkestoisten tallennettujen proseduurien suoritus asiakkaiden lähettämien transaktioiden sijaan [JAM10, s. 604]. Transaktioiden sarjallisen suorituksen suorituskykyä voidaan kasvattaa hajauttamalla tietokanta usealle itsenäisesti toimivalle pisteelle.

Pisteelle hajautettu tietokannan palanen voidaan edelleen hajauttaa pienemmiksi osioiksi moniprosessoripalvelimen eri prosessoreille, jolloin kukin osio on yhden palvelimen prosessorin hallinnassa [KKN08, s. 1497]. Prosessori toimii tällöin hajautetun tietokannan yhtenä itsenäisenä pisteenä suorittaen sille lähetettyjä transaktioita sarjallisesti muista pisteistä riippumatta [KKN08, s. 1497]. Hajautuksella voidaan parhaimmillaan saavuttaa ympäristö, jossa jokainen transaktio kohdistuu vain yhdelle pisteelle [SMA07 s.1157]. Yleisessä tapauksessa järjestelmässä esiintyy edelleen sekä vain yhden pisteen tietoja käsitteleviä transaktioita että usean pisteen tietoja käsitteleviä hajautettuja transaktioita. Hajautettujen transaktioiden haittana ovat niiden suoritukseen mahdollisesti liittyvät verkkoviipeet [SMA07, s. 983], sitoutumiskäytännöstä aiheutuvat odotukset [JAM10, s. 605] ja muiden transaktioiden suorittamisen estyminen pisteellä kesken olevan alitransaktion päättymiseen asti, kun pisteellä suoritettavilta transaktioilta edellytetään sarjallistuvuutta [JAM10, s. 606-607]. Tietokannan hajautuksen toteuttamiseen on kehitetty automaattisia menetelmiä, kuten Curinon ja kumppanien esittämä Schism [CZJ10], joka analysoi sille syötteenä annettuja tietokannassa suoritettavia kyselyjä ja analysoinnin tuloksen perusteella muodostaa hajautetun tietokannan, jossa syötteenä annetuista kyselyistä hajautuksen seurauksena muodostuvien hajautettujen transaktioiden esiintyminen olisi mahdollisimman vähäistä.

Sarjallinen suoritus ei välttämättä estä pitkäkestoisten transaktioiden suorittamista. Kemperin ja kumppanien esittämä HyPer [KeN11] on hybridi tietokanta, joka sallii pitkäkestoisten, vain lukuoperaatioita sisältävien transaktioiden samanaikaisen suorittamisen sarjallisesti suoritettavien, sekä luku- että päivitysoperaatioita sisältävien, lyhytkestoisten kyselyjen kanssa. Tietokannan toteutuksessa käytetään hyväksi käyttöjärjestelmän virtuaalimuistin käytäntöä, jossa sama muistisivu voi olla jaettuna useiden,

eri transaktioita suorittavien, prosessien kesken [KeN11, s. 198]. Pitkäkestoiset kyselyt operoivat transaktion käynnistyksessä muodostettuun virtuaalimuistin tilannevedokseen, jonka sivut ovat jaettuna käynnissä olevan lyhytkestoisen transaktion kanssa [KeN11, s. 198]. Lyhytkestoisen transaktion päivittäessä jollakin jaetulla sivulla olevaa monikkoa, muodostaa virtuaalimuistinhallinta uuden kopion sivusta johon monikon päivitys tallennetaan. Uusi kopioitu sivu on vain päivittävän transaktion näkyvissä vanhan sivun pysyessä edelleen sitä käsittelevien pitkäkestoisten kyselyjen näkyvissä [KeN11, s. 198].

Moniversioivat samanaikaisuuden hallinnat eivät ole yhtä herkkiä pitkäkestoisille transaktioille kuin lukituskäytäntöön perustuvat pessimistiset menetelmät moniversioivien menetelmien soveltuessa ympäristöihin, joissa esiintyy sekä lyhytkestoisia että pitkäkestoisia, vain lukuoperaatioita sisältäviä, transaktioita [LBD11, s.298 ja 309].

3 Hajautetun tietokannan spekuloiva samanaikaisuuden hallinta

Jonesin ja kumppaneiden [JAM10] esittämässä samanaikaisuuden hallinnan menetelmässä keskusmuistitietokannan oletetaan olevan hajautettuna usealle itsenäisesti toimivalle pisteelle. Pisteelle hajautettu tietokannan palanen on täysin pisteen hallinnassa ja pisteelle lähetetyt transaktiot suoritetaan sarjallisesti [JAM10, s. 604]. Eri pisteille hajautetun tietokannan seurauksena järjestelmässä oletetaan esiintyvän sekä vain yhden pisteen tietoja käsitteleviä transaktioita että usean pisteen tietoja käsitteleviä hajautettuja transaktioita. Hajautettujen transaktioiden sitoutuminen toteutetaan kaksivaiheisella sitoutumiskäytännöllä [JAM10, s. 605]. Sitoutumiskäytännöstä johtuen voi yhdellä pisteellä sitoutumisvaiheeseen päässyt alitransaktio joutua odottamaan lopullisen sitoutumispäätöksen saapumista, jos muut saman hajautetun transaktion alitransaktiot ovat vielä kesken toisilla pisteillä. Myös mahdolliset verkkoviiveet vaikuttavat odotuksen kestoon [JAM10, s. 605]. Menetelmän tavoitteena on käyttää hyödyksi sitoutumiskäytännön päätöksenteosta aiheutuva odotus suorittamalla muita pisteelle lähetettyjä transaktioita spekulatiivisesti odotuksen aikana [JAM10, s. 603-604]. Spekuloidusti suoritettujen transaktioiden oletetaan konfliktoivan muiden pisteellä suoritettujen transaktioiden kanssa, jolloin ei spekuloidusti suoritetun hajautetun transaktion keskeytyessä tulee hajautetun transaktion perumisen lisäksi perua myös spekuloidusti suoritetut trans-

aktiot [JAM10, s. 604]. Peruuntuneet alun perin spekuloidusti suoritetut transaktiot säilytetään pisteellä ja ne suoritetaan aikanaan uudelleen pisteen toimesta [JAM10, s. 604] ilman alkuperäisen lähettäjän tarvetta lähettää transaktiota uudelleen pisteelle suoritukseen.

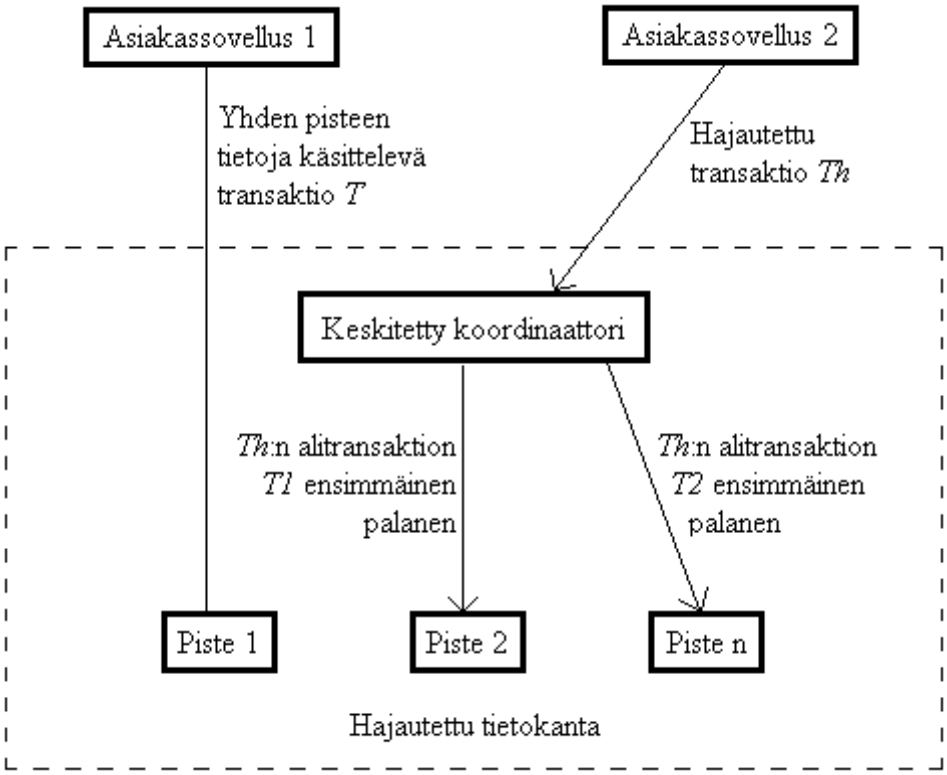
Hajautettujen transaktioiden suoritusta hallinnoi yksi keskitetty koordinaattori, joka huolehtii hajautettujen transaktioiden sarjallistuvasta järjestyksestä ja lukkiutumattomasta suorituksesta määrittämällä hajautetuille transaktioille globaalin järjestyksen [JAM10, s. 605]. Hajautetun transaktion suoritusta varten koordinaattori pilkkoo hajautetut transaktiot pienempiin palasiin, joista yksi palanen sisältää kaikki ne yhden hajautetun transaktion peräkkäiset osat, jotka voidaan suorittaa samalla pisteellä [JAM10, s. 605]. Palasten välillä voi olla tietoriippuvuuksia [JAM10, s. 605], jolloin palasen *P1* riippuessa *P2* tuloksista, ei palasta *P1* voi lähettää suoritukseen ennen palasen *P2* valmistumista. Tietoriippuvuuksista johtuen yhdelle pisteelle voi kohdistua useita saman hajautetun transaktion palasia. Saman hajautetun transaktion *Th* yhdelle pisteelle *i* lähettämät palaset muodostavat pisteellä *i* suoritettavan hajautetun transaktion *Th* alitransaktion *Ti*. Yhdelle pistelle suoritukseen lähetetyt samaan alitransaktioon *Ti* kuuluvat palaset suoritetaan pisteellä palasten saapumisjärjestyksessä [JAM10, s. 605], jolloin palasten keskinäinen järjestys säilyy [3 s.605]. Koordinaattori liittää kaksivaiheisen sitoutumiskäytännön sitoutumiseen valmistautumisviestin (prepare message) viimeiseen pisteelle *i* lähetettävään alitransaktion *Ti* palaseen [JAM10, s. 605].

Hajautettua transaktiota, jonka kaikki palaset voidaan suorittaa samanaikaisesti eri pisteillä, kutsutaan yksinkertaiseksi hajautetuksi transaktioksi [JAM10, s. 607]. Esimerkkinä yksinkertaisesta hajautetusta transaktiosta on transaktio, joka päivittää uuden arvon eri pisteille toisinnettuun monikkoon. Muita hajautettuja transaktioita kutsutaan yleisiksi hajautetuiksi transaktioiksi [JAM10, s. 609].

Tietokannan asiakassovelluksien oletetaan tietävän, miten tietokanta on hajautettuna eri pisteille, jolloin asiakassovellus voi lähettää vain yhden pisteen tietoja käsittelevät transaktiot suoraan sille pisteelle, jonka tietoja transaktio käsittelee [JAM10, s. 604-605]. Hajautetut transaktiot asiakassovelluksen tulee lähettää koordinaattorille [JAM10, s. 605].

Kuvassa 1 on esitetty tietokantajärjestelmän arkkitehtuuri. Kuvassa 1 asiakassovellus 1 on lähettänyt vain pisteen 1 tietoja käsittelevän transaktion *T* suoraan pisteelle 1 suori-

tettavaksi. Asiakassovellus 2 on lähettänyt hajautetun transaktion *Th* keskitetylle koordinaattorille, joka on pilkkonut transaktion *Th* kahteen pisteille 2 ja 3 lähetettyihin alitransaktioihin *T1* ja *T2*.



Kuva 1: Tietokantajärjestelmän arkkitehtuuri [JAM10, s. 605].

Spekuloivaa transaktioiden suoritusta varten joka pisteellä on kaksi jonoa transaktioiden hallintaa varten [JAM10, s. 606]. Ensimmäinen jonoista on suorittamattomien transaktioiden jono (unexecuted transaction queue), joka sisältää pisteelle saapuneet vielä suorittamattomat transaktiot [JAM10, s.606]. Toinen jonoista on sitoutumattomien transaktioiden jono (uncommitted transaction queue), joka sisältää pisteellä jo suoritetut transaktiot, jotka odottavat lupaa saada sitoutua [JAM10, s.606]. Ensimmäinen sitoutumattomien transaktioiden jonossa oleva transaktio on aina ei spekuloidusti suoritettu transaktio *T* [JAM10, s. 606]. Jonossa olevat muut transaktiot ovat aina spekuloidusti suoritettuja transaktioita, joiden sitoutuminen riippuu siitä, sitoutuuko ensimmäinen jonossa oleva transaktio *T* vai ei.

Hajautetun transaktion *Th* alitransaktion *Ti* sanotaan olevan aktiivisena pisteellä, jos se odottaa seuraavaa alitransaktion *Ti* palasta. Hajautetun transaktion *Th* alitransaktion *Ti* sanotaan olevan paikallisesti valmis pisteellä, kun se on suorittanut viimeisen alitransaktion *Ti* palasen ja odottaa lopullista sitoutumispäätöstä koordinaattorilta.

Kun pisteellä ei ole aktiivisena alitransaktiota, suoritetaan pisteelle saapunut transaktion palanen sen päätöksen asti normaalisti ja transaktion tulos palautetaan takaisin sen lähettäjälle [JAM10, s. 606]. Jos pisteelle saapunut transaktio oli jonkin hajautetun transaktion *Th* alitransaktion *Ti* ensimmäinen pisteelle kohdistuneen palanen, on alitransaktio *Ti* nyt aktiivisena pisteellä. Pisteelle saapunut transaktio lisätään suorittamattomien transaktioiden jonoon, jos se ei jatka pisteellä aktiivisena olevaa alitransaktiota *Ti* tai jos pisteellä on jo jokin toinen transaktio suorituksessa [JAM10, s. 607]. Jos pisteelle saapunut transaktio on pisteellä aktiivisena olevan alitransaktion *Ti* seuraava palanen, suoritetaan palanen normaalisti ja palasen tulos palautetaan takaisin koordinaattorille [JAM10, s. 607].

Kun alitransaktio *Ti* on paikallisesti valmis pisteellä, suoritetaan suorittamattomien transaktioiden jonosta transaktioita spekuloiden siinä järjestyksessä, jossa ne ovat jonoon lisätty [JAM10, s. 607]. Spekuloidusti suoritettu transaktio siirretään suorittamattomien transaktioiden jonosta sitoutumattomien transaktioiden jonon häntään. Transaktion spekuloiva suoritus tapahtuu transaktion tyypistä riippuen seuraavasti.

1. Yhteen pisteeseen kohdistuva etätransaktio *T*: Transaktio *T* suoritetaan normaalisti, mutta tulosta ei palauteta transaktion lähettäjälle vaan se puskuroidaan pisteelle [JAM10, s. 606]. Tulos voidaan palautetaan lähettäjälle vasta, kun alitransaktio *Ti* sitoutuu. *T*
2. Yksinkertainen hajautetun transaktion alitransaktio *Tb*: Alitransaktio *Tb* suoritetaan normaalisti ja tulos palautetaan koordinaattorille [JAM10, s. 606]. Tulokseen liitetään tieto, että tulos riippuu hajautetun transaktion *Th* sitoutumispäätöksestä [JAM10, s. 606-607].
3. Yleisen hajautetun transaktio *Thy* alitransaktio *Ty*: Alitransaktiosta *Ty* voidaan suorittaa spekuloiden vain alitransaktion ensimmäinen palanen [JAM10, s. 610]. Alitransaktion *Ty* ensimmäisen palasen tulos puskuroidaan pisteelle. Tulos voidaan palautetaan koordinaattorille vasta, kun alitransaktio *Ti* sitoutuu.

Tapauksessa 1 transaktion *Ti* tulos puskuroidaan pisteelle, sillä palauttaessa se palautuisi suoraan transaktion lähettäjälle, joka ei ole tietoinen transaktion spekuloidusta suorituksesta. Spekuloidusti suoritettu transaktio on voinut lukea alitransaktion kirjoittamia arvoja, jotka alitransaktion *Ti* peruuntuessa eivät olisi eheän tietokannan arvoja.

Transaktion *T* spekuloidun suorituksen päätyttyä piste voi jatkaa muiden jonossa olevien transaktioiden spekuloivaa suoritusta.

Tapauksessa 2 tulos voidaan palauttaa koordinaattorille, sillä yksinkertaisen hajautetun transaktion minkään alitransaktion suoritus ei riipu toisen hajautetun alitransaktion tuloksesta. Tapauksessa 2 koordinaattori voi aloittaa seuraavan hajautetun transaktion käsittelyn ja piste voi jatkaa muiden jonossa olevien transaktioiden spekuloivaa suoritusta. Jos alitransaktio *Ti* peruuntuukin, poistaa koordinaattori puskuristaan spekuloidusti suoritetun alitransaktion *Tb* tuloksen sen suorittaessa hajautetun transaktion *Th* peruutusta, jonka alitransaktio *Ti* oli [JAM10, s. 606-607]. Spekuloidusti suoritettu alitransaktio *Tb* suoritetaan aikanaan uudelleen pisteellä joko normaalisti tai spekuloiden, jolloin koordinaattori saa uuden transaktion *Tb* ajantasaisen tuloksen.

Tapauksessa 3 alitransaktion *Ty* tuloksia ei voida palauttaa, sillä tapauksen 1 tavoin, luetut arvot eivät alitransaktion *Ti* peruuntuessa olisikaan eheän tietokannan arvoja. Jos tulokset palautettaisiin, saattaisi jokin toisella pisteellä ei spekuloidusti suoritettu saman transaktion *Thy* alitransaktio *Thy2* käyttää alitransaktion *Ty* tuloksia operaatioissaan, jolloin koordinaattori joutuisi perumaan koko hajautetun transaktion *Thy* alitransaktion *Ty* spekuloidun suorituksen peruuntuessa. Tapauksen 3 transaktion suorituksen jälkeen piste on estynyt jatkamasta spekuloivaa transaktioiden suoritusta alitransaktion *Ti* sitoutumispäätöksen saapumiseen asti, sillä alitransaktio *Ty* on vielä aktiivinen pisteellä.

Paikallisesti valmiin alitransaktion *Ti* saadessa päätöksen sitouttaa transaktio, sitoutetaan sitoutumattomien transaktioiden jonosta alitransaktio *Ti* sekä tämän jälkeen jonossa olevat kaikki spekuloidusti suoritetut transaktiot ensimmäiseen spekuloidusti suoritettuun toisen hajautetun transaktion alitransaktioon asti [JAM10, s. 607]. Tämän alitransaktion sitouttamispäätös tulee aikanaan koordinaattorilta. Sitouttamispäätöksen ollessa transaktion *Th* keskeytys, peruutetaan ensin spekuloidusti suoritetut transaktiot niiden käänteisessä suoritusjärjestyksessä siirtäen peruttu transaktio sitoutumattomien transaktioiden jonosta takaisin suorittamattomien transaktioiden jonoon aikanaan tapahtuvaa uudelleensuoritusta varten [JAM10, s. 606-607]. Viimeisenä sitoutumattomien transaktioiden jonosta poistetaan alitransaktio *Ti*, joka perutaan normaalin käytännön mukaisesti.

Kun sitoutumattomien transaktioiden jono on tyhjä, jatkaa piste normaalia transaktioiden käsittelyä ilman transaktioiden spekuloivaa suorittamista [JAM10, s. 606].

Esitetty samanaikaisuuden hallinnan menetelmä soveltuu parhaiten ympäristöihin, joissa esiintyy runsaasti yhteen pisteen kohdistuvia transaktioita, joitakin yksinkertaisia hajautettuja transaktioita ja joissa transaktiot keskeytyvät harvoin [JAM10, s. 604, 611]. Yleiset hajautetut transaktiot heikentävät menetelmän suorituskykyä, sillä spekulointia voidaan suorittaa vasta hajautetun transaktion sitoutumisvaiheessa viimeisen hajautetun transaktion alitransaktion palasen saapuessa pisteelle [JAM10, s. 609]. Hajautettujen transaktioiden keskitetty koordinaattori saattaa muodostua pullonkaulaksi hajautettujen transaktioiden määrän kasvaessa [JAM10, s. 605]. Ongelma voidaan poistaa lisäämällä koordinaattorien määrää. Useamman koordinaattorin käyttäminen vaikuttaa spekuloivaan transaktioiden suoritukseen, jolloin tapauksessa 2 voidaan tulos palauttaa vain, jos kaikilla spekuloidusti suoritetuilla hajautettujen transaktioiden alitransaktioilla on sama koordinaattorin koordinoimia, jolloin koordinaattori tietää varsinaisen hajautetun transaktion perumisen yhteydessä poistaa puskurista spekuloidusti suoritetun alitransaktion tuloksen [JAM10, s. 606]. Lähdeartikkelissa ei käsitelty tarkemmin usean koordinaattorin tapausta.

4 Kaksivaiheiseen lukituskäytäntöön perustuva samanaikaisuuden hallinta

Kimin ja kumppaneiden [KCK02] esittämä samanaikaisuuden hallinta perustuu ankaraan kaksivaiheiseen lukituskäytäntöön, jossa transaktion etenemisvaiheessa suorittamia luku- ja kirjoitusoperaatioita varten varataan luku- ja kirjoituslukkoja ja varatut lukot vapautetaan vasta transaktion sitoutuessa [SKS11, s. 667]. Menetelmän tarkoituksena on vähentää lukitustietojen ylläpidosta aiheutuvia operointikustannuksia kasvattamalla lukittavan tietoalkion rakeisuuden kokoa ja ylläpitämällä lukitustietoja lukittavassa tietoalkiossa kaikille transaktioille yhteisen lukkotietorakenteen sijaan [KCK02 s. 635]. Menetelmässä käytetty lukituskäytäntö takaa transaktioiden sarjallistuvan suoritusjärjestyksen [SKS11, s. 667].

Lukittavana tietoalkiona käytetään keskusmuistialkiota, joka tietokantojen fyysisen sivun käsitteen tavoin on kiinteän kokoinen ja sisältää useita monikoita [KCK02 s. 636]. Keskitetyn lukkotietorakenteen sijaan tietoalkioiden lukitustietoja ylläpidetään jokaisessa tietoalkiossa sijaitsevassa lukko-otsakkeeksi (lockHeader) kutsutussa tietorakenteessa [KCK02 s. 637]. Lukko-otsakkeessa ylläpidettäviä lukitustietoja ovat tietoalkion sen

hetkisen aktiivisen lukon tyyppi (grandmode), kaksoislinkitetty lista tietoalkiolle varatuista tai odottavista lukoista (requestQueue) ja tieto, odottaako jokin transaktio lukkoa tietoalkioon (waiting) [KCK02 s. 637]. Lukon tyyppi voi olla joko kirjoitus- tai lukulukko. Lukko on odottava lukko, jos se ei ole yhteensopiva osiolla aktiivisena olevan lukon tyypin kanssa.

Jokaista transaktion pyytämää lukkoa kohti muodostetaan lukkopyynnöksi (lockRequest) kutsuttu tietue [KCK02 s. 637], jossa ylläpidetään pyydetyn lukon tietoja. Tietue sisältää pyydetyn lukon tyypin (grandmode), tilatiedon (status) ja osoittimen lukon omistavan transaktion transaktiotietueeseen aktiivisten transaktioiden taulussa [KCK02 s. 637]. Lukon tilatieto ilmaisee, onko lukko myönnetty (granted), odottava (waiting) tai korottava (converting) [KCK02 s. 637]. Lukko on korottavassa tilassa, kun transaktio haluaa korottaa nykyistä lukulukkoa kirjoituslukoksi [KCK02 s. 637]. Kaikkia saman transaktion muodostamia lukkopyyntöjä säilytetään linkitettynä listana lukkopyyntötietueissa sijaitsevien listalla seuraavaan (next) ja edelliseen (prev) lukkopyyntöön osoittavien osoitinkenttien avulla [KCK02 s. 637]. Listan avulla kaikki saman transaktion lukot voidaan vapauttaa nopeasti transaktion päättyessä [KCK02 s. 637]. Tietoalkion lukko-otsakkeesta alkavaa tietoalkion aktiivisten ja odottavien lukkojen lukkopyynnöistä koostuvaa kaksoislinkitettyä listaa säilytetään transaktion omistuksessa olevien lukkopyyntöjen listan tavoin lukkopyyntötietueissa olevien listalla seuraavaan (requestQueueNext) ja edelliseen (requestQueuePrev) lukkopyyntöön osoittavien osoitinkenttien avulla [KCK02 s. 637].

Transaktion T transaktiotunnisteessa ylläpidetään osoitinta transaktion T muodostamien lukkopyyntöjen linkitetyn listan kärkeen (lockRequestList) ja osoitinta transaktioiden odotuslistaan (lockWaitForList), joka koostuu niiden transaktioiden tunnisteista, joiden lukkojen vapautumista transaktio T odottaa [KCK02 s. 637].

Transaktioiden käsittelemät yhteiset tietorakenteet, kuten tietoalkioissa sijaitsevat lukko-otsakkeet, suojataan salpauskäytännöllä, joka sallii usean samanaikaisen lukijan ja vain yhden kirjoittajan [KCK02 s. 637].

Transaktion operoidessa tietoalkiossa sijaitsevaa monikkoa, lukitaan tietoalkio transaktiolle transaktion suorittaman operaation edellyttämällä lukkotyypillä kaksivaiheisen lukituskäytännön mukaisesti, jolloin yhteen tietoalkioon voi olla samanaikaisesti useita eri transaktioille myönnettyjä lukulukkoja, mutta vain yhdellä transaktiolla kerrallaan

voi olla kirjoituslukko myönnettynä samaan tietoalkioon. Jos transaktion *T* pyytämää lukkoa ei voida heti myöntää tietoalkiolla olevan aktiivisen yhteensopimattoman lukon takia, asetetaan transaktion *T* pyytämän lukon tila odottavaksi, lisätään kaikkien niiden transaktioiden transaktiotunnisteet, joiden lukkojen vapautumista transaktion *T* täytyy odottaa ennen kuin transaktion T pyytämälle lukolle voidaan myöntää lupa tietoalkioon transaktion *T* transaktiotietueessa olevaan transaktioiden odotuslistaan [KCK02 s. 639]. Lopuksi päivitetään tietoalkion lukko-otsakkeeseen tieto tietoalkioon odottavasta transaktiosta ja asetetaan transaktio *T* odottamaan lukon myöntämistä. Haamumonikoiden syntyminen estetään avainvälilukituksessa käytetyn menetelmän tavoin lukitsemalla varsinaisen lukittavan tietoalkion yhteydessä myös tietoalkio, jossa sijaitsee varsinaista operoitavaa monikkoa järjestyksessä seuraavan monikko [KCK02 s. 640].

Kaksivaiheisesta lukituskäytännöstä mahdollisesti aiheutuvat lukkiutumat havaitaan muodostamalla ajoittain transaktiotunnisteissa olevia transaktioiden odotuslistojen avulla aktiivisten transaktioiden odotusverkko (wait-for-graph) ja etsimällä verkosta syklejä [KCK02 s. 639, SKS11 s. 676].

5 Yhteenveto

Levyviipeiden puuttuminen keskusmuistitietokannoissa on johtanut muiden transaktioiden käsittelystä vastaavien komponenttien operointikustannuksien merkityksen kasvuun. Transaktioiden kasvaneet suoritusnopeudet keskusmuistitietokannoissa ovat mahdollistaneet pessimistisissä lukitukseen perustuvissa samanaikaisuuden hallinnan menetelmissä lukitusrakeisuuden koon kasvattamisen vähentäen lukkojen hallinnassa aiheutuvia operointikustannuksia. Samanaikaisuuden hallinasta aiheutuvia operointikustannuksia on saatu vähennettyä edelleen, kun lukitustietojen säilytys keskitetystä lukkotietorakenteesta on siirretty lukittavaan tietoalkioon.

Lukitusrakeisuuden koon kasvattaminen aina tietokannan tasolle asti johtaa transaktioiden sarjalliseen suorittamiseen, jolloin tarve samanaikaisuuden hallintaan poistuu kokonaan poistaen samalla siitä aiheutuvat operointikustannukset. Sarjallisen suorittamisen haittana ovat pitkäkestoiset transaktiot ja pieni rinnakkaisuuden aste moniprosessoripalvelimella. Sarjallisen suorittamisen suorituskykyä voidaan kasvattaa sallimalla vain lyhytkestoisten tallennettujen proseduurien suorittaminen ja hajauttamal-

la tietokanta usealle erilliselle itsenäisesti toimivalle pisteelle, jolloin hajautettujen transaktioiden kaksivaiheisen sitoutumiskäytännön lopullisen sitoutumispäätöksen odotus voidaan käyttää hyödyksi suorittamalla pisteellä muita transaktioita spekuloidusti.

6 Lähteet

CZJ10 C. Curino, Y. Zhang, E. P. C. Jones ja S. Madden. Schism: a workload-driven approach to database replication and partitioning. *Proceedings of the VLDB Endowment*, 3,1 (2010), sivut 48-57.

GaS92 H. Garcia-Molina ja K. Salem. Main memory database systems: An overview. *IEEE Transactions on Knowledge and Data Engineering*, 4,6 (1992), sivut 509-516.

HAM08 S. Harizopoulos, D. J. Abadi, S. Madden ja M. Stonebraker. OLTP through the looking glass, and what we found there. *SIGMOD '08, Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, sivut 981-992.

JAM10 E. P. C. Jones, D. J. Abadi ja S. Madden. Low overhead concurrency control for partitioned main memory databases. *SIGMOD'10, SIGMOD Conference*, 2010, sivut 603-614.

KCK02 S-W. Kim, W. Choi ja B-H. Kim. Design and implementation of the concurrency control manager in the main-memory DBMS Tachyon. *COMPSAC 2002. Proceedings. 26th Annual International Computer Software and Applications Conference*, 2002, sivut 635-641.

KeN11 A. Kemper ja T. Neumann. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. *ICDE 2011, IEEE 27th International Conference on Data Engineering*, 2011. sivut 195-206.

KKN08 R. Kallman, H. Kimura, J. Natkins, A. Pavlo, A. Rasin, S. Zdonik, E. P. C. Jones, S. Madden, M. Stonebraker, Y. Zhang, J. Hugg ja D. J. Abadi. H-store: a high-performance, distributed main memory transaction processing system. *Proceedings of the VLDB Endowment VLDB Endowment*, 1,2 (2008), sivut 1496-1499.

LBD11 P-Å. Larson, S. Blanas, C. Diaconu, C. Freedman, J. M. Patel ja M. Zwilling. High-performance concurrency control mechanisms for main-memory databases. *Proceedings of the VLDB Endowment*, 5,4 (2011), sivut 293-309.

SKS11 A. Silberschatz, H. F. Korth ja S. Sudarshan. *Database System Concepts*. McGraw-Hill, New York, 2011.

SMA07 M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem ja P. Helland. The end of an architectural era: (it's time for a complete rewrite). *VLDB 2007, Proceedings of the 33rd International Conference on Very Large Data Bases*, 2007, sivut 1150-1160.