Jarkko Väyrynen

TIETOKANTATUOTTEIDEN REPLIKOINTIOMINAISUUKSIEN ARVIOINTI. TAPAUS: PROSESSITIEDON VÄLITTÄMINEN JÄRJESTELMIEN VÄLILLÄ

Tietojärjestelmätieteen
pro gradu –tutkielma
12.3.2003

Jyväskylän yliopisto
Tietojenkäsittelytieteiden laitos
Jyväskylä

TIIVISTELMÄ

Väyrynen, Jarkko Tapio
Tietokantatuotteiden replikointiominaisuuksien arviointi. Tapaus: Prosessitiedon välittäminen järjestelmien välillä/Jarkko Väyrynen
Jyväskylä: Jyväskylän yliopisto, 2003.
128 s.
Tutkielma

Tietojen keskittämisestä ja hajauttamisesta päättäminen on eräs keskeisimmistä tietoteknisistä kysymyksistä useissa organisaatioissa. Tietojen hajauttamisella lähelle käyttäjiä tavoitellaan ennen kaikkea parempaa toimintavarmuutta ja suorituskykyä. Tietokannan hallintaohjelmat tarjoavat useita eri vaihtoehtoja hajautuksen toteuttamiseen. Tietojen replikointi eli toisintaminen useaan eri paikkaan on eräs runsaasti huomiota saanut vaihtoehto. Ongelmaksi replikoinnin soveltamisessa kuitenkin muodostuu se, että tietokannan hallintaohjelmien replikointiominaisuuksista on vaikea saada jäsenneltyä tietoa, koska välineet ominaisuuksien luokitteluun ja arviointiin puuttuvat.

Tämän tutkielman ensisijaisena tavoitteena on rakentaa käsitteellinen perusta tietojen hajautuksen ymmärtämiseen ja tietokantatuotteiden replikointiominaisuuksien arviointiin. Toisena tavoitteena on selvittää, voidaanko replikointia soveltaa teollisuuden prosessitietokantojen väliseen tiedonsiirtoon. Sovellusalueella korostuvat etenkin tietokannan hallinta, suorituskyky ja toimintavarmuus.

Tutkielmassa rakennetaan kirjallisuuden pohjalta replikointiominaisuuksien arvioinnin viitekehys, joka ottaa huomioon tietohallinnon, tietokannan hoitajan, sovellusohjelmoijan ja peruskäyttäjän näkökulmat. Viitekehyksellä arvioidaan kahta tietokantatuotetta. Arviointia varten tietokantatuotteilla rakennetaan paperitehtaan tuotannonhallinnan tietokannan prototyypit, joilla testataan replikoinnin hyväksikäytettävyyttä.

Tutkimus osoitti, että kokeiluun valitut tietokantatuotteet soveltuvat suorituskykynsä ja toimintavarmuutensa osalta prosessitietokantojen väliseen tiedonsiirtoon. Replikoinnin hallinta osoittautui tuotteissa kuitenkin varsin monimutkaiseksi. Tutkielmassa laadittu viitekehys havaittiin toimivaksi ja sitä voidaan soveltaa sellaisenaan tietokantatuotteiden replikoinnin arviointiin.

AVAINSANAT: hajautettu tietokanta, replikointi, viitekehys, prosessitietokanta

SISÄLLYS

1 JOHDANTO

1.1 Tausta

Tiedonhallinta on eräs keskeisimmistä tietoteknisistä alueista organisaatioissa. Ihmiset työskentelevät ryhmissä ja vaativat pääsyä tietovarantoihin ajasta ja paikasta riippumatta. Organisaatiot toimivat mahdollisesti useilla paikkakunnilla, eri maissa ja jopa eri maanosissa. Hallittavan tiedon määrä on myös kasvanut ja kasvaa edelleen. Uudetkin teknologiset ratkaisut, esimerkiksi elektroniset kauppapaikat, edellyttävät taustalle vankkaa tukipilaria. Tiedonhallinnan tehtävänä on vastata näihin haasteisiin ja tarjota perusta toiminnan kannalta kriittisten tietojärjestelmien rakentamiselle.

Tiedonhallinta on kokenut suuria muutoksia historiansa aikana. Siirtyminen tiedostojen käytöstä tietokantoihin on ollut merkittävimpiä askeleita. Tietokannat tarjoavat palveluita tiedon tallentamiseen ja hakemiseen. Ratkaisu helpotti tietojen ristiriidattomuuden takaamista ja ohjelmistojen ylläpitoa. Myös kokonaan uusien sovellusalueiden syntyminen on muuttanut tiedonhallintaa. Tietovarastointi (data warehousing), tiedon louhinta (data mining) ja multimedia ovat kaikki alueita, joiden esiinmarssi on edellyttänyt tietokantatuotteiden edelleen kehittämistä tiedon saantitarpeiden ja tietotyyppien muututtua.

Tiedonhallinnan keskeisiin kysymyksiin on jo vuosia kuulunut tietojen keskittämisestä ja hajauttamisesta päättäminen. Useimmilla markkinoilla olevilla tietokantatuotteilla voidaan toteuttaa ratkaisu, jossa tiedot sijaitsevat yhden tietokoneen sijasta useissa tietoliikenneverkon avulla toisiinsa kytketyissä tietokoneissa. Tietojen hajauttamisella lähelle käyttäjiä saavutetaan lukuisia etuja. Ensiksikin tietojen käsittely tehostuu, koska tiedot on sijoitettu lähelle käyttöpaikkaa. Tämä ilmenee lyhyempinä vasteaikoina. Toiseksi hajauttaminen lisää tietojen saatavuutta, koska yhden tietokoneen rikkoontuminen ei estä muiden koneiden käyttäjiä suorittamasta tehtäviään. Ratkaisuna hajautus on luonnollinen, ovathan useat organisaatiot luonnostaan hajautettuja, ainakin

loogisesti osastoihin ja usein myös fyysisesti toimipisteisiin. Tietojen hajautukseen liittyy eduistaan huolimatta myös haittoja, jotka ovat hidastaneet ratkaisun yleistymistä. Hajautettua tietojärjestelmää pidetään keskitettyä tietojärjestelmää vaikeampana hallita ja varmistaa. Hajautetun tietojärjestelmän hallinnan menettelytavat ovat monimutkaisempia kuin keskitetyssä tietojärjestelmässä. Sen vuoksi myös kustannukset ovat suurempia. Käytetyt ohjelmistot ovat laajempia kuin keskitetyssä tiedonhallintaratkaisussa ja näin ollen käytettäviin ohjelmistoihin jää myös todennäköisemmin virheitä.

Hajautettua tietojenkäsittelyä on tutkittu runsaasti erityisesti viimeisen vuosikymmenen aikana. Hajautetussa tietojärjestelmässä on usein erotettu kolme keskeistä komponenttia: 1) tieto, 2) käsittely (liiketoimintasäännöt) ja 3) esittäminen (käyttöliittymä) (esim. Simon, 1996). Näiden komponenttien erilaisia sijoitusvaihtoehtoja on tutkittu (esim. Simon, 1996). Hajautetut tiedostojärjestelmät (esim. Muthitacharoen ym., 2002), tiedon hajautus Internetissä (esim. Yasushi ym., 2001) ja uutena teknologiana esimerkiksi grid-pohjainen tietojenkäsittely (grid computing) (esim. Ranagathan ym., 2002) ovat olleet tutkimuksen kohteina. Myös hajautetun tietokannan ideaalityyppi, jossa tiedot sijoitetaan eri tietokantoihin hajautuksen silti näkymättä käyttäjälle, on kiehtonut tutkijoita jo kauan (mm. Stonebraker ym., 1983; Özsu ym., 1994b; Reddy ym., 1998). Hajautukseen liittyvää tutkimusta on aktiivisuudestaan huolimatta arvosteltu siitä, että tuloksia on ollut vaikea hyödyntää yli tutkimusalueiden rajojen (Wiesmann, 2000b).

Replikointi on eräs hajautusvaihtoehto. Sillä tarkoitetaan tietojen toisintamista useaan paikkaan. Replikoinnilla tavoitellaan tietojärjestelmiltä ennen kaikkea parempaa saatavuutta ja suorituskykyä. Saatavuuden paraneminen ilmenee siten, että yhden tietokopion käytön estyminen ei välttämättä estä muiden kopioiden käyttöä. Suorituskyvyn osalta replikoinnilla voidaan saavuttaa etuja, jos kyselyt ohjataan paikallisiin tietokopioihin kenties kaukana olevien muiden kopioiden sijaan. Etujensa vuoksi replikoinnista on tullut merkittävä tekniikka tietojärjestelmien toteutuksessa. Internetin nimipalvelimet (name server), sähköpostipalvelimet ja dokumenttien hallintajärjestelmät replikoivat (Watterson, 1996). Erityisen välttämättömäksi replikointi

on osoittautunut mobiiliympäristössä (esim. Ratner ym., 1997; Lubinski ym., 2000). Satunnaisesti tietoliikenneverkossa kiinni olevia ja tiedonsiirtokapasiteetiltaan rajallisia päätelaitteita voidaan käyttää tehokkaasti replikoinnin avulla.

Replikointi otettiin käyttöön kaupallisissa tietokantatuotteissa 1990-luvun aikana (UniF/X Inc., 2000). Tietokantojen replikoinnin erääksi tavoitteeksi on asetettu replikoinnin näkymättömyys käyttäjälle (Date, 1995). Käyttäjän tulisi voida käyttää tietokantaa ikään kuin replikointia ei käytettäisikään. Tämän tavoitteen täyttämisessä törmätään kuitenkin lukuisiin kysymyksiin. Kuinka käyttäjän tekemät muutokset välitetään muihin tietokopioihin? Miten menetellään tietoliikenneverkon vikaantuessa tai muissa teknisissä ongelmissa? Halutaanko muutosten välittyvän muihin kopioihin heti vai vasta tietyn ajan kuluttua? Sallitaanko käyttäjän ylipäänsä päivittää omaa tietokopiotaan? Miten menetellään eri käyttäjien päivittäessä yhtä aikaa eri tietokopioita? Vastaako tietokannan hallintajärjestelmä muutosten välittämisestä vai toteutetaanko päivityslogiikka sovellusohjelmaan? Hajautettujen tietokantojen tutkimus on pyrkinyt löytämään vastauksia näihin haasteellisiksi osoittautuneisiin kysymyksiin. Tuloksena on syntynyt lukuisia menetelmiä ja protokollia replikoinnin hallintaan (esim. Little ym., 1994; Theel, 1998; Breitbart ym., 1999; Saito, 2001) Osa tutkimuksen tuloksista on otettu mukaan myös kaupallisiin tietokannan hallintajärjestelmiin. Ongelmalliseksi tietokannan hoitajalle ja sovellusohjelmoijalle voi kuitenkin muodostua se, ettei ole olemassa juurikaan välineistöä tietokantatuotteiden replikoinnin arviointiin. Myös tuotteiden vertailu on hankalaa, sillä tietokantatoimittajat käyttävät usein hyvin erilaista terminologiaa.

1.2 Tutkimuksen tavoitteet ja tutkimusmenetelmät

Tässä tutkimuksessa tarkastellaan tietokannan hajautusta ja replikointia. Tutkimus-ongelmat voidaan kiteyttää seuraaviin kysymyksiin:

1) mitä tarkoitetaan tietokannan hajautuksella ja mitä erityisesti tietokannan replikoinnilla tarkoitetaan?
2) miten tietokantatuotteiden replikointiominaisuuksia tulisi arvioida?

Tutkimuksen tavoitteena on :

1) luoda käsitteellinen perusta tietojen hajautuksen, hajautettujen tietokantojen ja replikoinnin käsitteiden, rakenteiden ja periaatteiden määrittelemiseksi,
2) rakentaa viitekehys, jonka avulla voidaan arvioida tietokantatuotteiden replikointi-ominaisuuksia, ja
3) testata viitekehyksen toimivuutta käytännön tilanteessa.

Kahden ensimmäisen tavoitteen saavuttamiseksi sovelletaan käsitteellistä tutkimusta, jonka tuloksena saadaan:

- esitys hajautettujen tietojärjestelmien komponenteista
- jaottelu tiedon hajautusvaihtoehdoista ja arvioita vaihtoehdon valinnasta
- kuvaus hajautetuista tietokannoista ja hajautetusta tietokannan hallintajärjestelmästä
- kuvaus hajautetusta tapahtumankäsittelystä
- tietokannan hallintajärjestelmien replikointiominaisuuksien arvioinnin viitekehys.

Viitekehyksen toimivuuden testaamiseksi suoritetaan tutkimuksessa käytännön kokeilu. Sitä varten valitaan kaksi tietokannan hallintajärjestelmää, joilla toteutetaan paperitehtaan tuotannon prosessitietokantaa jäljittelevä prototyyppi. Prosessitietokannalla tarkoitetaan lähinnä mittaustietoja sisältävää tietokantaa, joka on toteutettu tietystä teollisuusprosessista vastaavien henkilöiden käyttöön. Paperiteollisuudessa tällaisia prosesseja ovat esimerkiksi massan valmistus ja jätevesien puhdistus. Myös eri prosessitietokantojen välillä on usein tarvetta siirtää tietoja. Tiedonsiirto on yleensä toteutettu sanomanvälitysjärjestelmillä tai tiedostopohjaisesti. Näissä tiedonsiirtotavoissa on kuitenkin ongelmana staattisuus eli tiedonsiirtotarpeiden muuttuessa joudutaan yleensä muuttamaan sovellusohjelmia. Tutkimuksessa pyritäänkin selvittämään, voidaanko tietokannan hallintajärjestelmien replikointiominaisuutta käyttää tiedonsiirtovaihtoehtona teollisuuden prosessitietokantojen välillä. Huomiota kiinnitetään etenkin replikoinnin hallintaan, suorituskykyyn ja toimintaan poikkeustilanteissa. Prototyyppiä rakennettaessa arvioidaan replikoinnin hallintaa valituissa tuotteissa laaditun viitekehyksen avulla. Prototyypillä suoritetaan koeajoja, joilla pyritään selvittämään replikoinnin suorituskykyä ja toimintaa poikkeustilanteissa.

Kokeellisella simuloinnilla voidaan tutkia järjestelmän tulevaa tilaa etenkin silloin, kun tulevan tilan selvittäminen muilla tutkimusmenetelmillä olisi hankalaa (Galliers, 1991). Tässä tutkimuksessa laadittava prototyyppi suurine päivitysintensiteetteineen edustaa tuota tulevaa tilaa. Tuloksina saadaan:

- kahden tietokannan hallintajärjestelmän replikoinnin arviointi
- arvio replikoinnin soveltuvuudesta prosessitietokantojen yhteyteen.

1.3 Tutkielman rakenne

Tutkielma on jäsennetty viiteen lukuun. Luvussa 2 määritellään hajautetun tietojenkäsittelyn keskeisimpiä käsitteitä. Luvun alussa hajautetun tietojärjestelmän komponentit esitellään lyhyesti. Tämän jälkeen keskitytään hajautettuihin tietokantoihin, pohditaan hajautuksen etuja ja haittoja sekä esitetään tiedon hajauttamisen vaihtoehdot. Luvun lopuksi tarkastellaan hajautettua tietokannan hallintajärjestelmää ja hajautettua tapahtumankäsittelyä.

Luvussa 3 esitetään tietokantatuotteiden replikointiominaisuuksien arvioinnin viitekehys. Luvun alussa täsmennetään replikointiominaisuutta ja pohditaan, kuinka ominaisuutta voidaan arvioida. Sen jälkeen kuvataan kehyksen soveltamista eri näkökulmista. Lopuksi esitetään viitekehyksen rakenne yksityiskohtaisine arviointikriteereineen.

Luvussa 4 sovelletaan replikointiominaisuuksien arvioinnin viitekehystä. Tutkimuksen toimeksiantajan osoittamassa käytännön kokeilussa vertaillaan kahta tietokantatuotetta ja esitetään arvioita replikoinnin soveltuvuudesta toimeksiantajan käyttöön.

Luvussa 5 esitetään yhteenveto tutkimuksen tuloksista ja ehdotetaan jatkotutkimus-aiheita.

2 HAJAUTETTU TIETOJENKÄSITTELY

Tässä luvussa määritellään hajautetun tietojenkäsittelyn keskeisimpiä käsitteitä. Aluksi tarkastellaan lyhyesti hajautetun tietojärjestelmän komponentteja - tietoa, käsittelyä ja esittämistä sekä esitetään näiden komponenttien erilaiset sijoitustavat. Kohdassa 2.2 käsitellään hajautettuja tietokantoja, ja kohdassa 2.3 tarkastellaan erilaisia tiedon sijoittamisvaihtoehtoja tietokantoja käytettäessä. Hajautettu tietokannan hallinta-järjestelmä kuvataan kohdassa 2.4. Luvun keskeisintä teemaa, hajautettua tapahtuman-käsittelyä, tarkastellaan kohdassa 2.5. Yhteenveto luvusta esitetään kohdassa 2.6.

2.1 Hajautetun tietojärjestelmän komponentit

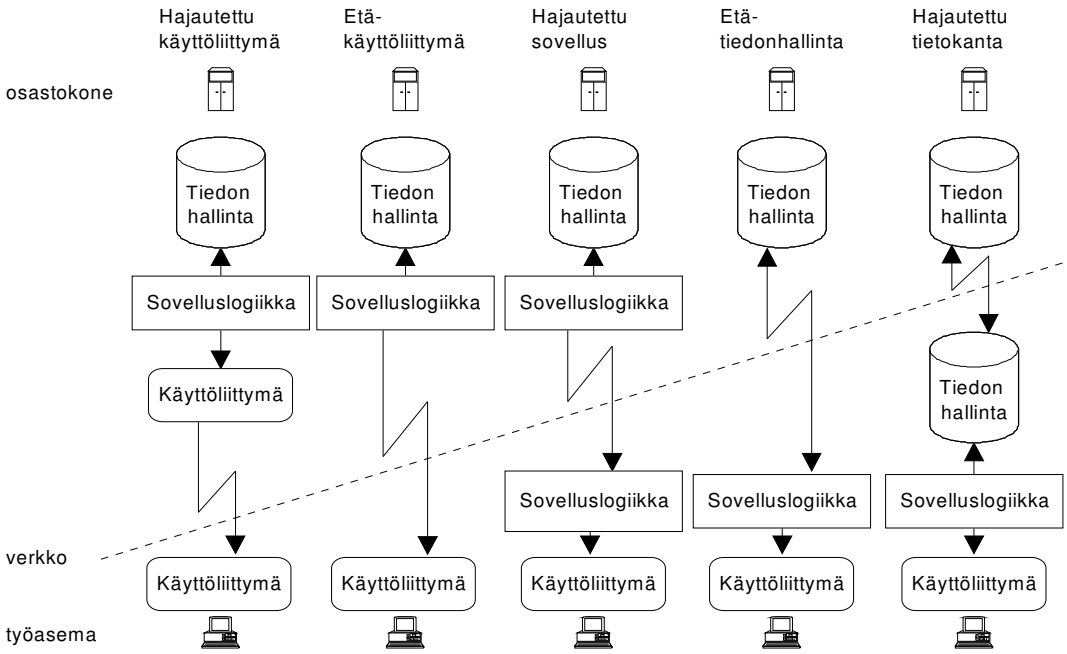
Hajautettua tietojärjestelmää suunniteltaessa joudutaan päättämään seuraavan kolmen komponentin keskittämisestä tai hajauttamisesta (Simon, 1996, s. 37).

Komponentit ovat:

- tieto
  Tieto-objekteja ovat esimerkiksi relaatiotietokanta ja sen kaava.
- käsittely
  Tieto-objektien käsittelyä ovat esimerkiksi katseluohjelmat, selaimet, hakukoneet ja sovelluslogiikka.
- esittäminen
  Tieto-objektien esittämisen lisäksi tähän kuuluu myös käyttäjän syötteiden käsittely.

Esittämisen, käsittelyn ja tiedon hajautuksesta on yleisimmin esitetty Gartner Groupin laatima malli (Semich, 1994). Esittäminen tarkoittaa sitä, miltä sovellus näyttää

käyttöliittymältään. Käsittelyyn kuuluvat organisaation liiketoimintasäännöt, jotka koodataan sovelluslogiikaksi. Tiedon hallinnalla tarkoitetaan toimintoja, joilla hoidetaan tietojen tallentaminen levylle ja haku levyltä. Tässä tutkimuksessa keskitytään tarkastelemaan hajautettua tietokantaa. Kuvassa 1 hajautuksen vaihtoehdot on esimerkinomaisesti sijoitettu osastokoneeseen ja työasemaan.



Kuva 1. Esittämisen, käsittelyn ja tiedon hajauttamisen vaihtoehdot (Semich, 1994).

Esitetyistä vaihtoehdoista käytetään tällä hetkellä eniten etätiedonhallintaa, hajautettua sovellusta, etäkäyttöliittymää ja jonkin verran hajautettuja tietokantoja. Etätiedonhallinta edustaa perinteistä työasemasovellusta, jossa ohjelmaa suoritetaan työasemassa, mutta tiedonhallinta on sijoitettu osastokoneeseen. Hajautetussa sovelluksessa osa sovelluslogiikasta on sijoitettu osastokoneeseen esimerkiksi tietokantaproseduurien ja herätteiden muodossa. Selainpohjainen sovellusratkaisu on esimerkki etäkäyttöliittymästä. Hajautettua tietokantaa voidaan käyttää esimerkiksi mobiiliratkaisussa, jossa hajautettu tiedonhallinta synkronoi kannettavan työaseman ja osastokoneen tiedot verkkoyhteyden päällä ollessa.

Hajautetulla tietojärjestelmällä tarkoitetaan ratkaisua, jossa edellä esitetyt komponentit (tieto, käsittely ja esittäminen) on sijoitettu vaihtelevasti verkon eri tietokoneisiin. Lisäksi ratkaisuun sisältyy olennaisena osana tiedon siirtoa tietoliikenneverkossa.

2.2 Hajautetut tietokannat

Tarkastellaan seuraavaksi hajautettuja tietokantoja yleisellä tasolla, ottamatta kantaa olemassa oleviin tietokantatuotteisiin. Aluksi määritellään hajautettu tietokanta. Tietokantoihin liitettyjä keskeisiä periaatteita pohditaan niin hajautettujen kuin keskitettyjenkin tietokantojen osalta. Hajautettujen tietokantojen etujen ja haittojen tarkastelun jälkeen esitetään lopuksi erilaisia tiedon sijoittamisvaihtoehtoja.

2.2.1 Hajautetun tietokannan määritelmä

Hajautetun tietokannan käsitteestä näyttää olevan hyvin erilaisia käsityksiä, jotka ovat muuttuneet tietokantateknologian kehittyessä vuosien kuluessa (vrt. Ceri, 1985, s. 1; Ozkarahan, 1986, s. 410; Korth, 1991, s. 473; Date, 1995, s. 593; Özsu ja Valduriez, 1996a, s. 1, Hoffer ym., 2002). Özsu ja Valduriez (1994b) ovat korostaneet, että hajautettu tietokanta ei ole hajautettu tiedostojärjestelmä eikä yhdessä verkon tietokoneessa sijaitseva tietokantajärjestelmä, jota käytetään muista tietokoneista. Tässä tutkimuksessa hajautettu tietokanta määritellään seuraavasti:

Hajautettu tietokanta on kokoelma loogisesti toisiinsa liittyviä tietokantoja, jotka on hajautettu tietokoneverkkoon (Özsu ym., 1996b, s. 1).

Hajautetulla tietokannan hallintajärjestelmällä tarkoitetaan ohjelmistoa, jolla hajautettua tietokantaa ylläpidetään ja jonka avulla tietoa voidaan käsitellä siten, ettei

hajautus näy käyttäjälle (Özsu ym., 1996b, s. 1).

Hajautetun tietokannan määritelmästä ei ilmene, kuinka hajautettu tietokanta on käytännössä toteutettu, joten täsmennetään määritelmää hiukan.

1. Tieto on sijoitettu useisiin tietokoneisiin.

   Tässä tutkimuksessa yhden tietokoneen, siinä toimivan hajautetun tietokannan hallintajärjestelmän ja siinä sijaitsevan hajautetun tietokannan osan muodostamaa kokonaisuutta kutsutaan *tietokantasolmuksi*[1]. Tietokoneet voivat poiketa toisistaan arkkitehtuuriltaan, ja joissakin koneissa voidaan hyödyntää moniprosessori-arkkitehtuuria. Olennaista kuitenkin on, että kukin tietokantasolmu sisältää loogisessa mielessä yhden prosessorin.
2. Tietokoneet on kytketty toisiinsa tietoliikenneverkon välityksellä.

   Tämä ehto sulkee pois tapaukset, joissa eri tietokoneet käyttäisivät jaettua muistia. Tietoliikenneverkon toteutustapaan ei oteta kantaa. Verkko voi olla esimerkiksi lähiverkko (LAN) tai alueverkko (WAN).
3. Looginen yhteenkuuluvuus.

   Tällä tarkoitetaan sitä, että tiedoilla on jokin yhteinen yhdistävä tekijä. Kyseessä ei siis ole vain joukko tietokantoja, jotka on sijoitettu verkon eri tietokoneisiin.

Özsu ym. (1996a, s. 4) on luonnehtinut tietokantaa käyttävän asiakas/palvelin -arkkitehtuurin suhdetta hajautettuihin tietokantoihin seuraavasti. Yksinkertaisimmillaan asiakas/palvelin -arkkitehtuurissa asiakkaat lähettävät palvelupyyntöjä yhdelle tietokantapalvelimelle. Hieman monimutkaisemmassa ja joustavammassa ratkaisussa on useita asiakkaita ja palvelimia ja tietokanta sijaitsee useilla palvelimilla. Asiakkaat lähettävät palvelupyyntöjä omalle ensisijaiselle palvelimelleen. Palvelinten välinen

[1] Hajautettujen tietokantojen termistön osalta näyttää vallitsevan epäselvyys siitä, mitä termiä hajautetun tietokannan hallintajärjestelmän ja paikallisen tietokannan sisältävästä tietokoneesta tulisi käyttää. Englanninkielisissä tutkimuksissa käytetään yleisimmin termiä ”site”, joskin myös termejä ”node” ja ”computer” käytetään asiayhteyden mukaan (Korth, 1991). Suomenkielisinä käännöksinä ovat esittäneet Huttunen (1996) ”tietokantasolmu”, Vuoniemi (1990) ”solmu” ja Lamminmäki ja Hannus (1993) ”paikka” ja ”piste”.

tietoliikenne ei näy käyttäjälle tietokantakyselyjä ja tietokantatapahtumia suoritettaessa. Useimmat käytössä olevat asiakas/palvelin -ratkaisut perustuvat näihin kahteen vaihtoehtoon. Varsinaisessa hajautetussa tietokantaratkaisussa jokainen tietokantasolmu voi toimia sekä asiakkaana että palvelimena. Tästä käytetään nimitystä tasavertaisuus (peer-to-peer). Ratkaisun monimutkaisuus on kuitenkin viivästyttänyt aidosti hajautettujen tietokannan hallintajärjestelmien kehittämistä.

2.2.2 Hajautettu ja keskitetty tietokanta - vertailua

Vertaillaan lyhyesti hajautettuja ja keskitettyjä tietokantoja muutamien keskeisten tietokantoihin liittyvien tekijöiden osalta.

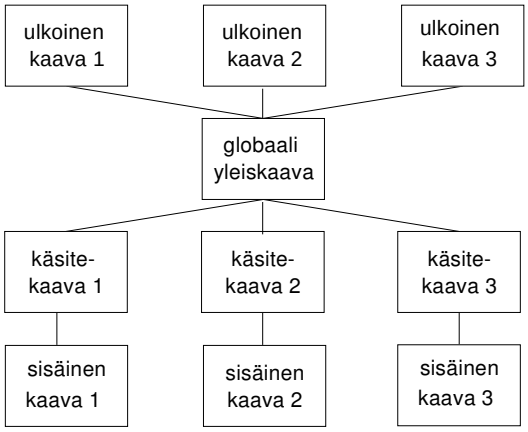
Tietokannan hallinta

Keskitetyissä tietokannoissa tietokannan hallinta on yleensä annettu yhden tietokannan hoitajan vastuulle. Hänen tehtäviinsä kuuluvat mm. tietohakemiston ylläpito, tietojen käytön valvonta, toipumismenettelyjen ja fyysisen tietokannan luonti, peruskäyttäjien neuvonta, suorituskyvyn optimointi, tietokantaohjelmiston päivittäminen ja tietokanta-teknologian kehityksen seuraaminen. Hajautettujen tietokantojen hoito voidaan antaa paikallisten tietokantojen hoitajille joko kokonaan tai osittain.

Tietoriippumattomuus

Eräs tietokantojen alkuperäinen tavoite oli tietoriippumattomuuden tarjoaminen. Fyysisellä tietoriippumattomuudella tarkoitetaan sitä, ettei tietojen tallennusrakenteen muuttaminen aiheuta muutoksia näitä tietoja käsittelevissä sovellusohjelmissa. Loogisella tietoriippumattomuudella tarkoitetaan puolestaan sitä, ettei tietokannan loogisen rakenteen muuttaminen (esimerkiksi sarakkeen lisääminen relaatioon) aiheuta muutoksia näitä tietoja käsittelevissä sovellusohjelmissa. Keskitetyissä tietokannoissa

tietoriippumattomuuteen pyritään ANSI/SPARC:n kolmitasoisella jaottelulla ulkoiseen kaavaan, käsitekaavaan ja sisäiseen kaavaan. Hajautetuissa tietokannoissa pyritään niin ikään tietoriippumattomuuteen. Kuvan 2 esimerkissä ANSI/SPARC:n arkkitehtuuria laajennetaan hajautettuun ympäristöön (Özsu ym., 1991, s. 83). Tietokannan fyysinen tallennusrakenne voi olla erilainen eri tietokantasolmuissa. Sen vuoksi kussakin tietokantasolmussa täytyy olla oma sisäinen kaava. Kunkin tietokantasolmun tietokannan looginen rakenne esitetään käsitekaavassa. Organisaation globaali käsitekaava koostuu paikallisista käsitekaavoista. Tietokannan käyttöä tuetaan ulkoisilla kaavoilla.



Kuva 2. Hajautetun tietokannan kaavioarkkitehtuuri (Özsu ym., 1991, s. 83).

Tietoylimäärä

Keskitetyissä tietokannoissa pyritään selkeästi välttämään tietoylimäärää. Saman tiedon tallentaminen useaan paikkaan lisää virhemahdollisuutta päivitysten yhteydessä. Hajautetuissa tietokannoissa tietoylimäärä voi olla sen sijaan hyvinkin suotavaa. Saman tiedon sijoittaminen useisiin eri tietokantasolmuihin lisää tehokkuutta, sillä sovellusten ei tarvitse hakea tietoa verkon yli toisesta tietokoneesta (Simon, 1996, s. 258). Järjestelmä on myös todennäköisemmin käytettävissä tietyllä hetkellä, sillä yhden tietokoneen rikkoontuminen ei estä muita jatkamasta toimintaansa (Simon, 1996, s. 257). Optimaalisen tietoylimäärän määrittäminen on kuitenkin monimutkainen tehtävä, sillä

huomioon on otettava tiedon paikasta toiseen siirtämiseen, sen tallentamiseen ja päivitykseen liittyviä tekijöitä.

Tallennusrakenteet

Keskitettyjen tietokantojen suorituskykyyn voidaan vaikuttaa muuttamalla tietokannan fyysistä tallennusrakennetta esimerkiksi peräkkäisrakenteesta hajarakenteeseen. Hajautettujen tietokantojen suorituskyvyn parantamisessa verkkoliikenteen vähentäminen on tallennusrakenteen muuttamista tärkeämmässä asemassa, sillä tietoliikenneverkkojen tiedonsiirtonopeus on, ehkä nopeimpia lähiverkkoja lukuun ottamatta, huomattavasti pienempi kuin levyasemissa.

Eheys, toipuminen ja samanaikaisuuden hallinta

Tietojenkäsittelyssä käytetyt laitteet ja ohjelmistot ovat alttiina lukuisille virheille. Ohjelmavirheet, laiteviat, sähköhäiriöt, käyttöjärjestelmävirheet jne. uhkaavat jättää tietovaraston tiedot epäyhtenäiseen ja ristiriitaiseen tilaan. Myös yhtäaikaisesti suoritettavat päivitykset voivat aiheuttaa epäyhtenäisyyttä. Eheys, toipuminen ja samanaikaisuuden hallinta liittyvät hyvin läheisesti toisiinsa.

Tietokanta on eheässä ja ristiriidattomassa tilassa, jos se täyttää sille asetetut eheysrajoitteet (esim. ikä välillä 0-130 vuotta). Tietokanta pyritään pitämään eheänä jakamattomien tietokantatapahtumien avulla. Jakamattomuuden mukaan joko kaikki tapahtuman sisältämät päivitystoimenpiteet viedään tietokantaan tai virhetilanteessa tietokanta palautetaan tapahtuman aloitusta edeltäneeseen tilaan. Keskitetyissä tietokannoissa jakamattomuuden takaaminen on hajautettua tietokantaa yksinkertaisempaa, sillä kaikki tieto sijaitsee yhden tietokannan hallintajärjestelmän alaisuudessa. Hajautetuissa tietokannoissa yksi tietokantatapahtuma voi aiheuttaa toimenpiteitä useissa eri tietokantasolmuissa ja mikäli jokin näistä solmuista tai tietoliikenneverkko on vikaantunut, kaikki toimenpiteet täytyy peruuttaa. Tällaisten hajautettujen

tietokantatapahtumien käsittelemiseksi on kehitetty lukuisia tapahtumanvahvistus-protokollia (Korth, 1991, s. 494; Khoshafian, Chan ym., 1991, s. 514).

Toipumisessa keskeneräiset tietokantatapahtumat peruutetaan ja jo hyväksytyt tietokantatapahtumat suoritetaan uudelleen. Palautuksessa käytetään niin keskitetyissä kuin hajautetuissakin tietokannoissa lokeja. Hajautetuissa tietokannoissa toipumisen monimutkaisuutta lisää mahdollisten useiden tietokopioiden olemassaolo (Korth, 1991, s. 494). Kun vikaantunut tietokantasolmu toipuu, sen täytyy saattaa tietokopionsa ajantasalle ennen kuin niiden käyttö sallitaan (Khoshafian ym., 1991, s. 518).

Samanaikaisuuden hallinnan päämääränä on taata, etteivät useat samaan aikaan suoritettavat tietokantatapahtumat häiritse toisiaan. Lukitus on yleisimmin käytetty samanaikaisuuden hallinnan keino. Lukituksessa käsiteltävä tietokanta tai sen osa eristetään tietokantatapahtuman ajaksi muiden ohjelmien käytöstä. Jos useat tietokantatapahtumat lukitsevat saman kohteen, puhutaan lukkiumasta (deadlock). Keskitetyissä tietokannoissa käytetään yleisesti lukkiumien tunnistamista ja niiden purkamista. Hajautetuissa tietokannoissa tilanne on monimutkaisempi, sillä niissä täytyy päättää, keskitetäänkö lukitusinformaatio yhteen tietokantasolmuun vai hajautetaanko se verkon eri puolille.

Kyselyjen käsittely

Käyttäjän suorittamien tietokantakyselyjen käsittelyssä kyselyjen optimointi on keskeisessä asemassa niin keskitetyissä kuin hajautetuissakin tietokannan hallintajärjestelmissä. Keskitetyissä tietokannoissa levyoperaatioiden määrä on ensisijainen kriteeri, jolla kyselyjen suoritusvaihtoehtoja arvioidaan (Korth, 1991, s. 487). Hajautettuja tietokantoja käytettäessä optimoinnissa on huomioitava tiedon siirrosta aiheutuva verkkoliikenne ja kyselyjen rinnakkaiskäsittelystä eri tietokantasolmuissa mahdollisesti saatava hyöty.

Tietokannan suunnittelu

Hajautettuja tietokantoja suunniteltaessa joudutaan ratkaisemaan keskitettyihin tietokantoihin verrattuna monia lisäkysymyksiä. Osittamisen suunnittelussa päätetään, kuinka globaalit relaatiot jaetaan osiin. Osien sijoittamisen suunnittelussa ratkaistaan, mihin tietokantasolmuun kukin osa sijoitetaan. Optimaalisen hajautuksen suunnittelu on monimutkaisempaa, mikäli päädytään käyttämään useita tietokopioita (Özsu ym., 1991, s. 144).

2.2.3 Hajautetun tietokannan edut

Seuraavassa kuvataan muutamia hajautettujen tietokantojen etuja. Aihetta on käsitelty varsin laajasti tietokantakirjallisuudessa (esim. Korth, 1991; Pratt, 1991; Date, 1995; Özsu ym., 1996a). Tämän kohdan tarkoituksena on selventää aihetta yhteenvedonomaisesti.

Organisationaaliset ja taloudelliset syyt

Hajautetut tietokannat sopivat useisiin organisaatioihin siksi, että organisaatiot ovat luonnostaan hajautettuja. Organisaatioilla on usein eri toimipisteitä ja ne koostuvat osastoista. Organisaatiojaon vuoksi myös sen käsittelemät tiedot on hajautettu. Kukin osasto käsittelee omaa toimintaansa koskevia tietoja, ja sen vuoksi hajautettu tietokanta voi noudattaa organisaation rakenteen muotoa (Date, 1995, s. 595). Tietojen saantiajat myös pienenevät, kun tiedot on sijoitettu lähelle käyttäjiä (Simon, 1996, s. 257). Hajautetut tietokannat tukevat myös organisaation kasvamista. Kun uusi organisaatioyksikkö lisätään tietokantajärjestelmään, järjestelmän olemassa oleviin osiin ei heijastu suuria vaikutuksia.

Özsun ym. (1996a, s. 4) mukaan hajautettua tietokantaratkaisua voidaan pitää laitteiston

osalta keskitettyä ratkaisua edullisempana. Tietokantasolmuissa käytettävät pienet tietokoneet ovat halvempia kuin suuret keskuskoneet. Tätä seikkaa ei pidä kuitenkaan ottaa itsestäänselvyytenä. Hajautettuja tietojärjestelmiä toteutettaessa eri komponentit hankitaan yleensä eri toimittajilta. Toimittajilla ei puolestaan ole välttämättä tietoa tuotteensa soveltuvuudesta jonkin tietyn tuotteen yhteyteen ja näin ollen yhteensovittaminen jää ostajan vastuulle. Yhteensovittamistyötä vaikeuttaa ongelmien hankala paikantaminen (Simon, 1996, s. 14). Laitteistohankinnoissa tehdyt säästöt voivat myös tulla nopeasti syödyiksi, mikäli järjestelmänhallintaan ei kiinnitetä huomiota (Lamminmäki ym., 1993, s. 158).

Olemassa olevien tietokantojen yhteenliittäminen

Organisaatioissa on tyypillisesti lukuisia tietokantoja eri puolille sijoitettuina. Nämä tietokannat pohjautuvat usein eri tietokantavalmistajien tuotteisiin. Eräänä suunnittelu-vaihtoehtona on tällöin näiden erilaisten tietokantojen yhteenliittäminen hajautetuksi tietokannaksi. Ratkaisulla tavoitellaan ennen kaikkea kustannussäästöä, voivathan olemassa olevat paikalliset järjestelmät jatkaa toimintaansa uusien globaalien järjestelmien rinnalla.

Tietokantatuotteet voivat tukea yhteenliittämistä kahdella vaihtoehtoisella tavalla. Ensiksi tietokannan hallinnassa käytettävä ohjelmanosa voi olla tietokannan hallintajärjestelmän ulkopuolella, jolloin kyseessä on väliohjelma (gateway). Toiseksi yhteistoiminnan takaava ohjelmanosa voidaan sijoittaa tietokannan hallintajärjestelmän sisään. Tällöin puhutaan monitietokannan hallintajärjestelmästä (multidatabase management system) (Simon, 1996, s. 275).

Yhteenliittäminen ei ole kuitenkaan yksinkertainen tehtävä. Vaikka keskityttäisiinkin pelkästään SQL-pohjaisiin tietomalleihin, eroavaisuudet nimeämiskäytännöissä sekä puuttuvissa ja ristiriitaisissa tietoalkioissa tekevät heterogeenisen hajautetun tietokannan hallintajärjestelmän toteuttamisen vaikeaksi (Khoshafian ym., 1991, s. 508).

Tietokantavalmistajat ovat huomanneet tämän ja keskittyneet tuottamaan ohjelmistaan versiot aluksi erilaisille laite- ja käyttöjärjestelmäalustoille.

Suorituskykynäkökohdat

Hajautetun tietokantajärjestelmän kokonaiskapasiteettia on pidetty keskitettyä ratkaisua suurempana (Frank, 1988, s. 232). Mikäli hajautettu tietokanta on suunniteltu oikein, sen kapasiteetin kasvattaminen on keskitettyä ratkaisua yksinkertaisempaa. Mahdollisesti vain yhtä tietokantaa laajennetaan, ja koko tietokantajärjestelmän kapasiteetin kasvattaminenkin onnistuu lisäämällä uusi tietokantasolmu (Pratt, 1991, s. 701). Kapasiteettia voidaan hyödyntää tehokkaasti käyttämällä rinnakkaiskäsittelyä kyselyissä ja päivityksissä (Simon, 1996, s. 14).

Mainittuihin positiivisiin arvioihin on suhtauduttava varauksella, sillä hajautetun tietokantajärjestelmän suorituskyvyn suunnittelunaikaiseen arviointiin ei ole vielä olemassa riittäviä malleja, välineitä ja menetelmiä (Özsu ym., 1996a, s. 13). Suorituskykyyn keskittyneissä tutkimuksissa on Özsun ym. (1996a) mukaan käytetty keinotekoisia verkkokuormia ja liikaa yksinkertaistavia malleja. Tietoliikennekustannukset on arvioitu kiinteiksi. Verkon koko ja kuorma sekä viestien koot ovat jääneet huomioitta. Keskitettyyn tietokantaympäristöön laadittua suorituskyvyn arvioinnin välineistöä ei ole vielä saatu riittävästi laajennettua hajautettuun ympäristöön.

Toimintavarmuus ja saatavuus

Hajautettujen tietokantojen etuna pidetään niiden tarjoamaa toimintavarmuutta ja parantunutta saatavuutta (esim. Korth, 1991; Pratt, 1991; Simon, 1996; Özsu ym., 1996a). Simon (1996, s. 175) määrittelee toimintavarmuuden ja saatavuuden seuraavasti. *Toimintavarmuus* (reliability) on todennäköisyys sille, että järjestelmä jatkaa toimintaansa tietyllä aikavälillä. *Saatavuudella* (availability) tarkoitetaan sitä tasoa, jolla järjestelmä on käytettävissä käyttäjän niin halutessa.

Hajautetuissa tietojärjestelmissä yhden komponentin vikaantuminen voi estää tietyn käyttäjäryhmän työskentelyn, mutta se ei estä muita käyttäjiä jatkamasta työskentelyään. Koko järjestelmän parantunutta toimintavarmuutta ei pidä kuitenkaan ottaa itsestään-selvyytenä. Useiden toisistaan poikkeavien komponenttien muodostama kokonaisuus voi osoittautua helposti vikaantuvaksi (Simon, 1996, s. 178). Järjestelmän suunnittelijan vastuulle jääkin erilaisten vikatyyppien tunnistaminen, tietyn palvelun vikaantumiseen johtavan kehityskulun selvittäminen ja erilaisten mekanismien kehittäminen vikojen peittämiseksi käyttäjiltä tai niiden esittämiseksi selvästi havaittavina virheinä (Simon, 1996, s. 179).

Paikallinen autonomia

Kullakin tietokantasolmulla on omat käyttäjänsä, oma fyysinen tietokanta, oma paikallinen hajautetun tietokannan hallintajärjestelmä, oma tapahtumanhallintaohjelmisto ja oma tietoliikenteen hallintaohjelmisto (Date, 1995, s. 594). Näin ollen kunkin paikallisen tietokannan hoitaja voi kontrolloida juuri kyseisen organisaatioyksikön tietojen käyttöä ja sallia tiedon käsittelyn muista tietokantasolmuista.

2.2.4 Hajautetun tietokannan haitat

Tässä kohdassa tarkastellaan muutamia hajautettujen tietokantojen haittoja. Hajautettu tietokantateknologia on eräiltä osiltaan vielä hyvin nuorta. Sen vuoksi tuotteissa on piirteitä, jotka muuttuvat tulevaisuudessa.

Vaikeampi hallita ja varmistaa

Simon (1996, s. 14) esittää eräänä hajautettujen tietojenkäsittelyn ongelmana järjestelmien vaikeaa hallittavuutta ja varmistamista. Tietoturvan, järjestelmän hoidon ja käyttäjien tuen järjestämiseen tarvittavat menettelytavat ovat hajautetuissa järjestelmissä

monimutkaisempia. Sen vuoksi myös kustannukset ovat suurempia.

Koulutetun henkilöstön ja ammattitaitoisen tuen puute

Hajautettu tietokantateknologia on iältään varsin nuorta. Ensimmäiset hajautetut tietokantatuotteet ilmestyivät markkinoille vasta 1990-luvun alkuvuosina, ja asiakas/palvelin -arkkitehtuuri on hallinnut tietotekniikan valtavirtaa aina viime vuosiin saakka. Selvää kiinnostusta hajautettuun tietokantateknologiaan on kuitenkin ilmennyt jo vuosia.

Hajautettujen järjestelmien tuen puolella asiakkaita on haitannut toimittajien kyvyttömyys tarjota yhtä kokonaisvaltaista tukea kuin aiempien keskitettyjen järjestelmien toimittajat tarjosivat. Hajautettujen järjestelmien komponenttien yhteensovittaminen on jäänyt asiakkaalle, joskin toimittajat ovat alkaneet tarjota konsultointi- ja yhteensovittamispalveluita (Kay, 1996).

Monimutkaisuus

Hajautettu tietokannan hallintajärjestelmä on monimutkainen ohjelmisto keskitettyyn tietokannan hallintajärjestelmään verrattuna (Hoffer ym., 2002, s. 498). Monimutkaisuutta lisää eri tietokantasolmujen virheettömän yhteistoiminnan takaaminen. Ilmenemismuotoina ovat suuremmat ohjelmiston kehityskustannukset ja suurempi virhetodennäköisyys rinnakkain suoritettavissa algoritmeissa (Korth, 1991, s. 477). Myös tietokantasolmujen välillä lähetettävät viestit lisäävät käsittelykuormaa (Korth, 1991, s. 477 ja Hoffer ym., 2002, s. 498).

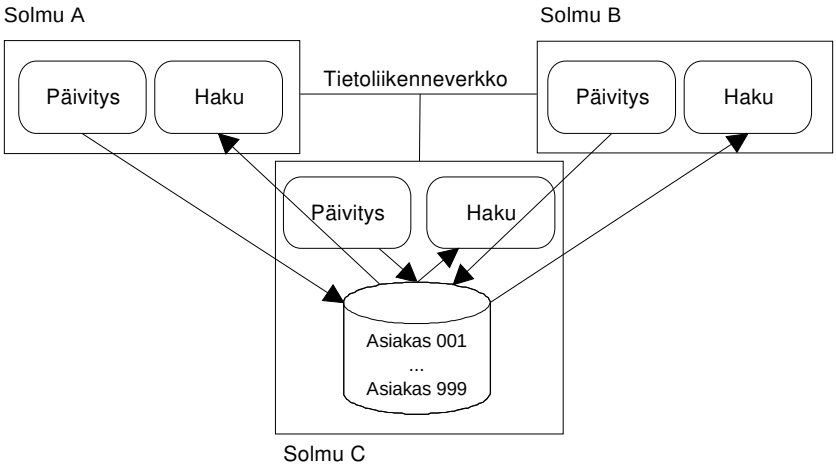
2.3 Tiedon hajautuksen vaihtoehdot

Tässä kohdassa keskitytään ensiksi erilaisiin tiedon sijoitusvaihtoehtoihin Simonin (1996, s. 258) esittämän jaottelun mukaan. Sen jälkeen esitetään muutamia vaihtoehdon

valintaan vaikuttavia tekijöitä. Lopuksi arvioidaan, mihin tilanteisiin esitetyt hajautusvaihtoehdot soveltuvat parhaiten.

Keskitetty tiedonhallinta

Tiedon sijoittaminen yhteen keskitettyyn tietokantaan voi olla useissa tapauksissa aivan perusteltu ratkaisu. Tällainen ratkaisu soveltuu esimerkiksi silloin, kun useat organisaatioyksiköt päivittävät samaa suurta tietojoukkoa ja tiedon täytyy olla ajantasaista. Hotellien, lentoyhtiöiden ja autovuokraamojen varausjärjestelmät ovat esimerkkejä tällaisista järjestelmistä. Kuvassa 3 on esimerkki keskitetystä tiedonhallintaratkaisusta.

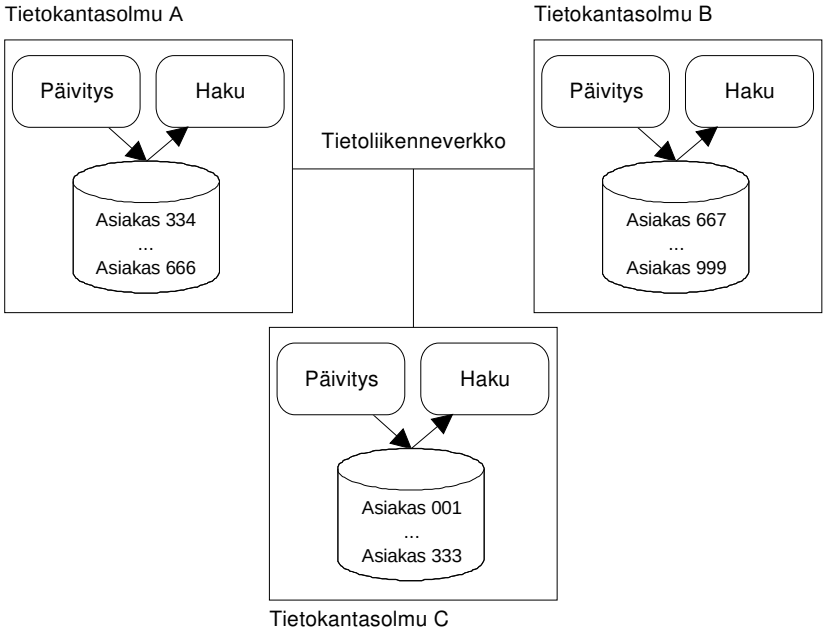


Kuva 3. Esimerkki keskitetystä tiedonhallinnasta (Simon, 1996, s. 259).

Ositettu tieto

Osittamisella (partitioning) tarkoitetaan relaatioiden jakamista toisensa poissulkeviin osiin, jotka sijoitetaan yleensä fyysisesti eri tietokantasolmuihin. Horisontaalisessa osittamisessa relaation rivit jaetaan kahteen tai useampaan osaan valintaoperaatiolla. Vertikaalisessa osittamisessa relaation sarakkeet jaetaan kahteen tai useampaan osaan käyttäen projektiota. Esimerkiksi pankin asiakastiedot sisältävä relaatio voidaan osittaa horisontaalisesti siten, että toimipisteen A relaatio sisältää kaikki kyseisen maantieteellisen alueen rivit ja toimipisteen B relaatio vastaavasti oman alueensa rivit.

Jos pankin asiakkaat käyttävät vain oman asuinalueensa toimipisteen palveluita, niin kaikki tietokantaan kohdistuvat päivitykset ja kyselyt voidaan rajoittaa paikalliseen tietokantasolmuun. Tässä tapauksessa osittaminen on paras vaihtoehto, sillä jokainen toimipiste käsittelee omaa tietoaan. Jos asiakkaat käyvät muissa toimipisteessä, osittamisen kannattavuuteen vaikuttaa lähinnä päivitysten määrä. Vertikaalisessa osittamisessa pankin työntekijätiedot sisältävä relaatio voitaisiin jakaa esimerkiksi siten, että osaan A sijoitettaisiin työntekijän henkilönumero, nimi ja osoite, ja osaan B henkilönumero, palkka ja esimiestunnus. Kuvassa 4 esitetään esimerkki horisontaalisesta osittamisesta.



Kuva 4. Esimerkki horisontaalisesta osittamisesta (Simon, 1996, s. 260).

Ote

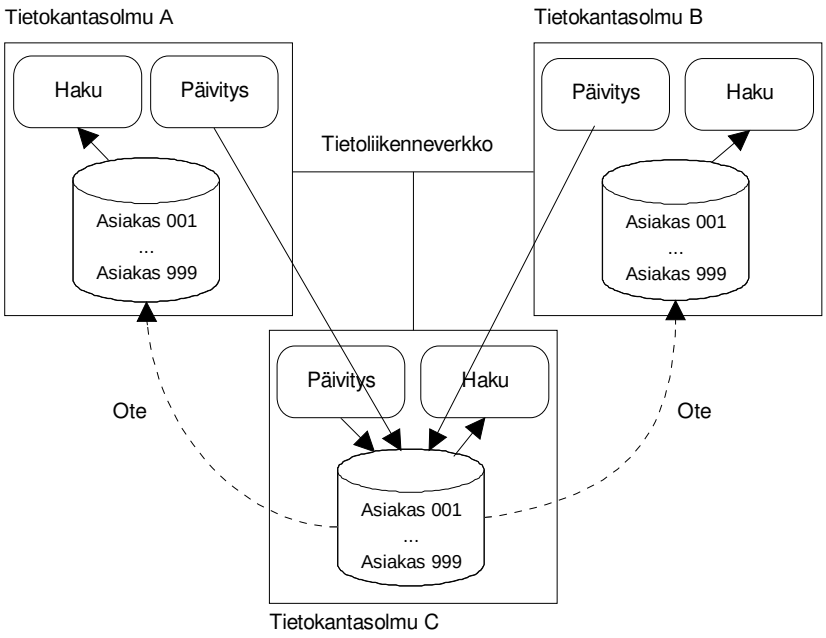
Otteella (extract) tarkoitetaan tiedon kopioimista tietokantasolmusta toiseen solmuun lukutarkoitusta varten. Otteen ottamisen tuloksena saadaan hetkellinen näkymä tietokannan tietoon. Simon (1996) esittää kolme erilaista otetyyppiä:

1. Käyttäjä suorittaa SQL-kyselyn, jonka tulos tallennetaan käyttäjän omaan tietokantasolmuun jatkokäsittelyä varten.

2. Aikaleimattu ote sisältää aikaleiman, josta sovellukset voivat päätellä otteen sisältämän tiedon ajantasaisuuden.
3. Virkistetty ote suoritetaan automaattisesti tietyin aikavälein käyttäjän sitä erikseen pyytämättä.

Khoshafian ym. (1991) esittävät, että ns. *tilannevedoksessa* (snapshot) yhteys otteen lähteenä olevan ja lopputuloksena saadun näkymän välillä säilyy prosessin päätyttyä ja prosessi toistetaan tietyin aikavälein suorittamalla uudelleen tulosnäkymään liitetty kysely. Tilannevedos vastaa siis Simonin (1996) esittämää virkistettyä otetta.

Otteella on lukuisia sovellusmahdollisuuksia. Raportoinnin järjestelmissä tietoa yleensä ainoastaan luetaan, joten ote on varteenotettava vaihtoehto tiedonsiirtoon raportoinnin tarpeisiin. Kuvan 5 esimerkissä otetta sovelletaan tietokantasolmuissa A ja B raportointiin. Ote on myös käyttökelpoinen, jos käyttäjien ei tarvitse nähdä ajantasaisinta versiota tiedosta.



Kuva 5. Esimerkki otteesta (Simon, 1996, s. 261).

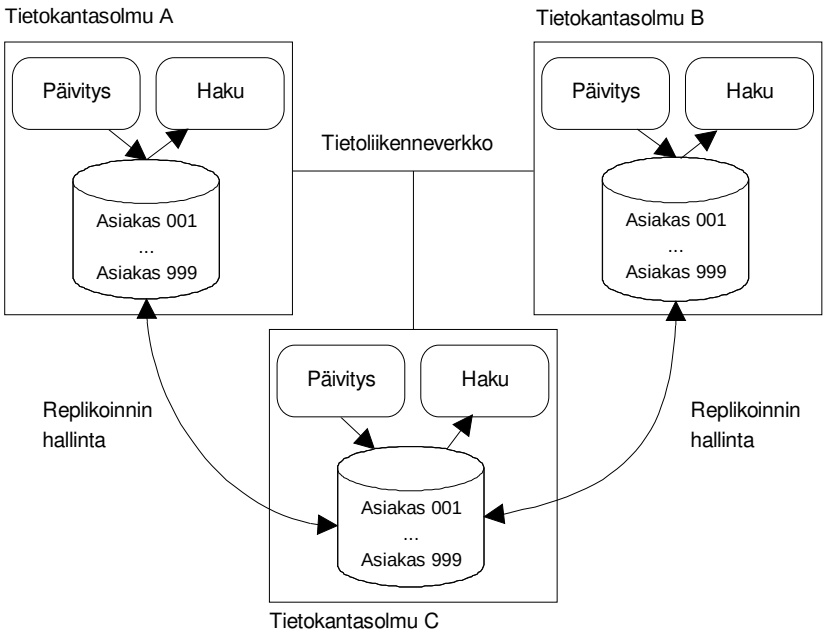
Replikointi

Simonin (1996) mukaan *replikoinnilla*[2] (replication) tarkoitetaan tiedon toisintamista tietokantasolmusta toiseen/toisiin päivitystarkoitusta varten. *Lähdejärjestelmällä* tarkoitetaan jatkossa tietokantasolmua, josta replikoidaan. *Kohdejärjestelmä* on puolestaan tietokantasolmu, johon replikoidaan. Toisintamisen tulosta eli toiseen tietokantasolmuun sijoitettua tietoa kutsutaan *replikaatiksi*. Replikoinnilla tavoitellaan ennen kaikkea parempaa suorituskykyä ja saatavuutta (Date, 1995, s. 602). Parempi suorituskyky johtuu siitä, että tietokantasolmun sovellukset voivat käyttää paikallista tietokopiotaan, eikä niiden tarvitse hakea tietoa toisesta tietokantasolmusta. Saatavuutta puolestaan parantaa se, että tieto on saatavilla niin kauan kuin ainakin yksi kopioista on saatavilla. Replikoinnin vaikein ongelma liittyy replikaattien pitämiseen eheänä päivitysten yhteydessä. Kuinka tietoihin kohdistuvat päivitykset välitetään muissa tietokantasolmuissa sijaitseviin kopioihin? Kuinka menetellään jonkin tietokantasolmun vikaannuttua? Näihin seikkoihin on tarjottu ratkaisuja lukuisissa replikoinnin-hallintaprotokollissa (Simon, 1996; Özsu ym., 1996a; Date, 1995). Kuvan 6 esimerkissä replikointia sovelletaan siten, että tietokantasolmut A ja B voivat myös päivittää tietoa.

Simonin (1996) kuvaus replikoinnista kaipaa eräiltä osin tarkennusta. Replikaattien päivitystarkoitusta korostetaan esityksessä liikaa, sillä replikointia sovelletaan pääasiassa tilanteisiin, joissa replikaatteja ei ole tarvetta päivittää. Esimerkiksi päätöstukijärjestelmien yhteydessä replikaatit ovat luonnostaan kyselyluonteisia (Schussel, 1996). Tietovarastointiratkaisuun (data warehousing) sovellettaessa replikointia käytetään niinikään kyselyominaisuudessa. Simon (1996) ei myöskään tuo esille sitä, että tilannevedosta voidaan käyttää kyselyluonteisen replikoinnin toteutustapana hajautetuissa tietokannan hallintajärjestelmissä (Oracle, 2002a, s. 3-2). Replikoinnin ja otteen välinen ero onkin tässä tapauksessa hämärä.

---

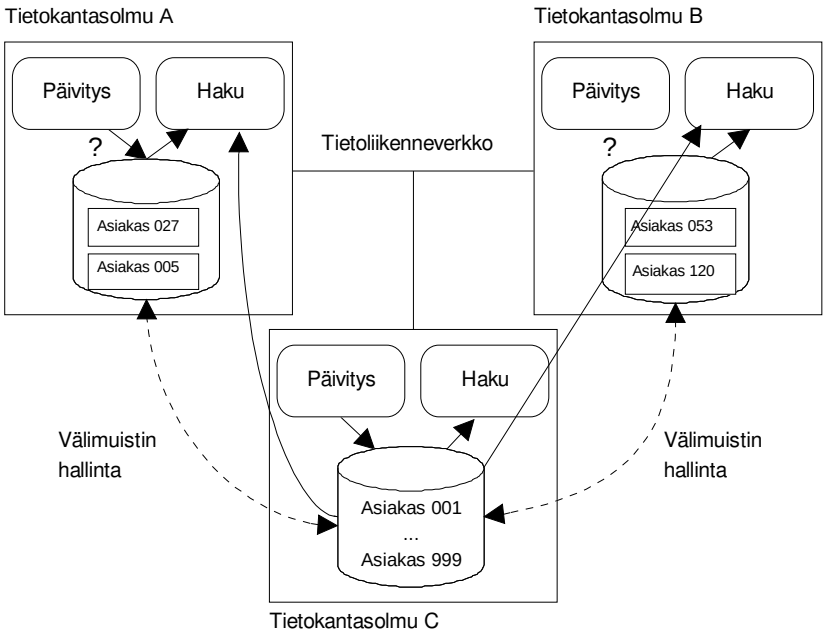
[2] Atk-sanakirja (1996) esittää sanan ”replicate” käännökseksi sanaa ”toisintaa”. Tässä raportissa käytetään sanaa ”replikoida”, sillä se näyttää vakiintuneen yleiseen kielenkäyttöön. Sanat ”replication” ja ”replicate” (subst.) käännetään vastaavasti sanoiksi ”replikointi” ja ”replikaatti”.



Kuva 6. Esimerkki replikoinnista (Simon, 1996, s.261).

Välimuistiin tallennettu tieto (cached data)

Välimuistia voidaan käyttää tiedon tilapäiseen replikointiin. Esimerkiksi vakuutusyhtiön toimipisteen tietokantasolmu voi hakea tietoa päivitettäväksi keskushallinnon tietokantapalvelimelta ja palauttaa tiedon päivityksen jälkeen takaisin. Tällä menettelyllä saadaan jaettua kuormaa tasaisemmin tietokantasolmujen kesken. Tässäkin vaihtoehdossa tarvitaan välimuistinhallintamenetelmä, jolla varmistetaan tiedon ajantasaisuus ja joka osaa käsitellä välimuisteissa olevia vanhentuneita tietoja. Kuvassa 7 esitetään esimerkki välimuistin käytöstä tietojen hajauttamisessa.



Kuva 7. Välimuistin käyttö tiedon hajautuksessa (Simon, 1996, s. 262).

2.3.1 Tiedon hajautusvaihtoehdon valintakriteerit

Simonin (1996, s. 262) mukaan seuraavat tekijät vaikuttavat hajautusvaihtoehdon valintaan.

Tiedon omistussuhteet

Tiedon omistussuhteet vaikuttavat siihen, kuinka hajautetun tietokannan globaalit relaatiot voidaan jakaa osiin. Ihannetapauksessa kukin osa voidaan sijoittaa siihen tietokantasolmuun, jossa käyttäjä suorittaa tiedon käsittelyä. Tiedon omistajuutta voi selvittää tunnistamalla tapahtumia, jotka luovat, muuttavat, lukevat ja poistavat tietoalkion. Tämän jälkeen voidaan yksilöidä ne käyttäjäryhmät, jotka käynnistävät tapahtumat (Simon, 1996, s. 260).

Hakujen paikallisuus

Tämän tekijän osalta selvitetään, kuinka usein muut tietokantasolmut lukevat tai päivittävät paikallista tietoalkiota.

Tiedon muuttumisnopeus

Muuttumisnopeus lasketaan tiedon määrän ja käsittelytaajuuden avulla. Esimerkiksi 200 000 tietuetta sisältävästä tietokannasta päivitetään 100 000 tietuetta kuukaudessa. Muuttumisnopeus on tällöin 100 000/200 000 = 0.5. Yleisesti ottaen osittaminen soveltuu hyvin suuriin, paljon vaihtelua sisältäviin tietokantoihin ja replikointi pieniin, vähän vaihtelua sisältäviin tietokantoihin.

Tiedon käyttöarvo

Tekijän osalta arvioidaan, mikä on tiedon arvo liiketoiminnalle. Kuinka ajantasaista tiedon täytyy olla? Vastaukset näihin kysymyksiin määrittelevät sen, mitä replikoinnin hallintamenetelmää käytetään, mikäli ylipäätään päädytään replikoinnin käyttöön.

Tiedon määrä

Tarkasteltavan tietokannan koko voi olla myös merkittävä tekijä hajautusvaihtoehtoa valittaessa. Kuinka suuri tietokanta on suhteessa käytettävissä olevaan levytilaan? Jos tietokanta on erittäin suuri ja sitä ei voida osittaa, niin keskitetty ratkaisu voi olla soveltuvin.

Tiedon saatavuus

Tekijän osalta selvitetään, kuinka tärkeä tietokanta on sitä käyttäville sovelluksille. Jos kysymyksessä on toiminnan kannalta kriittinen tietojärjestelmä, järjestelmä ei voi jatkaa

toimintaansa tiedon saatavuuden heikennyttyä. Mikäli tieto ei ole kriittinen tekijä, toimintaa voidaan jatkaa alennetulla palvelutasolla.

Taulukossa 1 on esitetty yhteenvetona kunkin hajautusvaihtoehdon ominaispiirteitä ja arvio sovelluskohteista.

Taulukko 1. Hajautusvaihtoehtojen ominaispiirteitä ja sovelluskohteita.

Hajautustapa	**Ominaispiirteitä**	**Mahdollisia sovelluskohteita**
Keskitetty tiedonhallinta - ei hajautusta	-Tieto yhden tietokannan hallintajärjestelmän alaisuudessa -Helppo hallita ja varmistaa -Laiteviat voivat estää tietojen käytön kaikilta	Pankkien, autovuokraamojen ja hotellien järjestelmät, joissa tietojen eheys on ensisijaisen tärkeää
Osittaminen	-Soveltamiskelpoinen, mikäli tiedon omistajuus selvitettävissä -Voi nopeuttaa paikallisten tietojen käsittelyä -Eri tietokantasolmuihin kohdistuvien tietokantatapahtumien koordinointi vaatii huomiota	Esimerkiksi vakuutusyhtiöiden tietojärjestelmät, joilla kukin toimisto käsittelee omaa aluetta koskevia tietojaan
Ote	-Valmiita toteutusmekanismeja tietokannan hallintajärjestelmissä -Erityisesti käyttötilanteisiin, joissa tietoa vain luetaan	Tietojenkäsittelykuorman jakaminen, raportoinnin tietojärjestelmät
Replikointi	-Teknologiana uusin, osittain kypsymätön (replikaattien päivityksen osalta) -Tietojen päivitys mahdollista -Parantaa saatavuutta ja suorituskykyä	Raportoinnin tietojärjestelmät, tietovarastointiratkaisu, liikkuvat tietojärjestelmät, tiedon jakelu
Välimuistin käyttö	-Tuntemattomin	Raportoinnin järjestelmät, tietojärjestelmät, joissa tieto muuttuu päivitysten jälkeen harvoin (koodistojen ja luetteloiden jakelu)

2.4 Hajautettu tietokannan hallintajärjestelmä

Tässä kohdassa esitetään ensin tavoitteita hajautetulle tietokannan hallintajärjestelmälle ja sen jälkeen kuvataan hallintajärjestelmän sisältämät komponentit.

2.4.1 Tavoitteet hajautetulle tietokannan hallintajärjestelmälle

Hajautettujen tietokantojen keskeisimmän tavoitteen mukaan hajautetun tietokannan tulisi näyttää käyttäjille täsmälleen samanlaiselta kuin keskitetynkin tietokannan (esim. Date, 1995, s. 596; Khoshafian ym., 1991, s. 528; Pratt, 1991, s. 708). Käyttäjillä tarkoitetaan tässä yhteydessä tietokannan käyttäjiä ja sovellusohjelmoijia. Daten (1995) luokittelussa on seuraavat 12 tavoitetta:

1. Paikallinen autonomia

   Kunkin tietokantasolmun tulisi omistaa paikalliset tietonsa ja kontrolloida itsenäisesti tietojen turvallisuutta, yhtenäisyyttä ja tallennusrakennetta.

2. Riippumattomuus keskuslaitteistosta

   Kaikkien tietokantasolmujen täytyy olla tasavertaisia. Järjestelmässä ei saa olla yhtä palveluja tarjoavaa solmua, jota kaikki muut käyttävät. Tätä tavoitetta noudattamalla yksikään solmu ei rajoita järjestelmän suorituskykyä tai heikennä sen toimintavarmuutta.

3. Keskeytyksetön toiminta

   Tietokantasolmun lisääminen tai poistaminen ei saa häiritä muiden solmujen toimintaa. Taulujen lisäämisen ja poistamisen sekä ohjelmaversion päivittämisen tulisi aiheuttaa mahdollisimman vähän, mielellään ei yhtään, häiriötä hajautetun tietokantajärjestelmän toiminnalle.

4. Paikkariippumattomuus

   Käyttäjien ei tulisi tietää, missä tiedot fyysisesti sijaitsevat. Tavoitteen täyttyessä sovelluslogiikka yksinkertaistuu ja tietoja on mahdollista siirtää tietokantasolmusta toiseen muuttamatta niitä käsitteleviä sovellusohjelmia.

5. Ositusriippumattomuus

   Järjestelmän tulisi piilottaa käyttäjältä tietojen osittaminen. Jos tavoite täyttyy, ositusta voidaan muuttaa sovellusohjelmia päivittämättä.

6. Riippumattomuus tietojen replikoinnista

   Hajautettu tietokannan hallintajärjestelmä tukee tietojen replikointia, jos tallennettu relaatio tai sen osa voidaan toisintaa useampaan tietokantasolmuun.

   Esimerkiksi N_EMP ja L_EMP relaatiot replikoidaan Lontooseen ja New York:iin seuraavasti. Komentoa REPLICATE ei välttämättä ole missään hajautetussa tietokannan hallintajärjestelmässä.

   ```
   REPLICATE N_EMP
           LN_EMP AT SITE ”London”;
   REPLICATE L_EMP
           NL_EMP AT SITE ”New York”;
   ```

   Tavoitteen mukaan järjestelmän tulee peittää useiden tietokopioiden olemassaolo käyttäjältä. Hajautetun tietokannan hallintajärjestelmän tehtävänä on suorituskyvyn kannalta parhaan tietokopion valitseminen ja päivitysten suorittaminen kaikkiin tietokopioihin. Tavoitteen täyttyessä replikaatteja voidaan luoda ja poistaa muuttamatta sovellusohjelmia.

7. Hajautettu kyselyjenkäsittely

   Hajautettu tietokantakysely voi kohdistua useissa tietokantasolmuissa sijaitseviin tietoihin. Kyselyä suoritettaessa tietokannan hallintajärjestelmän tulisi toimia niin, ettei tulisi tarvetta siirtää tietoja muista tietokantasolmuista kyselyn aloittaneeseen tietokantasolmuun lopputuloksen tuottamiseksi.

8. Hajautettu tapahtumien hallinta

   Yksi tietokantatapahtuma voi aiheuttaa hajautetuissa tietokannoissa muutoksia useiden tietokantasolmujen tietoihin. Hajautetun tietokannan hallintajärjestelmän täytyy varmistaa, että muutokset suoritetaan kaikkiin tietokantasolmuihin oikeassa järjestyksessä. Järjestelmän tulee myös säilyttää kaikki jo hyväksytyt tietokantapahtumat. Mikäli tapahtuman suorituksen aikana ilmenee virheitä, järjestelmän on tuettava automaattista toipumista.

9. Laitteistoriippumattomuus

   Saman hajautetun tietokannan hallintajärjestelmän tulisi toimia eri laitteistoissa.

10. Käyttöjärjestelmäriippumattomuus

    Saman hajautetun tietokannan hallintajärjestelmän tulisi toimia eri käyttöjärjestelmissä.

11. Verkkoriippumattomuus

    Eri tietoliikenneverkoissa (esim. LAN ja WAN) toimivien tietokantasolmujen tulisi toimia yhdessä.

12. Riippumattomuus tietokannan hallintajärjestelmästä

    Eri tietokantatuotteiden tulisi kyetä yhteistoimintaan samassa hajautetussa tietokantajärjestelmässä.

Khoshafian ym. (1991, s. 528) jakavat esitetyt tavoitteet kolmeen ryhmään. Tavoitteet 1-3 liittyvät autonomiaan, joka osoittaa, kuinka itsenäisesti tietokantasolmut voivat toimia. Tavoitteet 4-8 osoittavat, missä määrin tietojen sijoittaminen eri tietokantasolmuihin, mahdollinen osittaminen ja replikointi on piilotettu käyttäjältä. Tavoitteet 9-12 liittyvät heterogeenisuuteen ja homogeenisuuteen. Heterogeenisuudella tarkoitetaan eroavaisuutta tietomalleissa sekä laitteisto- ja ohjelmistovaatimuksissa. Homogeenisuus tarkoittaa samankaltaisuutta näissä tekijöissä.

Tavoitteita tarkasteltaessa on huomattava, että ne eivät ole kaikenkattavia, eivätkä kaikki yhtä merkittäviäkään (Date, 1995, s. 597). Ristiriitaisuuksia voidaan havaita esimerkiksi tavoitteissa ”1. Paikallinen autonomia” ja ”8. Hajautettu tapahtumien

hallinta”. Hajautettuun tapahtuman käsittelyyn osallistuessaan tietokantasolmu luovuttaa kontrollin tapahtuman vahvistuksesta koordinoivalle tietokantasolmulle ja näin ollen luopuu väliaikaisesti autonomiastaan (Khoshafian ym., 1991, s. 530). Tavoitteet auttavat kuitenkin hajautetun tietokantateknologian ymmärtämisessä, ja niitä käytetään yleisesti hajautetun tietokannan hallintajärjestelmien luokittelussa ja ominaisuuksien arvioinnissa (esim. Khoshafian ym., 1991, s. 551). Yhtä optimaalista hajautettua tietokannan hallintajärjestelmää ei ole olemassa, sillä käyttäjät painottavat tavoitteita eri tavoin (Khoshafian ym., 1991, s. 551).

2.4.2 Hajautetun tietokannan hallintajärjestelmän komponentit

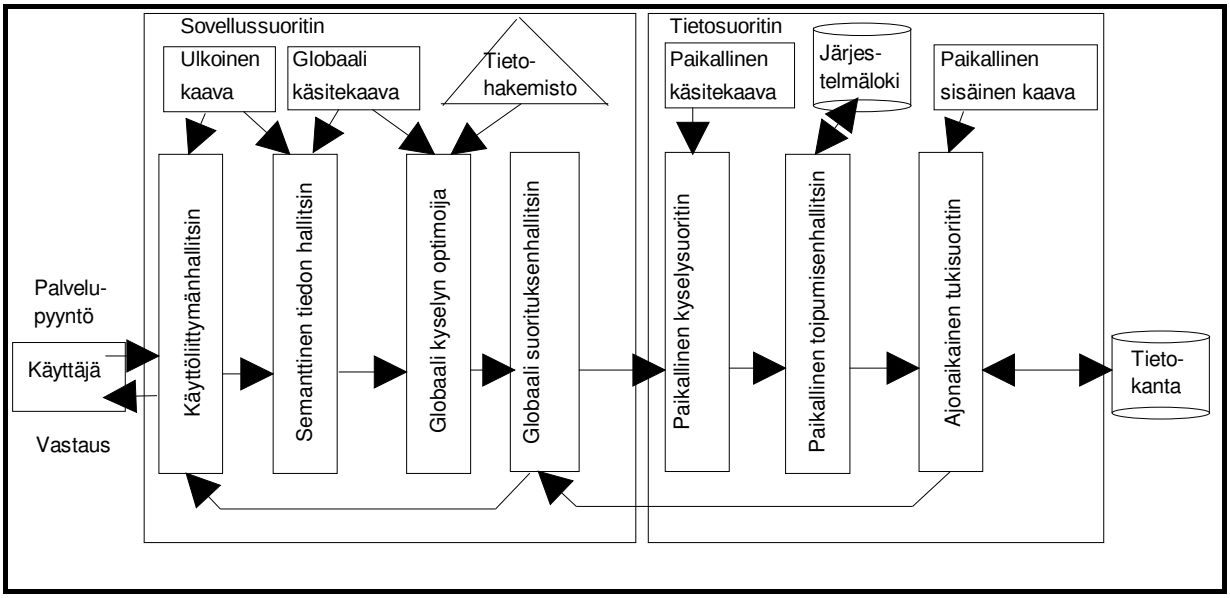
Tässä kohdassa tarkastellaan hajautetun tietokannan hallintajärjestelmän sisältämiä komponentteja. Esitys pohjautuu Özsun ym. (1991, s. 85) kuvaukseen komponenteista. Vaikka kuvaukseen perustuvia tietokantatuotteita ei kenties olekaan olemassa, on siitä apua hajautetun tietokannan hallintajärjestelmän toiminnan ymmärtämisessä.

Hajautetussa tietokannan hallintajärjestelmässä kukin tietokantasolmu sisältää sovellussuorittimen[3] (user processor) ja tietosuorittimen (data processor). Sovellussuorittimen tehtävänä on kommunikointi käyttäjän kanssa. Tietosuoritin käsittelee kussakin tietokantasolmussa suoritettavat tietokantaviittaukset.

Tietokantatapahtumien käsittely etenee hajautetussa tietokannan hallintajärjestelmässä pääpiirteissään kuvan 8 osoittamalla tavalla. Käyttöliittymänhallitsin tulkitsee käyttäjän kyselyn, joka välitetään semanttiselle tiedonhallitsimelle. Semanttinen tiedonhallitsin varmistaa tietokantatapahtuman eheydelle ja tietoturvalle asetettujen vaatimusten täyttymisen. Tapahtuma annetaan globaalille kyselyn optimoijalle, joka valitsee tehokkaimman suoritustavan eri tietokantasolmujen kesken. Tapahtuman suorituksen

[3] Suoritin on tässä yhteydessä loogisen tason käsite.

hallinnassa ajoitetut osatapahtumat lähetetään tietosuorittimille suoritettaviksi. Paikallinen kyselysuoritin valitsee parhaan vaihtoehdon osatapahtuman suorittamiseksi. Toipumisenhallitsin varmistaa, että tietokanta jää yhtenäiseen tilaan virhetilanteissakin. Ajonaikainen tukisuoritin suorittaa fyysisen tietokannan päivittämisen paikalliselta kyselysuorittimelta saamiensa ohjeiden perustella. Tieto palautetaan suorituksen-hallitsimelle, josta se välitetään edelleen muotoiltavaksi lopulliseksi vastaukseksi.



Kuva 8. Hajautetun tietokannan hallintajärjestelmän komponentit tietokantatapahtumaa suoritettaessa (Özsu ym., 1991, s. 85).

2.5 Hajautettu tapahtumankäsittely

Tässä kohdassa määritellään ensin hajautettu tietokantatapahtuma ja esitellään siihen yleisesti liitetyt ominaisuudet. Sen jälkeen tarkennetaan hajautetun tietokannan hallinta-järjestelmän komponenttikuvausta globaalin suorituksenhallitsimen osalta. Kokonais-kuvan saamiseksi hajautetusta tapahtumankäsittelystä tässä kohdassa esitetään eräs tapa, jolla samanaikaisuuden, replikoinnin, toipumisen ja tapahtumanvahvistuksen hallinnassa käytetyt algoritmit ja protokollat voidaan liittää hajautetun tietokannan hallintajärjestelmän komponentteihin.

2.5.1 Hajautettu tietokantatapahtuma

Tietokantatapahtuma on ristiriidaton ja luotettava tietojenkäsittelykokonaisuus, joka koostuu jakamattomasti suoritettavista tietokantaoperaatioista (Özsu ym., 1994b, s. 8). *Hajautetulla tietokantatapahtumalla* tarkoitetaan tietokantatapahtumaa, joka käsittelee useissa eri tietokantasolmuissa sijaitsevaa tietoa (Özsu ym., 1991, s. 269).

Mikäli hajautettu tietokannan hallintajärjestelmä tukee täysin hajautettuja tietokantatapahtumia, sovellukset voivat käsitellä yhtä loogista tietokantaa ja varmistua siitä, että hajautetut tietokantatapahtumat suoritetaan oikein tietoliikenteessä ja tietokantasolmuissa mahdollisesti ilmenevistä häiriöistä huolimatta.

Tietokantatapahtuman on täytettävä seuraavat neljä ominaisuutta niin hajautetussa kuin keskitetyssäkin ympäristössä (mm. Özsu, 1999, s. 283; Garcia-Molina ym., 2000, s. 10):

1. Jakamattomuus (atomicity)
   Tietokantatapahtuma on yksi toiminnallinen kokonaisuus. Kaikki sen sisältämät operaatiot suoritetaan kokonaisuudessaan loppuun tai virhetilanteessa peruutetaan tapahtuman aloitusta edeltäneeseen tilaan.
2. Ristiriidattomuus (consistency)
   Ristiriidattomuudella tarkoitetaan tietokantatapahtuman oikeellisuutta. Tapahtuma siis muuttaa tietokannan ristiriidattomasta tilasta toiseen ristiriidattomaan tilaan.
3. Eristyvyys (isolation)
   Eristyvyyden mukaan jokainen tietokantatapahtuma näkee tietokannan ristiriidattomassa tilassa. Tietokantatapahtuma ei siis näytä keskeneräisiä välituloksiaan muille samanaikaisesti suoritettaville tapahtumille.
4. Säilyvyys (durability)
   Tietokantatapahtuman vahvistuksen jälkeen tietokantaan tehdyt muutokset ovat pysyviä eikä niitä voi peruuttaa.

Näiden ominaisuuksien täyttäminen on hajautetuissa tietokannoissa huomattavasti keskitettyä tietokantaa vaikeampaa. Tämä johtuu siitä, että hajautetun tietokannan

hallintajärjestelmän täytyy varmistaa useiden tietokannan osien eheys yhden tietokannan sijaan (Simon, 1996, s. 13). Ominaisuuksien täyttämistä tarkastellaan alakohdassa 2.5.3.

2.5.2 Globaalin suorituksenhallitsimen komponentit

Edellä esitetty jaottelu hajautetun tietokannan hallintajärjestelmän komponenteista ei välttämättä pohjaudu mihinkään olemassa olevaan tietokantatuotteeseen. Jaottelusta on kuitenkin se hyöty, että sen avulla useita hajautetun tapahtumankäsittelyn tekijöitä voidaan tarkastella toisistaan erillään. Tässä kohdassa globaalin suorituksenhallitsimen toimintaa kuvataan tarkemmin hajautettua tietokantatapahtumaa suoritettaessa.

Globaali suorituksenhallitsin lähettää ajoitetut osatapahtumat tietosuorittimille suoritettaviksi. Erityisesti on huomattava, että yksi hajautettu tietokantatapahtuma voi aiheuttaa toimenpiteitä useissa tietokantasolmuissa ja näin ollen tapahtuman suoritukseen voi osallistua useita tietosuorittimia. Kuinka globaali suorituksenhallitsin sitten voi koordinoida tätä monimutkaista kokonaisuutta?

Kuvassa 9 esitetään globaalin suorituksenhallitsimen sisältämät osat. *Tapahtumanhallitsin* (TH) vastaa tietokantaoperaatioiden hallinnasta sovellusohjelman puolesta. Se tarjoaa sovellukselle rajapinnan, joka koostuu seuraavista komennoista: begin_transaction, read, write, commit ja abort. Replikoidussa tietokannassa komennot toimivat korkealla abstraktiotasolla seuraavasti:

- begin_transaction

  Komennolla aloitetaan uusi tietokantatapahtuma. Tapahtumanhallitsin voi luoda käyttöönsä muistirakenteen tapahtuman nimeä ja tapahtuman käynnistänyttä sovellusta varten.

- read

  Mikäli tietoalkio[4] on tapahtuman käynnistäneessä tietokantasolmussa, se palautetaan tapahtumalle. Jos se sijaitsee muualla, tapahtumanhallitsin valitsee yhden kopion ja pyytää sen arvon palauttamista.

- write

  Tapahtumanhallitsin koordinoi tietoalkion arvon päivittämisen niissä tietokantasolmuissa, joissa tietoalkio sijaitsee.

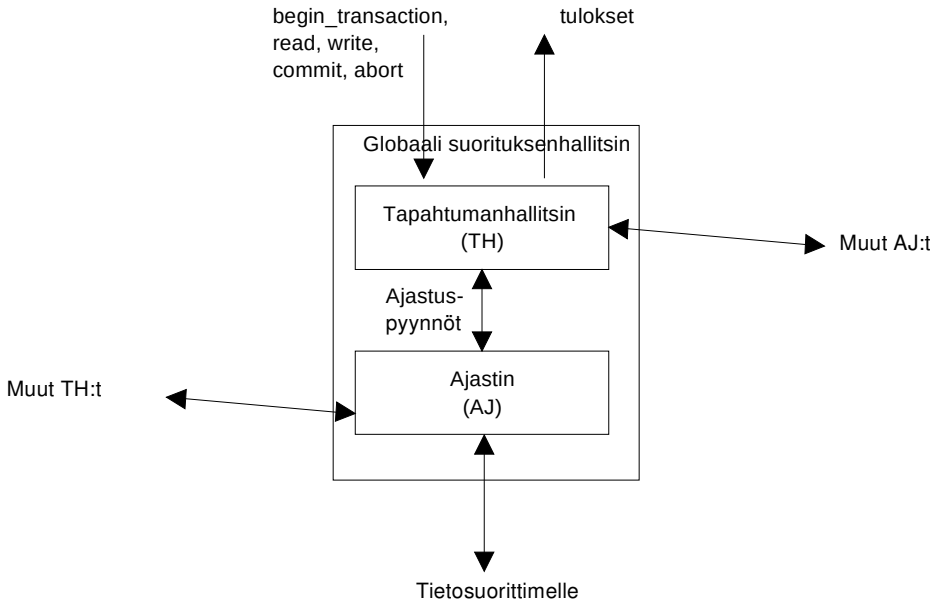
- commit

  Tapahtumanhallitsin koordinoi kaikkien niiden tietokantojen fyysisen päivittämisen, joita edeltävä write-komento koski.

- abort

  Tapahtumanhallitsin varmistaa, ettei tietokantaan tehdä muutoksia.

Ajastin (AJ) huolehtii, että tietokantatapahtumat suoritetaan oikeassa, synkronoidussa järjestyksessä.



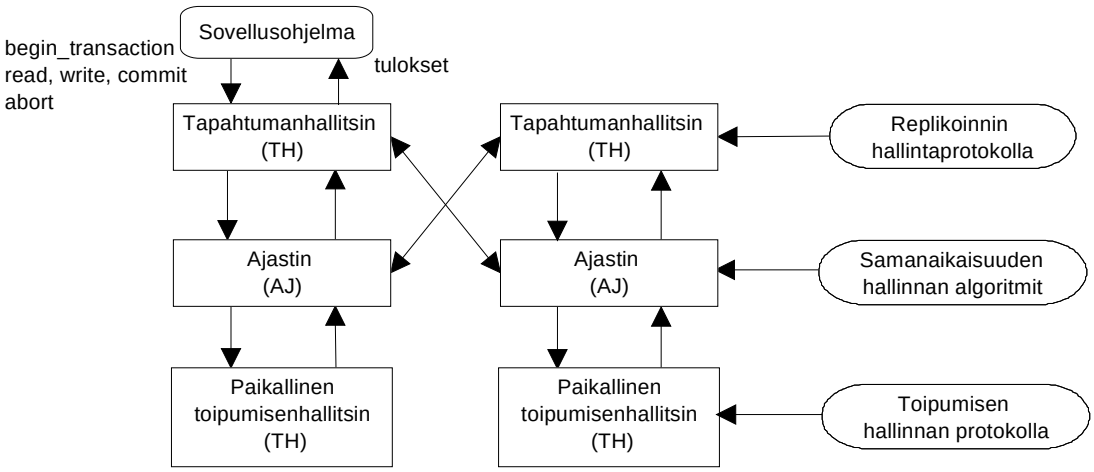
Kuva 9. Globaalin suorituksenhallitsimen osat (Özsu ym., 1991, s. 274).

[4] Tietokantatapahtumat käsittelevät loogisia tietoalkioita. Kullakin loogisella tietoalkiolla on replikoidussa tietokannassa vastineina joukko fyysisiä tietoalkioita eri tietokantasolmuihin tallennettuina (Özsu, 1996a, s. 11). Esimerkiksi henkilön palkka voidaan tallentaa kolmeen tietokantasolmuun.

2.5.3 Hajautetun tapahtumankäsittelyn osa-alueet

Tässä kohdassa esitetään, mihin hajautetun tietokannan hallintajärjestelmän osiin samanaikaisuuden, tapahtumanvahvistuksen, toipumisen ja replikoinnin hallinnassa käytettävät algoritmit ja protokollat vaikuttavat. Lisäksi kohdassa kuvataan, mihin tietokantatapahtuman ominaisuuksiin nämä menetelmät vaikuttavat. Kohdan tarkastelu perustuu Özsun ja Valduriezin (1994a, 1991) esityksiin.

Kuvassa 10 esitetään samanaikaisuuden, replikoinnin ja toipumisen hallinnassa käytettävien menetelmien suhde tietokannan hallintajärjestelmän komponentteihin. *Replikoinnin hallinnan protokollien* tehtävänä on loogisiin tietoalkioihin kohdistuvien luku- ja kirjoituskomentojen suorittaminen fyysisiin tietoalkioihin. Replikoinnin hallinnan protokollia kuvataan tarkemmin alakohdassa 2.5.4. *Samanaikaisuuden hallinnan algoritmien* tarkoituksena on synkronoida yhtäaikaisesti suorituksessa olevat tietokantaoperaatiot. Luokittelu algoritmeista esitetään alakohdassa 2.5.5. *Toipumisen hallinnan protokollien* avulla tietokanta saatetaan ristiriidattomaan tilaan häiriön jälkeen.



Kuva 10. Tapahtumanhallinnan menetelmien suhde tietokannan hallintajärjestelmän osiin (Özsu ym., 1994a, s. 25).

Tapahtumanvahvistuksen protokollat varmistavat tietokantatapahtuman jakamattomuuden joko vahvistamalla (commit) tai kumoamalla (abort) tapahtuman kaikissa tietokantasolmuissa. Tapahtumanvahvistuksen protokollien sijoittuminen kuvassa 10 esitettäviin tietokannan hallintajärjestelmän osiin on ongelmallista, ja näin ollen sitä ei ole esitetty kuvassa. Tämä johtuu siitä, että toipumisenhallinnan protokollien toteutus vaikuttaa siihen, kuinka samanaikaisuuden hallinnan protokollat voidaan toteuttaa. Özsu ym. (1991, s. 375) esittävätkin, että tapahtumanvahvistuksen protokollat voidaan toteuttaa jaetusti tapahtumanhallitsimen ja ajastimen kesken. Protokollista esitellään kaksivaiheinen vahvistus alakohdassa 2.5.6.

Mihin tietokantatapahtumien ominaisuuksiin (ks. alakohta 2.5.1) edellä mainitut algoritmit ja protokollat sitten vaikuttavat? Samanaikaisuuden hallinnan algoritmeilla pyritään takaamaan tapahtuman eristyvyys ja ristiriidattomuus. Tapahtumanvahvistuksessa ja toipumisessa käytettävät protokollat ovat toimintavarmuuden hallinnan protokollia, joilla taataan tietokantatapahtuman jakamattomuus ja säilyvyys. Replikoinnin hallinnan protokollilla pyritään varmistamaan eri tietokopioiden keskinäinen ristiriidattomuus.

2.5.4 Replikoinnin hallinnan protokollat

Replikaattien avulla pyritään lisäämään tiedon saatavuutta. Replikoidussa tietokannassa loogisilla tietoalkioilla on joukko fyysisiä vastineita, replikaatteja, eri tietokantasolmuihin tallennettuina. Sovellusten kannalta järjestelmän tulisi kuitenkin näyttää siltä, kuin käytössä olisi vain yksi tietoalkio. Tämän ehdon täyttäminen on suhteellisen helppoa, mikäli järjestelmässä ei ilmene häiriöitä. Koska häiriöitä yleensä esiintyy, ehdon täyttäminen onkin hankalaa. Monimutkaisuus piilee replikaattien eheyden varmistamisessa, kun samaan aikaan pyritään turvaamaan tiedon saatavuus (Liu, 1994a, s. 2). Ratkaisuksi on esitetty joukko replikoinnin hallinnan protokollia. Protokollat voidaan jakaa kahteen pääryhmään (Simon, 1996):

- *replikaattien eheyden takaavat protokollat* (strict consistency methods)

Protokollissa taataan, että replikaatit ovat eheässä tilassa millä tahansa hetkellä.

- *replikaattien eheyttä takaamattomat protokollat* (loose consistency methods)

Protokollissa taataan, että replikaatit ovat eheässä tilassa tietyn ajan kuluttua.

ROWA-protokolla (Read-One/Write-All) on eräs replikaattien eheyden takaava protokolla (Özsu ym., 1996a, s. 11). Siinä loogiseen tietoalkioon kohdistunut lukuoperaatio kohdistetaan yhteen fyysiseen tietoalkioon. Loogiseen tietoalkioon kohdistunut kirjoitusoperaatio muutetaan puolestaan kaikkiin fyysisiin tietoalkioihin kohdistuvaksi. Rakenteeltaan protokolla on siis yksinkertainen. Sen heikkoutena on kuitenkin se, että kaikkien fyysisten tietoalkioiden täytyy olla saatavilla tietokantatapahtumaa vahvistettaessa. Yhden tietokantasolmun vikaantuminen voi katkaista tapahtuman suorituksen.

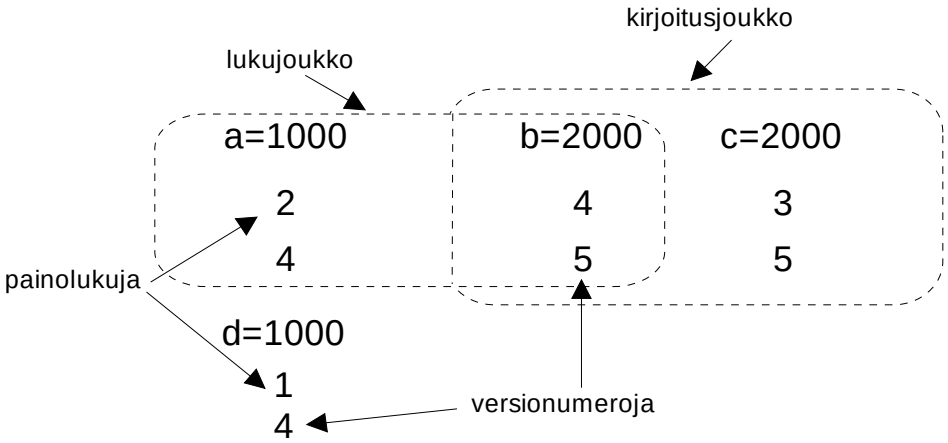
ROWA-protokollan vaihtoehdoksi on esitetty joukko protokollia, joissa fyysisten tietokopioiden saatavuusvaatimuksista tingitään tietokantatapahtumaa päätettäessä (Özsu ym., 1996a, s. 11). Seuraavassa tarkastellaan kahta yleisintä protokollaa, pääkopioprotokollaa ja äänestykseen perustuvaa joukkopohjaista protokollaa[5] (quorum consensus protocol).

Pääkopioprotokollassa (primary copy protocol) yksi fyysisistä tietoalkioista valitaan pääkopioksi, johon kaikki päivitysoperaatiot kohdistetaan (Simon, 1996, s. 186). Päivitysoperaation katsotaan päättyneen, kun pääkopio on päivitetty. Tietokantasolmu, jossa pääkopio sijaitsee, suorittaa toissijaisten kopioiden päivityksen haluamansa ajan kuluttua. Mikäli päivitys suoritetaan tapahtuman vahvistuksen jälkeen, on kyseessä *viivästetty päivitys* (delayed update propagation). Tällöin ei voida kuitenkaan enää varmistua tietokantatapahtuman ominaisuuksien täyttymisestä (Date, 1995, s. 610). Lukuoperaatiot voidaan Simonin (1996) mukaan kohdistaa mihin tahansa fyysiseen tietoalkioon, sillä protokolla takaa replikaattien eheyden. Tämä seikka ei kuitenkaan pidä paikkaansa, sillä viivästettyä päivitystä käytettäessä replikaatit eivät ole koko ajan

[5] Termistä ”quorum consensus protocol” ei ole esitetty suomenkielista käännöstä.

eheässä tilassa (Date, 1995). Toissijaisiin kopioihin kohdistettu lukuoperaatio voi palauttaa vanhentunutta tietoa. Pääkopioprotokollan etuna on sen käsitteellinen yksinkertaisuus. Myös lukuoperaatioiden vasteajat ovat lyhyitä, mikäli operaatiot voidaan kohdistaa toissijaisiin kopioihin. Eduistaan huolimatta pääkopioprotokolla ei takaa replikaattien ristiriidattomuutta tietoliikenneverkon jakaannuttua osiin häiriöiden vuoksi. Tämä johtuu siitä, että tietoliikenneverkon eri osat voivat pitää pääkopiona eri tietoalkiota. Protokollasta on esitetty laajennettuja toteutuksia, joissa tietoliikenneverkon osittumista hallitaan ylläpitämällä järjestelmänlaajuista näkymää replikointi-kokoonpanosta (Liu, 1994a, s. 3).

Äänestykseen pohjautuva joukkopohjainen protokolla (quorum consensus protocol) takaa replikaattien eheyden myös tietoliikenneverkon jakaannuttua osiin (Liu, 1994a) Protokollassa tietoalkion x jokaiselle replikaatille annetaan positiivinen painoluku. Tietoalkion x luku- ja kirjoitusoperaatiolle annetaan kynnysluvut RT ja WT siten, että 2*WT ja (RT+WT) ylittävät tietoalkion kaikkien replikaattien yhteenlasketun painoluvun. Tietoalkion x lukujoukko on mikä tahansa replikaattien joukko, jonka yhteenlaskettujen painolukujen summa on vähintään RT. Tietoalkion x kirjoitusjoukko on mikä tahansa replikaattien joukko, jonka yhteenlaskettujen painolukujen summa on vähintään WT. Jokaisella kirjoitusjoukolla on lisäksi vähintään yksi yhteinen replikaatti kaikkien lukujoukkojen ja muiden kirjoitusjoukkojen kanssa. Kirjoitusoperaatiota suoritettaessa tietoalkioon x operaatio kohdistetaan kaikkiin tietyn kirjoitusjoukon replikaatteihin. Tietoalkio x lukuoperaatiossa luetaan kaikki tietyn lukujoukon replikaatit. Jokaisessa replikaatissa ylläpidetään lisäksi versionumeroa. Luku- tai kirjoitus-operaatioissa suurimman versionumeron sisältävä replikaatti luku- tai kirjoitusjoukossa on ajantasaisin. Kirjoitusoperaatiossa versionumeroa kasvatetaan. Äänestykseen pohjautuvan joukkopohjaisen protokollan haittana pidetään tieto-objektin lukemisessa tarvittavia useita lukuoperaatioita. Kuvassa 11 esitetään esimerkki äänestykseen pohjautuvasta joukkopohjaisesta protokollasta.



Yhteenlaskettu painoluku: 2+4+3+1=10
Asetetut kynnysluvut:
WT=6, sillä 2xWT=2x6=12>10
RT=5, sillä RT+WT=5+6=11>10

Päivitysoperaatiossa kirjoitusjoukon muodostaa esimerkiksi b ja c, sillä
b:n painoluku + c:n painoluku = 4+3=7>6

Lukuoperaatiossa lukujoukon muodostaa esimerkiksi a ja b, sillä
a:n painoluku + b:n painoluku = 2+4=6>5
Suurin versionumero b:llä, joten palautettava arvo=2000

Kuva 11. Esimerkki äänestykseen pohjautuvasta joukkopohjaisesta protokollasta.

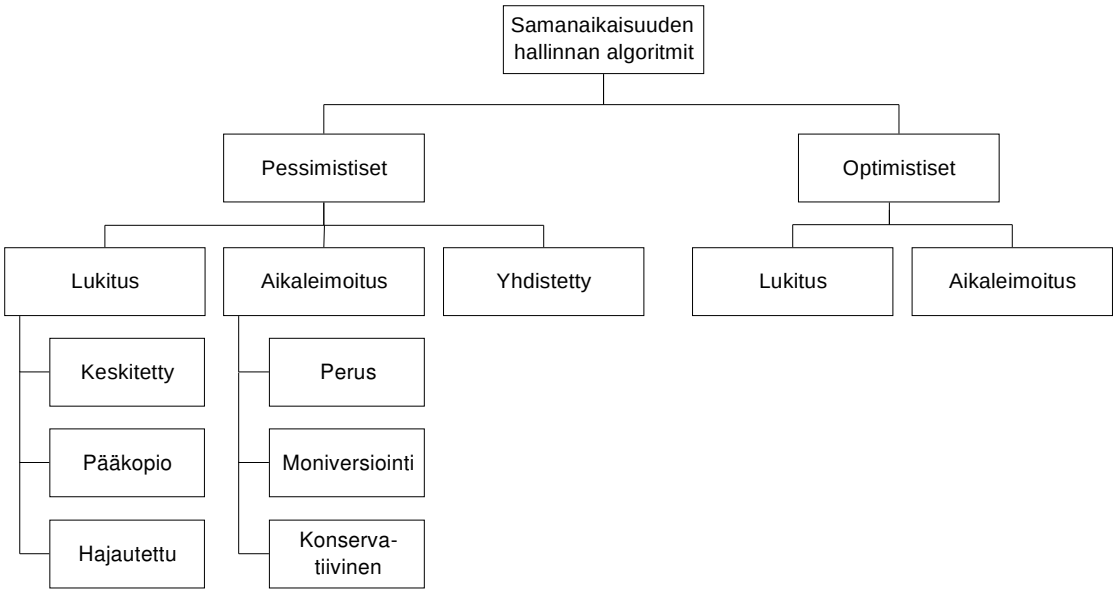
2.5.5 Samanaikaisuuden hallinnan menetelmät

Samanaikaisuuden hallinnassa käytettäviä menetelmiä voidaan luokitella lukuisilla eri perusteilla. Tietokannan hajautustapa on eräs peruste. Eräät käytetyistä menetelmistä toimivat replikoidun, eräät taas ositetun tietokannan yhteydessä. Myös verkon topologian mukainen jaottelu on mahdollinen. Käytetyin luokittelu on kuitenkin tietokantatapahtumien tahdistamistapa. Tämän luokittelun mukaan menetelmät on jaettavissa kahteen luokkaan:

- menetelmät, jotka estävät tietoalkioitten samanaikaisen käytön (lukitus),
- menetelmät, jotka pyrkivät järjestämään tapahtumien suorituksen sääntöjoukon avulla.

Ensimmäistä luokkaa kutsutaan *pessimistiseksi samanaikaisuuden hallintatavaksi*. Siinä tahdistus suoritetaan tietokantatapahtuman alkuvaiheessa. Toista luokkaa kutsutaan *optimistiseksi samanaikaisuuden hallinnaksi*. Siinä tahdistusta viivästetään tietokanta-tapahtuman päättymisen läheisyyteen.

Kuvassa 12 esitetään Özsun ym. (1991, s. 284) kehittämä jaottelu samanaikaisuuden hallinnan algoritmeista. Lukituspohjaisissa algoritmeissa tietokantatapahtumien tahdistus suoritetaan asettamalla lukkoja tarvittaviin tietokannan osiin. Keskitetyssä lukitusratkaisussa yksi tietokantasolmu ylläpitää lukitustaulua ja vastaa lukkojen myöntämisestä tapahtumille. Pääkopioratkaisussa yksi replikaateista (mikäli replikaatteja on olemassa) valitaan pääkopioksi, jonka tapahtuma lukitsee päivityksen yhteydessä. Mikäli käytössä ei ole replikaatteja, vastuu lukituksen hoitamisesta jaetaan tietokantasolmujen kesken. Hajautetussa lukitusratkaisussa vastuu lukituksen hoitamisesta on jaettu eri tietokantasolmujen kesken. Aikaleimoituksessa tapahtumat tahdistetaan antamalla kullekin tapahtumalle ja tietoalkiolle yksiselitteinen aikaleima. Yhdistetyssä ratkaisussa käytetään sekä lukitusta että aikaleimoja. Optimistisissa samanaikaisuuden hallinnan algoritmeissa oletetaan, että yhtäaikaa suoritettavat tapahtumat eivät ole keskenään ristiriidassa. Varsinainen tarkistus jätetään tapahtuman päättymisen yhteyteen.



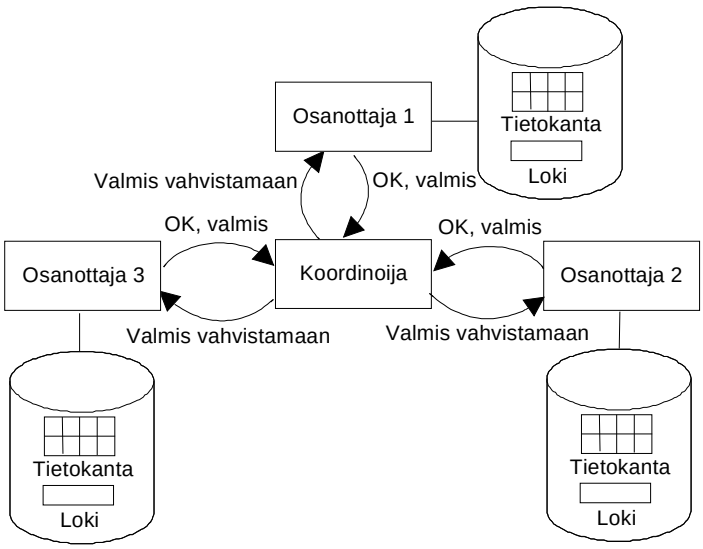
Kuva 12. Samanaikaisuuden hallinnan algoritmit (Özsu ym., 1991, s. 284).

2.5.6 Tapahtumanvahvistuksen protokollat

Hajautetussa tietokantaympäristössä tietokantatapahtuman jakamattomuuden turvaaminen on ensisijaisen tärkeää. Kaikkien tietokantatapahtuman suoritukseen osallistuvien tietokantasolmujen täytyy päätyä yhteiseen ratkaisuun tapahtumaa päätettäessä. Ratkaisu voi olla joko tietokantatapahtuman vahvistaminen (commit) tai sen kumoaminen (abort). Eri tietokantasolmut eivät saa missään tilanteessa, häiriöidenkään ilmetessä, päätyä vahvistuksessa erilaisiin ratkaisuihin. Tätä samaan lopputulokseen päätyvää tapahtumanvahvistusta varten on kehitetty tapahtumanvahvistuksen protokollia (commit protocols), joista seuraavassa esitetään kaksivaiheinen vahvistus (two-phase commit). Tarkastelu pohjautuu Simonin (1996) kuvaukseen protokollan toiminnasta.

Kaksivaiheisessa vahvistuksessa (two-phase commit, 2PC) tietokantatapahtuman suoritukseen osallistuneet tietokantasolmut päättävät yhdessä tapahtuman vahvistuksesta tai sen kumoamisesta. Yksi tietokantasolmuista valitaan tapahtuman *koordinoijaksi.* Muut tapahtuman suoritukseen osallistuvat tietokantasolmut ovat *osanottajia.* Jokainen osanottaja ilmoittaa tietokantatapahtumaan osallistumisestaan koordinoijalle ja vastaa tapahtuman toipumisessa käytettävien tietokantalokien ylläpidosta. Osanottajat suorittavat paikallisia tietokantaoperaatioita hajautetun tietokantatapahtuman puolesta aina siihen saakka, kun koordinoija pyytää tapahtuman vahvistusta tai kumoamista. Tällöin alkaa kaksivaiheisen vahvistuksen ensimmäinen vaihe. Kuvassa 13 esitetään, kuinka koordinoija kysyy osanottajilta, voivatko ne vahvistaa tietokantatapahtuman. Jokainen osanottaja kirjoittaa toipumisessa tarvittavat tiedot levyllä olevaan lokiin ja vastaa myöntävästi viestillä ”OK, valmis”. Mikäli osanottaja ei pysty vahvistamaan tapahtumaa, se pyytää tapahtuman kumoamista, jolloin koordinoija voi kumota tapahtuman lähettämällä ”Kumoa” viestin kaikille osanottajille.

Saatuaan kaikilta osanottajilta myöntävän vastauksen tapahtuman vahvistamiseksi koordinoija aloittaa kaksivaiheisen vahvistuksen toisen vaiheen. Koordinoija lähettää kaikille osanottajille ”Vahvista” viestin. Osanottajat tekevät tietokantamuutoksista

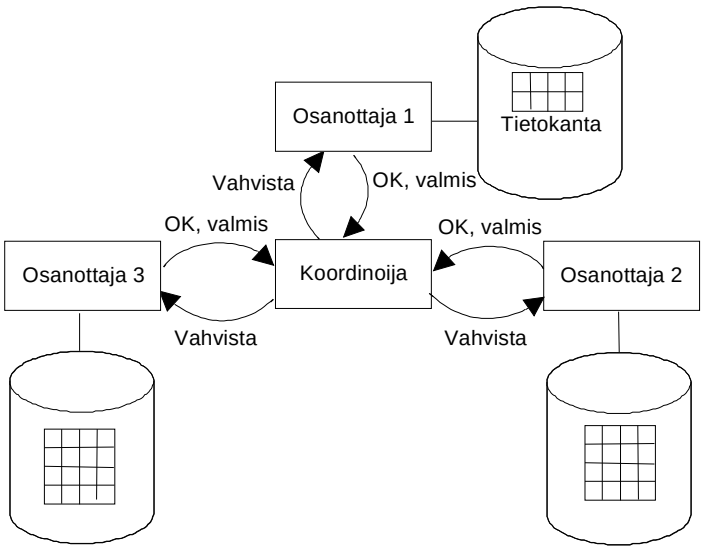


Kuva 13. Kaksivaiheinen vahvistus: vaihe 1 - tapahtuman vahvistukseen valmistautuminen (Simon, 1996, s. 269).

pysyviä ja lähettävät koordinoijalle ”OK, valmis” viestin. Koordinoija ilmoittaa asiakkaalle[6] tapahtuman vahvistamisesta. Jos jokin osanottaja ei pysty vahvistamaan tapahtumaa, koordinoija lähettää kaikille osanottajille ”Kumoa” viestin ja ilmoittaa asiakkaalle tapahtuman kumoamisesta. Kuvassa 14 esitetään kaksivaiheisen vahvistuksen toisen vaiheen eteneminen.

Tietoliikenneverkossa tai tietokantasolmuissa ilmenevien häiriöiden vuoksi osanottajat ja koordinoija eivät voi aina lähettää toisilleen protokollan edellyttämiä viestejä. Koordinoija voi todeta osanottajan menneen epäkuntoon aikakatkaisumekanismia (time-out mechanism) käyttäen. Jos jokin osanottaja ei vastaa tietyn ajan kuluessa, koordinoija voi pyytää muita osanottajia kumoamaan tapahtuman. Tilanne on monimutkaisempi, jos koordinoija ja yksi osanottajista menee epäkuntoon. Muut osanottajat eivät voi tällöin päättää tapahtuman lopputuloksesta ja vapauttaa lukkojaan, vaan niiden on odotettava, kunnes koordinoija tai vikaantunut osanottaja toipuu. Kaksivaiheisen vahvistuksen

[6] Asiakkaalla tarkoitetaan tässä yhteydessä tietokantatapahtuman suorituksen käynnistänyttä asiakasohjelmaa.



Kuva 14. Kaksivaiheinen vahvistus: vaihe 2 - tapahtuman vahvistaminen (Simon, 1996, s. 269).

sanotaankin olevan häiriötilanteissa luonteeltaan poissulkeva (blocking). Tällä tarkoitetaan sitä, että kontrolli palautuu asiakkaalle vasta kaksivaiheisen vahvistuksen päätyttyä. Suorituskyvyn kannalta poissulkevuus on haitallista, sillä palvelinten resursseja pidetään varattuina pidempään. Poissulkevuuden hallitsemiseksi on kehitetty kolmivaiheinen vahvistus (three-phase commit, 3PC), jota ovat esitelleet muiden muassa Simon (1996, s. 271), Özsu ym. (1991, s. 364) ja Khoshafian ym. (1991, s. 516).

2.6 Yhteenveto

Tässä luvussa on tarkasteltu hajautettua tietojenkäsittelyä. Tieto, sen esittäminen käyttäjälle ja käsittelysäännöt ovat kolme keskeisintä komponenttia, joiden keskittämisestä tai hajauttamisesta joudutaan päättämään hajautettua tietojärjestelmää suunniteltaessa. Erilaisina tiedon hajautuksen vaihtoehtoina kuvattiin keskitetty tiedonhallinta, osittaminen, ote, replikointi ja tiedon tallentaminen välimuistiin. Vaihtoehtojen vertailuun esitettiin muutamia kriteerejä.

Hajautetuilla tietokannoilla tarkoitetaan kokoelmaa toisiinsa liittyviä tietokantoja, jotka on hajautettu tietoliikenneverkkoon. Tietokannan hallintaan on kehitetty hajautettu tietokannan hallintajärjestelmä, jonka keskeisimmät komponentit kuvattiin. Ratkaisun etuina mainittiin mahdollinen suorituskyvyn ja saatavuuden parantuminen. Tietokannan hallinnan monimutkaisuus on kuitenkin viivästyttänyt toteutuksen yleistymistä. Koska teknologia on eräiltä osiltaan nuorta, esiintyy tietokantatuotteissa piirteitä, jotka tulevat muuttumaan lähivuosina.

Hajautetussa tapahtumankäsittelyssä tarkastellaan tietokantatapahtumia hajautetussa ympäristössä. Keskitetyltä tietokantatapahtumalta vaaditaan jakamattomuutta, ristiriidattomuutta, eristyvyyttä ja säilyvyyttä. Nämä ominaisuudet ovat tärkeässä roolissa myös hajautetussa ympäristössä. Tietokantatapahtumien samanaikaiseen suorittamiseen, vahvistukseen ja virhetilanteista toipumiseen on kehitetty protokollia ja algoritmeja, joista esiteltiin muutamia. Replikoinnin hallinnan protokollien avulla pyritään turvaamaan, että käyttäjä saa käyttöönsä ristiriidatonta tietoa.

3 REPLIKOINTIOMINAISUUKSIEN ARVIOINNIN VIITEKEHYS

Tässä luvussa esitetään tietokantatuotteiden replikointiominaisuuksien arvioinnin viitekehys, jota tullaan jatkossa soveltamaan tietokantatuotteiden arviointiin. Replikointia tarkastellaan neljästä eri näkökulmasta, joista kukin sisältää joukon arviointikriteerejä.

Luvun alussa kuvataan viitekehyksen taustaa ja tarkastellaan kehyksen käyttöä. Sen jälkeen, kohdassa 3.3, esitetään viitekehyksen yleisrakenne. Kohdassa 3.4 kehyksen arviointikriteerit määritellään yksityiskohtaisesti. Luvun yhteenveto on kohdassa 3.5.

3.1 Taustaa replikointiominaisuuden arvioinnille

Watterson (1996, s. 62) kuvaa replikointiin liittyvää käsitteiden runsautta seuraavasti:

> Replikoinnin maailmaan tutustuttaessa ei kulu kauankaan, ennen kuin henkilö hukkuu terminologiaan: symmetrinen replikointi, kaksisuuntainen replikointi, vertaisreplikointi (peer-to-peer), ”päivitä missä haluat”-replikointi (update-anyware), synkroninen replikointi, tietokantalokeihin, tilannevedokseen ja otteeseen perustuva replikointi, tapahtumapohjainen replikointi, ”vyöryttävä”-replikointi (cascading), sanomapohjainen replikointi ja jopa ”matkalaukku”-replikointi (briefcase). Tämä riittää tekemään tavallisen selväjärkisen tietohallintoihmisen hulluksi.

Replikoinnin termistö vaikuttaa siis hyvin runsaalta. Tämä on tyypillistä muillekin uusille teknologioille - uusia termejä syntyy alati, sillä uusia asioita kuvaavia sanoja ei yksinkertaisesti ole olemassa. Termeissä esiintyy kuitenkin myös päällekkäisyyksiä esimerkiksi siksi, että tuotteiden valmistajat voivat vältellä kilpailijan termien käyttöä

markkinoinnissaan. Asiakkaalle tällainen tilanne on usein hämmentävää. Aika ei riitä kaikkien termien selvittämiseen. Tilanne on hankala varsinkin ostopäätöstä tehtäessä, jolloin tuotteita olisi kyettävä vertailemaan. Tässä tutkimuksessa replikointia on pyritty tarkastelemaan neutraalisti ilman valmistajakohtaista terminologiaa.

Replikoinnin arvioinnin tarve on kasvanut selvästi. Replikointi ei rajoitu pelkästään tapahtumankäsittelyjärjestelmiin (online transaction processing, OLTP) vaan sitä käytetään muissakin yhteyksissä. Internetin nimipalvelimet (name server), sähköpostipalvelimet ja dokumenttien hallintajärjestelmät replikoivat (Watterson, 1996). Ominaisuutta voidaan käyttää taustalla myös esimerkiksi tietovarastointiratkaisuissa. Replikoinnin hallinnan protokollia on arvioitu runsaasti (mm. Ceri ym., 1994; Wiesmann ym., 2000a; Jiminez-Peris ym., 2001). Kaupallisten tietokantatuotteiden replikointia on myös arvioitu (mm. Stacey, 1995; Dobson, 1996; Dobson, 1995; Richman, 1996; Radosevisch, 1995; LaMonica, 1996; Baum, 1994; Eckerson, 1995; Watterson, 1996). Tietokantatuotteiden arviointiin liittyvissä tutkimuksissa esitetään kuitenkin vain muutamia arviointitekijöitä, joten päätöksenteko pelkästään niiden perusteella on mahdotonta. Vaikka muutamissa artikkeleissa esitetäänkin hyviä ohjeita replikointiympäristön suunnitteluun (esim. Watterson, 1996), häiritsevät kaupalliset vivahteet olennaisen aineksen erottamista.

Replikointi käsitetään tässä tutkimuksessa yhdeksi tietokannan hallintajärjestelmän[7] ominaisuudeksi. Varmuuskopiointi, toipuminen ja samanaikaisuuden hallinta ovat esimerkkejä muista tietokannan hallintajärjestelmiin sisällytetyistä ominaisuuksista. Koska tietokantatuotteisiin sisällytetyt replikointiominaisuudet vaihtelevat toimintaperiaatteiltaan ja tuotteet on osittain suunnattu erilaisiin käyttötarkoituksiin (Stacey, 1995; Schussel, 1996), ei replikointia voida ohittaa arvioinnissa pelkästään vastaamalla, onko tuotteessa replikointi vai ei. Lisäksi replikoinnin toteuttaminen ja ylläpito vaativat,

[7] Tutkimuksen ulkopuolelle rajataan muiden kuin tietokantatoimittajien valmistamat erilliset replikointiohjelmat (esim. Computer Associates Advantage Data Transformer).

ainakin toistaiseksi, huomattavan työpanoksen. Vaikutukset tuntuvat tietohallinnon tasolta aina tietokantasuunnitteluun ja järjestelmän ylläpitoon. Niinpä täsmällisemmässä arvioinnissa tulisikin miettiä, minkä ohjelmien[8] käyttämistä ominaisuuden hyödyntäminen vaatii, kuinka monimutkainen toteutus on ja millainen prosessi organisaation on läpikäytävä, jotta replikointiratkaisu toimisi. Replikointiominaisuus on sen vuoksi laajennettava kattamaan kaikki ne tietokannan hallintajärjestelmän ohessa toimivat ohjelmat, joiden käyttämistä replikoinnin suunnittelu, toteuttaminen ja ylläpito vaatii.

Erilaisten ohjelmistotuotteiden arviointiin ja valintaan on kehitetty lukuisia menetelmiä (mm. Bacon, 1992; Kitchenham, 1996; Mosley, 1992; Powell ym., 1996; Rowley, 1993 ja Franch ym., 2003). Näistä menetelmistä osa soveltuu yleiskäyttöisinä minkä tahansa ohjelmistotuotteen arviointiin, osaa menetelmistä käytetään määrätyn sovellustyypin arviointiin (esim. CASE:n valinta). Menetelmät sisältävät usein erilaisia näkökulmia (esim. tietohallinto, ohjelmoija ja käyttäjä), joita kutakin on edelleen tarkennettu yksityiskohtaisiksi arviointikriteereiksi tai mittareiksi. Vaikka menetelmissä keskitytäänkin yleensä koko tuotteeseen, voitaisiin niitä käyttää soveltuvin osin myös replikointiominaisuuden arviointiin. Replikoinnin hallintaohjelmien käytettävyyttä voitaisiin esimerkiksi selvittää käytettävyystutkimuksella. Tässä tutkimuksessa ei kuitenkaan sovelleta olemassa olevia arviointi- ja valintamenetelmiä, sillä ne eivät sovellu sellaisenaan replikoinnin arviointiin. Menetelmät ovat liian epätarkkoja replikointiin liittyvän teknisen tiedon arviointiin. Lisäksi ne soveltuvat huonosti replikointiympäristön toteutusprosessin arviointiin. Tietokantatuotteiden ja tiedon hallintaohjelmien vertailun apuvälineistöstäkään ei ole löydettävissä soveltuvaa välinettä arviointiin. Sen vuoksi jatkossa esitetään viitekehys replikointiominaisuuden arviointiin. Laaditussa viitekehyksessä on pitäydytty näkökulma-ajattelussa siten, että kutakin näkökulmaa tarkennetaan edelleen yksityiskohtaisiksi arviointikriteereiksi.

[8] Erillistä ohjelmaa ei välttämättä ole, mikäli replikointi on toteutettu tietokannan hallintajärjestelmään.

3.2 Viitekehyksen käyttötarkoitus

Replikointiominaisuuksien arvioinnin viitekehys on tarkoitettu käytettäväksi joko tietokannan hallintajärjestelmän valintaprosessin yhteydessä tai jo hankitun tietokannan hallintajärjestelmän replikointiominaisuuden arvioinnin yhteydessä. Kehystä ei varsinaisesti ole suunniteltu tietokannan hallintajärjestelmän valintatehtävään, eikä se näin ollen sisällä ohjelmistojen valintamenetelmissä usein käytettyä painokerroin-laskentaa. Painokertoimien määrittely jää kehyksen soveltajan tehtäväksi. Sen sijaan kehyksessä pyritään esittämään mahdollisemman laaja kriteeristö, joka toimii tarkistuslistana arviointiprosessissa. Kehyksen soveltaja käy läpi tuotteen piirteet kriteeristön mukaan ja kirjaa muistiin havainnot tietokantatuotteesta. Tehtävien suoritus voidaan jakaa eri henkilöille.

Viitekehyksen soveltamisella voidaan saavuttaa seuraavia hyötyjä:

- Organisaatio voi tiedostaa nykyisiä ja ennakoida tulevia replikointitarpeitaan ennen tietokantatuotteen hankintaa.
  Viitekehyksestä voidaan koota kysymyslistoja esimerkiksi tietokannan hallintajärjestelmän hankintaneuvotteluihin. Lisäksi laitteisto- ja ohjelmistovaatimuksia voidaan ennakoida paremmin.
- Replikointiympäristön suunnittelu tehostuu.
  Suunnittelussa voidaan edetä kehyksen esittämässä järjestyksessä, jolloin kehys toimii tehtävien suorituksen muistilistana. Myös suorituskyvyn rajoitteita ja toteutuksen vaatimaa työmäärää voidaan ennakoida paremmin.
- Replikointiympäristön toteutus tehostuu.
  Tietokannan hoitajat ja ohjelmoijat voivat palauttaa mieleen tuotteen ominaisuuksia kehyksen avulla.
- Selkeyttää ja yhtenäistää organisaatiossa käytettävää replikointikäsitteistöä.
  Kehyksen soveltamisen yhteydessä voidaan sopia, mitä käsitteitä käytetään jatkossa.

Pelkkä arviointikriteeristöön tutustuminen helpottaa tiedon poimimista esitteistä ja ohjekirjoista. Lisäksi viitekehys toimii johdatuksena replikointiin, joten keskeisten käsitteiden omaksuminen helpottaa kommunikointia asiakkaiden ja tietokantatoimittajien kanssa. Samalla voidaan saada ideoita replikoinnin soveltamismahdollisuuksista erilaisiin tilanteisiin (esim. data warehousing -ratkaisu).

Mitä viitekehys ei ole ja mihin sitä ei ole tarkoitettu:

- Kehys ei anna yksiselitteistä vastausta tietokantatuotteiden replikointiominaisuuksien paremmuudesta. Kehyksessä pyritään pikemminkin antamaan soveltajalle välineitä ominaisuuden arviointiin.
- Kehys ei ole varsinainen suorituskyvyn mittausväline. Tässä tutkimuksessa tuotteiden äärimmäistä suorituskykyä ei ole tutkittu monimutkaisten käytännön järjestelyjen vuoksi. Eräät tuotteet sisältävät laskentamalleja suorituskyvyn ja asennusvaatimusten laskemiseen (mm. tietokantatapahtumia aikayksikössä, maksimaalinen replikointi-yhteyksien määrä ja verkon kuormitus). Luvussa 4 on kuitenkin esitetty arvioita kahden tietokantatuotteen replikoinnin suorituskyvystä.
- Kehys ei ole tarkoitettu erillisten replikointiohjelmien vertailuun.

Replikointiominaisuuksien arvioinnin viitekehys on tarkoitettu organisaation tietohallintohenkilöstön käyttöön. Tietohallintopäälliköille ensisijaisia kysymyksiä ovat replikoinnin soveltuvuus organisaation sovellus- ja laitteistoarkkitehtuuriin, replikoinnin toteuttamisesta ja ylläpidosta aiheutuvat kustannukset ja sillä saatava hyöty sekä toimittajan tuki. Tietokannan hoitaja voi saada kehyksestä tukea replikointiympäristön suunnitteluun, toteutukseen ja ylläpitoon. Lisäksi hänen vastuullaan on myös arvioida replikoinnin luotettavuutta, toimintavarmuutta, suorituskykyä ja näkyvyyttä peruskäyttäjille (end-user). Viitekehyksen käyttöä näihin tehtäviin suositellaan. Sovellusohjelmoija voi kehyksen avulla saada nopeasti yhteisen käsityksen replikoinnista tietokannan hoitajan kanssa ja tällä tavoin nopeammin oppia replikointiympäristön toteuttamisessa mahdollisesti tarvittavat ohjelmointitehtävät.

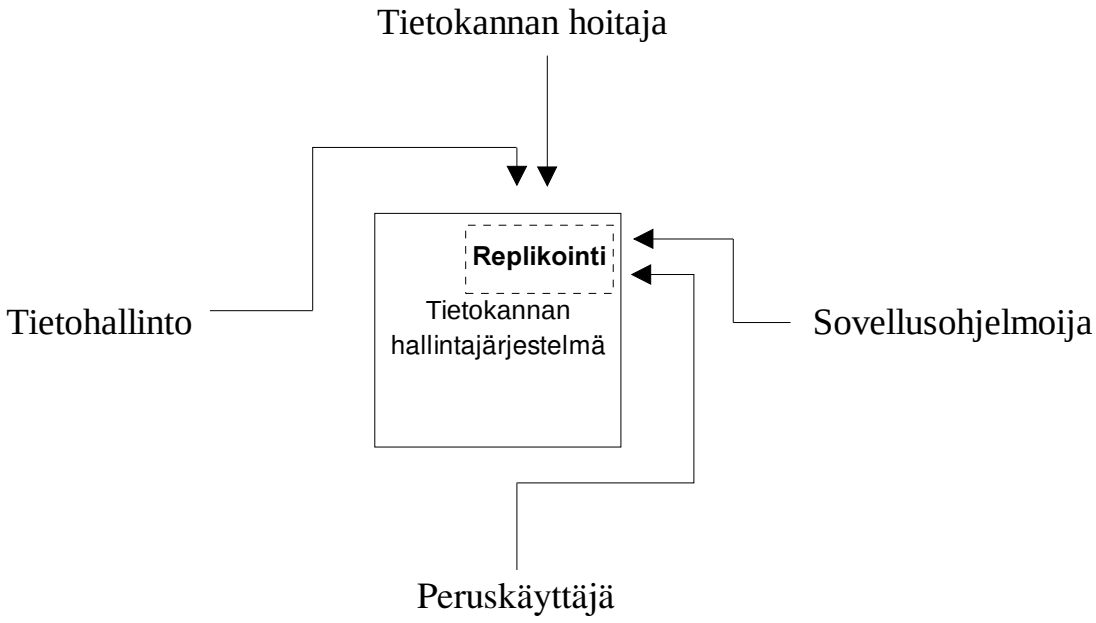
Viitekehyksen soveltaminen kokonaisuudessaan vaatii kontakteja tietokantatoimittajan kanssa, tuotteeseen ja sen ohjeistukseen tutustumista sekä kustannus/hyöty -analyysin osalta organisaation nykyisen tietojenkäsittelyn ja tulevien replikointitarpeiden tuntemusta. Tietokantatuotteen esitteet ja ohjekirjat luovat pohjan arvioinnille. Yksityiskohtainen kriteerien selvittäminen voidaan pääosin tehdä tuotetta asentamatta, tosin koekäyttö antaa paremman kuvan ominaisuudesta. Kehyksen kustannus/hyöty -analyysi poikkeaa tavanomaisten liiketoimintaa tukevien sovellusten (esim. tilausten hallintajärjestelmän, myynnin tukijärjestelmän) analyysista, sillä replikointi ei ole sellaisenaan liiketoimintaa tukeva sovellus. Replikointia voidaan käyttää taustalla perusratkaisuna tällaisten sovellusten laatimisessa. Kustannusten laskemisessa voidaan käyttää tietokantatoimittajien antamaa perustietoa. Replikoinnista saatavat hyödyt tulevat puolestaan niistä tavoista, joilla organisaatio soveltaa teknologiaa. Näin ollen kehyksessä esitetäänkin ainoastaan muutamia kriteerejä, joilla mahdollisia hyötyjä voidaan kartoittaa. Mikäli kehystä sovelletaan tuotevertailuun, kehyksen soveltaja päättää, täytyykö hyötyjä tarkastella tuotekohtaisesti. Viitekehyksen soveltamisessa voidaan tilannekohtaisesti painottaa eri näkökulmia ja tarvittaessa jättää osa näkökulmista tarkastelun ulkopuolelle.

3.3 Viitekehyksen yleisrakenne

Tässä kohdassa kuvataan replikointiominaisuuksien arvioinnin viitekehyksen rakennetta. Aluksi esitetään, millaisista näkökulmista replikointia voidaan tarkastella, ja kohdan lopuksi kuvataan, mitä tehtäviä näkökulmia edustavat henkilöt tekevät.

3.3.1 Näkökulmat tietokannan replikointiin

Kuvassa 15 esitetään replikointiominaisuuksien arvioinnin viitekehyksen neljä näkökulmaa: 1) tietohallinto, 2) tietokannan hoitaja, 3) sovellusohjelmoija ja 4) peruskäyttäjä. Näkökulmien valinta perustuu Daten (1990) esittämään tiedonhallintaohjelmien käyttäjäluokitteluun: tietokannan hoitaja, sovellusohjelmoija ja peruskäyttäjä. Lisäksi erääksi näkökulmaksi on valittu tietohallinto. Näkökulmien valinnalla korostetaan sitä, ettei replikointi ole ainoastaan tekninen, tietokannan hoitajalle kuuluva yksityiskohta, vaan sen tarjoamien mahdollisuuksien ja vaikutusten arvioinnissa on huomioitava myös tietohallinto ja peruskäyttäjät. Viitekehyksen painopiste on kuitenkin tietokannan hoitajan näkökulmassa, sillä replikointi vaikuttaa eniten juuri hänen tehtäviinsä. Tässä näkökulmassa on mukana ajallinen ulottuvuus suunnittelusta, toteutukseen ja käyttöön.

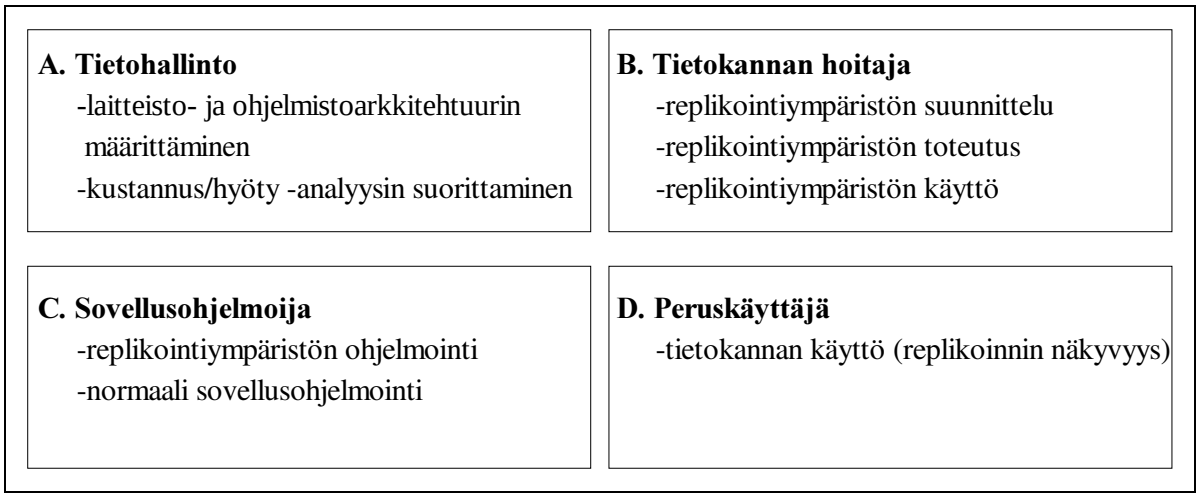


Kuva 15. Replikoinnin arvioinnin viitekehyksen näkökulmat.

3.3.2 Replikointiin liittyvät tehtävät

Toimivan replikointiratkaisun suunnittelu, toteuttaminen ja käyttö vaatii useiden tehtävien suorittamista. Tietohallinto pohtii, kuinka replikointi soveltuu yrityksen laitteisto- ja ohjelmistostrategiaan sekä mitä kustannuksia ja hyötyjä replikointiin liittyy.

Tietokannan hoitajan tehtäviin kuuluu replikointiympäristön suunnittelu, toteutus ja käyttö. Sovellusohjelmoija voi joutua toteutus- ja ylläpitovaiheessa ohjelmoimaan esimerkiksi liittymiä sellaisiin tietolähteisiin, joita tuote ei tue. Peruskäyttäjän tehtävissä replikoinnin tulisi näkyä ainoastaan nopeampina vasteaikoina, parantuneena luotettavuutena ja parempana suorituskykynä. Kuvassa 16 esitetään eräs mahdollinen tehtäväjaottelu, jota voidaan käyttää replikointiratkaisun suunnittelussa, toteutuksessa ja käytössä. Jatkossa kustakin kuvan 16 tehtäväluokasta käytetään nimitystä *pääryhmä.* Jaottelu ei ole suinkaan tarkka rajanveto eri näkökulmia edustavien henkilöiden välille, sillä esimerkiksi tietokannan hoitaja joutuu osallistumaan kustannus/hyöty -analyysiin, tietohallintoa edustavan järjestelmäpäällikön on osallistuttava replikointiympäristön suunnitteluun ja sovellusohjelmoijan on mahdollisesti osallistuttava toteutukseen.



Kuva 16. Viitekehyksen näkökulmakohtainen tehtäväjaottelu.

3.4 Viitekehyksen yksityiskohtainen rakenne

Kehyksessä esitetään valituista näkökulmista joukko arviointikriteerejä, joilla replikointiominaisuutta arvioidaan. Kriteerien taustalla ovat ne tehtävät, joita eri näkökulmia edustavat henkilöt tekevät. Kriteerien keräämisessä on käytetty useita eri lähteitä (mm. Praxis, 1996; Watterson, 1996) ja valinnassa on painotettu hajautetun tietokannan hallintaa, luotettavuutta ja suorituskykyä. Valtaosa kehyksen kriteereistä voidaan selvittää objektiivisesti, osa vaatii kehyksen soveltajan tulkintaa. Taulukoissa 2, 3, 4 ja 5 esitetään näkökulmittain viitekehyksen sisältämät kriteerit. Sen jälkeen jokaisesta kriteeristä esitetään yksityiskohtainen kuvaus.

Taulukko 2. Tietohallinnon replikoinnin arviointikriteerit.

A. Tietohallinto

Laitteisto- ja ohjelmistoarkkitehtuurin määrittäminen

A.1. Kuinka yhteensopiva tuote on muiden tuotteiden kanssa?
A.2. Kuinka kiinteästi ominaisuus on integroitu tietokannan hallintajärjestelmään?
A.3. Millaista tukea toimittaja tarjoaa?
A.4. Arvio ominaisuuden soveltuvuudesta erilaisiin käyttötarkoituksiin
A.5. Arvio ominaisuuden soveltuvuudesta aiottuun tehtävään

Kustannus/hyöty -analyysi

A.6. Kustannukset
- A.6.1. Replikointiohjelma
- A.6.2. Laitteisto
  - A.6.2.1. Levytila
  - A.6.2.2. Lähdejärjestelmän päivitys
  - A.6.2.3. Kohdejärjestelmän päivitys
- A.6.3. Tietoliikenneverkko
- A.6.4. Asennus ja toteutus
- A.6.5. Koulutus

A.7. Hyödyt
- A.7.1. Tiedon paikallinen saanti
  - A.7.1.1. Vasteaikojen nopeutuminen
  - A.7.1.2. Saatavuuden paraneminen
  - A.7.1.3. Prosessori- ja verkkopäivitysten siirtäminen
- A.7.2. Kopioidenhallintakustannusten aleneminen
- A.7.3. Replikoinnin tarjoamat mahdollisuudet
  - A.7.3.1. Uuteen teknologiaan siirtyminen
  - A.7.3.2. Tietoturvan paraneminen

Taulukko 3. Tietokannan hoitajan replikoinnin arviointikriteerit.

B. Tietokannan hoitaja

Replikointiympäristön suunnittelu

B.1. Tuotteen teoreettinen tausta
- B.1.1. Teoreettisen taustan esittely
- B.1.2. Replikoinnin hallinnan protokolla
- B.1.3. Replikointitarpeen ilmeneminen
- B.1.4. Replikoinnin kohde
- B.1.5. Käytettävä tekniikka
- B.1.6. Ajantasaisuus
- B.1.7. Päivitysmalli

B.2. Tietokannan hoitajalta edellytettävät taustatiedot

B.3. Ohjeistuksessa mahdollisesti olevat esimerkkisovellukset

B.4. Asennusvaatimukset

B.5. Suorituskyvyn ennakointi

Replikointiympäristön toteutus

B.6. Tuotteen asennus

B.7. Replikoinnin hallintaohjelma
- B.7.1. Ohjelman yleiset ominaisuudet
- B.7.2. Liittymät tietolähteeseen
  - B.7.2.1. Määrittäminen
  - B.7.2.2. Poistaminen

B.8. Replikaattien alustaminen

B.9. Tuotteen poistaminen

Replikointiympäristön käyttö

B.10. Replikoinnin hallintaohjelma
- B.10.1. Replikoinnin käynnistäminen ja pysäyttäminen
- B.10.2. Replikointiympäristön mukauttaminen
- B.10.3. Poikkeustilanteiden käsittely
- B.10.4. Replikointiympäristön varmistaminen
- B.10.5. Poikkeuksellisten pyyntöjen toteuttaminen
- B.10.6. Suorituskyvyn seuranta
- B.10.7. Etävalvonta

Taulukko 4. Sovellusohjelmoijan replikoinnin arviointikriteerit.

C. Sovellusohjelmoija	
Replikointiympäristön ohjelmointi	*Normaali sovellusohjelmointi*
C.1. Liittymien ohjelmointi tietolähteisiin C.2. Raportoinnin ja hälytysten ohjelmointi	C.3. Ohjelmointitehtävien huomioiminen C.4. Vertaisreplikoinnin päivityskonfliktit

Taulukko 5. Peruskäyttäjän replikoinnin arviointikriteerit.

D. Peruskäyttäjä
Replikoinnin vaikutusten näkyminen
D.1. Vasteaika D.2. Saatavuus D.3. Suorituskyky

3.4.1 Tietohallinnon arviointikriteerit

Tässä kohdassa esitetään replikointiominaisuuksien arvioinnin viitekehyksen tietohallinnon arviointikriteerit.

A. Tietohallinto

Laitteisto- ja ohjelmistoarkkitehtuurin määrittäminen

A.1. Kuinka yhteensopiva tuote on muiden tuotteiden kanssa?

Replikointia voidaan käyttää tiedon keräämiseen tietokantaan heterogeenisista, muiden kuin tietokantatoimittajan omista, tietolähteistä (Watterson, 1996). Vastaavasti voidaan replikoida myös muiden valmistajien tietokantatuotteisiin. Näissä tehtävissä replikoinnin tulisi luonnollisesti tukea mahdollisimman useita tuotteita. Vastauksena kriteeriin esitetään luettelo niistä tietokannan hallintajärjestelmistä, joita voidaan käyttää replikoinnissa.

A.2. Kuinka kiinteästi ominaisuus on integroitu tietokannan hallintajärjestelmään?

Replikointi voidaan toteuttaa joko tietokantaohjelman sisään tai sen rinnalla toimivaan erilliseen ohjelmaan. Tietokantaohjelmaan integroituna replikointi voi hidastaa tietokannan muuta käyttöä (Watterson, 1996). Ulkopuolinen ohjelma puolestaan joudutaan asentamaan jokaiseen replikointiin osallistuvaan tietokantasolmuun (Eckerson, 1995). Ihannetapauksessa käyttäjällä olisikin mahdollisuus valita, kumpaa tapaa käytetään (Watterson, 1996).

A.3. Millaista tukea toimittaja tarjoaa?

Tällä kriteerillä selvitetään, millaista tukea tietokantatoimittaja antaa replikointiympäristön elinkaaren eri vaiheisiin. Mahdollisia tukimuotoja ovat esimerkiksi puhelintuki, asiakaskäynnit, etäkonfigurointi, kirjalliset kyselyt, World Wide Web ja sähköposti. Myös tuotteen mukana tulevaan ohjeistukseen kannattaa tutustua ennakkoon.

A.4. Arvio ominaisuuden soveltuvuudesta erilaisiin käyttötarkoituksiin

Tuotteen esitteistä ja ohjeistuksesta esitetään yleensä erilaisia replikoinnin sovellusmahdollisuuksia. Niitä voidaan käyttää apuna pohdittaessa, kuinka replikointia voitaisiin hyödyntää omassa organisaatiossa. Vastauksena kriteeriin esitetään luettelo tietokantatoimittajan kuvaamista replikoinnin mahdollisista sovelluskohteista. Kriittisyys on tärkeää tämän arviointikriteerin yhteydessä.

A.5. Arvio ominaisuuden soveltuvuudesta aiottuun tehtävään

Replikointitarpeiden ja kehyksen muidenkin osien selkiinnyttyä pohditaan ominaisuuden soveltumista aiottuun tehtävään. Vastauksessa voidaan kuvata, millä edellytyksillä replikointi toimisi sovellustilanteessa.

Kustannus/hyöty -analyysi

Tietokantatuotteiden replikointiin liittyvien kustannuksien ja hyötyjen analysoimiseksi ei ole tiettävästi tehty tieteellistä tutkimusta. Kyseessä onkin erityislaatuinen tehtävä, sillä replikoinnin hyötyjen laskeminen on varsin monimutkaista. Hyödyt voivat tulla useista eri tekijöistä, joiden kaikkien ennakoiminen ja yhteismitallistaminen ei ole mahdollista. Miten esimerkiksi tulevia tiedonsiirtotarpeita osataan ennakoida siten, että tiedettäisiin, kuinka paljon replikoinnilla toteutettu ratkaisu tuottaisi? Tai kuinka osataan täsmällisesti arvioida tiedonsiirtoon aiemmin käytetyn ohjelmointityön vähenemisestä saatavaa hyötyä? Hyötyyn vaikuttaakin suuresti se, kuinka organisaatio tulee soveltamaan replikointia (Praxis, 1996c). Tehtävän monimutkaisuudesta huolimatta kustannus/hyöty -analyysiä kannattaa replikoinninkin osalta suorittaa. Seuraavassa esitetään Praxis International Inc.:n (Praxis, 1996c) laatima malli kustannus/hyöty -analyysin suorittamiseksi.

A.6. Kustannukset

A.6.1. Replikointiohjelma

Replikointiratkaisun keskeisin kustannus on itse replikointiohjelma. Toteutus voi vaatia joko tietokannan hallintajärjestelmästä erillään toimivan ohjelman hankkimista tai tietokannan hallintajärjestelmän päivittämistä. Tietokantatoimittaja osaa kertoa, kumpaa ratkaisua tuotteessa käytetään. Kustannusta laskettaessa kannattaa varmistaa, että toimittaja ymmärtää replikointitarpeen. Sen jälkeen on syytä vaatia toimittajaa kertomaan kaikki ratkaisun toteuttamiseen tarvittavat ohjelmistot hintoineen.

A.6.2. Laitteisto

Laitteistokustannusten arvioimisessa kannattaa pyytää tietokantatoimittajaa ilmoittamaan toteutuksessa tarvittava laitteistokokoonpano. Seuraavassa esitetään muutamia mahdollisia laitteistokustannuksia.

A.6.2.1. Levytila

Replikoinnissa eräs ilmeinen laitteistokustannus on useille tietokopioille tarvittava levytila. Mikäli tietokantatuotteen replikointitoteutus sallii horisontaalisen ja vertikaalisen osittamisen käytön, voidaan levytilan tarpeeseen vaikuttaa rajoittamalla replikointi vain tarvittaviin tietoalkioihin. Levytilaa tarvitaan myös lähdejärjestelmän päähän, sillä replikointiohjelma tallentaa replikoitavan tiedon yleensä väliaikaiseen tietovarastoon[9] odottamaan lähetystä kohdejärjestelmiin. Näiden tietovarastojen kokoihin vaikuttaa muun muassa replikoitaviin tietoihin kohdistuva päivitysintensiteetti ja käytettävä replikointinopeus. Levytilan tarvetta laskettaessa voidaan käyttää apuna toimittajalta mahdollisesti saatavia laskentakaavoja.

---

[9] Väliaikainen tietovarasto voi olla esimerkiksi levylle tehty päivitysjono tai tietokannan taulu, joka sisältää eteenpäin välitettävät päivitykset.

A.6.2.2. Lähdejärjestelmän päivitys

Tätä kriteeriä tarkasteltaessa on selvitettävä, lisääkö vai vähentääkö replikointiratkaisu lähdejärjestelmän kysely- ja päivitysintensiteettiä. Jos replikointia käytetään esimerkiksi korvaamaan keskitetty raportointijärjestelmä replikoimalla tieto yhteen tai useampaan kohdejärjestelmään, on selvää, että kyseinen kyselykuorma ei enää rasita lähdejärjestelmän palvelinta. Tällöin kapasiteetin lisäämisen tarvetta ei ole. Mikäli replikointia käytetään kokonaan uuden tietojärjestelmän luomiseen replikoidun tiedon pohjalta tai lähdejärjestelmän päivitysintensiteetti on hyvin suuri, voidaan palvelimen kapasiteettia joutua lisäämään.

A.6.2.3. Kohdejärjestelmän päivitys

Kohdejärjestelmän lisäkapasiteetin laskemisessa on myös tarkasteltava kysely- ja päivitysintensiteettiä. Jos kohdejärjestelmässä käytetään ainoastaan kyselysovelluksia, tarvittavan lisäkuorman laskeminen poikkeaa normaalista kuormituslaskennasta ainoastaan siten, että laskennasta on huomioitava muualta tulevien päivitysten replikaatteihin soveltamisen tuoma lisäkuormitus. Mikäli replikaatteja voi päivittää kohdejärjestelmässä, laskennasta tulee monimutkaisempaa. Tietokannan hallintajärjestelmän toimittaja voi tietää, kuinka paljon replikointi lisää tarvittavaa tietojenkäsittelykapasiteettia yhden päivitystapahtuman yhteydessä.

A.6.3. Tietoliikenneverkko

Replikointi ei yleensä lisää verkkoliikennettä keskitettyä ratkaisua hajautetulla korvattaessa. Tämä johtuu siitä, että vaikka päivitykset tuovatkin lisäkuormaa, kyselyiden aiemmin aiheuttama verkkoliikenne poistuu kokonaisuudessaan. Normaaleissa päätekäyttöisissä päivityssovelluksissa kyselyiden osuuden on arvioitu olevan jopa 90 prosenttia (Praxis, 1996c). Poikkeuksena voidaan kuitenkin pitää tilannetta, jossa sovelluksen päivitysten määrä suhteessa kyselyiden määrään on suuri. Näiden päivitysten aiheuttamat muutokset on replikoitava kaikkiin muihin tietokantasolmuihin.

Kokonaan uutta järjestelmää replikoidun tiedon pohjalle rakennettaessa verkkoliikenne luonnollisesti kasvaa.

A.6.4. Asennus ja toteutus

Asennuksen ja toteutuksen kustannusten erottaminen toisistaan on tärkeää kustannuksia laskettaessa. Asennuksella tarkoitetaan tässä yhteydessä replikointiohjelman ja mahdollisten verkkoprotokollien asennusta ja päivittämistä. Toteutuksella tarkoitetaan asennettujen ohjelmien ja tietokantojen mukauttamista halutun tehtävän suorittamiseksi. Molempien tehtävien kustannuksia on arvioitava koko järjestelmän elinkaaren ajalta.

A.6.5. Koulutus

Replikointiratkaisun tehokas käyttö edellyttää henkilöstön kouluttamista. Tietokantatoimittaja voi auttaa koulutuskustannusten arvioinnissa.

A.7. Hyödyt

A.7.1. Tiedon paikallinen saanti

Replikointi tarjoaa mahdollisuuden käsitellä tietoa paikallisesti. Seuraavassa esitetään esimerkinomaisesti muutamia tiedon paikallisesta saannista tulevia hyötyjä.

A.7.1.1. Vasteaikojen nopeutuminen

Vasteaikojen nopeutumisella saavutettava hyöty vaihtelee sovelluskohtaisesti. Vain muutaman kerran päivässä tietokantaa käyttävälle henkilölle vasteajan nopeutumisesta on tuskin hyötyä. Puhelinmyynnissä vasteaikojen nopeutuminen vaikkapa vain muutamalla sekunnilla voi merkitä suurtakin hyötyä, jos myyjiä on kymmeniä ja puheluja satoja päivittäin.

Vasteajan parantumisesta saatavaa hyötyä voidaan laskea esimerkiksi seuraavasti. Keskitetyssä ratkaisussa tieto haetaan kyselyillä verkon yli keskuskoneelta, joten vasteajassa on mukana tiedon siirrosta verkossa kulunut aika. Replikointia käytettäessä tätä siirtoviivettä ei ole. Päivittäin säästynyttä kokonaisaikaa voi selvittää esimerkiksi mittaamalla kyselyn siirtoon keskuskoneelle ja vastauksen palauttamiseen kulunut aika päivän eri aikoina, laskemalla keskimääräinen siirtoviive ja kertomalla keskiarvo päivässä suoritettavien kyselyiden määrällä. Siirtoviiveen lisäksi on huomioitava keskuskoneen ja tulevan kohdejärjestelmän väliset prosessointiteho- ja kuormituserot, sillä replikaattien sijoittaminen suorituskykynsä äärirajoilla toimivaan palvelimeen voi mitätöidä siirtoviiveen poistumisesta saatavan hyödyn.

Parantuneen vasteajan muuttaminen rahalliseksi hyödyksi on monimutkaisempaa. Kuinka antaa rahallinen arvo niille säästyneille tunneille, joita replikointi mahdollisesti tarjoaa? Eräänä vaihtoehtona on laskea hyötyä työntekijäkustannusten kautta. Vaihtoehtoisesti voitaisiin yrittää myös arvioida menetettyjen tilaisuuksien (solmittuja kauppoja, tehtyjä tuotteita jne.) arvoa.

A.7.1.2. Saatavuuden paraneminen

Jokainen käyttäjän ja hänen käsittelemänsä tiedon välillä oleva laite voi vikaantua. Verkon topologiasta riippuen välillä voi olla lähiverkkoja, kytkimiä, reitittimiä, alueverkkoja (WAN) jne. Tiedon replikointi lähelle käyttäjää poistaa osan näistä mahdollisista vikalähteistä. Mikäli jokin replikaateista ei ole tietoliikenteessä ilmenevien häiriöiden ym. laitevikojen vuoksi saatavissa, ei se estä muita käyttäjiä jatkamasta toimintaansa. Ainoastaan replikaattien yhtenäisessä tilassa pitäminen kärsii, joten tietokannan hallintajärjestelmän on vikojen poistuttua saatettava replikaatit yhtenäiseen tilaan. Seuraavassa esitetään, kuinka laitteiden vikaantumisvälien avulla voidaan laskea parantuneesta saatavuudesta koituvaa hyötyä.

1. Tutkitaan mitkä laitteisto- ja tietoliikennekomponentit eivät ole replikointiratkaisussa enää käyttäjän ja hänen käsittelemänsä tiedon välissä.

2. Selvitetään kunkin komponentin keskimääräinen vikaantumisväli (mean time between failures, MTBF) tunneissa toimittajalta tai ohjeista.
3. Arvioidaan komponenteittain korjaamiseen tai ohittamiseen kuluva aika tunneissa.
4. Lasketaan kokonaiskäyttöaika vuodessa tunteina. Esim. 24 tuntia vuorokaudessa x 365 vuorokautta = 8760 tuntia.
5. Lasketaan komponenteittain keskimääräinen katkoaika vuodessa seuraavasti: keskimääräinen katkoaika tuntia = (korjaamiseen tai ohittamiseen kuluva aika x kokonaiskäyttöaika) / keskimääräinen vikaantumisväli
6. Lasketaan kaikkien komponenttien katkoaika yhteensä.
7. Arvioidaan sovelluskohtaisesti yhden käyttötunnin rahallinen arvo.
8. Lasketaan vikojen kokonaiskustannukset vuodessa seuraavasti: vikojen kokonaiskustannukset = kaikkien komponenttien katkoaika yhteensä x keskiarvo (kaikkien sovellusten yhden käyttötunnin arvo)

Nämä odotettavat vikojen kokonaiskustannukset säästyvät siis replikointiratkaisussa. Laskennan huonona puolena on se, että laitteiden vikaantumisvälien ja sovellus-kohtaisten käyttötuntien arvon määrittäminen on hankalaa.

A.7.1.3. Prosessori- ja verkkopäivitysten siirtäminen

Tietojenkäsittelyssä käytetyillä laitteilla on taipumus aikaa myöten saavuttaa kapasiteettiensa äärirajat. Perinteisesti kapasiteetin äärirajoilla toimivien koneiden käyttöikää on jatkettu vaihtamalla tehokkaampia komponentteja (esim. prosessoreja, levyasema). Päivittäminen voi kuitenkin tulla varsin kalliiksi ja se voi olla jopa mahdotonta, mikäli laitteiston päivitysmahdollisuudet on käytetty loppuun. Replikoinnilla voidaan siirtää keskitetyn tietokantapalvelimen prosessorin ja tietoliikenneverkon päivitystä. Kun tietokopiot on sijoitettu lähelle käyttäjää, eivät kyselyt kuormita mainittuja komponentteja. Kriteerillä arvioidaan, voidaanko replikoinnilla mahdollisesti viivästyttää päivityksiä ja saavuttaa kustannussäästöä.

A.7.2. Kopioidenhallintakustannusten aleneminen

Raportoinnin, päätöstukijärjestelmien ja liiketoimintojen tietotarpeiden tyydyttäminen vaatii organisaatioissa varsin suuren työpanoksen. Näiden tehtävien suorittamisessa käytetään usein eräajoluonteista käsittelyä, joka voi jopa vaatia järjestelmien pysäyttämistä. Jos organisaatio toimii vain yhdessä maassa ja tietojenkäsittelytarpeet rajoittuvat vain normaaliin työaikaan, ei järjestely aiheuta ongelmia. Mikäli toiminta on kansainvälistä ja mukana on eri aikavyöhykkeillä toimivia yksikköjä, järjestelmiltä voidaan vaatia ympärivuorokautista saatavuutta. Replikointia voidaan käyttää tiedon kopiointiin ja korvaamaan eräajoluonteista käsittelyä. Soveltamisella voidaan saavuttaa seuraavia hyötyjä:

- Inhimillisiä erehdyksiä tapahtuu vähemmän, sillä replikointi suoritetaan automaattisesti.
- Siirrettävän tiedon määrä voi vähentyä, sillä ainoastaan muuttuneet tiedot siirretään.
- Järjestelmien käyttöaikaa voidaan pidentää, sillä replikoinnilla voidaan korvata eräajoja, joiden suorittaminen vaatii järjestelmien pysäyttämistä.

A.7.3. Replikoinnin tarjoamat mahdollisuudet

A.7.3.1. Uuteen teknologiaan siirtyminen

Organisaatiot joutuvat usein koodaamaan sovelluksia uudelleen tietojenkäsittely-laitteiston ja ohjelmaversioiden vaihtuessa. Työstä aiheutuvat kustannukset voivat olla suuria. Kokonaan uudelleen kirjoitetusta sovelluksesta saadaan hyötyä kuitenkin vasta järjestelmän käytön alettua. Eräänä vaihtoehtona on uudelleen rakennettavan sovelluksen jakaminen toimintoihin, jotka toteutetaan yksi kerrallaan budjetin niin salliessa. Ne toiminnot, joista arvioidaan saatavan nopeimmin hyötyä, toteutetaan ensin. Osa toiminnoista voidaan jättää toimimaan vanhaan laitteistoon, mikäli ne eivät hyödy uudesta teknologiasta. Vaikka menettelytapa onkin taloudelliselta kannalta järkevä, voidaan käytännön toteutuksessa törmätä teknisiin ongelmiin. Jos uusi tietokannan

hallintajärjestelmä ei ole yhteensopiva vanhan kanssa, vanhat ja uudet sovellukset eivät voi jakaa käsiteltävää tietoa. Ratkaisuna voidaan käyttää heterogeenisiä tietolähteitä tukevaa replikointia. Tällöin replikoinnin hallintaohjelma pitää vanhan ja uuden tietolähteen yhtenäisessä tilassa. Tällä kriteerillä selvitetään, voitaisiinko replikointia soveltaa organisaatiossa uuteen teknologiaan siirtymisessä ja mitä hyötyjä ratkaisulla saavutettaisiin.

A.7.3.2. Tietoturvan paraneminen

Käytössä olevat tietokannan hallintajärjestelmät tarjoavat varsin hyvät tiedon turvaamistavat. Siitä huolimatta voi joissakin tilanteissa olla tarvetta täydentää tietoturvaa. Replikointi tarjoaa tähän erään vaihtoehdon. Korkean tietoturvatason vaativat tiedot voidaan pitää tietokannan hallintajärjestelmän alaisuudessa ja muut tiedot voidaan replikoida tietoturvatasoltaan vaatimattomampaan ympäristöön. Ratkaisun hyödyn arvioinnissa on pyrittävä löytämään vastauksia muun muassa seuraaviin kysymyksiin:

- Kuinka todennäköisesti tietoturvaa yritetään murtaa?
- Kuinka paljon turvallisempi replikoinnilla toteutettu ratkaisu on?
- Kuinka paljon tietoturvan murtaminen maksaa?

3.4.2 Tietokannan hoitajan arviointikriteerit

Tässä kohdassa esitetään replikointiominaisuuksien arvioinnin viitekehyksen tietokannan hoitajan arviointikriteerit.

B. Tietokannan hoitaja

Replikointiympäristön suunnittelu

B.1. Tuotteen teoreettinen tausta

Tietokantatuotteen valmistajalla on ollut tuotetta toteuttaessaan oma käsityksensä replikoinnin teoriasta. Esimerkiksi jokin replikoinnin hallinnan protokolla on voitu toteuttaa hieman mukautettuna tuotteeseen. Seuraavien kriteerien avulla selvitetään tuotteen teoreettista taustaa.

B.1.1. Teoreettinen taustan esittely

Tuotteen teoreettisen taustan selvittäminen saattaa olla vaikeaa tuotteen esitteistä ja ohjeistuksesta. Tuotevertailussa joudutaan kuitenkin usein tarkastelemaan asioita yleisellä, valmistajariippumattomalla, terminologialla ja näin ollen olisikin suureksi avuksi, jos tuotteen valmistaja olisi kytkenyt erikoistermejään yleisesti tunnettuihin termeihin. Tällä kriteerillä arvioidaan yleisesti sitä tapaa, jolla tuotteen valmistaja esittelee teoreettista taustaa.

B.1.2. Replikoinnin hallinnan protokolla

Kriteerillä selvitetään käytettävä replikoinnin hallinnan protokolla (esim. pääkopioprotokolla ja äänestykseen perustuva joukkopohjainen protokolla).

B.1.3. Replikointitarpeen ilmeneminen

Kuinka replikointiohjelma saa tiedon replikointitarpeesta? Tietokantatuotteissa käytetään tehtävään yleisimmin seuraavia tapoja (Stacey, 1995):

1. Herätteet.

   Tietokantaan kohdistuva toimenpide laukaisee herätteen, joka edelleen käynnistää lähdetietokannassa replikointiprosessin.

2. Tietokantalokit

   Replikointiprosessi käynnistyy, kun lähdetietokannan tietokantalokissa havaitaan replikoitavaa tietoa.

Herätteen käyttäminen pidentää tietokantatapahtuman kestoa, sillä heräte suoritetaan tahdistetusti tietokantatapahtuman kanssa (Praxis, 1996c; Informix, 1996).

B.1.4. Replikoinnin kohde

Replikoinnin kohteena on perinteisesti ollut tieto ja siihen kohdistuneet muutokset. Mikä kuitenkaan estäisi käyttämästä replikointia myös toiminnan (esim. tallennetut proseduurit, herätteet) replikointiin? Özsun (1996) mukaan replikoinnin soveltumista käsittelyn välittämiseen on tutkittu suhteellisen vähän. Tällä kriteerillä selvitetään, mitä tuotteella voidaan replikoida (esim. tieto ja käsittely).

B.1.5. Käytettävä tekniikka

Replikointitekniikalla tarkoitetaan tässä yhteydessä sitä tapaa, jolla halutut muutokset välitetään replikaatteihin. Vaihtoehtoina ovat esimerkiksi tietokantatapahtumien ja proseduurikutsujen välittäminen. Proseduurikutsuja käytettäessä kohdejärjestelmään replikoidaan suoritettavan proseduurin nimi ja parametrit. Proseduurin suorittaminen kohdejärjestelmässä päivittää tiedot lähdejärjestelmää vastaaviksi. Myös tilannevedos (snapshot) voi toimia replikointitekniikkana. Tällä kriteerillä selvitetään, mitä tapoja on käytettävissä.

B.1.6. Ajantasaisuus

Ajantasaisuus on ehkä keskeisin replikointia jaotteleva tekijä (Watterson, 1996). Tahdistetussa replikoinnissa päivitys lähdetietokantaan ja yhteen tai useampaan

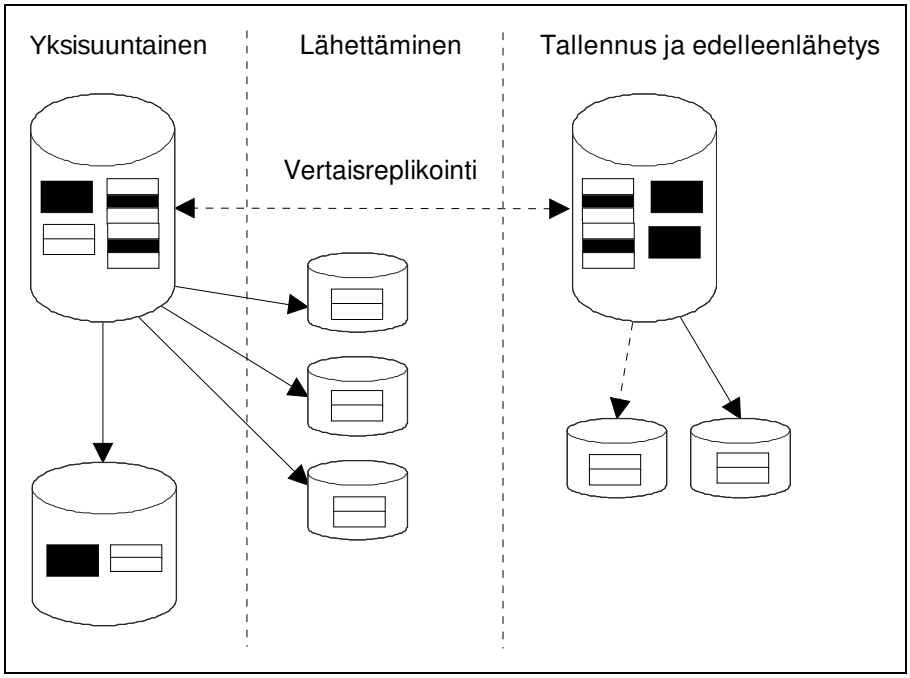
kohdetietokantaan suoritetaan samaan aikaan alkuperäisen tietokantatapahtuman kanssa käyttämällä kaksivaiheista vahvistusta (Stacey, 1995). Tahdistamattomassa replikoinnissa päivitetään ensin pääkopio ja vasta sen jälkeen, tietyn ajan kuluttua, toissijaiset kopiot. Replikoinnista puhuttaessa tarkoitetaan yleensä tahdistamatonta replikointia (Watterson, 1996). Tällä kriteerillä selvitetään, onko tuotteella mahdollisuus toteuttaa molemmat vaihtoehdot.

B.1.7. Päivitysmalli

Replikoinnin ajantasaisuuden ohella toinen tärkeä näkökulma replikointiin on se, sallitaanko käyttäjien ainoastaan lukea replikaatteja vai ovatko replikaatit myös päivitettävissä (Watterson, 1996). Kriteerillä haetaan vastausta tähän perus-kysymykseen. Päivitystapaa tarkasteltaessa voidaan käyttää seuraavaa luokittelua (Stepansky, 1996):

- Yksisuuntainen replikointi (Master-Slave, Primary Site Ownership ja Unidirectional)
  Yksisuuntainen ratkaisu, jossa Master-tietokantasolmu voi lukea ja kirjoittaa tietoa, kun Slave-tietokantasolmu voi ainoastaan lukea tietoa. Muunnelma yksisuuntaisesta replikoinnista on työnkulkutyyppinen (workflow) replikointiratkaisu, jossa tiedon omistajuus (master) siirtyy tietokantasolmulta toiselle.
- Lähettäminen (Broadcast)
  Yksisuuntainen replikointiratkaisu, jossa tieto lähetetään useille tietokantasolmuille. Horisontaalista osittamista käyttäen esimerkiksi valitut tietokantataulun rivit voidaan lähettää tiettyihin tietokantasolmuihin.
- Tallennus ja edelleenlähetys (Store-and-Forward, Cascade)
  Tässä yksisuuntaisessa ratkaisussa on ainakin kolme hierarkkista tasoa. Alkuperäisestä tietokantasolmusta tieto siirretään välillä olevaan tietokantasolmuun, josta se edelleen siirretään lopullisena kohteina oleviin tietokantasolmuihin.
- Vertaisreplikointi (Peer-to-Peer, Update Anywhere, Symmetric, Shared Ownership ja Bidirectional)
  Tässä kaksisuuntaisessa replikointiratkaisussa myös replikaatteja voidaan päivittää, joten päivityskonfliktit ovat mahdollisia.

Kuvassa 16 on esimerkkejä yksisuuntaisesta replikoinnista, lähettämisestä, tallennuksesta ja edelleenlähetyksestä ja vertaisreplikoinnista.



Kuva 16. Replikoinnin päivitysmalleja (Stepansky, 1996).

B.2. Tietokannan hoitajalta edellytettävät taustatiedot

Replikointiympäristön suunnittelu edellyttää tietoa relaatiotietokannoista, tiedon hajauttamisesta ja tietoliikenneverkoista. Tällä kriteerillä selvitetään, kuinka selkeästi taustatiedot esitetään tuotteen ohjeistuksessa.

B.3. Ohjeistuksessa mahdollisesti olevat esimerkkisovellukset

Tällä kriteerillä arvioidaan tietokantatuotteen replikointiesimerkkejä. Esimerkeistä voidaan tarkastella havainnollisuutta ja todenmukaisuutta, esimerkin rakentamisessa käytettyjä komentoja ja mahdollisia valmiita esimerkkiohjelmia.

B.4. Asennusvaatimukset

Replikoinnin asennusvaatimuksista on syytä selvittää niin lähde- kuin kohdejärjestelmistä seuraavat tekijät:

- keskusmuisti, prosessori, levytila, tietoliikenneverkko,
- käyttöjärjestelmä,
- tietokannan hallintajärjestelmä ja
- ohjelmointirajapinnassa käytettävä kieli.

B.5. Suorituskyvyn ennakointi

Replikointiympäristöä suunniteltaessa olisi kyettävä ennakoimaan tulevan ratkaisun suorituskykyä. Tuotteen ohjeistukseen mahdollisesti liitetyt suorituskyvyn laskenta-kaavat auttavat tehtävän suorittamista. Kriteerillä arvioidaan tuotteen mukana tulevaa suorituskyvyn ennakoinnin välineistöä.

Replikointiympäristön toteutus

B.6. Tuotteen asennus

Replikointiohjelmiston asennuksen helppouteen tulee kiinnittää erityistä huomiota (Schussel, 1996). Tällä kriteerillä pyritään arvioimaan seuraavia replikoinnin asennukseen liittyviä asioita:

- asennusohjelman eteneminen,
- manuaalisesti tehtävän työn osuus (esim. ini-tiedostojen muuttaminen),
- olemassa olevien tietokantojen muuttaminen ja
- etäasennuksen mahdollisuus.

B.7. Replikoinnin hallintaohjelma

Replikoinnin hallintaohjelmalla tarkoitetaan ohjelmaa, jolla tietokannan hoitaja määrittää replikoitavat tiedot, käynnistää, pysäyttää ja valvoo replikointiprosesseja sekä ohjaa toipumista virhetilanteista. Ohjelmaa voidaan siis joutua käyttämään replikointi-ympäristön asennuksen, toteutuksen ja käytön aikana. Ohjelmalle on esitetty muun muassa seuraavia tavoitteita:

- replikoinnin hallintaohjelman tulisi olla graafinen helppokäyttöisyyden saavuttamiseksi (Schussel, 1996; Watterson, 1996)
- ohjelmalla on voitava hallita kaikkia replikoinnin osa-alueita ja sitä tulisi voida käyttää mistä tahansa tietoliikenneverkkoon kuuluvasta tietokoneesta (Schussel, 1996)
- replikointiympäristöä on voitava muuttaa tietokannan tai replikoinnin käyttöä keskeyttämättä (Schussel, 1996)
- replikoinnin poikkeustilanteista on voitava saada sähköpostihälytys (Schussel, 1996)

Seuraavassa esitetään muutamia kriteerejä replikoinnin hallintaohjelman arviointiin replikointiympäristön toteutusvaiheessa.

B.7.1. Ohjelman yleiset ominaisuudet

Tällä kriteerillä arvioidaan ohjelman ulkoasua, havainnollisuutta, ohjelman toimintaa eri tietokantasolmuista, virheilmoitusten toteuttamistapaa ja ohjelman poistamista.

B.7.2. Liittymät tietolähteisiin

Liittymillä tietolähteisiin tarkoitetaan sitä tapaa, jolla replikoitavat tiedot määritetään replikoinnin hallintaohjelmassa. Ohjelma voi esimerkiksi näyttää olemassa olevan tietokannan taulumääritykset, joista tietokannan hoitaja valitsee replikoitavat tiedot. Seuraavilla kriteereillä arvioidaan liittymien määrittämistä ja poistamista.

B.7.2.1. Määrittäminen

Replikoitavien tietojen määrittämisen tulisi olla mahdollisimman yksinkertaista. Hallintaohjelmaa voidaan arvioida seuraavia kriteerejä käyttäen:

- Onko uuden tietolähteen määrittäminen vaikeaa?
- Kuinka tietolähde esitetään ohjelmassa?
- Varoittaako ohjelma suorituskyvyn ym. tekijöiden kannalta epäedullisista valinnoista?
- Mitä tietotyyppejä voidaan replikoida?
- Miten replikoinnin ajastus on määriteltävissä?

B.7.2.2. Poistaminen

Poistettaessa liittymää johonkin tietolähteeseen tulisi siitä aiheutua mahdollisimman vähän häiriötä tietokannan muulle käytölle. Poistamisesta voidaan arvioida toteutustapaa ja tehtävän suorituksen aikana saatavia varoituksia ja ohjeita.

B.8. Replikaattien alustaminen

Replikoitavien tietojen määrityksen jälkeen replikaateille voidaan joutua asettamaan alkuarvot ennen kuin niiden käyttö sallitaan. Alkuarvojen asettaminen voidaan tehdä esimerkiksi kopioimalla tiedot varmistusnauhalle, josta ne luetaan replikaatteihin. Replikoinnin hallintaohjelmassa voi olla käytettävissä muitakin alustamistapoja. Tällä kriteerillä selvitetään erilaiset alkuarvojen asettamistavat.

B.9. Tuotteen poistaminen

Tällä kriteerillä arvioidaan koko replikointiympäristön poistamista. Tämä tehtävä voidaan joutua suorittamaan esimerkiksi organisaation vaihtaessa tietokannan hallintajärjestelmää. Arvioinnin kohteina voivat olla poistamisen helppous sekä muutokset lähde- ja kohdejärjestelmiin.

Replikointiympäristön käyttö

Replikointiympäristön käytöllä tarkoitetaan tässä yhteydessä niitä päivittäisiä käytännön tehtäviä, joita tietokannan hoitaja suorittaa replikointiympäristön toiminnan ohjaamiseksi ja varmistamiseksi. Aiemmissa tutkimuksissa on esitetty vain hyvin yleisluonteisia vaatimuksia tehtävien suorittamisessa käytettäville ohjelmille. Watterson (1996) mainitsee, että tietokannan hoitajalla tulisi olla käytössään hyvä ”hallintapaketti” täsmentämättä juurikaan ilmaustaan. Praxis (1996b) esittää, että tehtävien suorittamisen tulisi olla helppoa ja ohjelmilla tulisi voida hallita koko replikointiprosessia. Schussel (1996) kuvaa vaatimuksia jo hieman yksityiskohtaisemmin, mutta esitettyjen vaatimusten suhde käytännön tehtäviin jää hämäräksi. Seuraavassa esitetäänkin replikointiin liittyvien käytännön tehtävien pohjalta muutamia käytönaikaisia arviointikriteerejä. Niillä pyritään arvioimaan muuan muassa replikoinnin suorituskyvyn seurantaa, poikkeustilanteiden käsittelyä, replikointiympäristön varmistamista ja laajentamista sekä poikkeuksellisten pyyntöjen toteuttamista. Osa näistä tehtävistä voidaan suorittaa replikoinnin hallintaohjelmalla (esimerkiksi replikoinnin käynnistäminen ja pysäyttäminen), ja osa suoritetaan tietokannan hallintajärjestelmän yleisillä toiminnoilla (esimerkiksi varmistaminen). Jaosta huolimatta molemmat tehtäväryhmät esitetään kriteerissä ”B.10. Replikoinnin hallintaohjelma”, sillä jaottelu vaihtelee tuotteittain.

B.10. Replikoinnin hallintaohjelma

Replikoinnin hallintaohjelma on keskeisessä asemassa replikoinnin käytön aikana. Ohjelma on paitsi apuvälineenä muutosten tekemisessä, niin se myös esittää replikoinnin tilan. Seuraavien kriteerien avulla arvioidaan ohjelmaa replikoinnin käytön aikana. Kriteereistä on esitetty ensin toiminnalliset kriteerit ja lopuksi informatiiviset kriteerit.

B.10.1. Replikoinnin käynnistäminen ja pysäyttäminen

Replikointi ei suinkaan jatku keskeytyksettä ensimmäisen käynnistyskerran jälkeen koko järjestelmän käyttöiän ajan. Ennemmin tai myöhemmin prosessi joudutaan tilapäisesti keskeyttämään joko virhetilanteiden tai tietokantamuutosten vuoksi. Käynnistämisen ja pysäyttämisen tulisikin olla helppoa ja se olisi voitava tehdä käyttämättä useita käyttöliittymiltään toisistaan poikkeavia ohjelmia. Tietokannan hoitajan on myös voitava helposti nähdä replikoinnin kulloinenkin tila. Tällä kriteerillä arvioidaan, kuinka replikointi käynnistetään ja pysäytetään. Mahdollinen vastaus tähän arviointikriteeriin voisi olla lyhyt tuotteen ohjeista koottu kuvaus.

B.10.2. Replikointiympäristön mukauttaminen

Käyttäjien tietotarpeiden muuttuminen voi useissa tapauksissa vaatia replikointimääritysten muuttamista. Tietokannan hoitajan on esimerkiksi muutettava määrityksiä lisättyään tietokannan tauluun uuden sarakkeen tai muutettuaan replikoinnin ajastusta. Kriteerillä arvioidaan, kuinka helposti replikointiympäristö on mukautettavissa esimerkiksi seuraavissa tilanteissa:

- replikoitavan taulun lisääminen ja poistaminen,
- replikoitavan sarakkeen lisääminen ja poistaminen ja
- tietokantasolmun lisääminen ja poistaminen.

Lisäksi voidaan arvioida, kuinka hyvin mukauttamisen vaikutukset voidaan ennakoida muihin osa-alueisiin (esimerkiksi suorituskyky ja varmistaminen). Vastauksena kriteeriin on ohjeista, toimittajalta tai kokein koottu arvio tehtävien suorittamisen vaikeudesta.

B.10.3. Poikkeustilanteiden käsittely

Replikoinnin poikkeustilanteisiin kuuluvat esimerkiksi tietokantasolmun vikaantuminen, replikaateille varatun levytilan täyttyminen ja vertaisreplikoinnin päivityskonfliktit. Poikkeustilanteissa järjestelmän tulisi tarjota tietokannan hoitajalle

mahdollisimman paljon apua toipumismenettelyn suorittamiseksi (Schussel, 1996). Schussel (1996) erottaa seuraavat viisi vaihetta toipumisessa:

1. selvitetään, mikä on rikkoontunut,
2. selvitetään, mikä vian aiheutti,
3. päätetään, kuinka vika korjataan,
4. suoritetaan korjaustoimenpiteet ja
5. varmistetaan, että palautetut tietokannat ovat ristiriidattomassa tilassa.

Tällä kriteerillä arvioidaan, kuinka replikoinnin hallintaohjelma tai tietokannan hallintajärjestelmän apuohjelmat esittävät poikkeustilanteet ja mitä tukea ohjelmat tarjoavat toipumiseen. Koska erilaisia poikkeustilanteita on suuri joukko, voidaan poikkeustilanteet asettaa tärkeysjärjestykseen ja arvioida ainoastaan vakavimmat häiriöt. Tuotteen ohjeistus on perustana tehtävän suorittamiselle.

B.10.4. Replikointiympäristön varmistaminen

Replikointiympäristön varmistamisen tulisi luonnollisesti olla yhtä yksinkertaista kuin normaalien tietokantojen varmistaminen. Useissa replikointitoteutuksissa tehtävä suoritetaan replikoinnin hallintaohjelman sijaan tietokannan hallintajärjestelmän apuohjelmilla. Huomiota on kiinnitettävä myös niihin replikointimäärityksiin, jotka on tallennettu varmistettavien tietokantojen ulkopuolelle. Tällä kriteerillä arvioidaan lähde- ja kohdejärjestelmien sekä replikointimääritysten varmistamista. Lisäksi voidaan selvittää, vaatiiko varmistaminen replikoinnin keskeyttämistä.

B.10.5. Poikkeuksellisten pyyntöjen toteuttaminen

Replikoinnin hallintaohjelmalla on voitava helposti toteuttaa ennakolta suunnitellusta toiminnasta poikkeavia tehtäviä. Replikointi tulisi esimerkiksi voida suorittaa pyynnöstä tietyllä hetkellä, vaikka se on ajastettu tunnin päähän. Myös kertaluonteinen replikointi johonkin tietokantasolmuun voi olla joissakin tilanteissa tarpeen. Tietokannan hoitaja voi myös haluta tallentaa ohjelman asetukset tai jonkin tehtävän kulun tiedostoon ja käynnistää ohjelman myöhemmin tätä tiedostoa käyttäen. Tällä kriteerillä arvioidaan,

kuinka hyvin replikoinnin hallintaohjelmalla voidaan toteuttaa poikkeuksellisia pyyntöjä. Vastausta kriteeriin voidaan hakea luetteloimalla edellä esitettyjen lisäksi muitakin tilanteita ja kokeilemalla käytännössä, kuinka ohjelmalla voidaan toteuttaa nämä pyynnöt.

B.10.6. Suorituskyvyn seuranta

Replikoinnin suorituskyvyn seurannan välineet näyttävät saaneen varsin vähän huomiota aiemmissa tutkimuksissa. Schussel (1996) mainitsee, että hajautetun tietokannan hoitajan tehtäviin kuuluu myös suorituskyvyn seuranta, muttei esitä mitään vaatimuksia suorituskyvyn seurannan ohjelmille. Praxis (1996b) ei myöskään mainitse suorituskyvyn seurantaa erillisenä tekijänä ottaessaan kantaa replikoinnin hallintaan. Yawin (1994) ottaa kantaa replikoinnin suorituskykyyn Lotus Notesin replikoinnin ajastuskysymystä tarkastellessaan. Hänen mukaansa ajastuksen kannalta huonosti suunniteltu replikointiratkaisu voi kuormittaa tarpeettomasti tietoliikenneverkkoa.

Suorituskyvyn seuranta ei kuitenkaan saisi olla pelkästään kerran, kenties juuri järjestelmän toteutuksen jälkeen, suoritettava tehtävä. Replikointitarpeet muuttuvat toteutuksen jälkeen useastikin, joten myös suorituskykyä olisi tarkasteltava uudelleen jokaisen muutoksen jälkeen. Esimerkiksi uuden tietolähteen lisäämisellä ja päivitysmäärien muuttumisella on vaikutuksia suorituskykyyn. Näin ollen tietokannan hoitajan tulisi voida tarkastella suorituskyvyn muutoksia joko tietokannan hallintajärjestelmän apuohjelmilla tai replikoinnin hallintaohjelmalla. Tällä kriteerillä arvioidaan, mitä apuvälineitä tehtävään on käytettävissä ja mitä suorituskyvyn mittareita (esimerkiksi tietokantatapahtumia aikayksikössä ja levyoperaatioiden määrä) ohjelmilla voidaan arvioida.

B.10.7. Etävalvonta

Schussel (1996) korostaa sähköpostin käyttömahdollisuutta replikoinnin virhetilanteista tiedotettaessa. Järjestely tekee hänen mukaansa mahdolliseksi valvonnan mistä tahansa

tietoliikenneverkon tietokoneesta. Valvonta ei kuitenkaan saisi jäädä pelkästään virhetilanteista tiedottamiseen, vaan myös esimerkiksi suorituskykyä olisi voitava seurata verkon eri puolilta. Watterson (1996) esittääkin, että replikoinnin hallinnan tulisi ihannetilanteessa toimia yhdessä muiden järjestelmän hallinnan ohjelmien kanssa. Edelleen voidaan pohtia, onko valvontafunktio riittävä vai pitäisikö replikointia myös voida mukauttaa mistä tahansa tietoliikennesolmusta. Tällä kriteerillä arvioidaan, voidaanko replikointia valvoa mistä tahansa tietoliikenneverkon tietokoneesta. Mikäli järjestely on mahdollinen, voidaan edelleen arvioida mitä osa-alueita (esimerkiksi virhetilanteet ja niistä toipuminen, suorituskyky, varmistaminen ja toipuminen) voidaan valvoa ja millaiset laitteisto- ja ohjelmistovaatimukset tällöin ovat.

3.4.3 Sovellusohjelmoijan arviointikriteerit

Tässä kohdassa esitetään replikointiominaisuuksien arvioinnin viitekehyksen sovellus-ohjelmoijan arviointikriteerit.

C. Sovellusohjelmoija

Replikoinnin vaikutukset sovellusohjelmoijan tehtäviin on myös selvitettävä replikointiominaisuutta arvioitaessa. Vaikka tietokantatoimittajat ovatkin alkaneet tarjota replikointimahdollisuutta yli tuoterajojen, saattaa replikointi ainakin toisten valmistajien tuotteista vaatia ohjelmointia (Schussel, 1996). Seuraavilla kriteereillä pyritään arvioimaan tilanteita, joissa replikointiympäristöä toteutettaessa voidaan joutua ohjelmoimaan. Kriteerit on jaettu kahteen pääryhmään: 1) replikointiympäristön ohjelmointi ja 2) normaali sovellusohjelmointi. Replikointiympäristön ohjelmointiin voi kuulua tuotteiden yhteensovittamistehtävän lisäksi esimerkiksi replikoinnin hälytysten (esim. sähköposti- tai hakulaitehälytysten) ja raportoinnin ohjelmointia. Tehtävät ovat luonteeltaan tyypillisesti kertaluonteisia, ja lopputulosta voidaan käyttää sellaisenaan tai hieman muuntaen useissa replikointia hyödyntävissä tietojärjestelmissä. Normaaliin

sovellusohjelmointiin kuuluu puolestaan jonkin tietojärjestelmän toteuttamiseen liittyvä sovellusohjelmointi.

Sovellusohjelmoinnin huomioiminen on tärkeää siksi, että on oleellista tietää, kuinka replikoidun tietokannan ohjelmointi poikkeaa replikoimattoman tietokannan ohjelmoinnista. Pääryhmäjaottelu toimii apuna myös replikoinnin edistyneisyyden arvioinnissa. Mikäli ohjelmoijan on toistuvasti käytettävä aikaansa ensimmäiseen pääryhmään kuuluviin tehtäviin, voidaan tuotetta pitää kehittymättömänä. Toisessa ääripäässä sovellusohjelmoijan ei tarvitse huolehtia taustalla olevan replikoinnin toimivuudesta, vaan hän voi keskittyä normaalin sovelluslogiikan toteuttamiseen. Sovellusohjelmoijan näkökulman muodostamista on vaikeuttanut huomattavasti se, että replikointiin liittyviä ohjelmointitehtäviä on tutkittu hyvin vähän. Sen vuoksi seuraavia kriteerejä on pidettävä vain esimerkkeinä ominaisuuden arviointiin. Arviointitilanteessa näkökulman huomioimisessa voidaankin edetä esittämällä tuotteen toimittajalle käytännön sovellustilanne ja pyytää toimittajaa kertomaan, tarvitaanko toteutuksessa ohjelmointia.

Replikointiympäristön ohjelmointi

Replikointiympäristön ohjelmoinnilla tarkoitetaan tässä yhteydessä kaikkia niitä ohjelmointitehtäviä, joilla taataan replikointia hyödyntävien tietojärjestelmien toimivuus. Tietokannan hoitajan ja sovellusohjelmoijan välinen tehtävänjako on usein vaikeaa, ja sen vuoksi näiden tehtävien arviointi on hieman ongelmallista. Kuuluuko raportoinnin, hälytysten ja liittymien ohjelmointi tietolähteisiin monimutkaisine herätteineen tietokannan hoitajalle vai ohjelmoijalle? Rajanveto kannattanee tehdä tilannekohtaisesti sopimalla molempien vastuualueet. Tämä pääryhmä kriteereineen liittyy hyvin läheisesti tietokannan hoitajan pääryhmiin ”Replikointiympäristön toteutus” ja ”Replikointiympäristön käyttö”. Kriteerien selvittämisessä voidaan käyttää tuotteen ohjeistuksesta ja toimittajalta saatavaa informaatiota.

C.1. Liittymien ohjelmointi tietolähteisiin

Mahdollisuus laajentaa tuotteen tukemien tietolähteiden joukkoa voi olla eräissä tapauksissa houkutteleva. Mikäli organisaatiolla on käytössään teknologialtaan vanhempia tietokantaratkaisuja, joista halutaan siirtää tietoa uudempiin tietokantoihin esimerkiksi raportointitarkoituksiin, replikoinnin soveltamisella voidaan kenties saavuttaa kustannussäästöä. Tämä on mahdollista kuitenkin ainoastaan, mikäli tietokantatuotteen replikointitoteutuksessa on tällainen laajennusmahdollisuus ja vanhasta tietokannasta voidaan kohtuullisella vaivalla saada esille replikoitava tieto. Ohjelmoijan eräänä arviointikriteerinä onkin arvioida tuotteen mahdollista ohjelmointirajapintaa laajentamisen näkökulmasta. Vastauksena tähän kriteeriin voidaan esittää, onko tuotteen tukemien tietolähteiden joukkoa mahdollista laajentaa. Mikäli laajentaminen on mahdollista, niin täsmennetään laajentamisessa käytettävä ohjelmointikieli ja esitetään arvio toteuttamisen vaikeudesta.

C.2. Raportoinnin ja hälytysten ohjelmointi

Replikoinnin hallintaohjelmat voivat esittää replikoinnin tilan tietokannan hoitajalle eri tavoilla. Tilatieto voidaan esittää esimerkiksi siten, että kullakin replikoinnissa mukana olevalla tietokantasolmulla on näytössä ikoni, joka vaihtaa väriään vikojen ilmetessä. Replikoinnin toiminnasta voidaan myös saada erilaisia paperiraportteja. Organisaatiolla voi kuitenkin olla tarve laajentaa olemassa olevia raportointi- ja hälytysmahdollisuuksia. Tietokannan hoitaja voi esimerkiksi vaatia, että replikaateille varatun levytilan täyttyessä hälytys välitetäänkin hakulaitteeseen. Replikoinnin toiminnan esittävä raportti voidaankin haluta tallentaa tietokantaan. Tällä arviointikriteerillä arvioidaankin, kuinka olemassa olevia raportointi- ja hälytysmahdollisuuksia voidaan laajentaa ohjelmoimalla.

Normaali sovellusohjelmointi

Replikointia hyödyntävää tietokantajärjestelmää rakennettaessa tietokannan hallintajärjestelmä piilottaa ihannetapauksessa replikoinnin sovellusohjelmoijalta (vrt.

kohta ”2.4.1. Tavoitteet hajautetulle tietokannan hallintajärjestelmälle”, tavoite 6). Tällöin replikointia hyödyntävän sovellusohjelman laatiminen ei eroa lainkaan normaalin sovellusohjelman laatimisesta. Käytännön ohjelmointityössä sovellusohjelmoijan vastuulle on kuitenkin annettu joukko huomioitavia seikkoja, joiden selvittämisessä voidaan käyttää seuraavia kriteerejä. Vastauksia kriteereihin voidaan etsiä tuotteen ohjeistuksesta ja haastattelemalla tietokantatoimittajaa.

C.3. Ohjelmointitehtävien huomioiminen

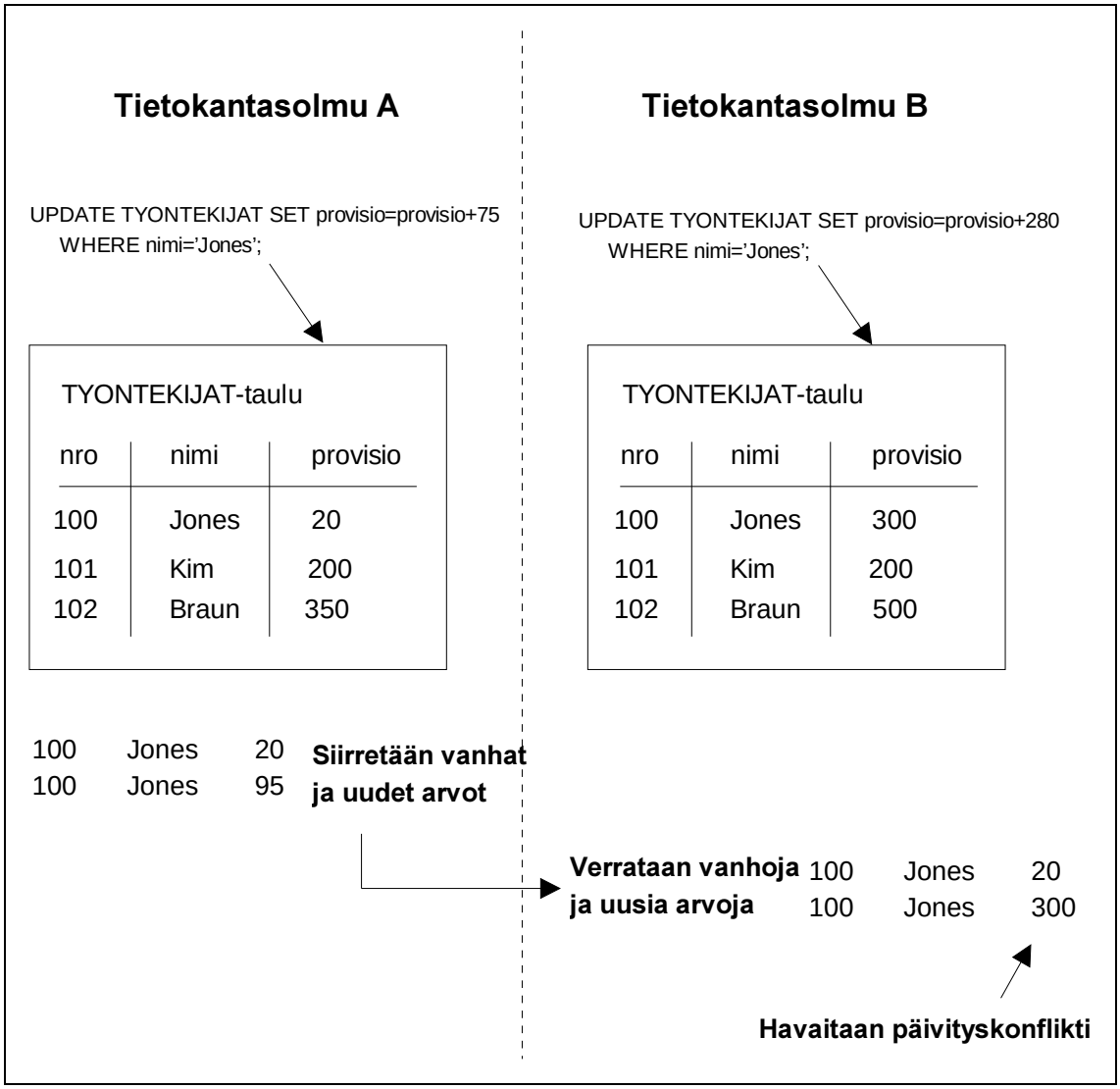
Kriteerin avulla tarkastellaan tietojärjestelmän suunnitteluvaiheessa, kuinka tuotteen valmistaja kuvaa replikoinnin vaikutukset sovellusohjelmointiin. Mikäli ohjeistuksesta ei ole löydettävissä yhtenäistä esitystä replikoinnin vaikutuksista, voivat esimerkiksi seuraavat seikat yllättää toteutusvaiheessa:

- Eräissä replikointitoteutuksissa sovellusohjelmoijan on huolehdittava, että ohjelma päivittää yksisuuntaisessa replikointiratkaisussa ainoastaan pääkopioita (esim. Sybase). Mikäli ohjelmassa päivitetään toissijaisia kopioita, tietojen ristiriidattomuutta ei voida enää taata.
- Vertaisreplikoinnin päivityksen rajoittaminen ainoastaan tiettyyn tietokantasolmuun voidaan tehdä sovellustasolla (Oracle).
- Työnkulkutyyppisessä replikointiratkaisussa sovelluksen on taattava, että tietokantasolmu päivittää ainoastaan omistamiaan tietoalkioita. Omistajuus voidaan toteuttaa esimerkiksi siten, että relaatiomääritykseen liitetään tilatietoa kuvaava sarake (laskurivillä tilatieto voi olla esim. ”syötetty”, ”hyväksytty”, ”toimitettu” ja ”laskutettu”).

Vastauksena tähän kriteeriin esitetään kirjallinen arvio replikoinnin vaikutuksista sovellusohjelmointiin. Arvioinnin jälkeistä asioiden mieleenpalauttamista varten mukaan voidaan liittää myös luettelo niistä ohjeistuksen kohdista, joissa sovellusohjelmointia käsitellään.

C.4. Vertaisreplikoinnin päivityskonfliktit

Vertaisreplikoinnin päivityskonfliktit voidaan tunnistaa ja ratkaista joko tietokantatoimittajan tai sovellusohjelmoijan laatimilla poikkeuskäsittelijöillä. Päivityskonfliktit tunnistetaan replikointitoteutuksissa tyypillisesti vertaamalla tietokantarivin kunkin muuttuneen sarakkeen vanhaa ja uutta arvoa keskenään, kuten kuvassa 17 osoitetaan. Mikäli vanha ja uusi arvo poikkeavat toisistaan, on kyseessä päivityskonflikti.



Kuva 17. Vertaisreplikoinnin päivityskonfliktin havaitseminen (Oracle, 1996, s. 6-3).

Päivityskonfliktien ratkaisemiseksi on tarjottu muun muassa seuraavia tapoja (Oracle, 1996):

- Ajallisesti ensimmäisenä tai viimeisenä käynnistetyn tietokantatapahtuman päivitys jää voimaan (earliest timestamp, latest timestamp).
- Taulujen sarakkeista voidaan muodostaa sarakeryhmiä (column groups), joille annetaan tietokantasolmuittain prioriteettiluku. Päivityksessä prioriteettiluvultaan suurimman sarakeryhmän arvo jää voimaan.
- Tietokantasolmuille annetaan prioriteettiluvut. Päivityksessä prioriteettiluvultaan suurimman tietokantasolmun arvo jää voimaan.

Ratkaisutapoja voidaan käyttää sellaisenaan joidenkin tietojärjestelmien toteuttamisessa. Käytännössä päivityskonfliktien ratkaiseminen on kuitenkin sovelluskohtaista, eikä tietokannan hallintajärjestelmiin voida toteuttaa sellaista konfliktien ratkaisulogiikkaa, joka toimisi kaikkien erilaisten tietokantamääritysten yhteydessä (Schussel, 1996). Mikäli replikointia sovellettaessa päädytään vertaisreplikoinnin käyttöön, on sovellusohjelmoijan ja tietokannan hoitajan näin ollen varauduttava ottamaan kantaa päivityskonfliktien hallintaan. Sovellusohjelmoijan tehtävänä onkin tämän kriteerin osalta selvittää tuotteen ohjeistuksesta, kuinka päivityskonfliktien hallinta voidaan toteuttaa ohjelmallisesti. Tarkastelun kohteina voivat olla käsittelijöiden laadinnassa käytettävä ohjelmointikieli ja omien käsittelijöiden liittäminen tietokannan hallintajärjestelmään. Koska kaikki replikointitoteutukset eivät varsinaisesti salli vertaisreplikoinnin käyttöä, voidaan kriteeri ohittaa arviointitilanteessa tällaisten tuotteiden osalta.

3.4.4 Peruskäyttäjän arviointikriteerit

Tässä kohdassa esitetään replikointiominaisuuksien arvioinnin viitekehyksen peruskäyttäjän arviointikriteerit.

D. Peruskäyttäjä

Replikoinnin vaikutusta peruskäyttäjän tehtäviin ja hänen käyttämiensä tietojärjestelmien ominaispiirteisiin (esim. suorituskykyyn ja tietojärjestelmän saatavuuteen) voidaan selvittää tarvittaessa. Olkoonpa tietokantatuotteen replikointitoteutus ominaisuuksiltaan kuinka kehittynyt tahansa, niin peruskäyttäjä kokee lopulta replikoinnin toimivuuden tai toimimattomuuden. Peruskäyttäjälle laadittu raportointijärjestelmä voidaan esimerkiksi laatia replikaattien varaan. Liikkuvien tietojärjestelmien yhteydessä peruskäyttäjä voi olla myös aktiivisessa roolissa käynnistäessään replikoinnin kannettavalta tietokoneeltaan. Tietohallinnon käyttöön suunnatuissa erityissovelluksissa tietokannan hoitaja voi toimia peruskäyttäjän roolissa. Vaikka käyttäjien suhde replikointiin vaihtelee siis toteutuksittain, tavoitteena on kuitenkin piilottaa teknologia peruskäyttäjiltä (vrt. hajautetun tietokannan tavoitteet).

Tässä tutkielmassa peruskäyttäjän näkökulman huomioimiseksi on esitetty ainoastaan kolme tietojärjestelmän käytönaikaista arviointikriteeriä. Kriteereillä pyritään yleisluontoisuudestaan huolimatta hakemaan vastausta siihen, ovatko replikointiin liitetyt edut todellisia. Lisäksi kriteerit voivat innoittaa käytännön kokeilujen laatimiseen. Tavoitteena ei ole ollut huomioida replikoinnin vaikutuksia peruskäyttäjän tehtäviin liikkuvissa tietojärjestelmissä. Tietojärjestelmien liikkuvuuden ja replikoinnin suhdetta on käsitelty useissa yhteyksissä (mm. Froemming, 1996a; Froemming 1996b; Ratner ym., 1997; Lubinski ym., 2000), ja aihe ansaitsisikin huomattavasti laajemman tarkastelun. Kriteereitä voidaan silti käyttää replikoinnin arviointiin myös tässä yhteydessä. Näkökulman huomioimista viitekehyksessä on vaikeuttanut se, ettei peruskäyttäjää ole huomioitu aiemmissa tutkimuksissa juuri lainkaan.

Replikoinnin vaikutusten näkyminen

Peruskäyttäjän replikoinnin arviointikriteereillä tutkitaan replikoinnin toimintaa jossakin tietyssä sovellustilanteessa. Tarkastelun kohteena ei siis ole esimerkiksi suorituskyvyn vaihtelujen esittäminen replikoinnin hallintaohjelmassa vaan käytännön suorituskyky.

Arviointikriteerien selvittämisessä tarvittavien järjestelyjen suorittaminen jää kriteeristön soveltajalle, sillä sovellusriippumattoman menettelytavan antaminen on mahdotonta. Näkökulman selvittämistä ei ole tarkoitettu ainoastaan peruskäyttäjän tehtäväksi, vaan tehtävä on suoritettava yhteistyössä tietohallinnon edustajan ja tietokannan hoitajan kanssa.

D.1. Vasteaika

Kriteerillä mitataan replikointia hyödyntävän ohjelman vasteaikaa. Mittauksessa ohjelmasta voidaan etsiä sellaisia kohtia, joissa käsitellään runsaasti tietokantaa. Sen jälkeen mitataan tietokantahaun käynnistämisen ja vastauksen palauttamiseen kuluvaa aikaa. Aikaa voidaan verrata tilanteeseen, jossa replikointia ei lainkaan käytetä. Tämä kriteeri on erityisen käyttökelpoinen, jos vanhaa ja uutta replikointiin perustuvaa tietojärjestelmää voidaan käyttää rinnakkain ja mikäli tietokantaan kohdistuva toimenpide (esimerkiksi monimutkainen kysely) toistuu muuttumattomana tietyin aikavälein. Tämä kriteerin selvittäminen on tarkoitettu suoritettavaksi samanaikaisesti tietohallinnon arviointikriteerin A.7.1.1. kanssa.

D.2. Saatavuus

Tietohallinnon arviointikriteerillä A.7.1.2. arvioidaan saatavuuden paranemisesta koituvaa hyötyä laskennallisesti komponenttien vikaantumisvälien avulla. Tällä kriteerillä on tarkoitus kerätä käyttäjien arvioita saatavuuden parantumisesta. Replikoinnin eräs sovelluskohde on varajärjestelmän rakentaminen replikaattien varaan. Käyttäjiltä voidaan tällaisessa järjestelyssä kysyä varajärjestelmään siirtymisen jälkeen, kuinka huomaamattomana he kokivat varajärjestelmään kytkeytymisen.

D.3. Suorituskyky

Replikoinnin käytännön nopeutta voidaan selvittää koejärjestelyin. Jos tulevasta replikointiratkaisusta tiedetään ennakkoon tietokantatapahtumien määrä aikayksikössä,

voi olla hyödyllistä kuormittaa replikointiohjelmaa suuremmalla päivitysnopeudella. Yksinkertaisimmillaan koejärjestely voidaan toteuttaa esimerkiksi laatimalla tuhansia tietojen lisäyskäskyjä sisältävä komentotiedosto, joka suoritetaan useissa työasemissa samaan aikaan. Tietokannan apuohjelmilla voidaan selvittää, toimiiko replikointi suurella päivitysnopeudella. Käyttäjä voi avustaa tietokannan hoitajaa kriteerin selvittämisessä. Vastauksena kriteeriin esitetään arvio replikoinnin toiminnasta suuren tapahtumamäärän yhteydessä.

3.5 Yhteenveto

Replikoinnin arvioinnin tarve on tullut entistä ajankohtaisemmaksi viime aikoina. Tietokannan hallintajärjestelmien lisäksi replikointiin törmää myös Internetiin liittyvissä teknologioissa ja dokumenttien hallintajärjestelmissä. Lähinnä replikointitermistön runsaus ja päällekkäisyys on tehnyt arvioinnista varsin hankalaa.

Luvussa 3 esitettiin tietokannan hallintajärjestelmien replikointiominaisuuksien arvioinnin viitekehys. Kehys perustuu replikoinnista julkaistuun tutkimustietoon ja lehtiartikkeleihin. Tietohallinnon, tietokannan hoitajan, sovellusohjelmoijan ja peruskäyttäjän näkökulmista esitetään joukko arviointikriteerejä, joihin haetaan vastauksia esitteistä, ohjekirjoista ja mahdollisesti käytännön kokein. Kehyksen soveltamisella voidaan tiedostaa replikointitarpeita, tehostaa replikoinnin suunnittelua ja toteutusta sekä yhtenäistää organisaation replikointikäsitteistöä.

4 KOKEILU

Tässä luvussa sovelletaan tietokannan hallintajärjestelmien replikointiominaisuuksien arvioinnin viitekehystä. Tutkimuksen toimeksiantajan, UPM-Kymmene Oyj Kajaanin, osoittamassa käytännön kokeilussa vertaillaan kahta tietokantatuotetta ja esitetään arvioita replikoinnin soveltuvuudesta toimeksiantajan käyttöön.

Kohdassa 4.1 kuvataan lyhyesti yrityksen taustaa ja prosessin tietojärjestelmiä. Kohdassa 4.2 esitetään koejärjestelyn tavoitteet, kohdassa 4.3 selvitetään kokeilua edeltäneitä toimenpiteitä ja kohdassa 4.4 esitellään suoritetut koeajot tuloksineen. Sen jälkeen sovelletaan viitekehystä kahden tietokantatuotteen arviointiin kohdassa 4.5. Kokeilun aikana tehdyt havainnot viitekehyksen toimivuudesta ja replikoinnin soveltuvuudesta toimeksiantajan käyttöön esitetään kohdassa 4.6. Kohdassa 4.7. on luvun yhteenveto.

4.1 Tutkimusympäristön esittely

4.1.1 UPM-Kymmene Oyj Kajaani

Euroopan suurin metsäteollisuusyhtiö UPM-Kymmene syntyi vuonna 1996, kun Repola Oy ja sen tytäryhtiö Yhtyneet Paperitehtaat Oy sekä Kymmene Oy sulautuivat yhdeksi suuryhtiöksi. UPM-Kymmene on myös maailman suurimpia metsäyrityksiä. Konsernin liikevaihto on noin 10 miljardia euroa. Paperintuotantokyky on noin 10 miljoonaa tonnia ja omien sellutehtaiden tuotantokyky 2,1 miljoonaa tonnia vuodessa. Konsernin palveluksessa on noin 36 000 henkilöä.

UPM-Kymmene Oyj Kajaani on osa tätä suuryhtymää. Kajaanin paperitehdas käsittää kaksi tulososastoa: SC-erikoispaperiosasto (paperikone 2) ja sanomalehtipaperiosasto (paperikone 3 ja paperikone 4). Näiden toimintaa tukevat kuitu- ja voimaosasto, tutkimus- ja kehitysosasto, tehdaspalvelu, henkilöstöosasto ja talousosasto. Kokonaistuotantokyky on yli puoli miljoonaa tonnia vuodessa.

4.1.2 Prosessin tietojärjestelmät

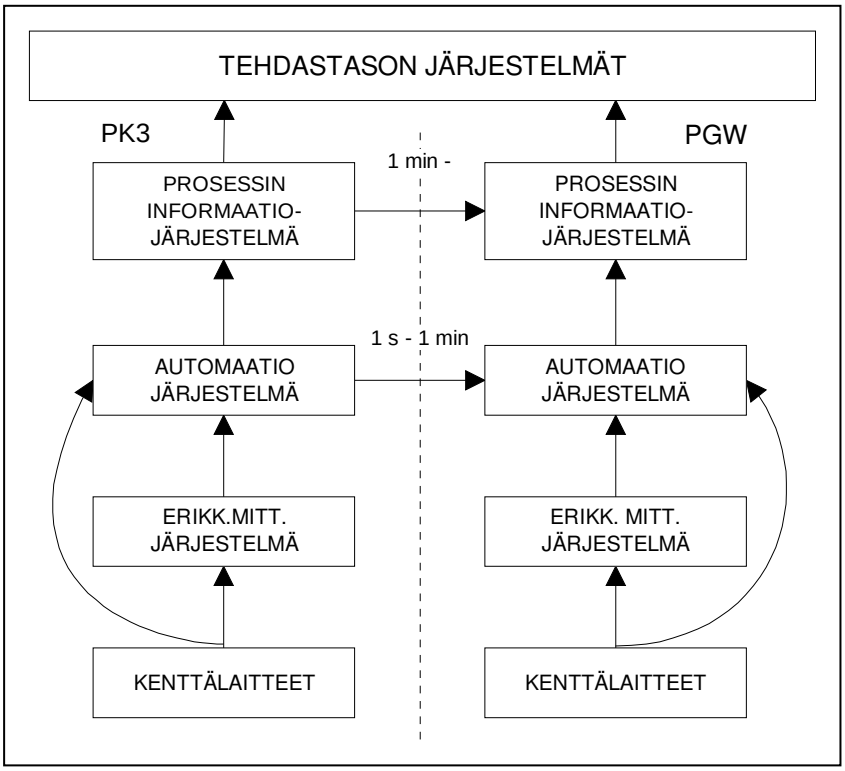
Paperitehtaiden, kuten muunkin prosessiteollisuuden, tuotantoprosesseilta edellytetään yhä enemmän tehokkuutta. Sen vuoksi on ollut tarve kehittää välineitä tuotantoprosessin parantamiseksi ja tehokkuuden lisäämiseksi. Prosessin tietojärjestelmät on tarkoitettu juuri tähän tehtävään. Niiden avulla kerätään tietoa prosessista ja sen tapahtumista. Kerättyä tietoa käytetään hyväksi prosessin optimoimiseen, toiminnan analysointiin sekä prosessin ja kunnossapitotoiminnan kehittämiseen.

Prosessin tietojärjestelmille on tyypillistä:

- Järjestelmien suorituskyky- ja toimintavarmuusvaatimukset ovat korkeat. Koska järjestelmiin ei ainoastaan syötetä tietoa käsin, vaan suurin osa tiedosta tulee automaatiojärjestelmien mittalaitteilta, on käytettävien laitteisto- ja ohjelmistokomponenttien kyettävä käsittelemään suuria tietomääriä. Käytettävien laitteisto- ja ohjelmistokomponenttien on toimittava 24 tuntia vuorokaudessa, ja niiden on myös toivuttava mahdollisimman nopeasti vikatilanteista.
- Prosessin tietojärjestelmät saavat tietoa mahdollisesti useistakin automaatiojärjestelmistä. Lukuisien liittymärajapintojen suunnittelu, toteutus ja ylläpito on haastava tehtävä. Ylläpitoa hankaloittaa myös jatkuvasti kasvava prosessimittausten määrä. Uusien mittausten lisääminen ja ylläpito aiheuttaa mittavan työmäärän.

- Prosessin tietojärjestelmät tuottavat tietoa lukuisiin muihin järjestelmiin. Esimerkiksi tilaustenhallinnan ja tehdaspalvelun tietojärjestelmät voivat saada tietoa prosessin tietojärjestelmistä.
- Järjestelmät on toteutettu useilla erilaisilla laitteisto- ja ohjelmistokomponenteilla. Automaatiojärjestelmien toimittajat ovat voineet käyttää järjestelmissään yhteensopimattomia komponentteja ja sen vuoksi prosessin tietojärjestelmänkin toteuttaminen voi edellyttää tietyn työkalun käyttöä. Avoimuus on kuitenkin lisääntynyt viime vuosina tälläkin alueella.
- Aika on keskeinen elementti järjestelmissä. Mittaukset on sidottu aikaan.

UPM-Kymmene Oyj Kajaanissa on tavoitteena kuvan 18 mukainen luokittelu prosessin tietojärjestelmien liittämisessä tehdastason järjestelmiin (Auvinen, 1997). Kenttälaitteisiin kuuluvat mitta-anturit ja toimilaitteet, jotka ovat osa prosessin fyysistä ohjausta. Erikoismittausjärjestelmät kykenevät analysoimaan mittausinformaatiota ja välittämään sitä automaatiojärjestelmille. Erikoismittausjärjestelmiin kuuluvat esimerkiksi Sensodec- ja Ulma-mittausjärjestelmät. Automaatiojärjestelmillä ohjataan prosessia ja hallitaan ohjaukseen tarvittavaa mittausinformaatiota. Lisäksi ne ovat prosessin tietojärjestelmien ensisijaisena tietolähteenä. Alcont, Damatic ja Packman ovat esimerkkejä automaatiojärjestelmistä. Prosessin tietojärjestelmiin sisältyy sovelluksia päivittäiseen toimintaan liittyvän tiedon hallintaan ja analysointiin. Niiden kehittämisen lähtökohtana on ollut tarve hallita tuotantoprosessista saatavaa tietoa huomattavasti automaatiojärjestelmiä pidemmällä ajanjaksolla. UPM-Kymmene Oyj Kajaanissa käytettäviä prosessin tietojärjestelmätuotteita ovat esimerkiksi Honeywell Oy:n HIC, Metso Automation Oy:n XIS ja ABB Oy:n RTDB. Kuvassa 18 prosessin tietojärjestelmästä käytetään nimitystä prosessin informaatiojärjestelmä. Tehdastason tietojärjestelmiin kuuluu hallinnon, osastojen yhteisiä ja tilaustenkäsittelyn tietojärjestelmiä. Kuvassa 18 on esitetty esimerkinomaisesti ainoastaan kaksi osastoa: paperikone 3 (PK3) ja painehiomo (PGW). Nuolilla kuvataan tiedonsiirtotarvetta. Kuvattu tavoitetila on pääosin toteutunut.



Kuva 18. Prosessin tietojärjestelmien liittyminen tehdastason tietojärjestelmiin (Auvinen, 1997).

4.2 Kokeilun tavoitteet

Prosessin tietojärjestelmien välisen tiedonsiirron kehittämiseksi UPM-Kymmene Oyj Kajaanissa asetettiin tavoitteeksi selvittää, mitä vaihtoehtoja tiedonsiirrossa käytetyille tiedostosiirrolle ja sanomanvälitykselle on olemassa. Tavoitteena on vähentää ohjelmointityötä siten, että tiedonsiirto olisi parametroinnilla muutettavissa. Erääksi vaihtoehdoksi havaittiin tietokantojen replikointi. Jotta replikointiin liittyvät tekniset yksityiskohdat tulisivat yksityiskohtaisesti selvitetyiksi, päätettiin suorittaa käytännön kokeilu joillakin tietokantatuotteilla. Kokeilussa pyrittiin selvittämään:

- Kuinka monimutkaista replikoinnin hallinta on?
- Kuinka replikointi toimii suurten tapahtumamäärien yhteydessä?
- Kuinka replikointi toimii poikkeustilanteissa?

Kysymyksiin haettiin vastauksia rakentamalla prototyyppi paperitehtaan tuotannon prosessitietokannasta, suorittamalla prototyypillä koeajoja ja arvioimalla tuotteita viitekehyksellä. Kokeilu tuli myös suorittaa siten, ettei tuotantokäytössä oleviin järjestelmiin heijastu minkäänlaisia sivuvaikutuksia.

4.3 Kokeilua edeltäneet toimenpiteet

Ennen kokeilua suoritettiin joukko edeltäviä toimenpiteitä. Kokeilua varten hankittiin koekäyttöön kaksi tietokannan hallintajärjestelmää. Ohjelmistojen valintaa ja yleisemminkin koekäyttöä selvitetään kohdassa 4.3.1. Kokeiluympäristön suunnittelua ja rakentamista kuvataan kohdassa 4.3.2. Kokeilun tavoitteiden selvittämiseksi viitekehyksestä valittiin osa arviointikriteereistä. Kriteerit ja niiden valinnan taustalla olleet tekijät esitellään ja perustellaan kohdassa 4.3.3.

4.3.1 Tietokannan hallintajärjestelmien valinta koekäyttöön

Kokeilua varten hankittiin koekäyttöön[10] kaksi tietokannan hallintajärjestelmää. Koska UPM-Kymmene Oyj Kajaanissa käytettiin jo Oraclen tietokantatuotteita, oli luonnollinen valinta hankkia toiseksi kokeiltavaksi tietokannan hallintajärjestelmäksi Oracle*9i* release 2 (9.2) for Windows. Valintaan vaikutti myös Oraclen merkittävä asema replikoinnin alueella. Toiseksi tietokannan hallintajärjestelmäksi valittiin yrityksen toimesta Sybase Adaptive Server Enterprise 12.5. Sybasen Replication Server -tuotetta on käytetty aiemminkin metsäteollisuudessa (Sybase Finland, 1996).

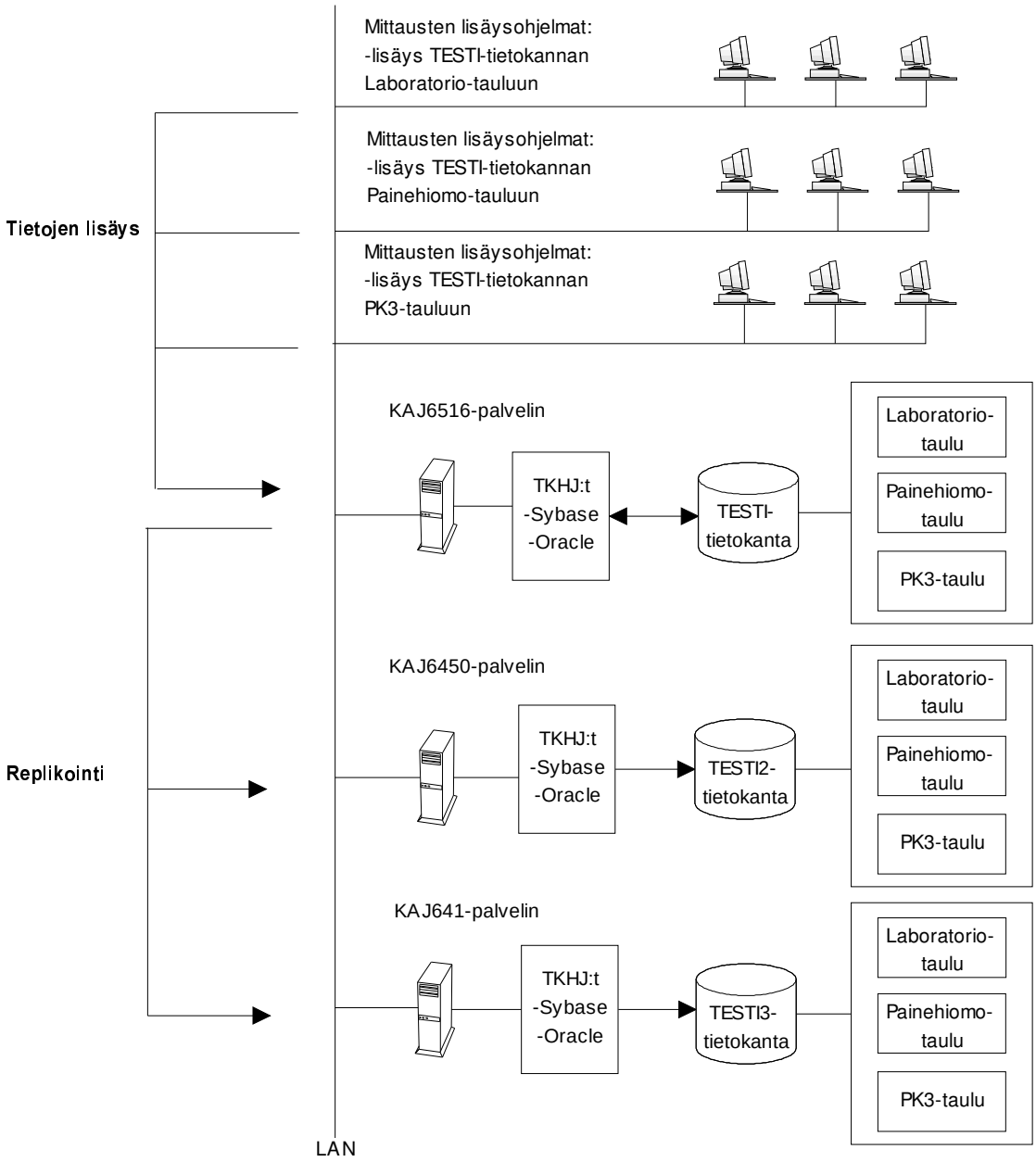
[10] Ohjelmien koekäytöllä tarkoitetaan järjestelyä, jossa asiakas on oikeutettu käyttämään ohjelmaa korvausta vastaan tietyn ajan kokeilumielessä. Ohjelmamedia toimitetaan asiakkaalle yleensä CD-ROM-levyllä, josta ohjelma voidaan asentaa tarvittaviin tietokoneisiin koekäyttösopimuksen lisenssimäärän huomioiden. Koekäytön päätyttyä ohjelmisto palautetaan myyjälle. Koekäyttöä käytetään varsinkin suurten kauppojen yhteydessä, jotta asiakkaan ei tarvitsisi ostaa ohjelmaa tutustumatta. Kaikkia ohjelmia ei välttämättä voi saada koekäyttöön.

4.3.2 Kokeiluympäristön suunnittelu ja rakentaminen

Asetettujen tavoitteiden selvittämiseksi päätettiin suunnitella ja toteuttaa replikointiratkaisu molemmilla tietokantatuotteilla. Tätä varten hankittiin kolme tuotantokäytöstä erillään olevaa mikrotietokonetta. Tietokoneet liitettiin tehtaan lähiverkkoon. Koneiden laitteisto- ja ohjelmistokokoonpanot on esitetty liitteessä 1. Koekäyttöön hankitut tietokannan hallintajärjestelmät asennettiin tietokoneisiin. Tuotannonhallinnan tietokantoja suunniteltaessa pyrittiin relaatiorakenteissa ja nimeämiskäytännöissä jäljittelemään todellista tilannetta. Koeajojen tietokantojen taulumääritykset on esitetty liitteessä 2. Tietokannat luotiin käyttämällä tietokannan hallintajärjestelmien oletustallennusrakennetta. Sen jälkeen laadittiin testiohjelmat keinotekoisten mittaustapahtumien lisäämiseksi tietokantaan. Ohjelmat asennettiin yhdeksään tehtaan lähiverkossa olevaan työasemaan. Näiden toimenpiteiden jälkeen aloitettiin replikointiin tutustuminen.

Kuvassa 19 esitetään kokeilun laitteisto- ja ohjelmistokomponenttien sijoittuminen. KAJ6516-palvelimen tietokannan hallintajärjestelmällä luotiin TESTI-niminen tietokanta, jonka tietokantatauluihin mittausten lisäysohjelmat tuottivat koeajoissa tietokantatapahtumia. Lisäksi KAJ6450- ja KAJ641 -palvelimiin toteutettiin TESTI2- ja TESTI3 -nimiset tietokannat, joiden tietokantataulujen rakenne vastasi TESTI-tietokannan taulujen rakennetta. Tavoitteeksi asetettiin, että mittaustapahtumat replikoitaisiin TESTI-tietokannasta TESTI2- ja TESTI3 –tietokantoihin. Replikointi-nopeus olisi mahdollisessa sovellustilanteessa noin 20 tietokantatapahtumaa/taulu/min. Replikoinnin perustamiseksi tarvittavat komennot annettiin palvelinten tietokannan hallintajärjestelmillä. Oraclen replikointi toteutettiin tahdistamattomana (vrt. B.1.6.) Advanced Replication –vaihtoehdolla, jossa tekniikkana käytettiin tilannevedosta (vrt. B.1.5.). Toinen vaihtoehto olisi ollut käyttää tuotteen Streams –replikointia, mutta koska Advanced Replication on ollut tuotteessa kauemmin, päätettiin käyttää sitä. Sybasen replikointi toteutettiin tahdistamattomana (vrt. B.1.6.) Replication Server –vaihtoehdolla, jossa käytettiin tekniikkana tietokantatapahtumien välittämistä (vrt.

B.1.5.). Sybasen ASE Replicator olisi ollut toinen vaihtoehto, mutta se sopii valmistajan mukaan suorituskyvyltään vähemmän kriittisiin ympäristöihin (Sybase, 2002a, s. 13).



Kuva 19. Kokeilun laitteisto- ja ohjelmistokomponentit.

4.3.3 Arviointikriteerien valinta

Viitekehyksen laajuuden ja käytettävissä olevien resurssien rajallisuuden vuoksi kokeiluun päätettiin valita ainoastaan osa kehyksen kriteereistä. Kokeilulle asetettujen tavoitteiden selvittäminen oli valinnan ensisijaisena lähtökohtana. Replikoinnin hallinta ja sen toiminta suurten tapahtumamäärien yhteydessä sekä poikkeustilanteissa olivat keskeisimmin näiden yrityksen lähtökohdista valittujen kriteerien taustalla. Toisaalta valinta antoi mahdollisuuden arvioida replikointia yleisemminkin. Esimerkiksi tuotteen teoreettisen taustan selvittäminen ei olisi ollut välttämätöntä pelkästään kokeilun tavoitteiden selvittämiseksi. Seuraavassa esitetään perustelut arviointikriteerien valinnalle.

Tietohallinnon näkökulmasta valittiin pääryhmä ”*Laitteisto- ja ohjelmistoarkkitehtuurin määrittäminen*” kriteeriä ”A.3 Millaista tukea toimittaja tarjoaa?” lukuun ottamatta, sillä pääryhmän katsottiin olevan merkittävä mahdollisessa tulevassa soveltamistilanteessa. Tietohallinnon kustannus/hyöty- analyysin suorittamista ei katsottu tarpeelliseksi, koska kustannusten ja hyötyjen selvittäminen ei ollut tutkimuksen tavoitteena. Tietokannan hoitajan näkökulmasta valittiin pääryhmät ”*Replikointiympäristön suunnittelu*”, ”*Replikointiympäristön toteutus*” ja ”*Replikointiympäristön käyttö*”, sillä ne ovat kaikki merkittäviä replikoinnin hallinnan kannalta ja niistä saatua tietoa voitaisiin käyttää vertailtaessa replikointia muihin tiedonsiirtovaihtoehtoihin. Kriteerit ”B.3. Ohjeistuksessa mahdollisesti olevat esimerkkisovellukset”, ”B.6. Tuotteen asennus”, ”B.9. Tuotteen poistaminen”, ”B.10.5. Poikkeuksellisten pyyntöjen toteuttaminen” ja ”B.10.7. Etävalvonta” päätettiin kuitenkin jättää tarkastelun ulkopuolelle, koska ne eivät olleet keskeisimpiä tavoitteiden kannalta. Etenkin ”*Replikointiympäristön toteutus*” -pääryhmän kriteerin ”B.7.2. Liittymät tietolähteisiin” ja ”*Replikointiympäristön käyttö*” -pääryhmän kriteerin ”B.10.3. Poikkeustilanteiden käsittely” selvittämistä pidettiin tärkeänä kokeilun tavoitteisiin pääsemiseksi. Ohjelmoijan näkökulmasta valittiin kriteeri pääryhmästä ”Normaali sovellusohjelmointi”, sillä sen katsottiin riittävän sovellusohjelmoijan tehtävien kattamiseksi. Lisäksi haluttiin selvittää replikointia

peruskäyttäjän näkökulmasta pääryhmän ”*Replikoinnin vaikutusten näkyminen*” kriteerillä ”D.3. Suorituskyky”. Taulukoissa 6, 7, 8 ja 9 esitetään tummennettuina kokeiluun valitut arviointikriteerit.

Taulukko 6. Kokeiluun valitut tietohallinnon arviointikriteerit.

A. Tietohallinto

Laitteisto- ja ohjelmistoarkkitehtuurin määrittäminen

A.1. Kuinka yhteensopiva tuote on muiden tuotteiden kanssa?
A.2. Kuinka kiinteästi ominaisuus on integroitu tietokannan hallintajärjestelmään?
A.3. Millaista tukea toimittaja tarjoaa?
A.4. Arvio ominaisuuden soveltuvuudesta erilaisiin käyttötarkoituksiin
A.5. Arvio ominaisuuden soveltuvuudesta aiottuun tehtävään

Kustannus/hyöty -analyysi

A.6. Kustannukset
- A.6.1. Replikointiohjelma
- A.6.2. Laitteisto
  - A.6.2.1. Levytila
  - A.6.2.2. Lähdejärjestelmän päivitys
  - A.6.2.3. Kohdejärjestelmän päivitys
- A.6.3. Tietoliikenneverkko
- A.6.4. Asennus ja toteutus
- A.6.5. Koulutus

A.7. Hyödyt
- A.7.1. Tiedon paikallinen saanti
  - A.7.1.1. Vasteaikojen nopeutuminen
  - A.7.1.2. Saatavuuden paraneminen
  - A.7.1.3. Prosessori- ja verkkopäivitysten siirtäminen
- A.7.2. Kopioidenhallintakustannusten aleneminen
- A.7.3. Replikoinnin tarjoamat mahdollisuudet
  - A.7.3.1. Uuteen teknologiaan siirtyminen
  - A.7.3.2. Tietoturvan paraneminen

Taulukko 7. Kokeiluun valitut tietokannan hoitajan arviointikriteerit.

B. Tietokannan hoitaja

Replikointiympäristön suunnittelu

B.1. Tuotteen teoreettinen tausta
- **B.1.1. Teoreettisen taustan esittely**
- **B.1.2. Replikoinnin hallinnan protokolla**
- **B.1.3. Replikointitarpeen ilmeneminen**
- **B.1.4. Replikoinnin kohde**
- **B.1.5. Käytettävä tekniikka**
- **B.1.6. Ajantasaisuus**
- **B.1.7. Päivitysmalli**

B.2. Tietokannan hoitajalta edellytettävät taustatiedot

B.3. Ohjeistuksessa mahdollisesti olevat esimerkkisovellukset

B.4. Asennusvaatimukset

B.5. Suorituskyvyn ennakointi

Replikointiympäristön toteutus

B.6. Tuotteen asennus

B.7. Replikoinnin hallintaohjelma
- **B.7.1. Ohjelman yleiset ominaisuudet**
- **B.7.2. Liittymät tietolähteeseen**
  - **B.7.2.1. Määrittäminen**
  - **B.7.2.2. Poistaminen**

B.8. Replikaattien alustaminen

B.9. Tuotteen poistaminen

Replikointiympäristön käyttö

B.10. Replikoinnin hallintaohjelma
- **B.10.1. Replikoinnin käynnistäminen ja pysäyttäminen**
- **B.10.2. Replikointiympäristön mukauttaminen**
- **B.10.3. Poikkeustilanteiden käsittely**
- **B.10.4. Replikointiympäristön varmistaminen**
- B.10.5. Poikkeuksellisten pyyntöjen toteuttaminen
- **B.10.6. Suorituskyvyn seuranta**
- B.10.7. Etävalvonta

Taulukko 8. Kokeiluun valittu sovellusohjelmoijan arviointikriteeri.

C. Sovellusohjelmoija

Replikointiympäristön ohjelmointi

C.1. Liittymien ohjelmointi tietolähteisiin

C.2. Raportoinnin ja hälytysten ohjelmointi

Normaali sovellusohjelmointi

C.3. Ohjelmointitehtävien huomioiminen

C.4. Vertaisreplikoinnin päivityskonfliktit

Taulukko 9. Kokeiluun valittu peruskäyttäjän arviointikriteeri.

D. Peruskäyttäjä
Replikoinnin vaikutusten näkyminen
D.1. Vasteaika D.2. Saatavuus **D.3. Suorituskyky**

4.4 Koeajot

Kokeiluympäristön rakentamisen jälkeen molemmilla tietokantatuotteilla suoritettiin koeajoja sen selvittämiseksi, kuinka replikointi toimii erilaisilla tapahtumamäärillä. Lisäksi syvennettiin arviointikriteereillä saatavaa tietoa replikoinnin toimivuudesta poikkeustilanteissa. Koeajoissa noudatettiin seuraavaa käytäntöä:

- Kukin koeajo ajettiin molemmilla tietokannan hallintajärjestelmillä.
- Palvelimet (KAJ6516, KAJ6450 ja KAJ641) käynnistettiin uudelleen ennen kutakin koeajoa.
- Palvelimissa ei ollut käynnissä koeajojen aikaan muita suorituskykyyn vaikuttavia ohjelmia.
- Ennen koeajon käynnistämistä varmistettiin, että tietokannan hallintajärjestelmät olivat käynnissä.
- Tietokantatauluista poistettiin kaikki rivit ennen kunkin koeajon aloittamista.
- Mittausten lisäysohjelmat käynnistettiin yhdeksässä työasemassa seuraavasti:
  -kolme työasemaa lisäsi mittaustietoja TESTI-tietokannan Laboratorio-tauluun
  -kolme työasemaa lisäsi mittaustietoja TESTI-tietokannan Painehiomo-tauluun
  -kolme työasemaa lisäsi mittaustietoja TESTI-tietokannan PK3-tauluun
- Mittausten lisäysohjelmassa asetettiin haluttu nopeus (1/sekunti tai 1/60 sekuntia) kunkin koeajon vaatimusten mukaan.

- Replikoitujen rivien määrää ja tietosisältöä seurattiin tietokannan hallinta-järjestelmien apuohjelmilla suorittamalla minuutin välein SQL:n SELECT-komentoja TESTI2- ja TESTI3 –tietokantojen tauluille.

Koeajoilla pyrittiin jäljittelemään useita mahdollisia käytännön tilanteita. Normaalin toiminnan selvittämisessä tarkasteltiin replikoinnin toimintaa pienellä päivitys-intensiteetillä. Erästä tyypillistä poikkeustilannetta, palvelimen vikaantumista, kokeiltiin sammuttamalla ja uudelleen käynnistämällä toinen palvelimista (KAJ6450). Muita mahdollisia poikkeustilanteita ovat esimerkiksi levyn rikkoontuminen ja verkkoliikenteen ongelmat. Näitä tilanteita ei kuitenkaan kokeiltu koeajoissa. Kokeilun palvelimissa ei ollut yrityksen muissa palvelimissa yleisesti käytettävää RAID (Redundant Arrays of Inexpensive Disks) –levyjärjestelmää, joten järjestely ei täysin vastaa todellista tilannetta. Tulevaisuuden sovellustilannetta jäljiteltiin huomattavan suurella päivitysintensiteetillä. Koeajojen kestot rajattiin alle puoleen tuntiin, sillä järjestelyllä saatiin edustava määrä replikointitapahtumia. Tietokantojen taulujen koon kasvamisella ei katsottu olevan vaikutusta koeajojen tuloksiin, sillä testiohjelmien lisäyskomennot (SQL:n INSERT-komennot) toimivat suorituskykyisesti koeajojen ajan.

Koeajojen tulokset vastasivat pääosin ohjeistuksesta ja ohjelmistojen maahantuojilta saatuja ennakkotietoja, eikä tuotteiden välillä ollut merkittäviä eroja. Molempien tuotteiden suorituskyky riittäisi aiottuun sovellustilanteeseen (noin 20 tietokantatapahtumaa/taulu/min), sillä replikointi toimi ongelmitta suuremmallakin päivitysintensiteetillä. Kokeiltaessa replikoinnin toimintaa suurella päivitys-intensiteetillä törmättiin kuitenkin odottamattomaan tietokantojen suunnitteluun liittyvään tekijään. Replikoinnissa kullakin tietokantataululla on oltava yksiselitteinen indeksi. Koeajossa tietokantatauluihin tuli useita tietokantatapahtumia sekunnissa, joten käytetty indeksi VVVVKKPPHHMMSS ei toiminut. Ongelma korjattiin muuttamalla indeksi muotoon VVVVKKPPHHMMSS+LAHDE, jossa LAHDE kuvasi mittauksen lähdetyöaseman nimeä (esimerkiksi 20030115193045W0008933, jossa W0008933=työaseman nimi). Korjauksen jälkeen uudelleensuoritetut koeajot toimivat moitteitta. Koeajoissa jäljiteltiin palvelimen vikaantumista sammuttamalla ja

käynnistämällä uudelleen eräs palvelimista (KAJ6450). Virheen korjaaminen ei vaatinut käyttäjältä toimenpiteitä, vaan replikointi jatkui uudelleenkäynnistyksen jälkeen. Ohjelmistoissa käytetään tapahtumajonoja tällaisten väliaikaisten käyttökeskeytysten hallintaan. Koeajoissa ei kokeiltu, kuinka ohjelmistot olisivat toimineet monimutkaisemmissa virhetilanteissa. Taulukossa 10 on esitetty yhteenveto suoritetuista koeajoista. Yksityiskohtaisempia arvioita replikoinnin toimivuudesta esitetään jatkossa arviointikriteerien läpikäynnin yhteydessä.

Taulukko 10. Suoritetut koeajot.

Koeajo	Tavoite	Kesto	Päivitysintensiteetti	Havainnot
1	Kokeilla replikoinnin toimintaa pienellä päivitysintensiteetillä	15 min	Laboratorio-taulu: 3 / min Painehiomo-taulu: 3 / min PK3-taulu: 3 / min	-Replikointi toimi moitteitta: TESTI2 - ja TESTI3 -tietokantoihin mittaustapahtumia
2	Kokeilla replikoinnin toimintaa pienellä päivitys-intensiteetillä poikkeus-tilanteessa. KAJ6450-palvelin sammutetaan 3 min kuluttua ajon alusta ja käynnistetään uudelleen 10 min kuluttua ajon alusta.	25 min	Laboratorio-taulu: 3 / min Painehiomo-taulu: 3 / min PK3-taulu: 3 / min	-Replikointi toimi moitteitta: TESTI2 - ja TESTI3 -tietokantoihin mittaustapahtumia
3	Kokeilla replikoinnin toimintaa suurella päivitysintensiteetillä	15 min	Laboratorio-taulu: 3 / sek Painehiomo-taulu: 3 / sek PK3-taulu: 3 / sek	-Replikointi toimi moitteitta: TESTI2 - ja TESTI3 -tietokantoihin mittaustapahtumia tälläkin päivitysintensiteetillä

4.5 Arvioinnin tulokset

Tässä kohdassa esitetään replikoinnista tehdyt havainnot arviointikriteereittäin. Kokeilun lisäksi lähteinä on käytetty ohjelmistojen ohjekirjoja ja esitteitä. Kriteerit on esitetty siinä järjestyksessä kuin ne ovat viitekehyksessä.

A. Tietohallinto

Laitteisto- ja ohjelmistoarkkitehtuurin määrittäminen

A.1. Kuinka yhteensopiva tuote on muiden tuotteiden kanssa?

Oracle replikoi ainoastaan Oracle-tietokannoista toisiin Oracle-tietokantoihin. Oracle-tietokantoihin on kuitenkin mahdollista replikoida myös muiden valmistajien tietokannoista (Oracle, 2002e, s. 39).

Sybase Replication Server replikoi seuraaviin tietokantoihin:

- Sybase Adaptive Server Enterprise, DB2 UDB, Oracle, Informix, Microsoft SQL Server ja mikä tahansa ODBC-yhteensopiva tietokanta.

Sybase Replication Server replikoi seuraavista tietokannoista (Sybase, 2002g):

- Sybase Adaptive Server Enterprise, DB2 UDB, Oracle, Microsoft SQL Server ja Informix.

A.2. Kuinka kiinteästi ominaisuus on integroitu tietokannan hallintajärjestelmään?

Oraclen replikointitoteutus on rakennettu tietokannan hallintajärjestelmän sisään (C-ohjelmointikielellä) ja ulkoisiin funktiokirjastoihin (PL/SQL-ohjelmointikielellä). (Oracle, 2001 ja Oracle, 2002a). Toteutuksessa käytetään hyväksi tietokannan hallintajärjestelmän eräajojonoja (job queues) ja eräajoprosesseja (job processes). PL/SQL-ohjelmointikielen tuntemusta suositellaan replikoinnin hallintaan (Oracle, 2002a).

Sybasen replikointi on toteutettu tietokannan hallintajärjestelmän rinnalla toimivaan Replication Server -ohjelmaan. Tämä ohjelma toimii esimerkiksi Windows NT -käyttöjärjestelmässä erillisenä prosessina, joka voidaan tarvittaessa pysäyttää ja käynnistää myös NT:n omilla toiminnoilla. Täysin irrallaan tietokannan

hallintajärjestelmästä toteutus ei kuitenkaan ole, sillä replikointikokoonpanoa ylläpidetään Sybase Adaptive Server -tietokannassa.

A.4. Arvio ominaisuuden soveltuvuudesta erilaisiin käyttötarkoituksiin

Oracle ja Sybase tarjoavat esitteissään seuraavia replikoinnin sovelluskohteita (Oracle, 2001; Oracle, 2002a; Oracle, 2002d ja Sybase, 2002e):

- tiedon kokoaminen tapahtumankäsittelyjärjestelmistä päätöstukijärjestelmään,
- tiedon hajauttaminen tapahtumankäsittelyjärjestelmien kesken ja
- tiedon varmuuskopiointi.

A.5. Arvio ominaisuuden soveltuvuudesta aiottuun tehtävään

Kehyksen soveltamisessa aiotulla tehtävällä tarkoitettiin sitä, voisiko replikointi korvata tulevissa tietojärjestelmissä nykyiset tiedonsiirtoratkaisut.

Oracle ja Sybase soveltuvat varauksin aiottuun tehtävään. Tuotteiden suorituskyky ja toimintavarmuus riittävät koeajojen perusteella prosessitietokantojen yhteyteen. Ongelmalliseksi tulevassa soveltamistilanteessa voisi sen sijaan muodostua tuotteiden replikoinnin hallinta ja seuranta. Sybasessa ongelmat ovat ennen kaikkea replikoinnin määrittelyssä (vrt. B.7.1. ja B.10.2.) ja Oraclessa suorituskyvyn ennakoinnissa (vrt. B.5.). Yrityksen tavoitteena ollut ylläpitotyön vähentäminen parametroitavalla tiedonsiirtotavalla ei juurikaan toteutuisi replikoinnilla, koska replikoinnin hallinta osoittautui varsin monimutkaiseksi (vrt. B.7.1. ja B.10.2.9.).

B. Tietokannan hoitaja

Replikointiympäristön suunnittelu

B.1. Tuotteen teoreettinen tausta

B.1.1. Teoreettisen taustan esittely

Oracle esittelee hajautetun tietojenkäsittelyn taustaa varsin runsaasti ohjekirjoissaan. Esillä ovat hajautetun tietojenkäsittelyn peruskäsitteistä esimerkiksi paikka-riippumattomuus, riippumattomuus tietojen replikoinnista ja kaksivaiheinen vahvistus (Oracle, 2002b). Replikointi määritellään tässä tutkimuksessa esitetyllä tavalla. Oracle ei ole kytkenyt käyttämiään käsitteitä tietokantateoriaan kovin vahvasti. Tahdistettu ja tahdistamaton replikointi sekä kaksivaiheinen vahvistus esitellään hyvin.

Sybase ei kuvaa hajautettua tapahtumankäsittelyä kattavasti. Replikointi kuvataan esimerkein lähinnä vaihtoehtoisena ratkaisuna keskitetylle tietojenkäsittelylle (Sybase, 2002e).

B.1.2. Replikoinnin hallinnan protokolla

Kummankaan tietokannan hallintajärjestelmän ohjeistuksesta ei ollut löydettävissä, mitä replikoinnin hallinnan protokollaa käytetään. Tutkimuksen edetessä kävi ilmi, että esitettyjen replikoinnin hallinnan protokollien toteuttaminen on jätetty ohjelmoijan ja tietokannan hoitajan vastuulle. Käytännössä tämä voi tarkoittaa esimerkiksi sitä, että päivitykset rajoitetaan sovelluksessa ainoastaan pääkopioon. Hajautetulle tietokannan hallintajärjestelmälle esitetty tavoite ”Riippumattomuus tietojen replikoinnista” ei siis täyty kummankaan tietokannan hallintajärjestelmän osalta.

B.1.3. Replikointitarpeen ilmeneminen

Oracle saa tiedon replikointitarpeesta herätteen avulla (Oracle, 2002a, s. 2-8, 2-10 ja 3-52).

Sybase saa tiedon replikointitarpeesta tietokantalokista. Erillinen Log Transfer Manager -prosessi tutkii replikoitavan tietokannan tietokantalokia ja mikäli lokista löytyy replikoitavaa tietoa, se välitetään Replication Serverille (Kirkwood, 1999, s. 29 ja Sybase, 2002g, s. 13)).

B.1.4. Replikoinnin kohde

Oracle replikoi tiedon lisäksi muun muassa tietokantatauluihin liittyviä näkymiä, herätteitä, indeksejä ja tallennettuja proseduureja (Oracle, 2002a, s. 1-4). Lisäksi tietokannan kuvauskielen komennot välittyvät myös replikaatteihin.

Sybase replikoi tiedon lisäksi tallennettuja proseduureja (Sybase, 2002e, s. 3).

B.1.5. Käytettävä tekniikka

Oraclessa käytetään replikoinnissa useita eri tekniikoita. Tahdistamattomassa replikoinnissa luodaan herätteen käynnistämänä tapahtumajonoon (deferred transaction queue) proseduurikutsu, joka lähetetään parametreineen suoritettavaksi kaikkiin haluttuihin tietokantasolmuihin eräajojonoa (job queue) käyttäen (Oracle, 2002a, s. 2-30 ja 2-31). Tahdistetussa replikoinnissa käytetään proseduurikutsuja ilman jonoja (Oracle, 2002a, s. 2-42). Tahdistamaton replikointi voidaan toteuttaa myös tilannevedosta käyttäen (Oracle, 2002a, s. 3-2).

Sybasessa replikoidaan joko tietokantatapahtumia tai kutsuja tallennettuihin proseduureihin (Sybase, 2002e, s.21).

B.1.6. Ajantasaisuus

Oraclella voidaan toteuttaa tahdistamaton ja tahdistettu replikointi (Oracle, 2002a, s. 2-7).

Sybasella voidaan toteuttaa tahdistamaton replikointi. Tahdistettua replikointia ei voida toteuttaa Replication Serverillä.

B.1.7. Päivitysmalli

Oraclella voidaan toteuttaa esitetyistä päivitysmalleista yksisuuntainen replikointi, lähettäminen, tallennus ja edelleenlähetys sekä vertaisreplikointi (Oracle, 2002a).

Sybasella voidaan toteuttaa yksisuuntainen replikointi, lähettäminen sekä tallennus ja edelleenlähetys (Sybase, 2002e).

B.2. Tietokannan hoitajalta edellytettävät taustatiedot

Oraclen ohjeistuksessa tietokannan hoitajan oletetaan tietävän relaatiotietokannoista, hajautetun tietokannan hallinnasta, PL/SQL-ohjelmointikielestä ja käyttöjärjestelmästä (Oracle, 2002a).

Sybasen ohjeistuksessa ei esitetä täsmällisesti ja kootusti tietokannan hoitajalta edellytettäviä taustatietoja. Sybase Adaptive Server Enterprisen hallinnan ja tietoliikenneverkon käsitteiden oletetaan olevan entuudestaan tuttuja tietokannan hoitajalle. Replication Serverin asennuksen apuna on mahdollista käyttää lomakkeistoa, jolla tarvittavat tiedot voidaan koota (Sybase, 2002d, s. 115). Sen täyttäminen voi kuitenkin osoittautua hankalaksi, elleivät taustatiedot ole selvillä.

Kokeiluympäristön rakentamisessa ilmeni, että replikoinnin perustaminen ja ylläpito edellyttävät hyvin vahvaa tuotteiden muidenkin ominaisuuksien hallintaa. Sen vuoksi kannattaa tarkkaan harkita, kuka ylläpitää tuotantoympäristöä.

B.4. Asennusvaatimukset

Oracle*9i* release 2 (9.2) for Windows asennusvaatimukset ovat:

- vähintään Pentium 166-prosessori,
- vähintään 128 MB keskusmuistia,
- vähintään 5 GB vapaata levytilaa,
- TCP/IP-lähiverkko ja
- Windows NT 4.0, Windows 2000 tai Windows XP –käyttöjärjestelmä (Oracle, 2002c, s. 2-4).

Sovellusohjelmoinnissa voidaan käyttää PL/SQL-ohjelmointirajapintaa (Oracle, 2002a).

Sybase Replication Server for Windows NT:n asennusvaatimukset ovat:

- vähintään Pentium -prosessori,
- vähintään 32 MB keskusmuistia,
- vähintään 160 MB vapaata levytilaa,
- TCP/IP -lähiverkko,
- Windows NT 4.0 tai Windows 2000 -käyttöjärjestelmä ja
- Sybase Adaptive Server -tietokannan hallintajärjestelmä (Sybase, 2002f, s.5).

Lisäksi lähde- ja kohdejärjestelmissä tarvitaan levytilaa virhetilanteiden hallinnassa käytettäviä tapahtumajonoja varten. Sovellusohjelmoinnissa voidaan käyttää mitä tahansa Sybase Open Server -ohjelmointirajapintaa tukevaa ohjelmointityökalua (Sybase, 2002g, s. 14).

B.5. Suorituskyvyn ennakointi

Oracle ei tarjoa minkäänlaisia välineitä, joilla replikoinnin suorituskykyä voitaisiin ennakoida.

Sybasessa on selkeät kaavat replikoinnin suorituskyvyn ennakointiin jo replikoinnin suunnitteluvaiheessa (Sybase, 2002e). Kaavoilla voidaan laskea, kuinka suuren tiedonsiirtokuorman replikointiratkaisu aiheuttaisi. Kokeilussa ei kuitenkaan sovellettu kaavoja, sillä laskenta olisi vaatinut huomattavan työmäärän.

Replikointiympäristön toteutus

B.7. Replikoinnin hallintaohjelma

B.7.1. Ohjelman yleiset ominaisuudet

Oraclen replikointia hallitaan Oracle Enterprise Manager –ohjelmalla. Ohjelmassa on toiminnot replikoinnin perustamiseen ja tilan seurantaan. Ulkoasussa on päädytty jakamaan työalue kahteen osaan. Vasemmalla olevasta luettelosta valitaan haluttu hallinnan osa-alue ja oikealle ilmestyvät osa-alueen kohteet (esimerkiksi virheelliset tietokantatapahtumat). Hiiren oikean näppäimen käyttömahdollisuus tehostaa käyttöliittymän käyttöä. Oracle Enterprise Manageria voi käyttää toisista tietokantasolmuista. Virheilmoitukset on toteutettu esille tulevilla virheikkunoilla. Ohjelman voi poistaa Oracle Installer –ohjelmalla.

Sybasessa replikointia hallitaan Replication Server Manager -ohjelmalla (Sybase, 2002g). Ohjelmalla ei voi suorittaa kaikkia replikoinnin hallinnan tehtäviä, sillä esimerkiksi Replication Serverin määritykset on tehtävä erillisellä rs_init-ohjelmalla. Replication Server Manager käyttämiseksi on ensin asennettava Replication Server Manager Server -ohjelma, joka kerää tietoa Replication Serveriltä replikoinnin toiminnasta. Ulkoasualtaan ohjelma noudattaa Windows-standardia. Värejä käytetään hyväksi tehokkaasti replikoinnin tilan ilmaisemisessa. Hiiren oikea näppäin avaa valikon. Virheilmoitukset on toteutettu esille tulevilla ikkunoilla. Ohjelmaa voidaan käyttää toisista tietokantasolmuista ja sen poistaminen on kuvattu ohjeistuksessa.

B.7.2. Liittymät tietolähteisiin

B.7.2.1. Määrittäminen

Oraclessa liittymät tietolähteisiin määritellään Oracle Enterprise Manager -ohjelmalla. Vaihtoehtoisesti voidaan käyttää DBMS_REPCAT-ohjelmapaketin komentoja. Ohjelma näyttää ikkunassa tietokannoittain replikoitavat kohteet (esimerkiksi taulut, näkymät ja indeksit), joista halutut kohteet ovat valittavissa. Suorituskyvyn kannalta epäedullisista valinnoista ei varoiteta. Käytettävissä on Oraclen tietotyypit LONG- ja LONG RAW -tietotyyppejä lukuun ottamatta. Replikoinnin ajastus on määriteltävissä aina sekunneista eteenpäin.

Sybasessa liittymät tietolähteisiin määritellään Replication Server Manager –ohjelmalla. Ohjelma näyttää ikkunassa olemassa olevat replikointimääritykset tietokantatauluittain. Kustakin tietokantataulusta voidaan tarvittaessa nähdä taulun sisältämät sarakkeet ja tietotyypit. Replikointimäärityksissä on käytettävissä SQL:n SELECT-komennon WHERE-valitsin. Suorituskyvyn kannalta epäedullisista valinnoista ei varoiteta. Replication Server replikoi kaikkia Sybase Adaptive Serverin tietotyyppejä käyttäjän luomia omia tietotyyppejä lukuun ottamatta. Ohjelmalla ei voi määrittää replikoinnin ajastusta.

B.7.2.2 Poistaminen

Oraclen Replication Manager -ohjelmalla voi poistaa esimerkiksi tietokantataulun kaikista tietokantasolmuista.

Sybase Replication Server Manager -ohjelmalla voi poistaa replikointimääritykset.

B.8. Replikaattien alustaminen

Oraclessa replikaattien alustamiseksi on kaksi vaihtoehtoa. Offline-alustuksessa käyttäjä voi laatia Enterprise Manager –ohjelmalla SQL-komentotiedoston ja varsinaiset tiedot sisältävän tiedoston, jotka toimitetaan siihen tietokantasolmuun, johon tietoja jatkossa replikoidaan. Online-alustuksessa replikaatit alustetaan ensimmäisen päivityskerran yhteydessä. Oraclen Enterprise Manager –ohjelmalla voidaan määritellä offline-alustukseen liittyviä tekijöitä.

Sybasen Replication Server Manager -ohjelmassa on käytettävissä seuraavat alustusvaihtoehdot (Sybase, 2002b, s. 325):

- Atomic:
  Lähdetietokannan taulut lukitaan ja kohdetietokantaan lisätään tietokantarivit yhtenä tietokantatapahtumana, jonka jälkeen lähdetietokannan lukitukset vapautetaan.
- Non-atomic:
  Kohdetietokantaan lisätään tietokantarivejä lähdetietokannasta kymmenen riviä kerrallaan. Lukitusta ei käytetä lähdetietokannassa.
- No materialization
  Replikaatteja ei alusteta.
- Bulk materialization
  Replikaatit alustetaan esimerkiksi magneettinauhalta.

Replikointiympäristön käyttö

B.10. Replikoinnin hallintaohjelma

B.10.1. Replikoinnin käynnistäminen ja pysäyttäminen

Oraclessa replikointi voidaan käynnistää ja pysäyttää Oracle Enterprise Manager -ohjelmalla.

Sybasessa replikointi käynnistetään ja pysäytetään Replication Server Manager -ohjelmalla (Sybase, 2002g). Lisäksi replikoinnin käynnistyminen on mahdollista yhdistää NT-käyttöjärjestelmän käynnistymiseen (Sybase, 2002d).

B.10.2. Replikointiympäristön mukauttaminen

Oraclessa replikointikokoonpanon muuttaminen on joustavaa, sillä tietokannan kuvauskielen komennot välittyvät replikaatteihin (Oracle, 2002a, s. 1-17). Tietokantasolmun lisääminen edellyttää ns. tietokantalinkin luomista. Sillä ilmaistaan, missä fyysinen tietokanta sijaitsee. Oracle Enterprise Manager:lla ei voi ennakoida muutosten vaikutuksia suorituskykyyn.

Sybasessa replikointiympäristön mukauttaminen on varsin monimutkaista. Taulumäärityksiä muutettaessa on ensin pysäytettävä replikointi, tehtävä muutokset ja käynnistettävä replikointi. Tietokantasolmuja lisättäessä ja poistettaessa on päivitettävä liitäntätiedostoja, joissa kerrotaan kunkin replikointikokoonpanon komponentin (Replication Server, Log Transfer Manager ja Sybase Adaptive Server) sijainti tietoliikenneverkossa (Sybase, 2002b, s. 73). Arvioitaessa mukauttamisen vaikutuksia suorituskykyyn, on sovellettava tuotteen mukana toimitettuja kaavoja uudelleen.

B.10.3. Poikkeustilanteiden käsittely

Oraclen Enterprise Manager -ohjelma esittää tietokantasolmuittain suorittamatta jääneet tietokantatapahtumat. Poikkeustilanteissa tapahtumat on mahdollista suorittaa uudelleen ohjelman avulla.

Sybasen Replication Server Manager -ohjelma tarjoaa hyvät mahdollisuudet replikoinnin poikkeustilanteiden seuraamiseen. Ohjelmalla voidaan havaita nopeasti, mistä mahdollinen ongelmatilanne johtuu. Käyttäjällä on mahdollisuus valita halutut virheluokat, vaikuttaa virhelokien päivitystiheyteen ja jopa vaihtaa lokissa esiintyvien virhetekstien värejä. Erittäin suuri puute kuitenkin on, että ohjelmalla ei voi korjata

esiintyviä ongelmia. Jos jotakin tietokantatapahtumaa ei esimerkiksi voida soveltaa kohdejärjestelmään, on käytettävä Replication Command Language:n (RCL) komentoja (Sybase, 2002c, s. 560).

B.10.4. Replikointiympäristön varmistaminen

Oraclessa varmistaminen hoidetaan tietokannan hallintajärjestelmän apuohjelmilla.

Sybasen replikointimääritykset tallennetaan Sybase Adaptive Server Enterprise -tietokantaan. Tämän tietokannan varmistaminen on osa ympäristön varmistamista.

B.10.6. Suorituskyvyn seuranta

Oraclen Enterprise Manager -ohjelmalla voi seurata replikoinnin suorituskykyä hyvin rajoitetusti. Tarkasteltavissa on lähinnä ainoastaan tapahtumajonot, joilla replikointi on toteutettu.

Sybasen Replication Server Manager -ohjelmalla voidaan mitata aikaa, joka kuluu tietokantatapahtuman vahvistamisen välillä lähde- ja kohdejärjestelmissä. Suorituskyvyn seurantamahdollisuudet ovat hyvin muunneltavissa ja laajennettavissa.

C. Sovellusohjelmoija

Normaali sovellusohjelmointi

C.3. Ohjelmointitehtävien huomioiminen

Oraclen ohjeistuksessa ei esitetä keskitetysti, kuinka replikointi vaikuttaa ohjelmointiin. Päivityskonfliktien hallinta on miltei ainoa kohta, jossa ohjelmoijan työtä käsitellään (Oracle, 2002a, s. 5-2).

Sybasessa keskeisimpänä periaatteena ohjelmoijan näkökulmasta on rajoittaa tietoihin kohdistuvat päivitykset ainoastaan pääkopioihin. Tämä seikka mainitaan ohjeistuksessa useissa eri kohdissa. Replication Server Design Guide kuvaa useita vaihtoehtoisia ratkaisuja pääkopion päivittämiseksi (Sybase, 2002e, s. 31).

D. Peruskäyttäjä

Replikoinnin vaikutusten näkyminen

D.3. Suorituskyky

Replikoinnin suorituskykyä selvitettiin koeajoilla. Paperitehtaan tuotannon prosessitietokannasta laadittiin prototyyppi, johon lisättiin keinotekoisia mittaus-tapahtumia suurella päivitysintensiteetillä. Koeajossa ilmeni, että Oracle ja Sybase pystyivät replikoimaan lähes kymmenen tietokantatapahtumaa sekunnissa. Koska mahdollisessa sovellustilanteessa replikointitarve olisi ainoastaan muutamia kymmeniä tietokantatapahtumia minuutissa, on tuotteiden suorituskyky hyvinkin riittävä.

4.6 Johtopäätökset

Seuraavassa esitetään johtopäätökset kolmesta keskeisestä osa-alueesta. Viitekehyksen kriteerien soveltamisella saatiin tietoa kahden tietokantatuotteen replikointi-ominaisuuksista. UPM-Kymmene Oyj Kajaanin kannalta oli keskeistä selvittää, voitaisiinko replikointia käyttää tuotannon prosessitietokantojen väliseen tiedonsiirtoon. Viitekehyksen soveltaminen antoi myös kuvan kehyksen toimivuudesta ja sen jatkokehitystarpeista yleisemminkin.

Tutkittujen tietokantatuotteiden replikoinnissa on sekä hyvää että huonoa. Sybasessa edut painottuvat etenkin replikoinnin suorituskyvyn ennakointiin ja seurantaan. Oraclessa myönteisenä voidaan pitää käyttöliittymän selkeyttä ja tapaa, jolla

replikoinnin hallintaohjelma on integroitu muuhun tietokannan hallinnan välineistöön. Puutteellisia piirteitä tarkasteltaessa Sybasessa erottuu replikoinnin hallintaohjelma. Se ei kata kaikkia replikoinnin hallinnan tehtäviä. Replikoinnin poikkeustilanteiden käsittely ei onnistu ohjelmalla kokonaisuudessaan. Sybase on päätynyt jakamaan replikoinnin hallinnan useisiin eri ohjelmiin ja sen vuoksi asennuksen ja käytön aikana tuntui kuin tuotteessa olisi ollut liikaa liikkuvia osia. Valmistajan kannattaisikin pohtia, onko asiakas/palvelin -arkkitehtuurin soveltaminen aina välttämätöntä. Esimerkiksi replikoinnin hallinnan tuotteiden Replication Server Manager Server ja Replication Server Manager yhdistäminen voisi tulla kyseeseen. Oraclessa puutteet painottuvat suorituskyvyn ennakointiin ja seurantaan. Valmistaja ei esitä missään, kuinka replikoinnin suorituskykyä voisi jo suunnitteluvaiheessa arvioida. Lisäksi replikoinnin hallintaohjelman avustustiedosto voisi olla kattavampi. Hajautetun tietokannan tavoitteena ollut riippumattomuus tietojen replikoinnista jää molemmissa tuotteissa niinikään saavuttamatta. Nähtäväksi jää, mikä replikoinnin rooli on tulevaisuudessa ja kuinka paljon tietokantatoimittajat panostavat ominaisuuden kehittämiseen. Todennäköisesti tietokantatoimittajat jatkavat replikoinnin edelleen kehittämistä, sillä he tuntevat erillisten replikointituotteiden valmistajia paremmin tietokantojen sisäisen toteutuksen. Tutkittujen tuotteiden perusteella on vaikea tehdä yleistäviä johtopäätöksiä muiden tietokannan hallintajärjestelmien replikoinnista. Valmistajat ovat kuitenkin törmänneet replikointiin liittyvään problematiikkaan ja tuotteistaneet omat ratkaisunsa.

Yrityksen kannalta tuotteiden replikointiominaisuudet riittävät varauksin prosessi-tietokantojen väliseen tiedonsiirtoon. Suoritettu kokeilu ja kriteeristön soveltaminen antoivat hyvän pohjan replikoinnin arvioinnille. Kokeilussa pyrittiin selvittämään:

1. Kuinka monimutkaista replikoinnin hallinta on?
2. Kuinka replikointi toimii suurten tapahtumamäärien yhteydessä?
3. Kuinka replikointi toimii poikkeustilanteissa?

Replikoinnin hallinta havaittiin kokeilussa varsin monimutkaiseksi. Kummankaan tuotteen replikoinnin hallintaohjelma ei ole niin kehittynyt, että yritys voisi sen avulla

joustavasti määritellä ja ylläpitää replikointia. Kohdealueellahan edellytetään nimenomaan joustavaa ylläpitoa, sillä tuotantoon liittyvien mittausten määrä kasvaa jatkuvasti. Lisäksi kokeilun aikana ilmeni lukuisia tilanteita, joiden selvittäminen edellytti myös tietokannan hallintajärjestelmän muiden osien perusteellista tuntemista.

Suorituskyvyn kannalta molemmat tuotteet toimisivat sovellustilanteessa. Mikäli nykyisten järjestelmien suorituskykyyn heijastuvat vaikutukset haluttaisiin mahdollisimman vähäisiksi, voidaan Sybasea pitää tuotteista parempana. Siinä tietokannan hallintajärjestelmä saa tiedon replikointitarpeesta tietokantalokista eikä tietokantatapahtuman yhteydessä suorittavasta herätteestä, joka pidentää tietokantatapahtuman kestoa (vrt. arviointikriteeri B.1.3).

Replikoinnin toimivuutta poikkeustilanteissa selvitettiin kokeilussa jäljittelemällä palvelimen vikaantumista. Molemmat ohjelmistot toimivat järjestetyssä poikkeustilanteessa. Koska tuotannon tietojärjestelmät ovat keskeisessä roolissa, kannattaa mahdollisessa sovellustilanteessa pyrkiä tekemään ratkaisusta mahdollisimman yksinkertainen ja panostaa poikkeustilanteiden hallinnan rutiinien kehittämiseen. Kokeilussa ei kyetty jäljittelemään kaikkia poikkeustilanteita.

Laadittu viitekehys soveltui varsin hyvin tietokannan hallintajärjestelmien replikoinnin arviointiin. Kehyksessä tarkastellaan replikointia neljästä näkökulmasta: 1) tietohallinto, 2) tietokannan hoitaja, 3) sovellusohjelmoija ja 4) peruskäyttäjä. Näkökulmista esitetyt kriteerit on edelleen ryhmitelty pääryhmien alle. Tietokannan hoitajan näkökulmassa on mukana ajallinen ulottuvuus replikoinnin suunnittelusta, toteutukseen ja käyttöön. Soveltamisen yhteydessä valittiin ensin näkökulmat, sen jälkeen pääryhmät ja lopuksi yksittäiset kriteerit. Tämä hierarkkinen rakenne ohjasi hyvin toimintaa. Yksittäisten kriteerien tasolla annetut ohjeet olivat pääosin riittäviä. Ohjelmistojen koekäyttö auttoi osaltaan soveltamista. Hieman epäselväksi jäi, kuinka soveltamiskelpoinen kehys olisi ollut ilman koekäyttöä. Kehystä laadittaessa soveltamisella arvioitiin saatavan hyötyä replikointikäsitteistön yhtenäistämisessä ja selkeyttämisessä. Tämä tavoite toteutui

selvästi. Viitekehys soveltunee sellaisenaan myös muiden tietokannan hallinta-järjestelmien replikoinnin arviointiin.

Viitekehyksessä ilmeni toki jatkokehitystä vaativia piirteitä. Täsmennettävää jäi etenkin peruskäyttäjän näkökulmaan, sillä vasteajan, saatavuuden ja suorituskyvyn mittaaminen esitetyillä kriteereillä ei onnistu helposti. Lisäksi muutamat kriteerit jäivät liian ympäripyöreiksi. Esimerkiksi arvioijan subjektiivinen vastaus tietokannan hoitajan kriteeriin ”B.1.1. Teoreettinen taustan esittely” ei välttämättä anna olennaista lisätietoa tuotteesta. Viitekehyksen laatimisen eräänä tavoitteena oli myös se, että kehys voisi ohjata replikoinnin suunnittelu- ja toteutusprosessia. Suoritetussa käytännön kokeilussa tämä tavoite ei tullut juurikaan esille. Osaltaan tämä johtui siitä, että lopputulos, tietynlainen kokeiluympäristö, oli selvillä jo varhaisessa vaiheessa. Tutkimuksen tekijän kokemattomuus relaatiotietokannoista ja replikoinnista kokeilun alkuvaiheessa vaikutti myös asiaan. Kehyksestä voisi tietenkin tehdä ohjaavamman. Koska replikoinnin suunnitteluun on olemassa kirjallisuutta (esimerkiksi Buretta, 1997), ei ohjaavuuden lisääminen viitekehykseen ole kuitenkaan ensisijainen jatkokehityskohde.

4.7 Yhteenveto kokeilusta

Luvussa 4 sovellettiin replikointiominaisuuksien arvioinnin viitekehystä tietokanta-tuotteiden arviointiin. Yrityksen tavoitteena oli selvittää, voidaanko tietokantojen replikointia käyttää tuotannon prosessitietokantojen yhteydessä. Etenkin replikoinnin hallinnasta, suorituskyvystä ja toiminnasta poikkeustilanteissa haluttiin lisää tietoa. Tavoitteeseen pääsemiseksi arvioitiin kahden koekäyttöön hankitun tietokantatuotteen replikointia valituilla viitekehyksen kriteereillä. Lisäksi laadittiin prosessitietokannan prototyyppi, jossa käytettiin replikointia. Koeajoissa tietokantaan tuotettiin keinotekoisia mittaustapahtumia replikoinnin toiminnan selvittämiseksi. Tulokset osoittivat, että replikointia voitaisiin suorituskyvyn ja toimintavarmuuden puolesta käyttää aiottuun tehtävään. Sen sijaan hankaliksi havaittiin replikoinnin hallintaan ja suorituskyvyn

seurantaan liittyviä piirteitä. Lisäksi replikointitoteutusten havaittiin olevan varsin kaukana hajautettujen tietokantojen tavoitteena olevasta riippumattomuudesta tietojen replikoinnista. Laadittu viitekehys osoittautui kokeilussa varsin toimivaksi.

5 YHTEENVETO

Tietokannan hallintajärjestelmien replikointiominaisuus on eräs runsaasti huomiota saanut tietojen hajautusvaihtoehto. Useimmilla markkinoilla olevista tietokannan hallinta-järjestelmistä voidaan toteuttaa ratkaisu, jossa tiedot toisinnetaan useisiin tietokoneisiin tietojärjestelmien saatavuuden ja toimintavarmuuden parantamiseksi. Replikointia on käytetty esimerkiksi tietovarastointi- ja varmistusratkaisuissa. Tietojen hajautusta suunniteltaessa ongelmaksi voi kuitenkin muodostua se, ettei ole olemassa juurikaan välineistöä tietokannan hallintajärjestelmien replikointiominaisuuksien arviointiin. Ongelmaa kärjistää usein myös valmistajien käyttämä vaihteleva terminologia, jonka ymmärtäminen esimerkiksi tietokannan hallintajärjestelmää valittaessa voi osoittautua hankalaksi. Replikoinnin suhde muihin hajautus-vaihtoehtoihin jää myös usein hämäräksi.

Tutkielmalle asetettiin seuraavat kolme tavoitetta:

1) luoda käsitteellinen perusta tietojen hajautuksen, hajautettujen tietokantojen ja replikoinnin käsitteiden, rakenteiden ja periaatteiden määrittelemiseksi
2) rakentaa viitekehys, jonka avulla voidaan arvioida tietokantatuotteiden replikointi-ominaisuuksia
3) testata viitekehyksen toimivuutta käytännön tilanteessa

Ensimmäistä tavoitetta lähestyttiin esittelemällä tutkielman luvussa 2 hajautetun tietojärjestelmän komponentit – tieto, käsittely ja esittäminen. Komponenttien erilaisten sijoitusvaihtoehtojen kuvaamisen jälkeen keskityttiin hajautettuihin tietokantoihin. Hajautettujen tietokantojen etuina pidetään yleisesti parantunutta suorituskykyä ja toimintavarmuutta. Ratkaisun yleistymistä on kuitenkin hidastanut monimutkaisuus ja vaikeampi hallittavuus. Replikointi tuotiin esille eräänä tiedon hajautusvaihtoehtona, ja vaihtoehdon valintaa luonnehdittiin yhteenvetotaulukolla. Hajautetun tietokannan hallintajärjestelmän esittelyn jälkeen keskityttiin hajautetun tapahtumankäsittelyn problematiikkaan. Tutkielman toinen luku toimii johdatuksena hajautettuun

tiedonhallintaan ja vastaa osaltaan tutkielman ensimmäiseen tavoitteeseen.

Tietokannan hallintajärjestelmien replikoinnin arviointiin esitettiin viitekehys tutkielman luvussa 3. Koska replikoinnin arviointiin ei ole olemassa juurikaan välineistöä, esitettiin luvussa tutkimuksien ja artikkeleiden perusteella kehys, jonka avulla tuotteiden replikointia voidaan arvioida. Lähtökohdaksi otettiin neljä eri näkökulmaa (tietohallinto, tietokannan hoitaja, sovellusohjelmoija ja peruskäyttäjä), joista kutakin tarkennettiin edelleen arviointikriteereiksi. Viitekehyksen arvioidaan soveltuvan sellaisenaan tietokannan hallintajärjestelmien replikoinnin arviointiin. Kehyksen hyödyt tulevat esille ennen kaikkea replikoinnin suunnittelun ja toteutuksen tehostumisena sekä käytettävän käsitteistön yhtenäistymisenä.

Tutkielman kolmanteen tavoitteeseen pääsemiseksi esitettyä viitekehystä kokeiltiin käytännössä. Toimeksiantajan osoittamassa käytännön kokeilussa viitekehystä sovellettiin kahden tietokannan hallintajärjestelmän arviointiin. Kokeilun tavoitteena oli selvittää, voisiko replikointi toimia tiedonsiirtovaihtoehtona prosessitietokantojen välillä. Erityistä huomiota pyrittiin kiinnittämään replikoinnin hallintaan, suorituskykyyn ja toimintavarmuuteen. Tavoitteeseen pääsemiseksi suunniteltiin prototyyppi paperitehtaan tuotannonhallinnan prosessitietokannasta. Prototyyppiä rakennettaessa tuotteita arvioitiin replikoinnin hallinnan näkökulmasta valituilla viitekehyksen kriteereillä. Replikoinnin suorituskykyä ja toimintavarmuutta arvioitiin koeajoilla ja viitekehyksen avulla. Tulokset osoittivat, että tutkittujen tuotteiden replikoinnin suorituskyky ja toimintavarmuus riittäisi toimeksiantajan prosessitietokantoihin. Tuotteiden hallinnasta ja suorituskyvyn seurannasta löytyi sen sijaan piirteitä, joita olisi syytä pohtia ennen mahdollista sovellustilannetta. Tutkielman neljäs luku antoi siis vastauksia viitekehyksen toimivuudesta ja replikoinnin soveltuvuudesta prosessitietokantojen väliseen tiedonsiirtoon.

Esitetyssä replikointiominaisuuksien arvioinnin viitekehyksessä ilmeni toki jatkokehitystä vaativia piirteitä. Eräät esitetyistä viitekehyksen kriteereistä jäivät tarkkuustasoltaan liian epätarkoiksi, eikä niiden selvittäminen tuotteesta tuo olennaisesti

lisää tietoa replikoinnista. Etenkin peruskäyttäjän kriteerejä olisi täsmennettävä. Viitekehyksessä ei myöskään ole esitetty kriteerejä tietoturvan näkökulmasta. Tietoturvan kriteerien lisääminen viitekehykseen olisi tärkeää, sillä replikointia toteutetaan myös Internetin kautta.

Tutkimuksen myötä esiin nousi mielenkiintoisia replikointiin liittyviä jatkotutkimus-aiheita. Tässä tutkielmassa replikointia arvioitiin relaatiopohjaisissa tietokannan hallintajärjestelmissä. Koska myös oliopohjaisissa tietokannoissa käytetään replikointia (esim. Objectivity), olisi kiinnostavaa tutkia, kuinka viitekehys toimisi oliopohjaisten tietokannan hallintajärjestelmien arvioinnissa. Toisen jatkotutkimusaiheen muodostaa tosiaikatietokannat (real time database, RTDB) ja niiden mahdolliset replikointi-mahdollisuudet.

LÄHTEET

Atk-sanakirja. 1996. Tietotekniikan liitto ry, Jyväskylä.

Auvinen J. 1997. UPM-Kymmene Oyj Kajaani Information Systems –kalvosarja.

Bacon C. 1992. The use of decision criteria in selecting information systems technology investments, MIS Quarterly, Vol. 16, No. 3, 335-353.

Breitbart Y., Komondoor R., Rastogi R., Seshadri S., Silberschatz A. 1999. Update propagation protocols for replicated databases [viitattu 8.2.2003]. Saatavilla ps-muodossa: <URL: http://www.bell-labs.com/ user/rastogi/replication.ps>.

Buretta M. 1997. Data replication tools and techniques for managing distributed information. John Wiley & Sons.

Ceri S., Houtsma M. A. W., Keller A. M., Samarati P. 1994. A classification of update methods for replicated databases [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://www-db.stanford.edu/ pub/keller/1991/fauve-survey.pdf>.

Ceri S., Pegalatti G. 1985. Distributed databases: principles and systems. McGraw-Hill, Singapore.

Date C. J. 1990. An introduction to database systems. 5th ed., Addison-Wesley.

Date C. J. 1995. An introduction to database systems. 6th ed., Addison-Wesley.

Dobson R. 1995. New replication options in Acces, Oracle, and Notes. Byte, Oct., 30.

Dobson R. 1996. Better replication coming for databases. Byte, May, 36.

Eckerson W. 1995. Oracle offers symmetric replication [viitattu 4.11.1996]. Saatavilla html-muodossa : <URL: http://www.oracle.com/products/servers/replication/html/press.html>.

Franch X., Carvallo J. 2003. Using quality models in software package selection. IEEE Computer, January / February, 34-41.

Frank L. 1988. Database theory and practice. Addison-Wesley.

Froemming G. 1996a. Design and replication issues with mobile applications, part 1 [viitattu 8.2.2003]. Saatavilla html-muodossa: <URL: http://www.dbmsmag.com/9603d13.html>.

Froemming G. 1996b, Moving forward with replication, part 2 [viitattu 8.2.2003]. Saatavilla html-muodossa: <URL: http://www.dbmsmag.com/9604d14.html>.

Galliers R. D. 1991. Choosing appropriate information systems research approaches: a revisited taxonomy. Teoksessa Nissen H., Klein H., Hirscheim R. (toim.) Information systems research: contemporary approaches and emergent traditions, North-Holland, Elsevier science publishers, 327-345.

Garcia-Molina H., Ullman J. D., Widom J. 2000. Database system implementation. Prentice-Hall.

Hoffer J., Prescott M., McFadden F. 2002. Modern database management. 6.th edition, Prentice-Hall.

Huttunen O. 1996. Tietokantapalvelimet hajautetuissa järjestelmissä. Jyväskylän yliopisto. Tietojärjestelmätieteen pro gradu –tutkielma.

Informix Software. 1996. Continous data replication [viitattu 6.3.1997]. Saatavilla html-muodossa: <URL: http://www.informix.com/informix/corpinfo/zines/whitpprs/datarep/datarep.htm>.

Jiminez-Peris R., Patino-Martinez M., Alonso G., Kemme B. 2001. How to select a replication protocol according to scalability, availability and communication overhead [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://lsd.ls.fi.upm.es/lsd/papers/2001/srds01.pdf>.

Kay A. S. 1996. Land of opportunity and turmoil. Datamation, June, 40-42.

Khoshafian S., Chan A., Wong A., Wong H. 1992. A guide to developing client/server SQL applications. Morgan Kaufmann, San Fransisco.

Kirkwood J., Arkle G. 1999. Sybase replication server. An administrator's guide. Kirkwood associates limited.

Kitchenham B. 1996-1997. Evaluating software engineering methods and tools. ACM Software Enigneering Notes, Vol. 21, no. 2 ja 4; Vol. 22, no. 1-2, 4-5.

Korth H. F., Silberschatz A. 1991. Database system concepts. 2nd ed., McGraw-Hill, Singapore.

Lamminmäki S., Hannus J. 1993. Avoimet ja hajautetut tietojärjestelmät. Arkkitehtuurit, teknologia, välineet [viitattu 11.9.1996]. Saatavilla html-muodossa: <URL: http://www.hmv.fi/avohaj2.html>.

Liu M. L., Agrawal D., Abbadi A. E. 1994a. An efficient implementation of the quorum consensus protocol [viitattu 9.9.1996]. Saatavilla ps-muodossa: <URL: ftp.cs.ucsb.edu/techreports/TRCS94-13.ps>.

Liu M. L., Agrawal D., Abbadi A. E. 1994b. What price replication? [viitattu 9.9.1996]. Saatavilla ps-muodossa: <URL: ftp.cs.ucsb.edu/techreports/TRCS94-14.ps>.

Little M. C., McCue D. L. 1994. The replica management system: a scheme for flexible and dynamic replication [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://arjuna.ncl.ac.uk/publications/papers/1994/1.pdf>.

Lubinski A., Heuer A. 2000. Configured replication for mobile applications [viitattu 8.2.2003]. Saatavilla ps-muodossa: <URL: http://wwwdb.informatik.uni-rostock.de/~lubinski/artikel/rib00.ps>.

Mosley V. 1992. How to assess tools efficiently and quantitatively. IEEE Software, No. 5, 29-32.

Muthitacharoen A., Morris R., Thomer M. G., Benjie C. 2002. Ivy: A read/write peer-to-peer file system [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://pdos.lcs.mit.edu/ivy/osdi02.pdf>.

Oracle Corporation. 2002a. Oracle9i advanced replication, release 2 (9.2) [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://download-east.oracle.com/docs/cd/B10501_01/server.920/a96567.pdf>.

Oracle Corporation. 2002b. Oracle9i database administrator’s guide, release 2 (9.2) [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://download-east.oracle.com/docs/cd/B10501_01/server.920/a96521.pdf>.

Oracle Corporation. 2002c. Oracle9i database installation guide, release 2 (9.2.0.1) [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://download-east.oracle.com/docs/pdf/A95493_01.pdf>.

Oracle Corporation. 2002d. Oracle9i data warehousing guide, release 2 [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL. http://download-east.oracle.com/docs/cd/B10501_01/server.920/a96520.pdf>.

Oracle Corporation. 2002e. Oracle9i heterogeneous connectivity administrator's guide [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://download-east.oracle.com/docs/cd/B10501_01/server.920/a96544.pdf>.

Oracle Corporation. 2001. Oracle9i replication [viitattu 8.2.2003]. Saatavilla pdf-muodossa:
<URL: http://otn.oracle.com/products/oracle9i/pdf/ oracle9i_replication_twp.pdf>.

Oracle Corporation. 1996. Replication in a heterogeneous environment [viitattu 15.12.1997]. Saatavilla pdf-muodossa: <URL: http://www.oracle.com/products/gateways/whitepaper/replication/orswhite.pdf>.

Ozkarahan E. 1986. Database machines and database management. Prentice-Hall.

Powell A., Vickers A., Williams E., Cooke B. 1996. A practical strategy for the evaluation of software tooles. Teoksessa Method Engineering, Working Conf. on Method Engineering, 165-185.

Praxis International Inc. 1996a. Data replication, make or buy? [viitattu 27.10.1996]. Saatavilla html-muodossa: <URL: http://www.praxisint.com/wpmvb.html>.

Praxis International Inc. 1996b. Selecting a data replicator [viitattu 27.10.1996]. Saatavilla html-muodossa: <URL: http://www.praxisint.com/wpsdr.html>.

Praxis International Inc. 1996c. The economics of data replication [viitattu 4.11.1996]. Saatavilla html-muodossa: <URL: http://www.praxisint.com/wpedr.html>.

Radosevich L. 1995. Replication mania [viitattu 8.2.2003]. Saatavilla html-muodossa: <URL: http://www.computerworld.com/news/1995/story/0,11280,16832,00.html>.

Ranagathan K., Foster I. 2002. Identifying dynamic replication strategies for a high-performance data grid [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://www.globus.org/research/papers/repstrat02.pdf>.

Ratner D., Reiher P., Popek G. J., Kuenning G. H. 1997. Replication requirements in mobile environments [viitattu 8.2.2003]. Saatavilla pakattuna ps-muodossa: <URL: http://fmg-www.cs.ucla.edu/ficus-embers/ratner/papers/dialm.ps.gz>.

Reddy P. K., Kitsuregawa M. 1998. Improving performance in distributed data base systems using speculative transaction processing [viitattu 8.2.2003]. Saatavilla ps-muodossa: <URL: http://www.tkl.iis.u-tokyo.ac.jp/Kilab/Research/Paper/1998/reddy/europds98.ps>.

Richman D. 1996. Replication heats up [viitattu 8.2.2003]. Saatavilla html-muodossa: <URL: http://www.computerworld.com/news/1996/story/0,11280,7129,00.html>.

Rowley J. 1993. Selection and evaluation of software. Aslib Proceedings, Vol. 45, No. 3, 77-81.

Saito Y., Levy M. H. 2001. Optimistic replication for internet data services [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://www.hpl.hp.com/personal/Yasushi_Saito/disc00.pdf>.

Schussel G. 1996. Replication, the next generation of distributed database technology [viitattu 8.11.1996]. Saatavilla html-muodossa: <URL: http://www.dciexpo.com/geos/replica.htm>.

Semich W. J. 1994. Where do C/S apps go wrong. Datamation, January 21, 30-36.

Simon E. 1996. Distributed information systems: from client/server to distributed multimedia. McGraw-Hill, London.

Stacey D. 1995. Replication: DB2, Oracle, or Sybase?. ACM SIGMOD Record, Vol. 24, No. 4, 95-101.

Stepansky R. 1996. Data replication: An RX for enterprise data management, part 2 [viitattu 24.10.1996]. Saatavilla html-muodossa: <URL: http://www.candle.com/about/pubs/8.96/ruby2-4.htm>.

Stonebraker M., Woodfill J., Randstrom J., Kalash J., Arnold K., Andersen E. 1983. Performance analysis of distributed data base systems [viitattu 8.2.2003]. Saatavilla pdf-muodossa:
<URL: http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/CSD-83-144.pdf>

Sybase Finland Oy. 1996. Enso uusii myyntijärjestelmänsä. Sybase Finland Oy:n asiakaslehti 1/96, 6-7.

Sybase Inc. 2002a. ASE replicator user's guide [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://download.sybase.com/pdfdocs/asg1250e/rl125ug.pdf>.

Sybase Inc. 2002b. Replication server 12.5 administration guide volume 1 [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://download.sybase.com/pdfdocs/rsg1250e/rsadmin.pdf>.

Sybase Inc. 2002c. Replication server 12.5 administration guide volume 2 [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://download.sybase.com/pdfdocs/rsg1250e/rsadm2.pdf>.

Sybase Inc. 2002d. Replication server 12.5 configuration guide. Sybase Inc:n ohjekirja, 35818-01-1250-01.

Sybase Inc. 2002e. Replication server 12.5 design guide [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://download.sybase.com/pdfdocs/rsg1250e/design.pdf>.

Sybase Inc. 2002f. Replication server 12.5 installation guide. Sybase Inc:n ohjekirja, 32236-01-1250-01.

Sybase Inc. 2002g. Replication Server An architecture for distributing and sharing corporate information [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://www.sybase.com/content/1019249/4427RepserverArch_3.pdf>.

Theel O., Pagnia H. 1998. Optimal replica control protocols exhibit symmetric operation availabilities [viitattu 28.1.2003]. Saatavilla pakattuna ps-muodossa: <URL: http://www.informatik.du-darmstadt.de/BS/Pagnia/Publications/ftcs-28.ps.gz>.

UniF/X Inc. 2000. Current Issues in Data Replication [viitattu 8.2.2003]. Saatavilla html-muodossa: <URL: http://www.unifx.com/article.html>.

Vuoniemi T. 1990. Liittolaistietokantajärjestelmät. Jyväskylän yliopisto. Tietojärjestelmätieteen pro gradu -tutkielma.

Watterson K. 1996. Database replication explained. Datamation, September, 62-66.

Wiesmann M., Pedone F., Schiper A. 2000a. Database replication techniques: a three parameter classification [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://lsewww.epfl.ch/Documents/acrobat/WPS+00b.pdf>.

Wiesmann M., Pedone F., Schiper A. 2000b. Understanding replication in databases and distributed systems [viitattu 8.2.2003]. Saatavilla pdf-muodossa: <URL: http://lsewww.epfl.ch/Documents/acrobat/WPS+00.pdf>.

Yawin D. 1994. Optimizing Notes replication. Byte, September, 201-202.

Özsu M. T., Valduriez P. 1991. Principles of distributed database systems. Prentice-Hall, New Jersey.

Özsu M. T., Valduriez P. 1994a. Distributed database systems slides: distributed transaction processing [viitattu 8.2.2003]. Saatavilla html-muodossa: <URL: http://www.cs.ualberta.ca/~ozsu/ddbs.html>.

Özsu M. T., Valduriez P. 1994b. Distributed data management: unsolved problems and new issues? [viitattu 8.2.2003]. Saatavilla ps-muodossa: <URL: http://db.uwaterloo.ca/~ddbms/publications/ozsu/distdb/UnsolvedIssues.ps>.

Özsu M. T., Valduriez P. 1996a. Distributed and parallel database systems [viitattu 5.9.1996]. Saatavilla html-muodossa: <URL: http://web.cs.ualberta.ca/~ozsu/ publications.html>.

Özsu M. T., Valduriez P. 1996b. Distributed and parallel database systems - technology and current state-of-the-art [viitattu 5.9.1996]. Saatavilla html-muodossa: <URL: http://web.cs.ualberta.ca/~ozsu/publications.html>.

Özsu M. T., Valduriez P. 1999. Principles of distributed database systems. 2nd edition, Prentice-Hall, New Jersey.

LIITE 1. Kokeilun tietokoneiden laitteisto- ja ohjelmistokokoonpanot.

KAJ641-, KAJ6450- ja KAJ6516 –palvelinten laitteistokokoonpanot olivat:

- HP Vectra VL VLI8.PII-400
- prosessori: Intel Pentium 400 Mhz
- keskusmuisti: 128 MB
- kovalevy: 8 GB
- näytönohjain: Matrox MGA-G200, 8 MB
- verkkokortti: 3Com Fast EtherLink XL 3C905B-TX

Tehtaan lähiverkko oli tyypiltään 100 Mbit/s Ethernet.

Palvelinten ohjelmistokokoonpanot ja asetukset olivat:

- käyttöjärjestelmä: Microsoft Windows NT 4.0 (Build 1381: Service Pack 6)
- verkkoprotokolla: TCP/IP
- tiedostojärjestelmä: NTFS
- muut ohjelmat: kokeilun tietokannan hallintajärjestelmien (Sybase ja Oracle) lisäksi palvelimissa ei ollut muita ohjelmia, joilla olisi ollut vaikutusta suorituskykyyn (esim. virustarkistusohjelmat)

LIITE 2. Koeajojen tietokantojen taulumääritykset.

Koeajojen TESTI-, TESTI2- ja TESTI3-tietokannoissa oli kolme tietokantataulua (LABORATORIO, PAINEHIOMO ja PK3). Tässä on esitetty esimerkinomaisesti SQL-komennot, joilla taulut luotiin Sybasen TESTI-tietokantaan.

```
CREATE TABLE TESTI.dbo.LABORATORIO (
        VVVVKKPPHHMMSS      varchar(14)     not null,
        LAHDE               varchar(14)     not null,
        LBMITTAUS1          float(16)   not null,
        LBMITTAUS2          float(16)   not null,
        LBMITTAUS3          float(16)   not null,
        LBMITTAUS4          float(16)   not null,
        LBMITTAUS5          float(16)   not null,
        LBMITTAUS6          float(16)   not null,
        LBMITTAUS7          float(16)   not null,
        LBMITTAUS8          float(16)   not null,
        LBMITTAUS9          float(16)   not null,
        LBMITTAUS10         float(16)   not null,
            CONSTRAINT LABORATORIO_320001141 PRIMARY KEY CLUSTERED
            (VVVVKKPPHHMMSS, LAHDE )
)
```

```
CREATE TABLE TESTI.dbo.PAINEHIOMO (
        VVVVKKPPHHMMSS      varchar(14)      not null,
        LAHDE               varchar(14)      not null,
        PHMITTAUS1          float(16)   not null,
        PHMITTAUS2          float(16)   not null,
        PHMITTAUS3          float(16)   not null,
        PHMITTAUS4          float(16)   not null,
        PHMITTAUS5          float(16)   not null,
        PHMITTAUS6          float(16)   not null,
        PHMITTAUS7          float(16)   not null,
        PHMITTAUS8          float(16)   not null,
        PHMITTAUS9          float(16)   not null,
        PHMITTAUS10         float(16)   not null,
            CONSTRAINT PAINEHIOMO_320001141 PRIMARY KEY CLUSTERED
            (VVVVKKPPHHMMSS, LAHDE )
)

CREATE TABLE TESTI.dbo.PK3 (
        VVVVKKPPHHMMSS      varchar(14)      not null,
        LAHDE               varchar(14)      not null,
        PKMITTAUS1          float(16)   not null,
        PKMITTAUS2          float(16)   not null,
        PKMITTAUS3          float(16)   not null,
        PKMITTAUS4          float(16)   not null,
        PKMITTAUS5          float(16)   not null,
        PKMITTAUS6          float(16)   not null,
        PKMITTAUS7          float(16)   not null,
        PKMITTAUS8          float(16)   not null,
        PKMITTAUS9          float(16)   not null,
        PKMITTAUS10         float(16)   not null,
            CONSTRAINT PK_320001141 PRIMARY KEY CLUSTERED
            (VVVVKKPPHHMMSS, LAHDE )
)
```