

IIO30200 Tietokannan hallinta

Tapahtumien hallinta

Jouni Huotari

(19.3.2008)



JYVÄSKYLÄN
AMMATTIKORKEAKOULU

Osaaminen kilpailukyvyksi

Tehtäviä/kysymyksiä

- Määrittele
 - tapahtuma (transaction)
 - tapahtumien hallinta
- Mitä ovat tapahtuman ACID-ominaisuudet?
- Mitä ovat seuraavat lukot:
 - Shared lock
 - Exclusive lock
 - Deadlock
- Mikä ero on optimistisella ja pessimistisellä lukituksella?
- Miksi tapahtumien hallinta on tärkeää?
- Mitä ongelmia voi esiintyä jos useita tapahtumia suoritetaan rinnakkain?
- Miten ongelmat voitaisiin ratkaista?

Menetelmä

- Tee määritykset ensin yksin
- Vertaa määrityksiäsi vierustoverisi kanssa
- Miettikää yhdessä (pienryhmissä) vastaukset miksi-, mitä- ja miten-kysymyksiin (Case: pankki tai lennonvarausjärjestelmä)
- Kootaan vastaukset yhteisesti

Määritelmät

- Tapahtuma (transaction) on käyttäjän antama komento tai ohjelma (tai sen osa), joka hakee tai muuttaa tietoa tietokannasta
- Tapahtumien hallinta (tai käsittely) tarkoittaa relaatiotietokannan kykyä hallita tapahtumia
 - jos jokin menee vikaan tapahtuman aikana, ohjelman käskyt perutetaan (rollback); muutoin ohjelman käskyt suoritetaan (commit)
 - esimerkiksi pankkitapahtuman käsittely: pitää tarkistaa sekä maksutili että saajatili

Tyypillisiä ongelmia

- Menetetyn päivityksen ongelma (lost update problem): tapahtuman suorittama päivitys jää toteutumatta, koska toinen tapahtuma korvaa sen (ts. kirjoittaa päälle)
- Tilapäisen päivityksen ongelma (temporary update problem, uncommitted dependency problem, "dirty read"): tapahtuman päivitys perustuu tietoon, joka muuttuu toisen tapahtuman keskeytyksen seurauksena
- Virheellisen yhteenvedon ongelma (incorrect summary problem): tapahtuma hakee tietoja useasta paikasta samalla kun toinen tapahtuma päivittää tietoja

Esimerkkejä ongelmista

- Menetetyn päivityksen ongelma

Time	T ₁	T ₂	bal _x
t ₁		begin_transaction	100
t ₂	begin_transaction	read(bal _x)	100
t ₃	read(bal _x)	bal _x = bal _x + 100	100
t ₄	bal _x = bal _x - 10	write(bal _x)	200
t ₅	write(bal _x)	commit	90
t ₆	commit		90

- Tilapäisen päivityksen ongelma

Time	T ₃	T ₄	bal _x
t ₁		begin_transaction	100
t ₂		read(bal _x)	100
t ₃		bal _x = bal _x + 100	100
t ₄	begin_transaction	write(bal _x)	200
t ₅	read(bal _x)	:	200
t ₆	bal _x = bal _x - 10	rollback	100
t ₇	write(bal _x)		190
t ₈	commit		190

AMMATTI KIRJASTO

Tapahtumilla pitää olla ACID-ominaisuudet

- **Atomicity:** tapahtuma suoritetaan joko kokonaisuudessaan tai ei lainkaan
- **Consistency:** tapahtuma siirtää tietokannan eheästä tilasta toiseen eli tapahtuman on säilytettävä kannan eheys
- **Isolation:** tapahtuman kantaan tekemät muutokset eivät saa näkyä muille tapahtumille ennen kuin se on lopettanut suorituksensa menestyksellisesti (committed; tapahtuma on eristetty muista tapahtumista)
- **Durability:** sitoutuneen tapahtuman kantaan tekemät muutokset ovat pysyviä, eivätkä ne saa missään tapauksessa kadota

Lähteet ja linkit

- Thomas Connolly & Begg, C. (2002), Database Systems – a Practical Approach to Design, Implementation, and Management, 3. painos, Addison-Wesley
- Mauri Leppänen (1999), Tietokannan hallintajärjestelmät, luentomoniste, Jyväskylän yliopisto
- Ryan K. Stephens & al. (2001), SQL Trainer Kit, Edita
- WWW, etsi esim. hakusanalla "Transaction processing" tai "Transaction management" + liitä tarkennus, esim. "lost update problem"
- Jaakko Rantanen (2006), Samanaikaisuuden hallinta ohjelmoinnin näkökulmasta:
<http://data.hamk.fi/materiaalit/tiedonhallinta2/ccr/eristyvyys.pps>

Esimerkki lukituksen käsittelystä

- Case: Microsoft Access, joka käyttää tiedostopohjaista lukitusta (.ldb)
- Avaa yhteisellä levyllä oleva mdb-tiedosto, esim. <\\ghost\huojo\mdb\ASIAKAS.MDB>
- Katso, mitä ko. hakemistoon ilmestyy
- Tutki ldb-tiedoston sisältö
- Tee muutos samaan tauluun kaverin kanssa, esim. FIRMA-tiluun
- Mitä tapahtuu, kun tallennat rivin?