Kotitehtävät vko 6
Tarkistusajot. Lukot, Deadlock:

a) Mitä tarkoittaa defragmentoituminen tietokannoissa?

Vastaus:
Tietokannan Database Engine pitää automaattisesti yllä indeksejä aina kun suoritetaan insert, update tai delete operaatio tietokannan sisältämään dataan. Ajan myötä nämä muutokset voivat aiheuttaa indeksien hajaantumisen eli fragmentoitumisen. Fragentoitumisessa indeksin sisältämä tieto hajaantuu eri puolelle indeksiä. Haut esimerkiksi eivät ole enää tehokkaita, kun tieto ei enää ole indeksissä lähellä toisiaan siten kuin sen tulisi olla tehokkuuden kannalta. Ks. http://msdn.microsoft.com/en-US/library/ms189858(v=sql.105).aspx. Defragmentoinnissa poitetaan tällainen indeksien ja taulujen hajaantuminen. Fragmentoitumisen poistamista käsiteltiin edellisen viikon tuntitehtävissä alter index lauseen yhteydessä.

b) Mitä deadlock:lla tarkoitetaan tietokannoissa? Millaisessa tilanteessa voidaan joutua tällaiseen tilanteeseen? Mitä vaikutuksia sillä on tietokantapalvelimeen? Miten tietokantapalvelin pyrkii selviämään deadlock tilanteesta? Anna käytännön esimerkki deadlock tilanteesta T-SQL lauseina.

Vastaus:
Deadlock tilanteella tarkoitetaan tilannetta, jossa kaksi eri transaktiota lukitsee pysyvästi toistensa tarvitsemia resursseja. Esimerkiksi suoritetaan kaksi erillistä transaktiota täsmälleen yhtä aikaa. Ensimmäisessä transaktiossa A halutaan suorittaa 50 euron tilisiirto asiakkaan 1 tililtä puhelin operaattorin tilille. Toisessa transaktiossa B halutaan palauttaa samasta puhelin operaattorin tililtä 10 euroa ylimaksuja samaiselle asiakkaalle 1. Jos kaikki etenisi alla olevalla tavalla:

- Transaktio A pyytää tietokannalta update lukon riville 111222, joka on asiakkaan 1 tili rivi tietokannassa.
  update tilit set saldo = saldo – 50 where asiakasid = 111222;
- Transaktio B pyytää tietokannalta exlusive lukon riville 999888, joka on puhelin operaattorin tili rivi tietokannassa.
  update tilit saldo = saldo -10 where asiakasid = 999888;
- Transaktio A pyytää update lukon riville 999888 (puhelin operaattorin tili).
  update tilit saldo = saldo + 50 where asiakasid = 999888;
  Transaktio A ei saa pyytämäänsä lukkoa, sillä se on jo varattu transaktio B:lle. Se saa sen vasta kun se vapautuu transaktio B:ltä.
- Transaktio B pyytää exclusive lukkoa riville 111222 (asiakas 1).
  update tilit saldo = saldo + 10 where asiakasid = 111222;
  Transaktio B ei saa pyytämäänsä lukkoa, sillä se on varattu transaktio A:lle. Se saa sen vasta kun se vapautuu transaktio A:ltä. Eli on päädytty täyteen lukko tilanteeseen, deadlock:iin.

Yleensä tietokannat ratkaisevat deadlock tilanteen siten, että jompikumpi transaktioista peruutetaan (rollback). Peruutettavaksi transaktioksi valikoituu se, josta on ehditty suorittaa vähiten tai vaihtoehtoisesti jäljellä on vielä niin vähän, että kyseinen transaktio kannattaa suorittaa loppuun asti. Lisäksi on huolehdittava, ettei käy niin, että jotain transaktiota ei koskaan saada suoritettua loppuun. Siis jos jokin transaktioista on valikoitunut database engine:n toimesta väistäväksi transaktioksi kyllin monta kertaa, ajetaan se silloin loppuun, ja muut transaktiot perutetaan. Ks. http://msdn.microsoft.com/en-us/library/ms177433(v=sql.105).aspx

c) Mitä merkitystä lukoilla on tietokannoissa? Millaisia lukkoja käytetään nykyään? Etsi asiasta tietoa Oracle:n, MySQL:n ja SQL Server:in osalta.

Vastaus:
SQL Server:in lukot on esitelty osoitteessa: http://msdn.microsoft.com/en-US/library/ms175519(v=sql.105).aspx:

- Shared lock
- Update lock
- Exclusive lock
- Intent lock
- Schema lock
- Bulk update lock
- Key-range lock

Oracle Database:n käyttämät lukot on esitelty osoitteessa: http://docs.oracle.com/cd/B19306_01/server.102/b14220/consist.htm#i5242.

MySQL:n lukoista on puolestaan kerrottu osoitteessa: http://dev.mysql.com/doc/refman/5.1/en/innodb-transaction-model.html.

d) Mitä tarkoittaa tietokannoissa isolation level eli eristys tasot? Mikä on oletus isolation level SQL Server:ssä? Millä komennolla voisit muuttaa sitä?

Vastaus:
Edellisessä pankki esimerkissä transaktio A ja transaktio B lukkiutuivat kun ne tarvitsivat ristikkäin toistensa varaamia resursseja. Transaktio A oli lukinnut rivin, jota transaktio B pyysi database engine:ltä. Näiden lukkojen toimintaa voi muuttaa. Eristys tason voi esimerkiksi muuttaa SQL Server:ssä Read Uncommited eristys tasolle, jolloin eri transaktioita ei käytännössä enää eristetä toisistaan. Tällöin esimerkkimme transaktio A ja transaktio B voisivat edetä loppuun asti ilman rivien lukitsemista. Tietokanta toimii nyt nopeasti, mutta tästä aiheutuu vakava ongelma. Mitä jos esimerkiksi pankin kontrolleri olisi tarkastelemassa vaikkapa puhelin operaattorin ja asiakas 1:n tilejä juuri samaan aikaan kun transaktio B olisi ehtinyt suorittaa vain ensimmäisen vaiheensa eli vähentää 10 euroa puhelin operaattorin tililtä. Asiakkaan tilille ei vastaavaa summaa olisi ehditty lisätä. Tällaisessa tapauksessa kontrolleri virheellisesti näkisi 10 euron vajauksen pankin tileissä. Sen vuoksi lukkoja ei yleensä haluta täysin vapauttaa, vaan eri transaktiot halutaan eristää toisistaan siten, että keskeneräisiä transaktioita eivät muut transaktiot näe. Käytännössä SQL Server:ssä käytetään Read Committed tason eristystä transaktioiden välillä.

SQL Server:ssä oletus isolation level on Read Commited. Sen voi muuttaa lauseella set transaction isolation level. Ks. http://msdn.microsoft.com/en-US/library/ms189122(v=sql.105).aspx, http://msdn.microsoft.com/en-US/library/ms173763(v=sql.105).aspx.