DIS - Transaktioner 1 - ACID, Locks, timestamp

ACID

Atomicity

Draconian error handling bruges meget i kodning – hvis noget fejler, så går det hele ned.

Hvordan transaktioner opfører sig når man læser og skriver i to forskellige

Enten virker hele molevitten eller også bliver det droppet.

Consistency

Sørger for at data er korrekte, fx at en ny række har en primærnøgle.

Isolation

Det betyder at en database skal have samme resultatet, hvis man kørte to trasnaktioenr på samme tid som de gjorde hvis vi gjorde det efter hinanden.

Sætter lighedstegn mellem samtidig og seriel kørsel af transaktioner.

Durability

Fast recovery – hvis en database går ned, så skal transkationer kunne overleve.

Hvis nu serveren eksploderer, så skal transaktioner kunne overleve

- typisk ved at være gemt på disken

Hvis man forsøger at lave en transkationer i en database der er **ACID-compliant**, så SKAL det virke det hele ellers så skal det slet ikke virke.

noSQL er ikke **ACID-complaint** – den er helt anderledes

Seriel ækvivalens

En måde at tilsikre, når man implementere transaktioner, at to transkationer SKAL give samme resultat

We say that an interleaving of two blocks is serially equivalent, if the result is equivalent to an execution in which one block was executed entirely before the other.

En-efter-en versus samtidig afvikling.

Hvis resultatet af den ene og efterfulgt af resultatet af den næste så har man en seriel ækvivalens.

Man kan sikre seriel ækvivalens ved fx at låse værdier eller ved at lave **wait-for-grafer (BILLEDE)**

Det faktum at man kører forskellgie transkationer, og de giver samem resultatet – hvordan kan man sikre det..

Låse – locks og locking #

Locks & Locking

Billede

Når man taler om en lås. Helt overovrnet set – man sætter en lås på noget data – man låser noget så nogen kan gøre noget med dataen, mens andre ikke kan. Hvis jeg skal ind på noget data, så låser jeg den, så andrei kke kan komme ind på den.

Sletter man noget slå skriver man også til den.

Deadlocking

Billede

2-Phased Locking

Udfordringer

Unnecessary or early lock and therefore wait

Deadlock

Cascading rollback

Growing and shrinking phases

- Hvis en værdi ikke allerede er låst, så lås den.
- Hvis værdien har en lås i forvejen, der er i konflikt med operationen, så vent til låsen fjernes.
- Hvis værdien har en lås i forvejen, der ikke er i konflikt, så fortsæt som planlagt.
- Hvis værdien allerede er låst i samme transaktion, så forfrem den eventuelt og fortsæt som planlagt

Locking er en form for **concurrency control**.

Optimistic concurrency control

- Optimistic concurrency control er en antagelse om at flere forskellige transaktioner kan gennemføres uden at påvirke hinanden.
- Når en transaktion bliver kørt, så foregår det uden brug af locking.
- Når resultatet af en transaktion bliver gemt, så undersøges hvorvidt de data, den har brug, er blevet ændret i mellemtiden.
- Hvis ja, så forsøges transaktionen gentaget på ny.

Det kan skabe bedre performance at være optimistisk.

Locking bruges meget i distribueret systemer, men det gør dem ikke hurtige.
Man undersøger om værdierne man ændre har ændret sig i mellemtiden, inden man gemmer

Timestamp ordering

- Timestamp ordering har som regel at en transaktion ikke må gemme en værdi, der er blevet læst på et senere tidspunkt af en anden transaktion.
- En transaktion må ikke gemme en værdi, der er blevet gemt på et senere tidspunkt af en anden transaktion.
- En transaktion må ikke læse en værdi, der er blevet gemt på et senere tidspunkt af en anden transaktion.

Altså: adgang til værdier styres af tidsstempler.

Et tidsstempel kan være et tidspunkt eller blot en værdi som stiger.

The lost update problem

Når der eksempelvis bliver læst fra to computere på samme tid, og den ene opdatere værdien, mens den anden stadig har den oprindelige værdi loadet. Så når den anden opdatere, så er der ikke registreret to opdateringer.

Transaktioner

En måde at behandle data på **sekventielt** så man har noget kontrol over hvad der sker med dataen

En transaktion er en behandling af informationer, der enten lykkes eller ikke lykkes.

Transaktioner er en måde at skabe et forløb på, så behandlinger af data ikke foregår kaotisk.

Transactions all limit to some extent the potential for concurrent operation. I det øjeblik man har transkationer at noget er rigtigt og det er konsistent osv – og når man gør det, så skal det gennem en masse regler, som gør det langsomt. Gør man ikke i i NOSQL

Transaktioner er distribuerede når de anvender værdier fra flere forskellige steder. Fx når en transaktion læser værdier fra et cluster af database-

servere.