NTNU
Kunnskap for en bedre verden

Øving 5: Normalisering og Transaksjoner (obligatorisk)

NTNU
Innleveringsfrist: se Blackboard
Tidligste godkjenning: datoer blir annonsert

Løsningsforslag legges ut i etterkant.

Alle obligatoriske øvinger må være godkjente for å få karakter i emnet.

DEL 1: Normalisering

Et firma som leier ut feriehus lagrer følgende informasjon knytet til utleie:

kundedata: navn, adresse, telefon
data om eiendommen: adresse
data om eieren: navn, adresse, telefon
data om utleieforholdet: fra og med uke, til og med uke, pris pr uke

Gitt at vi har én tabell med 13 attributter (kolonner) som ser altså slik ut (skrevet på relasjonell form uten nøkler):

```
leieforhold (kunde_id, kunde_navn, kunde_adresse, kunde_tlf, eiendom_id,
eiendom_adresse, eier_id, eier_navn, eier_adresse, eier_tlf, fra_uke,
til_uke, pris)
```

I tillegg til dataene nevnt innledningsvis er det lagt inn kunde_id, eier_id og en eiendoms_id, som entydig identifiserer henholdsvis en person og en eiendom.

Foreslå kandidatnøkler for denne tabellen. Anta at en person kun kan leie en eiendom av gangen, og at en eiendom kan leies ut til kun en person av gangen.

Tabellen er ikke problemfri mht til registrering og sletting av data. Forklar hvorfor.

Tegn ett (kun ett) diagram (tilsvarende figur som i læreboka kap. 3) som viser funksjonelle avhengigheter mellom alle attributtene.

Du skal nå bruke funksjonelle avhengigheter og BCNF til å foreslå en oppsplitting av tabellen i mindre tabeller slik at problemene vedr. registrering og sletting av data unngås.

Sett opp, direkte fra figuren, relasjoner som er på BCNF.

Kan vi løse denne oppgaven ved å gjennomføre prosessen 1NF --> 2NF --> 3NF? Begrunn svaret.

DEL 2: Transaksjoner

I denne øvingen forventes det at man har gjennomført det som omhandler transaksjoner (resten er beskrevet som frivillig). Øvingslærerne kan sjekke/stille kontrollspørsmål om transaksjoner ved levering av øving 5.

Gjennomføring:

Denne øvingen er i utgangspunktet tenkt utført i grupper av to personer, om man ikke har noen å jobbe sammen med kan man bruke to konsoller (mot databasen) på samme maskin.

Velg en brukers MySQL-konto som dere kobler opp mot.

Opprett innhold i database (evt. slett gammelt - delete from konto).

> Kjør konto.sql eller create table skript fra foiler.

Frivillig - Manuell bruk av skrive- og leselåser

For de som har lyst å prøve å sette lese- og skrivelåser manuelt. Her vil det settes låser på hele tabeller. Merk at man normalt heller (ihvertfall når man programmerer) vil bruke transaksjoner med ønsket isolering (se neste «kapittel»).

Her må man være to stykker (eller bruke to konsoller på en maskin). Kjør manuell skrive- og leselåser. For syntaks på manuell låsing se: https:/dev.mysql.com/doc/refman/5.7/en/lock-tables.html

Hvis en klient blir hengende i denne oppgaven, trykk ctrl-c for å kunne skrive nye kommandoer. Merk også at manuell låsing (som på lenken over) låser hele tabellen!

Start med at en person sørger for å sette leselås på konto-tabellen.

- Prøv å kjøre «SELECT * from konto;» i begge klientene. Hva skjer?
- Er det mulig å sette leselås også i den siste klienten?
- Fjern leselåsen i den ene klienten og kjør følgende i begge:
- «UPDATE konto SET saldo=0 WHERE kontonr=1;» ?
    - Forklar resultatet.

Fjern låsene!

Start med at en person/klient sørger for å sett skrivelås på konto-tabellen.

- Prøv å kjøre «SELECT * from konto;» i begge klientene. Hva skjer?
- Lås opp tabellen og sett skrivelås på nytt.
- Er det mulig å sette leselås i den siste klienten?
    - hvis ikke (dvs. at den henger) avbryt med ctrl-c for å kunne skrive nye kommandoer
- Er det mulig å kjøre «UPDATE konto SET saldo=0 WHERE kontonr=1;» ?
    - Sjekk i begge klienter og forklar svaret.

Transaksjoner

Bruk MySQL-dokumentasjonen til å finne ut hvordan man starter og avslutter en transaksjon. For å hjelpe litt på vei så se på denne siden: https://dev.mysql.com/doc/refman/5.7/en/commit.html

Sjekk innhold på kontoer, start en transaksjon og oppdater en konto. Sjekk at commit og rollback gir ønskede resultater (her må du starte transaksjonen to ganger, en gang avslutt med commit og en gang med rollback).

Teori

Hvilke typer låser har databasesystemene?

Hva er grunnen til at at man gjerne ønsker lavere isolasjonsnivå enn SERIALIZABLE?

Hva skjer om to pågående transaksjoner med isolasjonsnivå serializable prøver select sum(saldo) from konto?

Hva er to-fase-låsing?

Hvilke typer samtidighetsproblemer (de har egne navn) kan man få ved ulike isolasjonsnivåer? Hva er optimistisk låsing/utførelse? Hva kan grunnen til å bruke dette være?

Hvorfor kan det være dumt med lange transaksjoner (som tar lang tid)? Vil det være lurt å ha en transaksjon hvor det kreves input fra bruker?

Oppgaver om transaksjoner

Når kommando for Klient 1 og Klient 2 er satt på samme linje så spiller det ingen rolle i hvilken rekkefølge de utføres.

Oppgave 1

Tid	Klient 1	Klient 2
1	Sett isolasjonsnivå til **read uncommitted**.	Sett isolasjonsnivå til **serializable**.
2	Start transaksjon.	Start transaksjon.
3		select * from konto where kontonr=1;
4	select * from konto where kontonr=1;	
5	update konto set saldo=1 where kontonr=1;	
6		commit;
7	commit;	

Hva skjer og hvorfor? Hva hadde skjedd om Klient 2 hadde brukt et annet isolasjonsnivå?

Svar:

Oppgave 2

a)

Tid	Klient 1	Klient 2
1	Sett isolasjonsnivå til **read uncommitted**.	Sett isolasjonsnivå til **read uncommitted**.
2	Start transaksjon	Start transaksjon
3	update konto set saldo=1 where kontonr=1;	
4		update konto set saldo=2 where kontonr=1;
5	update konto set saldo=1 where kontonr=2;	
6	**commit;**	update konto set saldo=2 where kontonr=2;
7		**commit;**

Hva blir resultatet? Hvorfor må det være slik?

Svar:

b)

Her vil Klient 2 utføre sine update-setninger i motsatt rekkefølge av over!

Tid	Klient 1	Klient 2
1	Sett isolasjonsnivå til **read uncommitted**.	Sett isolasjonsnivå til **read uncommitted**.

2	Start transaksjon	Start transaksjon
3	update konto set saldo=1 where kontonr=1;	
4		update konto set saldo=2 where kontonr=2;
5	update konto set saldo=1 where kontonr=2;	
6		update konto set saldo=2 where kontonr=1;
7		

Hva blir resultatet? Forklar forskjellen fra oppgave a).

Vil det ha noe å si om man endrer isolasjonsnivå på klientene?

Svar:

Oppgave 3

Tid	Klient 1	Klient 2
1	Sett isolasjonsnivå til **read uncommitted**.	Sett isolasjonsnivå til **serializable**.
2	Start transaksjon.	Start transaksjon.
3	select sum(saldo) from konto;	
4		update konto set saldo=saldo + 10 where kontonr=1;
5	select sum(saldo) from konto;	
6		commit;
7	select sum(saldo) from konto;	
8	commit;	

Hva skjer?

Svar:

Hva vil skje om Klient 1 bruker read committed, repeatable read eller serializable?

Svar:

Oppgave 4

Lag en kjøring med to klienter som tester phantom reads. Her kan det være lurt å tenke igjennom isolasjonsnivå. Om resultatet ikke er som forventet så kan det være lurt å sjekke dokumentasjonen.

Svar: