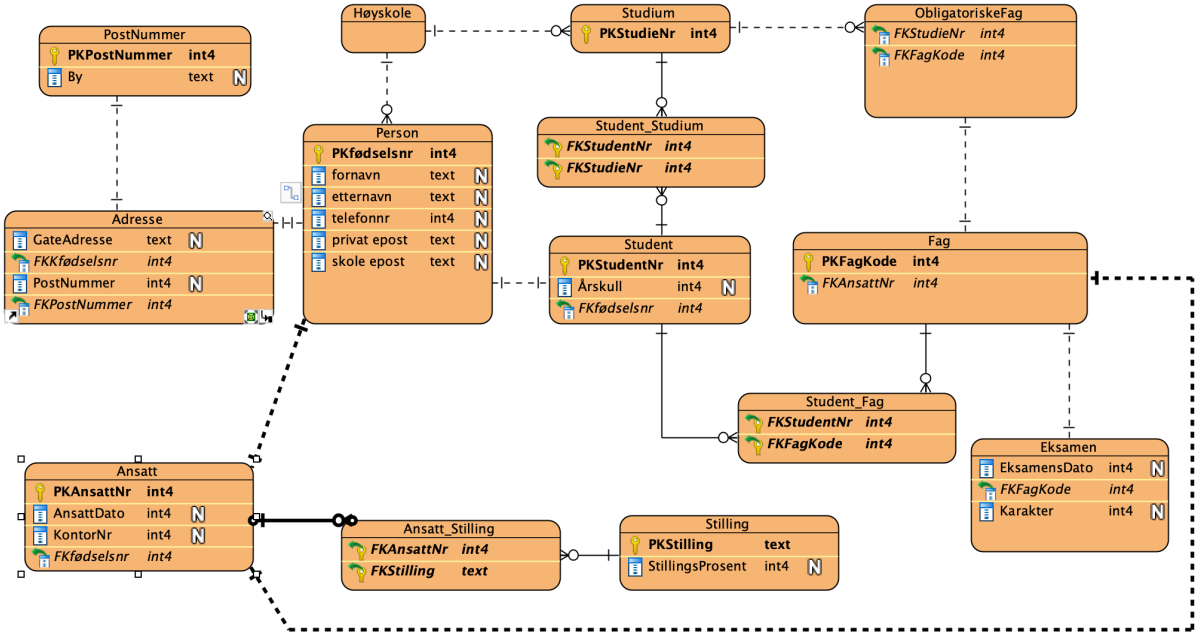
DAT107
Oblig 2
587728 Joachim Leiros

Oppgave 1:



Ettersom en person kan være både student og ansatt har jeg valgt å dele opp diss entitetene og deres idenfifikatorer. En person registreres med fødselsnummer og vil være unikt for hver og en. Derfor kan vi knytte både ansatte og studenter til fødselsnummer. Vi trenger ikke knytte opp ansatt mot student da det vil være duplikasjon av data ettersom vi allerede har personen I personentiteten.

Oppgave 2

Databasen over er 3NF da databasen:

Tilfreddstiller krav om at verdier er atomære. Diverse vi har delt opp fornavn og etternavn, og delt opp adresse til å inneholde både postnummer, by og gateadresse. Databasen tilfredstiller krav til å være 1NF.

Tilfredstiller krav om at det ikke finnes ikke-nøkkelattributter som er avhengige en del av en kandidatnøkkel. Eksempelvis ser vi i diagrammet at vi har flyttet postnummer og by ut til en egen tabell. Dette er fordi en by er avhengig av postnummer og vi da kun trenger referere til postnummer i adressetabellen for å vite by. Da slipper vi overflødig data. Databasen tilfredstiller derfor kravene for å være 2NF.

Da databasen tilfredstiller krav for å være 1NF, 2NF og i tillegg er ingen av attributtene til entitetene er transitivt avhengig av noe annet enn primærnøkkelen. Eksempelvis kan vi se dette I samme eksempel som nevnt over, at en by har ett postnummer og vi bruker postnummer som referanse for å finne hvilken by en person bor i. På lik linje har en ansatt en stilling som innehar en stillingsprosent. Ettersom flere ansatte kan ha samme stilling flytter vi stillingen ut I en egen tabell og referer til Ansatt nummeret.

Det er i dette tilfellet hensiktsmessig da vi ikke arbeider med utrolig store tabeller og har derfor ikke behov for å denormalisere dataen for ytelse. I motsetning er det heller hensiktsmessig at vi lett kan knytte de respektivefeltene til personer.

Dersom vi skulle denormalisert noe kunne vi denormalisert Studium til å inkludere obligatoriske fag.

Oppgave 3

Med utgangspunkt i spørringen:

```
SELECT

Poststed.postnr,

Poststed.poststed,

Ordrelinje.prisprenhet,

Ordrelinje.antall

FROM ((Poststed INNER JOIN Kunde ON Poststed.postnr = kunde.postnr)

INNER JOIN Ordre ON Kunde.Knr = Ordre.Knr)

INNER JOIN Ordrelinje ON Ordrelinje.Ordrenr = Ordre.Ordrenr

WHERE Ordre.Ordrenr IS NOT NULL
```

Har jeg kommet usikkert fram til:

$$\Pi_{poststed.postnr,poststed.poststed,ordrelinje.prisprenhet,ordrelinje.antall}$$
$$(\sigma\ Kunde.Knr = Ordre.Knr$$
$$(Poststed \bowtie_{Poststed.postnr = Kunde.postnr} (Kunde)$$
$$(Kunde \bowtie_{Kunde.Knr = Ordre.Knr} (Ordre)$$
$$Ordre \bowtie_{Ordre.OrdreNr = Ordrelinje.OrdreNr} (Ordrelinje))))$$

Oppgave 4

En **transaksjonslog** er en logg som kartlegger planlagte transaksjoner før de blir utført. Dette gjør at dersom det skulle skje en feil med transaksjonen, eller at systemet svikter på grunn av f.eks strømbrudd og blir avbrutt mitt I transaksjonen så vil databasen kunne rulle tilbake til siste lovlige tilstand før alle påbegynte transaksjoner. Så kan databasen forsøke kjøre transaksjonene på nytt.

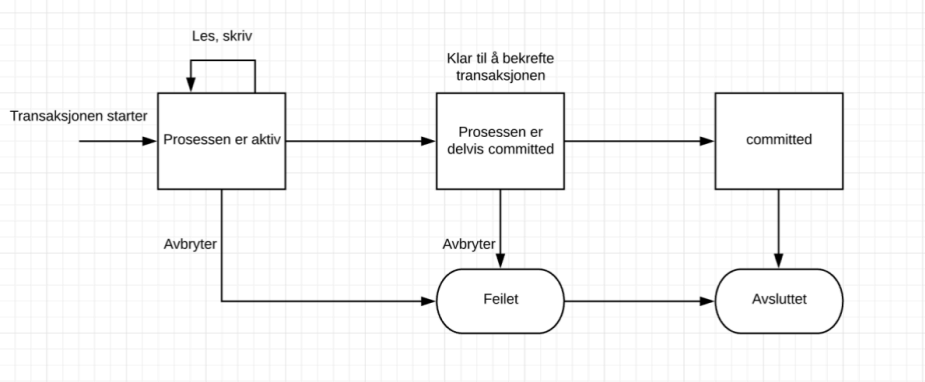
Ett eksempel på varekjøp som blir avbrutt:

VareID	Action	Attributt	Før	Etter	Tid
17	Starter transaksjon				0
17	Endrer	Antall	100	110	1
17	Endrer	PrisEndring	99.99	119.50	2
Strømbrudd	Avbrutt pga strømbrudd				3
Strøm på igjen	Sjekker log og starter fra sist lovlige tilstand				4
17	Starter transaksjon				5
17	Endrer	Antall	100	110	6
17	Endrer	PrisEndring	99.99	119.50	7
17	Committer				8

Låser er en ventemekanisme som sørger for at riktig data blir lagret og hentet ut. F.eks det er opprettet en transaksjon som bruker X for å beregne Y. Samtidig blir det opprettet en transaksjon som bruker Y for å beregne Z. Den første transaksjonen legger en lås på Y til transaksjonen er fullført slik at kommende transaksjoner vil bruke riktig og oppdatert Y.

Ett eksempel på tapt oppdatering ser vi I tabellen under:

Her unngår vi tapt oppdatering da det blir lagt på skrivelås slik at verdiene blir riktig. Dersom vi ikke hadde hatt skrivelås ville vi endt opp med T2 som satte verdi på disk til 20 for så T1 som bruker gammel verdi og ville satt verdien til 5 istedet for korrekte 15.



På samme måte da en transaksjon foregår på som figuren over, ville en angret oppdatering gitt samme resultat dersom vi ikke hadde skrivelås. Si at T2 hadde avbrutt sin transaksjon på Tid=4 committed. Og T1 henter ut verdien fra disk på Tid=4. Da vil T1 hente ut en verdi som ikke blir skrevet til databasen og da ende opp med å skrive feil verdi.

Lokalkopi T1	T1	Verdi på disk	T2	Lokalkopi T2	Tid
		10	Skrivelås		1
	Vil skrive	10	Les inn	10	2
	Venter	10	Legg på 10	20	3
	Venter	20	Skriv til disk	20	4
	Venter	20	Lås opp		5
	Skrivelås	20			6
20	Les inn	20			7
15	Trekk fra 5	20			8
15	Skriv inn	15			9

	Lås opp	15			10

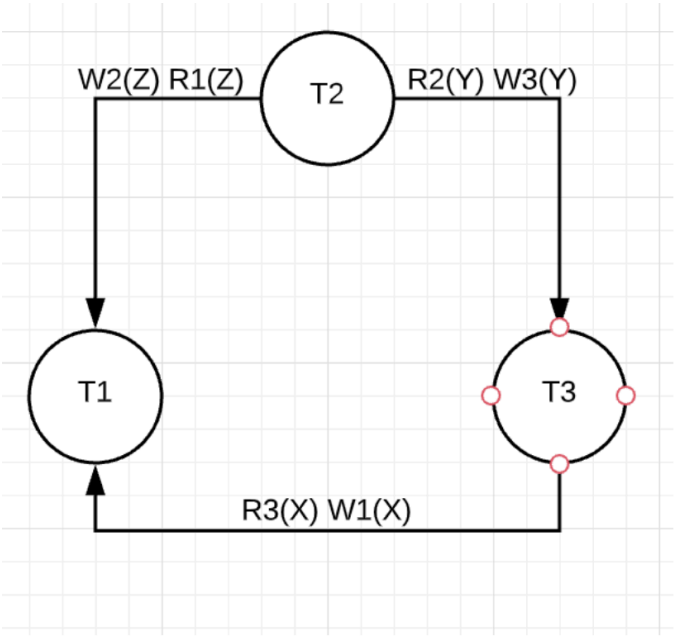
Serialiserbarhet er en måte vi kan kjøre flere transaksjoner samtidig som vil bruke samme data.

Da kjører en transaksjon seg ferdig med dataen før neste transaksjon får tilgang. Låser I seg selv garanterer ikke serialiserbarhet da f.eks en transaksjon T1 skal lese X og Y. Her vil T1 vise feil resultat.

Lokalkopi T1	T1	X	Y	T2	Lokalkopi T2	Tid
		10	10	Låser Y		0
	Låser X	10	10	Leser Y	Y:10	1
X:10	Leser X	10	10	Legger på 10 på Y	Y:20	2
X:10	Låser opp X	10	20	Skriver Y	Y:20	3
	Låser Y	20	20	Låser opp Y		4
	Leser Y	20	20			5
	Låser opp Y	20	20			6
20	Viser X+Y = 30	20	20			7

Samtidig vil en serialiserbar schedule se ut som under:

T1	T2	T3
R(X)		
		R(Y)
		R(X)
	R(Y)	
	R(Z)	
		W(Y)
	W(Z)	
R(Z)		
W(X)		
W(Z)		



Siden:

3R(X) og 1W(X)	T3 → T1
2R(Y) og 3W(Y)	T2 → T3
2R(Z) og 1W(Z)	T2 → T1
2W(Z) og 1R(Z)	T2 → T1
2W(Z) og 1W(Z)	T2 → T1

Den ekvivalente schedulen blir T2 – T3 – T1.

Tofaselåsing betyr at alle låseoperasjoner utføres før opplåsingsoperasjonene. Diverse at betyr dette at dersom vi har låst ett datafelt, så blokkerer vi andre transaksjoner fra å benytte seg av feltet til den første transaksjonen er har låst opp feltet. Vi garanterer derfor serialiserbarhet.

Tid	1	2	3	4	5
T1	Låser X	Leser X		Slipper X	
T2			Låser X		Leser X

En **vranglås** er når flere transaksjoner venter på hverandres låser. F.eks at T1 har låst Y og trenger X for å vise X+Y, samtidig har T2 Låst Y og trenger X for å vise Y*X. Dersom de har låst disse samtidig så vil transaksjonene gå i vranglås og kan ikke utføres.

Vranglåser kan håndteres ved at vi benytter oss av en timeout, deriblant at dersom en transaksjon ikke får gjort noe på så så lang tid så avbrytes transaksjonen og den kan prøve igjen senere. Eller man kan velge en av transaksjonene og fullføre den, samt avbryte den andre og utføre den etterpå.

Lokalkopi T1	T1	X	Y	T2	Lokalkopi T2	Tid
	Låser X	15	10	Låser Y		0
15	Leser X	15	10	Leser Y	10	1
15	Låser Y	15	10	Låser X		2
	Venter på Y..	15	10	Venter på X..	Y:20	3
	Venter på Y..	15	10	Venter på X..		4

Isolasjonsnivåer er fire forskjellige nivåer av hvor åpent data som blir brukt I en transaksjon er for andre transaksjoner I systemet.

Vi definerer isolasjonsnivået utifra hvorvidt det er mulig med usikker lesing, ikke repeterbar lesing og lesing av fantomer.

Fra mest synlig til minst synlig har vi:

Read uncommited → Read commited → Repeatable read → Serializable.

I read uncommited vil transaksjonen ha tilgang til data som ikke er committed enda. Transaksjonene er ikke isolert.

I read commited garanterer vi at data som blir lest har blitt skrevet før de blir lest. Vi kan fortsatt oppnå forskjellige verdier dersom andre transaksjoner bruker verdien som har blitt lest.

I repeatable read unngår vi at data vi har lest vil gi andre verdier dersom de blir lest på nytt ved at en transaksjon holder lese lås på felt I transaksjonen til de er committed.

Ett eksempel på hvor det kan være greit å tillate litt mer åpenhet er f.eks en nettbutikk. Si det er kun 2 varer igjen på lager, 2 personer har lagt de I handlekurven sin men fortsetter og kikke rundt I butikken. Så kommer en 3. Person og vil kjøpe den samme tingen. Dersom vi ikke hadde tillatt samtidig tilgang ville den vært "reservert" av de to første selv om det ikke er sikkert at de vil kjøpe den gjenstanden.

Dersom vi tillatter samtidig tilgang så vil den 3. Personen kunne kjøpe gjenstanden.