Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

LØSNINGSFORSLAG TIL EKSAMENSOPPGAVE I FAG TDT4145 – DATAMODELLERING OG DATABASESYSTEMER, ver 18.mai 2011

Faglig kontakt under eksamen: Svein Erik Bratsberg

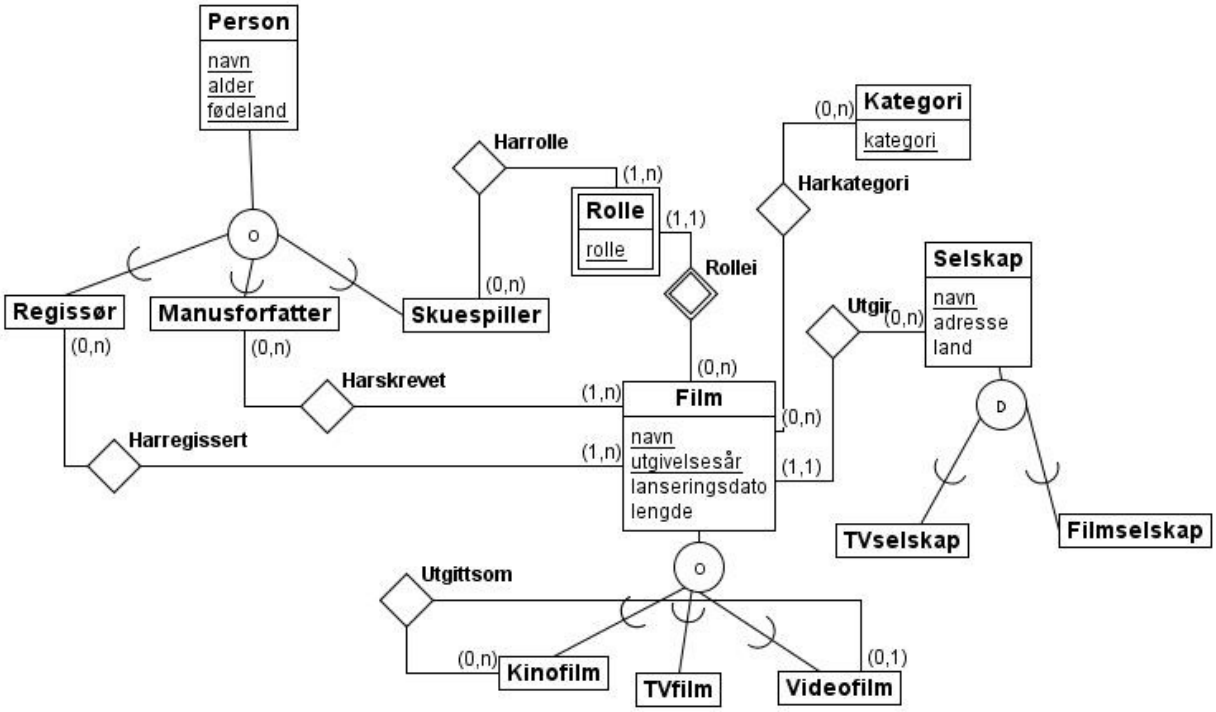
Tlf.: 50382

Eksamensdato:	**26. mai 2009**
Eksamenstid:	**09.00-13.00**
Tillatte hjelpemiddel:	**D: Ingen trykte eller håndskrevne hjelpemiddel tillatt. Bestemt, enkel kalkulator tillatt.**

Språkform: Bokmål

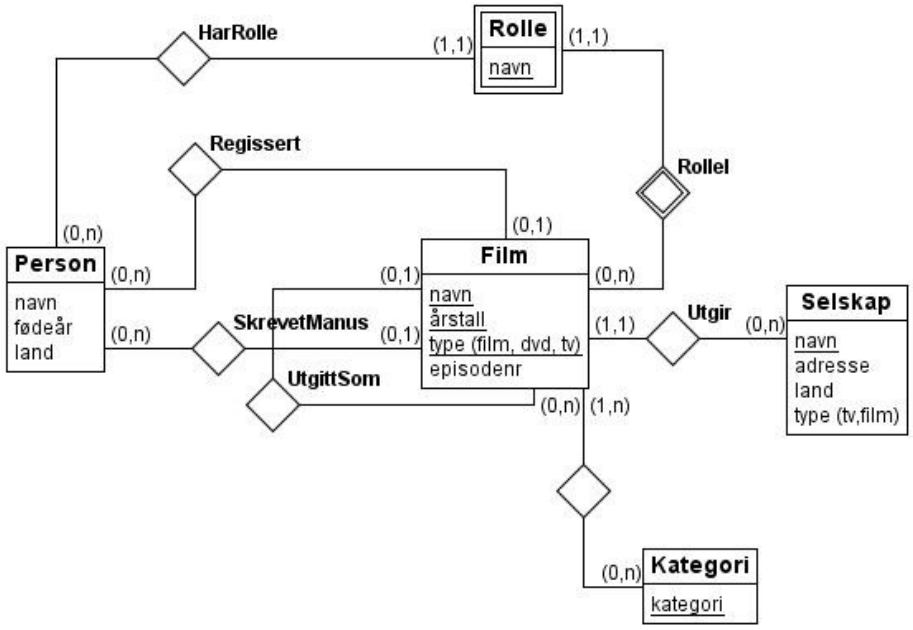
Sensurdato: 16. juni 2009

Oppgave 1 – Datamodellering – 20 %



Det kan gjøres en del antagelser og forenklinger her. For eksempel antar vi at en rolle innehas kun av en skuespiller. Det kan legges på flere eksistensavhengigheter her, for eksempel at et selskap ikke er et selskap uten å ha gitt ut noen film og lignende.

En annen løsning er å forenkle hierarkiene, slik at vi har Person, Film, Rolle, Kategori, Serie og Selskap som entitetsklasser, og har mange relasjoner mellom disse.



Oppgave 2 – Relasjonsalgebra og SQL – 25 %

NB. Det er ikke spurt om SQL i a) og b), kun algebra.

a)

PROJECT_fagnavn (SELECT_(studnavn='Ole Brum')Student JOIN
Oving JOIN Fag)

```
SELECT fagnavn
FROM Fag, Oving, Student
WHERE Student.studnavn='Ole Brum' AND
      Student.studnr=Oving.studnr AND
      Oving.fagnr=Fag.fagnr;
```

b)

PROJECT_fagnavn (FAG) -
PROJECT_fagnavn (SELECT_(studnavn='Ole Brum')Student JOIN
Oving JOIN Fag)

```
SELECT fagnavn
FROM Fag
```

```
EXCEPT
SELECT fagnavn
FROM Fag, Oving, Student
WHERE Student.studnavn='Ole Brum' AND
        Student.studnr=Oving.studnr AND
        Oving.fagnr=Fag.fagnr;
```

c)

```
SELECT epost
FROM Oving, Student
WHERE Student.studnr=Oving.studnr AND
        Oving.fagnr='TDT4145';
```

d)

```
CREATE VIEW Svarteliste AS
SELECT Student.studnr, Student.studnavn
FROM Student, Oving
WHERE Student.studnr=Oving.studnr AND
         Oving.fagnr='TDT4145' AND
        Oving.godkjent='Ja'
GROUP BY Student.studnr, Student.studnavn
HAVING count(*)<5;
```

e)

```
SELECT studnavn, COUNT(*)
FROM Student, Oving
WHERE Student.studnr=Oving.studnr AND
Oving.godkjent='Ja'
GROUP BY studnavn
ORDER BY COUNT(*) DESC;
```

f)

```
SELECT fagnavn
FROM Fag, Oving
WHERE Fag.fagnr=Oving.fagnr AND
        Oving.godkjent='Nei'
GROUP BY Fag.fagnavn
HAVING count(*)>= ALL (SELECT count(*)
                       FROM Fag, Oving
                       WHERE Fag.fagnr=Oving.fagnr AND Oving.godkjent='Nei'
                       GROUP BY Fag.fagnavn);
```

Oppgave 3 – Lagring og indekser – 25 %

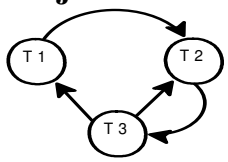
a) Variabellengdeposter er beskrevet i 9.7.2 med to forskjellige formater, enten med feltpekere eller med feltseparasjonstegn. Blokkformater er beskrevet i læreboka i 9.6. Blokker med variabellengdeposter kan ha postene lagret ”unpacked” fra starten av blokka, mens slutten har et ”slot directory” med pekere til postene og antall pekere.

b) Antar at det ikke er slettet noen poster i databasen.
   i. Heapfil: **200** blokker med 120 poster hver i gjennomsnitt.
   ii. Clustered B-tre-indeks: Løvnivå: 200/0.67 = 300 blokker. 1.indeks: 300 poster * 12 byte = 3600 byte, dvs en blokk. Svar **301**, evt. 300 eller 299 (avhengig av avrunding for løvnivå).
   iii. Clustered hash-indeks: 200/0.8 = **250** blokker.
   iv. Unclustered B-tre-index med heapfil: 200 (heapfil) + 24000*12/(8192*0.67) gir 53 (B-tre løvnivå) + 1 (B-tre nivå 1) = **254**.

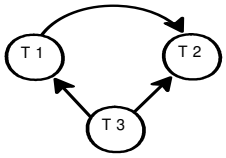
b) Av de måtene som er nevnt ovenfor, bestem hvilke/hvilken som er de/den mest hensiktsmessige måten å lagre tabellen gitt at de følgende SQL-setningene er dominerende. Begrunn svarene:
   i. **Clustered hashindex**. Suveren på direkteaksess på hashnøkkel.
   ii. **Heapfil**. Billig å sette inn.
   iii. **Unclustered B-tre**. Bruk kun indeksen, ikke heapfil. Trenger bare å lese 54 blokker.
   iv. **Clustered B-tre**, slipper sortering. Evt **heapfil** med intern sortering i minne.

Oppgave 5 – Transaksjoner – 15 %

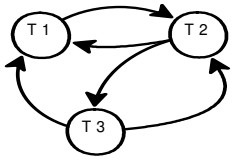
i)



ii)



iii)



(1) T3 -> T1, T3 -> T2, T1 ->T2, T2->T3: sykel, **ikke konliktserialiserbar**
(2) T3 -> T1, T1->T2, T3->T2, **konliktserialiserbar**
$r_3(X)$; $r_3(Y)$; $w_3(Y)$;$r_1(X)$; $r_1(Z)$; $w_1(X)$; $r_2(Z)$; $r_2(Y)$;$w_2(Z)$;$w_2(Y)$
(3) T3-> T1, T1->T2, T1->T3, T2->T3, T3 ->T2. Sykler, **ikke konliktserialiserbar**.

b) ARIES bruker **Dirty Page Table** (DPT) og **recLSN** som sier hvilke datablokker som var skitne (dirty) ved et sjekkpunkt og den første loggposten (recLSN) som gjorde datablokka skitten siden den sist var skrevet til disk. Dette er informasjon som er skrevet i sjekkpunktloggposten. Dette gjør at ARIES slipper å lese inn en del datablokker ved REDO recovery. Når redo ser på en loggpost (med referanse til ei datablokk):
   - Hvis datablokka ikke er DPT, trenger redo ikke å lese den inn.
   - Hvis datablokka er i DPT, men har en recLSN som er større enn LSNen til loggposten, da trenger ikke redo lese datablokka inn.

Oppgave 6 – Normalisering – 15%

Gitt følgende tabell som registrerer en fast ukeplan for forelesninger ved NTNU:

```
Forelesning(auditorium,bygning,faglærer,antallSittepl,ukedag,tid,fagnr)
```

a) Foreslå hvilke funksjonelle avhengigheter som gjelder for denne tabellen.
Tanken her var å modellere NTNU.

Auditorium -> bygning, antallSitteplasser
Auditorium, ukedag, tid -> fagnr
Fagnr -> faglærer

Antar her at det ikke er flere faglærere for et fag og at auditorium har unike navn.

Hvis det er flere faglærere for et fag, blir avhengighetene:
Ukedag,tid,fagnr->faglærer,auditorium
Auditorium->bygning,antallSitteplasser
Ukedag,tid,auditorium->fagnr,faglærer

b) Finn alle kandidatnøkler for tabellen

Tillukningen {**auditorium,ukedag,tid**}+ = alle attributtene i tabellen
Dette er en supernøkkel og den er minimal. Altså, en nøkkel for tabellen.
{**fagnr,ukedag,tid**} vil også være en kandidatnøkkel hvis det er flere faglærere per fag som antas.

c) Vis hvordan tabellen bør normaliseres slik at alle de resulterende tabellene oppfyller kravene til Boyce-Codd normalform (BCNF).

Tar tak i funksjonelle avhengigheter som bryter med BCNF. Dekomponerer til tre tabeller
R1(audiorium,bygning,antallsittepl) R2(fagnr,faglærer) R3(auditorium,ukedag,tid,fagnr)
Tabellene kan omdøpes til:
Auditorium(auditorium,bygning,antallSittepl)
Fag(fagnr,faglærer)
Forlesning(auditorium,ukedag,tid,fagnr)

Med den ekstra antagelsen får vi følgende oppdeling:
Forelesning(ukedag,tid,fagnr,auditorium,faglærer)
Auditorium(auditorium,bygning,antallSitteplasser)
Ukedag,tid,auditorium er også kandidatnøkkel for Forlesning.