Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

LØSNINGSFORSLAG TIL KONTINUASJONSEKSAMEN I FAG
TDT4145 – DATAMODELLERING OG DATABASESYSTEMER

Eksamensdato: **10. august 2011**

Eksamenstid: **09.00-13.00**

Tillatte hjelpemiddel: **D: Ingen trykte eller håndskrevne hjelpemiddel tillatt. Bestemt, enkel kalkulator tillatt.**

Oppgave 1 – ER- og relasjonsmodell – 10 %

Vi forutsetter at et Photo kan beskrives med en Tag uten at det gjøres av en Photographer. Vi registrerer bare den første Photographer som annoterer et Photo med en Tag. Vi forutsetter videre at et Photo som har en TaggedBy-relasjon med en Tag, også har en DescribedBy-relasjon med den samme Tag-en.

Photo(Id, Name, Date, PhotographerEmail)

PhotographerEmail er fremmednøkkel mot Photographer-tabellen og kan ikke ha verdien NULL.

Photographer(Email, Name, URL)

Tag(Id, Name, Description, DefinedByEmail, DefinedDate)

DefinedByEmail er fremmednøkkel mot Photographer og kan ikke ha verdien NULL.

DescribedBy(PhotoId, TagId)

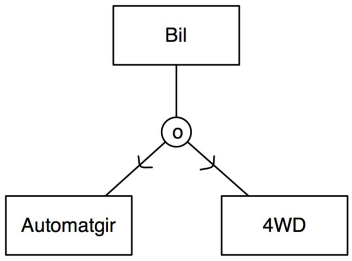
PhotoId er fremmednøkkel mot Photo-tabellen og TagId er fremmednøkkel mot Tag-tabellen.

TaggedBy(PhotoId, TagId, PhotographerEmail, TaggedByDate)

PhotoId er fremmednøkkel mot Photo-tabellen, TagId er fremmednøkkel mot Tag-tabellen. PhotographerEmail er fremmednøkkel mot Photographer-tabellen.

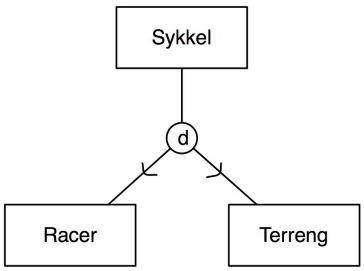
Oppgave 2 – ER-modeller – 10 %

Delvis og overlappende



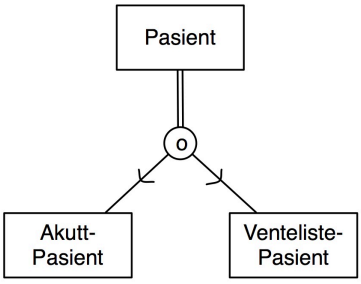
Biler kan ha både automatgir og firehjulstrekk (4WD), eller bare en av delene. Det kan finnes biler som ikke har noen av delene.

Delvis og disjunkt



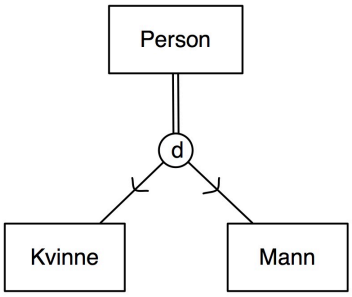
Sykler er enten racersykler eller terrengsykler, eller ingen av delene. Det kan ikke finnes sykler som er både racer- og terrengsykkel.

Total og overlappende



En pasient kan stå på venteliste og komme inn som akuttpasient. Det finnes ikke pasienter som ikke er noen av delene.

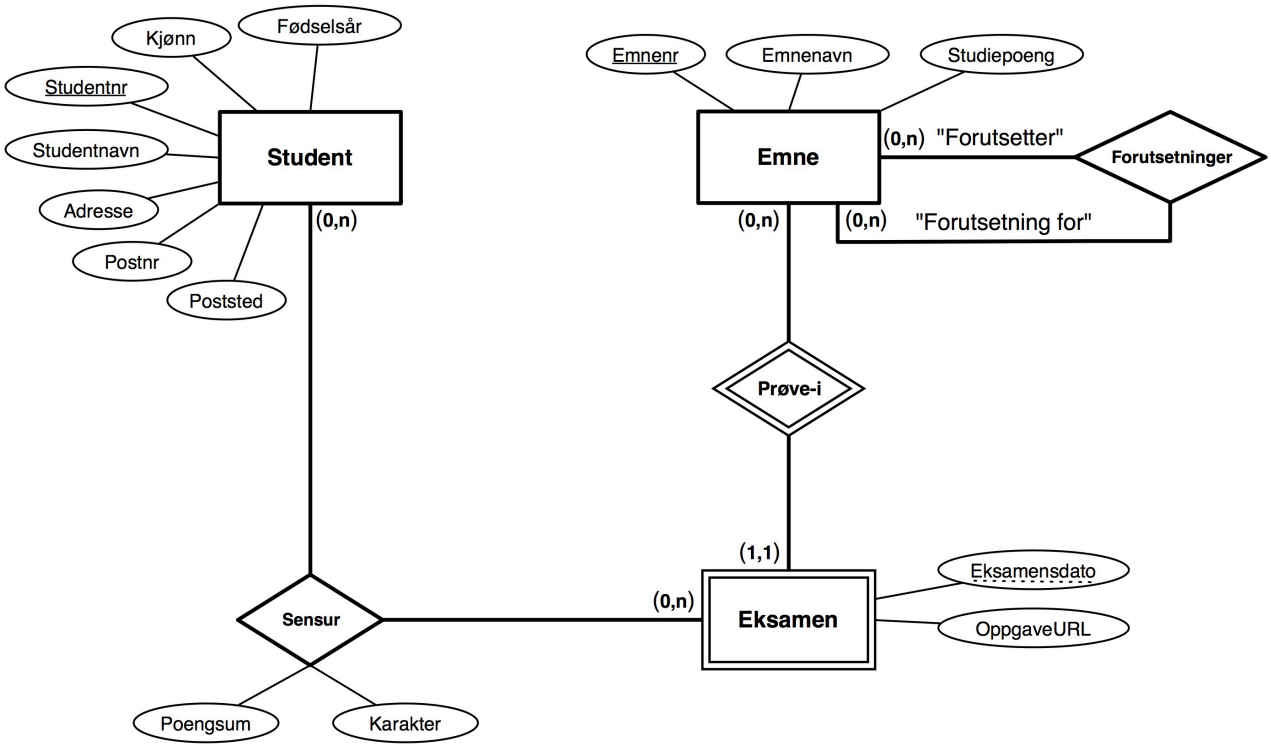
Total og disjunkt



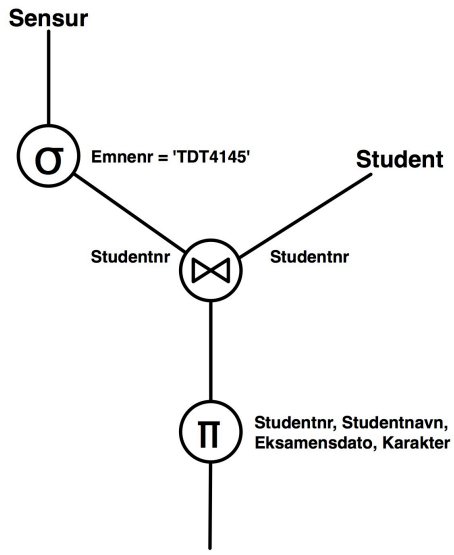
Alle personer er enten kvinner eller menn, det finnes ingen som er begge deler.

Oppgave 3 – Relasjonsalgebra og SQL – 20 %

a) ER-diagram:



b) Relasjonsalgebra:



c) SQL-spørring:

```
SELECT Emnenr, Emnenavn
FROM   Emne
WHERE  Emnenr IN ( SELECT Byggerpaaemnenr
                   FROM Forutsetninger
                   WHERE Emnenr = 'TDT4145' )
```

d) SQL-spørring:

```
SELECT min(Poengsum), max(Poengsum), avg(Poengsum)
FROM Sensur
WHERE Emnenr = 'TDT4145'
  AND Eksamensdato = '2011-6-1'
  AND Karakter = 'C'
```

e) SQL-spørring:

```
UPDATE Sensur
SET Karakter = 'A'
WHERE Emnenr = 'TDT4145'
  AND Eksamensdato = '2011-6-1'
  AND Poengsum > 87
```

f) SQL-spørring.

```
SELECT Studentnr, Studentnavn
FROM Student ST, Sensur SE
WHERE ST.Studentnr = SE.Studentnr
  AND SE.Emnenr = 'TDT4145'
  AND Poengsum = ( SELECT max(Poengsum)
                   FROM Sensur
                   WHERE Studentnr = ST.Studentnr )
```

Oppgave 4 – Normaliseringsteori – 20 %

a) Tillukningen (X^+) til en mengde attributter (X) er mengden av alle attributter som er funksjonelt avhengig av attributtene i X.

b) $a^+ = ab$
$ac^+ = abcd$

c) En *kandidatnøkkel* er en minimal supernøkkel eller identifikator for tabellen. En *supernøkkel* er en mengde attributter som entydig bestemmer et bestemt tuppel i tabellen – det kan aldri finnes to tupler med samme verdi for attributtene i supernøkkelen. En supernøkkel er *minimal* når det ikke finnes noen ekte delmengde som er en supernøkkel.

Kandidatnøkler i R: ac (verken a eller c er funksjonelt avhengig av andre attributter, de må derfor være med i alle supernøkler, sammen er de en minimal supernøkkel).

d) Andre normalform (2NF) forbyr delvise avhengigheter fra en kandidatnøkkel til et ikke-nøkkel-attributt. I R er attributtet b delvis avhengig av nøkkelen (ac), tabellen er derfor ikke på 2NF. Den høyeste normalformen er 1NF.

e)

i) Boyce-Codd-normalform (BCNF) krever at alle avhengigheter går ut fra supernøkler. I Hund har vi avhengigheten Rase → Hundegruppe der Rase er ikke en supernøkkel. Tabellen oppfyller derfor ikke BCNF. Tredje normalform forbyr avhengigheter blant ikke-nøkkel-attributtene, avhengigheten Rase → Hundegruppe gjør derfor at tabellen ikke er på 3NF. Andre normalform forbyr delvise avhengigheter med utgangspunkt i kandidatnøkler. Siden nøkkelen er kun ett attributt, kan vi ikke ha slike og tabellen oppfyller kravene til 2NF.

ii) Hunder(Registreringsnr, Rase, Fødselsår, Kjønn)

Hundegrupper(Rase, Hundegruppe)

I begge tabellene er alle ikke-nøkkel-attributter funksjonelt avhengige av hele nøkkelen og det finnes ikke andre funksjonelle avhengigheter.

iii) Alle atributter er med i minst en del-tabell, dette sikrer attributtbevaring.

Del-tabellenes felles attributter (Rase) er en supernøkkel i den ene del-tabellen, dette sikrer at del-tabellene kan joines tapsløst tilbake til utgangspunktet.

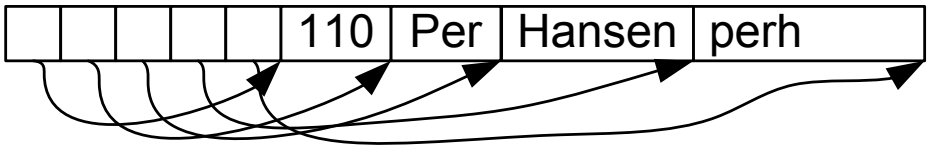
Alle funksjonelle avhengigheter kan sjekkes i en av del-tabellene, dette sikrer bevaring av de funksjonelle avhengighetene.

iv) **Fordeler**: Forholdet mellom rase og hundegruppe lagres en gang og uavhengig av om det finnes hunder av alle raser. Dette fjerner redundans i den opprinnelige tabellen, sikrer konsistens og gjør at vi unngår anomalier knyttet til innsetting, oppdatering og sletting av hunder.

Ulemper: Det er ressurskrevende å joine sammen del-tabellene hvis vi trenger den opprinnelige tabellen. En del spørringer vil være enklere å formulere mot den opprinnelige tabellen.

Oppgave 5 – Lagring og indekser – 25 %

a) I og med at vi har to variabellengde strenger her kan det være lurt å bruke et format som tillater variabellengde felter. Bruker "peker-vector"-løsningen:



b) `aNr` er fastlengde. `Fornavn` og `etternavn` er variabellengde, mens `epost` er av fast lengde med "padding" av blanke.

Dette taler for at tabellen kan lagres som en clustered hash på `epost`. Da vil hvert oppslag ta 1,2 diskaksesser i gjennomsnitt. Det er lite som taler for bruk av B-tre her, for hva kan man bruke en sortering av epostadresser til, annet enn evt. til merge-join?

c) Hvis tabellen hadde vært lagret som en clustered hash på `fornavn`, ville det være mange poster som hadde samme verdi da det ikke finnes så mange forskjellige fornavn. Da måtte hashstrukturen tillate mange poster med samme nøkkel. I tillegg ville det bli stor fare for overflyt da poster med samme nøkkel vil hashe til samme blokk.

d) Her kan det være lurt å lage en primærindeks (anr, epost) som gjerne kan være en hashindeks. Det er viktig å bruke en unik indeks for å kunne sørge for at det maksimalt kun finnes en post for hver verdi av anr. Da antar vi løsningen fra b) for lagringen av selve tabellen.

e) Her har vi to lagringsstrukturer:

1) Clustered hash på epost for lagring av selve postene: I og med at vi har valgt et variabellengde postformat, må det her gjøres antagelser om postlengder. Vi antar eksempelposten over er mindre enn gjennomsnittsposten. Vi antar at en "feltpeker" er to byte og at anr er 4 byte og de to strengene til sammen er 30 byte og epost er 8 byte. Da er hver post 5*2+4+30+8 = 52 byte. Vi antar fyllingsgrad 80 % for hashing.

5000*52/(8192*0.8) blokker = 40 blokker

2) Primærindeks (anr, epost) som hash: Vi antar fastlengde representasjon 4 + 8 byte = 12 byte.

5000*12/(8192*0.8) blokker = 10 blokker.

Til sammen 50 blokker.

Hvis andre lagringsstrukturer er valgt, vil det være andre svar. Ved bruk av B-tre for 1) vil det være 5000*52/(8192*0.67) blokker = 48 blokker. Ved bruk av heapfil for 1): 5000*52/8192 = 32 blokker. Da må det være to indekser i tillegg, en for aNr og en for epost.

Oppgave 6 – Transaksjoner – 15 %

a) (10%) De fire isolasjonsnivåene i SQL forklareres som regel ved den følgende tabellen:

Level	Dirty read	Unrepeatable read	Phantom
READ UNCOMMITTED	Maybe	Maybe	Maybe
READ COMMITTED	No	Maybe	Maybe
REPEATBLE READ	No	No	Maybe
SERIALIZABLE	No	No	No

SERIALIZABLE setter låser før lesing og skriving av objekter, inkludert mengder av objekter som må være uendret og holder på låsene inntil avslutning av transaksjon (strikt 2PL). Indekslåser eller tabellåser brukes for mengder.

REPEATBLE READ setter de samme låsene som SERIALIZABLE bortsett fra låser på mengder av objekter.

READ COMMITTED setter skrivelåser før skriving og holder de til commit. Leselåser slippes med en gang etter lesing.

READ UNCOMMITTED krever READ ONLY, og ingen leselåser settes av en slik transaksjon.

b) Det er fordelaktig å implementere durability ved å skrive loggen ved commit, dvs. NO-FORCE. Loggen er lagret sekvensielt både i minne og på disk. Derfor vil det ved commit holde å skrive en gang til disken. FORCE kan involvere skriving av mange datablokker til forskjellige steder på disken.