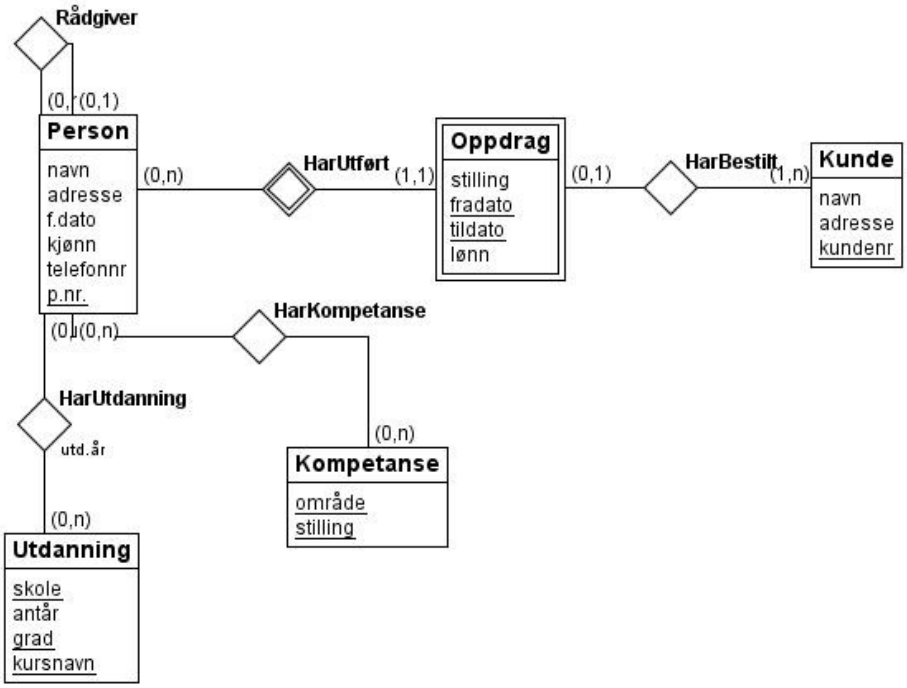
Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitskap

LØSNINGSFORSLAG TIL
EKSAMENSOPPGAVE I FAG TDT4145 – DATAMODELLERING OG DATABASESYSTEMER
4. juni 2008

Oppgave 1 – Datamodellering – 20 %



Litt ullen tekst, men det viktige her å finne fornuftige entitetsklasser, attributter og relasjoner. Av og til må attributter oppfinnes, også nøkler må lages.

Oppgave 2 – Relasjonsalgebra og SQL – 20 %

a) *PROJECT_sensorNavn(Sensor JOIN (Eksamen JOIN PROJECT_emnekode(SELECT_emnenavn='Objektorientert systemutvikling' (Emne))))*

b) *PROJECT_studnavn(Student JOIN (Eksamen JOIN (Ansvarlig JOIN SELECT_faglærernavn='Per Hansen'(Faglærer))))*

```
Oppgaven har med både Faglærer og Ansvarlig. Dette kan tolkes slik at det er en
annen faglærer som er ansvarlig enn den som er referert fra Emnetabellen. Det
finnes hovedsaklig tre muligheter: En som joiner
Student, Eksamen, Ansvarlig og Faglærer (mest korrekt); en som joiner
Student, Eksamen, Emne og Faglærer; og en som joiner Student, Eksamen, Emne,
Ansvarlig og Faglærer.
```

c) *SELECT sensornavn*
FROM sensor, eksamen, emne
WHERE emnenavn='Systemutvikling' AND sensor.sensorid=eksamen.sensorid AND eksamen.emnekode=emne.emnekode;

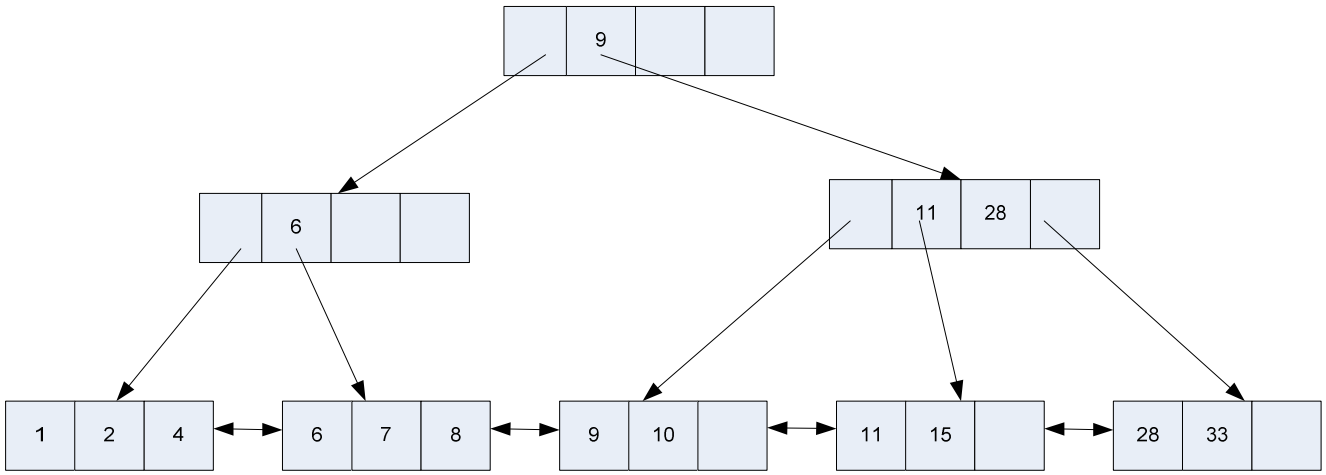
d) *SELECT studnavn, karakter*
FROM student, eksamen
WHERE student.studnr=eksamen.studnr AND eksamen.emnekode='TDT4145'
ORDER BY karakter;

e) *SELECT COUNT(*)*
FROM Eksamen
WHERE emnekode='TDT4145' AND karakter='A';

f) *SELECT studnr*
FROM eksamen
WHERE poengsum = (SELECT MAX(poengsum) FROM eksamen);
Godtar ikke svar med bruk av LIMIT her.

Oppgave 3 – Lagring og indekser – 20%

a)



B-treet er avhengig av rekkefølgen av innsettinger. Svar som ikke opererer med 3 nøkler og fire pekere i indeksblokker, som da ikke får splitt av indeksblokker og tre nivåer, bør det trekkes for.

b) Tre blokker rett ned: 30 ms

c) Tre blokker rett ned, og så resten av løvnodene til høyre: 70 ms

d) 30 ms evt 40 ms hvis du må sjekke den neste blokka også, men kan vite dette ved å studere neste nøkkel på blokka på nest nederste nivå.

e) Postene settes inn fortløpende fra starten. 4 blokker (alle fulle) gir 40 ms evt 30 ms hvis du antar unike nøkler, du stopper da når du har funnet posten. Noen kan også forutsette at du leser en directoryblokk før du leser selve heapfila. Da får du 40 ms eller 50 ms.
f) 40 ms = lesing av 4 blokker pluss evt en directoryblokklesing.
g) 40 ms = lesing av 4 blokker pluss evt en directoryblokklesing.

Oppgave 4 – Queryevaluering – 10%

a) 1500 I/Oer. Alle blokkene må leses.
b) Ca 150 I/Oer. Evt 151, 152, 153 eller 154, avhengig om hvor høyt treet er og om du må sjekke en ekstra løvblokk for å vite at du skal stoppe. Scanner B-treet sekvensielt langs løvnodene til betingelsen feiler.
c) Ca 150*100 = 15000 I/Oer. Får en I/O per post i B-tre-indeksen.

Oppgave 5 – Transaksjoner – 15%

a) **Reading uncommited data**, Write-read conflicts eller "Dirty read": Leser ikke-committede data, kan da ha fått committet endringer basert på data som senere blir abortert bort. For eksempel $r_1(A)$, $w_1(A)$, $r_2(A)$,$w_2(A)$,C_2,$r_1(B)$,$w_1(B)$,$Abort_1$
b) **Unrepeatable read**, Read-write conflicts. Ved rescan (for eksempel nested loops) kan verdiene ha endret seg siden siste scan. For eksempel: $r_1(A)$,$r_2(A)$,$w_2(A)$,C_2,$r_1(A)$,$w_1(A)$,C_1
c) **Overwriting uncommitted** data, write-write conflicts. Data blir overskrevet. For eksempel $w_1(A)$,$w_2(A)$,$w_2(B)$,C_2,$w_1(B)$,C_1
Her har vi "blind writes", dvs. data blir skrevet uten å bli lest først. Dette kan for eksempel være at transaksjon 1 setter lønna til A og B til samme verdi: 1000, mens transaksjon 2 setter lønna til A og B til samme verdi 2000. Denne historien sørger for at lønna er 2000 for A, men 1000 for B.

Oppgave 6 – Normalisering – 15%

a) AB er kandidatnøkkel
b) E er delvis avhengig av nøkkel AB, D og G er også delvis avhengig av nøkkel. Da er ikke tabellen på 2. normalform. Vi antar da at tabellen er på 1.normalform.
c) R1(A,B,C)
   R2(A,D,G)
   R3(B,E)
   R4(C,G,F)

.