NTNU
Det skapende universitet

Institutt for datateknikk og informasjonsvitenskap

Løsningsskisse til eksamensoppgave i TDT4145 Datamodellering og databasesystemer

Eksamensdato: 12. august 2013

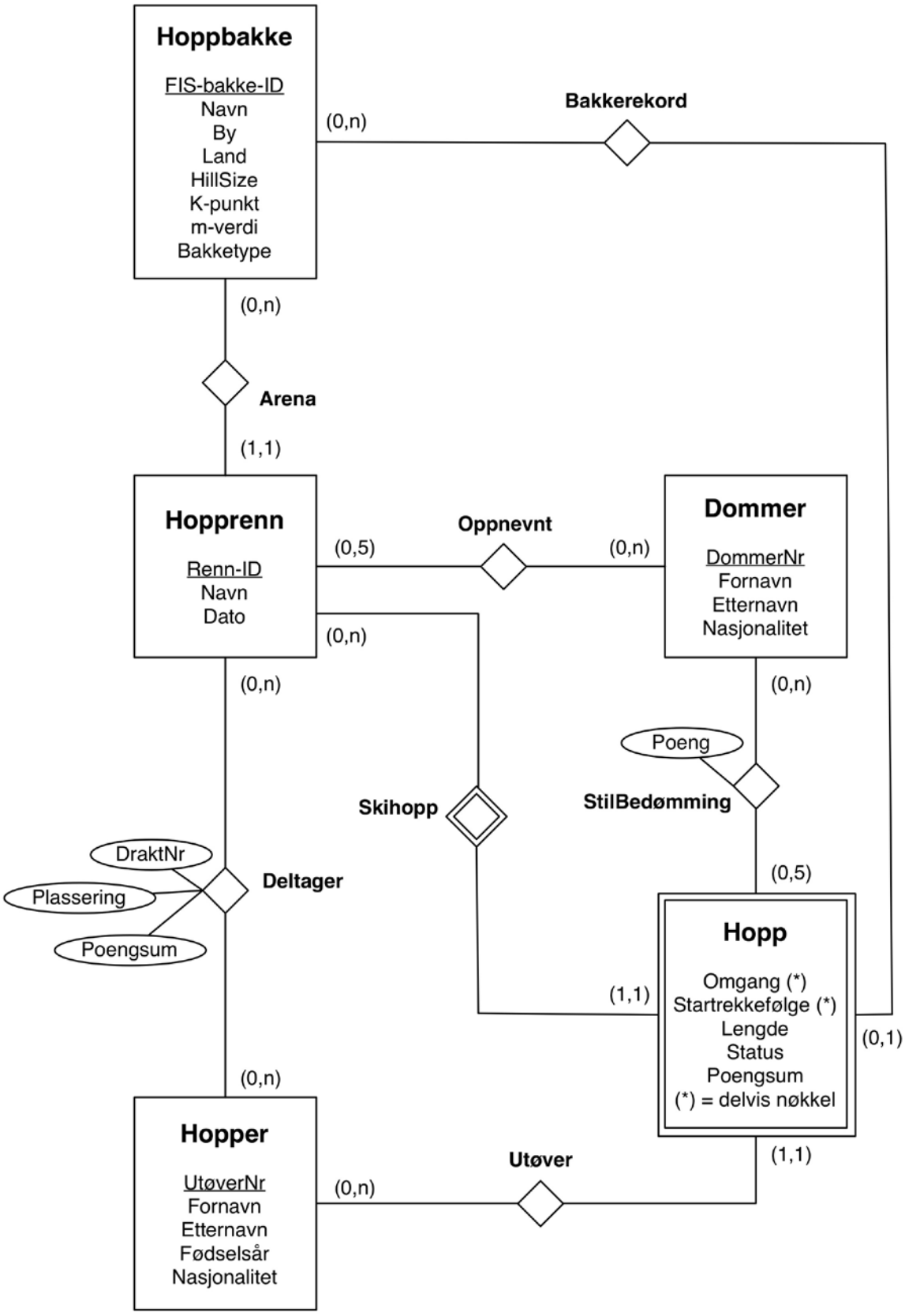
Eksamenstid (fra-til): 15:00 - 19:00

Hjelpemiddelkode/Tillatte hjelpemidler:

D – Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

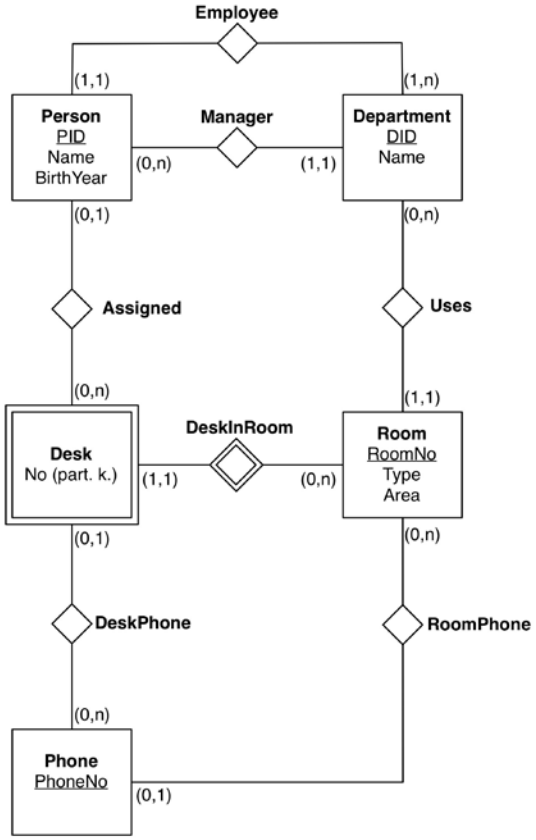
Annen informasjon:

Oppgave 1 – Datamodeller (20 %)

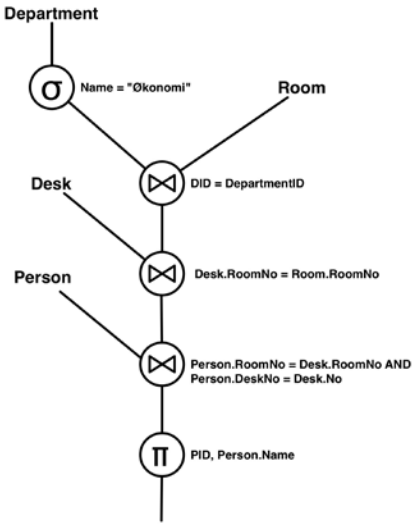


Oppgave 2 – Relasjonsalgebra og SQL (20 %)

a) Vi gjør følgende forutsetninger: En person kan være leder for flere avdelinger, men trenger ikke å være leder. En avdeling har minst en ansatt. En avdeling trenger ikke å ha noe rom. Et rom trenger ikke å ha noen arbeidsplass, men kan ha flere. Et rom trenger ikke å ha en telefon, men kan ha flere. Et telefonnummer trenger ikke å være tilknyttet en arbeidsplass, men kan være tilknyttet flere. En arbeidsplass trenger ikke å ha en tilknyttet person, men kan ha flere.



b) Relasjonsalgebra:



c) SQL:

```
UPDATE Department
SET LeaderID = 100
WHERE Name = 'Økonomi';
```

d) SQL:

```
CREATE VIEW Okonomi
AS SELECT PID, P.Name, RoomNo
   FROM Person AS P, Department AD D
   WHERE DepartmentID = DID
     AND D.Name = 'Økonomi';
```

e) SQL:

```
SELECT RoomNo, count(PID)
FROM Person
GROUP BY RoomNo
HAVING count(PID) > 4
ORDER BY count(PID) DESC;
```

Oppgave 3 – Teori (20 %)

a) ID er den eneste kandidatnøkkelen for tabellen. Siden vi ikke har en sammensatt nøkkel, kan vi ikke ha delvise avhengigheter av nøkkelen. Tabellen vil derfor være på andre normalform (2NF).

   Vi har to avhengigheter, Club → Region og PostalCode → City, der venstreside-attributtene ikke er en supernøkkel og høyresideattributtene ikke er nøkkelattributter. Tabellen er derfor *ikke* på tredje normalform (3NF).

b) Clubs(Club, Region), PostalCodes(PostalCode, City) og Athletes(ID, Name, BirthYear, Sex, Club, PostalCode).

   Komponent-tabellene er på BCNF siden alle funksjonelle avhengigheter har venstresider som er supernøkkel aktuell tabell. Dekomponeringen har attributtbevaring siden alle attributter i utgangspunktet er med i minst en tabell. Alle avhengighetene i F er ivaretatt i minst en komponent-tabell. Dette sikrer bevaring av funksjonelle avhengigheter. Athletes joiner tapsløst med Clubs siden det felles attributtet (Club) er supernøkkel i Clubs. Resultatet av denne foreningen joiner tapsløst med PostalCodes siden det felles attributtet (PostalCode) er supernøkkel i PostalCodes. Resultatet av denne foreningen er lik Athlete-tabellen som vi startet med. Dekomponeringen med andre ord tapsløst join egenskapen.

   Dekomponeringen har god kvalitet siden den oppfyller alle de fire kravene som bør oppfylles ved dekomponering av en tabell.

c) Utfylt tabell, nye verdier er markert med grønn bakgrunnsfarge:

A	B	C	D
1	2	4	5
2	b_{22}	4	6
1	2	b_{33}	b_{34}
2	b_{22}	4	6
3	b_{52}	4	b_{55}

I tillegg vil det være slik at b_{55} ikke kan være lik 5 eller 6. Kombinasjonen av b_{33} og b_{34} kan ikke ha verdien (4,6) eller (4, b_{55}).

Oppgave 4 – Lagring, indekser og queryutføring (20 %)

a) Hvis du bare setter inn, er heapfil veldig greit da du ikke trenger å søke.

b) Her vil de fleste poster sannsynligvis kvalifisere og en heapfil er da grei nok da alle postene må inspiseres.

c) Her kan et B+-tre organisert på søkenøkkel (exerno, studno) være en god løsning. Da er postene clustret omkring exerno. Ellers er en heapfil et mulig svar her også. Det er exerno som det spørres etter primært, så enten en hash på exerno (neppe lurt da det er få øvinger) eller et clustered B+-tre på (exerno, studno). Kanskje ikke så supergunstig det heller? I og med at det er få øvinger sannsynligvis, vil en indeks på exerno være lite effektiv. Så kanskje en heapfil er greit her?

d) Her kan et B+-tre med søkenøkkel med studno evt. (studno, exerno) være en god løsning. Eller en hash på studno, slik at alle rader for en student sannsynligvis havner i samme hashblokk.

e) Her kan et B+-tre med søkenøkkel (studno, exerno) være en god løsning. Scan sideveis i B+-treet og tell opp hvor mange godkjente det er for hver student, for de kommer etterhverandre pga. leksikalsk sortering.

Oppgave 5 – Transaksjoner (10 %)

a) Det ene eksempelet bør være om flerbrukerproblematikk. Det andre eksempelet bør være om abortering og halvgjorte transaksjoner.

b) De fire isolasjonsnivåene i SQL forklares 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

Oppgave 6 – Join (10 %)

a) Nested loop join. For bruk av buffer kan vi ha ei blokk til resultat og ei blokk til den ene tabellen og 6 til den andre. Vi må da lese inn Fakultet i to runder (antar 5 i hver) og for hver runde må hele Student leses: 2 * (5 + 2000) = 4010 I/Oer.

b) Sort-merge join. Bruker notasjonen fra læreboka Elmasri/Navathe.
Sorterer Fakultet. b = 10. $n_B = 8$. n_r = Ceiling (10/8) = 2 sorterte delfiler.
Merge: $d_M = n_B - 1 = 7$. Ceiling ($\log_7(2)$) = 1 pass.
2 * 10 + 2*10*1 = 40 I/Oer.

Sorter Student. b = 2000, $n_B = 8$. n_r = ceiling(2000/8) = 250 sorterte delfiler.
Merge: $d_M = n_B - 1 = 7$. Ceiling($\log_7(250)$) = 3 pass.
2 * 2000 + 2*2000*3 = 16000 I/Oer.

Join av sorterte delfiler: 10 + 2000 = 2010 I/Oer,

Totalt: 40+16000 + 2010 = 18050 I/Oer.