

Institutt for datateknikk og informasjonsvitenskap

**Løsningsforslag for
Eksamensoppgave i
TDT4145 Datamodellering og databasesystemer**

Faglig kontakt under eksamen:

Svein Erik Bratsberg: 99539963

Roger Midtstraum: 99572420

Eksamensdato: 6. august 2014

Eksamenstid (fra-til): 09:00 - 13:00

Hjelpemiddelkode/Tillatte hjelpemidler:

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

Annen informasjon: Versjon 27. mai 2015.

Målform/språk: Norsk bokmål

Antall sider: 5

Antall sider vedlegg: 0

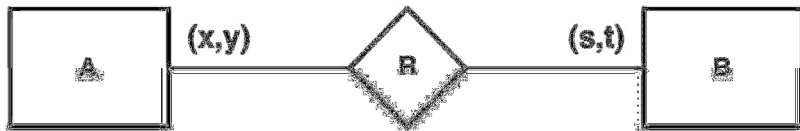
Kontrollert av:

Dato

Sign.

Oppgave 1 – Datamodeller (20 %)

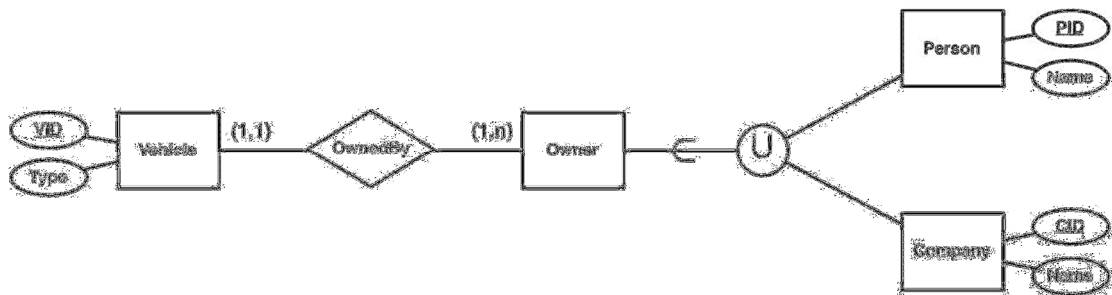
- a) (3 %) I ER-modeller kan vi uttrykke kardinalitets-restriksjoner som vist i figuren under. Forklar hvilke restriksjoner (eng: constraints) som kan uttrykkes ved hjelp av x , y , s og t .



x bestemmer det minste antallet R-relasjoner en entitet i A må delta i, s har tilsvarende rolle for entiteter i B. Dersom x (s) har verdien 0, betyr det at entiteter i A (B) kan eksistere uten å ha noen R-relasjon. Andre (positive) verdier for x (s) betyr at entiteter i A (B) må inngå i minst dette antallet R-relasjoner.

y bestemmer det største antallet R-relasjoner en entitet i A kan delta i, t har tilsvarende rolle for entiteter i B.

- b) (5 %) I ER-diagrammet under har vi vist en ER-modell med bruk av kategori (eng: category). Vis hvordan dette ER-diagrammet kan oversettes til et relasjonsskjema som i størst mulig grad samsvarer med ER-modellen.



Siden superklassene til Owner har ulike nøkler oppretter vi en surrogatnøkkel, OwnerID, for denne klassen som også brukes som fremmednøkkel for å representere sammenhengene mellom entitetene.

Vehicle(VID, Type, OwnerID). OwnerID er fremmednøkkel mot Owner-tabellen og kan ikke ha NULL-verdi.

Owner(OwnerID)

Person(PID, Name, OwnerID). OwnerID er fremmednøkkel mot Owner-tabellen og kan ha NULL-verdi.

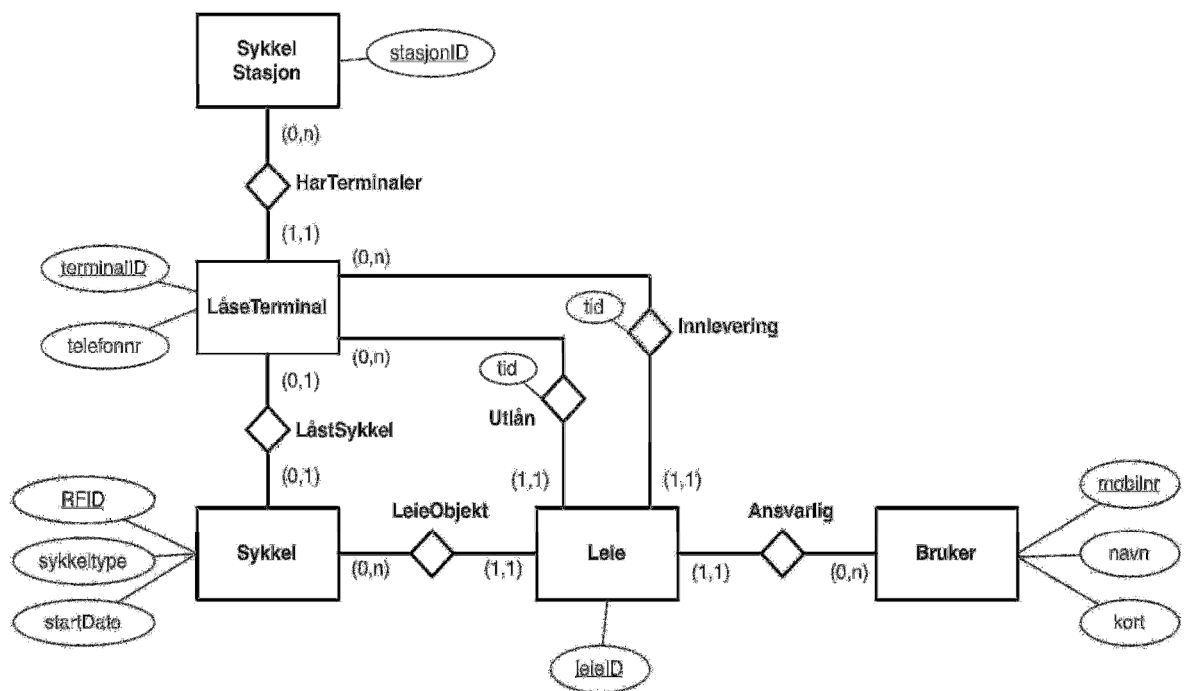
Company(CID, Name, OwnerID). OwnerID er fremmednøkkel mot Owner-tabellen og kan ha NULL-verdi.

- c) (12 %) Lag en ER-modell (du kan bruke alle virkemidler som er med i pensum) for følgende situasjon:

I Nice er det mulig å leie bysykler. Rundt om i byen finnes en del sykkelstasjoner der det er låseterminaler med plass til et antall bysykler. For å kunne leie sykler registrerer man seg med mobiltelefonnummer, navn og et betalingskort. Når man ønsker å leie en sykkel bruker man den registrerte mobiltelefonen og ringer et telefonnummer som er oppgitt for den låseterminalen der sykkelen er låst. Dersom alt er i orden, blir sykkelen frigjort og så betaler man for tiden frem til man igjen låser sykkelen fast i en låseterminal. Alle sykler har en unik RFID-kode som gjenkjennes av låseterminalen når sykkelen leveres inn. Sykkel-leien avsluttes automatisk når låseprosedyren er fullført. De som driver sykkelutleien ønsker å holde oversikt over hvor syklene er låst fast, hvilke sykler som er i bruk, registrerte brukere, hvem som har en sykkel til låns og hvilke utlån som er fullført. Databasen må inneholde nok informasjon til å dokumentere kundenes bruk av syklene. Det finnes ulike typer sykler og man holder rede på når en sykkel settes i drift i systemet.

Gjør kort rede for eventuelle forutsetninger som du finner det nødvendig å gjøre.

I figuren under er det vist et (minimalistisk) ER-diagram som dekker problembeskrivelsen.



Oppgave 2 – Relasjonsalgebra og SQL (20 %)

Ta utgangspunkt i følgende relasjonsdatabase (primærnøkler er understreket) for en enkel student-emne-eksamen-database:

Student(StudentNo, Name, Email)

Subject(SubjectNo, Name)

Exam(ExamNo, SubjectNo, Year, Month, Date)

– SubjectNo er fremmednøkkel mot Subject-tabellen. SubjectNo kan ikke ha NULL-verdi.

ExamRegistration(StudentNo, ExamNo, Grade)

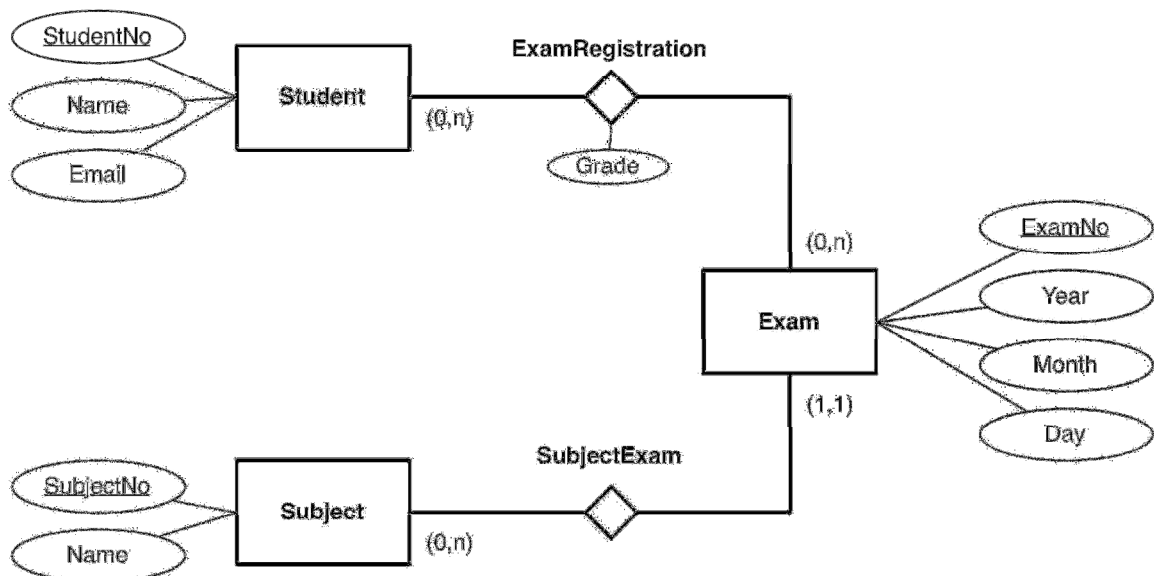
– StudentNo er fremmednøkkel mot Student-tabellen. StudentNo kan ikke ha NULL-verdi.

– ExamNo er fremmednøkkel mot Exam-tabellen. ExamNo kan ikke ha NULL-verdi.

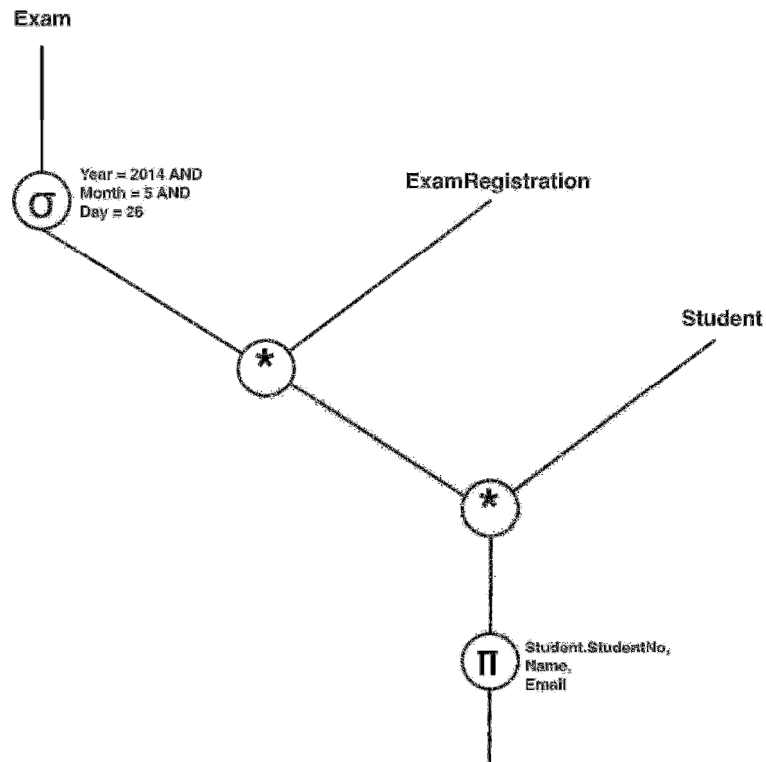
– Data legges inn ved oppmelding til eksamen. Grade settes til NULL inntil sensuren er klar.

Relasjonsalgebra kan formuleres som tekst eller grafer. Hvis du behersker begge notasjonene foretrekker vi at du svarer med grafer, men du blir ikke trukket for å svare med tekst.

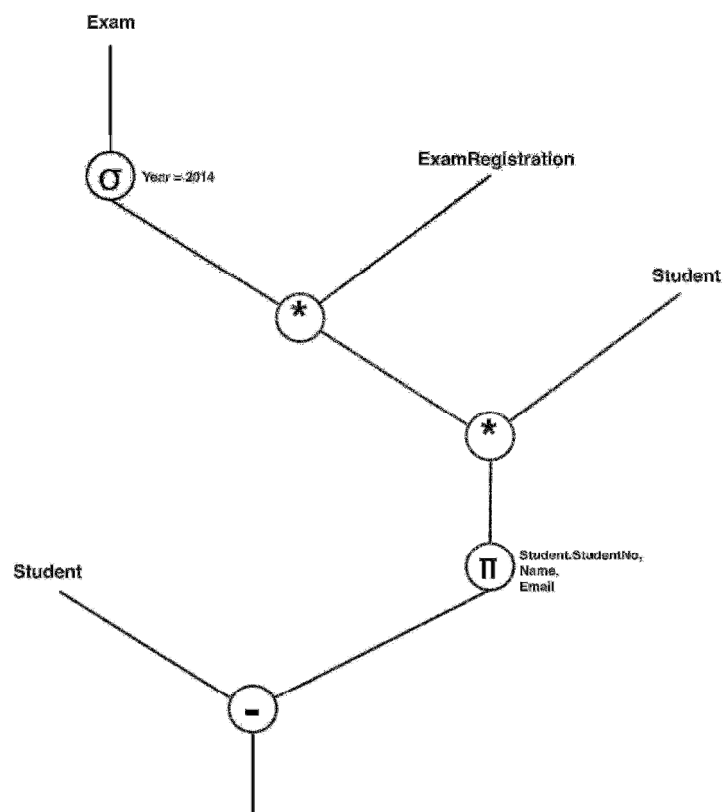
- a) (4 %) Lag et ER-diagram (du kan bruke alle virkemidler som er med i pensum) som i størst mulig grad samsvarer med relasjonsskjemaet. Gjør rede for eventuelle antagelser du finner det nødvendig å gjøre.



- b) (3 %) Lag en spørring i relasjonsalgebra som finner StudentNo, Name og Email for alle studenter som var oppmeldt til en av de eksamenene som ble arrangert 26. mai 2014.



- c) (3 %) Lag en spørring i relasjonsalgebra som finner alle studenter som ikke er oppmeldt til noen eksamen i 2014. I resultatet skal du ta med StudentNo, Name og Email.



- d) (3%) Lag en spørring i SQL som finner alle emner som er registrert med minst en eksamen i 2014. Resultatet skal bestå av SubjectNo og Name, det skal ikke være duplikater i resultatet, og resultattabellen skal være sortert etter SubjectNo i stigende rekkefølge.

```
SELECT DISTINCT SubjectNo, Name
FROM Subject NATURAL JOIN Exam
WHERE Year = 2014
ORDER BY SubjectNo ASC
```

- e) (3 %) Lag en spørring i SQL som finner antall ganger karakteren A har vært gitt siden emnet ble opprettet, i hvert emne som har hatt en eller flere eksamener i løpet av 2014. Resultattabellen skal ha kolonnene SubjectNo, Name og "Number of A". Resultatet skal sorteres etter "Number of A" i synkende rekkefølge.

```
SELECT SubjectNo, Name, COUNT(Grade) AS 'Number of A'
FROM Subject NATURAL JOIN Exam NATURAL JOIN ExamRegistration
WHERE Grade = 'A'
  AND SubjectNo IN (SELECT SubjectNo
                    FROM Exam
                    WHERE Year = 2014)
GROUP BY SubjectNo, Name
ORDER BY COUNT(Grade) DESC
```

- f) (4 %) Anta at alle fremmednøkklene er definert med restriksjonen "ON DELETE RESTRICT" eller "ON DELETE NO ACTION" (disse er likeverdige). Vi ønsker å slette studenten som har StudentNo 100. Vis hvordan du vil gjøre dette i SQL.

Dersom studenten har rader i ExamRegistration vil systemet nekte å slette studenten på grunn av restriksjonene som er satt på fremmednøkkelen StudentNo i ExamRegistration. For å slette studenten må vi først slette evt. rader i StudentRegistration som gjelder denne studenten, så kan vi slette studenten:

```
DELETE FROM ExamRegistration WHERE StudentNo = 100;
DELETE FROM Student WHERE StudentNo = 100;
```

Oppgave 3 – Teori (20 %)

- a) (4 %) Gitt $R = \{A, B, C, D, E\}$ og $F = \{AB \rightarrow C, C \rightarrow E, AE \rightarrow B\}$. Tillukningen $AB^+ = ABCE$. Forklar hva dette betyr. Finn alle kandidatnøkler (eng: candidate key) for tabellen R. Svaret må begrunnes.

$AB^+ = ABCD$ betyr at ABCD er funksjonelt avhengig av AB, dvs. at $AB \rightarrow ABCD$ vil finnes i mengden av alle funksjonelle avhengigheter som gjelder, F^+ .

En kandidatnøkkel er en minimal (ingen overflødige attributter) supernøkkel (identifikator) for en tabell. Supernøkler vil ha egenskapen at tillukningen til attributtene i supernøkkel er alle attributtene i den aktuelle tabellen.

A og D er ikke med på noen høyresider i funksjonelle avhengigheter, de må derfor være med i alle supernøkler. $AD^+ = AD$. $AD^+ \subset R$, slik at AD alene ikke er en supernøkkel. Alle supernøkler må dermed ha 3 eller flere attributter.

Kandidatnøkler:

- **ABD** – $ABD^+ = ABCDE = R$ og ingen delmengde av ABD er en supernøkkel.
- **ACD** – $ACD^+ = ABCDE = R$ og ingen delmengde av ACD er en supernøkkel.
- **ADE** – $ADE^+ = ABCDE = R$ og ingen delmengde av ADE er en supernøkkel.

Det er ingen supernøkler med 4 eller 5 attributter som vil være minimale.

- b) (6 %) Tabellen $R = \{A, B, C, D\}$ dekomponeres i $R_1 = \{A, C\}$ og $R_2 = \{B, C, D\}$. Finn en mengde funksjonelle avhengigheter (eng: functional dependencies), F , som gjør at denne dekomponeringen har tapsløs-join-egenskapen (eng: lossless join property). Med utgangspunkt i din F , skal du finne alle kandidatnøkler og høyeste normalform for tabellene R , R_1 og R_2 . Alle svarene må begrunnes.

En dekomponering har tapsløs-join-egenskapen hvis komponenttabellenes felles attributter er en supernøkkel for en eller begge komponenttabellene.

I dette tilfellet har vi: $R_1 \cap R_2 = C$. Det enkleste er å gjøre C til supernøkkel i R_1 ved å innføre $C \rightarrow A$. En mulig løsning blir da $F = \{C \rightarrow A\}$.

Når $F = \{C \rightarrow A\}$ vil følgende gjelde mht. kandidatnøkler:

- BCD er kandidatnøkkel i R . Det er den eneste supernøkkel som er minimal.
- C er kandidatnøkkel i R_1 siden C er en minimal supernøkkel. Det er den eneste kandidatnøkkel siden A ikke er en supernøkkel og AC ikke er en minimal supernøkkel.
- BCD er kandidatnøkkel i R_2 siden det ikke er noen funksjonelle avhengigheter som gjelder i R_2 . Dette vil derfor være den eneste supernøkkel.

Med hensyn til normalform vil vi ha følgende:

- I R har vi en delvis avhengighet, $C \rightarrow A$, fra kandidatnøkkel (BCD) til et ikke-nøkkel-attributt (A). R er derfor ikke på 2NF, og 1NF vil være den høyeste normalformen (vi antar at den gjelder).
- R_1 er på BCNF (4NF) siden venstresiden (C) i den eneste avhengigheten i F_1 , $C \rightarrow A$, er en supernøkkel.
- R_2 er på BCNF (4NF) siden det ikke finnes funksjonelle avhengigheter som gjelder i tabellen, $F_2 = \emptyset$.

- c) (4 %) Gitt $R = \{A, B, C, D, E\}$ og $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow E\}$. Gå ut fra at R oppfyller 1. normalform. Bestem den høyeste normalformen som oppfylles av R . Svaret må begrunnes.

A er den eneste kandidatnøkkel i R siden alle andre supernøkler må inneholde A .

Det er ikke mulig å ha en delvis avhengighet når kandidatnøkkel består av bare ett attributt, slik at 2NF vil være oppfylt. Det er mange avhengigheter blant ikke-nøkkel-attributtene, for eksempel $B \rightarrow C$, slik at 3NF ikke er oppfylt. Den høyeste normalformen er 2NF.

d) (6 %) Gitt $R = \{A, B, C, D, E\}$ og $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow E\}$. Gå ut fra at R oppfyller 1. normalform. R skal om nødvendig dekomponeres slik at alle komponenter er på Boyce-Codd normalform (BCNF). Dekomponeringen skal ha tapsløs-join-egenskapen, attributtbevaring og bevaring av funksjonelle avhengigheter. Du må forklare hvordan din løsning oppfyller alle kravene.

Dekomponering:

- $R_1 = \{A, B\}$, $F_1 = \{A \rightarrow B\}$. BCNF siden A er en supernøkkel i R_1 .
- $R_2 = \{B, C\}$, $F_2 = \{B \rightarrow C\}$. BCNF siden B er en supernøkkel i R_2 .
- $R_3 = \{C, D\}$, $F_3 = \{C \rightarrow D\}$. BCNF siden C er en supernøkkel i R_3 .
- $R_4 = \{D, E\}$, $F_4 = \{D \rightarrow E\}$. BCNF siden D er en supernøkkel i R_4 .

Alle (komponent-)tabellene er på BCNF.

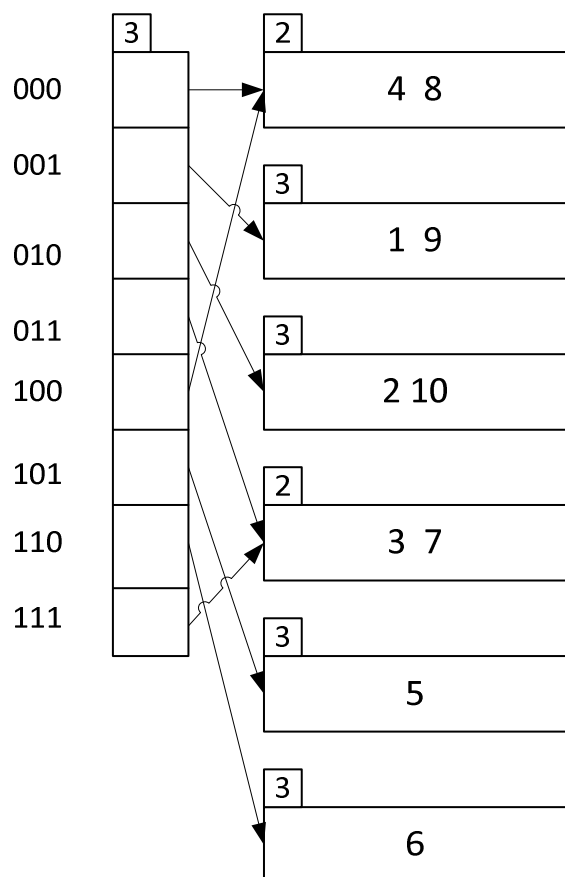
Vi har attributtbevaring fordi $R_1 \cup R_2 \cup R_3 \cup R_4 = R$.

Vi har bevaring av funksjonelle avhengigheter fordi $F_1 \cup F_2 \cup F_3 \cup F_4 = F$.

Dekomponeringen har tapsløs-join-egenskapen fordi vi kan joine sammen til utgangspunktet (R), for eksempel: $((R_1 * R_2) * R_3) * R_4$, på en slik måte at operandtabellene i hver join har felles attributter som er supernøkkel i en eller begge operandtabellene.

Alternativt kan vi bruke *tabellmetoden* for å vise at dekomponeringen er tapsløs.

Oppgave 4 – Extendible hashing (10 %)



Oppgave 5 – Lagring og queryutføring (15 %)

- i) Ved kun innsetting er en **heapfil** helt suveren.
- ii) Her henter vi ut mange felter basert på kun *startyear*. Enten en **heapfil** eller et **unclustered B+-tre** (*startyear*) og **heapfil**. *Startyear* er neppe så selektiv at det lønner seg med indeksering.
- iii) En **heapfil** er helt grei her da alle poster hentes ut.
- iv) **Heapfil + unclustered B+-tre** (*lastname*). Her trenger vi kun å lese indeksen som allerede er sortert (*index-only query*).
- v) Her er det lurt med en **clustered indeks**, enten hash eller B+-tre på *empno*.

Oppgave 6 – Transaksjoner - SERIALIZABLE (5 %)

Tabell 20.1 i Elmasri & Navathe: SQLs SERIALIZABLE unngår

- **Dirty Read**
- **Nonrepeatable read**
- **Phantom.**

Oppgave 7 – Transaksjoner – historie og låser (10 %)

- a) Problemet med denne sekvensen er «**Lost update**». Begge transaksjonene leser X i samme tilstand og begge oppdaterer X og så skrives X tilbake til databasen. I dette tilfellet mistes T1s oppdatering av X.
- b) Hvis transaksjonene setter låser etter behov (basic , strict og rigorous 2PL): Begge transaksjonene ville satt en leselås på X, så vil T1 prøve å sette en skrivelås (oppgradering), men får ikke lov fordi T2 også har en leselås. Så ville T2 prøvd å sette en skrivelås (oppgradering), men får ikke lov til å gjøre dette. Vi får en vranglås og (en av) transaksjonene vil sannsynligvis rulles tilbake og prøve å kjøre på nytt.

Alternativt kunne vi si at man bruker konservativ 2PL og alle låser må settes på forhånd. Da ville de to transaksjonene kunne kjøre serielt (etter hverandre).