Prøveeksamen IN2090 Høsten 2020

Tid: 4 timer

Tillatte hjelpemider: Alle hjelpemidler tillatt

Prøveeksamen består av 3 deler (maksimal peongsum i parentes):

1. Modellering (40)
2. SQL (40)
3. Relasjonsmodellen og dekomponering (20)

1 Modellering (40)

Lag en ER-modell for databasen til et nettsted (kalt FileShare) som har som formål å gjøre det mulig for brukere å lagre, konvertere og dele filer.

- Du må gjerne dele modellen opp i deler (f.eks. over flere sider), men sørg for at det fremkommer tydelig hvilke deler av modellen som besvarer hvilke oppgaver.
- Du må gjerne inkludere kommentarer i modellene. Dersom det er uklarheter eller tvetydigheter i oppgavebeskrivelsen, bruk sunn fornuft og skriv en kommentar til modellene dine om hvilke antagelser og tolkninger du eventuelt gjør.
- Dersom du ser at det er mulig å modellere deler av oppgaven på flere ulike måter, diskuter kort hvordan din fremgangsmåte er i forhold til de ulike alternativene.
- Tegn og skriv tydelig.

1 Brukere (7)

Lag en ER-modell som fanger opp informasjon om brukere og brukergrupper på FileShare: En viktig funksjon for nettstedet FileShare er å kunne holde orden på informasjon om brukerne som er registrert på nettsiden. For enkelhets skyld identifiserer FileShare sine brukere internt med unike IDer (f.eks. `324`). Informasjon som kreves av brukere av FileShare er e-postadresse, passord (kryptert, selvfølgelig), og tidspunktet da brukeren registrerte seg på FileShare. FileShare tillater ikke brukere å registrere mer enn én e-postadresse, og den samme adressen kan ikke benyttes av flere brukere. I tillegg kan brukere legge inn navn (bestående av for- og etternavn), stedet der brukeren bor, samt brukerens telefonnumre (potensielt fler nummere per bruker).

Brukere organiseres i brukergrupper som er tilknyttet organisasjoner. FileShare lagrer navnet på organisasjonen brukergruppe er tilknyttet, samt organisasjonens nettside. Organisasjoner har et unikt navn. Hver bruker er tilknyttet én eller fler organisasjoner, og en organisasjon kan ha mange brukere.

Løsningsforslag

Se figur 1.

2 Filer og mapper (13)

Utvid modellen med informasjon om filer og mapper og hvordan de lagres på FileShare: En av kjernefunksjonene til FileShare er å lagre informasjon om filer lastet opp av brukere på FileShare. Hver bruker kan opprette én eller fler mapper, men en mappe er opprettet av nøyaktig én bruker. Hver mappe har en unik sti (f.eks. `/home/jon/xls/`). En mappe kan så inneholde potensielt mange filer.

Filer kan lastes opp av brukere, og blir lagret i filsystemet til FileShare. Filene identifiseres internt av en unik kombinasjon av mappen der filen er lagret i filsystemet og filens navn (f.eks. `lønn.xls`). FileShare vil lagre informasjon om hvilke brukere som lastet opp hvilke filer, samt når hver fil ble lastet opp til FileShare. I tillegg til navn og lokasjon må FileShare for filer også lagre informasjon om språket til filen. F.eks. `test.pdf` er et dokument skrevet på engelsk. For eksekverbare filer vil språket være programmeringsspråket filene ble skrevet i (f.eks. `run.exe` ble skrevet i C). En fil kan bli lastet opp av nøyaktig én bruker, men en bruker kan laste opp mange filer.

Brukere kan også endre filer, og FilShare ønsker å lagre når en bruker sist endret en fil (altså kun ett tidspunkt per kombinasjon av bruker og fil). Hver bruker kan endre mange filer, og hver fil kan endres av mange brukere.

Løsningsforslag

Se figur 1.

Tidspunktet knyttet til `LASTET_OPP` kunne også vært flyttet til `FIL`. Realiseringen av de to modellene blir derimot helt lik.

3 Operasjoner (10)

Støtte for å utføre operasjoner på de lagrede filene er en annen viktig funksjon hos FileShare. En operasjon har et unikt navn (f.eks. `FlyttFil` eller `KjørFil`). Operasjoner krever rettigheter for å kunne kjøre. Men for å gjøre det enkelt har FileShare laget nøyaktig én rettighet per operasjon. Altså krever hver operasjon næyaktig én rettigheter og en rettighet er påkrevd for nøyaktig én operasjon. Rettigheter er identifisert med en kode, og kan ha en eller fler navn, f.eks. `Administrator` eller `Eier`.

Brukere kan ha ingen eller fler rettigheter, og hver rettighet kan naturligvis være gitt til mange brukere.

Brukere kan utføre operasjoner på filer. F.eks. “Brukeren `324` utførte `KjørFil`-operasjonen på filen `run.exe` (i mappen `/home/jon/xls`)”. En bruker kan utføre

mange operasjoner på samme fil; en operasjon kan utføres på mange filer av en bruker; og en operasjon kan utføres av mange brukere på samme fil. Ettersom opprettelsen av en fil også blir lagret som en operasjon vil alle filer være del av minst én operasjonsutførelse.

Løsningsforslag

Se figur 1.

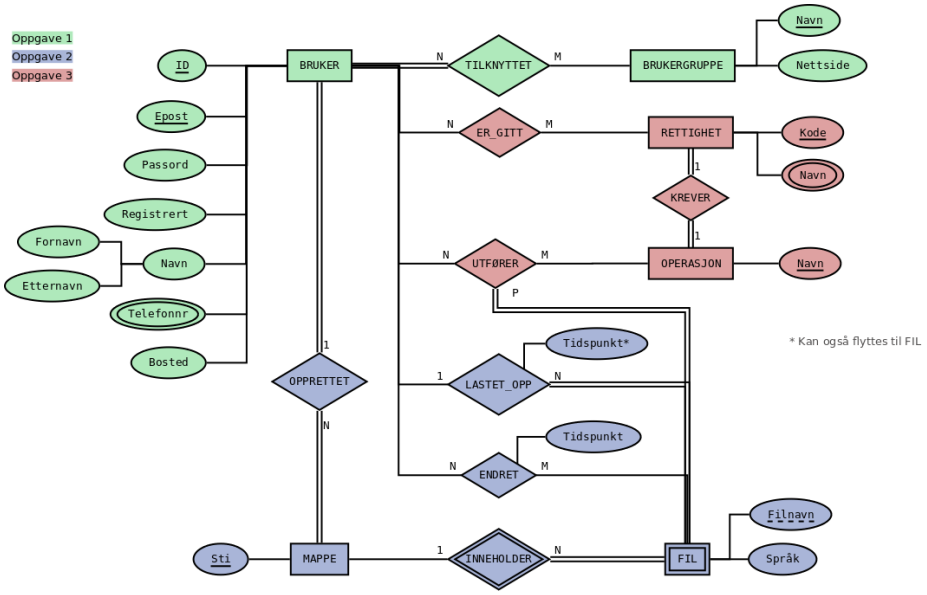

Figure 1: Løsningsforslag modellering

4 Realisering (10)

Realiser ER-modellen i figur 2 til et relasjonsdatabaseskjema. Relasjonsdatabaseskjemaet skal være korrekt (tilsvare ER-modellen), effektivt (unngå redundans og begrens antallet tabeller), og tydelig (lett å forstå). Bruk realiseringsalgoritmen for å danne et slikt relasjonsdatabaseskjema. For hver relasjon, spesifiser relasjonens navn og navnet til hvert attributt. Du skal ikke spesifisere datatyper/domener for attributtene, og ikke benytte SQL i denne oppgaven. Marker primærnøkler med én understrek. Dersom en relasjon har flere kandidatnøkler, marker alle kandidatnøkler med én strek, og primærnøkkelen med fet skrift i tillegg. Skriv fremmednøkler enten med ord ($T(A)$ referer til $S(B)$) eller på formen $T(A) \rightarrow S(B)$ for å indikere at relasjon T sin A-attributt er en fremmednøkkel som peker på relasjon S sin B-attributt.

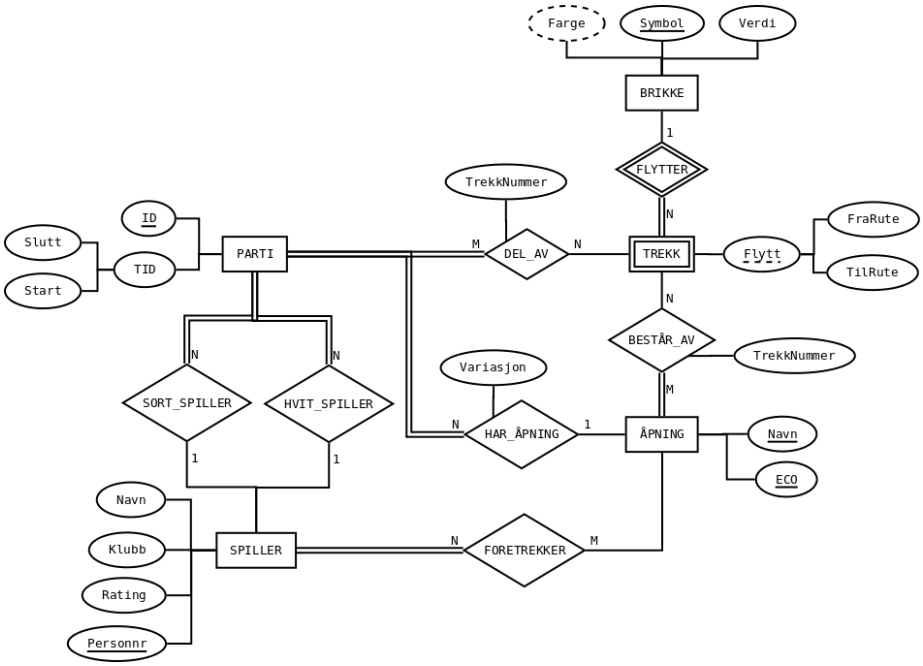

Figure 2: ER-modell

Løsningsforslag

1. Starter med å realisere normale entiteter:
   - $Parti(\underline{ID}, Start, Slutt)$
   - $Spiller(Navn, Klubb, Rating, \underline{Personnr})$
   - $Brikke(\underline{Symbol}, Verdi)$
   - $Åpning(\underline{Navn}, \underline{\mathbf{ECO}})$
2. Deretter realiserer vi svake entiteter:
   - $Trekk(\underline{FraRute, TilRute, Symbol})$ hvor vi har fremmednøkkelen $Trekk(\underline{Symbol}) \rightarrow Brikke(Symbol)$
3. Så realiserer vi relasjoner. Har ingen 1-1, så fortsetter med 1-N:
   - Velger å realisere både `SORT_SPILLER` og `HVIT_SPILLER` som attributter til $Parti$, siden disse er nødvendige for alle parti: $Parti(\underline{ID}, Start, Slutt, HvitSpiller, SortSpiller)$ hvor
     - $Parti(HvitSpiller) \rightarrow Spiller(Personnr)$
     - $Parti(SortSpiller) \rightarrow Spiller(Personnr)$.
   - Velger å realisere `HAR_ÅPNING` som attributt til $Parti$, siden hvert parti har nøyaktig én åpning: $Parti(\underline{ID}, Start, Slutt, HvitSpiller, SortSpiller, Åpning, Variasjon)$ hvor

- $Parti(\text{Åpning}) \rightarrow \text{Åpning}(ECO)$

4. Realiserer så N-M-relasjoner:
   - $DelAv(\underline{Parti, FraRute, TilRute, Symbol}, TrekkNummer)$ hvor
     - $DelAv(Parti) \rightarrow Parti(ID)$
     - $DelAv(FraRute, TilRute, Symbol) \rightarrow Trekk(FraRute, TilRute, Symbol)$
   - $Foretrekker(\underline{Spiller, \text{Åpning}})$ hvor
     - $Foretrekker(Spiller) \rightarrow Spiller(personnr)$
     - $Foretrekker(\text{Åpning}) \rightarrow \text{Åpning}(ECO)$
   - $BestårAv(\underline{\text{Åpning}, FraRute, TilRute, Symbol}, TrekkNummer)$ hvor
     - $BestårAv(\text{Åpning}) \rightarrow \text{Åpning}(ECO)$
     - $BestårAv(FraRute, TilRute, Symol) \rightarrow Trekk(FraRute, TilRute, Symol)$

Det er ingen multi-verdi attributter å realisere til slutt. Vi får da følgende endelig databaseskjema:

- $Parti(\underline{ID}, Start, Slutt, HvitSpiller, SortSpiller, \text{Åpning}, Variasjon)$
- $Spiller(Navn, Klubb, Rating, \underline{Personnr})$
- $Brikke(\underline{Symbol}, Verdi)$
- $\text{Åpning}(\underline{Navn}, \mathbf{ECO})$
- $DelAv(\underline{Parti, FraRute, TilRute, Symbol}, TrekkNummer)$
- $Foretrekker(\underline{Spiller, \text{Åpning}})$
- $BestårAv(\underline{\text{Åpning}, FraRute, TilRute, Symbol}, TrekkNummer)$

med fremmednøkler:

- $Parti(HvitSpiller) \rightarrow Spiller(Personnr)$
- $Parti(SortSpiller) \rightarrow Spiller(Personnr).$
- $DelAv(Parti) \rightarrow Parti(ID)$
- $DelAv(FraRute, TilRute, Symbol) \rightarrow Trekk(FraRute, TilRute, Symbol)$
- $Parti(\text{Åpning}) \rightarrow \text{Åpning}(ECO)$
- $Foretrekker(Spiller) \rightarrow Spiller(personnr)$
- $Foretrekker(\text{Åpning}) \rightarrow \text{Åpning}(ECO)$
- $BestårAv(\text{Åpning}) \rightarrow \text{Åpning}(ECO)$
- $BestårAv(FraRute, TilRute, Symol) \rightarrow Trekk(FraRute, TilRute, Symol)$

2 SQL (40)

Du er ansatt som ny databaseadministrator hos Julenissens Verksted, hvor det er mye som skal gjøres før jul. Databasen til Julenissen er laget av følgende SQL-script:

```
CREATE TABLE barn(
    bid int PRIMARY KEY,
    navn text NOT NULL,
    snill boolean NOT NULL
```

```
);

CREATE TABLE gave(
    gid int PRIMARY KEY,
    navn text NOT NULL,
    nyttig boolean NOT NULL
);

CREATE TABLE ønskeliste(
    barn int REFERENCES barn(bid),
    gave int REFERENCES gave(gid)
);
```

Tabellen `barn` inneholder informasjon om alle barn, hvor hvert barn har en unik ID (`bid`), et navn og en boolsk verdi som er sann om barnet har vært snill det siste året og usann hvis ikke.

Tabellen `gave` inneholder informasjon om gavene som Julenissens Verksted kan lage. Hver gave har en unik ID (`gid`), et navn samt en boolsk verdi som sier om gaven er en nyttig gave eller ikke.

Den siste tabellen, `ønskeliste`, inneholder alle barns ønsker, hvor hver rad inneholder et barns ID sammen med IDen til en gave det barnet ønsker seg. Merk: Et barn kan ønske seg mange gaver og en gave kan naturligvis bli ønsket av mange barn.

Under er et eksempel på hvordan databasen *kan* se ut.

barn

bid	navn	snill
0	Ola	t
1	Kari	t
2	Per	f
3	Nils	t
4	Mari	f
5	Kine	f
6	Jasmin	t

gave

gid	navn	nyttig
0	Lekebil	f
1	Sokker	t
2	Sykkel	t
3	Lekepistol	f

```
  4 | Dataspill  | f
  5 | Fuglebrett | t
  6 | Dukke      | f
  7 | Bok        | t
```

ønskeliste

```
 barn | gave
------+------
    1 |    0
    1 |    6
    0 |    6
    0 |    5
    3 |    2
    5 |    1
    5 |    3
    5 |    4
    4 |    3
    4 |    0
    6 |    7
```

5 Snille barn (5)

Skriv en SQL-kommando som oppdaterer barnet med `bid` lik `0` sin `snill`-verdi til `false`.

Løsningsforslag

```sql
UPDATE barn
SET snill = false
WHERE bid = 0;
```

6 Nyttige gaver (5)

Skriv en spørring som finner alle barn som ønsker seg nyttige gaver som har navn som starter med strengen `'Sokker'`. Spørringen skal returnere navnet på barnet og navnet på gaven.

Løsningsforslag

```sql
SELECT b.navn AS barn, g.navn AS gave
FROM barn AS b
     INNER JOIN ønskeliste AS ø ON (b.bid = ø.barn)
     INNER JOIN gave AS ø ON (ø.gave = g.gid)
WHERE g.nyttig AND g.navn LIKE 'Sokker%';
```

7 Oversikts-VIEW (5)

Skriv en SQL-kommando som lager et `VIEW` som heter `oversikt` og inneholder én rad for hvert ønske med `bid` og navnet på barnet som har ønsket, `gid` og navnet på gaven som er ønsket, hvorvidt barnet har vært snill, og hvorvidt gaven er nyttig. `VIEW`et skal være sortert først på barnets navn, og så på gavens navn i alfabetisk rekkefølge.

Innholdet i `VIEW`et kan f.eks. se slik ut:

```
 bid |  barn  | gid |    gave    | snill | nyttig
-----+--------+-----+------------+-------+--------
   6 | Jasmin |   7 | Bok        | t     | t
   1 | Kari   |   6 | Dukke      | t     | f
   1 | Kari   |   0 | Lekebil    | t     | f
   5 | Kine   |   4 | Dataspill  | f     | f
   5 | Kine   |   3 | Lekepistol | f     | f
   5 | Kine   |   1 | Sokker     | f     | t
   4 | Mari   |   0 | Lekebil    | f     | f
   4 | Mari   |   3 | Lekepistol | f     | f
   3 | Nils   |   2 | Sykkel     | t     | t
   0 | Ola    |   6 | Dukke      | t     | f
   0 | Ola    |   5 | Fuglebrett | t     | t
```

Løsningsforslag

```
CREATE VIEW oversikt AS
SELECT b.bid, b.navn AS barn, g.gid, g.navn AS gave, b.snill, g.nyttig
FROM barn AS b
     INNER JOIN ønskeliste AS ø ON (b.bid = ø.barn)
     INNER JOIN gave AS g ON (ø.gave = g.gid)
ORDER BY barn, gave;
```

8 Like ønkser (5)

Skriv en SQL-spørring som finner alle par av *ulike* barn som har ønsket seg det samme. Svaret skal kun inneholde unike rader. Du kan benytte `VIEW`et fra oppgave 7 om du ønsker.

Løsningsforslag

```
WITH
  likt_ønske AS (
    SELECT b1.barn AS barn1, b2.barn AS barn2
    FROM ønskeliste AS b1
         INNER JOIN ønskeliste AS b2 USING (gave)
    WHERE b1.barn != b2.barn
  )
```

```sql
SELECT DISTINCT b1.navn, b2.navn
FROM likt_ønske AS l
     INNER JOIN barn AS b1 ON (l.barn1 = b1.bid)
     INNER JOIN barn AS b2 ON (l.barn2 = b2.bid);
```

eller med `VIEW`et fra oppgave 7:

```sql
SELECT DISTINCT b1.barn, b2.barn
FROM oversikt AS b1
     INNER JOIN oversikt AS b2 USING (gid)
WHERE b1.bid != b2.bid;
```

9 Populære gaver (10)

Skriv en spørring som finner de tre mest populære nyttige gavene, og de tre mest populære unyttige gavene. Resultatet skal inneholde navn på gaven, antall barn som ønsker gaven, samt hvorvidt den er nyttig eller ikke. Du kan benytte view'et fra oppgave 7.

Løsningsforslag

```sql
SELECT gave, count(*) AS antall_ønsker, true AS nyttig
FROM oversikt
WHERE nyttig
GROUP BY gave
ORDER BY antall_ønsker
LIMIT 3
UNION
SELECT gave, count(*) as antall_ønsker, false AS nyttig
FROM oversikt
WHERE NOT nyttig
GROUP BY gave
ORDER BY antall_ønsker
LIMIT 3;
```

10 Gaveliste (10)

Skriv en spørring som tilordner gaver til barn i henhold til følgende regler:

- Snille barn får alt de ønsker seg
- Usnille får kun nyttige ting de ønsker seg. Om de ikke ønsker seg noen nyttige ting får de gaven med navn `'Genser'`.

Spørringen skal skrive ut navnet på barnet og navnet på gaven. Merk, du kan bruke view'et fra oppgave 7.

Løsningsforslag

```
WITH
  usnille_med_ønsker AS (
    SELECT bid, barn, gave
    FROM oversikt
    WHERE snill = false AND
          nyttig = true
  ),
  usnille_uten_ønkser AS (
    SELECT navn AS barn, 'Genser' AS gave
    FROM barn
    WHERE snill = false AND
          bid NOT IN (SELECT bid FROM usnille_med_ønsker)
  )
SELECT barn, gave
FROM oversikt
WHERE snill = true
UNION ALL
SELECT barn, gave
FROM usnille_med_ønsker
UNION ALL
SELECT barn, gave
FROM usnille_uten_ønsker;
```

3 Relasjonsmodellen og dekomponering (20)

11 Relasjonsalgebra (5)

I denne oppgaven skal du bruke samme databaseskjame som oppgavene i SQL.

Sriv et uttrykk i relasjonsalgebraen som finner navn på alle barn som ønsker seg `hoppestokk`.

Løsningsforslag

$\pi_{\text{navn}}(\text{barn} \bowtie_{\text{bid}} \text{ønskeliste} \bowtie_{\text{gid}} \pi_{\text{gid}}(\sigma_{\text{navn='hoppestokk'}}(\text{gave})))$

eller

$\pi_{\text{navn}}(\text{barn} \bowtie_{\text{bid}} \text{ønskeliste} \bowtie_{\text{gid}} \rho_{\text{navn}\to\text{gavenavn}}(\sigma_{\text{navn='hoppestokk'}}(\text{gave})))$

12 Nøkler (5)

Gitt følgende relasjon

$R(A, B, C, D, E)$

med FDene

1. $A \rightarrow B$
2. $BC \rightarrow D$
3. $DE \rightarrow A$

Hvilke kandidatnøkler har R? Vis hvordan du kommer frem til svaret.

Løsningsforslag

Attributter aldri på høyreside: CE

Attributter kun på høyresider: ingen

Alle kandidatnøkler må altså ha med CE, og vi må potensielt utvide med de andre.

Sjekker først om CE er en kandidatnøkkel: $CE^+ = CE$, altså ikke en kandidatnøkkel.

- Utvider med A: $ACE^+ = ACEBD$, altså er ACE en kandidatnøkkel.
- Utvider med B: $BCE^+ = BCEDA$, altså er BCE en kandidatnøkkel.
- Utvider med D: $CDE^+ = CDEAB$, altså er CDE en kandidatnøkkel.

Siden både ACE, BCE og CDE er kandidatnøkler kan vi ikke utvide noen av dem, siden vi da ikke lenger får en minimal nøkkel. Altså er ACE, BCE og CDE de eneste kandidatnøklene til R.

13 Tapsfri dekomponering (10)

Gitt følgende relasjon

$R(A, B, C, D, E, F)$

med kandidatnøkler AB og CD, og FDer:

1. $AB \rightarrow C$
2. $AB \rightarrow D$
3. $CD \rightarrow A$
4. $CD \rightarrow B$
5. $A \rightarrow E$
6. $C \rightarrow F$

Dekomponer relasjonen tapsfritt til BCNF. Vis hvordan du kommer frem til svaret. For hver relasjon du får underveis i dekomponeringen, list opp hvilke FDer som holder, og hvilke kandidatnøkler relasjonen har.

Løsningsforslag

Vi går igjennom hver FD og sjekker om de bryter med BCNF:

1. $AB \rightarrow C$: AB er supernøkkel så bryter ikke med BCNF.
2. $AB \rightarrow D$: AB er supernøkkel så bryter ikke med BCNF.
3. $CD \rightarrow A$: CD er supernøkkel så bryter ikke med BCNF.

4. $CD \to B$: CD er supernøkkel så bryter ikke med BCNF.
5. $A \to E$: A er ikke en supernøkkel så bryter med BCNF.

- Dekomponerer R til $S_1(A, E)$ og $S_2(A, B, C, D, F)$
- For S_1 holder kun FDen $A \to E$ og har dermed kandidatnøkkel A, og er dermed på BCNF.
- For S_2 holder alle de andre FDene og får dermed kandidatnøkler AB og CD. Må derfor (muligens) dekomponere S_2 videre.

6. Sjekker defor S_2 videre med $C \to F$: C er for S_2 ikke en supernøkkel, må derfor dekomponere S_2 videre.

- Dekomponerer S_2 til $S_{21}(C, F)$ og $S_{22}(C, A, B, D)$.
- For S_{21} holder kun $C \to F$ og har da kandidatnøkkel C og er derfor på BCNF.
- For S_{22} holder FDene 1. til 4., og har derfor kandidatnøkler AB og CD. Av samme grunn som over bryter ingen av disse BCNF, og $S_{22} er på BCNF.

R kan altså dekomponeres tapsfritt til $S_1(A, E)$, $S_{21}(C, F)$ og $S_{22}(A, B, C, D)$.