Eksamen IN2090 Høsten 2022

Skriftlig eksamen i IN2090

2022 HØST

Tid: 7. desember 09:00 til 7. desember 13:00.

Sted: Silurveien

Tillatte hjelpemidler: Alle trykte og skrevne hjelpemidler tillatt.

Det er viktig at du leser denne forsiden nøye før du starter.

Håndtegning:

På denne eksamen er det oppgaver som krever håndtegning. Du bruker skisseark du får utdelt. Det er anledning til å bruke flere ark per oppgave. Se instruksjon for utfylling av skisseark i lenken under oppgavelinjen.

Struktur:

Eksamen består av 3 deler (maksimal poengsum i parentes):

- Modellering og realisering (30)
- SQL (45)
- Relasjonsmodellen, normalformer og dekomposisjon (25)

1 Modellering (30)

1.1 ER: Oppskrifter (15)

Du er ansatt i et firma som skal lage en nettside hvor brukere skal kunne lage og dele matoppskrifter. Du er derfor blitt bedt om å lage en ER-modell for databasen som skal lagre brukere, oppskrifter, ingredienser, osv. Etter diskusjon med skaperne bak nettsiden, har dere kommet frem til at følgende vil gjelde:

- Brukere indentifiseres med sitt brukernavn. Ellers har brukere et navn, et passord, samt en dato for når de ble medlem.
- Brukere kan legge inn oppskrifter. En oppskrift identifiseres med en egen ID, og har ellers et navn, og en mengde søkbare nøkkelord (f.eks. `lørdagskos`, `restemiddag`, osv.). En oppskrift kan altså ha flere slike nøkkelord knyttet til seg.

- En bruker kan legge inn mange oppskrifter, men en oppskrift må være lagt inn av nøyaktig én bruker. En bruker trenger ikke ha lagt inn noen oppskrifter.
- En oppskrift bruker ingredienser. Hver ingrediens har et unikt navn (f.eks. `sitron`, `pepper`). Ingredienser har også en beskrivelse (f.eks. `gul sitrusfrukt`, `vanlig krydder`, osv.).
- En oppskrift kan bruke mange ingredienser og en ingrediens kan brukes av mange oppskrifter. En oppskrift må derimot bruke minst én ingrediens. En oppskrift bruker også en bestem mengde av en ingrediens beskrevet som tekst (f.eks. `2dl`, `300g`, osv.), som igjen består av måleenhet (f.eks. `dl` eller `g`) og et antall (f.eks. `2` eller `300`).
- En oppskrift består også av steg som beskriver hvordan maten skal lages. Hvert steg har et nummer som kun er unikt for oppskriften den tilhører (altså kan en oppskrift kun ha ett steg med nummer `1`, men mange forskjellige oppskrifter kan ha steg med nummer `1`). Et steg har også en beskrivelse av hva som skal gjøres i det steget (f.eks. `kok opp vann`, `kutt sitronen i båter`, osv.).
- En oppskrift må ha minst ett steg, men kan ha mange steg.
- En ingrediens kan inneholde andre ingredienser (f.eks. `sitron` inneholder `sitronskall`), og dette skal også lagres. En ingrediens kan inneholde mange andre ingredienser, men en ingrediens kan kun være inneholdt i én ingrediens. En ingrediens hverken trenger å inneholde eller være inneholdt i noen ingrediens.
- Ettersom dette skal være en plattform for å dele oppskrifter, skal brukere også kunne kommentere på andres oppskrifter. En kommentar har et tidspunkt knyttet til seg, som er tidspunktet kommentaren ble lagt ut. Dette tidspunktet er unikt for kommentarer laget av samme bruker og på samme oppskrift (altså kan vi ha kommentarer med likt tidspunkt, men da med forskjellig brukere eller forskjellige oppskrifter). Ellers har kommentarer et innhold, samt en alder. Alderen kan derimot utledes fra kommentarens tidspunkt. En kommentar kan kun legges inn av én bruker, og kan kun tilhøre én oppskrift. En bruker kan derimot legge inn mange kommentarer, og en oppskrift kan også ha mange kommentarer knyttet til seg.

Lag derfor en ER-modell som inneholder informasjonen over.

- I denne oppgaven skal du svare med digital håndtegning. Bruk eget skisseark (utdelt). Se instruksjon for utfylling av skisseark i lenken under oppgavelinjen.
- Tegn og skriv tydelig.
- 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 deler av oppgaven.
- 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.

Løsningsforslag

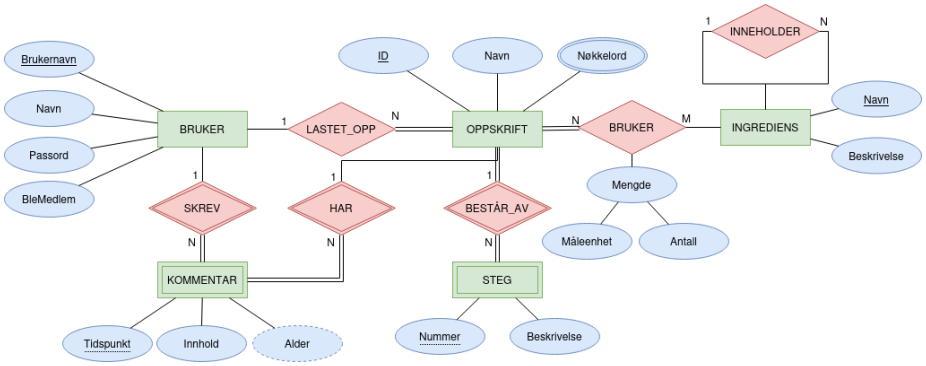

Figure 1: Oppskrift-modell

Sensorveiledning

- -0.5 for feil plassering av total deltakelse eller riktig kardinalitet, men feil plassering
- -1 for feil kardinalitet eller manglende total deltakelse
- -2 for manglende kardinalitet på hele relasjonen
- -1 for manglende attributt-markeringer (nøkkel eller multi-valued)
- -1 for manglende/feil plassering av vanlig attributt
- -2 for både manglende markering av svak entitet og identifiserende relasjon (-1 for kun å mangle én av dem)
- -1 for å bruke identifiserende 1:1:N-ternær-relasjon med total deltakelse fra svak KOMMENTAR for siste del

1.2 ER: Romvesner (5)

HUFF (Departementet for Hemmelige, Utenomjordiske og Farefulle Fenomener) har omsider fått kontakt med noen romvesner de er svært interessert i å kommunisere med. Romvesnene kommuniserer utelukkende med ER-diagrammer, og har sendt diagrammet under.

Hvilke av disse setningene er sanne i henhold til ER-diagrammet. Kryss av "JA" dersom du mener setningen er modellert av ER-diagrammet, og "NEI" hvis ikke.

Hvert riktig svar gir 1 poeng, hvert uriktige svar gir -1 poeng, og hvert ubesvart gir 0 poeng. Total poengsum for hele oppgaven vil ikke være lavere enn 0.

1. To forskjellige ZABONG kan ha samme Fy-verdi
2. To forskjellige FUFF kan være DING-relatert til samme ZABONG
3. En SHLEX er alltid relatert til minst én FUFF

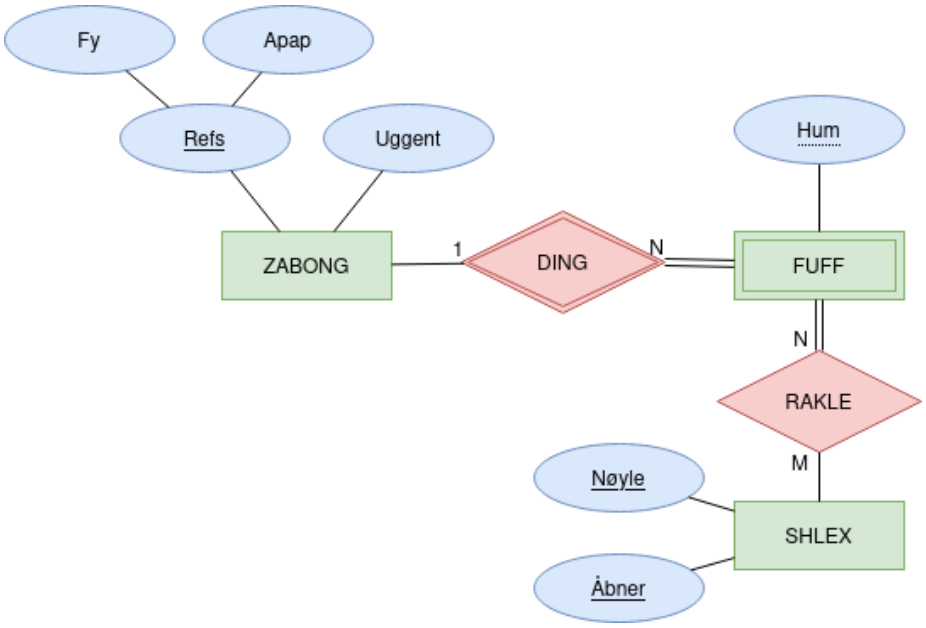

Figure 2: romvesen-modell

4. Alle SHLEX har både en Nøyle-verdi og en Åbner-verdi
5. Alle ZABONG har en Uggent-verdi

Løsningsforslag

1. Ja
2. Ja
3. Nei
4. Ja
5. Nei

Automatisk rettet.

1.3 Realisering: Nissens verksteder (10)

Nissen ønsker å profesjonalisere sin produksjon av leker, og ønsker derfor å lage en database over informasjon knyttet til sine mange verksteder. Nissen har derfor laget følgende ER-modell som beskriver verkstedene, alvene og selve produksjonen av leker:

Hjelp Nissen ved å realisere modellen til et relasjonelt databaseskjema. Det resulterende databaseskjemaet skal være korrekt (samsvare med modellen), effektivt (unngå overflødige tabeller og kolonner med mange `NULL`-verdier), og tydelig (bør være lett å forstå). Bruk algoritmen for realisering for å lage et

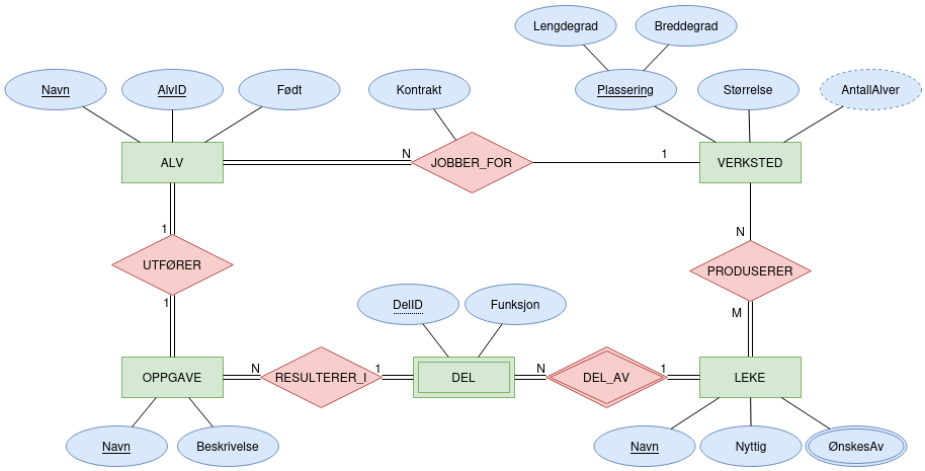

Figure 3: ER-modell over Nissens verksteder

slikt skjema, og forklar eventuelle valg du tar underveis. For hver relasjon, oppgi relasjonens navn, og navnet til hvert attributt. Du trenger ikke oppgi datatyper for attributtene, og du skal ikke bruke SQL i denne oppgaven. Gjør det tydelig hvilke attributter som utgjør kandidatnøkler og fremmenøkler. Dersom det er flere kandidatnøkler, marker også valgt primærnøkkel. F.eks. bruk understreking for å markere kandidatnøkler, fet skrift for primærnøkler, og piler for å markere fremmednøkler (f.eks. "T(A) -> S(B)" for å uttrykke at relasjon T sin attributt A er en fremmednøkkel som peker på relasjon S sin B attributt).

Om du ønsker å ha bilde større eller i en egen tab, høyreklikk på bildet og velg "Se bildet/View Image".

Løsningsforslag

Bruker her `KN`, `PN` og `FK` som forkortelse for henholdsvis *kandidatnøkler*, *primærnøkkel*, og *fremmednøkkel*.

Starter med å realisere vanlige entiteter:

```
Alv(Navn, AlvID, Født)
    - KN: {Navn}, {AlvID}
    - PN: {AlvID}
Verksted(Lengdegrad, Breddegrad, Størrelse)
    - KN: {Lengdegrad, Breddegrad}
    - PN: {Lengdegrad, Breddegrad}
Oppgave(Navn, Beskrivelse)
    - KN: {Navn}
    - PN: {Navn}
Leke(Navn, Nyttig)
```

```
    - KN: {Navn}
    - PN: {Navn}
```

Velger `AlvID` som PN for `Alv`.

Realiserer så den svake entieten:

```
Del(DelID, Leke, Funksjon)
    - KN: {DelID, Leke}
    - PN: {DelID, Leke}
    - FK: Leke -> Leke(Navn)
```

Realiserer så 1:1-relasjonen. Velger her å realisere relasjonen som fremmednøkkel i `Oppgave`, ettersom dette virker mest naturlig:

```
Oppgave(Navn, Beskrivelse, UtføresAv)
    - KN: {UtføresAv}
    - FK: UtføresAv -> Alv(AlvID)
```

Realiserer så 1:N-relasjonene. Velger å realisere `JOBBER_FOR` som fremmednøkkel i `Alv` for å redusere antall tabeller (alle alver må jo jobbe for et verksted). Velger tilsvarende for `RESULTERER_I`:

```
Alv(Navn, AlvID, Født, VerkstedL, VerkstedB, Kontrakt)
    - FK: (VerkstedL, VerkstedB) -> Verksted(Lengdegrad, Breddegrad)

Oppgave(Navn, Beskrivelse, UtføresAv, Del, Leke)
    - FK: (Del, Leke) -> Del(DelID, Leke)
```

Realiserer så M:N-relasjonen som egen relasjon:

```
Produserer(VerkstedL, VerkstedB, Leke)
    - FK: (VerkstedL, VerkstedB) -> Verksted(Lengdegrad, Breddegrad)
    - FK: Leke -> Leke(Navn)
```

Til slutt realiserer vi flerverdi-attributten `ØnskesAv`:

```
ØnskesAv(Person, Leke)
    - KN: {Person, Leke}
    - PN: {Person, Leke}
    - FK: Leke -> Leke(Navn)
```

Hele skjemaet blir da:

```
Alv(Navn, AlvID, Født, VerkstedL, VerkstedB, Kontrakt)
    - KN: {Navn}, {AlvID}
    - PN: {AlvID}
    - FK: (VerkstedL, VerkstedB) -> Verksted(Lengdegrad, Breddegrad)

Verksted(Lengdegrad, Breddegrad, Størrelse)
    - KN: {Lengdegrad, Breddegrad}
    - PN: {Lengdegrad, Breddegrad}
```

```
Oppgave(Navn, Beskrivelse, UtføresAv, Del, Leke)
    - KN: {Navn}, {UtføresAv}
    - PN: {Navn}
    - FK: (Del, Leke) -> Del(DelID, Leke)
    - FK: UtføresAv -> Alv(AlvID)

Leke(Navn, Nyttig)
    - KN: {Navn}
    - PN: {Navn}

Del(DelID, Leke, Funksjon)
    - KN: {DelID, Leke}
    - PN: {DelID, Leke}
    - FK: Leke -> Leke(Navn)

Produserer(VerkstedL, VerkstedB, Leke)
    FK: (VerkstedL, VerkstedB) -> Verksted(Lengdegrad, Breddegrad)
    FK: Leke -> Leke(Navn)

ØnskesAv(Person, Leke)
    KN: {Person, Leke}
    PN: {Person, Leke}
    FK: Leke -> Leke(Navn)
```

Alternativ realisering av UTFØRER:

Siden `UTFØRER` er 1:1 og totale deltakelser på begge sider, kan man her også bruke "merged-relation approach", og legge både `ALV`, `OPPGAVE` og relasjonen inn i én relasjon:

```
AlvOppgave(AlvNavn, AlvID, Født, OppgaveNavn, Beskrivelse)
    - KN: {AlvNavn}, {AlvID}, {OppgaveNavn}
    - PN: {AlvID}
```

Merk at vi må omdøpe attributtene, siden `Navn` forekommer i begge. Vi må også velge en primærnøkkel, og velger da `AlvID` for denne også.

Sensorveiledning

- 0.5 for hver vanlige entitet
- 3 poeng for den svake entiteten (inkludert de identifiserende relasjonene)
- 1 poeng for hver relasjon, og 1.5 for hver relasjon med attributt
- 0.5 for multi-valued attributten, utledbar attributt, og sammensatt nøkkel

Trekk:

- -0.5 for manglende attributt
- -0.5 for manglende markering av kandidatnøkler og primærnøkkel

- -0.5 for manglende fremmednøkkelmarkering
- -0.5 for manglende forklaring på valg
- -0.5 for å realisere den identifiserende relasjonen dobbelt

2 SQL (45)

Oppgavene i denne delen er om SQL. Alle oppgavene vil bruke det samme databaseskjemaet, og en beskrivelse av dette er vedlagt som PDF for hver oppgave.

Du må gjerne inkludere kommentarer i spørringene (gjøres med "–" i SQL). Dersom det er uklarheter eller tvetydigheter i oppgavebeskrivelsen, bruk sunn fornuft og skriv en kommentar om hvilke antagelser og tolkinger du gjør.

Sensorveiledning

Generelt:

- -1 for `NATURAL JOIN` som joiner på mer enn tiltenkt
- -1 for manglende/feil tabeller i joins i `FROM`
- -0.5 for manglende kolonner i `SELECT`
- -1 for manglende begrensning i `WHERE`
- Ingen trekk for å kun ha primærnøkkel i `GROUP BY` men andre attributter (fra samme tabell) i `SELECT` (støttes av PostgreSQL)
- Ingen trekk for enkle syntaksfeil man enkelt ville funnet ut av om man hadde mulighet til å kjøre spørringen

2.1 Flervalg (12)

Anta at databasen inneholder nøyaktig de dataene som er vist i eksempel-dataene i den vedlagte PDFen. For hver spørring under, oppgi hvor mange rader det er i resultatet. Hvert riktig svar gir 2 poeng, mens hvert uriktige svar gir -0.5 poeng. Blankt svar/ubesvart gir 0 poeng. Laveste poengsum på hele oppgaven er 0.

1.

```sql
SELECT *
FROM spiller
WHERE navn LIKE '%i';
```

2.

```sql
SELECT *
FROM spiller
WHERE NOT (rating > 2200);
```

3.

```sql
SELECT *
FROM spiller AS s
     JOIN parti AS p ON (s.sid = p.sort)
WHERE s.navn = 'Mina' AND
      p.vinner = 'sort';
```

4.

```sql
SELECT navn
FROM turnering
WHERE spilltype = 'hurtigsjakk'
UNION ALL
SELECT navn
FROM turnering
WHERE tid >= 3;
```

5.

```sql
SELECT *
FROM parti LEFT JOIN turnering USING (tid)
WHERE hvit = 3;
```

6.

```sql
SELECT DISTINCT count(*)
FROM parti
GROUP BY vinner;
```

Løsningsforslag

Spørring	0	1	2	3	4
1			X		
2			X		
3		X			
4					X
5					X
6			X		

Automatisk rettet.

2.2 Lynsjakk i november (5)

Skriv en spørring som finner navn på alle turneringer med spilltype `lynsjakk` som hadde startdato i november 2022 (altså fra og med `2022-11-01` og til og med `2022-11-30`).

Løsningsforslag

```sql
SELECT navn
FROM turnering
WHERE spilltype = 'lynsjakk' AND
      startdato >= '2022-11-01' AND
      startdato <= '2022-11-30'; -- evt. startdato < '2022-12-01'
```

Sensorveiledning

- Ikke trekk for `startdato LIKE '2022-11-%'`, ettersom det kun mangler å caste `startdato` til streng, noe som kunne tenkes at gjøres at databasen automatisk (som i enkelte andre tilfeller)

2.3 Tenketid (5)

Skriv en spørring som finner gjennomsnitlig tid brukt på alle trekk for alle partier på turneringen med navn `'Oslo Open'`.

Løsningsforslag

```sql
SELECT avg(tr.tid_brukt) AS gjenomsnitts_rating
FROM trekk AS tr
     JOIN parti AS p USING (pid)
     JOIN turnering AS tu USING (tid)
WHERE tu.navn = 'Oslo Open';

-- evt. om man antar at vi ønsker gj.tid per parti:

SELECT p.pid, avg(tr.tid_brukt) AS gjenomsnitts_rating
FROM trekk AS tr
     JOIN parti AS p USING (pid)
     JOIN turnering AS tu USING (tid)
WHERE tu.navn = 'Oslo Open'
GROUP BY p.pid;
```

2.4 Sjakkelakke (5)

Etter at en turnering med navn `'Sjakkelakke'` ble avholdt, klarte en av de ansatte å legge inn en rekke partier med feil resultat, alle partiene som hadde resultat `'remis'` inneholder feil. Skriv derfor en SQL-kommando som sletter alle rader fra `parti`-tabellen for partiene som har resultat `'remis'` og som er tilknyttet turneringen med navn `'Sjakkelakke'`.

Løsningsforslag

```
DELETE
FROM parti
WHERE vinner = 'remis' AND
      tid = (
        SELECT tid
        FROM turnering
        WHERE navn = 'Sjakkelakke'
      );
```

Sensorveiledning

- -2 for å ikke kun ha `parti` i `FROM`-klausulen, slik som:
  - å joine `parti` med `turnering` i `FROM`-klausulen
  - bruke en `SELECT`-delspørring i `FROM`-klausulen
  - osv.

2.5 Resultater (8)

Skriv en SQL-kommando som lager et `VIEW` med navn `resultat` som inneholder resultater av partier for hver spiller. Det skal inneholde tre kolonner: `pid` for partiet, `sid` for spilleren, og en kolonne med navn `utfall` som enten er `'vant'`, `'tapte'` eller `'remis'` avhengig av om spilleren henholdsvis vant partiet, tapte partiet, eller partiet ble remis/uavgjort.

Innholdet i viewet kan f.eks. se slik ut (om dataene er slik som i eksempeldataene):

```
 pid | sid | utfall
-----+-----+--------
   1 |   5 | vant
   1 |   3 | tapte
   2 |   2 | vant
   2 |   3 | tapte
   3 |   2 | vant
   3 |   8 | tapte
   4 |   1 | remis
   4 |   3 | remis
   5 |   8 | vant
   5 |   3 | tapte
   6 |   6 | remis
   6 |   2 | remis
   7 |   1 | tapte
   7 |   8 | vant
   8 |   6 | tapte
   8 |   7 | vant
   9 |   3 | tapte
```

```
  9 |   5 | vant
 10 |   1 | remis
 10 |   3 | remis
```

Løsningsforslag

```
CREATE VIEW resultat(pid, sid, utfall) AS
SELECT pid, hvit, 'vant'
FROM parti
WHERE vinner = 'hvit'
UNION
SELECT pid, sort, 'vant'
FROM parti
WHERE vinner = 'sort'
UNION
SELECT pid, hvit, 'tapte'
FROM parti
WHERE vinner = 'sort'
UNION
SELECT pid, sort, 'tapte'
FROM parti
WHERE vinner = 'hvit'
UNION
SELECT pid, hvit, 'remis'
FROM parti
WHERE vinner = 'remis'
UNION
SELECT pid, sort, 'remis'
FROM parti
WHERE vinner = 'remis';
```

2.6 Vinnere (10)

I sjakkturneringene får man 1 poeng hver gang man vinner et parti og 0 poeng om man taper. Dersom et parti ender med remis (uavgjort) vil hver spiller få 0.5 poeng hver. Vinneren av turneringen er den med flest poeng. Dersom det er uavgjort legges det til ekstra partier for de med flest poeng, slik at det alltid ender med at én har flest poeng i hver turnering.

Skriv en spørring som finner vinneren på hver turnering. Spørringen skal returnere navn på turneringen og navnet på spilleren som vant turneringen. Du kan bruke viewet fra forrige oppgave om du ønsker.

Løsningsforslag

```
WITH
  poeng AS (
```

```
    SELECT tid, sid, 1 AS poeng
    FROM resultat JOIN parti USING (pid)
    WHERE utfall = 'vant'
    UNION
    SELECT tid, sid, 0.5 AS poeng
    FROM resultat JOIN parti USING (pid)
    WHERE utfall = 'remi'
  ),
  sum_poeng AS (
    SELECT tid, sid, sum(poeng) AS total
    FROM poeng
    GROUP BY tid, sid
  )
SELECT t.navn, s.navn AS vinner
FROM turnering AS t,
     spiller AS s
WHERE s.sid = (
        SELECT p.sid
        FROM sum_poeng AS p
        WHERE p.tid = t.tid
        ORDER BY total DESC
        LIMIT 1
      );
```

Sensorveiledning

- 5 poeng for å finne riktig poengsum for hver spiller på hver turnering
- 5 poeng for så å klare å finne høyest poengsum per turnering

3 Relasjonsmodellen og normalformer (25)

3.1 Nøkler (7) (flervalg)

Gitt følgende relasjon

`R(A, B, C, D, E, F, G)`

med følgende FDer:

1. $D \rightarrow B$
2. $BC \rightarrow E$
3. $A \rightarrow FE$
4. $F \rightarrow A$

Angi for hvert attributt hvorvidt det er et nøkkelattributt (JA) eller ikke (NEI). Hvert riktige svar gir 1 poeng, mens hvert gale svar gir -1 poeng. Ubesvart gir alltid 0 poeng. Laveste poengsum for hele oppgaven er 0.

Er attributtet et nøkkelattributt i R?

Attributt	JA	NEI
A		
B		
C		
D		
E		
F		
G		

Løsningsforslag

Attributt	JA	NEI
A	X	
B		X
C	X	
D	X	
E		X
F	X	
G	X	

Automatisk rettet.

3.2 Normalformer (8) (flervalg)

Gitt følgende relasjon:

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

med eneste kandidatnøkkel `BCF`.

For hver av FDene under, anta at FDen gjelder for relasjonen R over, og bruk algoritmen for å finne normalform til å avgjøre hvilken normalform FDen (alene) tilsier at R er på.
Hvert riktige svar gir 2 poeng, hvert uriktige svar gir -0.5 poeng, ubesvart gir 0 poeng, men total poengsum på hele oppgaven vil ikke bli lavere enn 0.

FD	1NF	2NF	3NF	BCNF
$AB \to E$				
$BCF \to D$				
$D \to A$				
$F \to A$				

Løsningsforslag

FD	1NF	2NF	3NF	BCNF
$AB \to E$		X		
$BCF \to D$				X
$D \to A$		X		
$F \to A$	X			

Automatisk rettet.

3.3 Tapsfri dekomposisjon (10)

Gitt følgende relasjon:

`R(A, B, C, D, E, F, G)`

med kandidatnøklene `ACF` og `ADF`, samt følgende FDer:

1. $A \to B$
2. $C \to DE$
3. $D \to C$
4. $EF \to G$

Dekomponer R tapsfritt til BCNF. Vis stegene du gjør og list opp kandidatnøkler og FDer på alle relasjonene underveis.

Løsningsforslag

Starter med å splitte opp høyresider, slik at det kun er ett attributt per:

1. $A \to B$
2. $C \to D$
3. $C \to E$
4. $D \to C$
5. $EF \to G$

Så dekomponerer vi, og starter med $A \to B$: A er ikke supernøkkel, så brudd på BCNF. $A^+ = AB$, og vi får da relasjonene

- $S_1(A, B)$ med FD 1 og dermed kandidatnøkkel A, og er dermed på BCNF.
- $S_2(A, C, D, E, F, G)$ med FDene 2, 3, 4, 5. Den vil da få følgende kandidatnøkler ACF og ADF. FD 2 bryter med BCNF siden C ikke er en supernøkkel, og vi dekomponerer videre: $C^+ = CDE$
  - $S_{21}(C, D, E)$ med FDene 2, 3, 4, og dermed kandidatnøkler C og D. Ingen av FDene bryter BCNF, og S_{21} er på BCNF.
  - $S_{22}(A, C, F, G)$, og har ingen FDer, kandidatnøkkel $ACFG$ og er dermed på BCNF.

Altså kan R dekomponeres tapsfritt til $S_1(A, B)$, $S_{21}(C, D, E)$ og $S_{22}(A, C, F, G)$ med FDer og kandidatnøkler som gitt over.

Sensorveiledning

- 5 poeng for riktig første dekomponering (inkludert nøkler og FDer på de nye relasjonene)
- 2.5 poeng for hver av de to andre dekomponeringene (inkludert nøkler og FDer på de nye relasjonene)
- -1 poeng for manglende/feil FD på relasjon
- -1 poeng for feil utregning av tillukning
- -1 poeng for manglende/feil kandidatnøkler
- -3 poeng for ingen forklaringer