Eksamen IN2090 Høsten 2023

Skriftlig eksamen i IN2090

2023 HØST

Tid: 13. desember 09:00 til 13. 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 (50)
- Relasjonsmodellen, normalformer og dekomposisjon (20)

1 Modellering (30)

1.1 ER: Bank (15)

Du er ansatt av en bank som skal lage en ny database over deres kunder, konti (flertall av "konto" er "konti") og transaksjoner. Databasen skal inneholde følgende informasjon:

1. Kunder identifiseres med deres personnummer, og har ellers et navn.
2. En konto har et unikt kontonummer. Kontonummeret består av en prefiks og en suffiks, hvor prefiksen igjen består av et registernummer og et gruppenummer, mens suffiksen består av et kundenummer og et kontrollsiffer (f.eks. vil kontonummeret `1234 56 78901` ha prefiks `1234 56` og suffiks `78901`, hvor `1234` er registernummeret, `56` er gruppenummeret, `7890` er kundenummeret og `1` er kontrollsifferet). Ellers har hver konto en balanse

(altså mengden kroner på kontoen), samt en type (f.eks. `sparekonto`, `forbrukskonto`, osv.). Typen er derimot utledbar fra kontonummeret.

3. Hver konto eies av nøyaktig én kunde. En kunde kan eie mange konti, men trenger ikke eie noen. Videre kan en konto disponeres av mange kunder, og en kunde kan disponere mange konti.
4. Banken ønsker også å lagre alle transaksjoner. En transaksjon blir utført fra én konto til en annen konto på et gitt tidspunkt. Kombinasjonen av tidspunktet, til-kontoen og fra-kontoen er unik for transaksjoner. Ellers kan en konto være innvolvert i mange transaksjoner, både som til-konto og fra-konto. Videre har transaksjoner en verdi, som er beløpet som overføres.
5. Ettersom mange kunder kan disponere en konto ønsker banken også å lagre hvilken kunde som utførte transaksjonen. En transaksjon må utføres av en kunde, men kan høyst bli utført av én kunde. En kunde kan utføre mange transaksjoner, men trenger ikke å ha utført noen transaksjoner.
6. Til slutt ønsker banken å lagre fakturaer. En faktura identifiseres med et unikt KID-nummer, og har ellers et beløp og en forfallsdato. En faktura er til nøyaktig én kunde. En kunde kan derimot få mange faktura, men trenger ikke å ha fått noen fakturaer.
7. En transaksjon kan brukes for å betale en faktura. Banken ønsker derfor også å lagre hvilke transaksjoner som eventuelt betaler en faktura. En faktura kan høyst bli betalt av én transaksjon, og en transaksjon kan maks betale én faktura. Derimot trenger ikke en faktura bli betalt av noen transaksjon, og en transaksjon trenger heller ikke betale en faktura.

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

Se Figur 1.

1.2 ER: Hemmelig (5)

En hemmelig organisasjon (så hemmelig at selv navnet er hemmelig) trenger hjelp fra deg til å forstå et ER-diagram for en database de har. Innholdet i databasen er fryktelig hemmelig, og ER-diagrammet for databasen er derfor

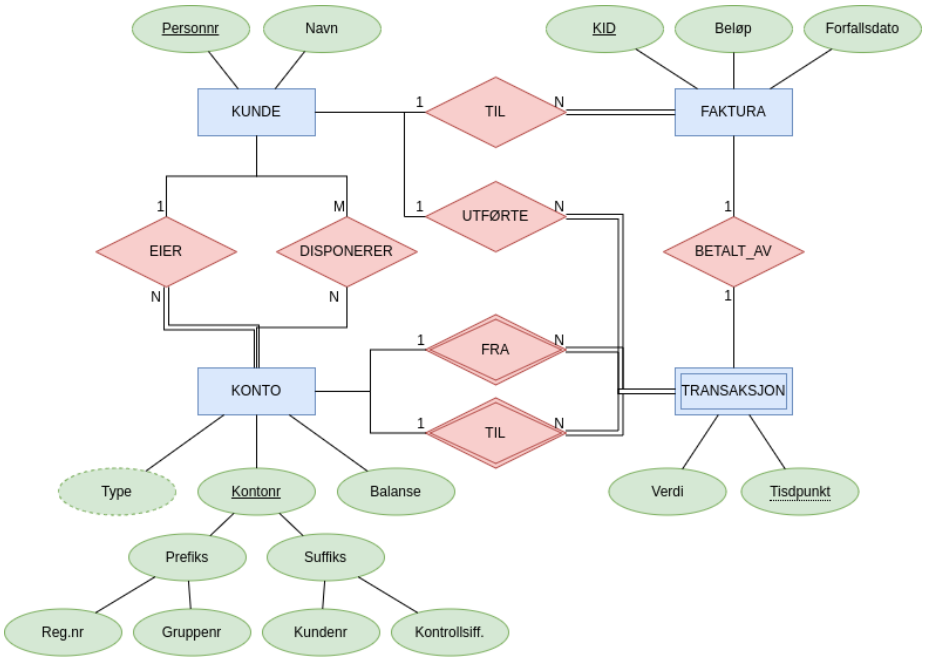

Figure 1: Bank-modell

kryptert.

Organisasjonen trenger hjelp til å vite 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. En AK er unikt identifisert med dens Agn-verdi.
2. Alle AK er relatert til minst én BO.
3. Dersom det finnes en BO i databasen, vil det også finnes en CU.
4. Gitt en CU og en BO, så kan disse være RACB-relatert til mange AK.
5. En AK må alltid ha minst én Asso-verdi.

Løsningsforslag

1. JA
2. NEI
3. JA
4. JA
5. NEI

Automatisk rettet.

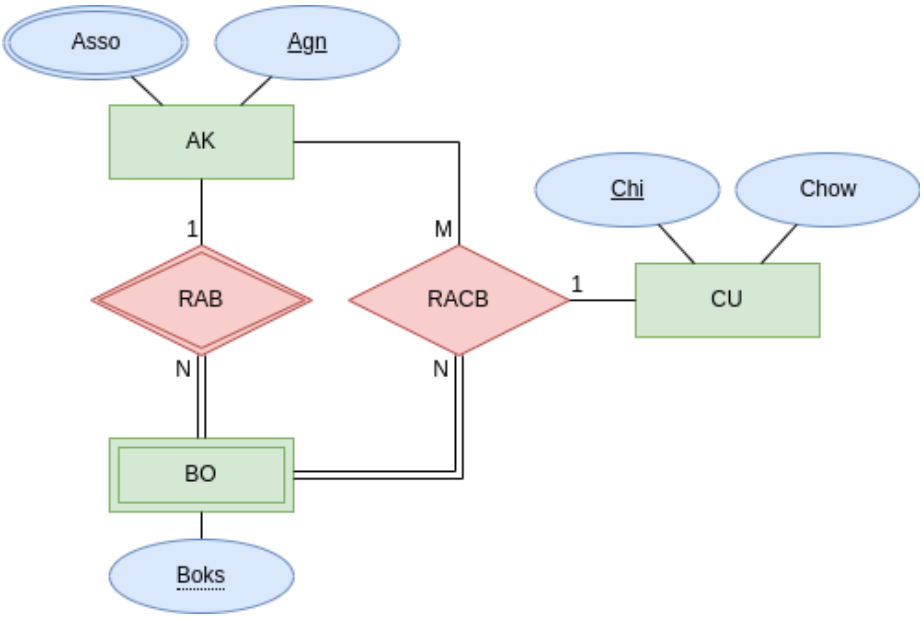

Figure 2: Hemmelig-modell

1.3 Realisering: (10)

Realiser denne 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 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:

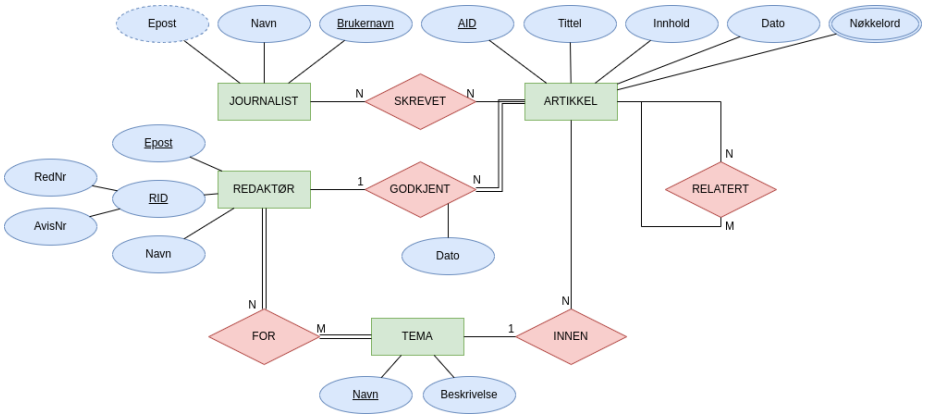

Figure 3: ER-modell over databasen til en nettavis

```
Journalist(brukernavn, navn)
- KN/PN: {brukernavn}

Artikkel(aid, tittel, innhold, dato)
- KN:PN: {aid}

Redaktør(epost, rednr, avisnr, navn)
- KN: {epost}, {rednr, avisnr}
- PN: {epost}

Tema(navn, beskrivelse)
- KN/PN: {navn}
```

Velger her `{epost}` som PN for `Redaktør` ettersom det er kortere og en naturlig attributt å bruke som referanse til en redaktør.

Ingen svake entiteter eller 1:1-relasjoner, så realiserer så 1:N-relasjoner. Velger å realisere `GODKJENT` som fremmednøkkel i `Atrikkel` ettersom alle artikler har en redaktør som har godkjent artikkelen:

```
Artikkel(aid, tittel, innhold, dato, redaktør, godkjent_dato)
- KN:PN: {aid}
- FN: (redaktør) -> Redaktør(epost)
```

Velger videre å realisere `INNEN` som fremmednøkkel i `ARTIKKEL` ettersom det virker naturlig at de fleste artikler har et tema:

```
Artikkel(aid, tittel, innhold, dato, redaktør, godkjent_dato, tema)
- KN:PN: {aid}
- FN: (redaktør) -> Redaktør(epost)
      (tema) -> Tema(navn)
```

Realiserer så M:N-relasjonene som egne relasjoner:

```
For(redaktør, tema)
- KN/PN: {redaktør, tema}
- FN: (redaktør) -> Redaktør(epost)
      (tema) -> Tema(tittel)

Skrevet(journalist, artikkel)
- KN/PN: {journalist, artikkel}
- FN: (jounralist) -> Jounralist(brukernavn)
      (artikkel) -> Artikkel(aid)

Relatert(artikkel1, artikkel2)
- KN/PN: {artikkel1, artikkel2}
- FN: (artikkel1) -> Artikkel(aid)
      (artikkel2) -> Artikkel(aid)
```

Til slutt realiserer vi flerverdi-attributten:

```
Nøkkelord(artikkel, ord)
- KN/PN: {artikkel, ord)
- FN: (artikkel) -> Artikkel(aid)
```

Hele skjemaet blir da:

```
Journalist(brukernavn, navn)
- KN/PN: {brukernavn}

Artikkel(aid, tittel, innhold, dato, redaktør, godkjent_dato, tema)
- KN:PN: {aid}
- FN: (redaktør) -> Redaktør(epost)
      (tema) -> Tema(navn)

Redaktør(epost, rednr, avisnr, navn)
- KN: {epost}, {rednr, avisnr}
- PN: {epost}

Tema(navn, beskrivelse)
- KN/PN: {navn}

For(redaktør, tema)
- KN/PN: {redaktør, tema}
- FN: (redaktør) -> Redaktør(epost)
      (tema) -> Tema(navn)

Skrevet(journalist, artikkel)
- KN/PN: {journalist, artikkel}
- FN: (jounralist) -> Journalist(brukernavn)
```

```
      (artikkel) -> Artikkel(aid)

Relatert(artikkel1, artikkel2)
- KN/PN: {artikkel1, artikkel2}
- FN: (artikkel1) -> Artikkel(aid)
      (artikkel2) -> Artikkel(aid)

Nøkkelord(artikkel, ord)
- KN/PN: {artikkel, ord)
- FN: (artikkel) -> Artikkel(aid)
```

2 SQL (50)

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.

2.1 Flervalg (10)

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 DISTINCT oid
FROM sensor
WHERE sid >= 5;
```

2.

```sql
SELECT *
FROM sensor
WHERE oid = 1
LIMIT 4;
```

3.

```sql
SELECT * FROM måling
WHERE nedbør = vind;
```

4.

```sql
SELECT *
FROM område AS o
WHERE NOT EXISTS (
  SELECT * FROM sensor AS s
  WHERE s.oid = o.oid
);
```

5.

```sql
SELECT oid FROM sensor
EXCEPT ALL
SELECT oid FROM område;
```

Løsningsforslag

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

Automatisk rettet.

2.2 Vedlikehold i Oslo (5)

Skriv en spørring som finner alle vedlikeholdsdatoer for sensorer i Oslo sortert kronologisk.

Løsningsforslag

```sql
SELECT vedlikeholdt
FROM område JOIN sensor USING (oid)
WHERE navn = 'Oslo'
ORDER BY vedlikeholdt;
```

2.3 Gjennomsnittstemperatur i Bærum (5)

Skriv en spørring som finner gjennomsnittstemperaturen i Bærum når det regner (`nedbør` er større enn `0`) eller blåser mye (mer enn `5` m/s vind).

Løsningsforslag

```
SELECT avg(temp) AS gj_temp
FROM område AS o
     JOIN sensor AS s USING (oid)
     JOIN måling AS m USING (sid)
WHERE o.navn = 'Bærum' AND
      (nedbør > 0 OR vind > 5);
```

2.4 Få sensorer (5)

Skriv en spørring som finner navn på de tre områdene med færrest sensorer per kvadratkilometer Det kan forekomme områder uten sensorer, og disse skal også med. *Hint:* Dersom et område har areal `A` og `N` sensorer vil det ha `N/A` sensorer per kvadratkilometer

Løsningsforslag

```
SELECT o.navn, count(s.sid)/o.areal AS sensorer_per_areal
FROM område AS o LEFT JOIN sensor AS s ON (o.oid = s.oid)
GROUP BY o.oid, o.navn, o.areal
ORDER BY sensorer_per_areal
LIMIT 3;
```

2.5 Analyse (10)

I målingene kan det som sagt være noen `NULL`-verdier, og disse kan gi problemer i enkelte analyser hvor dataene brukes. Skriv derfor en SQL-kommando som lager et `VIEW` med navn `analyse_vind` som inneholder `sid`, `tidspunkt` og `vind`-verdiene fra `måling`-tabellen, men som erstatter alle `NULL`-verdier i `vind`-kolonnen med gjennomsnittet av målingene (som ikke er `NULL`) fra samme sensor på samme dag.

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

```
 sid |      tidspunkt      | vind
-----+---------------------+------
   2 | 2023-12-02 12:00:00 |  0.3
   2 | 2023-12-02 12:01:00 |  0.4
   3 | 2023-11-30 09:01:30 |    0
   3 | 2023-11-30 09:04:01 |  4.1
   3 | 2023-12-01 11:57:46 |  3.2
   7 | 2023-10-16 16:11:47 |  7.9
   7 | 2023-10-16 16:11:59 |  7.8
   7 | 2023-10-17 16:15:22 |  7.8
   8 | 2023-11-09 22:48:12 |  0.5
   4 | 2023-11-16 03:11:01 |  0.7
   4 | 2023-11-16 04:11:02 |  0.8
```

```
  2 | 2023-12-02 12:02:11 | 0.35
```

Hint: Du kan omgjøre et `timestamp` til en `date` ved å *caste* den til `date`, altså `tidspunkt::date` henter ut dato-verdien i `tidspunkt`.

Løsningsforslag

```
CREATE VIEW analyse_vind AS
SELECT sid, tidspunkt, vind
FROM måling
WHERE vind IS NOT NULL
UNION ALL
SELECT m.sid, m.tidspunkt, avg(o.vind)
FROM måling AS m JOIN måling AS o USING (sid)
WHERE m.tidspunkt::date = o.tidspunkt::date AND
      m.vind IS NULL AND
      o.vind IS NOT NULL
GROUP BY m.sid, m.tidspunkt;

-- eller eventuelt følgende
-- (men denne vil kunne ha NULL-verdier for vind
--  dersom det ikke finnes noen andre målinger for en sensor på samme dag)

CREATE VIEW analyse_vind AS
SELECT sid, tidspunkt, vind
FROM måling
WHERE vind IS NOT NULL
UNION ALL
SELECT sid, tidspunkt,
  (SELECT avg(vind)
   FROM måling AS o
   WHERE o.sid = m.sid AND
         o.tidspunkt::date = m.tidspunkt::date AND
         o.vind IS NOT NULL
  ) AS vind
FROM måling AS m
WHERE vind IS NULL;
```

2.6 Nedbørsmengder (10)

Skriv en spørring som finner navnet på område med høyest gjennomsnittlig nedbørsmengde per dag i 2023. Ta kun med områder og dager som har mer enn 10 målinger. Du kan i denne oppgaven anta at alle målinger på nedbør har en verdi (altså ikke er `NULL`). Du kan også anta at summen av alle `nedbør`-verdier for et område for en dag er den totale nedbørsmengden for det område den dagen. *Hint:* Du kan bruke `tidspunkt::date` for å hente ut datoen/dagen for en måling.

Løsningsforslag

```sql
WITH
  nedbør_pd AS (
    SELECT s.oid, tidspunkt::date, sum(m.nedbør) AS nedbør_pd
    FROM måling AS m JOIN sensor AS s USING (sid)
    WHERE '2023-01-01'::date <= m.tidspunkt AND
          m.tidspunkt <= '2023-12-31'::date -- kan også bruke current_date
    GROUP BY s.oid, tidspunkt::date
    HAVING count(*) > 10
  ),
  nedbør_gj AS (
    SELECT oid, avg(nedbør_pd) AS nedbør_gj
    FROM nedbør_pd
    GROUP BY oid
  )
SELECT o.navn
FROM nedbør_gj AS g JOIN område AS o USING (oid)
ORDER BY nedbør_gj DESC
LIMIT 1;
```

2.7 Programmering med SQL (5)

Metrologisk Institutt vedlikeholder alle sensorer med jevne mellomrom, og alle sensorer i et område blir vanligvis vedlikeholdt på samme dag.

I denne oppgaven skal du derfor lage en funksjon/metode i Java eller Python som er del av et større program som vedlikeholderne bruker. Metoden/funksjonen du lager skal brukes for å oppdatere `vedlikeholdt`-datoene i `sensor`-tabellen til dagens dato etter en slik vedlikeholdsrunde. Du kan velge om du ønsker å bruke Python eller Java i denne oppgaven. For Python skal funksjonens signatur se slik ut:

```python
def vedlikeholdt(conn, omrade):
    # TODO
```

For Java ser metode-signaturen slik ut:

```java
void vedlikeholdt(Connection conn, String omrade) throws SQLException {
    // TODO
}
```

Funksjonen/metoden tar to argumenter, et tilkoblingsobjekt `conn` med en aktiv tilkobling til databasen, samt en streng-verdi `omrade` fra brukeren som inneholder navnet på område som vedlikeholderen har gått runde i. Funksjonen skal så lage og eksekvere en SQL-kommando som oppdaterer `vedlikeholdt`-datoen til alle

sensorer i det angitte område til dagens dato. (I SQL er dagens dato tilgjengelig via `current_date`.)

Du trenger ikke håndtere feilmeldinger (`Exception`) eller liknende i oppgaven, men løsningen din skal ikke være sårbar for SQL-injection-angrep via `omrade`-strengen.

Løsningsforslag

Python:

```python
def vedlikeholdt(conn, omrade):
    cur = conn.cursor()
    cur.execute("UPDATE sensor "
                "SET vedlikeholdt = current_date "
                "WHERE oid = (SELECT oid FROM område WHERE navn = %s);",
                (omrade,))
    conn.commit()
```

Java:

```java
void vedlikeholdt(Connection conn, String omrade) throws SQLException {
    PreparedStatement stmt = conn.prepareStatement(
        "UPDATE sensor " +
        "SET vedlikeholdt = current_date " +
        "WHERE oid = (SELECT oid FROM område WHERE navn = ?);"
    );
    stmt.setString(1, omrade);
    stmt.execute();
    conn.commit();
}
```

3 Relasjonsmodellen og normalformer (20)

3.1 Nøkler (5) (flervalg)

Gitt følgende relasjon

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

med følgende FDer:

1. $A \rightarrow B$
2. $B \rightarrow C$
3. $C \rightarrow A$
4. $C \rightarrow D$

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		

Løsningsforslag

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

Automatisk rettet.

3.2 Funksjonelle avhengigheter (5) (flervalg)

Gitt følgende relasjon:

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

med følgende funksjonelle avhengigheter:

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

og med følgende tupler:

A	B	C	D	E	F
1	2	?	4	5	6
?	3	4	8	9	7
1	2	5	?	5	7
2	3	5	8	?	1
1	?	5	4	5	8

I tuplene over er det én verdi per attributt som er erstattet med `?`, og din oppgave er å finne riktig verdi basert på de andre verdiene og de funksjonelle avhengighetene. Altså, for hvert attributt, angi verdien som dens `?` må erstattes med for at relasjonen skal være riktig i henhold til de funksjonelle avhengighetene:

Attributt	Verdi
A	
B	
C	
D	
E	

Hvert riktige svar gir 1 poeng, ubesvart gir 0 og uriktig svar gir -0.5 poeng. Laveste poengsum på hele oppgaven er 0.

Løsningsforslag

En funksjonell avhengighet uttrykker at for alle tupler/rader med like verdier i venstresidens attributter, så må også høyresidens attributter være like. F.eks. må derfor A sitt `?` ha verdien `1` fordi $F \rightarrow A$, og F har verdi `7` hvor A har `?`, og i raden under har F også verdi `7` og der er As verdi `1`.

På denne måten kan vi derfor komme frem til følgende verdier:

Attributt	Verdi
A	1
B	2
C	5
D	4
E	9

Altså ser hele tabellen slik ut:

A	B	C	D	E	F
1	2	5	4	5	6
1	3	4	8	9	7
1	2	5	4	5	7
3	3	5	8	9	1
1	2	5	4	5	8

Automatisk rettet.

3.3 Tapsfri dekomposisjon (10)

Gitt følgende relasjon:

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

med kandidatnøklene AB, BC, samt følgende funksjonelle avhengigheter:

1. $B \rightarrow E$
2. $AB \rightarrow C$
3. $E \rightarrow D$
4. $C \rightarrow A$

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

Løsningsforslag

Starter med å finne en FD som bryter med BCNF, og ser at $B \rightarrow E$ bryter med BCNF: Vi dekomponerer så med hensyn på denne, $B^+ = BED$:

- Første relasjon i dekomponeringen av R blir $S_1(B, E, D)$, med KN B og FDene 1 og 3. Her ser vi at FD 3 ($E \rightarrow D$) bryter så dekomponerer videre med denne, $E^+ = ED$
  - Førse relasjon i dekomponeringen av S_1 blir $S_{11}(E, D)$ med KN E og FD 3. Denne ser vi er på BCNF ettersom FD 3 ikke bryter med BCNF.
  - Den andre relasjonen i dekomponeringen av S_1 er $S_{12}(E, B)$ med KN E og FD 1, og også denne er på BCNF av samme grunn som over.
- Den andre relasjonen i dekomponeringen av R er $S_2(B, A, C)$ med KN AB og BC, og FDer 2 og 4. Her ser vi at FD 4 ($C \rightarrow A$) bryter, så dekomponerer videre med hensyn på denne, $C^+ = CA$:
  - Første relasjon blir da $S_{21}(C, A)$ med KN C og FD 4, som vi er på BCNF.
  - Andre relasjon blir $S_{22}(C, B)$ med KN CB og ingen FDer og er derfor på BCNF.

Altså kan R dekomponeres tapsfritt til $S_{11}(E, D)$ $S_{12}(E, B)$ $S_{21}(C, A)$ $S_{22}(C, B)$

med kandidatnøkler og FDer som vist over.