Oppgaver til repetisjonsforelesning H23 (devlis basert på Prøveeksamen H20)

1 Modellering

1.1 Fileshare-model [Redusert fra prøveeksamen H20]

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.

Modellen skal inneholde følgende:

- FileShare identifiserer sine brukere internt med unike IDer (f.eks. `324`). FileShare krever at brukeren oppgir en e-postadresse. FileShare tillater ikke brukere å registrere mer enn én e-postadresse, og den samme eposten kan ikke benyttes av flere brukere. I tillegg kan brukere legge inn navn (bestående av for- og etternavn), samt brukerens telefonnumre (potensielt fler nummere per bruker).
- 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 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`).

- Brukere kan utføre operasjoner på filer, og en operasjon identifiseres med dens navn. 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.

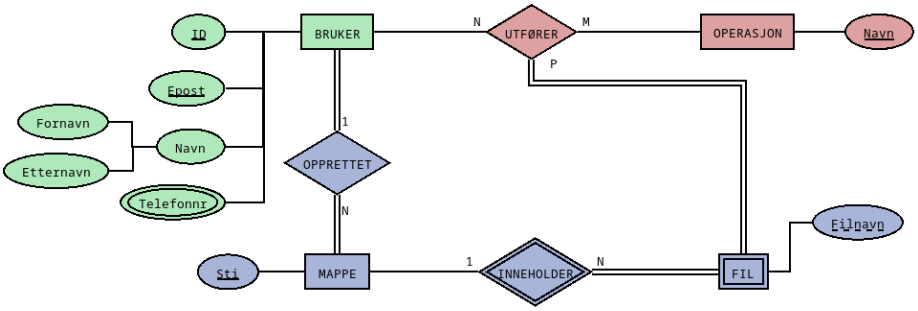

Figure 1: Løsningsforslag modellering

1.2 Realisering [Redusert fra prøveeksamen H20]

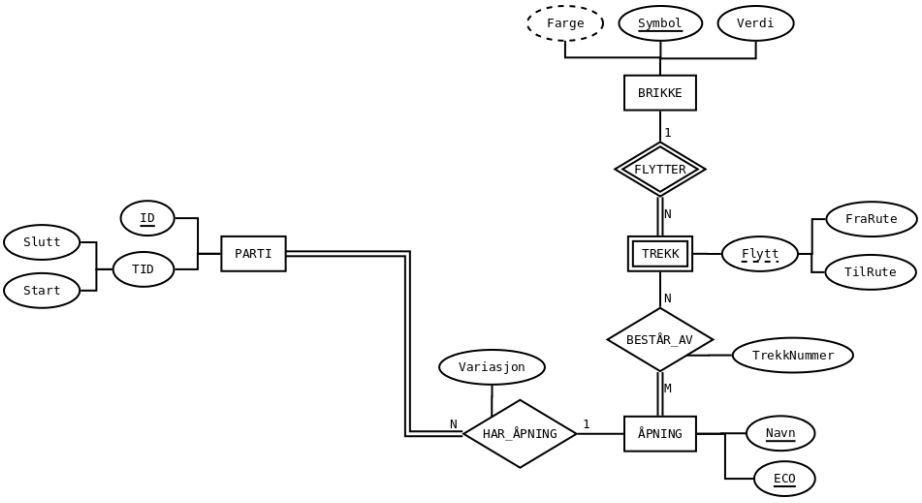

Figure 2: ER-modell

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 og kandidatnøkler tydelig. 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.

Løsningsforslag

1. Starter med å realisere normale entiteter:
   - `Parti(ID, Start, Slutt)`
     - KN/PN: `{ID}`
   - `Brikke(Symbol, Verdi)`
     - KN/PN: `{Symbol}`
   - `Åpning(Navn, ECO)`
     - KN: `{Navn}`, `{ECO}`
     - PN: `{ECO}`
2. Deretter realiserer vi svake entiteter:
   - `Trekk(FraRute, TilRute, Symbol)`
     - KN/PN: `{FraRute, TilRute, Symbol}`
     - FN: `Trekk(Symbol) -> Brikke(Symbol)`
3. Så realiserer vi relasjoner. Har ingen 1-1, så fortsetter med 1-N: Velger å realisere `HAR_ÅPNING` som attributt til `Parti`, siden hvert parti har nøyaktig én åpning:
   - `Parti(ID, Start, Slutt, HvitSpiller, SortSpiller, Åpning, Variasjon)`
     - KN/PN: `{ID}`
     - FN: `Parti(Åpning) -> Åpning(ECO)`
4. Realiserer så N-M-relasjoner:
   - `BestårAv(Åpning, FraRute, TilRute, Symbol, TrekkNummer)` hvor
     - PN/KN: `{Åpning, FraRute, TilRute, Symbol}`
     - FN: `BestårAv(Åpning) -> Åpning(ECO)` og `BestårAv(FraRute, TilRute, Symol) -> Trekk(FraRute, TilRute, Symol)`

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

- `Parti(ID, Start, Slutt, HvitSpiller, SortSpiller, Åpning, Variasjon)`
  - KN/PN: `{ID}`
  - FN: `Parti(Åpning) -> Åpning(ECO)`
- `Brikke(Symbol, Verdi)`
  - KN/PN: `{Symbol}`
- `Åpning(Navn, ECO)`
  - KN: `{Navn}`, `{ECO}`
  - PN: `{ECO}`
- `Trekk(FraRute, TilRute, Symbol)`
  - KN/PN: `{FraRute, TilRute, Symbol}`
  - FN: `Trekk(Symbol) -> Brikke(Symbol)`
- `BestårAv(Åpning, FraRute, TilRute, Symbol, TrekkNummer)` hvor
  - PN/KN: `{Åpning, FraRute, TilRute, Symbol}`
  - FN: `BestårAv(Åpning) -> Åpning(ECO)` og `BestårAv(FraRute, TilRute, Symol) -> Trekk(FraRute, TilRute, Symol)`

2 SQL

(Skjema og enkelte oppgaver basert på Prøveeksamen H20)

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

2.1 Antall gaver [Endret fra prøveeksamen H20]

Skriv en spørring som finner antall nyttige gaver hvert barn ønsker seg. Merk at det kan finnes barn som ikke ønsker seg noen nyttige gaver, og disse skal også med i svaret. Resultatet skal inneholde navn på barnet og antall gaver barnet ønsker seg.

Løsningsforslag

```
WITH
  nyttig_gave AS (
    SELECT gid
    FROM gave
    WHERE nyttig
  )
SELECT b.navn, count(ng.gid) AS antall_gaver
FROM barn AS b
     LEFT OUTER JOIN ønskeliste AS ø ON (b.bid = ø.barn)
     LEFT OUTER JOIN nyttig_gave AS ng ON (ø.gave = ng.gid)
GROUP BY b.navn;

-- eller

WITH
  nyttig_gave_ønske AS (
    SELECT gid, bid
    FROM gave AS g
         JOIN ønskeliste AS ø ON (g.gid = ø.gave)
    WHERE g.nyttig
  )
SELECT b.navn, count(ng.gid) AS antall_gaver
FROM barn AS b
     LEFT OUTER JOIN nyttig_gave_ønske AS ø ON (b.bid = ø.bid)
GROUP BY b.navn;
```

2.2 Like ønkser [Endret fra prøveeksamen H20]

Skriv en SQL-spørring som finner alle par av *ulike* barn som har ønsket seg det samme. Svaret skal kun inneholde unike rader.

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
  )
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);
```

```sql
-- eller

SELECT b1.navn AS barn1, b2.navn AS barn2
FROM ønskeliste AS ø1
     INNER JOIN ønskeliste AS ø2 USING (gave)
     INNER JOIN barn AS b1 ON (ø1.barn = b1.bid)
     INNER JOIN barn AS b2 ON (ø2.barn = b2.bid);
WHERE ø1.barn != ø2.barn;
```

2.3 Populære gaver [Fra prøveeksamen H20]

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.

Løsningsforslag

```sql
WITH
  ønsket_gave AS (
    SELECT g.navn AS gave, g.nyttig
    FROM ønskeliste AS ø
         INNER JOIN gave AS g ON (ø.gave = g.gid)
  )
(SELECT gave, count(*) AS antall_ønsker, true AS nyttig
 FROM ønsket_gave
 WHERE nyttig
 GROUP BY gave
 ORDER BY antall_ønsker DESC
 LIMIT 3)
UNION ALL -- eller bare UNION
(SELECT gave, count(*) as antall_ønsker, false AS nyttig
 FROM ønsket_gave
 WHERE NOT nyttig
 GROUP BY gave
 ORDER BY antall_ønsker DESC
 LIMIT 3);
```

2.4 Gaveliste [Fra prøveeksamen H20]

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.

Løsningsforslag

```sql
WITH
  ønsket_gave AS (-- snille barn får alt de ønsker seg
    SELECT b.bid, b.navn AS barn, 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)
  ),
  usnille_med_ønsker AS ( -- usnille barn får alle nyttige gaver de ønsker seg
    SELECT bid, barn, gave
    FROM ønsket_gave
    WHERE NOT snill AND nyttig
  ),
  usnille_uten_ønsker AS (-- usnille uten nyttige ønsker får 'Genser'

    SELECT navn AS barn, 'Genser' AS gave
    FROM barn
    WHERE NOT snill AND
          bid NOT IN (SELECT bid FROM ønsket_gave WHERE nyttig)
  )
SELECT barn, gave
FROM ønsket_gave
WHERE snill
UNION ALL
SELECT barn, gave
FROM usnille_med_ønsker
UNION ALL
SELECT barn, gave
FROM usnille_uten_ønsker;

-- eller litt kortere:

WITH
  ønsket_gave AS (
    SELECT b.bid, b.navn AS barn, g.navn AS gave, b.snill, g.nyttig
    FROM barn AS b
         JOIN ønskeliste AS ø ON (b.bid = ø.barn)
         JOIN gave AS g ON (ø.gave = g.gid)
  )
SELECT barn, gave -- snille barn får alt de ønsker seg
FROM ønsket_gave
WHERE snill
UNION ALL
SELECT barn, gave -- usnille barn får alle nyttige gaver de ønsker seg
FROM ønsket_gave
```

```sql
WHERE NOT snill AND nyttig
UNION ALL
SELECT navn, 'Genser' AS gave -- usnille uten nyttige ønsker får 'Genser'
FROM barn
WHERE NOT snill AND
      bid NOT IN (SELECT bid FROM ønsket_gave WHERE nyttig);
```

2.5 Programmering med SQL

Implementer en funksjon/metode som setter inn et nytt barn i `barn`-tabellen. Du velger selv om du vil bruke Python eller Java. Metoden har følge signatur i Python:

```python
def sett_inn_barn(conn, navn):
    # TODO
```

og følgende signatur i Java:

```java
void settInnBarn(Connection conn, String navn) throws SQLException {
    // TODO
}
```

Funksjonen/metoden tar som første argument et `Connection`-object med tilkobling til databasen. Det andre argument er en streng som inneholder navnet til barnet som skal settes og er oppgitt av en bruker av programmet. Barnet skal få en `bid`-verdi som er `1` høyere enn høyeste `bid`-verdi til barn allerede i databasen. Du kan anta at alle barn er snille til det motsatte er bevist, og skal derfor ha `snill=true` når de settes inn i databasen.

I oppgaven trenger du ikke håndtere feil (`Exceptions`), men implementasjonen skal ikke være sårbar for SQL-injection-angrep via `navn`-argumentet.

Løsningsforslag

Python:

```python
def sett_inn_barn(conn, navn):
    cur = conn.cursor()
    cur.execute(
        "INSERT INTO barn VALUES (%, 1+(SELECT max(bid) FROM barn), true);",
        (navn,))
    conn.commit()

# eller

def sett_inn_barn(conn, navn):

    cur = conn.cursor()
    cur.execute("SELECT max(bid) FROM barn;")
```

```
    bid = cur.fetchone()[0] + 1
    cur.execute("INSERT INTO barn VALUES (%, %, true);", (navn,bid))
    conn.commit()
```

Java:

```
void settInnBarn(Connection conn, String navn) throws SQLException {
    PreparedStatement stmt = conn.prepareStatement(
        "INSERT INTO barn VALUES (?, 1+(SELECT max(bid) FROM barn), true);");
    stmt.setString(1, navn);
    stmt.execute();
    conn.commit();
}

// eller

void settInnBarn(Connection conn, String navn) throws SQLException {

    Statement bidStmt = conn.createStatement();
    ResultSet res = bidStmt.executeQuery("SELECT max(bid) FROM barn;");
    res.next();
    int bid = res.getInt(1) + 1;

    PreparedStatement stmt = conn.prepareStatement(
        "INSERT INTO barn VALUES (?, ?, true);");
    stmt.setString(1, navn);
    stmt.setInt(2, bid);
    stmt.execute();
    conn.commit();
}
```

2.6 Sikkerhet

Gitt følgende funksjon i Python:

```
def skriv_ut_gave(conn, gave):
    cur = conn.cursor()
    q = "SELECT gid, navn, nyttig FROM gave WHERE navn = '" + gave + "';"
    cur.execute(q)
    for row in cur:
        print("Gave: " + str(row[0]) + ", " + row[1] + ", " + str(row[2]))
```

eller følgende metode i Java:

```
void skrivUtGave(Connection conn, String gave) throws SQLException {

    Statement stmt = conn.createStatement();
```

```
    String q = "SELECT gid, navn, nyttig FROM gave WHERE navn = '" + gave + "';"
    ResultSet res = stmt.executeQuery(q);
    while (res.next()) {
        System.out.println("Gave: " + res.getInt(1) + ", "
            + res.getString(2) + ", " + res.getBoolean(3));
    }
}
```

Oppgi en verdi for argumentet `gave` som fører til at funksjonen/metoden heller printer ut informasjon om alle barn (og ingenting annet).

Løsningsforslag

```
gave = "' AND false UNION SELECT * FROM barn;--"
```

3 Relasjonsmodellen og dekomponering

3.1 Nøkler [Fra prøveeksamen H20]

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.

3.2 Tapsfri dekomponering [Fra prøveeksamen H20]

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:

- $AB \rightarrow C$: AB er supernøkkel så bryter ikke med BCNF.
- $AB \rightarrow D$: AB er supernøkkel så bryter ikke med BCNF.
- $CD \rightarrow A$: CD er supernøkkel så bryter ikke med BCNF.
- $CD \rightarrow B$: CD er supernøkkel så bryter ikke med BCNF.
- $A \rightarrow E$: A er ikke en supernøkkel så bryter med BCNF.
  - $A^+ = AE$
  - Dekomponerer R til $S_1(A, E)$ og $S_2(A, B, C, D, F)$
  - For S_1 holder kun FDen $A \rightarrow E$ og har dermed eneste kandidatnøkkel A, og er dermed på BCNF.
  - For S_2 holder alle de andre FDene (1-5) og får dermed kandidatnøkler AB og CD. Må derfor (muligens) dekomponere S_2 videre.
  - Sjekker defor S_2 videre. De fire første FDene bryter ikke med BCNF, men i $C \rightarrow F$ er ikke C supernøkkel for S_2, må derfor dekomponere S_2 videre.
    - $C^+ = CF$
    - Dekomponerer S_2 til $S_{21}(C, F)$ og $S_{22}(C, A, B, D)$.
    - For S_{21} holder kun $C \rightarrow F$ 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 $S_{22}(A, B, C, D)$ med FDene og kandidatnøklene som vist over.