Roskilde Universitetscenter

Skriftlig eksamen i Databaser, Vinter 2001/2002
Opgaver med løsninger

På opfordring har jeg udarbejdet mulige løsninger på eksamensopgaverne, men har ikke haft tid til at polere dem i alle detaljer.

Formuleringerne nedenfor skal ikke forstås som ideelle eksamensbesvarelser. Generelt er det en god idé at skrive mere tekst: Hvis dem, der skal rette en opgave, kan se hvad den studerende har gjort sig af ovejelser, kunne det måske ende med at give næsten fuld point, selvom resultatet er forkert. Hvis f.eks. en opgave kan besvares med a, b, c eller d med a som det rigtige, kan man ikke give mange point for en besvarelse "b", vel? Men hvis der er argumenteret smukt og nydeligt, og det fremgår klart at den studerende ved en nervøs trækning har skrevet et enkelt bogstav forkert i konklusionen, så stiller sagen sig anderledes.

Med venlig hilsen *Henning Christiansen*, 18. marts 2001

Opgave 1 (25%)

Betragt følgende relationer A, B og C angivet med skema og indhold:

A:

a	b	c
1	æble	abe
2	banan	ko
3	kiwi	gris
4	radise	giraf

B:

b	c	d
æble	abe	2
kiwi	gris	3
banan	abe	7

C:

d	e
banan	hamster
kiwi	kiwi
æble	abe
radise	giraf
radise	elefant

Spørgsmål 1

Hvad er resultatet af det relationelle udtryk $\rho_{D(b,c,a)}(B) \cap A$? ($D$ er et tilfældigt relationsnavn forskelligt fra A, B og C).

Svar på spørgsmål 1

Byt rundt på søjlerne for B og sammenlign med A:

a	b	c
3	kiwi	gris

Spørgsmål 2

Hvad er resultatet af det relationelle udtryk $A \bowtie B$?

Svar på spørgsmål 2

a	b	c	d
1	æble	abe	2
3	kiwi	gris	3

Spørgsmål 3

Skriv et relationelt udtryk, som giver de tupler af A af formen $\langle x, y, z \rangle$ hvor $\langle y, z \rangle$ ligger i C.

Svar på spørgsmål 3

Kan gøres på mange måder, f.eks.

$$A \bowtie \rho_{D(b,c)}$$

Spørgsmål 4

Formulér i relational algebra, en begrænsning som siger, at de dyr, som er nævnt i A eller B også skal forekomme i C.

Svar på spørgsmål 4

Den nemmeste måde er vist:

$$\pi_c(A) \cup \pi_c(B) \subseteq \pi_c(C)$$

En mere klodset version, som også vil blive godkendt er:

$$\rho_{D(dyr)}(\pi_b(A)) \cup \rho_{D(dyr)}(\pi_c(B)) \subseteq \rho_{D(dyr)}(\pi_e(C))$$

Spørgsmål 5

Kan igen gøres på mange måder; her er nogle stykker:

$$\pi_d(C \bowtie_{d=d' \wedge e \neq e'} \rho_{D(d',e')}(C))$$

$$\pi_d(\sigma_{d=d' \wedge e \neq e'}(C \times \rho_{D(d',e')}(C)))$$

$$\pi_d(\sigma_{e \neq e'}(C \bowtie \rho_{D(d,e')}(C)))$$

Læg mærke til i den sidste, at det kun er e som ombenævnes til e' men at d bevarer sit navn, så den naturlige join udnyttes fuldt ud.

Opgave 2 (25%)

Denne opgave går ud på, ud fra en uformel beskrivelse, at opbygge et E/R-diagram. Vi ser på et system med studerende som er tilmeldt forskellige kurser, og de kan have gået til eksamen i de respektive kurser og fået en karakter.

Vi har følgende oplysninger:

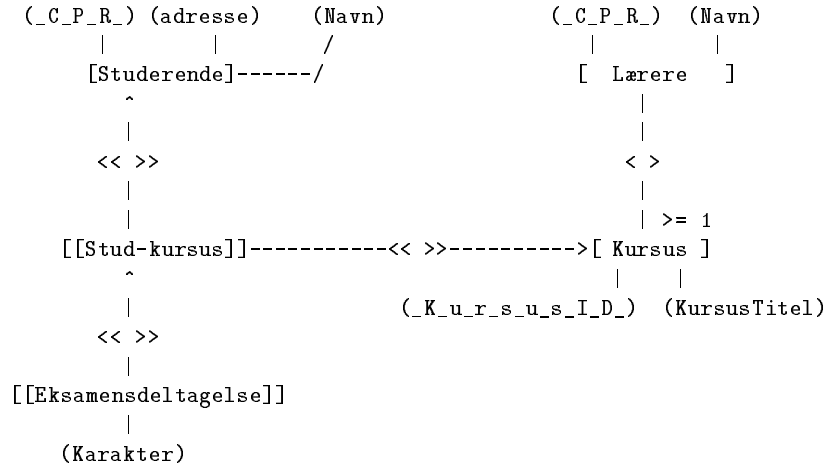
- En studerende har et entydigt CPR-nummer, et navn, en adresse og kan være tilmeldt nul eller flere kurser.
- Et kursus har en titel, en entydig kode (f.eks. DBE01), en eller flere lærere, og nul eller flere studerende har gået til eksamen i kurset.
- En lærer har et entydigt CPR-nummer, et navn, og underviser på nul eller flere kurser.
- En studerende kan være registreret for eksamen i et kursus, som vedkommende er tilmeldt, og har i givet fald fået karakter for denne eksamen. (Men en studerende kan godt være tilmeldt et kursus uden der foreligger en eksamen).

Opgaven lyder: Tegn et E/R-diagram med angivelse af nøgler og eventuelle "weak sets", og som beskriver de indikerede begrænsninger.

Vink: Det kan være en fordel at indføre entitetsmængder, som svarer til "studerende-tilmeldt-kursus" og "eksamens-deltagelse".

Svar på opgave 2

I mangel på egnet tegneprogram, må læseren gætte sig til, hvordan følgende skrivemaskineopstilling skal forstås



Opgave 3 (25%)

Denne opgave handler om SQL og opdatering af relationer. Vi betragter en database som handler om bjørnebanditter og deres tilholdssteder, som vi kalder baser. Databasen har to relationer:

`Bjørnebandit` med to attributter:

> `banditnummer` som er et heltal
>
> `basenummer` som er et heltal

`Base` med tre attributter

> `basenummer` som er et heltal
>
> `længdegrad` som er et heltal mellem `-179` og `180`
>
> `breddegrad` som er heltal mellem `-90` og `90`

Der gælder følgende afhængigheder:

- Nøglen for `Bjørnebandit` er `banditnummer`.
- Nøglen for `Base` er `basenummer`.
- Et `basenummer` for en bjørnebandit i `Bjørnebandit` skal findes i `Base` relationen. (*Enhver bjørnebandit opholder sig på en eksisterende base.*)
- En given base i `Base` skal have tilknyttet mindst én bjørnebandit. (*Baser kan ikke eksistere, hvis der ikke er banditter på dem.*)

To baser kan som udgangspunkt godt have de samme længde- og breddegrader, hvis f.eks. den ene af dem er underjordisk.

NB: Ved bedømmelsen af besvarelserne på de følgende spørgsmål lægges der ikke vægt på den detaljerede syntaks for SQL, men det skal naturligvis kunne genkendes som SQL af læseren.

Spørgsmål 1

Skriv de SQL-sætninger `CREATE TABLE Bjørnebandit` $\cdots$ og `CREATE TABLE Base` $\cdots$ som opretter disse tabeller og medtager de nævnte begrænsninger (dvs. så de blot forhindrer indsættelse eller sletninger som ellers ville ødelægge begrænsningerne).

(NB: Der skal sandsynligvis nogle komplicerede transaktioner til for rent faktisk at få lagt information ind i de tabeller, men det interesserer os ikke i denne opgave.)

Svar på spørgsmål 1

Bemærk: I opgaveformuleringens ånd, er disse løsninger ikke aftestet i Oracle eller lignende, og små syntaktiske variationer kan forekomme.

```
CREATE TABLE Bjørnebandit (
   banditnummer INT,
   basenummer INT REFERENCES Baser(basenummer),
   PRIMARY KEY banditnummer
);

CREATE TABLE Base (
   basenummer INT,
   længdegrad INT CHECK (-179 <= længdegrad AND længdegrad <= 180)
   breddegrad INT CHECK (-90 <= breddegrad AND breddegrad <= 90)
   PRIMARY KEY basenummer
        CHECK (EXISTS (SELECT * FROM Bjørnebandit WHERE
               basenummer = Base.basenummer))
);
```

Spørgsmål 2

Tilpas de to `CREATE`-sætninger fra spg. 1 og indfør evt. triggere om nødvendigt, så vi opnår følgende afledte ændringer ved indsættelse og sletning af tupler:

a) Slettes en tupel i `Base` skal samtlige tupler i `Bjørnebandit`, som refererer til det berørte `basenummer` slettes. (*Dvs. destrueres en base, ryger dens banditter med.*)

b) Slettes den sidste bjørnebandit i `Bjørnebandit`, som referer til et bestemt `basenummer`, skal den tilsvarende `Base`-tupel slettes. (*Dvs. når den sidste bjørnebandit forlader basen, sprænger han den efter sig.*)

c) Oprettes en ny tupel i `Bjørnebandit` med et `basenummer`, som ikke kendes, oprettes en ny `Base`-tupel med "`NULL`"-værdier som `længdegrad` og `breddegrad`.

Løsning på spørgsmål 2

a) Tilføj `ON DELETE CASCADE` til `basenummer` i `Bjørnebandit`.

b) `CHECK`-dimsen i `Base` slettes og tilføj følgende trigger:

```
CREATE TRIGGER Base∅delægger
BEFORE DELETE ON Bjørnebandit
REFERENCING OLD as ExitBanditTupel
WHEN (NOT EXISTS (SELECT * FROM Bjørnebandit
                  WHERE Basenummer = ExitBanditTupel.Basenummer
    AND Banditnummer <> ExitBanditTupel))
      DELETE FROM Base WHERE Basenummer = ExitBanditTupel.Basenummer
FOR EACH ROW
```

Man kan måske også bruge `AFTER DELETE` og i givet fald bliver så kan `SELECT`'en simplere idet "`AND Banditnummer <> ExitBanditTupel`" kan undværes.

Iøvrigt: Har jeg oplevet subtile tekniske problemer med `AFTER` i ORACLE, men selv om Oracle måske ikke vil æde det, så er det OK med en løsning som er konsistent med lærebogen — som f.eks. ovenstående.

b) Kan laves ved denne trigger hvor det er essentielt at der benyttes `BEFORE`:

```
CREATE TRIGGER BaseOpretter
BEFORE INSERT ON Bjørnebandit
REFERENCING NEW as NyBanditTupel
WHEN (NOT EXISTS (SELECT * FROM Base
                  WHERE Basenummer = NyBanditTupel.Basenummer))
      INSERT INTO Base(Basenummer) VALUES (NyBanditTupel.Basenummer)
FOR EACH ROW
```

Opgave 4 (25%)

Vi betragter et databaseskema $R(a, b, c, d, e, f)$ med følgende funktionelle afhængigheder givet.

$$\begin{array}{rcl} a & \rightarrow & f \\ ab & \rightarrow & e \\ de & \rightarrow & c \end{array}$$

Spørgsmål 1

Bestem en nøgle for R og gør rede for, at der kun er den ene nøgle.

Svar på spørgsmål 1

Vi kan slå de to første f.a. sammen til $ab \rightarrow ef$. Vi kan ikke komme fra ab til c eller d, men udfra $de \rightarrow c$ kan vi se at (*) $abd \rightarrow$ "resten" som er cef. Vi kan se, at a ikke kan fjernes uden at (*) ødelægges, tilsvarende for b og e. Ergo er abd en nøgle.

Om der andre nøgler? Der er ingen funktionelle afhængigheder som (via transitivitet) kan føre frem til a, b eller d. Ergo må de være med i en nøgle, og ergo er der ingen andre nøgler end abd.

Spørgsmål 2

Gør rede for, at R ikke er på BCNF og konstruér dernæst nye skemaer, som svarer til R normaliseret til BCNF. Det forventes en begrundelse for, at de nye skemaer er på BCNF.

Svar på spørgsmål 2

Vi kan se, at se, at alle tre f.a. er oppe at slås med BCNF. Dvs, at ingen af venstresiderne er supernøgler.

Vi kan f.eks. tage udgangspunkt i $ab \rightarrow e$, som vi udvider med den første til $ab \rightarrow ef$. Vi kan så efter splitte R op lige efter bogen:

$$R_1(a, b, c, d) \qquad \text{og} \qquad R_2(a, b, e, f)$$

Vi må så kigge efter om R_1 og R_2 er i orden, og derfor må vi kortlægge deres f.a.'er:

R_1**:** Her får vi abd som nøgle, og da vi ikke har nogen $\ldots c \rightarrow \ldots$ er den BCNF.

R_2**:** Er ikke på plads endnu, da vi har $a \rightarrow f$ men a er ikke en nøgle.

Vi hakker nu R_2 op efter $a \rightarrow f$ og får:

$$R_{21}(a, b, e), R_{22}(a, f)$$

Det er nemt at se, at disse overholder betingelsen for BCNF og vi konkluderer at $R_1 + R_{21} + R_{22}$ en opsplitning af R i BCNF!

En anden løsning fås ved at starte dekomposition efter $de \rightarrow c$ så vi får

$$R_1(d, e, c), R_2(a, b, d, e, f)$$

hvor R_2 må dekomponeres efter $a \rightarrow f$:

$$R_{21}(a, b, d, e), R_{22}(a, f)$$

og nu R_{21} efter $ab \rightarrow e$:

$$R_{211}(a, b, e), R_{212}(a, b, d)$$

Altså $R_1 + R_{211} + R_{212} + R_{22}$.