Institut for Matematik og Datalogi
Syddansk Universitet, Odense

8. marts 2000
KSL

DM26 – EKSAMEN
Obligatorisk opgave, del 1

Da dette er en eksamensopgave, er der en række begrænsninger på, hvad der tilladt; lige som til en skriftlig eksamen. Sidst i denne opgave præciseres disse krav nærmere.

Opgaven er stillet onsdag, d. 8. marts 2000 og skal afleveres senest

onsdag, d. 29. marts 2000 kl. 14:15

på IMADA's sekretariat eller til forelæseren.

Denne opgave må løses i grupper på 2 personer.

Opgaven handler om SQL, relationel algebra og relationel kalkyle, som beskrevet i [1]. Desuden skal nogle af SQL forespørgslerne afvikles i PostgreSQL. Der findes en delvis beskrivelse af PostgreSQL i denne opgaveformulering. Det er en del af opgaven selv at finde (vha. man-siderne) eller eksperimentere sig frem til evt. yderligere information, man måtte have brug for i forbindelse med PostgreSQL programmeringen.

Opgave 1

Nedenfor beskrives et antal forespørgsler.

- Alle forespørgslerne skal formuleres i SQL. Man skal så vidt muligt forsøge at bruge den igennem eksempler anbefalede stil fra [1]. Der må ikke være dubletter i svarrelationerne.
- De forespørgsler, der kan (om)formuleres, så de kan afvikles af PostgreSQL (det vil være stort set alle), skal afvikles af PostgreSQL. Man skal selv konstruere passende testdata. Der skal vedlægges en udskrift af en kørsel på formen

```
psql -f filename >& logfile
```

- De forespørgsler, der kan formuleres i relationel algebra, skal angives dels i relationel algebra og dels i enten tupel eller domæne relationel kalkyle. For de resterende forespørgsler gives der en kort forklaring på, hvorfor forespørgslen ikke kan formuleres.

Forespørgslerne drejer sig alle om en database indeholdende tre relationer med følgende skemaer:

- Student(<u>*sid:* **integer**</u>, *name:* **string**, *address:* **string**, *department:* **string**)
- Book(<u>*bid:* **integer**</u>, *author:* **string**, *title:* **string**, *publisher:* **string**, *year:* **integer**, *classification:* **string**, *price:* **integer**, *duration:* **integer**)
- Loan(<u>*bid:* **integer**, *sid:* **integer**</u>, *date:* **datetype**)

Det drejer sig altså om et universitetsbibliotek, hvor der er en række studerende (Student), der kan være lånere. Disse har en entydig identifikation (sid), et navn (name) og en adresse (address). Desuden er deres hjemmeinstitut angivet.

Biblioteket har også alle bøger registreret (Book). Disse har en entydig identifikation (bid), samt angivelse af forfatter(e) (author), titel (title), forlag (publisher), udgivelsesår (year), klassifikation med henblik på korrekt anbringelse på hylderne (classification), anskaffelsespris (price), samt maksimal udlånstid (duration).

Endelig er der en relation (Loan), der angiver hvilke studerende, der i øjeblikket er lånere af de udlånte bøger, samt hvornår de lånte bøgerne.

Feltet date er af typen datetype. Vi antager, at der findes en funktion today, der leverer dags dato. Desuden har man rimelige operationer på typen. F.eks. må man skrive $d+50$, hvor d er af typen datetype. Resultatet er datoen 50 dage senere end d. Man kan naturligvis også benytte alle sammenligningsoperationerne på typen.

De forespørsler, opgaven drejer sig om, opremses nedenfor.

1. Find navnene på de studerende, der har lånt en bog, som har kostet mindst 1000 kr.
2. Find sid på de studerende, der har lånt bøger til en samlet værdi af højst 1000 kr.
3. For at finde ud af, hvilke fagområder der er "dyrest i drift", vil vi interessere os for den gennemsnitlige værdi af studerendes boglån separat for hvert institut. Find det maksimale gennemsnitlige beløb.
4. Find de institutter, hvis lånere samlet har lånt for det mindste beløb.
5. Find sid på de studerende, der har lånt præcis to bøger.
6. Lav en relation af alle par (sid,cnt), hvor cnt er antallet af bøger, som den studerende sid har lånt.
7. Find sid på de studerende, der enten har lånt en gratis bog, eller har lånt én, der kostede mere end 10000 kr.
8. Find sid på de studerende, der har lånt alle bøger om databasesystemer (klassifikationen DBMS).
9. Find sid på de kunder, der kun låner bøger med en pris på mindst 1000 kr., med mindre bogen har været gratis.
10. Lav en relation bestående af alle de par (sid,bid), hvor bogen bid er mere værd end gennemsnittet for sid's boglån.
11. Find alle de bøger, der ikke er blevet afleveret til tiden.

Opgave 2

Når forespørgsler skal afvikles, er det interessant at kunne ændre dem til ækvivalente forespørgsler, der evt. kan afvikles mere effektivt. Nedenfor ser vi på mulige kandidater til omskrivningsregler.

Relationerne R, S og T har samme skema, og C og D er boolske udtryk over R-tupler.

1. $(R \cap S) - T \equiv R \cap (S - T)$
2. $R \cap (S \cup T) \equiv (R \cap S) \cup (R \cap T)$
3. $\sigma_{C \vee D}(R) \equiv \sigma_C(R) \cup \sigma_D(R)$

Tegnet $\equiv$ betyder, at de to udtryk er ækvivalente. Dvs. at de uanset indholdet af relationerne altid giver samme resultat.

Bevis de tre ækvivalenser. Beviset for hvert ækvivalens struktureres som to delbeviser, hvor man viser, at hvis et tupel tilhører udtrykket på den ene side, så tilhører det også udtrykket på den anden side.

I SQL tillader man til tider dubletter. Man kunne overveje, at formalisere operationer på relationer med dubletter, så man kunne sige noget om, hvor store relationerne blev. En rimelig måde at gøre det på er følgende:

Se på et fastholdt tupel t. Antag, at R_1 indeholder k_1 t-tupler, og at R_2 indeholder k_2 t-tupler.

Da er indholdet af t-tupler i følgende udtryk som følger:

Udtryk	Antal t-tupler
$R_1 \cup R_2$	$k_1 + k_2$
$R_1 \cap R_2$	$\min\{k_1, k_2\}$
$R_1 - R_2$	$\max\{k_1 - k_2, 0\}$
$\sigma_C(R_1)$	0 eller k_1

I sidste tabelindgang er resultatet k_1, hvis t-tuplet opfylder betingelsen C.

Ovenstående skal så naturligvis gælde for alle tupler.

To relationer er nu i denne formalisme ens, hvis de indeholder samme antal af alle tupler, og to udtryk er så ækvivalente, hvis de altid evaluerer til to ens relationer.

Bevis, at de tre ækvivalenser givet først i denne opgave *ikke* holder for denne udgave med dubletter. Et bevis kan bestå af et lille eksempel på et konkret indhold af relationerne, og en forklaring på hvorfor de to sider evaluerer til forskellige relationer.

Postgresql

PostgreSQL er et af de sprog, der understøtter en "extended subset" af SQL.

PostgreSQL har en hjemmeside; det er dog ikke én, I har nogen grund til at besøge, med mindre I vil vide mere eller vil forsøge at hente systemet hjem til jeres egen PC. URL'en er: `http://www.postgresql.org/`.

Kun studerende, der optræder på deltagerlisten har fået en PostgreSQL-konto. Den aktuelle deltagerliste kan findes på:

`http://www.imada.sdu.dk/~kslarsen/dm26/deltagere.html`

Før PostgreSQL kan startes, er der et par ting, der skal sættes op. I den fil, hvor jeres `path` variabel defineres (for de flestes vedkommende drejer det sig om `.tcshrc` filen), gøres følgende:

- Directory'et `/home/POSTGRES/bin` føjes til jeres `path`.
- Følgende tre linier tilføjes filen:

```
setenv MANPATH /home/POSTGRES/man:/usr/local/man:/usr/man
setenv PGHOST helena
setenv LD_LIBRARY_PATH /home/POSTGRES/lib
```

Log nu ud og log in igen.

Skriv nu først `createdb`. Derved får I lavet jer en database. Derefter kan POSTGRESQL startes ved at skrive `psql`.

Dokumentation findes på `man`-siderne. Prøv f.eks. `man select` og `man sql`.

Inde i `psql` er der on-line hjælp at hente. Her er nogle små eksempler og vink til at komme i gang.

- Eksempel på en relationsdefinition:

  ```
  create table employees(name text, age integer);
  ```

- Tupler kan indsættes i ovenstående relation som følger:

  ```
  insert into employees(name, age) values('Larsen', 37);
  ```

- Relationen fjernes igen ved at skrive:

  ```
  drop table employees;
  ```

- Resultatet af en forespørgsel kan navngives f.eks. således:

  ```
  select R.A, count(R.B) as C
  into table NewRelation
  from Relation R
  group by R.A;
  ```

  På den måde kan man vha. en sekvens af forespørgsler nogle gange løse et problem, der, hvis det var formuleret som én forespørgsel, ville involvere nested'e anvendelser af `select`.

- Kommandoen `\z` giver en liste over alle relationer.

- Med kommandoen `\q` forlader man POSTGRESQL.

Det vil nok være en fordel at have alle sine kommandoer på en fil; specielt indtastningen af data i relationer. Kommandoer på en fil `filename` kan køres ind i systemet vha. UNIX-kommandoen `psql -f filename`.

Med henblik på at kunne dokumentere, at ens forespørgsler virker, kan man bruge `psql -f filename >& logfile`, der bevirker at både input fra `filename` samt det tilhørende output skrives ud på `logfile`.

POSTGRESQL kører kun på vores LINUX-maskiner. Sidder man ikke på sådan én, kan man altid lave en `rlogin` til én.

Generelle krav i forbindelse med eksamensprojekter

Status

En obligatorisk opgave er en eksamensform. Alle sædvanlige regler for afvikling af eksaminer gælder derfor også for obligatoriske opgaver. Den eneste væsentlige forskel fra en almindelig eksamen er det manglende opsyn. Det ikke tilladt på nogen form at modtage eller afgive information i forbindelse med opgaven med mindre det forudgående eksplicit er blevet tilladt af den ansvarlige lærer. Diskussion, afskrift eller elektronisk kopiering af dele af opgaverne vil blive betragtet og behandlet som eksamenssnyd. Man er forpligtiget til at beskytte sine filer mod at andre kan læse dem (f.eks. vha. `chmod`). Hvis du ikke ved, hvordan man gør, så spørg.

Er du i tvivl om, hvad du må, så tænkt på, om det ville være i orden til en normal eksamen.

Hvis eksamensprojektet laves i grupper har alle gruppens medlemmer på alle måder fuldt ansvar for alt, hvad der afleveres.

Besvarelsen

Besvarelsen skal være velstruktureret og opdelt logisk i afsnit. Der skal skrives præcist og klart.

Der skal læses grundig korrektur på besvarelsen. Der forventes et sprog, herunder bl.a. stavning og tegnsætning, af en rimelig kvalitet. Gør opmærksom på, hvis der er særlige forhold, der er skyld i, at ens besvarelse ikke kan blive sprogligt fejlfri.

Besvarelsen skal udfærdiges på tekstbehandlingsanlæg og udskrives på en højkvalitetsprinter. Udskriften skal være tydelig. Sider, der er skrevet ud med for lav toner, slidt farvebånd eller andet, betragtes som ikke-afleveret.

På forsiden skal der være fuldt navn, fødselsdato og brugernavn (login).

Programmer

Programmer skal være velstrukturerede, letlæselige og kommenterede. Kræves programudskrift vedlagt besvarelsen, gælder der som hovedregel samme regler for denne som for udskrift af besvarelsen.

Hvis der både kræves elektronisk aflevering af programmer samt aflevering af programudskrift, skal udskriften og de elektronisk afleverede programmer være identiske.

Aflevering

Der gives altid betragteligt mere tid til opgaverne end nødvendigt. Til gengæld accepteres for sent afleverede opgaver ikke; heller ikke bare 5 minutter!

Man bør lave og aflevere opgaven mindst 48 timer før afleveringsfristen. At systemet eller alle printere eksempelvis er nede de sidste 24 timer før afleveringsfristen betragtes ikke nødvendigvis som en legal undskyldning.

Der gælder samme regler for en evt. elektronisk aflevering.

Litteratur

[1] Raghu Ramakrishnan. *Database Management Systems*. McGraw-Hill, 1998.

Kvittering for aflevering af
Obligatorisk opgave, del 1, DM26, foråret 2000

Udfyldes af de(n) studerende inden afleveringen:

Navne: ....

Fødselsdatoer: ....

Brugernavne: ....

Udfyldes af sekretariatet eller forelæseren:

Modtaget den .... kl. .... af ....
(dato) (klokkeslet) (initialer)