Afleveringsopgave

Baseret på opgavetekst forfattet af Troels Andreasen, forår 2001
Let redigeret af Henning Christiansen, oktober 2001

Aflevering

Opgaven afleveres senest mandag d. 5. november 2001 kl. 10.00 i Henning Christiansens postbakke i hus 42.1 (elektroniske besvarelser ønskes ikke); opgaven skulle dog ikke være større end de fleste skulle kunne aflevere den i god ro og orden inden weekended. Hvor det er relevant skal svar præsenteres som kørsler i SQL*Plus mod en Oracle-database med et indhold svarende til dataene i `modul2.bibdata` (se nedenfor). Sæt venligts lidt forklarende tekst rund om, men ikke noget der tangerer en projektrapport. Spar på papiret: Regn med at der er fejl i jeres SQL-udtryk, hvis I får meget store svar. Brug f.eks. SQL*Plus formattering såsom ”column tit format a40” og ”set pagesize 300”.

Nedenfor følger først en opgaveformulering med at antal spørgsmål og til sidst nogle få, nyttige teknikaliteter. NB: Den omtalte database, som skal benyttes som udgangspunkt for opgavens løsning garanteres først at være på plads 24. oktober.

Opgaveformulering

I denne opgave skal der eksperimenteres med et "rigtigt" databasesystem og der skal fokuseres på et konkret applikationsbehov, til hvilket vi skal konstruere en *database*.

Databasen skal holde styr på en samling bøger til et bibliotek. Udgangspunktet for designet af et skema til databasen er en beskrivelse af en række egenskaber, der skal registreres oplysninger om. Egenskaberne og et forslag til hvordan de kan navngives er følgende:

id	entydigt nummer for bogen
tit	bogens titel
subtit	bogens undertitel
isbn	bogens isbn-nummer
aar	bogens udgivelsesår
dk5	bogens klassifikation i dk5
udid	udgiverens identifikationsnummer
udnavn	udgiverens navn
fid	en forfatters identifikationsnummer
fnavn	forfatterens navn
oid	nummer på et emneord tilknyttet bogen
ord	emneord tilknyttet bogen

De fleste af egenskaberne kræver ikke uddybende forklaring. Hvad angår ”dk5” skal det nævnes, at det er klassifikationen i det meget anvendte system DK5 (der bl.a. er udbredt på folkebibliotekerne). Det er et hierarkisk system af klasser, der repræsenterer en emnemæssig inddeling. Systemet har en hierarkisk nummerering (f.eks. 37.1 og 37.2 er underklasser til 37) og det er kun denne nummerering – ikke beskrivelsen af den – som vi skal beskæftige os med her. Bøger karakteriseres ved at tilknytte en klasse (bøger klassificeres). Endvidere karakteriseres bøger ved at tilknytte en række emneord fra et emneords-vokabularium (bøger indekseres), der her er Dansk BiblioteksCenters emneordsliste. Det skal bemærkes at klassifikationen og indekseringen er to beskrivelsesformer, der i udgangspunktet ikke har noget med hinanden at gøre.

Betragt dette som en kravspecifikation fra kunden. I den anden ende – i databasen – kunne de nævnte egenskaber så evt. optræde som attributter.

Hermed kan man sige at opgaven er temmelig konkret og at et bud på et relationelt skema til en database til formålet er:

bibdata(id, tit, subtit, isbn, aar, dk5, udid, udnavn, fid, fnavn, oid, ord)

Imidlertid er der allerede i listen over egenskaber ovenfor noget der tyder på, at dette måske ikke er den smarteste løsning.

Problem 1

Raffinering af skemaet – normalisering.

Punkt 1.A

På baggrund af spørgsmål til kunden er det blevet afklaret at der må gælde følgende funktionelle afhængigheder:

F1: id -> tit, subtit, isbn, aar, dk5, udid
F2: udid -> udnavn
F3: fid -> fnavn
F4: oid -> ord

Bestem, udfra disse afhængigheder, en nøgle for det simple relationsskema bibdata.

Punkt 1.B

Prøv endvidere med udgangspunkt i bibdata og de nævnte afhængigheder, at nå frem til et bedre bud, ved at normalisere bibdata til Boyce-Codd normalform.

Punkt 1.C

Detaljér endelig det fremkomne skema ved at angive nøgler for alle relationer.

Problem 2

Nu har vi et skema, der er fremkommet på en – måske lidt atypisk – måde. Angiv, af hensyn til overskueligheden, et design i ER-notation, der ved oversættelse kan give anledning til det relationelle skema fra problem 1.C.

Problem 3

Detaljér løsningen på problem 1 yderligere ved at skrive et tilsvarende skema noteret i SQL-DDL. Sørg for at medtage nøgler og at få formuleret begrænsninger svarende til referentiel integritet i det detaljerede skema.
Opret herefter en database i Oracle svarende hertil (dvs. udfør skemaet i Oracle) og vis ved eksempler ("kørsler") at Oracle håndhæver integriteten.
BEMÆRK at Problem 4 har bud på attributtyper – brug disse eller vælg evt. nogle mindre generelle.
BEMÆRK også at Problem 3 og 4 kan løses under et, idet man kan vælge konstruktioner som ”create table TT as select ...” i stedet for ”create table TT as (attr-navn attr-type, ...). Vælges denne fremgangsmåde er det dog nødvendigt også at udføre nogle ”alter table ...” for at få begrænsningerne med. SE TEKNISKE HINTS TIL SIDST.

Problem 4

Der skal nu tilføjes data til databasen. Hertil er der i en Oracle database oprettet en relation svarende til "bibdata" ovenfor. Relationen omfatter knap 40.000 tupler, der indeholder information om "rigtige" bøger (der er tale om et udsnit af bøger som er registreret i DanBib og som findes på bl.a. folkebiblioteker). Relationen hedder "bibdata" og I kan finde den under bruger "modul2". Dens skema fremgår af:

```
SQL> describe modul2.bibdata
 Navn                              NULL?    Type
 --------------------------------- -------- ------------------
 ID                                         NUMBER(38)
 TIT                                        VARCHAR2(300)
 SUBTIT                                     VARCHAR2(300)
 ISBN                                       VARCHAR2(20)
 AAR                                        NUMBER(38)
 DK5                                        VARCHAR2(25)
 UDID                                       NUMBER(38)
 UDNAVN                                     VARCHAR2(100)
 FID                                        NUMBER(38)
 FNAVN                                      VARCHAR2(100)
 OID                                        NUMBER
 ORD                                        VARCHAR2(100)
```

```
SQL>
```

Fra jeres egen Oracle-bruger kan I referere relationen ved blot at notere modul2 som prefiks, som det gøres i dette eksempel, hvor man gemmer samtlige årstal i en selvstændig relation:

```
create table test(aar number(4));
insert into test
select distinct aar from modul2.bibdata;
```

Problem 5

Forespørgsler til databasen.

Punkt 5.A

Hvem (vis forfatter-navn) har fået udgivet bøger hos udgiveren med navnet 'Haase' siden 1/1-1994.

Punkt 5.B

Find de største udgivere. Vis med navne og antal bøger de udgivere, der har mindst 60 udgivelser. Ordn således at udgivere med flest bøger kommer først.

Punkt 5.C

Beskriv "sammenhængen" imellem udgiverne 'Teknisk Forlag' og 'Socialforskningsinstituttet' ved emneord, idet der vises ord, som er anvendt i indeksering af bøger fra begge udgivere.

Punkt 5.D

Hvilke forfattere har fået udgivet bøger hos mere end fire forskellige udgivere?

Punkt 5.E

Vis forfattere der har mindst 10 bøger i databasen.

Punkt 5.F

Karakterisér forfattere der har mindst 10 bøger i databasen, ved at angive ord, der er brugt mindst 4 gange i indeksering af deres bøger.

Problem 6

Der skal nu konstueres en hjælperelation med redundant data, som kan være anvendelig i forbindelse med udskrivning. Denne skal have skemaet pbog(id, p1, p2), hvor id er et identifikationsnummer på en bog og p1 og p2 er to "print-linier". p1 skal være sammensat af forfatternavn og titel (konkateneret) og p2 skal indeholde alle de emneord som beskriver bogen. Emneordene i p2 skal være komma-separeret.

Punkt 6.A

Skriv en PL/SQL procedure, der tager et id-nummer som argument og opretter en tupel i pbog som skitseret. Det er en god ide at bruge en PL/SQL cursor.

Punkt 6.B

Skriv nu et "program" (en PL/SQL-blok), der indsætter en tupel i pbog-relationen for hver bog i databasen.

Punkt 6.C

Skitsér endelig hvordan pbog-relationen kan vedligeholdes automatisk af databasesystemet ved hjælp af en trigger.

Problem 7

Endelig skal programmeres en "applikation" til søgning efter bøger. Funktionaliteten skal være som følger. Der skal understøttes to former for søgning - en "Bedste bud" udfra en specifikation af en liste af emneord samt en "lignende bøger" udfra en given bog.

"Bedste bud": Der skal kunne indtastes et eller flere emneord som en forespørgsel og der skal ud fra disse dannes et svar, der omfatter de 10 (eventuelt ca. 10) bedste bud på bøger i databasen ordnet efter "bedste først". En bog kan være med i svaret, hvis mindst et emneord fra denne er med i ("match'er") forespørgslen. De bedste bøger er dem der match'er flest emneord fra forespørgslen.

"Nærmeste bøger": Der skal kunne angives en bog (evt. blot ved indtastning af id for denne) som forespørgsel og udfra denne dannes et svar bestående af de 10 "nærmeste" bøger. Jo flere emneord to bøger har tilfælles, jo nærmere er de hinanden.

Det er funktionaliteten og hvordan den realiseres ved manipulation i databasen, der er afgørende her. Det har kun mindre betydning hvorledes funktionalitet aktiveres i grænsefladen. De to former for søgning kan evt. implementeres som to selvstændige "programmer".

Applikations-programmet/programmerne kan blot dannes som et script skrevet i PL/SQL, som kan køre direkte under SQL*Plus.

Tekniske hints

Sådan som vi har lært det på kurset, må man angive et detaljeret skema incl. begrænsninger før man begynder at hælde data ind i relationen, f.eks:

```
CREATE TABLE Studio (
    name CHAR(30) PRIMARY KEY,
    address VARCHAR(255),
    pressC# INT REFERENCES MovieExec(cert#)
   );
...
INSERT INTO Studio VALUES( 'Disney', ... );
```

Når man som her baserer de relationer, man selv opretter, på en relation, som allerede findes kan man så at sige få overført typerne for attributterne til den nye ved at bruge en variant af `CREATE` som benytter sig af en `SELECT`-statement, til at udregne relationens "startværdi" og dens skema bliver også skabt ved denne udregning, f.eks.:

```
CREATE TABLE Studio AS
    (SELECT name, address, pressC# FROM SomeGivenRelation);
...
```

Begrænsninger kan så hægtes på bagefter vha. `ALTER`:

```
ALTER TABLE Studio ADD PRIMARY KEY(name);
```

Problem 7 og 8 kræver en smule indlæsning og udskrift, som man kan behandle med de primitive redskaber som PL/SQL stiller til rådighed (omend man nok ikke skal basere realistiske applikationer derpå!!).

Udskrift kan foregå ved at man genererer en midlertidig tabel, som script'et skriver ud, eller man kan slå `serveroutput` til og skrive ud med `dbms_output.put_line` som i følgende eksempel; `S` forventes at være en relation med en attribut, som hedder `sname`.

```
SQL> set serveroutput on
SQL> list
1 DECLARE
2 CURSOR c IS SELECT * FROM S;
3 srec s%rowtype;
4 BEGIN
5 FOR srec IN c LOOP
6 dbms_output.put_line('Et navn er f.eks. '||srec.sname||'.');
7 END LOOP;
8* END;
SQL> /
Et navn er f.eks. Smith .
Et navn er f.eks. Jones .
Et navn er f.eks. Blake .
Et navn er f.eks. Clark .
Et navn er f.eks. Adams .
```

```
PL/SQL procedure successfully completed.
```