**Datalogi** **Database-kurset**
**Efter r 2001**

# Afleveringsopgave

Baseret p opgavetekst forfattet af Troels Andreasen, for r 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 ¿nskes ikke); opgaven skulle dog ikke v re st¿rre end de fleste skulle kunne aflevere den i god ro og orden inden weekended. Hvor det er relevant skal svar pr senteres som k¿rsler i SQL*Plus mod en Oracle-database med et indhold svarende til dataene i `modul2.bibdata` (se nedenfor). S t venligts lidt forklarende tekst rund om, men ikke noget der tangerer en projektrapport. Spar p papiret: Regn med at der er fejl i jeres SQL-udtryk, hvis I f r meget store svar. Brug f.eks. SQL*Plus formattering s som column tit format a40 og set pagesize 300 .

Nedenfor f¿lger f¿rst en opgaveformulering med at antal sp¿rgsm l og til sidst nogle f , nyttige teknikaliteter. NB: Den omtalte database, som skal benyttes som udgangspunkt for opgavens l¿sning garanteres f¿rst at v re p plads 24. oktober.

## Opgaveformulering

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

Databasen skal holde styr p en samling b¿ger til et bibliotek. Udgangspunktet for designet af et skema til databasen er en beskrivelse af en r kke egenskaber, der skal registreres oplysninger om. Egenskaberne og et forslag til hvordan de kan navngives er f¿lgende:

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

De fleste af egenskaberne kr ver ikke uddybende forklaring. Hvad ang r dk5 skal det n vnes, at det er klassifikationen i det meget anvendte system DK5 (der bl.a. er udbredt p folkebibliotekerne). Det er et hierarkisk system af klasser, der repr senterer en emnem ssig 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 besk ftige os med her. B¿ger karakteriseres ved at tilknytte en klasse (b¿ger klassificeres). Endvidere karakteriseres b¿ger ved at

tilknytte en r kke emneord fra et emneords-vokabularium (b¿ger indekseres), der her er Dansk BiblioteksCenters emneordsliste. Det skal bem rkes at klassifikationen og indekseringen er to beskrivelsesformer, der i udgangspunktet ikke har noget med hinanden at g¿re.

Betragt dette som en kravspecifikation fra kunden. I den anden ende — i databasen — kunne de n vnte egenskaber s evt. optr de som attributter.
Hermed kan man sige at opgaven er temmelig konkret og at et bud p et relationelt skema til en database til form let 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 p , at dette m ske ikke er den smarteste l¿sning.

## Problem 1

Raffinering af skemaet — normalisering.

### Punkt 1.A

P baggrund af sp¿rgsm l til kunden er det blevet afklaret at der m g lde f¿lgende funktionelle afh ngigheder:

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

Bestem, udfra disse afh ngigheder, en n¿gle for det simple relationsskema bibdata.

### Punkt 1.B

Pr¿v endvidere med udgangspunkt i bibdata og de n vnte afh ngigheder, at n frem til et bedre bud, ved at normalisere bibdata til Boyce-Codd normalform.

### Punkt 1.C

Detalj r endelig det fremkomne skema ved at angive n¿gler for alle relationer.

## Problem 2

Nu har vi et skema, der er fremkommet p en — m ske lidt atypisk — m de. Angiv, af hensyn til overskueligheden, et design i ER-notation, der ved overs ttelse kan give anledning til det relationelle skema fra problem 1.C.

## Problem 3

Detalj r l¿sningen p problem 1 yderligere ved at skrive et tilsvarende skema noteret i SQL-DDL. S¿rg for at medtage n¿gler og at f formuleret begr nsninger svarende til referentiel integritet i det detaljerede skema.
Opret herefter en database i Oracle svarende hertil (dvs. udf¿r skemaet i Oracle) og vis ved eksempler ("k¿rsler") at Oracle h ndh ver integriteten.
BEMfiRK at Problem 4 har bud p attributtyper — brug disse eller v lg evt. nogle mindre generelle. BEMfiRK ogs at Problem 3 og 4 kan l¿ses under et, idet man kan v lge konstruktioner som create table TT as select ... i stedet for create table TT as (attr-navn attr-type, ...). V lges denne fremgangsm de er det dog n¿dvendigt ogs at udf¿re nogle alter table ... for at f begr nsningerne med. SE TEKNISKE HINTS TIL SIDST.

## Problem 4

Der skal nu tilf¿jes 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" b¿ger (der er tale om et udsnit af b¿ger som er registreret i DanBib og som findes p bl.a. folkebiblioteker). Relationen hedder "bibdata" og I kan finde den under bruger "modul2". Dens skema fremg r 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 g¿res i dette eksempel, hvor man gemmer samtlige rstal i en selvst ndig relation:

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

## Problem 5

Foresp¿rgsler til databasen.

### Punkt 5.A

Hvem (vis forfatter-navn) har f et udgivet b¿ger hos udgiveren med navnet Haase siden 1/1-1994.

### Punkt 5.B

Find de st¿rste udgivere. Vis med navne og antal b¿ger de udgivere, der har mindst 60 udgivelser. Ordn s ledes at udgivere med flest b¿ger kommer f¿rst.

### Punkt 5.C

Beskriv sammenh ngen imellem udgiverne 'Teknisk Forlag' og 'Socialforskningsinstituttet' ved emneord, idet der vises ord, som er anvendt i indeksering af b¿ger fra begge udgivere.

### Punkt 5.D

Hvilke forfattere har f et udgivet b¿ger hos mere end fire forskellige udgivere?

### Punkt 5.E

Vis forfattere der har mindst 10 b¿ger i databasen.

### Punkt 5.F

Karakteris r forfattere der har mindst 10 b¿ger i databasen, ved at angive ord, der er brugt mindst 4 gange i indeksering af deres b¿ger.

## Problem 6

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

### 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 inds tter en tupel i pbog-relationen for hver bog i databasen.

### Punkt 6.C

Skits r endelig hvordan pbog-relationen kan vedligeholdes automatisk af databasesystemet ved hj lp af en trigger.

## Problem 7

Endelig skal programmeres en "applikation" til s¿gning efter b¿ger. Funktionaliteten skal v re som f¿lger. Der skal underst¿ttes to former for s¿gning - en "Bedste bud" udfra en specifikation af en liste af emneord samt en "lignende b¿ger" udfra en given bog.

"Bedste bud": Der skal kunne indtastes et eller flere emneord som en foresp¿rgsel og der skal ud fra disse dannes et svar, der omfatter de 10 (eventuelt ca. 10) bedste bud p  b¿ger i databasen ordnet efter "bedste f¿rst". En bog kan v re med i svaret, hvis mindst et emneord fra denne er med i ("match'er") foresp¿rgslen. De bedste b¿ger er dem der match'er flest emneord fra foresp¿rgslen.

"N rmeste b¿ger": Der skal kunne angives en bog (evt. blot ved indtastning af id for denne) som foresp¿rgsel og udfra denne dannes et svar best ende af de 10  n rmeste  b¿ger. Jo flere emneord to b¿ger har tilf lles, jo n rmere er de hinanden.

Det er funktionaliteten og hvordan den realiseres ved manipulation i databasen, der er afg¿rende her. Det har kun mindre betydning hvorledes funktionalitet aktiveres i gr nsefladen. De to former for s¿gning kan evt. implementeres som to selvst ndige "programmer".

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

## Tekniske hints

S dan som vi har l rt det p  kurset, m  man angive et detaljeret skema incl. begr nsninger f¿r man begynder at h lde 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', ... );
```

N r man som her baserer de relationer, man selv opretter, p  en relation, som allerede findes kan man s  at sige f  overf¿rt typerne for attributterne til den nye ved at bruge en variant af `CREATE` som benytter sig af en `SELECT`-statement, til at udregne relationens "startv rdi" og dens skema bliver ogs  skabt ved denne udregning, f.eks.:

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

Begr nsninger kan s  h gtes p  bagefter vha. `ALTER`:

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

Problem 7 og 8 kr ver en smule indl sning og udskrift, som man kan behandle med de primitive redskaber som PL/SQL stiller til r dighed (omend man nok ikke skal basere realistiske applikationer derp !!).

Udskrift kan foreg ved at man genererer en midlertidig tabel, som script'et skriver ud, eller man kan sl `serveroutput` til og skrive ud med `dbms_output.put_line` som i f¿lgende eksempel; `S` forventes at v re 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.
```