Extra tentamen i DATABASER

Svar:

Obs! Lärare-version, med lösningar

DAG: fr, 21 jan 2000 **TID:** kl. 14 – 18 **SAL:** grupprum 7, Matematiskt Centrum

Ansvarig:	Martin Weichert
Förfrågningar:	Martin Weichert, tel. 772 10 68
Hjälpmedel:	utdraget ur *Oracle7 Server SQL Language* ("det gula pappret") resp. *Appendix B Syntax* ur referensmanualen för *Oracle SQL* samt *SQL*Plus Quick Reference.*

Lycka till!

Uppgift 1.8p (a) Ange tre orsaker hur felaktigheter i en databas kan uppstå. Till varje av dem, ange en lämplig metod som används som åtgärd mot dessa.

Svar:

- Redundans, motsägande data på olika ställen → Normaliseringar
- Nonsens-data, t ex "31 juni" → typvillkor, undertypvillkor, inklusionsvillkor
- Tomma referenser → referensvillkor
- Ofullständiga ändringar → transaktionskonceptet
- Datorkrasch → loggfil, återställningsmetoder (fördröjd/omedelbar ändring)
- Parallellism, flera användare samtidigt → låsning (2-fas-, konservativ, strikt, ...), serialiserbarhet
- Obehörigt tillträde → säkerhetsmekanismer ("discretionary", "mandatory")

– **5p.**

(c) Vad är skillnaden mellan SQL-satserna `drop table ANSTÄLLDA` och `delete from ANSTÄLLDA`?

Svar: `delete` tömmer (rensar) tabellen *ANSTÄLLDA*, d v s tar bort hela **innehållet** ur tabellen, men lämnar kvar tabellstrukturen (namn och typer på fält, nyckel-, referens- och andra villkor). Med `drop` tas bort även allt information om tabellstrukturen. – **1p.**

9 poäng.

Uppgift 2.9p Låt $r(A, B, C)$ och $s(B, C, D)$ vara tabeller, a ett element av A:s domän (datatyp), b av B:s domän osv. Attribut med samma namn har alltså samma domän, attribut med olika namn har olika domän.

Vilka av de följande är giltiga relationsalgebra-uttryck?

(a) $r \cup s$

(b) $\pi_B(r) - \pi_B(s)$

(c) $\sigma_{B=b_1}(r)$

(d) $\sigma_{A=a_1 \wedge B=b_1}(s)$

(e) $r \bowtie s$ (naturlig samkörning)

(f) $\pi_A(r) \times \pi_D(s)$

För de uttryck som är ogiltiga, motivera varför! För de giltiga uttrycken, beräkna resultatet, med tabellerna r och s givna som nedan. (Beakta att resultatet är **mängder**!)

r:

A	B	C
a_0	b_0	c_0
a_0	b_1	c_1
a_0	b_2	c_1
a_1	b_1	c_0

s:

B	C	D
b_1	c_1	d_0
b_2	c_1	d_1
b_2	c_0	d_0

Svar:

(a) $r \cup s$: ogiltigt (icke mängd-kompatibla – namn och domäner av attributen stämmer inte överens) – **1p.**

(b)

(c) $\pi_B(r) - \pi_B(s)$:

B
b_0

– **2p.**

$\sigma_{B=b_1}(r)$:

A	B	C
a_0	b_1	c_1
a_1	b_1	c_0

– **1p.**

(d) $\sigma_{A=a_1 \wedge B=b_1}(s)$: ogiltigt ($s$ har inget A) – **1p.**

(e)

(f) $r \bowtie s$:

A	B	C	D
a_0	b_1	c_1	d_0
a_0	b_2	c_1	d_1

– **2p.**

$\pi_A(r) \times \pi_D(s)$:

A	D
a_0	d_0
a_0	d_1
a_1	d_0
a_1	d_1

– **2p.**

Poängavdrag (– **1p.**) om det finns dubletter.

9 poäng.

Uppgift 3.12p En skivaffär vill hålla reda på sina skivor och använder en databas med följande tabeller:

- *skiva(Nr, Titel, Typ, Kategori, Land, Pris)*
  En lista över alla skivor, med ett nummer *Nr* som unikt identifierar varje skiva, *Titel*, *Typ* ('LP', 'CD' eller dyl.), *Kategori* ('klassisk', 'folkmusik' eller dyl.), *Land* (landet där skivan producerades) och *Pris*.
- *artist(Namn, Född, Land)*
  En lista över alla artister som medverkade i skivorna, med *Namn* (antas vara unikt), födelsedatum *Född* och *Land* där artisten föddes.
- *medverkar(Nr, Artist)*
  Lista över alla artister som medverkar i någon skiva. *Nr* är referens till *skiva*, *Artist* är referens till *artist*.
- *sålda(Nr, År, Månad, Antal)*
  En lista över hur många exemplar av varje skiva som såldes i *Månad* i *År*. *Nr* är referens till *skiva*.

Skriv SQL-satser för följande uppgifter:

(a) Lista nummer och titel på alla skivor där John Lennon medverkar.

Svar:

```
select NR, TITEL
  from SKIVA
 where NR in (select NR
                from MEDVERKAR
               where ARTIST = 'John Lennon');
```

eller

```
select NR, TITEL
  from SKIVA S, MEDVERKAR M
 where S.NR = M.NR and ARTIST = 'John Lennon';
```

– **2p.** Obs – villkoret `S.NR = M.NR` får inte glömmas!

(b) Lista nummer och titel på alla skivor där *bara* John Lennon medverkar.

Svar:

```
select NR, TITEL
  from SKIVA
 where NR in (select NR
                from MEDVERKAR
               where ARTIST = 'John Lennon')
and NR not in (select NR
                from MEDVERKAR
               where ARTIST <> 'John Lennon');
```

eller

```
select NR, TITEL
  from SKIVA S
 where (select ARTIST
          from MEDVERKAR M
         where M.NR = S.NR) = ('John Lennon');
```

och många variationer. – **2p.**

(c) Ange alla kombinationer (utan dubletter) av artister och skivtitlar där artisten medverkar i en skiva som producerades i ett annat land än det där artisten föddes.

Svar:

```
select distinct A.NAMN, S.TITEL
  from SKIVA S, MEDVERKAR M, ARTIST A
 where M.NR = S.NR and M.ARTIST = A.NAMN
   and A.LAND <> S.LAND;
```

– **2p.** Obs – inga samkörningsvillkor får glömmas!

(d) Ange sammanlagt antal klassiska skivor sålda i september 1996 samt den sammanlagda summan pengar som betalades för dessa.

Svar:

```
select SUM(ANTAL), SUM(ANTAL*PRIS)
  from SKIVA SK, SÅLDA S
 where SK.NR = S.NR and SK.KATEGORI = 'klassisk'
   and S.ÅR = 1996 and S.MÅNAD = 'september';
```

– **2p.**

(e) Ange den kategori (eller de kategorier) som har högsta genomsnittliga pris.

Svar: För denna uppgift behöver vi en vy som mellanresultat.

```
create view KAT_PRIS(KAT,MEDELPRIS) as
select KATEGORI, avg(PRIS)
  from SKIVA
group by KATEGORI;

select KAT
  from KAT_PRIS
 where MEDELPRIS = (select max(MEDELPRIS)
                      from KAT_PRIS);
```

– **2p.**

(f) Sänk priset på alla hårdrockskivor med 10%.

Svar:

```
update SKIVA
   set PRIS = 0.9 * PRIS
 where KATEGORI = 'hårdrock';
```

– **2p.**

12 poäng.

Uppgift 5. 9p Betrakta relationen *sjukvård(Patient, Sjukhus, Läkare, Kommun)* med funktionella beroenden

1. *Läkare* → *Sjukhus*
2. *Patient, Sjukhus* → *Läkare*
3. *Sjukhus* → *Kommun*

Beroendena skall alltså uttrycka att:

1. Varje *Läkare* arbetar på **ett** visst *Sjukhus*.
2. Varje *Patient* som kommer till ett *Sjukhus* blir alltid behandlad av samma *Läkare* där.
3. Varje *Sjukhus* finns i **en** viss *Kommun*.

En *Patient* kan alltså behandlas på olika *Sjukhus* (givetvis av olika *Läkare*).

(a) Bestäm alla nycklar till relationen *sjukvård*.

Svar: $\{P, S\}$ och $\{P, L\}$. – **2p.**

(b) Uppfyller *sjukvård* Boyce-Codd-normalformen? Motivera!

Svar: Nej. Beroendet 1. $L \to S$ bryter mot BCNF, eftersom L inte är någon (över)nyckel; och beroendet 3. $S \to K$ bryter mot BCNF, eftersom S inte är någon (över)nyckel. – **1p.**

(c) Uppfyller *sjukvård* tredje normalformen? Motivera!

Svar: Nej. Beroendet 1. $L \to S$ uppfyller 3NF, eftersom S är ett nyckelattribut (ingår i nyckeln PS), men beroendet 3. $S \to K$ bryter även mot 3NF, eftersom K inte är nåt nyckelattribut.
(Beroendet 2. $PS \to L$ uppfyller 3NF eftersom det redan uppfyller det starkare BCNF.) – **1p.**

(d) Om vi delar upp relationen *sjukvård* för att uppfylla en bättre (d.v.s. strängare) normalform, vilka delrelationer får vi? Vad förlorar vi? Vilka felaktiga data är möjliga att mata in i tabellen?

Svar: Först: Vi delar upp med beroendet 3. $S \to K$ för att få åtminstone 3NF, och vi får relationer $psl(\underline{P, S}, \underline{L})$ och $sk(\underline{S}, K)$ med referens $psl.S \to sk.S$. Vi förlorar inget. **Sedan:** Vi delar upp psl med beroendet 1. $L \to S$ för att få BCNF, och vi får relationer $ls(\underline{L}, S$ och $pl(\underline{P, L})$. Vi förlorar beroendet 2. $PS \to L$. Felaktiga data kan matas in som bryter mot beroendet 2., d.v.s. samma patient som har två olika läkare vid samma sjukhus. – **4p.**

(e) Om vi **inte** delar upp relationen *sjukvård*, vilka felaktiga data är då möjliga att mata in i tabellen?

Svar: Felaktiga data kan matas in som bryter mot beroendet 1. $L \to S$, d.v.s. samma läkare tilldelas olika sjukhus; eller som bryter mot beroendet 3. $S \to K$, d.v.s. samma sjukhus tilldelas olika kommuner.
– **2p.**

(f) Vilka andra problem kan uppstå med relationen *sjukvård* om vi inte delar upp?

Svar:

- Borttagningsanomalier: Om sista patient eller läkare till ett sjukhus borttages, så försvinner informationen om vilken kommun det tillhör. (Inte heller kan sådan information lagras om ett nytt sjukhus som ännu inte fått patienter/läkare.)
- Uppdateringsanomalier: Ändras kommunen till ett sjukhus, så ändras den inte automatiskt på alla ställen → bryter mot 3. $S \to K$.

– **1p.**

11 poäng.

Uppgift 6. 8p Ett databassystem har kraschat. Vid nystart av systemet måste innehållet i databasen återställas till ett konsistent tillstånd. För detta används loggfilen, vilken har följande innehåll:

```
BEGIN_TRANSACTION(T2)
WRITE(T2,A,80,50)
BEGIN_TRANSACTION(T3)
COMMIT_TRANSACTION(T2)
CHECKPOINT
WRITE(T3,B,40,30)
BEGIN_TRANSACTION(T1)
WRITE(T1,A,50,100)
WRITE(T3,A,100,110)
WRITE(T3,B,30,75)
WRITE(T1,A,110,150)
COMMIT_TRANSACTION(T1)
WRITE(T3,A,150,120)
```

där `WRITE(T2,A,80,50)` betyder att transaktionen T2 har ändrat värdet på dataobjektet A från 80 till 50.

Beskriv vad som händer vid återställningen efter kraschen

(a) om **fördröjd ändring** *(deferred update)* används,

(b) om **omedelbar ändring** *(immediate update)* används.

Hur ser det återställda databastillståndet ut? Vad händer sen?

Svar: Alla ändringar som gjordes innan `CHECKPOINT`, alltså `WRITE(T2,A,80,50)`, är inskrivna i databasen. T1 hann avsluta, `COMMIT_TRANSACTION`, innan kraschen, alltså måste T1:s ändringar, `WRITE(T1,A,50,100)`, `WRITE(T1,A,110,150)`, gälla. T3 hann inte avsluta, alltå får T3:s ändringar, `WRITE(T3,B,40,30)`, `WRITE(T3,A,100,110)`, `WRITE(T3,B,30,75)`, `WRITE(T3,A,150,120)` inte lämna några spår i databasen.

(a) Vid **fördröjd ändring** har T1:s och T3:s ändringar inte skrivits in i databasen ännu. T1:s ändring måste göras om, `REDO WRITE(T1,A,50,100)`, `REDO WRITE(T1,A,110,150)`, men T3:s kan vi strunta i.

(b) Vid **omedelbar ändring** har T1:s och T3:s ändringar redan skrivits in i databasen. Först kollar man loggfilen **baklänges** för alla ändringar som gjorts av misslyckade (avbrutna) transaktioner och gör dem ogjorda: `UNDO WRITE(T3,A,150,120)`, `UNDO WRITE(T3,B,30,75)`, `UNDO WRITE(T3,A,100,110)`, `UNDO WRITE(T3,B,40,30)`. På så sätt återställs B till dess ursprungliga värde 40 och A till 100. **Sen** kollar man loggfilen **framlänges** för de lyckade transaktionernas alla ändringar och gör om dem: `REDO WRITE(T1,A,50,100)`, `REDO WRITE(T1,A,110,150)`.

I båda två fall ska resultatet alltså se ut så: $A = 150$, $B = 40$.

Sen får det inte glömmas att den avbrutna transaktionen T3 ska startas på nytt.

8 poäng.