

Omtentamen i DATABASER

Svar:

Obs! Lärarversion, med lösningar

DAG: lördag 28/4, 2001 **TID:** kl. 8.45 – 12.45 **PLATS:** M-huset

Ansvarig: Patrik Jansson & Dennis Björklund
Förfrågningar: Dennis Björklund, ankn. 5402
Resultat: anslås den 17 maj 2001
Poängantal: sammanlagt maximalt 60 poäng.
Betygsgränser: Chalmers: 3:a 24 p., 4:a 36 p., 5:a 48 p.
GU: Godkänd 28 p., Väl godkänd 48 p.
Hjälpmedel: inga

Observera:

- Skriv tydligt och disponera pappret på ett lämpligt sätt.
- Börja varje uppgift på nytt blad. Skriv endast på en sida av pappret.
- Alla svar skall **motiveras väl** och ej vara onödigt komplicerade.
- Ange på tentan om du går på GU eller Chalmers. För Chalmers ange även vilken linje du går.

Lycka till!

Uppgift 1. a) Vad är och var används fragmentering? Vad är horisontell och vertikal fragmentering?

4*3p

Svar: Fragmentering innebär att tabellerna i en databas delas upp på olika maskiner/noder/serverar i ett nät. Fragmentering används i distribuerade databaser.

Vid h. f. delas en tabell i grupper av rader (σ) och man återfår den med union (\cup).

Vid v. f. delas en tabell i grupper av kolumner (π) och återfås med samkörning (\bowtie).

b) Vad är **falska tuppler** (*spurious tuples*) och hur uppstår de? Vilket enkelt kriterium kan garantera att inga falska tuppler uppstår?

Svar: När man delar upp en relation R med projektion i delrelationer $R_X = \pi_X(R)$, $R_Y = \pi_Y(R)$ och sen återförenar delarna med samkörning (där alltså $X \cup Y =$ alla R :s attribut), då kan det hända att samkörningen $R_X \bowtie R_Y$ innehåller tuppler som **inte** fanns med i den ursprungliga R . Dessa kallas för **falska tuppler**.

Vi får garanterat inga falska tuppler i fallet att samkörningsattributet/en $X \cap Y$ på $R_X \bowtie R_Y$ är nyckel till R_X eller till R_Y (eller båda).

c) Normalformernas ordning. Antag att relationen *fyra* uppfyller 4NF, *tre* uppfyller 3NF och *boyce* uppfyller BCNF samt att ingen av relationerna uppfyller någon högre normalform än den angivna. Ange för var och en relationerna ovan vilken eller vilka andra normalformer bland BCNF, 1NF, 4NF, 3NF som den också uppfyller.

	relation	NF enl. uppgift	övriga normalformer
Svar:	<i>tre</i>	3NF	1NF
	<i>boyce</i>	BCNF	3NF, 1NF
	<i>fyra</i>	4NF	BCNF, 3NF, 1NF

d) Beskriv vad som händer när man lägger till ett element i ett B^+ -träd där varje nod och varje löv är fullt.

Svar:

(a) Lokalisera lövet där det nya elementet skall skjutas in.

(b) Dela detta löv i två delar och skjut in det nya elementet på sin plats.

(c) En pekare till det nya lövet infogas i föräldernoden som då i sin tur delas i två noder.

(d) Detta noddelande försätter uppåt till trädets rot.

(e) Roten delas och bli två noder.

(f) En ny rot skapas med endast två pekare till de två halvorna av föregående rot.

Uppgift 2. Betygsstatistik

4*4p

För att hålla reda på tentaresultaten på databaskursen använder läraren Ess Kuell följande databastabeller:

- *utbildning*(Linje, *Universitet*) där *Universitet* måste vara GU eller Chalmers.
- *elev*(PNr, *ENamn*, *FNamn*, *Linje*) där varje *Linje* måste finnas med i tabellen *utbildning*.
- *tenta*(PNr, TentaNr, *Resultat*) där *TentaNr* är 0 för ordinarie tentan och 1 för omtentan, *Resultat* är antalet poäng på tentan (måste vara mellan 0 och 60), och varje personnummer måste förekomma i tabellen *elev*.

I dina svar får du gärna förkorta attributnamnen till en bokstav.

a) Definiera de tre tabellerna i SQL inklusive nycklar, främmande nycklar och de begränsningar som anges ovan.

Svar:

```

create table Utbildning
( Linje          varchar(3)  -- Utbildningslinjens kortnamn
, Universitet   varchar(10)
, constraint    Utb_nyckel  primary key (Linje)
, constraint    Univkorrekt check (Universitet in ('GU', 'Chalmers'))
);

```

```

create table Elev
( PNr           char(10)    -- Personnummer
, FNamn        varchar(20)  -- Förnamn
, ENamn        varchar(30)  -- Efternamn
, Linje        varchar(3)   -- Linjekod
, constraint    Elevnyckel  primary key (PNr)
, constraint    Linjefinns  foreign key (Linje) references Utbildning(Linje)
);

```

```

create table Tenta
( PNr           char(10)    -- Personnummer
, TentaNr      numeric(1)   -- 0 = ordinare, 1 = omtenta
, Resultat     numeric(2)   -- Antal poäng på tentan
, constraint    Tentanyckel  primary key (PNr,TentaNr)
, constraint    KorrektTnr   check (TentaNr between 0 and 1)
, constraint    Korrektpoäng check (Resultat between 0 and 60)
, constraint    Elevfinns    foreign key (PNr) references Elev(PNr)
);

```

- b) Hjälp herr Kuell generera resultatlistor genom att definiera en vy *lista(Universitet, PNr, ENamn, FNamn, TentaNr, Resultat, Betyg)* där *Betyg* för varje tenta beräknas som för denna tenta. Tänk på att betygsgränserna beror på om eleven tillhör GU eller Chalmers. Tips: För beräkningen av betyget kan det vara bra att använda SQLs **case**-uttryck med syntax som i detta exempel:

```

select  Linje, -- Lista linjerna med långt namn för universitetet
        case
          when Universitet = 'GU' then 'Göteborgs universitet'
          when Universitet = 'Chalmers' then 'Chalmers tekniska högskola'
        end
        as Långt_Universitetsnamn
from    utbildning;

```

Svar:

```

create view Lista
as select Utbildning.Universitet
        , Elev.PNr
        , Elev.ENamn
        , Elev.FNamn
        , Tenta.TentaNr
        , Tenta.Resultat
        , case
          when Utbildning.Universitet='Chalmers' then
            case
              when Tenta.Resultat < 24 then 'U'

```

```

        when Tenta.Resultat < 36 then '3'
        when Tenta.Resultat < 48 then '4'
        when Tenta.Resultat <=60 then '5'
    end
    when Utbildning.Universitet='GU' then
    case
        when Tenta.Resultat < 28 then 'U'
        when Tenta.Resultat < 48 then 'G'
        when Tenta.Resultat <=60 then 'VG'
    end
    end
    Betyg
from Tenta, Elev, Utbildning
where Tenta.PNr = Elev.PNr
    and Elev.Linje = Utbildning.Linje;

```

- c) Definiera en vy *stat* som kan ge betygsstatistik för den ordinarie tentan på denna form:

```

UNIVERSITET BETYG ANTAL
-----
Chalmers    3         6
Chalmers    4         8
Chalmers    5         2
Chalmers    U         2
GU          G         4
GU          U         3
GU          VG        2

```

Svar:

```

create view stat as
    select Universitet, Betyg, count(PNr) Antal
    from Lista
    where TentaNr = 0
    group by Universitet, Betyg;

```

- d) Definiera en vy som ger personnummer och den bättre tentans betyg för de Chalmers elever som gjort bättre ifrån sig på omtentan än på den ordinarie tentan. Även de elever som inte gick upp på den ordinarie tentan, men som klarade omtentan skall finnas med.

Svar:

```

create view bättre as
    (select L1.PNr
        , L2.Betyg
    from Lista L1, Lista L2
    where L1.PNr = L2.PNr
        and L1.TentaNr = 0
        and L1.Universitet = 'Chalmers'
        and L2.TentaNr = 1
        and ( (L1.Betyg = 'U' and L2.Betyg in ('3','4','5'))
            or (L1.Betyg = '3' and L2.Betyg in ('4','5'))
            or (L1.Betyg = '4' and L2.Betyg in ('5'))))

```

```

union
(select PNr
    , Betyg
    from Lista
 where TentaNr = 1
    and Universitet = 'Chalmers'
    and Betyg <> 'U'
    and PNr not in (select PNr from Lista where TentaNr = '0'));

```

Uppgift 3. Givet är tabellerna $R(\underline{A}, B)$, $S(\underline{B}, \underline{C}, A)$ och $T(\underline{C}, D)$ med följande innehåll:

7p

R:

<u>A</u>	B
a ₁	b ₂
a ₂	b ₃

S:

<u>B</u>	<u>C</u>	A
b ₃	c ₃	a ₂
b ₁	c ₄	a ₁
b ₃	c ₄	a ₂

T:

<u>C</u>	D
c ₂	d ₃
c ₃	d ₁

Beräkna resultatet på följande relationsalgebra-uttryck. I denna uppgift antar vi att det är NAMN som gäller när vi identifierar kolumnerna. **Obs!** Beakta att det alltid är **mängder** det handlar om!

- $\sigma_{C=c_4}(S)$
- $\pi_{AB}(S) - R$
- $R \times \pi_C(S)$

Svar: $\sigma_{C=c_4}(S)$

B	C	A
b ₁	c ₄	a ₁
b ₃	c ₄	a ₂

$\pi_{AB}(S) - R$

A	B
a ₁	b ₁

$R \times \pi_C(S)$

A	B	C
a ₁	b ₂	c ₃
a ₁	b ₂	c ₄
a ₂	b ₃	c ₃
a ₂	b ₃	c ₄

– (a) **1p.** – (b) **2p.** – (c) **2p.**

- $S \bowtie_{S.C=T.C} T$ (right outer join)

Svar: $S \bowtie_{S.C=T.C} T$

B	S.C	A	T.C	D
b ₃	c ₃	a ₂	c ₃	d ₁
–	–	–	c ₂	d ₃

eller

B	C	A	D
b ₃	c ₃	a ₂	d ₁
–	c ₂	–	d ₃

2p. Med “–” menas *NULL*. Kolumner får vara i annan ordning.

Uppgift 4. Tre transaktioner T_1 , T_2 och T_3 skall utföra följande operationer

5p+3p

T_1 : Läs A, skriv värdet A-1 till B och skriv värdet A+1 till C

T_2 : Läs B, skriv värdet B+10 till C

T_3 : Läs C, skriv värdet C+20 till A

- Ange för dessa transaktioner en operationsföljd som med tvåfasläsning leder till deadlock. Rita serialiserbarhetsgraf för din operationsföljd.

Svar: Denna ger deadlock:

T_1 T_2 T_3 T_3 T_2 T_1 T_1
read A **read B** **read C** **write A** **write C** **write B** **write C**

Graf: Tre noder (T_1 , T_2 , T_3) och sex bågar: $T_i \leftrightarrow T_j$ för alla $i \neq j$.

- b) Antag att startvärdena för A, B och C är 0. Ange två serialiserbara operationsföljder som ger olika slutvärden på A.

Svar: Exempel: två seriella (och därmed trivialt serialiserbara) följder:

T1 T2 T3 → A=29

T1 T3 T2 → A=21

Uppgift 5. Musikälskaren N. Apster har en musiksamling som han vill hålla reda på med hjälp av en databas. Ditt jobb blir att hjälpa till med designen genom att:
10p

- a) Tillverka ett ER-diagram. (Du får lägga till nya attribut för nycklar i de fall det behövs.) Om du gör antaganden utöver de som beskrivs nedan skall dessa redovisas i svaret.
b) Översätta ER-diagrammet till relationer där nycklar och främmande nycklar skall anges (behöver inte vara SQL).

Följande saker säger N. Apster om sin databas: (forts. på nästa sida)

- En artist har ett namn bestående av förnamn och efternamn.
- En skiva har en titel.
- På en skiva så finns ett antal låtar som hittas med hjälp av ett spårnummer (skivans första låt är på spår nummer 1 osv.).
- Låtar har en titel och en längd, samma låttitel kan förekomma på olika skivor.
- Skivans längd är summan av de ingående låtarnas längd. (Vid översättning till relationer blir detta en vy.)
- Varje låt har ett antal medverkande artister, för varje medverkande så sparas vilken funktion han/hon hade på den låten (t.ex. sångare, trumslagare, kaffehämtare).
- Varje skiva släpps av precis en grupp.
- En grupp har ett namn och består av en eller flera artister. (Så en ensam artist som släpper en skiva betraktas som en enmansgrupp.)
- Artister kan givetvis vara med i flera grupper.

Och för dem som undrar så kan vi meddela att kaffehämtare här räknas som artister. Det är mest en fråga om hur man hämtar kaffe för att få räknas som artist.

Uppgift 6. Antag att vi har en relation $R(A, B, C, D, E, F)$ och beroenden $A \rightarrow B$, $B \rightarrow C$, $DE \rightarrow F$ och $F \rightarrow D$. För varje uppdelning nedan markera vilka av följande alternativ som gäller: $3NF$, $BCNF$, Förlustfri (FF) och Beroendebevarande (BB). Markera med kryss i svarsformuläret nedan och lämna in hela denna sida med tentan.
7p

- a) $R(ABC) R(DEF)$
b) $R(AB) R(BC) R(DEF)$
c) $R(ABC) R(DEF) R(AEF)$
d) $R(AB) R(BC) R(ADE) R(DEF)$
e) $R(AB) R(BC) R(DF) R(EF)$
f) $R(DF) R(AB) R(AEF) R(BC) R(EF)$
g) $R(ABCDEFG)$

Svarsformulär för uppgift 6

Namn:

Personnummer:

Löpande sidnummer:

	BB	FF	3NF	BCNF
a)	X			
b)	X		X	
c)	X	X		
d)	X	X	X	
e)			X	X
f)		X	X	X
g)	X	X		

Svar: Svaret finns ovan, här kommer en detaljerad motivering.

Beroendebevarande: följ alla beroenden genom deluppgifterna:

$A \rightarrow B$ bevaras i alla (det finns alltid ett schema R med $AB \in R$)

$B \rightarrow C$ bevaras i alla (samma för BC)

$F \rightarrow D$ bevaras i alla (samma för FD)

$DE \rightarrow F$ finns ej med i något schema i deluppgift e och f och kan ej härledas från de övriga.

Alltså är alla uppdelningar utom e, f beroendebevarande.

För BCNF och 3NF gäller det att kontrollera alla scheman, deras beroenden och deras nycklar.

För att ta fram nycklarna använder vi hölje av attributmängd:

För den universella relationen:

$ADE^+ = ABCDEF$ kandidatnyckel

$AEF^+ = ABCDEF$ kandidatnyckel

Nycklar i vissa delar: ($A^+ = ABC$, $B^+ = BC$, $DE^+ = DEF$, $EF^+ = DEF$, $F^+ = DF$)

$R(ABCDEF)$: nycklar ADE och AEF , f.b. alla enligt uppgiften

$R(ABC)$: nyckel A , f.b. $A \rightarrow B$, $B \rightarrow C$

$R(DEF)$: nycklar DE och EF , f.b. $DE \rightarrow F$, $F \rightarrow D$

$R(ADE)$: nyckel ADE , inga f.b.

$R(AEF)$: nyckel AEF , inga f.b.

$R(EF)$: nyckel EF , inga f.b.

$R(AB)$: nyckel A , f.b. $A \rightarrow B$

$R(BC)$: nyckel B , f.b. $B \rightarrow C$

$R(DF)$: nyckel F , f.b. $F \rightarrow D$

Varje schema utan (icke-triviala) f.b. (här $R(ADE)$, $R(AEF)$ och $R(EF)$) är automatiskt på BCNF och 3NF eftersom det inte finns några beroenden som kan bryta mot dem.

Varje två-attribut-schema (t.ex. $R(AB)$, $R(BC)$ och $R(DF)$) uppfyller automatiskt BCNF (testa gärna). Kvar att undersöka är $R(ABC)$, $R(ABCDEF)$ och $R(DEF)$:

$R(ABC)$: B är ej övernyckel \Rightarrow ej BCNF. C är ej nyckelattribut \Rightarrow ej 3NF.

$R(ABCDEF)$ Samma här.

$R(DEF)$ Klassiskt exempel på "3NF men ej BCNF": F är inte en övernyckel, men D är ett nyckelattribut.

Sammanfattningsvis:

ingen $R(ABC)$, $R(ABCDEF)$

3NF $R(DEF)$

3NF+BCNF $R(AB)$, $R(BC)$, $R(DF)$, $R(EF)$, $R(ADE)$, $R(AEF)$

Och om vi kombinerar ihop dem till uppgiftens variationer:

3NF a) $R(ABC)$ $R(DEF)$

3NF b) $R(AB)$ $R(BC)$ $R(DEF)$

3NF c) $R(ABC)$ $R(DEF)$ $R(AEF)$

3NF d) $R(AB)$ $R(BC)$ $R(ADE)$ $R(DEF)$

3NF BCNF e) $R(AB)$ $R(BC)$ $R(DF)$ $R(EF)$

3NF BCNF f) $R(DF)$ $R(AB)$ $R(AEF)$ $R(BC)$ $R(EF)$

g) $R(ABCDEF)$

Förlustfri uppdelning innebär att samkörningen inte ger några falska tuppler. Om bara scheman är givna skall detta gälla för alla data som uppfyller de funktionella beroendena i den universella tabellen.

- (a) här finns inget gemensamt attribut, så samkörning blir samma som kartesisk produkt. (Ger helt klart för många tuppler.)
- (b) Samma här: $R(DEF)$ är helt skild från de övriga.
- (c) Samkörning av $R(DEF)$ och $R(AEF)$ på nyckeln EF i DEF garanterar förlustfrihet. Samkörning av $R(ABC)$ med resultatet på nyckeln A i $R(ABC)$ ger också förlustfrihet.
- (d) P.s.s samkör $R(AB)$ med $R(BC)$ på nyckeln B i $R(BC)$, resultatet, dvs. $R(ABC)$, samkörs med $R(ADE)$ på nyckeln A i $R(ABC)$. Slutligen samkörning med $R(DEF)$ på nyckeln DE i $R(DEF)$.
- (e) Samma problem som i (b).
- (f) Samma stil som i (d): $R(DF)$ samkörs med $R(EF)$ på nyckeln F i $R(DF)$ osv.
- (g) Trivialt förlustfri.

Appendix: Några kommentarer om syntax

SQL-frågor

```
select attributlista from relationslista where villkor  
group by attributlista having villkor
```

- *select attributlista*: Anger de attribut som den resulterande relationen skall innehålla. Kan också vara *grupperingsfunktioner* eller **case**-uttryck.
- *from relationslista*: Anger de relationer som skall ingå i frågan.
- *where villkor*: Kan t.ex. vara:
 - *Jämförelser*: T.ex. “R.A < S.B” eller “R.C = 'hej'”
 - *Mängdoperationer*: T.ex. “X in (select ... from ...)”
 - *Logiska operationer*: T.ex. “X=12 and Y<'hej' and ...”

where kan utelämnas.

- *group by attributlista*: Skapar grupper baserade på de ingående attributen. *group by* kan utelämnas.
- *having villkor*: Används tillsammans med *group by*. Filtrerar ut vissa grupper. *having* kan utelämnas. Får endast användas tillsammans med *group by*.
- *Grupperingsfunktioner*: *min(...)*, *max(...)*, *count(...)*, *avg(...)*
- I *select* och *grupperingsfunktioner* kan *distinct* användas.

Skapa vyer

```
create view relation(attributlista) as sql-fråga
```

Attributlista kan ofta utelämnas.

Insättning, borttagning, uppdatering, etc.

```
insert into relation values ( värde, värde, ...)
```

```
delete from relation where villkor
```

where-klausulen kan utelämnas

```
update relation set nya värden where villkor
```

where-klausulen kan utelämnas

```
drop komponent relation
```

komponent någon av *table*, *view*, etc.

Rättigheter

```
grant rättighet on relation to användare
```

```
revoke rättighet on relation from användare
```

rättighet: all, select, delete, insert, update

insert och update kan förses med en attributlista.